

Guia de preparação MLA-C02

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	**72%** 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 MLA-C02: 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 72%).

Mapa do exame

Cada prova do curso tem 65 questões e 170 minutos, no formato do exame. Referência de aprovação: 72%.

Domínio (capítulo)	Peso	Neste guia
1. Desenvolvimento de modelos de ML e de modelos de base (FM)	24%	18 conceitos · 14 pegadinhas
2. Implantação e orquestração de fluxos de trabalho de ML e IA	24%	18 conceitos · 14 pegadinhas
3. Operação, monitoramento e segurança de soluções de ML e IA	24%	18 conceitos · 14 pegadinhas
4. Preparação de dados para ML e IA	28%	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 · Desenvolvimento de modelos de ML e de modelos de base (FM)

Um engenheiro de ML executa um script de treinamento em modo script com o estimador TensorFlow do SageMaker AI e define hiperparâmetros como epochs e batch_size no estimador. O job roda, mas o script continua usando os valores padrão escritos no código. Qual ajuste deve ser feito no script?

A. Ler os hiperparâmetros como argumentos de linha de comando com argparse
B. Ler os hiperparâmetros de uma tabela do Amazon DynamoDB
C. Gravar os hiperparâmetros dentro do arquivo model.tar.gz
D. Definir os hiperparâmetros como tags de recurso do job de treinamento no console

D2 · Desenvolvimento de modelos de ML e de modelos de base (FM)

Um banco treina uma rede neural para detectar lavagem de dinheiro. As curvas mostram a perda de treino oscilando fortemente de uma época para outra, às vezes subindo muito, sem convergir. Os dados foram validados e estão normalizados. Qual ajuste é o primeiro a testar?

A. Reduzir a taxa de aprendizado ou usar um agendamento que a diminua
B. Aumentar o número de épocas mantendo os hiperparâmetros atuais
C. Aumentar a taxa de aprendizado para escapar de mínimos locais
D. Remover a normalização dos dados para ampliar os gradientes

D3 · Desenvolvimento de modelos de ML e de modelos de base (FM)

Um job de treinamento com o algoritmo integrado XGBoost do SageMaker AI usa um arquivo CSV de pedidos com cabeçalho na primeira linha e a coluna alvo na última posição. O modelo gerado apresenta resultados sem sentido. Qual correção deve ser feita nos dados de entrada?

A. Normalizar as colunas numéricas entre zero e um
B. Aumentar o valor do hiperparâmetro num_round
C. Colocar a coluna alvo na primeira posição e remover a linha de cabeçalho
D. Converter o arquivo para JSON Lines mantendo o cabeçalho e a coluna alvo no fim

D4 · Implantação e orquestração de fluxos de trabalho de ML e IA

Uma rede de supermercados usa Batch Transform do SageMaker AI para pontuar um arquivo CSV de clientes. A primeira coluna é o ID do cliente, que não deve ser enviado ao modelo, mas cada previsão precisa sair associada ao ID correspondente no arquivo de saída. Qual configuração atende a esse requisito?

A. Ordenar a saída por ordem alfabética das previsões
B. Executar o job com SplitType=None e um único registro
C. Treinar o modelo novamente incluindo o ID do cliente como feature
D. Usar InputFilter para tirar o ID e JoinSource=Input na saída

D5 · Implantação e orquestração de fluxos de trabalho de ML e IA

Uma edtech hospeda um LLM em um endpoint do SageMaker AI com inference components. O tráfego varia muito e há madrugadas sem nenhuma requisição. A empresa quer escalar as cópias do modelo conforme a demanda e não pagar instâncias com GPU quando não há uso. Qual configuração atende?

A. Configurar o auto scaling pela CPUUtilization das instâncias e manter uma instância ligada a noite toda
B. Definir zero cópias como mínimo no inference component, com target tracking e step scaling
C. Migrar o modelo para um endpoint serverless com GPU e concorrência provisionada nos horários de pico
D. Criar uma programação fixa que dobra as instâncias a cada hora útil e as mantém ligadas de madrugada

D6 · Implantação e orquestração de fluxos de trabalho de ML e IA

Uma rede de farmácias usa o mesmo pipeline do SageMaker Pipelines nas contas de desenvolvimento e de produção. Em desenvolvimento, o treino deve usar instâncias pequenas e uma amostra dos dados; em produção, instâncias maiores e o conjunto completo. Como configurar o pipeline sem manter duas definições?

A. Usar o mesmo tipo de instância grande nas duas contas
B. Duplicar a definição do pipeline e alterar os valores fixos de instância e dados em cada conta
C. Definir parâmetros do pipeline para tipo de instância e caminho dos dados
D. Alterar o código do pipeline antes de cada execução

D7 · Operação, monitoramento e segurança de soluções de ML e IA

Uma universidade armazena 200 milhões de vetores de 1.024 dimensões gerados pelo Amazon Titan Text Embeddings V2 em um domínio do Amazon OpenSearch Service, e o custo de memória cresceu. Testes mostraram que vetores de 256 dimensões mantêm precisão aceitável na busca. O que fazer?

A. Aumentar o número de réplicas do índice para distribuir os vetores entre mais nós
B. Regerar os embeddings com 256 dimensões no Titan Text Embeddings V2 e reindexar o acervo
C. Trocar para um modelo de 3.072 dimensões para compensar a perda de precisão esperada na busca
D. Duplicar os vetores em um segundo índice para acelerar as consultas mais frequentes

D8 · Operação, monitoramento e segurança de soluções de ML e IA

Uma rede de hospitais quer que analistas de dados consultem, pelo Amazon Athena, tabelas do AWS Glue Data Catalog sem ver as colunas com dados pessoais de pacientes, mantendo uma única cópia dos dados. Qual controle atende a esse requisito?

A. Conceder s3:GetObject aos analistas nos prefixos das tabelas
B. Conceder permissões em nível de coluna com o AWS Lake Formation
C. Criar uma cópia das tabelas sem essas colunas para cada analista
D. Criptografar o bucket com SSE-S3 e liberar a leitura aos analistas

D9 · Operação, monitoramento e segurança de soluções de ML e IA

Antes da inferência em lote diária de um modelo de demanda de energia de uma companhia estadual, a equipe quer detectar automaticamente mudanças estatísticas nos lotes de entrada, como queda brusca de volume ou mudança na média de consumo, sem definir limites fixos à mão. O que usar?

A. Usar o AWS Config para registrar mudanças de esquema nas tabelas de entrada
B. Usar uma regra DQDL fixa de contagem de linhas definida no primeiro dia de operação
C. Usar a detecção de anomalias do AWS Glue Data Quality com analisadores nos lotes
D. Usar o Amazon Macie para identificar mudanças estatísticas nos arquivos dos lotes

D10 · Preparação de dados para ML e IA

Uma empresa de logística tem centenas de linhas de código PySpark, com bibliotecas nativas específicas e ajustes finos de configuração do Spark, para processar 50 TB de registros de entregas. A equipe quer migrar esse processamento para a AWS mantendo controle sobre versões do Spark, tipos de instância e configurações do cluster. Qual opção é mais adequada?

A. Reescrever a lógica como receitas do AWS Glue DataBrew
B. Executar os scripts no Amazon EMR com cluster configurado pela equipe
C. Executar os scripts em funções AWS Lambda acionadas a cada novo arquivo no S3
D. Converter o processamento em consultas SQL do Amazon Athena

D11 · Preparação de dados para ML e IA

Uma rede de supermercados precisa rotular 500 mil imagens de prateleiras para treinar um modelo de classificação de produtos. A empresa quer reduzir o custo e o tempo de rotulagem, mas cada rótulo deve ser conferido por uma pessoa antes de entrar no conjunto de treinamento. Qual abordagem atende a esse requisito?

A. Criar receitas do AWS Glue DataBrew que atribuam a classe de produto a cada imagem
B. Usar os rótulos genéricos do Rekognition DetectLabels direto no treino, sem revisão humana
C. Gerar pré-rótulos das imagens com um FM multimodal do Amazon Bedrock e enviá-los à verificação humana
D. Treinar o Amazon Rekognition Custom Labels com as imagens ainda não rotuladas pela equipe

D12 · Preparação de dados para ML e IA

Um modelo de regressão linear de uma rede de supermercados usa a forma de pagamento (dinheiro, débito, crédito, PIX e vale-alimentação) como feature. A equipe codificou as categorias como 1, 2, 3, 4 e 5, e o modelo passou a tratar o vale-alimentação como cinco vezes o dinheiro. Qual codificação é mais adequada?

A. Aplicar transformação logarítmica aos códigos numéricos
B. Aplicar codificação ordinal, ordenando as categorias pelo faturamento médio
C. Aplicar normalização min-max aos códigos de 1 a 5
D. Aplicar one-hot encoding, com uma coluna binária por forma de pagamento

01

CAPÍTULO 1 · 24% DA PROVA

Desenvolvimento de modelos de ML e de modelos de base (FM)

Aqui eu preciso escolher a abordagem certa para cada problema: serviço de IA pronto, algoritmo integrado do SageMaker AI, script próprio em modo script ou FM do Bedrock, e entre prompt engineering, RAG, fine-tuning, pré-treinamento contínuo e destilação. Depois vem treinar bem (AMT, Spot, paralelismo, precisão mista, warm pools) e diagnosticar curvas de perda (overfitting, underfitting, convergência, esquecimento catastrófico). A terceira parte é avaliar: métricas de classificação e regressão pelo custo do erro, métricas de NLP (BLEU, ROUGE, BERTScore, F1), LLM-as-a-judge, avaliação humana, avaliação de RAG, explicabilidade e reprodutibilidade com MLflow. Vale 24% e cai muito como 'qual técnica atende com menor esforço, custo ou latência'. Onde mais erro: BLEU x ROUGE, paralelismo de dados x de modelo, warm start IDENTICAL x TRANSFER_LEARNING e quando usar pré-treinamento contínuo em vez de fine-tuning.

Neste capítulo

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

Tópicos-chave

Serviços de IA prontos

Quando o problema é comum e não há equipe de ML ou dados rotulados, o serviço gerenciado vence: Rekognition (moderação de conteúdo; Custom Labels para objetos próprios com poucas centenas de fotos rotuladas), Textract (AnalyzeDocument com FORMS, TABLES e QUERIES; AnalyzeExpense para faturas), Comprehend (classificação personalizada, toxicidade), Comprehend Medical (entidades clínicas), Transcribe (vocabulário personalizado; Call Analytics para call center), Translate (terminologia personalizada em CSV, TSV ou TMX) e Personalize (recomendações). Polly faz o caminho inverso: texto para fala.

Algoritmos integrados do SageMaker AI

XGBoost para dados tabulares (em CSV, alvo na primeira coluna e sem cabeçalho); Linear Learner com binary_classifier para regressão logística interpretável; Factorization Machines para dados esparsos e interações cliente-item, como previsão de cliques; DeepAR para muitas séries temporais relacionadas e itens com pouco histórico; Random Cut Forest para pontuação de anomalia sem rótulos; BlazingText supervisionado (rótulo com prefixo __label__) para classificar texto rápido, ou Word2Vec para embeddings.

Modo script

Você entrega o script como entry_point de um estimador de framework (PyTorch, TensorFlow, scikit-learn) e o SageMaker AI o executa em contêiner pré-construído, lendo dados de SM_CHANNEL_* e salvando o modelo em SM_MODEL_DIR. Os hiperparâmetros do estimador chegam ao script como argumentos de linha de comando (leia com argparse) e em /opt/ml/input/config/hyperparameters.json. Biblioteca Python ausente se resolve com requirements.txt no source_dir, sem imagem própria.

Ajuste automático de modelos (AMT)

Estratégias: bayesiana (aprende com os jobs anteriores; paralelismo alto reduz essa vantagem), aleatória, grade (só hiperparâmetros categóricos) e Hyperband (interrompe cedo configurações fracas). Parada antecipada com TrainingJobEarlyStoppingType em Auto corta jobs sem chance; taxa de aprendizado pede escala logarítmica. Em script próprio, a métrica objetiva vem dos logs por regex em MetricDefinitions. Warm start IDENTICAL_DATA_AND_ALGORITHM exige mesmos dados e imagem; TRANSFER_LEARNING aceita dados novos. Managed Spot Training barateia os jobs do ajuste.

Redução do tempo de treino

Precisão mista (FP16 ou BF16 com FP32) acelera GPUs modernas e reduz memória. Paralelismo de dados distribui lotes quando o modelo cabe em uma GPU e o gargalo é o volume; paralelismo de modelo ou dados fragmentados (biblioteca do SageMaker AI ou FSDP) quando parâmetros, gradientes e estados do otimizador não cabem. Warm pools gerenciados (KeepAlivePeriodInSeconds) eliminam o provisionamento entre jobs curtos consecutivos. Na exploração, use amostra estratificada e reserve o conjunto completo para o treino final.

Overfitting e underfitting

Treino ótimo e validação ruim (ou perda de validação subindo enquanto a de treino cai) é overfitting: reduza max_depth, aumente lambda, alpha, gamma ou min_child_weight no XGBoost, use dropout, L2, menos épocas e parada antecipada. Erro alto e parecido em treino e validação é underfitting: mais épocas se as curvas ainda caem, modelo mais expressivo (como gradient boosting), melhores atributos e menos regularização. Remover a validação nunca é solução.

Problemas de convergência

Perda oscilando ou divergindo, principalmente no início, indica taxa de aprendizado alta: reduza, use agendamento, passos de aquecimento (warmup) e gradient clipping. Perda virando NaN se investiga com histogramas de gradientes e pesos no TensorBoard do SageMaker AI. Para não gastar horas com treino parado, publique a perda no CloudWatch via MetricDefinitions e crie alarme que, pelo EventBridge, interrompa o job.

Hiperparâmetros fundamentais

Passos de atualização = (exemplos / batch size) x épocas: 12.000 exemplos, batchSize 8 e epochCount 3 dão 4.500 passos. Falta de memória na GPU no primeiro lote se resolve reduzindo o batch size por GPU com acumulação de gradientes para manter o lote efetivo. Épocas demais em conjunto pequeno levam a overfitting; a taxa de aprendizado não muda o uso de memória.

Combinação de modelos

Bagging (como floresta aleatória com amostras bootstrap) reduz variância de árvores instáveis; boosting reduz viés. Stacking treina um metamodelo nas previsões fora da amostra de modelos que acertam em casos diferentes. Cascata por confiança manda o fácil para um modelo barato e só o incerto para o LLM, reduzindo custo. A pontuação de anomalia do Random Cut Forest pode virar atributo de um XGBoost supervisionado para pegar fraudes novas.

Personalização de FMs

Comece por engenharia de prompt específica da tarefa (instruções, esquema de saída, exemplos few-shot), sobretudo com prazo curto e sem dados rotulados. Fine-tuning supervisionado com pares prompt-resposta em JSONL ensina estilo, formato e tom quando o prompt não basta. Pré-treinamento contínuo usa grandes volumes de texto sem rótulo para aprender vocabulário do domínio, antes do ajuste por tarefa. Destilação usa um modelo professor para ajustar um aluno menor, mantendo qualidade próxima com menos custo e latência e sem rotular à mão.

Esquecimento catastrófico

Ocorre quando o ajuste fino sobrescreve capacidades gerais: o modelo melhora na tarefa nova e passa a errar resumos e perguntas simples. Reduza com PEFT como LoRA (pesos-base congelados), mistura de exemplos gerais (replay), menos épocas e taxa de aprendizado moderada. Prompt de sistema mais longo não recupera capacidades perdidas nos pesos.

Padrões de RAG

Conhecimento que muda com frequência e exige citação da fonte pede RAG com Bedrock Knowledge Bases, não fine-tuning. Dados dinâmicos de sistemas pedem RAG agêntico, com o agente chamando a API como ferramenta. Perguntas de múltiplos saltos entre entidades pedem GraphRAG com Neptune Analytics; perguntas analíticas sobre tabelas do Redshift pedem Knowledge Bases com dados estruturados (texto para SQL). Códigos e siglas exatos pedem busca híbrida; perguntas factuais curtas, chunks menores com sobreposição moderada; idioma diferente, modelo de embeddings multilíngue com reindexação.

Desempenho, latência e custo de FMs

Escolha o menor modelo que atinge a qualidade mínima nas avaliações; modelos maiores tendem a ser mais lentos e caros. Tokens de saída pesam muito: instrua concisão e defina limite baixo de tokens de saída. A API ConverseStream reduz a latência percebida ao entregar tokens à medida que são gerados. Temperatura muda aleatoriedade, não tamanho nem custo.

Métricas de classificação e regressão

A métrica segue o custo do erro: falso negativo caro pede recall (com precisão monitorada); com classe rara, a acurácia esconde o problema. Em regressão, RMSE penaliza mais os erros grandes porque eleva os resíduos ao quadrado, enquanto MAE é mais robusto a extremos. AUC e acurácia são de classificação e não servem para valores contínuos.

Métricas de NLP

BLEU é precisão de n-gramas e padrão de tradução; ROUGE é recall de n-gramas e padrão de sumarização (cobertura da referência). BLEU e ROUGE penalizam sinônimos e paráfrases; BERTScore e similaridade semântica por embeddings capturam equivalência de sentido. Em perguntas e respostas, exact match zera qualquer diferença e F1 sobre palavras dá crédito parcial. Perplexidade não compara com referência.

Avaliações do Amazon Bedrock

Jobs de avaliação comparam FMs com dataset próprio por métricas programáticas, LLM-as-a-judge (pontua correção, completude, utilidade e registra a justificativa de cada nota, aceitando métricas personalizadas com rubrica) ou trabalhadores humanos da própria equipe para critérios que só especialistas julgam. As avaliações de RAG são retrieve-only (context relevance e context coverage, esta exigindo ground truth) ou retrieve-and-generate (correctness, faithfulness, citações).

Explicabilidade

Explicação local de cada decisão em tempo real: valores SHAP calculados no código de inferência, com tree SHAP rápido e exato para XGBoost. Efeito médio de um atributo ao longo de uma faixa (temperatura de 900 a 1.300 graus): gráfico de dependência parcial (PDP). Palavras de um comentário que pesaram na previsão de uma rede neural: gradientes integrados; modelos sem gradiente: Kernel SHAP. Drift de atribuição compara a importância SHAP em produção com a do treino.

Experimentos, sombra e monitoramento

MLflow gerenciado no SageMaker AI registra parâmetros, métricas e artefatos; reproduzir exige também versão do dataset, commit do código e semente aleatória. Prompt Management guarda versões imutáveis de prompt, modelo e parâmetros. Teste de sombra envia cópia do tráfego real à variante sombra sem responder ao cliente e permite promovê-la. Drift de dados: captura no endpoint, baseline do treino e comparação agendada; qualidade do modelo exige ground truth no S3. Humano no circuito: Step Functions com task token aguardando o revisor.

Macetes para memorizar

💡 BLEU x ROUGE

R de ROUGE é de Recall e de Resumo; BLEU traduz com precisão.

💡 Ensembles

Bagging acalma (variância); Boosting corrige (viés); Stacking pergunta a um juiz (metamodelo).

Paralelismo

Cabe na GPU? divide os dados. Não cabe? divide o modelo.

Personalização de FM

Prompt primeiro; par rotulado vira fine-tuning; texto cru vira CPT; custo alto vira destilação; dado que muda vira RAG.

Curvas de perda

Treino cai e validação sobe: overfitting. Os dois altos: underfitting. Oscila e explode: taxa de aprendizado alta.

Warm start

IDENTICAL é idêntico (mesmos dados e imagem); TRANSFER é quando algo mudou.

Explicabilidade

SHAP explica um caso; PDP desenha a curva média; gradientes integrados apontam as palavras.

Tabela de memorização: não confunda

Quando é...	...e quando é
Fine-tuning supervisionado Há pares de prompt e resposta revisados e o objetivo é tarefa, formato ou tom consistentes.	**Pré-treinamento contínuo** Há grande volume de texto do domínio sem rótulos e o objetivo é aprender vocabulário e estilo antes de qualquer ajuste por tarefa.
RAG O conhecimento muda com frequência (taxas, estoque, portarias) e a resposta precisa citar a fonte.	**Fine-tuning** O problema é estilo, formato ou comportamento consistentes, e a engenharia de prompt não bastou.
Bagging O modelo tem alta variância: pequenas mudanças nos dados geram árvores e previsões muito diferentes.	**Boosting** O modelo tem viés alto e precisa de modelos sequenciais que corrijam os erros dos anteriores.
Paralelismo de dados O modelo cabe com folga em uma GPU, mas o volume de dados torna o treino lento.	**Paralelismo de modelo** Parâmetros, gradientes e estados do otimizador não cabem na memória de uma GPU, mesmo com lote 1.
ROUGE Avaliar resumos: quanto do conteúdo da referência foi coberto (recall de n-gramas).	**BLEU** Avaliar tradução: precisão dos n-gramas gerados em relação à referência.
Warm start IDENTICAL_DATA_AND_ALGORITHM O novo ajuste usa os mesmos dados de entrada e a mesma imagem de treinamento.	**Warm start TRANSFER_LEARNING** O conjunto recebeu dados novos ou mudaram faixas ou versões, e ainda se quer reaproveitar o ajuste anterior.

Valores SHAP locais É preciso justificar cada decisão individual, como os atributos que pesaram contra um cliente.	**Gráfico de dependência parcial (PDP)** É preciso ver como a previsão muda, em média, conforme um atributo varia ao longo de uma faixa.

Números que caem

Item	Valor
Peso do domínio no exame	**24%**
Passos de atualização em fine-tuning	**(exemplos / batchSize) x epochCount; 12.000 / 8 x 3 = 4.500**
Formato CSV do XGBoost integrado	**alvo na primeira coluna e nenhuma linha de cabeçalho**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Aumentar a regularização quando treino e validação têm perda alta e ainda caindo.
Como não cair: Isso é underfitting; regularização combate overfitting. Treine por mais épocas ou aumente a capacidade do modelo.

⚠ **Alerta de prova:** Aumentar a taxa de aprendizado para 'atravessar' a região instável quando a perda oscila ou diverge.
Como não cair: Taxa alta é a causa típica. Reduza a taxa, use agendamento, warmup e gradient clipping.

⚠ **Alerta de prova:** Usar paralelismo de modelo para um modelo que cabe em uma GPU, ou paralelismo de dados simples para um modelo que não cabe.
Como não cair: Cabe na GPU e o problema é volume: paralelismo de dados. Não cabe: paralelismo de modelo ou dados fragmentados, pois a cópia completa em cada GPU não cabe.

⚠ **Alerta de prova:** Usar busca em grade para a taxa de aprendizado contínua ou aumentar ao máximo os jobs paralelos na busca bayesiana.
Como não cair: A grade aceita só hiperparâmetros categóricos, e paralelismo alto reduz o aprendizado da bayesiana. Use escala logarítmica e paralelismo moderado.

⚠ **Alerta de prova:** Usar warm start IDENTICAL_DATA_AND_ALGORITHM depois que o conjunto de treino recebeu dados novos.
Como não cair: IDENTICAL exige os mesmos dados e a mesma imagem. Com dados novos, use TRANSFER_LEARNING.

⚠ **Alerta de prova:** Fazer fine-tuning diário ou pré-treinamento contínuo para informações que mudam todo dia.
Como não cair: Ajustar com frequência é caro, lento e não cita fontes. Dado que muda e exige citação é RAG com sincronização da base.

⚠ **Alerta de prova:** Escolher fine-tuning supervisionado com um corpus enorme sem rótulos, ou CPT com pares rotulados.
Como não cair: Rótulo em pares vai para fine-tuning; texto bruto sem rótulo vai para pré-treinamento contínuo.

⚠ **Alerta de prova:** Trocar para um modelo maior para responder mais rápido ou aumentar max_tokens para melhorar qualidade.
Como não cair: Modelos maiores e mais tokens de saída aumentam latência e custo. Use o menor modelo aprovado, limite de saída e streaming.

⚠ **Alerta de prova:** Avaliar resumos com BLEU ou traduções corretas com sinônimos por ROUGE-1 ou exact match.
Como não cair: Resumo é ROUGE; tradução é BLEU; paráfrase correta pede BERTScore ou similaridade semântica.

⚠ **Alerta de prova:** Usar exact match quando respostas corretas com uma palavra a mais recebem zero.
Como não cair: Exact match é binário. O F1 sobre palavras dá crédito parcial pela sobreposição com a referência.

⚠ **Alerta de prova:** Anexar um relatório mensal de importância global dos atributos para justificar cada decisão de crédito.
Como não cair: Explicação global não justifica decisões individuais no momento em que ocorrem. Calcule SHAP local no código de inferência e devolva com a previsão.

⚠ **Alerta de prova:** Monitorar precisão e recall em produção só com a captura de dados ou com o monitor de qualidade de dados.
Como não cair: Precisão e recall exigem os rótulos reais (ground truth) enviados ao S3 e unidos às previsões capturadas.

⚠ **Alerta de prova:** Escolher teste A/B 50/50 quando os clientes não podem receber respostas do modelo novo durante a validação.
Como não cair: No A/B metade dos clientes recebe o modelo novo. Sem afetar ninguém, use teste de sombra e depois promova a variante.

⚠ **Alerta de prova:** Adicionar mais exemplos da tarefa nova, mais épocas e taxa maior para corrigir um modelo que esqueceu tarefas gerais.
Como não cair: Isso intensifica o esquecimento catastrófico. Use LoRA, misture dados gerais e treine mais leve.

Como cai na prova

📝 **Como cai:** Cenário de negócio pedindo a técnica de personalização ou o padrão de arquitetura com FM.
Como resolver: Verifique: os dados mudam muito ou exigem citação (RAG)? há pares rotulados (fine-tuning)? só texto sem rótulo (CPT)? o problema é custo e latência de um modelo grande (destilação)? prazo curto e sem dados (prompt engineering)? A resposta certa casa o tipo de dado disponível com a técnica.

📝 **Como cai:** Descrição de curvas de treino e validação ou de comportamento da perda.
Como resolver: Classifique o sintoma (overfitting, underfitting, divergência, NaN, falta de memória) antes de olhar as opções e descarte as que agravam o sintoma, como regularizar em underfitting ou aumentar a taxa em divergência.

📝 **Como cai:** Escolha de métrica de avaliação.
Como resolver: Identifique o tipo de tarefa (classificação, regressão, resumo, tradução, QA, RAG) e o custo de cada erro. Depois elimine métricas de outro tipo de tarefa, como AUC em regressão ou BLEU em resumo.

📝 **Como cai:** Problema comum com restrição de equipe, dados ou prazo (sem cientista de dados, sem rótulos, poucas semanas).
Como resolver: Prefira o serviço de IA gerenciado que resolve o caso (Textract, Rekognition, Transcribe, Comprehend, Personalize) e use personalização leve como Custom Labels, vocabulário ou terminologia antes de treinar modelo próprio.

📝 **Como cai:** Ajuste de hiperparâmetros caro ou ineficiente.
Como resolver: Mapeie o sintoma: jobs ruins rodando até o fim (Hyperband ou parada antecipada Auto), busca concentrada errada (escala logarítmica), custo alto com prazo flexível (Managed Spot), dados novos (warm start TRANSFER_LEARNING), métrica ausente (MetricDefinitions).

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Fraude não detectada custa muito mais que um alerta falso	**Priorizar o recall, monitorando a precisão**

Errar por 60 dias é muito pior que errar várias vezes por 5	**RMSE**
Algumas centenas de fotos rotuladas de objetos próprios e nenhuma equipe de ML	**Amazon Rekognition Custom Labels**
Faturas com layouts variados; formulários com chave e valor e tabelas	**Textract AnalyzeExpense; Textract AnalyzeDocument com FORMS e TABLES**
Transcrição erra nomes de fundos e siglas do setor	**Amazon Transcribe com vocabulário personalizado**
Dados muito esparsos em one-hot e interações cliente-produto para prever cliques	**Factorization Machines**
Milhares de séries relacionadas e produtos novos com pouco histórico	**DeepAR**
Pontuação de anomalia sem rótulos de falha	**Random Cut Forest**
Qualidade do modelo grande com custo e latência de um menor, sem rotular à mão	**Amazon Bedrock Model Distillation (professor e aluno)**
Dezenas de jobs curtos por dia perdendo minutos no provisionamento	**Warm pools gerenciados do SageMaker AI**
AMT com muitas combinações ruins já nas primeiras épocas	**Hyperband ou parada antecipada em Auto**
Métrica objetiva do AMT não encontrada, mas aparece impressa nos logs	**Regex em MetricDefinitions**
Hiperparâmetros do estimador ignorados pelo script, ou ModuleNotFoundError no modo script	**Ler argumentos com argparse; requirements.txt no source_dir**
Recuperou certo? Inventou além dos trechos?	**Context relevance e faithfulness nas avaliações de RAG do Bedrock**

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

- Dado que muda e precisa de citação é RAG; estilo e formato com pares rotulados é fine-tuning; texto sem rótulo é CPT.
- Cabe na GPU: paralelismo de dados. Não cabe: paralelismo de modelo.
- Resumo é ROUGE, tradução é BLEU, paráfrase é BERTScore, QA com crédito parcial é F1.
- Precisão e recall em produção exigem ground truth; teste sem afetar o cliente é sombra.
- Explicar cada decisão em tempo real é SHAP local no código de inferência.

Essenciais para a prova (checklist)

☐ Sei escolher entre Rekognition, Textract, Comprehend, Comprehend Medical, Transcribe, Translate e Personalize pelo problema descrito.

☐ Sei qual algoritmo integrado usar para tabular, esparso, séries, anomalia e texto, e o formato CSV esperado pelo XGBoost.

☐ Sei configurar o modo script com argparse, requirements.txt, SM_CHANNEL_* e SM_MODEL_DIR.

☐ Sei configurar o AMT: estratégia, escala logarítmica, parada antecipada, MetricDefinitions, warm start e Spot.

☐ Sei diagnosticar overfitting, underfitting, divergência e NaN pelas curvas e aplicar a correção certa.

☐ Sei quando usar precisão mista, paralelismo de dados, paralelismo de modelo e warm pools.

☐ Sei escolher entre prompt engineering, fine-tuning, CPT, destilação e RAG e evitar esquecimento catastrófico.

☐ Sei escolher métricas de classificação, regressão e NLP pelo custo do erro e pelo tipo de tarefa.

☐ Sei usar avaliações do Bedrock automáticas, LLM-as-a-judge, humanas e de RAG (retrieve-only e retrieve-and-generate).

☐ Sei aplicar SHAP local, PDP e gradientes integrados e registrar experimentos reproduzíveis no MLflow.

Questões de revisão do capítulo

R1.1 · Desenvolvimento de modelos de ML e de modelos de base (FM)

Uma agência de notícias ajustou finamente, com os pesos liberados, um LLM de código aberto para escrever manchetes no estilo da casa. As manchetes ficaram ótimas, mas o modelo passou a errar tarefas gerais que fazia bem antes, como resumir e responder perguntas simples. Qual abordagem reduz esse problema no próximo ajuste?

A. Usar o dobro de exemplos de manchetes, sem outros tipos de tarefa
B. Usar mais épocas e uma taxa de aprendizado maior no ajuste completo
C. Usar um prompt de sistema mais longo e detalhado no modelo já ajustado e implantado
D. Usar LoRA com os pesos-base congelados e incluir exemplos de tarefas gerais

R1.2 · Desenvolvimento de modelos de ML e de modelos de base (FM)

Uma siderúrgica usa um modelo para prever a probabilidade de falha de fornos. Os engenheiros querem entender como a probabilidade prevista muda, em média, à medida que a temperatura do forno sobe de 900 para 1.300 graus, considerando o efeito dos demais atributos. Qual técnica de explicabilidade atende a esse objetivo?

A. Matriz de confusão do modelo no conjunto de validação
B. Métrica de desequilíbrio de classes (CI) para a temperatura
C. Gráfico de dependência parcial (PDP) para a temperatura
D. Valores SHAP de uma única previsão de um forno específico

R1.3 · Desenvolvimento de modelos de ML e de modelos de base (FM)

Um script de treinamento em modo script, executado com o estimador de scikit-learn do SageMaker AI, falha com ModuleNotFoundError porque usa uma biblioteca adicional que não vem no contêiner do framework. Qual é a forma mais simples de resolver?

A. Aumentar o tamanho do volume de armazenamento do job de treinamento
B. Instalar a biblioteca no notebook que dispara o job de treinamento
C. Incluir um arquivo requirements.txt no diretório source_dir do script
D. Trocar a instância de treinamento por uma família com GPU

R1.4 · Desenvolvimento de modelos de ML e de modelos de base (FM)

Uma rede de farmácias concluiu um teste de sombra no SageMaker AI. A variante sombra apresentou latência e taxa de erros iguais ou melhores que as da variante de produção durante todo o período. O que a equipe deve fazer para que o novo modelo passe a responder aos clientes?

A. Repetir o teste de sombra até a variante superar a produção em 50%
B. Excluir o endpoint atual e recriar a aplicação apontando para outro endpoint
C. Converter a variante sombra em um job de Batch Transform
D. Promover a variante sombra para assumir o papel de variante de produção

Respostas comentadas

R1.1. D) Usar LoRA com os pesos-base congelados e incluir exemplos de tarefas gerais
Por quê: O esquecimento catastrófico ocorre quando o ajuste fino sobrescreve conhecimento prévio. Técnicas PEFT como LoRA, taxa de aprendizado moderada e mistura de dados gerais (replay) reduzem o problema (tarefa 2.2).

✗ A) Incorreta. Concentrar os dados em uma tarefa reforça o esquecimento das demais.
✗ B) Incorreta. Atualizações mais intensas sobrescrevem ainda mais o conhecimento geral.
✗ C) Incorreta. O prompt não recupera capacidades perdidas nos pesos do modelo.

> **Dica de prova:** Modelo ajustado esqueceu o que sabia: LoRA e mistura de dados gerais.

R1.2. C) Gráfico de dependência parcial (PDP) para a temperatura
Por quê: Os gráficos de dependência parcial (PDPs) mostram como a resposta prevista depende de um atributo de interesse, com o efeito médio dos demais; são uma explicação global do modelo, enquanto valores SHAP de uma previsão explicam um caso individual (tarefa 2.3).

✗ A) Incorreta. Mostra acertos e erros por classe, sem relacionar a previsão à temperatura.
✗ B) Incorreta. O CI é uma métrica de viés de dados, não de explicação do modelo.
✗ D) Incorreta. Explica uma previsão isolada, não o comportamento médio ao longo da faixa de temperatura.

> **Dica de prova:** Como a previsão muda conforme um atributo varia: PDP.

R1.3. C) Incluir um arquivo requirements.txt no diretório source_dir do script

Por quê: Nos contêineres de framework do SageMaker AI, um requirements.txt colocado no source_dir é instalado automaticamente antes do entry_point, evitando a criação de uma imagem própria (tarefa 2.2).

✗ A) Incorreta. Espaço em disco não instala pacotes ausentes.

✗ B) Incorreta. O job roda em outra instância; instalar no notebook não afeta o contêiner de treinamento.

✗ D) Incorreta. O tipo de instância não fornece bibliotecas Python ausentes.

> **Dica de prova:** Dependência extra no modo script: requirements.txt no source_dir.

R1.4. D) Promover a variante sombra para assumir o papel de variante de produção

Por quê: Ao concluir um teste de sombra, a equipe compara as métricas da variante sombra com as da produção e pode promover a variante sombra, que passa a substituir o modelo atual no mesmo endpoint (tarefa 2.3).

✗ A) Incorreta. Não é necessário, pois os resultados já atendem aos critérios.

✗ B) Incorreta. Causa indisponibilidade e trabalho desnecessário.

✗ C) Incorreta. Não colocaria o modelo para responder em tempo real.

> **Dica de prova:** Variante sombra aprovada: promover para produção.

02

CAPÍTULO 2 · 24% DA PROVA

Implantação e orquestração de fluxos de trabalho de ML e IA

Aqui eu preciso escolher onde e como o modelo roda: tempo real, serverless, assíncrono ou Batch Transform; multimodelo, multicontêiner, pipeline de inferência ou inference components; Bedrock sob demanda, Provisioned Throughput, perfil entre Regiões ou Custom Model Import. Também preciso provisionar com segurança e automação (VpcConfig, endpoints de VPC, CloudFormation e CDK, contêineres, auto scaling pela métrica certa) e montar MLOps com SageMaker Pipelines, Model Registry, EventBridge e as ferramentas Code* (CodePipeline, CodeBuild, CodeDeploy, CodeConnections). A parte nova pesa: agentes no AgentCore (Runtime, Gateway, Memory, Code Interpreter, MCP e A2A) e configuração de RAG (reranking, busca híbrida, filtros, sincronização). Vale 24% e as opções erradas quase sempre são serviços reais usados fora do limite deles. Onde mais erro: serverless com GPU (não existe), multimodelo x multicontêiner, pipeline de inferência x SageMaker Pipelines e blue/green x rolling.

Neste capítulo
18 tópicos-chave · 14 pegadinhas · 8 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Opções de inferência do SageMaker AI

Tempo real para respostas síncronas em milissegundos. Serverless para tráfego intermitente em CPU que tolera cold start, escalando a zero; não oferece GPU, e picos previsíveis usam Provisioned Concurrency agendada. Assíncrona para payloads de até 1 GB e processamento de até uma hora, com fila, saída no S3, notificação pelo SNS e escala a zero. Batch Transform para volumes offline no S3 sem endpoint persistente, com InputFilter, JoinSource e OutputFilter para associar a previsão ao ID do registro.

Multimodelo, multicontêiner e pipeline de inferência

Endpoint multimodelo hospeda muitos modelos do mesmo framework e tamanho semelhante, carregados do S3 sob demanda em frota compartilhada, ideal para milhares de modelos pouco acessados. Endpoint multicontêiner aceita até 15 contêineres de frameworks diferentes, com invocação direta por TargetContainerHostname. Pipeline de inferência é uma sequência linear de 2 a 15 contêineres que processam a mesma requisição em ordem (pré-processamento, modelo, pós-processamento).

Inference components

Permitem hospedar vários modelos, inclusive LLMs, no mesmo endpoint, cada um com sua alocação de aceleradores, memória e número de cópias e com auto scaling independente. Também permitem escalar a zero: capacidade mínima zero no componente, target tracking por cópia e step scaling acionado por NoCapacityInvocationFailures. Resolvem GPU ociosa de madrugada sem trocar para serverless.

Hospedagem de LLMs no SageMaker AI

LLMs grandes trazidos de fora usam contêineres Large Model Inference (LMI) com paralelismo de tensores entre as GPUs da instância, sem escrever servidor do zero. Se o endpoint falha na verificação de integridade porque download e carregamento demoram, aumente ModelDataDownloadTimeoutInSeconds e ContainerStartupHealthCheckTimeoutInSeconds. SageMaker JumpStart atende modelos abertos quando é preciso escolher a instância com GPU, ficar na própria VPC e ajustar o servidor de inferência.

Opções de FM no Amazon Bedrock

Sob demanda, cobrado por tokens de entrada e saída, para uso baixo, imprevisível ou piloto. Provisioned Throughput reserva unidades de modelo com vazão dedicada, cobradas por hora e com prazo de compromisso, para volume alto e estável; UpdateProvisionedModelThroughput associa um novo modelo personalizado do mesmo modelo base sem mudar o ARN. Perfil de inferência entre Regiões absorve picos sem compromisso. Custom Model Import traz pesos ajustados fora da AWS (Llama, Mistral, Mixtral, Flan) para invocação serverless.

Contêineres de inferência

Contêiner próprio precisa de servidor HTTP na porta 8080 respondendo a GET /ping (integridade) e POST /invocations (previsões). Dependência de sistema, como ffmpeg, pede estender o Deep Learning Container da AWS com FROM no Dockerfile e publicar no ECR; pacote Python basta requirements.txt. Modelo de framework treinado fora (scikit-learn, PyTorch, TensorFlow, XGBoost) vai como model.tar.gz no S3 com contêiner pré-construído. No ECR, tags imutáveis preservam a rastreabilidade e políticas de ciclo de vida expiram imagens antigas.

Implantação programática de endpoints

Com Boto3, a ordem é create_model, create_endpoint_config (variantes, tipo e número de instâncias) e create_endpoint. Para trocar o modelo sem indisponibilidade e sem mudar o nome, crie novo modelo e nova configuração e chame UpdateEndpoint; configurações de endpoint não são editáveis e excluir o endpoint causa queda.

Métricas de auto scaling

SageMakerVariantInvocationsPerInstance serve a cargas comuns. Para LLMs com requisições de dezenas de segundos, ConcurrentRequestsPerModel e ConcurrentRequestsPerCopy, de alta resolução e emitidas a cada 10 segundos, contam requisições em andamento e reagem mais rápido. Endpoint assíncrono escala por ApproximateBacklogSizePerInstance, inclusive até zero. Picos no mesmo horário pedem ações programadas do Application Auto Scaling. Em instâncias com GPU, CPUUtilization pode ficar baixa com o modelo saturado.

Rede de endpoints e jobs

VpcConfig no modelo (sub-redes e grupos de segurança) conecta o contêiner à VPC para acessar recursos privados como RDS. Interface endpoint (AWS PrivateLink) para o SageMaker Runtime permite invocar o endpoint sem passar pela internet; o acesso privado ao S3 usa gateway endpoint. Gateway endpoints existem só para S3 e DynamoDB. Isolamento de rede bloqueia chamadas de saída do contêiner, inclusive ao RDS.

Infraestrutura como código

No CloudFormation, a stack de rede exporta IDs nas Outputs e a stack de ML importa com Fn::ImportValue (mesma conta e Região). O AWS CDK define infraestrutura em Python com constructs reutilizáveis e cria automaticamente export e Fn::ImportValue ao referenciar recurso de outra stack. Runtime de agentes e seus endpoints também devem ser definidos como código e implantados por pipeline entre contas.

Capacidade e custo de treino

SageMaker training plans reservam capacidade de GPU por período para jobs de treinamento e clusters do HyperPod; Savings Plans dão desconto mas não reservam, e Spot não garante. Managed Spot Training exige use_spot_instances, max_wait maior ou igual a max_run e checkpoint_s3_uri; o volume local se perde na interrupção. Modo FastFile expõe o S3 como sistema de arquivos POSIX com streaming sob demanda, começando o treino sem baixar tudo e sem mudar o código; o modo Pipe exige reescrever o leitor.

SageMaker Pipelines

Parâmetros (ParameterString, ParameterInteger) trocam tipo de instância e caminho dos dados por ambiente com uma única definição. CacheConfig reaproveita etapas cujos argumentos não mudaram. ConditionStep avalia métricas do relatório de avaliação e decide entre registrar o modelo ou encerrar com FailStep; um ProcessingStep valida os dados novos contra a linha de base antes do treino. O EventBridge inicia o pipeline por chegada de objeto no S3, mudança de estado de alarme do CloudWatch ou agendamento do EventBridge Scheduler.

Model Registry, linhagem e MLflow

O Model Registry versiona modelos em grupos com status como PendingManualApproval e Approved; a mudança de status gera evento no EventBridge que pode iniciar o CodePipeline. ML Lineage Tracking liga dados, jobs, hiperparâmetros e artefatos para auditoria. O MLflow gerenciado, com a sincronização do Model Registry habilitada, cria grupo e versão de pacote para cada modelo registrado sem abandonar o MLflow.

Ferramentas de CI/CD

CodePipeline orquestra estágios e tem ação de aprovação manual com notificação pelo SNS. CodeBuild roda testes unitários e constrói imagens: docker build exige modo privilegiado, e acesso negado ao ECR ou ao S3 se corrige na função de serviço. CodeDeploy faz canário ou linear em alias do Lambda (como LambdaCanary10Percent5Minutes) com rollback por alarme. CodeConnections liga GitHub, GitLab ou Bitbucket como origem sem token pessoal. Testes certos: unitários, integração em homologação, contrato de esquema de entrada e saída, suíte adversarial de prompts e regressão de prompt com limiar.

Deployment guardrails

Blue/green (all at once, canary ou linear) desloca o tráfego para uma frota nova com período de observação e rollback automático por alarmes do CloudWatch. Rolling provisiona a nova frota em lotes, sem dobrar a capacidade, útil quando não há cota de GPU para duplicar a frota. Teste de sombra não é liberação gradual, pois a variante sombra não responde aos clientes.

Amazon Bedrock AgentCore

Runtime hospeda agentes de qualquer framework (Strands Agents, LangGraph, CrewAI) sem servidor, com isolamento de sessão em microVMs e sessões de até 8 horas; cada atualização cria versão imutável, o endpoint DEFAULT segue a mais recente e endpoints nomeados ficam fixados para promoção e rollback. Gateway converte APIs REST e funções Lambda em ferramentas MCP com autenticação; Memory guarda curto e longo prazo (preferências, fatos semânticos, resumos); Code Interpreter executa código em sandbox; Browser automatiza páginas; Identity gerencia credenciais.

Comunicação e ferramentas de agentes

MCP expõe ferramentas para agentes; A2A (Agent2Agent) permite que agentes de frameworks diferentes descubram capacidades e deleguem tarefas entre si. Um ponto único de contato para vários especialistas é um agente supervisor que usa os especialistas como ferramentas (agent-as-tool); o Bedrock Agents (agora Agents Classic) não está aberto a novos clientes. Ação que só a aplicação cliente alcança usa ferramenta de função inline, executada no cliente (retorno de controle).

Configuração de RAG

Retrieve devolve trechos para montar o prompt no código (controle total, combinar com CRM, escolher modelo); RetrieveAndGenerate entrega a resposta pronta. Ajustes: numberOfResults maior quando a resposta exige muitos documentos, reranking quando o trecho certo está no top 30 mas não no top 5, overrideSearchType HYBRID para códigos exatos, filtros por .metadata.json, decomposição de consulta para perguntas compostas e parser de FM para tabelas em PDF. Índice próprio precisa de knn_vector com a dimensão do modelo de embeddings; StartIngestionJob sincroniza de forma incremental, orquestrado por EventBridge Scheduler ou Step Functions.

Macetes para memorizar

💡 Assíncrona

Arquivo grande, Aguarda minutos, Apaga até zero.

💡 Multimodelo x multicontêiner

Multimodelo é Mesmo framework e Muitos modelos; multicontêiner é Mistura de frameworks, até 15.

💡 Pipeline de inferência x SageMaker Pipelines

Pipeline de inferência é fila de contêineres em uma chamada; SageMaker Pipelines é a linha de montagem do treino.

💡 Endpoint via API

M-C-E: Modelo, depois Configuração, depois Endpoint. Trocar modelo: nova configuração e Update.

💡 Endpoints de VPC

Gateway só para as duas casas de dados (S3 e DynamoDB); todo o resto é interface (PrivateLink).

💡 Agentes

MCP é agente usando ferramenta; A2A é agente conversando com agente.

💡 Contêiner próprio

Porta 8080: /ping para saúde, /invocations para trabalho.

💡 Escalabilidade

Pico com hora marcada, agenda; pico surpresa, métrica; fila, backlog; LLM, concorrência.

Tabela de memorização: não confunda

Quando é...	...e quando é
Inferência assíncrona Payload grande (até 1 GB), processamento de minutos (até uma hora), resultado por notificação e escala a zero.	**Inferência serverless** Modelo leve em CPU, tráfego intermitente, respostas curtas e tolerância a cold start.
Endpoint multimodelo Centenas ou milhares de modelos do mesmo framework, pouco acessados, carregados sob demanda.	**Endpoint multicontêiner** Poucos modelos (até 15 contêineres) de frameworks diferentes, chamados de forma independente por invocação direta.
Pipeline de inferência Pré-processamento, modelo e pós-processamento devem rodar em sequência dentro de uma única chamada ao endpoint.	**SageMaker Pipelines** É preciso orquestrar em lote etapas de processamento, treino, avaliação, condição e registro do modelo.
Provisioned Throughput Volume alto e estável 24 horas por dia com capacidade garantida e prazo de compromisso.	**Perfil de inferência entre Regiões** Throttling só em picos, volume baixo no resto do mês e dados que podem ser processados em outras Regiões da mesma geografia.

Blue/green canário Há capacidade para provisionar a frota nova completa e enviar uma fatia pequena do tráfego antes da troca.	**Rolling** Não há cota de GPU para duplicar a frota; a atualização vai em lotes com rollback por alarme.
AgentCore Gateway (MCP) O agente precisa usar APIs REST e funções Lambda existentes como ferramentas.	**Protocolo A2A** Um agente precisa descobrir as capacidades de outro agente, de outro framework, e delegar tarefas a ele.
API Retrieve A equipe quer controlar o prompt final, combinar trechos com outros dados e escolher o modelo de geração.	**API RetrieveAndGenerate** Basta a resposta integrada da base de conhecimento com o prompt padrão.

Números que caem

Item	Valor
Peso do domínio no exame	**24%**
Inferência assíncrona	**payload de até 1 GB e processamento de até uma hora**
Endpoint multicontêiner	**até 15 contêineres**
Pipeline de inferência	**sequência de 2 a 15 contêineres**
Porta do contêiner de inferência	**8080 (GET /ping e POST /invocations)**
Sessão no AgentCore Runtime	**até 8 horas**
ConcurrentRequestsPerModel e PerCopy	**emitidas a cada 10 segundos**
Tempo máximo do Lambda	**15 minutos**
Managed Spot Training	**max_wait maior ou igual a max_run**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Escolher endpoint serverless com GPU para LLM ou para eliminar falha de instância.
Como não cair: Endpoints serverless do SageMaker AI não oferecem GPU. Para GPU ociosa, use inference components com escala a zero ou inferência assíncrona.

⚠ **Alerta de prova:** Usar endpoint multicontêiner para centenas ou milhares de modelos.
Como não cair: Multicontêiner aceita no máximo 15 contêineres. Muitos modelos do mesmo framework vão para endpoint multimodelo.

⚠ **Alerta de prova:** Editar a configuração de endpoint existente ou excluir e recriar o endpoint para trocar o modelo.
Como não cair: Configurações não são editáveis e excluir causa indisponibilidade. Crie modelo e configuração novos e chame UpdateEndpoint.

⚠ **Alerta de prova:** Criar gateway endpoint para o SageMaker Runtime ou para o RDS.
Como não cair: Gateway endpoints existem só para S3 e DynamoDB. Invocação privada usa interface endpoint; acesso ao RDS usa VpcConfig no modelo.

⚠ **Alerta de prova:** Ativar isolamento de rede para o contêiner acessar o banco privado.
Como não cair: O isolamento bloqueia chamadas de saída do contêiner, inclusive ao RDS. Use VpcConfig com sub-redes e grupos de segurança.

⚠ **Alerta de prova:** Escolher modo Pipe para começar o treino mais cedo sem alterar o código de leitura.
Como não cair: Pipe exige reescrever o leitor. FastFile transmite sob demanda com acesso POSIX e não muda o código.

⚠ **Alerta de prova:** Definir max_wait menor que max_run no Managed Spot Training ou guardar o estado no volume local.
Como não cair: max_wait deve ser maior ou igual a max_run, e os checkpoints vão para um URI do S3.

⚠ **Alerta de prova:** Usar Savings Plans ou Spot com espera longa para garantir GPUs em uma data futura.
Como não cair: Savings Plans só dão desconto e Spot não garante. Reserva de capacidade de GPU por período é training plans.

⚠ **Alerta de prova:** Usar teste de sombra como liberação gradual com rollback.
Como não cair: A sombra não responde aos clientes. Liberação gradual é blue/green canário ou linear com alarmes do CloudWatch.

⚠ **Alerta de prova:** Apontar a aplicação de produção para o endpoint DEFAULT do AgentCore Runtime.
Como não cair: O DEFAULT acompanha cada atualização e leva versões não testadas à produção. Use endpoint nomeado fixado em versão e reaponte para reverter.

⚠ **Alerta de prova:** Hospedar agente com sessões de horas em uma função Lambda com o tempo máximo.
Como não cair: O Lambda para em 15 minutos. Agentes longos, isolados e sem servidor vão para o AgentCore Runtime.

⚠ **Alerta de prova:** Listar o ffmpeg no requirements.txt.
Como não cair: requirements.txt instala pacotes Python; binário do sistema exige estender a imagem pré-construída com Dockerfile.

⚠ **Alerta de prova:** Escalar LLM em GPU por CPUUtilization ou endpoint assíncrono por InvocationsPerInstance.
Como não cair: LLM com requisições longas escala por ConcurrentRequestsPerModel ou PerCopy; assíncrono escala por ApproximateBacklogSizePerInstance.

⚠ **Alerta de prova:** Recriar a base de conhecimento a cada hora ou fazer ajuste fino diário para refletir documentos novos.
Como não cair: StartIngestionJob sincroniza só o que mudou; agende com EventBridge Scheduler ou orquestre com Step Functions.

Como cai na prova

📝 **Como cai:** Descrição do padrão de tráfego, tamanho do payload e tempo de processamento pedindo a opção de inferência.
Como resolver: Classifique: milissegundos e síncrono (tempo real), intermitente em CPU com cold start aceito (serverless), payload grande e minutos com escala a zero (assíncrona), volume offline agendado (Batch Transform). Elimine qualquer opção que dependa de GPU em serverless.

📝 **Como cai:** Muitos modelos ou várias etapas para hospedar com menor custo.
Como resolver: Pergunte: os modelos são alternativos do mesmo framework (multimodelo), independentes de frameworks diferentes (multicontêiner), etapas da mesma requisição (pipeline de inferência) ou FMs que precisam de recursos e escala por modelo (inference components)?

📝 **Como cai:** Automatizar retreino e implantação com governança.
Como resolver: Monte a cadeia: gatilho (EventBridge por S3, alarme ou agenda), SageMaker Pipelines com validação de dados e ConditionStep, Model Registry com aprovação, EventBridge iniciando o CodePipeline e deployment guardrails com rollback por alarme.

📝 **Como cai:** Liberar nova versão de modelo, função ou agente com segurança.
Como resolver: Endpoint do SageMaker AI: blue/green canário ou rolling se faltar cota. Lambda: CodeDeploy canário no alias. Agente: versão imutável e endpoint nomeado. Sombra só valida, não libera.

📝 **Como cai:** Assistente de RAG com respostas incompletas ou imprecisas.
Como resolver: Localize a falha: poucos resultados (numberOfResults), trecho certo mal posicionado (reranking), identificador exato (HYBRID), mistura de áreas (filtro de metadados), pergunta composta (decomposição), tabela em PDF (parser de FM).

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Exames de 800 MB, 5 a 10 minutos por inferência, carga irregular e escala a zero	**Endpoint de inferência assíncrona**
Toda madrugada, milhões de registros no S3 e resultado usado na manhã seguinte	**Job agendado de Batch Transform**
Previsão em lote precisa sair associada ao ID que não pode ir ao modelo	**InputFilter para tirar o ID e JoinSource=Input na saída**
Milhares de modelos XGBoost parecidos, cada um com poucas chamadas por dia	**Endpoint multimodelo**
Pré-processamento scikit-learn e modelo XGBoost em uma única chamada	**Pipeline de inferência serial**
Vários LLMs compartilhando GPUs com recursos e escala por modelo	**Inference components**
Carga triplica todo dia no mesmo horário e o auto scaling reage tarde	**Ação de escalabilidade programada antes do pico**
Modelo aberto ajustado fora da AWS, invocado pelo Bedrock sob demanda sem gerenciar endpoint	**Amazon Bedrock Custom Model Import**
Cannot connect to the Docker daemon no CodeBuild	**Modo privilegiado no ambiente do projeto**
Push no GitHub deve iniciar o CodePipeline sem token pessoal	**Conexão do AWS CodeConnections**
Versão aprovada no Model Registry deve implantar sozinha	**Regra do EventBridge para a mudança de status iniciando o CodePipeline**
Registrar o modelo só se a AUC passar do limite; caso contrário falhar com mensagem clara	**ConditionStep seguido de RegisterModel ou FailStep**
O trecho certo aparece entre os 30 primeiros, mas não entre os 5 enviados	**Recuperar mais candidatos e aplicar reranking**
Busca semântica não acha o documento com o número exato de processo ou sigla	**overrideSearchType HYBRID**

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

- Serverless do SageMaker AI não tem GPU.
- Gateway endpoint é só para S3 e DynamoDB; invocar endpoint privado é interface endpoint do SageMaker Runtime.
- Configuração de endpoint não se edita: nova configuração e UpdateEndpoint.
- Aprovação no Model Registry e falha de pipeline viram eventos do EventBridge.
- Produção de agente fica em endpoint nomeado fixado em versão, nunca no DEFAULT.

Essenciais para a prova (checklist)

☐ Sei escolher entre tempo real, serverless, assíncrona e Batch Transform pelo payload, tempo e padrão de tráfego.

☐ Sei diferenciar endpoint multimodelo, multicontêiner, pipeline de inferência e inference components.

☐ Sei escolher entre Bedrock sob demanda, Provisioned Throughput, perfil entre Regiões, Custom Model Import e JumpStart.

☐ Sei o contrato do contêiner próprio e quando estender a imagem ou usar requirements.txt.

☐ Sei criar e atualizar endpoints por Boto3 e escolher a métrica de auto scaling de cada tipo de endpoint.

☐ Sei configurar VpcConfig, interface endpoint do SageMaker Runtime e gateway endpoint do S3.

☐ Sei integrar stacks com Outputs e Fn::ImportValue e usar o CDK com referências entre stacks.

☐ Sei montar SageMaker Pipelines com parâmetros, cache, ConditionStep, FailStep e gatilhos do EventBridge.

☐ Sei usar CodePipeline, CodeBuild, CodeDeploy, CodeConnections e deployment guardrails com rollback.

☐ Sei hospedar e versionar agentes no AgentCore e configurar retrieve, reranking, busca híbrida e sincronização de RAG.

Questões de revisão do capítulo

R2.1 · Implantação e orquestração de fluxos de trabalho de ML e IA

Uma fabricante de baterias executa várias vezes por dia um pipeline do SageMaker Pipelines enquanto ajusta apenas a etapa de avaliação. As etapas de processamento e treinamento, com as mesmas entradas e parâmetros, são refeitas em cada execução e consomem horas. Qual configuração reduz esse tempo?

A. Dividir o pipeline em dois e iniciar cada parte a partir de um notebook
B. Aumentar o tipo de instância das etapas de processamento e treino
C. Ativar o cache das etapas com CacheConfig nas etapas que não mudaram
D. Ativar a parada antecipada do treinamento em cada nova execução

R2.2 · Implantação e orquestração de fluxos de trabalho de ML e IA

Uma operadora mantém uma stack do AWS CloudFormation de rede, que cria a VPC e as sub-redes, e outra stack que provisiona domínios e endpoints do SageMaker AI. A stack de ML precisa usar os IDs das sub-redes sem copiá-los à mão. Como integrar as stacks?

A. Exportar os IDs nas Outputs da stack de rede e usar Fn::ImportValue na stack de ML
B. Copiar os IDs para um arquivo no Amazon S3 e lê-lo com um parâmetro sem valor padrão
C. Recriar a VPC e as sub-redes dentro da stack de ML para manter tudo em um só lugar
D. Gravar os IDs em comentários do template de ML e atualizá-los a cada mudança de rede

R2.3 · Implantação e orquestração de fluxos de trabalho de ML e IA

Uma rede de rádio e TV tem três agentes especialistas hospedados no Amazon Bedrock AgentCore Runtime: um de grade de programação, um de direitos autorais e um de vendas de anúncios. Os usuários querem um único ponto de contato que entenda a pergunta, acione os agentes especialistas necessários e consolide a resposta. Qual configuração atende ao requisito?

A. Configurar um endpoint para cada agente e deixar o usuário escolher o especialista pela interface
B. Criar um agente supervisor que aciona os especialistas como ferramentas e combina as respostas
C. Configurar um único agente com os prompts dos três especialistas juntos
D. Configurar uma base de conhecimento compartilhada pelos três agentes

R2.4 · Implantação e orquestração de fluxos de trabalho de ML e IA

Uma universidade usa um modelo personalizado no Amazon Bedrock com Provisioned Throughput. Um novo job de ajuste fino, feito a partir do mesmo modelo base, gerou uma versão melhor. A aplicação deve passar a usar a nova versão sem mudar o ARN do Provisioned Throughput que ela chama. O que fazer?

A. Excluir o modelo antigo e renomear o novo modelo personalizado com o nome do anterior
B. Criar um alias de agente que aponte para o novo modelo personalizado e mantenha o ARN
C. Sobrescrever os pesos do modelo personalizado antigo com os gerados pelo novo job de ajuste
D. Atualizar o Provisioned Throughput existente para associá-lo ao novo modelo personalizado

Respostas comentadas

R2.1. C) Ativar o cache das etapas com CacheConfig nas etapas que não mudaram

Por quê: O cache de etapas do SageMaker Pipelines reutiliza os resultados de etapas cujos argumentos não mudaram, encurtando execuções repetidas durante o desenvolvimento (tarefa 3.3).

✗ A) Incorreta. Adiciona trabalho fora do pipeline e perde a orquestração.
✗ B) Incorreta. Acelera um pouco, mas continua refazendo trabalho idêntico.
✗ D) Incorreta. Muda o treino em si e não evita refazer o processamento.

> **Dica de prova:** Pipeline refazendo etapas iguais: cache de etapas (CacheConfig).

R2.2. A) Exportar os IDs nas Outputs da stack de rede e usar Fn::ImportValue na stack de ML

Por quê: Automatizar o provisionamento com comunicação entre stacks usa Outputs exportados e Fn::ImportValue, ou referências entre stacks no AWS CDK (tarefa 3.2).

✗ B) Incorreta. Cria um processo manual frágil fora do controle do CloudFormation.
✗ C) Incorreta. Duplica a rede e quebra a separação entre as stacks.
✗ D) Incorreta. Comentários não são lidos pelo CloudFormation.

> **Dica de prova:** Valor de uma stack em outra: Export + Fn::ImportValue.

R2.3. B) Criar um agente supervisor que aciona os especialistas como ferramentas e combina as respostas

Por quê: No Amazon Bedrock AgentCore, o padrão multiagente com supervisor é implementado expondo os agentes especialistas como ferramentas (agent-as-tool) ou com frameworks como o Strands Agents no AgentCore Runtime; o Bedrock Agents (agora Agents Classic) não está aberto a novos clientes (tarefa 3.1).

✗ A) Incorreta. Transfere ao usuário a escolha do especialista, o oposto do ponto único de contato.
✗ C) Incorreta. Misturar as instruções em um agente reduz a especialização e dificulta a manutenção.
✗ D) Incorreta. Compartilhar conhecimento não cria a coordenação entre os agentes.

> **Dica de prova:** Vários agentes especialistas com ponto único de entrada: supervisor que usa os agentes como ferramentas.

R2.4. D) Atualizar o Provisioned Throughput existente para associá-lo ao novo modelo personalizado

Por quê: Para automatizar a troca de versões de modelos ajustados, a API UpdateProvisionedModelThroughput associa ao mesmo Provisioned Throughput um novo modelo personalizado derivado do mesmo modelo base (tarefa 3.3).

✗ A) Incorreta. Renomear não substitui a associação do Provisioned Throughput.

✗ B) Incorreta. Aliases de agente versionam agentes, não o throughput provisionado de modelos.

✗ C) Incorreta. Os pesos de um modelo personalizado não podem ser sobrescritos.

> **Dica de prova:** Trocar o modelo ajustado sem trocar o ARN: atualizar o Provisioned Throughput.

03

CAPÍTULO 3 · 24% DA PROVA

Operação, monitoramento e segurança de soluções de ML e IA

Aqui eu preciso manter modelos, FMs e agentes saudáveis, baratos e seguros em produção. Monitoramento: A/B x sombra, métricas e logs do endpoint, alarmes com detecção de anomalias, drift de dados, de qualidade e de viés, observabilidade de IA generativa do CloudWatch, invocation logging do Bedrock e AgentCore Observability. Custo: opções de compra (SageMaker Savings Plans, Spot), famílias de instância (inf2, G, P, CPU), prompt caching, cache semântico, roteamento de prompts, inferência em lote e ferramentas como Budgets e Cost Anomaly Detection. Segurança: IAM com menor privilégio (PassRole, chaves de condição, ABAC), KMS, rede (isolamento, endpoints de VPC), auditoria (CloudTrail, Config, Model Cards) e Bedrock Guardrails. Vale 24% e as alternativas erradas costumam ser serviços que existem mas medem outra coisa; onde mais erro é trocar CloudTrail, Config, CloudWatch e invocation logging entre si.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 8 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Teste A/B e teste de sombra

Teste A/B usa production variants no mesmo endpoint com pesos de tráfego (por exemplo 0,9 e 0,1), métricas do CloudWatch por variante e ajuste dos pesos pela API sem recriar o endpoint; mede o efeito real no comportamento do usuário. No teste de sombra, as respostas da nova versão não chegam aos usuários, então ele valida desempenho técnico, não efeito de negócio. Alternar o modelo inteiro por semana mistura o efeito do modelo com o do calendário.

Métricas e logs de endpoints do SageMaker AI

ModelLatency mede o tempo do contêiner e OverheadLatency o tempo adicional da plataforma; GPUUtilization e GPUMemoryUtilization mostram saturação; Invocation5XXErrors alerta erros do contêiner. Rastreamentos de exceção ficam no CloudWatch Logs em /aws/sagemaker/Endpoints com o nome do endpoint. Erro que só aparece no log (resposta com sucesso, mas sem resultado) pede filtro de métricas com alarme; investigar milhões de linhas JSON pede CloudWatch Logs Insights.

Alarmes, dashboards e rastreamento

Alarmes com detecção de anomalias do CloudWatch modelam tendência e sazonalidade e servem para métricas com padrão diário, onde limite fixo gera alarme falso ou tardio. Dashboards do CloudWatch reúnem latência, invocações, 5XX e GPU com atualização automática, inclusive entre contas e Regiões. O AWS X-Ray mostra a latência de cada segmento em cadeias como API Gateway, Lambda e endpoint ou base de conhecimento.

Drift de dados, de qualidade e de viés

Qualidade de dados compara estatísticas dos dados capturados no endpoint com a linha de base do treino e detecta mudança na distribuição sem rótulos. Qualidade do modelo exige juntar os rótulos reais, que chegam com atraso, às previsões pelo ID da inferência para calcular precisão e recall. Drift de viés recalcula periodicamente métricas por grupo, como DPPL, e alerta ao passar do limite. Para inferência em lote, o monitoramento é uma etapa do pipeline após o Batch Transform; o agendamento pode ser um job do SageMaker Processing disparado pelo EventBridge.

Detecção de anomalias do Glue Data Quality

Coleta estatísticas dos dados a cada execução e usa ML para apontar desvios em relação ao padrão histórico, como queda brusca de volume ou mudança de média, sem limites fixos. É a resposta para jobs que não falham, mas processam dados estranhos, e para checar lotes de entrada antes da inferência. Job bookmarks, crawlers e auto scaling de workers não detectam anomalias.

Monitoramento de FMs no Bedrock

O Bedrock publica no CloudWatch Invocations, InvocationLatency, InvocationThrottles, InvocationClientErrors e contagens de tokens. O model invocation logging, desativado por padrão, grava entradas, saídas e metadados no CloudWatch Logs e/ou no S3. A observabilidade de IA generativa do CloudWatch traz painéis prontos de tokens, latência P90 e P99 e erros. Qualidade em produção se mede com avaliações do Bedrock sobre amostras dos logs (respostas próprias com LLM-as-a-judge) ou com dataset fixo a cada nova versão do modelo.

Sinais de FMs e guardrails

A métrica InvocationsIntervened do Guardrails mostra quantas vezes o guardrail bloqueou, e merece alarme após mudanças de prompt. O stopReason no evento de fim de mensagem do Converse (end_turn, max_tokens, guardrail_intervened) detecta respostas truncadas: max_tokens indica corte por limite. Throttling gera erro de limitação na chamada, não respostas cortadas nem acesso negado.

AgentCore Observability

Emite traces, spans e métricas compatíveis com OpenTelemetry no CloudWatch, mostrando cada etapa da sessão, chamadas de ferramentas, escolha de agentes pelo supervisor, latência e tokens. Loops entre agentes se detectam com alarme sobre chamadas repetidas nos traces e se contêm com limite de iterações por sessão. Consumo por agente vem das métricas CPUUsed-vCPUHours e MemoryUsed-GBHours do AgentCore Runtime, que é serverless e não tem instâncias do cliente.

Otimização de custo de FMs

Cache de prompts reaproveita prefixos longos e fixos, que devem ficar no início com um checkpoint após eles, reduzindo custo e latência de tokens de entrada. Cache semântico guarda embeddings de perguntas e respostas (por exemplo no ElastiCache for Valkey com busca vetorial) e reaproveita respostas de perguntas parecidas. Intelligent prompt routing manda cada requisição ao modelo da mesma família com melhor relação qualidade e custo. Inferência em lote (JSONL no S3) custa cerca de 50% menos que a sob demanda, com janela de conclusão de 24 horas.

Custo de embeddings e alocação por aplicação

Reindexar tudo todo dia gera embeddings de documentos que não mudaram; a sincronização da base de conhecimento é incremental. Reduzir a dimensão do Titan Text Embeddings V2 (256, 512 ou 1.024) corta memória do índice com pequena perda de precisão, exigindo regerar e reindexar; réplicas multiplicam a memória. Para separar custo de tokens por aplicação ou equipe no mesmo modelo e conta, use application inference profiles com tags ativadas como tags de alocação de custos.

Opções de compra para ML

SageMaker Savings Plans dá desconto em troca de compromisso de gasto por hora por 1 ou 3 anos, com flexibilidade de tipo de instância e Região, para uso estável. Compute Savings Plans cobre EC2, Fargate e Lambda, não o SageMaker AI, e Reserved Instances do EC2 não se aplicam a instâncias gerenciadas de ML. Managed Spot Training com checkpoints serve a treinos interrompíveis com prazo flexível; endpoints de tempo real não usam Spot.

Famílias de instância de inferência

ml.inf2 (AWS Inferentia2) dá menor custo por inferência de deep learning, com o modelo compilado pelo AWS Neuron SDK. Família G (g5, g6) equilibra custo e desempenho em GPU para modelos de porte médio; família P (p4d, p5) é para treino em larga escala e modelos muito grandes. Modelos de árvore como XGBoost servem bem em CPU (por exemplo c7i), e GPUUtilization perto de zero indica desperdício; ml.t3 esgota créditos com carga contínua. O Inference Recommender faz testes de carga em tipos e quantidades de instância.

Disponibilidade e ferramentas de custo

Endpoints de produção devem ter duas ou mais instâncias, que o SageMaker AI distribui entre zonas de disponibilidade; mais instâncias menores equilibram custo e resiliência. AWS Budgets com ações aplica política IAM ou SCP, ou interrompe instâncias, ao atingir o limite, transformando orçamento em bloqueio. Cost Anomaly Detection alerta gastos fora do padrão histórico por serviço, conta ou tag sem definir limite por recurso. Cost Explorer, Compute Optimizer e Trusted Advisor só informam ou recomendam.

IAM com menor privilégio

Aplicações usam funções com credenciais temporárias (função de execução do Lambda, IAM role da tarefa do ECS), com ações como sagemaker:InvokeEndpoint restritas ao ARN; nunca chaves de usuário ou root em variáveis de ambiente. Uma função de execução por projeto limitada ao prefixo do S3 segrega projetos. Erro de iam:PassRole se corrige permitindo PassRole na função específica com iam:PassedToService; permissions boundaries limitam funções criadas por usuários delegados. ABAC compara aws:ResourceTag e aws:PrincipalTag, e o IAM Access Analyzer gera política a partir da atividade no CloudTrail.

Chaves de condição e SCPs

sagemaker:VpcSubnets, sagemaker:VpcSecurityGroupIds e sagemaker:VolumeKmsKey impõem rede e criptografia na criação de jobs; Deny com condição Null em VpcSubnets impede jobs fora da VPC de forma preventiva. bedrock:GuardrailIdentifier em política IAM força o guardrail aprovado em Converse, ConverseStream, InvokeModel e InvokeModelWithResponseStream. Na inferência entre Regiões, IAM e SCPs precisam permitir as Regiões de destino do perfil; SCP restritiva causa AccessDeniedException em parte das chamadas.

KMS e proteção de artefatos

Com SSE-KMS e chave gerenciada pelo cliente, ler o objeto também exige kms:Decrypt, concedido pela política da chave ou pelo IAM quando a chave delega ao IAM; restringir essa permissão limita quem lê mesmo com acesso ao bucket. Com SSE-S3, quem tem s3:GetObject recebe os objetos já decifrados. AccessDenied com s3:GetObject presente em bucket SSE-KMS aponta para falta de kms:Decrypt; política de bucket negando outros principais completa o controle.

Rede segura para ML

EnableNetworkIsolation impede que o contêiner de treino ou inferência faça chamadas de saída; o próprio serviço transfere dados e artefatos. Treino distribuído em VPC precisa de regra de entrada autorreferente no grupo de segurança, e a criptografia entre contêineres protege o tráfego entre nós. Jobs em sub-rede privada sem NAT travam no S3 até ganhar um endpoint de gateway do S3. Interface endpoint do bedrock-runtime com política de endpoint restringe quais modelos podem ser invocados a partir da VPC.

Auditoria, governança e Guardrails

CloudTrail registra quem chamou APIs como DeleteEndpoint, de qual IP e quando, mas não o conteúdo de prompts; a validação de integridade dos arquivos de log comprova que não foram alterados. AWS Config avalia configuração continuamente, com regras como sagemaker-endpoint-configuration-kms-key-configured. SageMaker Model Cards documentam uso pretendido, risco, treino e avaliação, exportáveis em PDF; Lake Formation dá permissões por coluna, linha e célula. Bedrock Guardrails mascara ou bloqueia PII, nega tópicos, filtra ataques de prompt em conteúdo marcado com input tags e faz contextual grounding check contra alucinações em RAG.

Macetes para memorizar

Quem registra o quê
CloudTrail: quem chamou. Invocation logging: o que foi dito. Config: como está configurado. CloudWatch: como está performando.

A/B x sombra
A/B Afeta o usuário; Sombra Só observa.

Monitores
Sem rótulo, olha os dados; com ground truth, olha a precisão; por grupo, olha o viés.

Savings Plans
SageMaker Savings Plan para SageMaker; Compute é EC2, Fargate e Lambda.

Caches de FM
Cache de prompt: mesmo começo. Cache semântico: mesma pergunta com outras palavras.

Instâncias
inf2 barateia inferência; G infere em GPU; P treina pesado; árvore roda em CPU.

Custo
Budgets com ação bloqueia; Anomaly Detection avisa o que foge do padrão; o resto só recomenda.

Criptografia
SSE-S3 abre para quem lê; SSE-KMS do cliente pede kms:Decrypt.

Tabela de memorização: não confunda

Quando é...	...e quando é
Teste A/B com production variants É preciso medir o efeito real no comportamento dos usuários, como títulos assistidos ou taxa de conclusão.	**Teste de sombra** É preciso validar latência e erros com tráfego real sem que nenhum usuário receba a resposta nova.
Monitoramento de qualidade de dados Os rótulos reais ainda não chegaram e o objetivo é detectar mudança na distribuição dos atributos.	**Monitoramento de qualidade do modelo** Os rótulos reais chegam (mesmo com atraso) e o objetivo é acompanhar precisão e recall em produção.
AWS CloudTrail É preciso saber quem fez uma chamada de API, de onde e quando.	**Model invocation logging do Bedrock** É preciso guardar o texto completo dos prompts e das respostas para investigar e avaliar qualidade.
AWS Config É preciso comprovar continuamente que configurações de recursos seguem uma regra, como endpoint com chave KMS.	**Amazon CloudWatch** É preciso acompanhar métricas, logs, latência, erros e alarmes de desempenho.
SageMaker Savings Plans O uso estável é de endpoints e instâncias gerenciadas do SageMaker AI.	**Compute Savings Plans** O uso estável é de EC2, Fargate ou Lambda; não cobre o SageMaker AI.

Cache de prompts Toda requisição repete o mesmo prefixo longo (prompt de sistema, manual de estilo) seguido de conteúdo variável.	**Cache semântico** Muitas perguntas são variações com outras palavras das mesmas dúvidas frequentes.
SSE-KMS com chave gerenciada pelo cliente Só funções específicas podem decifrar os artefatos, mesmo que outros tenham leitura no bucket.	**SSE-S3** Basta criptografia em repouso, sem controlar quem decifra; quem lê recebe os dados abertos.

Números que caem

Item	Valor
Peso do domínio no exame	**24%**
Inferência em lote do Bedrock	**cerca de 50% menos que sob demanda, janela de conclusão de 24 horas**
Compromisso do SageMaker Savings Plans	**1 ou 3 anos, gasto por hora**
Instâncias mínimas em endpoint de produção	**duas ou mais, distribuídas entre zonas de disponibilidade**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Usar o AWS Config para medir latência, contar falhas ou analisar logs e invocações.
Como não cair: O Config avalia configuração de recursos. Métricas, logs e alarmes são CloudWatch; conteúdo de invocações é invocation logging.

⚠ **Alerta de prova:** Ativar o CloudTrail para ler o texto dos prompts e respostas.
Como não cair: O CloudTrail registra chamadas de API, não o conteúdo. Ative o model invocation logging do Bedrock.

⚠ **Alerta de prova:** Contratar Compute Savings Plan ou Reserved Instances do EC2 para endpoints do SageMaker AI.
Como não cair: Nenhum dos dois cobre instâncias gerenciadas de ML. Use SageMaker Savings Plans.

⚠ **Alerta de prova:** Usar instâncias Spot em endpoints de tempo real de produção.
Como não cair: Endpoints de tempo real não usam Spot. Spot é para treino interrompível com checkpoints.

⚠ **Alerta de prova:** Comprar Provisioned Throughput ou subir a temperatura para baratear perguntas repetidas.
Como não cair: Capacidade reservada não evita processamento repetido e temperatura não controla custo. Use cache semântico ou cache de prompts.

⚠ **Alerta de prova:** Mover o conteúdo fixo do prompt para depois da pergunta variável.
Como não cair: O cache de prompts reaproveita prefixos: fixo no início, variável no fim, com checkpoint após o prefixo.

⚠ **Alerta de prova:** Criar alarme de Invocation5XXErrors para falhas em que o contêiner responde com sucesso.
Como não cair: Sem erro HTTP não há 5XX. Use filtro de métricas no grupo de logs do endpoint com alarme.

⚠ **Alerta de prova:** Agendar o monitor de qualidade do modelo usando rótulos do treino, ou o de qualidade de dados para medir precisão.
Como não cair: Sem rótulos reais, monitore qualidade de dados. Precisão e recall exigem ground truth unido pelo ID da inferência.

⚠ **Alerta de prova:** Atribuir AccessDenied em bucket SSE-KMS a falta de endpoint de rede, versionamento ou isolamento de rede.
Como não cair: Problema de rede dá tempo esgotado, não acesso negado. Com s3:GetObject presente, falta kms:Decrypt na chave.

⚠ **Alerta de prova:** Tratar AccessDeniedException na inferência entre Regiões como falta de cota ou de streaming.
Como não cair: Falta de cota gera throttling. Acesso negado com IAM correto aponta para SCP bloqueando as Regiões de destino.

⚠ **Alerta de prova:** Usar tópico negado, Macie ou KMS para esconder telefone e e-mail nas respostas do chatbot.
Como não cair: Tópico negado bloqueia a resposta inteira e Macie analisa objetos no S3. Use filtro de informações sensíveis do Guardrails com ação de mascarar.

⚠ **Alerta de prova:** Ativar a varredura básica do ECR para pegar CVEs novas em pacotes de linguagem.
Como não cair: A básica não reavalia continuamente nem cobre linguagens. Use varredura aprimorada com Amazon Inspector; no build, Inspector com SBOM Generator e Scan API antes do push.

⚠ **Alerta de prova:** Aumentar a temperatura para conter injeção de prompt, loop de agentes ou roteamento errado.
Como não cair: Temperatura só aumenta aleatoriedade. Injeção pede filtro de ataques de prompt com input tags; loop pede limite de iterações e alarme nos traces.

⚠ **Alerta de prova:** Usar alerta por e-mail do Cost Explorer ou do Budgets para bloquear novos jobs ao estourar o teto.
Como não cair: Aviso não bloqueia. Use ação do AWS Budgets que aplica política IAM de negação ou SCP.

Como cai na prova

📝 **Como cai:** Problema em produção descrito pelo sintoma (lentidão, erro silencioso, resposta cortada, custo fora do padrão) pedindo a ferramenta de diagnóstico.
Como resolver: Defina o que precisa ser visto: métrica (CloudWatch), texto do log (filtro de métricas ou Logs Insights), cadeia de serviços (X-Ray), conteúdo de prompts (invocation logging), sessão de agente (AgentCore Observability), autoria de ação (CloudTrail), configuração (Config). Descarte serviços que medem outra coisa.

📝 **Como cai:** Reduzir custo de inferência mantendo qualidade.
Como resolver: Identifique o desperdício: prefixo repetido (cache de prompts), perguntas parecidas (cache semântico), mistura de perguntas fáceis e difíceis (prompt routing), volume grande sem pressa (inferência em lote), GPU ociosa (CPU ou inf2), uso estável (SageMaker Savings Plans), reindexação total (sincronização incremental).

📝 **Como cai:** Erro de acesso em job, endpoint ou chamada ao Bedrock.
Como resolver: Leia a mensagem: AccessDenied com SSE-KMS é kms:Decrypt; iam:PassRole é permitir PassRole na função específica com iam:PassedToService; acesso negado entre Regiões é SCP; tempo esgotado em sub-rede privada é falta de endpoint de VPC.

📝 **Como cai:** Exigência de segurança preventiva e centralizada.
Como resolver: Prefira controle que impede no momento da ação (política IAM com chave de condição, SCP, isolamento de rede, guardrail obrigatório) a alarmes, revisões manuais ou CloudTrail, que só registram depois.

📝 **Como cai:** Monitorar drift de um modelo em produção.
Como resolver: Pergunte se há rótulos: sem rótulos, qualidade de dados contra o baseline; com rótulos atrasados, ground truth pelo ID da inferência; diferença entre grupos, métrica pós-treinamento agendada; inferência em lote, etapa de drift no pipeline; FM, avaliações do Bedrock sobre os logs.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Medir efeito real no comportamento dos usuários com 10% do tráfego	**Production variants com pesos no mesmo endpoint (teste A/B)**
Métrica com variação por turno ou sazonalidade, sem limite fixo	**Alarme com detecção de anomalias do CloudWatch**
Filtrar e agregar milhões de linhas de log JSON sem exportar	**CloudWatch Logs Insights**
Descobrir em qual etapa da cadeia API Gateway, Lambda e endpoint está a lentidão	**AWS X-Ray**
Quem excluiu o endpoint, de qual IP e quando	**Evento DeleteEndpoint no histórico do CloudTrail**
Auditores exigem prova de que os logs não foram alterados	**Validação de integridade dos arquivos de log do CloudTrail**
Nenhuma aplicação pode chamar o Bedrock sem o guardrail corporativo	**Chave de condição bedrock:GuardrailIdentifier em política IAM**
Impedir criação de jobs de treinamento fora da VPC	**Deny em CreateTrainingJob com condição Null em sagemaker:VpcSubnets**
Contêiner de terceiros não pode fazer nenhuma chamada de rede de saída	**EnableNetworkIsolation**
Separar o custo de tokens de várias áreas que usam o mesmo modelo na mesma conta	**Application inference profiles com tags de alocação de custos**
Mesmo prompt de sistema de milhares de tokens em toda requisição	**Cache de prompts com checkpoint após o prefixo fixo**
Resposta do RAG traz prazos e valores que não estão nos trechos recuperados	**Contextual grounding check do Bedrock Guardrails**
Respostas terminam no meio da frase com ConverseStream	**stopReason igual a max_tokens**
Endpoint XGBoost em GPU com GPUUtilization perto de zero	**Migrar para instância de CPU, como ml.c7i**

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

- CloudTrail diz quem chamou, invocation logging guarda o que foi dito, Config avalia configuração, CloudWatch mede desempenho.
- SageMaker Savings Plans para SageMaker; endpoint de tempo real nunca usa Spot.
- SSE-KMS com chave do cliente exige kms:Decrypt para ler; SSE-S3 não segrega acesso.
- Controle preventivo vence detecção: chaves de condição IAM, SCP, isolamento de rede e guardrail obrigatório.
- Aplicações usam funções IAM com credenciais temporárias; chave de API de longo prazo do Bedrock só para exploração.

Essenciais para a prova (checklist)

☐ Sei quando usar teste A/B com production variants e quando usar teste de sombra.

☐ Sei ler ModelLatency, OverheadLatency, GPUUtilization e Invocation5XXErrors e onde ficam os logs do endpoint.

☐ Sei configurar alarmes com detecção de anomalias, filtros de métricas, Logs Insights, dashboards e X-Ray.

☐ Sei montar monitoramento de qualidade de dados, de qualidade do modelo com ground truth e de drift de viés.

☐ Sei monitorar FMs com métricas do Bedrock, invocation logging, observabilidade de IA generativa, InvocationsIntervened e stopReason.

☐ Sei reduzir custo com cache de prompts, cache semântico, prompt routing, inferência em lote e inference profiles com tags.

☐ Sei escolher SageMaker Savings Plans, Spot e famílias inf2, G, P e CPU pelo perfil da carga.

☐ Sei aplicar menor privilégio com funções por projeto, PassRole, permissions boundary, ABAC e chaves de condição do SageMaker e do Bedrock.

☐ Sei resolver AccessDenied com SSE-KMS e configurar isolamento de rede, endpoints de VPC e treino distribuído seguro.

☐ Sei usar CloudTrail, Config, Model Cards, Lake Formation, Inspector e Bedrock Guardrails para auditoria e proteção.

Questões de revisão do capítulo

R3.1 · Operação, monitoramento e segurança de soluções de ML e IA

Um conglomerado financeiro tem três áreas (cartões, seguros e investimentos) que usam o mesmo modelo do Amazon Bedrock na mesma conta. O time de FinOps precisa separar o custo de inferência por área no AWS Cost Explorer. O que deve ser feito?

A. Criar um perfil de inferência de aplicação por área, com tags de custo ativadas
B. Criar uma chave do AWS KMS por área para cifrar os prompts de cada uma delas
C. Criar um alias de modelo por área e dividir o custo total pelo número de áreas no fim do mês
D. Criar um guardrail por área e filtrar o custo pelo identificador do guardrail

R3.2 · Operação, monitoramento e segurança de soluções de ML e IA

A equipe de uma operadora quer uma visão consolidada das invocações de modelos do Amazon Bedrock, com contagem de tokens de entrada e saída, latência P90 e P99, erros e uma tabela com o conteúdo das requisições e respostas. Qual combinação atende com o menor esforço?

A. Ativar o Amazon Inspector e analisar os achados relacionados às chamadas de modelos
B. Ativar o model invocation logging e usar o painel Model Invocations do CloudWatch para IA generativa
C. Ativar o AWS Config e criar regras que registram cada chamada feita ao Amazon Bedrock
D. Ativar o AWS CloudTrail e consultar o conteúdo dos prompts nos eventos de InvokeModel

R3.3 · Operação, monitoramento e segurança de soluções de ML e IA

O número de erros 5XX de um endpoint do SageMaker AI que estima prazos de entrega aumentou de repente. A equipe precisa ver os rastreamentos de exceção gerados pelo código de inferência dentro do contêiner. Onde essas informações ficam disponíveis?

A. No CloudWatch Logs, no grupo /aws/sagemaker/Endpoints do endpoint
B. No mapa de serviços do AWS X-Ray sem instrumentação
C. No relatório de violações do Model Monitor
D. No histórico de eventos de gerenciamento do AWS CloudTrail

R3.4 · Operação, monitoramento e segurança de soluções de ML e IA

Um banco entrega aos auditores os logs do AWS CloudTrail que registram as ações sobre modelos, endpoints e dados de treinamento. Os auditores exigem uma forma de comprovar que os arquivos de log não foram alterados nem excluídos depois de entregues ao S3. O que deve ser ativado?

A. Ativar o versionamento do bucket e conceder escrita aos auditores
B. Ativar o Amazon Macie para classificar os arquivos de log da trilha
C. Ativar a validação de integridade dos arquivos de log da trilha
D. Ativar o registro de invocações do Bedrock no mesmo bucket da trilha

Respostas comentadas

R3.1. A) Criar um perfil de inferência de aplicação por área, com tags de custo ativadas

Por quê: Perfis de inferência de aplicação do Amazon Bedrock permitem rastrear uso e custo por aplicação ou equipe por meio de tags, que precisam ser ativadas como tags de alocação de custos (tarefa 4.2).

✗ B) Incorreta. Chaves de criptografia não separam o custo de inferência.

✗ C) Incorreta. O rateio igual não reflete o consumo real de cada área.

✗ D) Incorreta. Guardrails controlam conteúdo e não servem para alocar o custo das invocações.

> **Dica de prova:** Custo do Bedrock por equipe na mesma conta: perfil de inferência de aplicação com tags.

R3.2. B) Ativar o model invocation logging e usar o painel Model Invocations do CloudWatch para IA generativa

Por quê: A observabilidade de IA generativa do Amazon CloudWatch oferece painéis prontos de invocações de modelos (tokens, latência, erros), complementados pelo invocation logging do Bedrock (tarefa 4.1).

✗ A) Incorreta. O Inspector trata de vulnerabilidades.

✗ C) Incorreta. O Config registra configurações de recursos, não invocações.

✗ D) Incorreta. O CloudTrail registra as chamadas de API, mas não o conteúdo dos prompts e das respostas.

> **Dica de prova:** Painel pronto de tokens e latência de FMs: CloudWatch generative AI observability.

R3.3. A) No CloudWatch Logs, no grupo /aws/sagemaker/Endpoints do endpoint

Por quê: Os contêineres de endpoints do SageMaker AI enviam a saída padrão e de erro ao CloudWatch Logs, no grupo /aws/sagemaker/Endpoints seguido do nome do endpoint, onde ficam os rastreamentos de exceção do código de inferência (tarefa 4.1).

✗ B) Incorreta. Sem instrumentação, o X-Ray não exibe exceções internas do contêiner.

✗ C) Incorreta. Mostra violações de qualidade, não erros de execução.

✗ D) Incorreta. Registra chamadas de API, não exceções do código.

> **Dica de prova:** Erro dentro do contêiner do endpoint: CloudWatch Logs em /aws/sagemaker/Endpoints.

R3.4. C) Ativar a validação de integridade dos arquivos de log da trilha

Por quê: A validação de integridade dos arquivos de log do CloudTrail usa hashes e assinaturas digitais em arquivos de resumo para comprovar que os logs não foram modificados, excluídos ou forjados (tarefa 4.3).

✗ A) Incorreta. O versionamento ajuda a recuperar objetos, mas não comprova integridade, e dar escrita aos auditores cria risco.

✗ B) Incorreta. O Macie procura dados sensíveis e não verifica alterações em logs.

✗ D) Incorreta. Registra prompts e respostas e não comprova a integridade dos logs do CloudTrail.

> **Dica de prova:** Provar que o log do CloudTrail não foi adulterado: validação de integridade.

04

CAPÍTULO 4 · 28% DA PROVA

Preparação de dados para ML e IA

Aqui eu preciso dominar o caminho do dado até o treino: onde guardar (classes do S3, Feature Store, bancos vetoriais), como ingerir (Kinesis, Firehose, Flink, DataSync, Glue), como transformar (Glue, DataBrew, EMR, Data Wrangler, Lambda) e como validar qualidade e viés antes de treinar. É o domínio de maior peso (28%) e quase sempre vem como cenário com uma restrição decisiva: sem código, sem servidor, leitura em milissegundos, menor custo ou sem afetar a produção. Também cai muita engenharia de features (one-hot, log, binning, padronização, feature splitting), divisão de dados sem vazamento e tratamento de classe rara. Na parte de IA generativa, preciso saber preparar documentos para RAG (chunking, metadados, embeddings) e dados para fine-tuning, pré-treinamento contínuo e destilação. Onde mais erro: confundir quem descobre PII (Macie) com quem mascara (Glue e DataBrew), misturar métricas de viés pré e pós-treinamento e trocar as soluções de vazão do Kinesis (chave de partição, enhanced fan-out, modo sob demanda, retenção).

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 8 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Formatos colunares e particionamento

Parquet e ORC permitem ler só as colunas usadas, e particionar pelas chaves dos filtros (como data) reduz o volume varrido no Athena e no Spark. Compactar CSV com gzip ou trocar para JSON não resolve, porque continuam orientados a linhas. Milhões de arquivos de poucos KB degradam Spark e Athena: a correção é consolidar em arquivos Parquet maiores, de centenas de MB. No Athena com SerDe de JSON, cada registro precisa ficar em uma única linha.

Amazon Data Firehose

Entrega streams no S3 e oferece, sem código, conversão de formato de registro (JSON para Parquet ou ORC) usando o esquema de uma tabela do AWS Glue Data Catalog e particionamento dinâmico por chaves dos registros. Transformações leves por registro (remover campos, padronizar unidade) são feitas por uma função Lambda de transformação. Não faz junção com tabelas de banco, não calcula agregações com estado e não tem o Feature Store como destino.

Kinesis Data Streams: vazão e retenção

Cada shard aceita até 1 MB/s ou 1.000 registros/s de gravação, e a chave de partição define o shard: chave de baixa cardinalidade cria shard quente mesmo com capacidade sobrando. Sem enhanced fan-out, os consumidores dividem os 2 MB/s de leitura de cada shard; com ele, cada consumidor registrado ganha vazão dedicada. A retenção padrão é de 24 horas e vai até 365 dias, o que permite reprocessar após falha de consumidor. Para picos imprevisíveis sem gerenciar shards, use o modo sob demanda.

Amazon Managed Service for Apache Flink

Executa aplicações Flink sem servidores para gerenciar, mantendo estado e calculando agregações em janelas (tumbling, sliding, session) sobre streams do Kinesis ou do Kafka, como média móvel de 5 minutos ou visualizações por minuto. O Lambda atende melhor transformações simples por registro. Quando um consumidor Lambda fica atrás do stream (IteratorAge subindo), aumente o ParallelizationFactor do mapeamento de origem de eventos ou o número de shards.

SageMaker Feature Store

Um feature group com online e offline store grava cada registro nos dois: o online guarda o valor mais recente para leitura em milissegundos na inferência e o offline acumula o histórico para treino, evitando divergência entre treino e inferência. PutRecord ingere dados de streaming; para bilhões de linhas em lote, use o conector Spark (por exemplo no EMR). Consultas point-in-time no offline store usando o event time evitam vazamento, e TtlDuration expira registros antigos do online store.

AWS Glue para ETL

Job bookmarks registram o que já foi processado e permitem processamento incremental de arquivos novos no S3 e em fontes JDBC. Crawlers só inferem esquema e partições, sem alterar nem validar dados. Um job do Glue com conexão JDBC junta dados do RDS com arquivos do S3 e grava em Parquet, sem servidor. A transformação de ML FindMatches casa registros sem identificador comum e com grafias diferentes, e o Glue Schema Registry com Avro controla a evolução de esquema em streaming.

AWS Glue Data Quality

Usa a Data Quality Definition Language (DQDL) com regras como Completeness, ColumnValues, IsUnique, IsComplete e ColumnLength; dentro do job de ETL, o resultado pode interromper a carga de dados ruins. Também recomenda regras a partir do perfil das tabelas do Data Catalog e executa avaliações agendadas sem job de ETL. A detecção de anomalias aprende o padrão histórico das execuções e aponta quedas de volume ou valores fora do comum sem limites fixos.

AWS Glue DataBrew

Ferramenta visual de preparação sem código, com mais de 250 transformações prontas; os passos ficam em receitas versionadas e reutilizáveis, executadas por jobs sob demanda ou agendados. Rulesets de qualidade associados a um job de perfil validam regras automaticamente. Tem etapas para outliers (sinalizar, remover, substituir ou reescalar por escore Z) e para PII (mascaramento, hashing, criptografia determinística, substituição). Processa conjuntos em lote, nunca registro a registro em streaming.

SageMaker Data Wrangler

Preparação visual dentro do SageMaker AI (acessível pelo SageMaker Canvas), com centenas de transformações, relatório de qualidade e insights dos dados e exportação do fluxo para jobs de processamento ou para o SageMaker Pipelines. A transformação Balance data oferece subamostragem aleatória, sobreamostragem aleatória e SMOTE, que cria exemplos sintéticos por interpolação entre vizinhos. Também oferece escalonamento robusto (mediana e IQR).

Armazenamento para treino

S3 Express One Zone usa buckets de diretório em uma única AZ com latência de milissegundos de um dígito, para dados muito acessados e recriáveis. FSx for Lustre vinculado ao S3 dá alta vazão compartilhada entre jobs e elimina a cópia inicial do modo File; o modo Pipe com RecordIO-protobuf transmite os dados durante o treino dos algoritmos integrados. Object Lock em modo de conformidade impede alteração e exclusão até pelo root, e regras de ciclo de vida para o Glacier Deep Archive minimizam custo em retenção longa com horas de espera aceitas. Erro 503 Slow Down se resolve espalhando objetos por mais prefixos.

Engenharia de features

One-hot para variáveis nominais de poucas categorias (códigos inteiros criam ordem falsa); transformação log para cauda longa à direita como renda; binning por quantis para faixas interpretáveis com frequências parecidas; padronização z-score para algoritmos de distância como k-NN e k-means; escalonamento robusto quando há outliers legítimos que comprimiriam o min-max. Feature splitting extrai hora do dia, dia da semana e mês de um timestamp. Toda transformação é ajustada só no treino e reaplicada com os mesmos parâmetros em validação, teste e inferência.

Divisão de dados sem vazamento

Embaralhe antes de dividir quando não há dependência temporal, inclusive quando os dados vêm ordenados por região ou por classe. Use divisão estratificada para manter a classe rara em todos os conjuntos, divisão por grupo (por cliente) quando a mesma entidade tem muitos registros e divisão cronológica quando os dados ou atributos mudam no tempo. Sobreamostragem, SMOTE ou cópias só depois da divisão, aplicadas ao treino; validação não deve ser reaproveitada como teste final.

Classes desbalanceadas

Com classe rara, a acurácia engana (99,5% de acerto sem detectar nenhum defeito); avalie com recall, F1 e AUC-PR. No XGBoost, ajuste scale_pos_weight perto da razão entre negativos e positivos sem alterar o dataset. Em imagens, aumento de dados da classe rara (rotação, espelhamento, brilho) e pesos de classe na função de perda; em texto, paráfrase e retrotradução com terminologia personalizada do Amazon Translate. Subamostragem descarta dados das classes majoritárias.

Métricas de viés pré-treinamento

Analisam dados e rótulos antes de existir modelo: CI (Class Imbalance) mede se um grupo tem muito menos amostras, DPL (Difference in Proportions of Labels) mede diferença na proporção de rótulos positivos entre facetas, e KL e Jensen-Shannon comparam distribuições de rótulos, inclusive multiclasse. DI, AD e DPPL são pós-treinamento, pois dependem de previsões. Metadados como região servem de faceta para texto e imagem, e a AWS recomenda calcular as fórmulas com pandas e scikit-learn, por exemplo em um job do SageMaker Processing.

Dados sensíveis: descobrir e tratar

O Amazon Macie descobre e classifica dados sensíveis no S3, mas não remove nem mascara. O tratamento vem da transformação Detect Sensitive Data do AWS Glue (com redação) ou de receitas de PII do DataBrew. Para laudos clínicos, o Comprehend Medical (DetectPHI) encontra PHI no texto livre. Anonimizar exige tratar também quase identificadores (data de nascimento, CEP) com faixas e generalização; hash isolado é só pseudonimização, e hash ou criptografia determinística preserva a chave de junção.

Bancos vetoriais

Aurora ou RDS PostgreSQL com pgvector (índices HNSW ou IVFFlat) mantém vetores e dados relacionais juntos para filtros SQL sem operar outro banco. Amazon S3 Vectors (vector buckets e vector indexes) é a opção de menor custo para centenas de milhões de vetores com consultas esporádicas e latência abaixo de um segundo. O OpenSearch Service pode descarregar vetores para o S3 Vectors com o mecanismo s3vector, mantendo os campos de texto no cluster. Neptune Analytics habilita GraphRAG para perguntas que ligam entidades entre documentos.

Embeddings e chunking para RAG

Documentos e consultas precisam ser codificados pelo mesmo modelo de embeddings; trocar o modelo exige regerar os vetores e reindexar. O Titan Text Embeddings V2 aceita 256, 512 ou 1.024 dimensões e o parâmetro normalize (com vetores normalizados, produto interno e cosseno dão a mesma ordenação); o Cohere Embed pede input_type search_document no índice e search_query nas perguntas; o Titan Multimodal Embeddings permite buscar imagens por texto. Chunking semântico corta onde o assunto muda, hierárquico busca nos filhos e entrega os pais, e arquivos .metadata.json ou uma Lambda de transformação na ingestão adicionam metadados filtráveis.

Dados para customizar FMs

Fine-tuning no Bedrock usa JSON Lines no S3, um exemplo por linha, no formato exigido pelo modelo (mensagens do Converse no Amazon Nova); pré-treinamento contínuo usa texto bruto limpo e sem rótulos; destilação pode usar os logs de invocação do Bedrock como fonte, desde que o invocation logging esteja ativo. Antes do job, valide JSON, campos obrigatórios e limite de tokens, aplique regras do Glue Data Quality, faça triagem de conteúdo com ApplyGuardrail, com a detecção de toxicidade do Comprehend ou com DetectModerationLabels do Rekognition para imagens, e trate quase duplicatas com MinHash e LSH.

Macetes para memorizar

PII

Macie é o detetive (acha); Glue e DataBrew são a faxina (mascaram, redigem, aplicam hash).

Kinesis

Gravação travada é chave; leitura disputada é fan-out; pico imprevisível é sob demanda; reler o passado é retenção.

Métricas de viés

PRÉ olha dados: CI conta gente, DPL conta rótulo, KL e JS comparam distribuições. PÓS olha previsões: DI, AD, DPPL.

Escalonamento de features

Distância pede z-score; cauda longa pede log; outlier legítimo pede robusto; categoria sem ordem pede one-hot; faixa interpretável pede binning por quantis.

Divisão de dados

Tem tempo? corta na data. Cliente repetido? corta por cliente. Classe rara? estratifica. Fora isso, embaralha.

Feature Store

Online é o agora (milissegundos); offline é a memória (histórico e point-in-time para treino).

Chunking hierárquico

Filho para achar, pai para responder.

Embeddings

Trocou o modelo de embeddings, reindexa o acervo inteiro.

Tabela de memorização: não confunda

Quando é...	...e quando é
Amazon Macie É preciso descobrir em quais buckets e objetos do S3 existem dados pessoais antes de liberar um data lake.	**Glue Detect Sensitive Data e DataBrew** É preciso redigir, mascarar, aplicar hash ou criptografia determinística nos dados pessoais antes do treino.
Chave de partição de alta cardinalidade Os produtores recebem ProvisionedThroughputExceededException na gravação e um único shard concentra os registros.	**Enhanced fan-out** Vários consumidores leem o mesmo stream e disputam os 2 MB/s de leitura por shard.
Managed Service for Apache Flink A transformação exige estado e janelas de tempo, como média móvel ou contagem por minuto.	**Firehose com Lambda de transformação** Basta ajustar cada registro em trânsito (remover campos, padronizar unidade) antes de gravar no S3.
Amazon EMR A equipe precisa controlar versão do Spark, tipos de instância, bibliotecas nativas e configurações do cluster.	**Amazon EMR Serverless** Os scripts Spark já validados no runtime do EMR rodam poucas vezes e a equipe não quer provisionar nem gerenciar cluster, pagando pelo uso.

Object Lock em modo de conformidade Nenhum usuário, nem o root, pode alterar ou excluir os dados até o fim da retenção.	**Object Lock em modo de governança** Usuários com s3:BypassGovernanceRetention podem remover a proteção quando necessário.
CI (Class Imbalance) A pergunta é se um grupo está sub-representado, com muito menos amostras que o outro.	**DPL (Difference in Proportions of Labels)** A pergunta é se a proporção de rótulos positivos (aprovado, bolsa concedida) difere entre os grupos nos dados históricos.
Amazon S3 Vectors Há centenas de milhões de vetores, consultas pouco frequentes, latência abaixo de um segundo basta e a prioridade é o menor custo.	**Amazon OpenSearch Service** É preciso busca híbrida e baixa latência com alto volume de consultas, aceitando o custo de manter nós.

Números que caem

Item	Valor
Peso do domínio no exame	**28%**
Gravação por shard do Kinesis Data Streams	**até 1 MB/s ou 1.000 registros/s**
Leitura por shard sem enhanced fan-out	**2 MB/s divididos entre os consumidores**
Retenção do Kinesis Data Streams	**24 horas por padrão, até 365 dias**
Requisições por prefixo particionado no S3	**ao menos 3.500 PUT/COPY/POST/DELETE e 5.500 GET/HEAD por segundo**
Tamanho máximo de item do DynamoDB	**400 KB**
DynamoDB Streams	**guarda as alterações das últimas 24 horas**
Dimensões do Titan Text Embeddings V2	**256, 512 ou 1.024**
Transformações prontas do AWS Glue DataBrew	**mais de 250**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Compactar os CSV com gzip ou converter para JSON para baratear o Athena.
Como não cair: Compressão reduz bytes, mas o Athena continua lendo linhas inteiras. O que reduz a varredura é formato colunar (Parquet ou ORC) com particionamento pelas chaves de filtro.

⚠ **Alerta de prova:** Dobrar o número de shards para acabar com o throttling de gravação, mantendo a mesma chave de partição.
Como não cair: Com a mesma chave de baixa cardinalidade, os registros continuam indo para um único shard. Troque por uma chave de alta cardinalidade, como o identificador do veículo.

⚠ **Alerta de prova:** Ativar enhanced fan-out ou aumentar a retenção para resolver erro de gravação ou atraso do consumidor.
Como não cair: Fan-out só melhora leitura; retenção só define por quanto tempo o dado fica disponível. IteratorAge alto com Lambda pede ParallelizationFactor maior.

⚠ **Alerta de prova:** Usar o Amazon Macie para apagar dados pessoais dos objetos.
Como não cair: O Macie descobre e classifica; quem remove, mascara ou aplica hash é o Glue (Detect Sensitive Data) ou o DataBrew.

⚠ **Alerta de prova:** Criptografar com SSE-KMS achando que o modelo deixará de ver os dados pessoais, ou enviar a chave junto ao parceiro externo.
Como não cair: O treino lê os dados já descriptografados, e quem recebe a chave vê tudo. Proteção contra exposição no treino é mascaramento, redação ou anonimização.

⚠ **Alerta de prova:** Trocar o nome por um hash e manter data de nascimento e CEP exatos para considerar a base anônima.
Como não cair: Quase identificadores combinados reidentificam pessoas. Generalize idade em faixas e CEP em região além de remover identificadores diretos.

⚠ **Alerta de prova:** Calcular mínimo e máximo da normalização sobre o conjunto completo, incluindo teste.
Como não cair: Ajuste a transformação só no treino e reaplique os mesmos parâmetros nos demais conjuntos e na inferência; caso contrário há vazamento e métricas otimistas.

⚠ **Alerta de prova:** Aplicar SMOTE ou duplicar a classe rara antes de separar treino e teste.
Como não cair: Exemplos derivados do teste vazam para o treino e inflam as métricas. Divida primeiro e balanceie só o treino.

⚠ **Alerta de prova:** Embaralhar de novo com outra semente quando o modelo vai bem na validação e mal em produção com dados temporais ou com vários registros por cliente.
Como não cair: Outra semente mantém o vazamento. Divida por tempo (treino no passado, validação no período recente) ou por cliente.

⚠ **Alerta de prova:** Escolher Disparate Impact (DI) ou Accuracy Difference (AD) para medir viés antes do treino.
Como não cair: DI, AD e DPPL dependem de previsões de um modelo treinado. Antes do treino, use CI, DPL, KL ou JS.

⚠ **Alerta de prova:** Montar o conjunto de treino consultando o online store ou ler o offline store via Athena na inferência.
Como não cair: O online store guarda só o valor mais recente (causa vazamento no treino) e o Athena leva segundos. Treino usa consultas point-in-time no offline store; inferência lê o online store.

⚠ **Alerta de prova:** Normalizar os vetores das perguntas ou trocar cosseno por euclidiana depois de mudar o modelo de embeddings.
Como não cair: Vetores de modelos diferentes vivem em espaços incompatíveis. Regere os embeddings do acervo com o novo modelo e reindexe.

⚠ **Alerta de prova:** Manter remoção de stopwords, stemming e corte de palavras raras antes de ajustar um transformer.
Como não cair: Limpezas agressivas eliminam negações como 'não' e contexto. Use o texto natural com o tokenizador de subpalavras do próprio modelo.

⚠ **Alerta de prova:** Imputar a média ou zero em uma coluna de renda assimétrica com ausentes.
Como não cair: A média é puxada pelos extremos e zero é um valor real, diferente de ausente. Em variáveis assimétricas, impute a mediana.

Como cai na prova

📝 **Como cai:** Pipeline de dados com uma restrição forte: sem código, sem servidor, sem gerenciar cluster ou mantendo Spark existente.
Como resolver: Marque a restrição antes de olhar as opções. Sem código e com receitas: DataBrew. ETL Spark sem servidor e incremental: Glue com bookmarks. Spark existente com controle fino: EMR; sem gerenciar cluster: EMR Serverless. Ajuste por registro em stream: Firehose com Lambda; janela com estado: Flink.

📝 **Como cai:** Erro de ingestão ou desempenho com mensagem específica (ProvisionedThroughputExceededException, 503 Slow Down, IteratorAge, download lento no modo File).
Como resolver: Identifique se o gargalo é gravação, leitura, consumo ou cópia inicial. Gravação no Kinesis: chave de partição ou modo sob demanda. Leitura: enhanced fan-out. Consumo Lambda: ParallelizationFactor. 503 no S3: prefixos. Download do treino: FSx for Lustre ou modo Pipe.

📝 **Como cai:** Modelo excelente na validação e fraco em produção.
Como resolver: Procure vazamento: divisão aleatória em dado temporal, registros do mesmo cliente nos dois lados, normalização calculada com o teste, SMOTE antes da divisão ou features posteriores ao evento rotulado. A resposta certa corrige a origem do vazamento, não troca a semente.

📝 **Como cai:** Medir ou tratar viés e desbalanceamento antes do treino.
Como resolver: Pergunte se já existe modelo. Não existe: CI para representação, DPL para rótulos, KL ou JS para distribuição multiclasse. Para corrigir: estratificar, aumentar dados, SMOTE no treino, scale_pos_weight no XGBoost, e avaliar por recall, F1 e AUC-PR.

📝 **Como cai:** Preparar documentos para uma aplicação de RAG.
Como resolver: Converta mídia em texto (Transcribe para áudio, Textract para documento digitalizado), escolha o chunking pelo tipo de texto (semântico para texto corrido, hierárquico para precisão e contexto), adicione metadados para filtros e use o mesmo modelo de embeddings em índice e consulta.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Consulta do Athena usa poucas colunas e um intervalo de datas, mas varre tudo	**Converter para Parquet ou ORC particionado pela data**
HTTP 503 Slow Down ao gravar milhões de objetos no mesmo prefixo	**Distribuir os objetos entre vários prefixos**
Stream JSON precisa chegar ao S3 em Parquet sem escrever código	**Conversão de formato de registro do Data Firehose com esquema do Glue Data Catalog**
Agregação contínua em janela de minutos mantendo estado sobre Kinesis ou Kafka	**Amazon Managed Service for Apache Flink**
Tráfego do stream se multiplica de forma imprevisível e a equipe não quer ajustar shards	**Kinesis Data Streams no modo de capacidade sob demanda**
Feature lida em milissegundos na inferência e com histórico para retreino	**Feature group com online store e offline store**
Job do Glue reprocessa todo o histórico a cada execução	**Job bookmarks do AWS Glue**
Unificar cadastros sem identificador comum e com grafias diferentes	**Transformação de ML FindMatches do AWS Glue**
Extrair dados do RDS de produção sem degradar a instância primária	**Réplica de leitura ou exportação de snapshot para o S3 em Parquet**

Snapshot completo do DynamoDB no S3 sem consumir capacidade de leitura	**Exportação nativa do DynamoDB para o S3 com recuperação point-in-time**
Copiar imagens de um NAS local via NFS para o S3 de forma incremental e agendada	**AWS DataSync**
Dados de treino não podem ser alterados nem excluídos por anos, nem pelo root	**S3 Object Lock em modo de conformidade**
Vários jobs leem o mesmo dataset de TB por muitas épocas e o download inicial atrasa	**Amazon FSx for Lustre vinculado ao bucket do S3**
Perguntas que exigem relacionar entidades espalhadas em muitos documentos	**GraphRAG com Amazon Neptune Analytics na base de conhecimento**

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

- Toda transformação, balanceamento ou cópia é ajustada no treino depois da divisão; nada calculado com o teste.
- Macie só descobre PII; Glue e DataBrew tratam; Comprehend Medical acha PHI em texto clínico.
- Online store serve a inferência; offline store com point-in-time monta o treino sem vazamento.
- Antes do treino: CI, DPL, KL, JS. DI, AD e DPPL só existem com previsões.
- Embeddings de índice e consulta vêm sempre do mesmo modelo; trocou, reindexa.

Essenciais para a prova (checklist)

☐ Sei quando usar Parquet ou ORC com partições e por que gzip em CSV não reduz a varredura do Athena.

☐ Sei diferenciar chave de partição, enhanced fan-out, modo sob demanda, retenção e ParallelizationFactor no Kinesis.

☐ Sei escolher entre Firehose com Lambda, Managed Service for Apache Flink, Glue, DataBrew, EMR e EMR Serverless pela restrição do cenário.

☐ Sei configurar um feature group com online e offline store e montar treino com consulta point-in-time.

☐ Sei aplicar one-hot, log, binning por quantis, z-score, escalonamento robusto e feature splitting no caso certo.

☐ Sei escolher divisão embaralhada, estratificada, por grupo ou cronológica para evitar vazamento.

☐ Sei tratar desbalanceamento com SMOTE, aumento de dados, pesos de classe e scale_pos_weight, avaliando por recall, F1 e AUC-PR.

☐ Sei calcular e interpretar CI, DPL, KL e JS antes do treino e separá-las das métricas pós-treinamento.

☐ Sei descobrir PII com Macie e tratá-la com Glue, DataBrew ou Comprehend Medical, incluindo quase identificadores.

☐ Sei escolher banco vetorial, estratégia de chunking e parâmetros de embeddings (dimensão, normalize, input_type) para RAG.

Questões de revisão do capítulo

R4.1 · Preparação de dados para ML e IA

Uma equipe de engenharia de rede prepara medições de latência para treinar um modelo. Falhas de coleta geraram picos absurdos de milhões de milissegundos. Ela quer detectar e tratar esses outliers por meio de uma ferramenta visual, sem escrever código. Qual opção atende?

A. Aplicar o Amazon Macie para classificar as medições anômalas da rede
B. Aplicar as etapas de outliers do AWS Glue DataBrew, como a de reescala por escore Z
C. Aplicar a codificação one-hot do DataBrew às colunas de latência medida
D. Aplicar normalização min-max mantendo os valores extremos na coluna

R4.2 · Preparação de dados para ML e IA

Uma transportadora envia leituras de sensores de caminhões ao Amazon Data Firehose, que entrega os dados no Amazon S3. Antes da gravação, cada registro precisa ter campos internos removidos e a unidade de medida padronizada. A empresa não quer gerenciar servidores nem clusters. Qual solução atende a esse requisito?

A. Criar um cluster do Amazon EMR para ler e corrigir os arquivos já gravados no S3
B. Agendar um crawler do AWS Glue sobre o destino no S3
C. Configurar a transformação de dados do Firehose com uma função AWS Lambda
D. Executar um job do AWS Glue DataBrew a cada registro recebido

R4.3 · Preparação de dados para ML e IA

Uma rede varejista executa repetidamente jobs de treinamento curtos que leem os mesmos arquivos de features centenas de vezes por hora. As instâncias ficam em uma única zona de disponibilidade, a latência de acesso aos dados é o gargalo e os dados podem ser regenerados a partir da origem se houver perda. Qual opção de armazenamento é mais adequada?

A. S3 Glacier Deep Archive com recuperação em lote
B. S3 Standard-IA com regra de ciclo de vida para os arquivos de features
C. S3 Express One Zone em um bucket de diretório na mesma AZ
D. S3 Glacier Instant Retrieval no bucket atual

R4.4 · Preparação de dados para ML e IA

Uma empresa de logística quer prever o tempo de carregamento de caminhões em seus centros de distribuição. A única informação temporal é um timestamp completo de chegada, e a análise mostra que o tempo varia muito conforme a hora do dia e o dia da semana. Qual técnica de engenharia de features é mais adequada?

A. Extrair do timestamp a hora do dia e o dia da semana como features
B. Remover o timestamp para reduzir a dimensionalidade
C. Aplicar one-hot encoding ao valor completo do timestamp
D. Usar o timestamp convertido em número inteiro de segundos desde 1970

Respostas comentadas

R4.1. B) Aplicar as etapas de outliers do AWS Glue DataBrew, como a de reescala por escore Z

Por quê: A limpeza de dados inclui detectar e tratar outliers; o AWS Glue DataBrew oferece etapas de receita para sinalizar, remover, substituir ou reescalar outliers (tarefa 1.3).

✗ A) Incorreta. O Macie identifica dados sensíveis, não anomalias numéricas.
✗ C) Incorreta. One-hot é para variáveis categóricas.
✗ D) Incorreta. Com os picos mantidos, a normalização comprime os valores normais em uma faixa estreita.

> **Dica de prova:** Outliers sem código: receitas do DataBrew.

R4.2. C) Configurar a transformação de dados do Firehose com uma função AWS Lambda

Por quê: O Amazon Data Firehose pode invocar uma função Lambda para transformar registros em trânsito, como filtrar campos e padronizar valores, sem gerenciar infraestrutura (tarefa 1.2).

✗ A) Incorreta. Exige gerenciar clusters e só transforma depois da gravação.
✗ B) Incorreta. O crawler cataloga esquema; não altera registros.
✗ D) Incorreta. O DataBrew processa conjuntos em lote, não registro a registro em streaming.

> **Dica de prova:** Transformação leve por registro no Firehose: Lambda de transformação.

R4.3. C) S3 Express One Zone em um bucket de diretório na mesma AZ

Por quê: O S3 Express One Zone armazena dados em buckets de diretório em uma única zona de disponibilidade, com latência de milissegundos de um dígito, adequado a dados muito acessados por computação na mesma zona e que podem ser recriados (tarefa 1.1).

✗ A) Incorreta. A recuperação leva horas.
✗ B) Incorreta. É voltado a acesso infrequente e cobra por recuperação, o que encarece leituras repetidas.
✗ D) Incorreta. É projetado para dados raramente acessados e cobra por recuperação.

> **Dica de prova:** Latência mínima no S3, mesma AZ e dado recriável: S3 Express One Zone.

R4.4. A) Extrair do timestamp a hora do dia e o dia da semana como features

Por quê: Feature splitting decompõe uma variável em partes mais informativas, como hora, dia da semana e mês a partir de um timestamp, revelando padrões sazonais para o modelo (tarefa 1.2).

✗ B) Incorreta. Descarta justamente a informação que explica a variação.

✗ C) Incorreta. Gera uma coluna por instante, sem generalização.

✗ D) Incorreta. Cria uma feature crescente que não captura a sazonalidade diária ou semanal.

> **Dica de prova:** Timestamp com padrão por hora e dia: dividir em hora e dia da semana.

DIA DA PROVA

Estratégia de prova

O MLA-C02 é quase todo cenário: uma empresa com um problema concreto e uma restrição que decide a resposta, como menor esforço, sem servidor, sem código, sem afetar clientes, menor custo ou leitura em milissegundos. As alternativas erradas raramente são absurdas: são serviços reais da AWS usados fora do que fazem (Config para medir latência, Macie para mascarar dados, CloudTrail para ler prompts, serverless com GPU). O exame cobra ML tradicional e IA generativa, com SageMaker AI e Amazon Bedrock lado a lado, incluindo agentes no AgentCore, RAG e avaliação de FMs. A resposta certa costuma ser o recurso gerenciado e específico para o problema, aplicado de forma preventiva e automatizada, em vez de processos manuais, scripts próprios ou soluções genéricas.

Como responder

✓ Conheça o formato: 65 questões pontuadas (o exame real tem mais 20 não pontuadas, que não são identificadas), 170 minutos, pontuação de 100 a 1000 com mínimo de 720, em múltipla escolha e múltipla resposta. Como as 20 não pontuadas também ocupam os 170 minutos, controle o tempo contando com todas as questões.

✓ Responda todas: questão em branco conta como errada e não há penalidade por chute.

✓ Antes de ler as opções, sublinhe mentalmente a restrição decisiva do enunciado (menor esforço, sem código, sem servidor, sem afetar clientes, menor custo, tempo real) e use-a para eliminar alternativas.

✓ Elimine primeiro os serviços de outra categoria: Config avalia configuração, CloudTrail registra chamadas, Macie descobre dados sensíveis, Inspector procura vulnerabilidades, Trusted Advisor e Compute Optimizer só recomendam.

✓ Em múltipla resposta, procure duas ações complementares que juntas cumprem o requisito (detectar e alertar, tratar dentro do fluxo e notificar, permissão no S3 e no KMS) e confira se marcou exatamente a quantidade pedida.

✓ Desconfie de opções que adicionam processo manual, revisão periódica, planilha ou e-mail quando o enunciado pede automação ou controle preventivo.

✓ Distribua a revisão pelos pesos: Preparação de dados vale 28% e os outros três domínios 24% cada; a pontuação é compensatória, então basta passar no total.

✓ Quando ficar entre duas opções parecidas, releia apenas a restrição do enunciado e fique com a que a cumpre por inteiro, com o recurso gerenciado e específico para o problema.

Erros que derrubam candidatos

✗ Escolher serviço que existe mas mede outra coisa, como Config para latência, CloudTrail para conteúdo de prompts ou Macie para mascarar PII.

✗ Esquecer limites que eliminam opções: serverless do SageMaker AI sem GPU, Lambda com 15 minutos, multicontêiner com até 15 contêineres, item do DynamoDB com 400 KB.

✗ Confundir pares próximos: A/B x sombra, multimodelo x multicontêiner, pipeline de inferência x SageMaker Pipelines, BLEU x ROUGE, CI x DPL.

✗ Aceitar métricas infladas por vazamento: divisão aleatória em dado temporal, normalização com o teste, SMOTE antes da divisão.

✗ Preferir fine-tuning para dados que mudam todo dia, quando o caso pede RAG com citação de fonte.

✗ Trocar controle preventivo por detecção: alarme ou CloudTrail no lugar de política IAM com chave de condição ou SCP.

✗ Escolher o maior modelo ou mais capacidade para resolver custo e latência, quando o caso pede modelo menor, cache ou lote.

Na semana da prova

- ☐ Refaça as provas práticas por domínio, começando por Preparação de dados (28%), e anote cada par de serviços que confundiu.
- ☐ Revise as seções de pegadinhas e gatilhos de cada capítulo até reconhecer a restrição decisiva só pelo enunciado.
- ☐ Memorize os números que eliminam opções: limites de shard do Kinesis, retenção, 400 KB do DynamoDB, 15 minutos do Lambda, assíncrona com 1 GB e uma hora, porta 8080, 15 contêineres.
- ☐ Faça uma revisão final dos pares IA generativa: RAG x fine-tuning x CPT x destilação, cache de prompts x cache semântico, MCP x A2A, avaliações automáticas x LLM-as-a-judge x humanas.
- ☐ Na véspera, revise só os itens 'lembrar' dos quatro capítulos e descanse; no dia, controle o tempo para chegar ao fim com margem para reler as questões em que ficou em dúvida.

REVISÃO FINAL

Folha de cola

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

1. Desenvolvimento de modelos de ML e de modelos de base (FM)

‣ Fraude não detectada custa muito mais que um alerta falso → **Priorizar o recall, monitorando a precisão**

‣ Errar por 60 dias é muito pior que errar várias vezes por 5 → **RMSE**

‣ Algumas centenas de fotos rotuladas de objetos próprios e nenhuma equipe de ML → **Amazon Rekognition Custom Labels**

‣ Faturas com layouts variados; formulários com chave e valor e tabelas → **Textract AnalyzeExpense; Textract AnalyzeDocument com FORMS e TABLES**

‣ Transcrição erra nomes de fundos e siglas do setor → **Amazon Transcribe com vocabulário personalizado**

⚠ Isso é underfitting; regularização combate overfitting. Treine por mais épocas ou aumente a capacidade do modelo.

⚠ Taxa alta é a causa típica. Reduza a taxa, use agendamento, warmup e gradient clipping.

⚠ Cabe na GPU e o problema é volume: paralelismo de dados. Não cabe: paralelismo de modelo ou dados fragmentados, pois a cópia completa em cada GPU não cabe.

2. Implantação e orquestração de fluxos de trabalho de ML e IA

‣ Exames de 800 MB, 5 a 10 minutos por inferência, carga irregular e escala a zero → **Endpoint de inferência assíncrona**

‣ Toda madrugada, milhões de registros no S3 e resultado usado na manhã seguinte → **Job agendado de Batch Transform**

‣ Previsão em lote precisa sair associada ao ID que não pode ir ao modelo → **InputFilter para tirar o ID e JoinSource=Input na saída**

‣ Milhares de modelos XGBoost parecidos, cada um com poucas chamadas por dia → **Endpoint multimodelo**

‣ Pré-processamento scikit-learn e modelo XGBoost em uma única chamada → **Pipeline de inferência serial**

⚠ Endpoints serverless do SageMaker AI não oferecem GPU. Para GPU ociosa, use inference components com escala a zero ou inferência assíncrona.

⚠ Multicontêiner aceita no máximo 15 contêineres. Muitos modelos do mesmo framework vão para endpoint multimodelo.

⚠ Configurações não são editáveis e excluir causa indisponibilidade. Crie modelo e configuração novos e chame UpdateEndpoint.

3. Operação, monitoramento e segurança de soluções de ML e IA

‣ Medir efeito real no comportamento dos usuários com 10% do tráfego → **Production variants com pesos no mesmo endpoint (teste A/B)**

‣ Métrica com variação por turno ou sazonalidade, sem limite fixo → **Alarme com detecção de anomalias do CloudWatch**

‣ Filtrar e agregar milhões de linhas de log JSON sem exportar → **CloudWatch Logs Insights**

‣ Descobrir em qual etapa da cadeia API Gateway, Lambda e endpoint está a lentidão → **AWS X-Ray**

‣ Quem excluiu o endpoint, de qual IP e quando → **Evento DeleteEndpoint no histórico do CloudTrail**

⚠ O Config avalia configuração de recursos. Métricas, logs e alarmes são CloudWatch; conteúdo de invocações é invocation logging.

⚠ O CloudTrail registra chamadas de API, não o conteúdo. Ative o model invocation logging do Bedrock.

⚠ Nenhum dos dois cobre instâncias gerenciadas de ML. Use SageMaker Savings Plans.

4. Preparação de dados para ML e IA

‣ Consulta do Athena usa poucas colunas e um intervalo de datas, mas varre tudo → **Converter para Parquet ou ORC particionado pela data**

‣ HTTP 503 Slow Down ao gravar milhões de objetos no mesmo prefixo → **Distribuir os objetos entre vários prefixos**

‣ Stream JSON precisa chegar ao S3 em Parquet sem escrever código → **Conversão de formato de registro do Data Firehose com esquema do Glue Data Catalog**

‣ Agregação contínua em janela de minutos mantendo estado sobre Kinesis ou Kafka → **Amazon Managed Service for Apache Flink**

‣ Tráfego do stream se multiplica de forma imprevisível e a equipe não quer ajustar shards → **Kinesis Data Streams no modo de capacidade sob demanda**

⚠ Compressão reduz bytes, mas o Athena continua lendo linhas inteiras. O que reduz a varredura é formato colunar (Parquet ou ORC) com particionamento pelas chaves de filtro.

⚠ Com a mesma chave de baixa cardinalidade, os registros continuam indo para um único shard. Troque por uma chave de alta cardinalidade, como o identificador do veículo.

⚠ Fan-out só melhora leitura; retenção só define por quanto tempo o dado fica disponível. IteratorAge alto com Lambda pede ParallelizationFactor maior.

GABARITO

Respostas do diagnóstico

D1. A) Ler os hiperparâmetros como argumentos de linha de comando com argparse → revise o capítulo 1

Por quê: No modo script, os hiperparâmetros definidos no estimador são passados ao script como argumentos de linha de comando e também ficam em /opt/ml/input/config/hyperparameters.json; o script deve lê-los, por exemplo com argparse (tarefa 2.2).

✗ B) Incorreta. Não é o mecanismo do modo script e exige código extra.

✗ C) Incorreta. O artefato do modelo é saída do treino, não entrada.

✗ D) Incorreta. Tags organizam recursos; não chegam ao script como parâmetros.

> **Dica de prova:** Hiperparâmetro ignorado no modo script: ler os argumentos com argparse.

D2. A) Reduzir a taxa de aprendizado ou usar um agendamento que a diminua → revise o capítulo 1

Por quê: Perda oscilando ou divergindo é sinal típico de taxa de aprendizado alta. Reduzi-la, usar agendamento de taxa ou gradient clipping ajuda a depurar problemas de convergência (tarefa 2.3).

✗ B) Incorreta. Mais épocas não corrigem um treino que não converge.

✗ C) Incorreta. Uma taxa maior tende a piorar as oscilações.

✗ D) Incorreta. Dados sem normalização costumam piorar a estabilidade do treino.

> **Dica de prova:** Perda oscilando sem convergir: reduza a taxa de aprendizado.

D3. C) Colocar a coluna alvo na primeira posição e remover a linha de cabeçalho → revise o capítulo 1

Por quê: No formato CSV, os algoritmos integrados do SageMaker AI, como o XGBoost, esperam a variável alvo na primeira coluna e nenhum cabeçalho; com o layout errado, o modelo aprende o alvo incorreto (tarefa 2.2).

✗ A) Incorreta. Árvores de decisão não dependem de escala, e a normalização não corrige o layout do arquivo.

✗ B) Incorreta. Mais rodadas não corrigem um alvo lido da coluna errada.

✗ D) Incorreta. O XGBoost integrado treina com CSV, LIBSVM, Parquet e RecordIO-protobuf; JSON Lines não é formato de treinamento suportado, e o alvo continuaria fora da primeira posição.

> **Dica de prova:** XGBoost integrado com CSV: alvo na primeira coluna e sem cabeçalho.

D4. D) Usar InputFilter para tirar o ID e JoinSource=Input na saída → revise o capítulo 2

Por quê: O Batch Transform permite filtrar colunas de entrada com InputFilter, juntar a entrada à saída com JoinSource e selecionar campos com OutputFilter, associando cada previsão ao seu registro original (tarefa 3.1).

✗ A) Incorreta. Perderia a associação com os registros.

✗ B) Incorreta. Não associa as previsões aos IDs.

✗ C) Incorreta. Usa um identificador como preditor, o que é incorreto.

> **Dica de prova:** Associar previsão ao ID no Batch Transform: InputFilter e JoinSource.

D5. B) Definir zero cópias como mínimo no inference component, com target tracking e step scaling → revise o capítulo 2

Por quê: Endpoints com inference components podem escalar para zero instâncias: registre o componente com capacidade mínima zero, use target tracking por cópia e um step scaling acionado por NoCapacityInvocationFailures (tarefa 3.2).

✗ A) Incorreta. CPU não reflete a carga de um LLM em GPU, e manter uma instância ligada gera custo ocioso.

✗ C) Incorreta. Endpoints serverless do SageMaker AI não oferecem GPU.

✗ D) Incorreta. Uma programação fixa não acompanha a demanda e paga GPU ociosa de madrugada.

> **Dica de prova:** GPU ociosa de madrugada: inference components com escala até zero.

D6. C) Definir parâmetros do pipeline para tipo de instância e caminho dos dados → revise o capítulo 2

Por quê: O SageMaker Pipelines aceita parâmetros, como ParameterString e ParameterInteger, que substituem valores como tipo de instância e URI de dados em cada execução, mantendo uma única definição (tarefa 3.3).

✗ A) Incorreta. Aumenta o custo em desenvolvimento e não troca o dataset.

✗ B) Incorreta. Mantém duas definições para sincronizar, o que se quer evitar.

✗ D) Incorreta. É um processo manual e sujeito a erro.

> **Dica de prova:** Mesmo pipeline com valores diferentes por ambiente: parâmetros do pipeline.

D7. B) Regerar os embeddings com 256 dimensões no Titan Text Embeddings V2 e reindexar o acervo → revise o capítulo 3

Por quê: Otimizar o custo de bancos vetoriais inclui escolher a dimensão dos embeddings: o Titan Text Embeddings V2 permite 256, 512 ou 1.024 dimensões, trocando um pouco de precisão por menos memória (tarefa 4.2).

✗ A) Incorreta. Réplicas multiplicam a memória consumida em vez de reduzi-la.

✗ C) Incorreta. Mais dimensões aumentam o custo de armazenamento e de memória.

✗ D) Incorreta. Duplicar vetores dobra o custo.

> **Dica de prova:** Vetor menor, índice mais barato: reduzir dimensões (e reindexar).

D8. B) Conceder permissões em nível de coluna com o AWS Lake Formation → revise o capítulo 3

Por quê: O AWS Lake Formation oferece permissões finas, em nível de coluna, linha e célula, para tabelas do Data Catalog consultadas por serviços como o Athena, aplicando o menor privilégio sem duplicar dados (tarefa 4.3).

✗ A) Incorreta. Dá acesso aos arquivos inteiros, inclusive às colunas sensíveis.

✗ C) Incorreta. Viola o requisito de manter uma única cópia dos dados.

✗ D) Incorreta. Quem tem leitura vê todas as colunas descriptografadas.

> **Dica de prova:** Esconder colunas sensíveis mantendo uma cópia: Lake Formation em nível de coluna.

D9. C) Usar a detecção de anomalias do AWS Glue Data Quality com analisadores nos lotes → revise o capítulo 3

Por quê: Detectar mudanças na distribuição dos dados de entrada pode usar o AWS Glue Data Quality com detecção de anomalias, que aprende padrões históricos sem limites fixos (tarefa 4.1).

✗ A) Incorreta. O Config monitora a configuração de recursos da AWS, não o conteúdo dos dados.
✗ B) Incorreta. Limites fixos não acompanham a evolução normal dos dados.
✗ D) Incorreta. O Macie busca dados sensíveis, não desvios estatísticos.

> **Dica de prova:** Mudança estatística sem limiar fixo: detecção de anomalias do Glue Data Quality.

D10. B) Executar os scripts no Amazon EMR com cluster configurado pela equipe → revise o capítulo 4

Por quê: O Amazon EMR é a opção para Spark em larga escala quando a equipe precisa controlar versões, tipos de instância, bibliotecas nativas e configurações do framework, algo que serviços visuais ou sem servidor não oferecem no mesmo nível (tarefa 1.2).

✗ A) Incorreta. Exigiria reescrever tudo em uma ferramenta sem código, perdendo as bibliotecas nativas.
✗ C) Incorreta. O Lambda tem limite de 15 minutos e de memória, inadequado para Spark distribuído sobre 50 TB.
✗ D) Incorreta. Exigiria reescrever a lógica em SQL e não preserva as bibliotecas e os ajustes do Spark.

> **Dica de prova:** Spark existente com controle fino do cluster: Amazon EMR.

D11. C) Gerar pré-rótulos das imagens com um FM multimodal do Amazon Bedrock e enviá-los à verificação humana → revise o capítulo 4

Por quê: A pré-rotulagem com modelos de IA generativa do Amazon Bedrock, seguida de verificação por pessoas, é uma prática recomendada para acelerar a rotulagem e reduzir o custo mantendo o controle de qualidade (tarefa 1.3).

✗ A) Incorreta. O DataBrew prepara dados tabulares; não rotula imagens.
✗ B) Incorreta. Rótulos genéricos não identificam os produtos específicos da rede, e o requisito exige conferência humana.
✗ D) Incorreta. O Custom Labels precisa de imagens já rotuladas para treinar o modelo.

> **Dica de prova:** Rotular muito com custo menor e conferência humana: pré-rótulos de um FM e revisão por pessoas.

D12. D) Aplicar one-hot encoding, com uma coluna binária por forma de pagamento → revise o capítulo 4

Por quê: Variáveis nominais com poucas categorias devem ser codificadas com one-hot encoding; a codificação por inteiros cria relações de ordem e magnitude inexistentes, prejudicando modelos lineares (tarefa 1.2).

✗ A) Incorreta. Não remove a relação numérica artificial.
✗ B) Incorreta. Ainda impõe uma ordem artificial a categorias nominais.
✗ C) Incorreta. Muda a escala, mas mantém a ordem falsa entre as categorias.

> **Dica de prova:** Categoria sem ordem e com poucos valores: one-hot encoding.

Agora, treine no formato real

Faça as 6 provas completas do curso MLA-C02: 390 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 Amazon Web Services (AWS). As questões são inéditas e não reproduzem a prova real.