Strecht Simulados · Guia de preparação



Guia de preparação DEA-C01

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 DEA-C01: 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 130 minutos, no formato do exame. Referência de aprovação: 72%.

Domínio (capítulo)	Peso	Neste guia
1. Domínio 1: Ingestão e transformação de dados	34%	18 conceitos · 14 pegadinhas
2. Domínio 2: Gerenciamento do armazenamento de dados	26%	18 conceitos · 14 pegadinhas
3. Domínio 3: Operações e suporte de dados	22%	17 conceitos · 14 pegadinhas
4. Domínio 4: Segurança e governança de dados	18%	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 · Domínio 1: Ingestão e transformação de dados

Um stream do Amazon Data Firehose de uma rede de lojas usa uma função Lambda para transformar os registros antes da entrega no Amazon S3. A contagem diária no S3 está menor que a quantidade enviada, a função não mostra erros de timeout, e a equipe quer saber quais registros não foram entregues e por quê. Onde deve procurar?

A. No buffer do Firehose, que retém os registros com falha por 24 horas para reenvio pelo console
B. No histórico de eventos do AWS CloudTrail, que guarda os registros rejeitados pelo Firehose
C. Na fila de mensagens mortas configurada na função Lambda para invocações assíncronas com falha
D. Na pasta processing-failed do bucket de destino, com o código e a mensagem de erro de cada registro

D2 · Domínio 1: Ingestão e transformação de dados

Cientistas de dados de uma varejista escreveram em pandas um script de limpeza que funciona com amostras, mas falta memória ao processar a base completa de 200 GB em uma única máquina. A equipe quer executar no AWS Glue 4.0 com o mínimo de reescrita, sem aprender a API de DataFrame do Spark. Qual abordagem atende?

A. Usar a pandas API on Spark (pyspark.pandas), que mantém a sintaxe do pandas com execução distribuída
B. Reescrever o script em Scala com RDDs para executar o processamento distribuído nos workers do Glue
C. Executar o script em um job Python shell do Glue com 1 DPU, carregando a base completa em memória
D. Ler a base em lotes de 1 GB com pandas em uma Lambda e unir os resultados parciais em um único arquivo

D3 · Domínio 1: Ingestão e transformação de dados

Um job Spark do AWS Glue 4.0 de uma empresa de cartões processa volumes muito diferentes: nas segundas-feiras precisa de 60 workers, nos outros dias termina bem com 10. Hoje está fixo em 60 workers, e a maior parte da capacidade fica ociosa durante a semana. Como reduzir o custo sem ajustar o job manualmente a cada dia?

A. Ativar o auto scaling do job e definir 60 como número máximo de workers
B. Mudar o job para a classe de execução Flex mantendo os 60 workers fixos
C. Reduzir o job para 10 workers e aceitar o atraso maior nas segundas-feiras
D. Trocar o tipo de worker para G.025X para pagar menos por cada um dos workers

D4 · Domínio 2: Gerenciamento do armazenamento de dados

Um hospital guarda exames de imagem com mais de 6 meses no Amazon S3. Eles são acessados cerca de uma vez por trimestre, mas, quando um médico abre um exame antigo, a imagem precisa aparecer em milissegundos. A instituição quer o menor custo de armazenamento que atenda a esse padrão. Para qual classe a regra de Lifecycle deve mover os objetos?

A. S3 Standard-IA, com recuperação em milissegundos e mínimo de 30 dias de cobrança
B. S3 Glacier Instant Retrieval, com recuperação em milissegundos e mínimo de 90 dias
C. S3 Glacier Flexible Retrieval, com recuperação expedita feita em um a cinco minutos
D. S3 Glacier Deep Archive, com o menor custo por GB entre as classes do Amazon S3

D5 · Domínio 2: Gerenciamento do armazenamento de dados

Uma plataforma de imagens precisa indexar 2 bilhões de vetores de 128 dimensões no Amazon OpenSearch Service para busca por similaridade. Um índice HNSW exigiria mais de um terabyte de memória, acima do orçamento. O produto aceita recall bem menor em troca de custo muito mais baixo. Qual configuração de índice k-NN atende?

A. Usar busca exata com script score para não manter grafos de vizinhança em memória
B. Usar o método IVF do faiss com product quantization (PQ), treinando o modelo antes
C. Usar HNSW com o parâmetro m maior para reduzir o número de conexões mantidas em memória
D. Usar HNSW com ef_search maior para compensar o recall perdido ao reduzir a memória

D6 · Domínio 2: Gerenciamento do armazenamento de dados

Uma cooperativa agrícola precisa buscar toda madrugada arquivos de cotações que uma bolsa de mercadorias publica no próprio servidor SFTP externo e gravá-los no Amazon S3. A bolsa não envia arquivos aos clientes, e a cooperativa não quer manter servidores nem scripts em EC2. Qual solução atende?

A. Configurar S3 Event Notifications para buscar os arquivos no SFTP da bolsa quando publicados
B. Criar um conector SFTP do AWS Transfer Family e agendar a recuperação com StartFileTransfer
C. Criar um servidor SFTP no AWS Transfer Family para que a bolsa deposite os arquivos no bucket
D. Usar o Amazon Data Firehose com o servidor SFTP da bolsa como origem de dados agendada

D7 · Domínio 3: Operações e suporte de dados

Um engenheiro publicou uma DAG nova na pasta dags/ do bucket de um ambiente Amazon MWAA, mas ela nunca aparece na interface do Airflow. Não há erro visível na tela e as outras DAGs continuam funcionando. Onde procurar a causa mais provável primeiro?

A. Nos logs de processamento de DAGs do ambiente no CloudWatch Logs
B. Nos logs de tarefas da DAG nova, procurando a primeira execução
C. Nas métricas de CPU do ambiente no painel do Amazon CloudWatch
D. No Event history do CloudTrail, buscando o PutObject do arquivo novo

D8 · Domínio 3: Operações e suporte de dados

No Amazon Athena, a tabela pedidos tem a coluna itens do tipo array, com os produtos de cada pedido. O time comercial quer uma linha por produto vendido, mantendo o número do pedido, para contar quantas vezes cada produto aparece. Qual construção SQL usar?

A. CARDINALITY(itens) na lista do SELECT, agrupando por pedido
B. ARRAY_AGG(itens) com GROUP BY pelo número do pedido
C. CROSS JOIN UNNEST(itens) AS t(produto) na cláusula FROM
D. LATERAL VIEW com split da coluna itens convertida em texto

D9 · Domínio 3: Operações e suporte de dados

Ontem à tarde, a tabela pedidos sumiu do AWS Glue Data Catalog e as consultas do Amazon Athena começaram a falhar. A conta não tem nenhuma trilha do AWS CloudTrail configurada. Como descobrir rapidamente qual identidade excluiu a tabela?

A. Rodar uma consulta do Logs Insights no log group padrão do AWS Glue
B. Consultar um event data store do CloudTrail Lake criado nesta manhã
C. Abrir o histórico de versões da tabela no Data Catalog e ver quem a excluiu
D. Pesquisar o evento DeleteTable no Event history do CloudTrail no console

D10 · Domínio 4: Segurança e governança de dados

Uma nova instância do Amazon RDS for PostgreSQL vai alimentar pipelines de dados. A equipe não quer que ninguém conheça ou guarde a senha do usuário master, que deve ficar em um cofre e ser rotacionada sem que a equipe implemente uma função de rotação. Qual é a opção com menor esforço?

A. Guardar a senha master como SecureString no Systems Manager Parameter Store
B. Criar um segredo manualmente e escrever uma Lambda própria para a rotação
C. Definir a senha master no template do CloudFormation como parâmetro NoEcho
D. Habilitar o gerenciamento da senha master pelo RDS no AWS Secrets Manager

D11 · Domínio 4: Segurança e governança de dados

Uma empresa adotou o Amazon SageMaker Unified Studio. A equipe do caso de uso de churn (engenheiros, cientistas e analistas) precisa de um espaço comum para compartilhar dados, notebooks e recursos de computação, sem que as demais equipes do mesmo domínio vejam esse trabalho. Como organizar o acesso?

A. Publicar os notebooks no SageMaker Catalog para que todos do domínio possam encontrá-los e assinar
B. Criar um projeto para o caso de uso e adicionar os integrantes como membros do projeto
C. Criar um grupo IAM com acesso a todos os buckets do domínio e incluir os integrantes
D. Criar um domínio separado do Unified Studio para cada integrante da equipe de churn

D12 · Domínio 4: Segurança e governança de dados

No Amazon Redshift, a tabela funcionarios tem 25 colunas, entre elas salario e cpf. A equipe de RH analítico pode consultar todas as colunas, exceto salario e cpf, e a empresa não quer criar nem manter views para isso. Qual comando atende?

A. GRANT SELECT nas 23 colunas permitidas da tabela funcionarios ao grupo de RH analítico
B. CREATE RLS POLICY que retorna nulo nas colunas salario e cpf para o RH analítico
C. GRANT SELECT na tabela funcionarios ao grupo e REVOKE da coluna salario e da coluna cpf
D. GRANT USAGE no schema do RH e SELECT nas tabelas em que não exista a coluna salario

01

CAPÍTULO 1 · 34% DA PROVA

Domínio 1: Ingestão e transformação de dados

É o domínio mais pesado da prova (34%), então preciso dominar como os dados entram (streaming, lote, CDC, SaaS, APIs) e como são transformados e orquestrados. Os enunciados quase sempre trazem um sintoma concreto (throttling em um shard, job do Glue reprocessando tudo, Lambda passando de 15 minutos, arquivo gzip lido por uma task só) e pedem o ajuste exato com menor esforço operacional. Onde mais erro é nos parâmetros finos: ParallelizationFactor x enhanced fan-out, concorrência reservada x provisionada, Standard x Express, .sync x waitForTaskToken. Também caem muito as escolhas de serviço: Glue x EMR x EMR Serverless x Lambda x Batch x Fargate, e DMS x DataSync x AppFlow x zero-ETL. A regra que mais me salva aqui: se a pergunta diz sem código e sem servidor, a resposta é o recurso nativo do serviço, não uma Lambda de cola.

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

Tópicos-chave

Kinesis Data Streams: shards e chave de partição

Um hash MD5 da chave de partição define o shard; cada shard aceita até 1 MB/s ou 1.000 registros/s de escrita. Se um shard sofre throttling e os outros estão ociosos, o problema é a chave de baixa cardinalidade: troque por uma de alta cardinalidade (ex.: id do veículo). Se todos os shards estouram com picos imprevisíveis e ninguém quer gerenciar shards, mude para o modo on-demand.

Kinesis: leitura compartilhada x enhanced fan-out

No modo padrão, todos os consumidores dividem 2 MB/s e 5 chamadas GetRecords por segundo em cada shard. O enhanced fan-out dá a cada consumidor registrado um pipe próprio de 2 MB/s por shard, com menor latência. Ele só melhora a leitura: não muda limite de escrita, não aumenta retenção e não trata erros.

Retenção e agregação no Kinesis

Para reprocessar eventos de dias atrás direto do stream, aumente a retenção com IncreaseStreamRetentionPeriod (até 365 dias). Muitos registros minúsculos batendo no limite de registros por shard se resolvem com agregação da KPL, que junta vários registros pequenos em um registro do Kinesis; comprimir com gzip não adianta, porque cada leitura continua sendo um registro.

Event source mapping da Lambda com streams

ParallelizationFactor processa vários lotes do mesmo shard ao mesmo tempo mantendo a ordem por chave; BatchSize e MaximumBatchingWindowInSeconds acumulam mais registros por invocação; filter criteria descartam eventos antes de invocar. Registro venenoso travando o shard: BisectBatchOnFunctionError, MaximumRetryAttempts limitado e destino on-failure (ex.: SQS). Agregação por janela sem banco externo: TumblingWindowInSeconds devolvendo o estado na resposta.

Amazon Data Firehose

Entrega em lote para S3, Redshift e outros destinos conforme buffer size e buffer interval: arquivos pequenos demais se resolvem aumentando os dois. A transformação com Lambda devolve cada registro como Ok, Dropped ou falha, e as falhas vão para a pasta processing-failed do bucket. Record format conversion gera Parquet usando o esquema de uma tabela do Glue; o Firehose também aceita um tópico do MSK como origem sem código.

Jobs Spark do AWS Glue: leitura eficiente

Job bookmarks (job-bookmark-enable + transformation_ctx + job.commit) evitam reprocessar arquivos já lidos. push_down_predicate filtra partições antes da leitura; groupFiles e groupSize juntam milhares de arquivos pequenos por task; hashfield ou hashexpression com hashpartitions paralelizam a leitura JDBC. Um gzip enorme é lido por uma task só porque gzip não é divisível: divida o arquivo ou use bzip2.

Capacidade e custo de jobs do Glue

Auto scaling com número máximo de workers ajusta a capacidade à carga variável. A classe Flex usa capacidade ociosa mais barata para jobs sem pressa. Python shell (0,0625 ou 1 DPU) atende scripts Python simples, sem Spark, que passam de 15 minutos. G.025X é para streaming de baixo volume; executors saturados com tasks pendentes pedem mais workers.

Transformações do Glue (DynamicFrame)

Relationalize achata JSON aninhado e gera tabelas separadas para arrays com chaves de junção; ResolveChoice trata campos com mais de um tipo; ApplyMapping renomeia e converte; DropNullFields remove campos nulos. Registros malformados vão para errorsAsDynamicFrame com limites como totalThreshold, sem parar o job. FindMatches (ML) casa o mesmo cliente em fontes sem chave comum.

Otimização de Spark

Tabela grande com tabela pequena e shuffle pesado: broadcast join. UDF Python lenta: troque por funções nativas (upper, trim, to_date). Mesmo DataFrame usado em várias ações: cache ou persist. Código pandas sem memória: pandas API on Spark. JSON que vira _corrupt_record: opção multiLine. Evite collect e coalesce(1) em dados grandes.

Glue streaming x Managed Service for Apache Flink

O job de streaming do Glue lê Kinesis ou Kafka em micro-lotes com checkpoint no S3 e reaproveita código Spark, com latência de segundos a minutos. O Flink mantém estado por chave, detecta padrões entre eventos com latência de milissegundos e, ao atualizar a aplicação, restaura o estado a partir do snapshot.

Step Functions: integrações e estados

glue:startJobRun.sync espera o job terminar; .waitForTaskToken pausa até um sistema externo chamar SendTaskSuccess; StartExecution .sync chama uma state machine filha. Parallel roda etapas independentes, Choice ramifica pelo resultado, Map distribuído processa milhões de objetos do S3 com ItemReader. Retry com backoff para erro transitório e Catch para notificação; HTTP Task com conexão do EventBridge chama API SaaS com OAuth; integração otimizada chama o Bedrock sem Lambda.

Agendamento e eventos (EventBridge)

EventBridge Scheduler aceita fuso horário (ex.: America/New_York) e acompanha o horário de verão, tem flexible time window para espalhar disparos e agendamentos únicos at() com exclusão automática. EventBridge Pipes liga origem, filtro, enriquecimento e alvo sem código de cola. Archive + replay reprocessa eventos antigos de um barramento. Eventos do S3 enviados ao EventBridge permitem filtro por sufixo e tamanho e alvo Step Functions.

Orquestração do Glue e MWAA

Glue workflow orquestra só crawlers e jobs do Glue, com grafo, parâmetros compartilhados e retomada. Trigger agendado usa cron em UTC; trigger condicional observa estado de jobs e crawlers; trigger EVENT com BatchSize e BatchWindow inicia o workflow por chegada de arquivos via EventBridge. DAGs do Airflow existentes vão para o Amazon MWAA com mínimo de mudança.

CDC, migração e integrações sem pipeline

AWS DMS faz full load e replicação contínua (CDC) de bancos; no MySQL o CDC exige backups automáticos e binlog_format ROW, e o destino pode ser Kinesis Data Streams. Zero-ETL leva Aurora para o Redshift com atraso de segundos; o Glue zero-ETL replica SaaS como Salesforce para tabelas Iceberg. AppFlow agendado com transferência incremental traz SaaS (Zendesk etc.) para o S3.

Limites e tuning da Lambda

Tempo máximo de 15 minutos; /tmp configurável até 10.240 MB e separado da memória; mais memória dá mais CPU proporcional. EFS para arquivos grandes compartilhados entre execuções, layers para dependências repetidas, arm64 (Graviton) para reduzir custo sem mudar código Python puro. RDS Proxy compartilha conexões quando milhares de execuções estouram o banco.

Mensageria: SQS e SNS

Fan-out com retenção: um tópico SNS com uma fila SQS por consumidor. Ordem e sem duplicidade: SNS FIFO + SQS FIFO com o ID da conta como message group ID. Filas padrão entregam pelo menos uma vez, então efeito duplicado se evita com idempotência (escrita condicional attribute_not_exists). Filter policy no SNS entrega a cada assinante só parte das mensagens.

Ingestão no Amazon Redshift

COPY é paralelo por slice: divida um arquivo único em N partes de tamanho parecido, N múltiplo do número de slices. Streaming ingestion lê Kinesis ou MSK com schema externo e materialized view com auto refresh, sem S3 no meio. JSON aninhado e mutável vai para coluna SUPER consultada com PartiQL. Redshift Data API executa SQL de Lambda sem driver nem conexões; CREATE EXTERNAL MODEL do tipo BEDROCK chama LLM direto no SELECT.

IaC e CI/CD de pipelines

CloudFormation: DeletionPolicy Retain mantém bucket e tabela ao excluir a stack, change set mostra o que será substituído antes de aplicar, StackSets com permissões gerenciadas pelo serviço implantam em muitas contas, inclusive as novas. SAM: DeploymentPreference canary com alarmes, sam local generate-event + sam local invoke e DefinitionSubstitutions. CDK oferece constructs reutilizáveis em Python; CodePipeline com CodeBuild e aprovação manual, origem GitHub via CodeConnections.

Macetes para memorizar

Concorrência da Lambda

Reservada = Restringe (teto). Provisionada = Pré-aquece (cold start).

Kinesis: o que melhora o quê

Escreve muito: chave, shard, on-demand, KPL. Lê muito: fan-out. Função lenta: paraleliza. Erro de dado: bisect + on-failure.

Step Functions

Express é expresso: até 5 minutos e sem esperar ninguém. Standard é paciente: até 1 ano, callback e redrive.

Integrações .sync x token

.sync = espera um serviço da AWS terminar. Token = espera alguém de fora responder.

Formatos

Avro anda (streaming, evolução de esquema); Parquet pensa (consulta colunar analítica).

Glue: os três parâmetros de leitura

Partição? push_down_predicate. Muitos arquivos? groupFiles. JDBC lento? hashpartitions.

Spot no EMR

Spot só no task: o primário é a cabeça e o core guarda o HDFS.

Mover dados

Banco = DMS. Arquivo = DataSync. SaaS = AppFlow. Aurora/DynamoDB/Salesforce sem pipeline = zero-ETL.

Tabela de memorização: não confunda

Quando é...	...e quando é
Step Functions Standard Execuções longas (até 1 ano), exactly-once, histórico auditável, callback com waitForTaskToken e redrive.	**Step Functions Express** Altíssimo volume e curta duração (até 5 minutos), cobrado por execução e duração; sem padrão de callback.
Concorrência reservada Impor um teto de execuções simultâneas para proteger um banco downstream e garantir capacidade à função.	**Concorrência provisionada** Manter ambientes pré-inicializados para eliminar cold start em API sensível à latência; não limita nem reduz custo.
Enhanced fan-out Vários consumidores disputando os 2 MB/s de leitura do shard; cada um ganha vazão dedicada.	**ParallelizationFactor** Uma função lenta atrasada no stream; processa vários lotes do mesmo shard em paralelo sem aumentar shards e mantendo ordem por chave.
AWS Glue (Spark serverless, Glue Studio) ETL Spark sem gerenciar cluster, com editor visual, bookmarks e integração com o Data Catalog.	**Amazon EMR / EMR Serverless / EMR on EKS** Hive, Pig, MapReduce, Presto ou ajuste fino do cluster (EMR); Spark intermitente sem cluster ligado (EMR Serverless); runtime do EMR dentro de um EKS existente (EMR on EKS).
AWS DMS Tabelas de bancos de dados: full load, CDC pelo log de transações, destino S3 em Parquet ou Kinesis.	**AWS DataSync** Arquivos e objetos entre sistemas de armazenamento (NFS, SMB, HDFS, S3), com agendamento e verificação; não lê log de banco.
Glue streaming job Reaproveitar código Spark em streaming, com micro-lotes e checkpoint no S3.	**Managed Service for Apache Flink** Estado por chave, detecção de padrões entre eventos e decisão em milissegundos.
SNS + uma fila SQS por consumidor O mesmo evento precisa chegar a vários sistemas independentes, cada um no seu ritmo e com retenção se ficar fora do ar.	**Uma única fila SQS** Distribuir trabalho entre workers concorrentes: cada mensagem é consumida por um único consumidor.

Números que caem

Item	Valor

Tempo máximo de execução da Lambda	**15 minutos**
Armazenamento efêmero /tmp da Lambda	**512 MB padrão, configurável até 10.240 MB**
Escrita por shard do Kinesis (provisionado)	**até 1 MB/s ou 1.000 registros/s**
Leitura compartilhada por shard do Kinesis	**2 MB/s e 5 GetRecords/s divididos entre consumidores; enhanced fan-out dá 2 MB/s por consumidor**
Retenção máxima do Kinesis Data Streams	**até 365 dias**
Step Functions Express x Standard	**até 5 minutos x até 1 ano**
BatchWriteItem do DynamoDB	**máximo de 25 solicitações por chamada**
Atraso máximo de entrega do SQS	**15 minutos**
Glue Python shell	**0,0625 ou 1 DPU**
Taxa de requisições do S3 por prefixo	**3.500 PUT/COPY/POST/DELETE e 5.500 GET/HEAD por segundo**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Ativar enhanced fan-out para resolver throttling de escrita no Kinesis.
Como não cair: Enhanced fan-out só amplia a leitura dos consumidores. Escrita se resolve com chave de alta cardinalidade, mais shards, on-demand ou agregação da KPL.

⚠ **Alerta de prova:** Dobrar os shards mantendo a mesma chave de partição dominante.
Como não cair: Com a mesma chave quente, os registros continuam caindo no mesmo shard; o conserto é a chave de alta cardinalidade.

⚠ **Alerta de prova:** Usar job bookmarks para paralelizar leitura JDBC, tratar registros malformados ou enfileirar execuções.
Como não cair: Bookmarks só controlam o que já foi processado entre execuções. Paralelismo JDBC é hashfield/hashpartitions; malformados é errorsAsDynamicFrame.

⚠ **Alerta de prova:** Aumentar workers ou chamar repartition para acelerar a leitura de um gzip gigante ou de uma tabela JDBC lida por uma task.
Como não cair: repartition age depois da leitura e mais workers não dividem um arquivo não divisível; o gargalo está na forma de leitura.

⚠ **Alerta de prova:** Configurar a DLQ da Lambda para capturar registros que falharam no Firehose ou no Step Functions.
Como não cair: Firehose e Step Functions invocam a Lambda de forma síncrona; a DLQ vale só para invocações assíncronas. No Firehose, olhe a pasta processing-failed.

⚠ **Alerta de prova:** Aumentar o visibility timeout ou ativar long polling para acabar com mensagens duplicadas.
Como não cair: Fila padrão entrega pelo menos uma vez. Efeito duplicado se evita com chave de idempotência e escrita condicional.

⚠ **Alerta de prova:** Usar rate(1 day) ou trigger agendado do Glue para rodar às 6h no horário local com horário de verão.
Como não cair: rate conta intervalos de 24 horas e o cron do Glue é em UTC; só o EventBridge Scheduler com time zone acompanha o horário de verão.

⚠ **Alerta de prova:** Usar um workflow Express para esperar aprovação humana ou um estado Wait em loop para saber quando o job terminou.
Como não cair: Express dura até 5 minutos e não tem callback. Use .waitForTaskToken para aprovação e glue:startJobRun.sync para esperar o job.

⚠ **Alerta de prova:** Criar gateway endpoint para a API do parceiro ou associar Elastic IP à Lambda para ter IP fixo.
Como não cair: Gateway endpoints existem só para S3 e DynamoDB, e Lambda não recebe Elastic IP. IP fixo de saída é NAT gateway com Elastic IP e rota 0.0.0.0/0 das sub-redes privadas.

⚠ **Alerta de prova:** Usar trigger condicional do Glue ou notificação do S3 para iniciar o workflow quando chegarem 100 arquivos.
Como não cair: Trigger condicional observa jobs e crawlers, e notificação do S3 não inicia workflow do Glue. O certo é regra do EventBridge + trigger EVENT com BatchSize e BatchWindow.

⚠ **Alerta de prova:** Ativar compressão GZIP ou Snappy no Firehose para gerar menos arquivos ou acelerar o Athena.
Como não cair: Compressão reduz bytes, mas não muda o número de objetos nem torna o formato colunar. Menos arquivos: buffer maior. Athena mais barato: conversão para Parquet.

⚠ **Alerta de prova:** Usar UpdateReplacePolicy ou proteção contra encerramento para manter o bucket ao excluir a stack.
Como não cair: UpdateReplacePolicy só atua em substituição durante update e a proteção bloqueia a exclusão inteira. Manter recurso na exclusão é DeletionPolicy Retain.

⚠ **Alerta de prova:** Aumentar o BatchSize de BatchWriteItem acima de 25 ou trocar por TransactWriteItems quando itens somem sem erro.
Como não cair: O limite é 25 por chamada; itens rejeitados voltam em UnprocessedItems e devem ser reenviados com backoff exponencial e jitter.

⚠ **Alerta de prova:** Usar concorrência provisionada ou mais memória para limitar conexões ao RDS.
Como não cair: Teto de execuções simultâneas é concorrência reservada; compartilhar conexões em pico é RDS Proxy.

Como cai na prova

📝 **Como cai:** Cenário com sintoma de desempenho e quatro ajustes plausíveis (mais workers, mais memória, outro tipo de worker, um parâmetro específico).
Como resolver: Identifique onde está o gargalo (leitura, shuffle, escrita, uma task só, driver). Descarte o aumento genérico de capacidade quando o problema é a forma de leitura ou a distribuição; escolha o parâmetro que ataca exatamente aquele ponto.

📝 **Como cai:** Pede a solução com menor esforço operacional ou sem código, e uma alternativa é uma Lambda que funciona.
Como resolver: A Lambda de cola costuma ser a alternativa que funciona mas não atende ao requisito. Procure o recurso nativo: filter policy, Pipes, input transformer, HTTP Task, integração otimizada, zero-ETL.

📝 **Como cai:** Escolha entre serviços de computação para um trabalho com duração e forma definidas.
Como resolver: Compare a duração com 15 minutos (Lambda), veja se há Spark (Glue/EMR), contêiner (Batch para muitos jobs com fila, Fargate para um job longo) ou frameworks Hadoop (EMR).

📝 **Como cai:** Multiple response com duas etapas de uma mesma configuração (ex.: bookmarks + transformation_ctx; NAT gateway + rota; Retry + Catch).
Como resolver: Procure as duas peças que juntas fecham o mecanismo; desconfie de alternativas isoladas que resolvem só metade ou mudam o requisito.

📝 **Como cai:** Orquestração: workflow que precisa esperar, ramificar, paralelizar ou repetir.
Como resolver: Mapeie o verbo ao estado: esperar job = .sync; esperar pessoa = token; independentes = Parallel; decidir = Choice; milhões de itens = Map distribuído; erro transitório = Retry; fim com falha = Catch.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Um shard com throttling e os outros ociosos	**Chave de partição de alta cardinalidade**
Vários consumidores no mesmo stream ficando atrasados	**Enhanced fan-out**
Registros de ~100 bytes estourando o limite de registros por shard	**Agregação com a KPL**
Job do Glue reprocessa todos os arquivos a cada execução	**Job bookmarks + transformation_ctx + job.commit**
Centenas de milhares de arquivos de poucos KB deixando o Glue lento	**groupFiles inPartition + groupSize (no EMR: S3DistCp --groupBy)**
Script de mais de 15 minutos, sem Spark, barato	**Glue Python shell; se for contêiner longo, ECS no Fargate**
Milhares de jobs em contêiner com filas, prioridade e Spot	**AWS Batch**
Mudanças de um banco relacional viram eventos sem tocar na aplicação	**DMS full load + CDC com destino Kinesis Data Streams**
Aurora para Redshift em segundos, sem pipeline	**Integração zero-ETL**
Kinesis ou MSK no Redshift em segundos, sem S3 no meio	**Streaming ingestion com materialized view**
Origem, filtro, enriquecimento e alvo sem código de cola	**EventBridge Pipes**
LLM sobre milhões de registros sem pressa e com custo menor	**Bedrock batch inference com JSONL no S3**
Imagem ou documento com layout variável para campos estruturados	**Modelo multimodal do Bedrock pela Converse API com saída JSON**
Clientes HTTPS sem SDK enviando para o Kinesis sem código	**API Gateway com integração de serviço PutRecord**

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

- Lambda tem teto de 15 minutos; tudo que passa disso vai para Glue Python shell, Fargate, Batch ou loop no Step Functions.
- Gzip não é divisível: um arquivo gzip gigante sempre vira uma task só no Spark.
- DLQ da Lambda só vale para invocação assíncrona; Firehose, Step Functions e event source mapping têm seus próprios mecanismos de falha.
- Quando o enunciado pede sem código, a alternativa com Lambda de cola quase sempre é a errada.

Essenciais para a prova (checklist)

☐ Sei diagnosticar throttling de escrita no Kinesis (chave quente x capacidade total x registros pequenos) e a correção de cada caso.

☐ Sei os parâmetros do event source mapping: ParallelizationFactor, BatchSize, batching window, bisect, filter criteria e tumbling window.

☐ Sei ativar job bookmarks corretamente e quando usar push_down_predicate, groupFiles e hashpartitions.

☐ Sei escolher entre Glue, EMR, EMR Serverless, EMR on EKS, Lambda, Batch e Fargate pela duração e pelo tipo de trabalho.

☐ Sei a diferença entre Standard e Express e quando usar .sync, waitForTaskToken, Parallel, Choice e Map distribuído.

☐ Sei configurar DMS com CDC no MySQL (backups automáticos e binlog ROW) e quando preferir zero-ETL ou AppFlow.

☐ Sei por que o COPY do Redshift precisa de arquivos divididos em múltiplos do número de slices.

☐ Sei diferenciar concorrência reservada, concorrência provisionada e RDS Proxy.

☐ Sei montar fan-out com SNS + SQS e garantir ordem com SNS FIFO + SQS FIFO.

☐ Sei proteger recursos com DeletionPolicy Retain e prever substituições com change set.

Questões de revisão do capítulo

R1.1 · Domínio 1: Ingestão e transformação de dados

Uma stack do AWS CloudFormation de uma empresa de pesquisa cria o bucket da zona raw do data lake, uma tabela do DynamoDB de controle e os jobs do Glue. A equipe vai substituir a stack por outra arquitetura e precisa excluí-la, mas o bucket e a tabela devem continuar existindo com os dados. O que deve ser feito antes da exclusão?

A. Definir UpdateReplacePolicy como Snapshot no bucket e na tabela do template
B. Definir DeletionPolicy como Retain no bucket e na tabela e atualizar a stack
C. Executar a detecção de drift para marcar o bucket e a tabela como externos
D. Ativar a proteção contra encerramento da stack para manter os recursos com dados

R1.2 · Domínio 1: Ingestão e transformação de dados

Funções Lambda de uma plataforma de ingressos gravam pedidos em um Amazon RDS for MySQL. Em grandes vendas, milhares de execuções simultâneas abrem conexões novas e o banco responde com erro de conexões em excesso. A empresa não quer limitar a concorrência das funções, porque precisa absorver o pico. Qual solução atende?

A. Aumentar o timeout das funções para que elas esperem por uma conexão livre no banco de dados
B. Colocar um Amazon RDS Proxy entre as funções e o banco para compartilhar e reutilizar conexões
C. Criar uma réplica de leitura do RDS e distribuir as gravações das funções entre as duas instâncias
D. Configurar concorrência reservada baixa nas funções para limitar as conexões abertas no banco

R1.3 · Domínio 1: Ingestão e transformação de dados

Um job do AWS Glue lê uma tabela de 900 milhões de linhas de um Amazon RDS for PostgreSQL por uma conexão JDBC. O Spark UI mostra uma única task fazendo toda a leitura enquanto os demais executors ficam ociosos. Como o engenheiro pode paralelizar a leitura sem alterar o banco de origem?

A. Aumentar o número de workers para que o driver JDBC abra mais cursores na tabela automaticamente
B. Trocar o tipo de worker para G.8X para que a única task de leitura tenha mais memória e CPU
C. Definir hashfield (ou hashexpression) e hashpartitions nas opções de leitura JDBC do Glue
D. Ativar job bookmarks para que o Glue divida a tabela entre linhas já lidas e linhas novas

R1.4 · Domínio 1: Ingestão e transformação de dados

Uma fintech lê transações de um stream do Kinesis Data Streams com 4 shards por meio de uma função Lambda. Cada lote leva cerca de 2 segundos porque a função consulta um serviço de score de crédito, e a métrica IteratorAge cresce sem parar. A ordem das transações de um mesmo cliente (chave de partição) precisa ser mantida, e a equipe não quer aumentar o número de shards. O que deve ser feito?

A. Configurar concorrência provisionada na função com o dobro do número de shards para aumentar as invocações
B. Aumentar o ParallelizationFactor do event source mapping para processar vários lotes de cada shard ao mesmo tempo
C. Registrar a função como consumidor enhanced fan-out para ampliar a vazão de processamento de cada shard do stream
D. Aumentar o BatchSize para 10.000 registros para que cada invocação leia uma quantidade maior de dados do shard

Respostas comentadas

R1.1. B) Definir DeletionPolicy como Retain no bucket e na tabela e atualizar a stack

Por quê: O atributo DeletionPolicy define o que acontece com um recurso quando a stack é excluída ou ele sai do template: Delete (padrão para a maioria), Retain (mantém o recurso) ou Snapshot (para recursos que suportam). É uma proteção importante para recursos com dados em IaC.

✗ A) Incorreta. UpdateReplacePolicy só atua quando um recurso é substituído em uma atualização, não na exclusão da stack.
✗ C) Incorreta. A detecção de drift apenas compara a configuração; não altera o comportamento na exclusão.
✗ D) Incorreta. A proteção contra encerramento impede a exclusão da stack inteira, o que bloquearia a substituição planejada.

> **Dica de prova:** Excluir a stack mas manter bucket ou tabela com dados: DeletionPolicy Retain.

R1.2. B) Colocar um Amazon RDS Proxy entre as funções e o banco para compartilhar e reutilizar conexões

Por quê: O Amazon RDS Proxy cria um pool de conexões e as reutiliza entre clientes, evitando o custo de abrir conexões novas a cada invocação e protegendo o banco contra excesso de conexões. É o padrão recomendado para aplicações serverless com picos.

✗ A) Incorreta. Esperar mais tempo não reduz o número de conexões e aumenta a duração e o custo das funções.
✗ C) Incorreta. Réplicas de leitura não aceitam gravações; os pedidos continuariam indo para a instância principal.
✗ D) Incorreta. Limitar a concorrência reduz a capacidade de absorver o pico, o que a empresa não aceita.

> **Dica de prova:** Lambda em pico estourando conexões do RDS sem limitar concorrência: RDS Proxy.

R1.3. C) Definir hashfield (ou hashexpression) e hashpartitions nas opções de leitura JDBC do Glue

Por quê: Por padrão, uma leitura JDBC no Glue usa uma partição. As opções hashfield, hashexpression e hashpartitions fazem o Glue dividir a tabela em várias consultas paralelas, aproveitando os executors do job.

✗ A) Incorreta. Mais workers não mudam a leitura JDBC padrão, que usa uma única conexão e uma única partição.

✗ B) Incorreta. Um worker maior acelera pouco uma leitura que continua sequencial em uma task e desperdiça os demais recursos.

✗ D) Incorreta. Bookmarks controlam o que já foi processado entre execuções; não paralelizam a leitura dentro de uma execução.

> **Dica de prova:** Glue lendo tabela JDBC grande em uma task só: hashfield/hashexpression + hashpartitions.

R1.4. B) Aumentar o ParallelizationFactor do event source mapping para processar vários lotes de cada shard ao mesmo tempo

Por quê: Por padrão, a Lambda processa um lote por vez em cada shard. Com ParallelizationFactor de até 10, o event source mapping executa vários lotes do mesmo shard simultaneamente, preservando a ordem por chave de partição. É a forma de escalar o consumo quando o processamento é lento e não se deseja resharding.

✗ A) Incorreta. Sem paralelização, o event source mapping continua processando um lote por shard; instâncias extras ficariam ociosas.

✗ C) Incorreta. O enhanced fan-out amplia a leitura do stream, mas o gargalo é o tempo de processamento da função, não a vazão de leitura.

✗ D) Incorreta. Se o tempo cresce com o número de registros, lotes maiores não aumentam a vazão; com um lote por shard, o atraso continua.

> **Dica de prova:** Lambda atrasada no Kinesis, função lenta, sem aumentar shards e mantendo ordem por chave: ParallelizationFactor.

02

CAPÍTULO 2 · 26% DA PROVA

Domínio 2: Gerenciamento do armazenamento de dados

Com 26% da prova, aqui preciso escolher o armazenamento certo e mantê-lo barato e rápido: classes do S3 e Lifecycle, modelagem no Redshift e no DynamoDB, Data Catalog e crawlers, particionamento e tabelas Apache Iceberg. A versão 1.1 do guia trouxe S3 Tables, índices vetoriais (HNSW, IVF), Bedrock Knowledge Bases e SageMaker Catalog, e esses temas entraram no escopo da prova. Os enunciados dão o padrão de acesso (frequência, latência, tempo de retenção, filtros usados) e pedem o recurso exato. Onde mais erro: confundir DISTKEY com sort key, GSI com LSI, MSCK REPAIR com partition projection, governance com compliance no Object Lock e as opções de saída do crawler. A pergunta-chave é sempre: qual é o padrão de acesso e o que o enunciado proíbe (reescrever, copiar, instalar, gastar RCU)?

Neste capítulo

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

Tópicos-chave

Classes de armazenamento do S3

Glacier Instant Retrieval: leitura em milissegundos, acesso trimestral, mínimo de 90 dias. Glacier Deep Archive: menor custo, restauração em até 48 horas, mínimo de 180 dias. Intelligent-Tiering: padrão de acesso desconhecido ou que muda, sem taxa de recuperação. S3 Express One Zone (directory bucket): latência de um dígito de milissegundo com computação na mesma AZ.

S3 Lifecycle

Expiration apaga objetos atuais por idade (filtro por prefixo ou tag); em bucket versionado ela só cria delete markers. NoncurrentVersionExpiration apaga versões antigas, e a remoção de expired object delete markers limpa marcadores órfãos. AbortIncompleteMultipartUpload remove partes de uploads incompletos, custo invisível na listagem. Por padrão, objetos menores que 128 KB não transicionam: agregue-os em objetos maiores.

Versioning, Object Lock e replicação

Versioning recupera objeto sobrescrito ou excluído por engano. Object Lock compliance impede exclusão até pelo root (WORM regulatório); governance pode ser contornado com s3:BypassGovernanceRetention; legal hold preserva sem data de fim. Cross-Region Replication exige versionamento nos dois buckets e copia cada objeto novo para outra Região.

Distribuição no Redshift

Duas tabelas grandes unidas sempre pela mesma coluna: DISTKEY nessa coluna nas duas. Dimensões pequenas e estáveis redistribuídas nos joins: DISTSTYLE ALL. EVEN espalha em rodízio e obriga redistribuição nos joins. Distribuir por data concentra consultas recentes em poucas fatias.

Sort key, compressão e nós RA3

Filtros por intervalo de data lendo a tabela inteira: sort key na data, para pular blocos pelos zone maps. AZ64 é a codificação para colunas numéricas, DATE e TIMESTAMP. Dados crescendo com computação estável: nós RA3 com Redshift Managed Storage apoiado no S3.

Redshift e o data lake

Schema externo FROM DATA CATALOG (Spectrum) lê tabelas do Glue sem carregar. UNLOAD FORMAT PARQUET + PARTITION BY exporta para o lake; PARALLEL OFF + ORDER BY gera arquivo único ordenado (até 6,2 GB por arquivo). Histórico frio: UNLOAD para o S3 e consulta pelo Spectrum. Materialized view com refresh incremental evita repetir a mesma agregação sobre tabela Iceberg.

Compartilhamento e carga no Redshift

Data sharing isola cargas lendo dados vivos sem cópia (produtor e consumidores). Federated query (CREATE EXTERNAL SCHEMA FROM POSTGRES/MYSQL) lê RDS e Aurora atuais sem copiar. COPY com manifest e mandatory true carrega arquivos exatos de vários buckets; auto-copy (COPY JOB com AUTO ON) carrega cada arquivo novo de um prefixo sem duplicar. DR regional: cópia automática de snapshots para outra Região.

DynamoDB: modelagem e índices

Mesma partition key com sort keys prefixadas (PERFIL, PEDIDO#data) lê entidade e itens relacionados em uma chamada. GSI consulta por atributo fora da chave e pode ser criado em tabela existente, mas só tem leitura eventualmente consistente. LSI mantém a partition key, oferece outra ordenação com leitura fortemente consistente e só pode ser criado junto com a tabela. Partição quente por data: write sharding com sufixo calculado.

DynamoDB: TTL, Streams, DAX e global tables

TTL exige atributo Number em epoch segundos; a exclusão é assíncrona, então filtre itens vencidos na consulta. Streams com NEW_AND_OLD_IMAGES dá valor anterior e novo; para arquivar o que o TTL apagou, filtre REMOVE com userIdentity do serviço. DAX dá microssegundos em leituras repetidas. Global tables dão escrita local em vários continentes; PITR restaura em nova tabela para um segundo exato.

Crawlers do Glue

Exclude patterns em glob ignoram .tmp e _staging; Create a single schema for each S3 path junta pastas compatíveis; Add new columns only preserva tipos curados e ainda adiciona colunas; tabela do catálogo como fonte mantém a tabela aprovada por DDL. Bucket enorme com poucas mudanças: modo de eventos do S3 com notificações para SQS; log em formato próprio: classificador grok. Banco em sub-rede privada: conexão JDBC do Glue na VPC + include path banco/esquema/% (DocumentDB: database/collection). Tabela DynamoDB de produção: amostragem e scanning rate baixo.

Partições no Athena e no Data Catalog

Partition projection calcula partições previsíveis (data e hora) sem buscar no catálogo. MSCK REPAIR TABLE só entende pastas chave=valor; layout sem esse padrão pede ALTER TABLE ADD PARTITION com LOCATION. Índice de partição do Data Catalog acelera milhões de partições também para Spectrum e EMR. Bucketing para coluna de alta cardinalidade dentro da partição; partições de poucos MB pedem particionamento mais grosso.

Apache Iceberg no Athena e no Glue

MERGE INTO aplica upsert de CDC; FOR TIMESTAMP AS OF faz time travel; ALTER TABLE ADD COLUMNS e CHANGE COLUMN mexem só nos metadados; partition evolution troca hours por days sem reescrever o histórico. Exclusão definitiva (LGPD): DELETE + OPTIMIZE REWRITE DATA + VACUUM com retenção reduzida. Z-order agrupa filtros em várias colunas; compactação automática do Data Catalog resolve arquivos pequenos sem agendar nada.

Amazon S3 Tables

Table bucket é um tipo de bucket próprio para tabelas Iceberg, com manutenção como compactação e expiração de snapshots já automática. Para consultar no Athena e no Redshift, ative a integração com serviços de analytics da AWS, que cria o catálogo s3tablescatalog no Glue; crawler e LOCATION de pasta não são o caminho.

Índices e bancos vetoriais

HNSW monta o grafo à medida que os vetores entram, sem treino; recall baixo sem rebuild se corrige aumentando ef_search (m é de construção). IVFFlat só depois de ter dados; bilhões de vetores com memória limitada: IVF + PQ (faiss, com treino). pgvector + HNSW dá busca semântica no Aurora existente; S3 Vectors é a opção mais barata para muitos vetores e poucas consultas.

Amazon Bedrock Knowledge Bases

Hierarchical chunking busca por chunks filhos e entrega ao modelo os chunks pais, unindo precisão e contexto. StartIngestionJob agendado faz sync incremental só do que mudou. Trocar modelo de embeddings ou dimensão exige nova base e novo índice com reingestão; UpdateKnowledgeBase não altera embeddings.

Escolha de banco pelo padrão de acesso

Terabytes com joins e agregações analíticas: Redshift (colunar, MPP). Relações de muitos saltos: Neptune. Redis/Valkey como banco primário durável: MemoryDB. Aplicação MongoDB: DocumentDB. Cassandra com CQL: Keyspaces. BI pesado degradando OLTP no RDS: réplica de leitura.

Migração e transferência de dados

Banco rodando com parada mínima: DMS full load + CDC. Converter esquema heterogêneo sem instalar nada: DMS Schema Conversion, com IA generativa para procedures que as regras não convertem (revisando antes). Teradata com scripts BTEQ: AWS SCT. NFS on-premises recorrente: DataSync. Parceiros enviando por SFTP: servidor do Transfer Family; buscar no SFTP do parceiro: conector SFTP.

SageMaker Catalog: governança de negócio

Glossário de negócio define termos ligados a ativos e colunas; metadata form com regra de enforcement no domain unit obriga metadados antes de publicar; data product agrupa ativos consumidos juntos em uma assinatura; descrições geradas por IA precisam ser aceitas ou editadas antes de publicar. Linhagem vem das execuções do Glue e Redshift e de ferramentas externas por eventos OpenLineage via PostLineageEvent.

Macetes para memorizar

💡 Classes do S3

Instant = instantâneo (ms, 90 dias). Deep = profundo (48 h, 180 dias). Intelligent = não sei o padrão. Express One Zone = latência mínima na mesma AZ.

💡 Redshift: DIST x SORT

DIST decide ONDE a linha mora; SORT decide em que ORDEM ela fica para pular blocos.

💡 Estilos de distribuição

Grande com grande pela mesma coluna = KEY; pequena e estável = ALL; sem padrão = EVEN.

💡 Índices do DynamoDB

LSI = Local, nasce com a tabela, mesma chave, leitura forte. GSI = Global, cria quando quiser, leitura eventual.

💡 Partições

Baixa cardinalidade = partição. Alta cardinalidade = bucketing ou Z-order. Partição minúscula = engrossar (dia ou mês).

💡 Comandos Iceberg

MERGE junta, OPTIMIZE compacta, VACUUM esquece snapshots, AS OF volta no tempo, ALTER só mexe em metadados.

💡 Bancos especializados

Redis durável = MemoryDB; Mongo = DocumentDB; Cassandra = Keyspaces; grafo = Neptune.

💡 Vetores

HNSW cresce sem treino (ajuste ef_search); IVF precisa de dados e treino; muitos vetores e pouca consulta = S3 Vectors.

Tabela de memorização: não confunda

Quando é...	...e quando é

GSI Consultar por atributo fora da chave, em tabela já existente; aceita leitura eventualmente consistente.	**LSI** Mesma partition key com outra ordenação e leitura fortemente consistente; só na criação da tabela.
Object Lock compliance WORM regulatório: ninguém, nem o root, exclui ou encurta a retenção.	**Object Lock governance / legal hold** Governance: quem tem s3:BypassGovernanceRetention contorna. Legal hold: preservação por litígio sem data de fim.
DISTKEY Colocar no mesmo nó as linhas que se juntam pela mesma coluna, evitando redistribuição.	**Sort key** Pular blocos com zone maps em filtros por intervalo, como data.
Partition projection Partições previsíveis (data e hora) consultadas pelo Athena, sem registrar nem buscar no catálogo.	**Índice de partição do Glue Data Catalog** Milhões de partições com valores imprevisíveis, consultadas também por Redshift Spectrum e EMR.
S3 Glacier Instant Retrieval Acesso raro e conhecido (trimestral) com leitura em milissegundos.	**S3 Intelligent-Tiering** Padrão de acesso desconhecido ou que muda, sem taxa de recuperação.
HNSW Tabela vazia ou crescendo continuamente; sem etapa de treino; recall ajustado por ef_search.	**IVF / IVFFlat** Dados já carregados (centróides calculados na criação); IVF + PQ para bilhões de vetores com memória limitada.
AWS DMS Schema Conversion Converter esquema pelo console, sem instalar software, com opção de IA generativa.	**AWS SCT** Aplicativo instalado localmente; usado para data warehouse Teradata com scripts BTEQ para o Redshift. Saiu da lista de serviços da versão 1.1 do guia.

Números que caem

Item	Valor
S3 Glacier Instant Retrieval	**leitura em milissegundos; cobrança mínima de 90 dias**
S3 Glacier Deep Archive	**restauração em até 48 horas; cobrança mínima de 180 dias**
S3 Standard-IA	**cobrança mínima de 30 dias**
Tamanho mínimo padrão para transição no Lifecycle	**128 KB**
UNLOAD com PARALLEL OFF	**arquivos de até 6,2 GB cada**
Atributo de TTL do DynamoDB	**Number em epoch segundos**
Retenção do DynamoDB Streams	**24 horas**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Usar MSCK REPAIR TABLE para registrar pastas como 2026/10/13 ou para consolidar partições Iceberg.
Como não cair: MSCK só reconhece pastas chave=valor de tabelas Hive. Layout sem esse padrão: ALTER TABLE ADD PARTITION com LOCATION. Iceberg: nada de MSCK.

⚠ **Alerta de prova:** Particionar pela coluna de alta cardinalidade (user_id, id_evento) para acelerar o filtro.
Como não cair: Isso cria milhões de pastas e arquivos minúsculos. Partição para baixa cardinalidade; bucketing ou Z-order para a alta.

⚠ **Alerta de prova:** Criar um LSI em tabela existente para consultar por outro atributo.
Como não cair: LSI só nasce com a tabela e mantém a mesma partition key; em tabela existente use GSI.

⚠ **Alerta de prova:** Colocar DAX ou mudar para on-demand para resolver partição quente de escrita.
Como não cair: DAX acelera leituras e on-demand não elimina o limite por partição; escrita quente pede write sharding.

⚠ **Alerta de prova:** Gravar o TTL como String em ISO 8601 ou buscar uma configuração para acelerar a exclusão.
Como não cair: TTL exige Number em epoch segundos e a exclusão é assíncrona; filtre os vencidos na consulta.

⚠ **Alerta de prova:** Aplicar Expiration em bucket versionado para apagar versões antigas.
Como não cair: Expiration em versão atual só cria delete marker; versões antigas exigem NoncurrentVersionExpiration.

⚠ **Alerta de prova:** Escolher Object Lock governance quando nem o root pode apagar.
Como não cair: Governance pode ser contornado com s3:BypassGovernanceRetention; só compliance garante WORM regulatório.

⚠ **Alerta de prova:** Definir a data como DISTKEY para acelerar filtros recentes.
Como não cair: DISTKEY por data concentra a carga em poucas fatias; filtro por intervalo pede sort key.

⚠ **Alerta de prova:** Usar Multi-AZ ou datashare como recuperação de desastre regional do Redshift.
Como não cair: Multi-AZ protege falha de AZ e o consumidor de datashare não guarda cópia; DR regional é cópia de snapshots para outra Região.

⚠ **Alerta de prova:** Executar VACUUM com a retenção padrão para apagar de vez as linhas excluídas de uma tabela Iceberg.
Como não cair: Sem reescrita, o snapshot atual ainda aponta para os arquivos originais; faça OPTIMIZE REWRITE DATA e depois VACUUM com retenção reduzida.

⚠ **Alerta de prova:** Selecionar ignorar mudanças (LOG) no crawler para preservar tipos curados.
Como não cair: LOG preserva os tipos mas deixa de adicionar colunas novas; a opção certa é Add new columns only.

⚠ **Alerta de prova:** Atualizar a base de conhecimento com UpdateKnowledgeBase para trocar o modelo de embeddings.
Como não cair: Essa API só muda nome, descrição e role; trocar modelo ou dimensão exige nova base, novo índice e reingestão.

⚠ **Alerta de prova:** Trocar a regra de Lifecycle para Deep Archive ou Intelligent-Tiering para arquivar bilhões de arquivos de poucos KB.
Como não cair: O padrão de 128 KB vale para todas as classes e o Intelligent-Tiering não monitora objetos menores; agregue em objetos maiores antes.

⚠ **Alerta de prova:** Apontar o Multi-AZ do RDS para receber as consultas de BI.
Como não cair: A standby de uma instância Multi-AZ não aceita leituras; BI degradando OLTP pede réplica de leitura.

Como cai na prova

📝 **Como cai:** Descreve o padrão de acesso (frequência, latência, retenção) e pede a classe do S3 mais barata que atende.
Como resolver: Primeiro elimine o que não cumpre a latência (Flexible e Deep Archive levam minutos ou horas). Depois, entre as que cumprem, fique com a de menor armazenamento para aquela frequência; se o padrão é desconhecido, Intelligent-Tiering.

📝 **Como cai:** Consulta lenta no Redshift com joins ou filtros e opções de DISTKEY, SORTKEY, ALL, EVEN e codificação.
Como resolver: Join entre tabelas grandes = DISTKEY comum; dimensão pequena = ALL; filtro por intervalo = sort key. Compressão só reduz tamanho, não pula blocos.

📝 **Como cai:** Tabela Iceberg com requisito de não reescrever dados (renomear coluna, mudar partição, ver estado antigo, upsert).
Como resolver: Descarte CTAS e cópias. Use a operação nativa: ALTER TABLE, partition evolution, FOR TIMESTAMP AS OF, MERGE INTO. Desconfie de qualquer ação que mexa nos arquivos por fora do Iceberg.

📝 **Como cai:** Crawler produzindo resultado errado (tabelas demais, esquema poluído, tipos desfeitos, lentidão).
Como resolver: Mapeie sintoma a opção: tabela por pasta = single schema; .tmp e _staging = exclude patterns; tipos desfeitos = Add new columns only; listagem lenta = modo de eventos do S3; formato próprio = grok.

📝 **Como cai:** Modelagem no DynamoDB com novo padrão de consulta.
Como resolver: Pergunte: a tabela já existe? precisa de leitura fortemente consistente? a partition key é a mesma? Isso separa GSI de LSI; leitura de entidade + itens relacionados em uma chamada = mesma partition key com sort key prefixada.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Acesso trimestral com abertura imediata e menor custo	**S3 Glacier Instant Retrieval**
Retenção de anos, acesso raríssimo, horas de espera aceitáveis	**S3 Glacier Deep Archive**
Fatura do S3 maior que a soma dos objetos listados	**AbortIncompleteMultipartUpload no Lifecycle**
Nem o usuário root pode apagar por 5 anos	**Object Lock modo compliance**
Milhares de partições previsíveis por data e hora no Athena	**Partition projection**
Filtros em várias colunas com a mesma importância em Iceberg	**Compactação Z-order**
Iceberg com manutenção automática em bucket próprio para tabelas	**Amazon S3 Tables (table bucket)**
Ver a tabela como estava em uma data e hora	**Time travel com FOR TIMESTAMP AS OF**
DynamoDB com latência de microssegundos para itens lidos repetidamente	**DAX**

Valor anterior e novo de cada alteração no DynamoDB	**DynamoDB Streams com NEW_AND_OLD_IMAGES**
DynamoDB para Redshift sem pipeline e sem gastar RCU	**Zero-ETL com PITR ativado**
Crawler desfazendo tipos ajustados à mão, mas colunas novas devem entrar	**Add new columns only**
Precisão de chunks pequenos com contexto de seções grandes	**Hierarchical chunking**
Qual dataset e quais jobs geraram o modelo implantado	**SageMaker ML Lineage Tracking (dados entre tabelas e jobs: linhagem do SageMaker Catalog)**

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

- MSCK REPAIR TABLE só entende pastas chave=valor e não se aplica a Iceberg.
- Object Lock compliance é o único que nem o root contorna.
- Em Iceberg, nunca mexa nos arquivos por fora (Lifecycle, UNLOAD, restauração de versões do S3): isso corrompe a tabela.
- Snowball Edge não está mais disponível para novos clientes; transferência online recorrente é DataSync.

Essenciais para a prova (checklist)

☐ Sei escolher a classe do S3 pela latência, frequência de acesso e cobrança mínima.

☐ Sei montar regras de Lifecycle para objetos atuais, versões antigas, delete markers e uploads incompletos.

☐ Sei a diferença entre Object Lock compliance, governance e legal hold.

☐ Sei escolher DISTKEY, DISTSTYLE ALL, EVEN e sort key no Redshift.

☐ Sei quando usar Spectrum, federated query, data sharing, auto-copy e UNLOAD.

☐ Sei modelar no DynamoDB com GSI, LSI, write sharding, TTL, Streams, DAX e global tables.

☐ Sei configurar crawlers (exclude patterns, single schema, Add new columns only, JDBC, eventos do S3).

☐ Sei quando usar partition projection, índice de partição, ADD PARTITION e bucketing.

☐ Sei as operações Iceberg: MERGE, time travel, ALTER TABLE, partition evolution, OPTIMIZE e VACUUM.

☐ Sei escolher índice vetorial (HNSW, IVF, PQ) e configurar chunking e sync no Bedrock Knowledge Bases.

Questões de revisão do capítulo

R2.1 · Domínio 2: Gerenciamento do armazenamento de dados

Uma operadora de telecom mantém uma tabela Apache Iceberg de chamadas no AWS Glue Data Catalog. As consultas filtram ora por id_antena, ora por numero_origem, ora pelas duas colunas, com a mesma frequência. Ordenar por uma única coluna beneficia sobretudo os filtros feitos nela. Qual estratégia de compactação reduz os arquivos lidos nos dois tipos de filtro?

A. Particionar a tabela pelas duas colunas, criando uma pasta para cada combinação de valores
B. Binpack com tamanho-alvo maior, juntando os arquivos pequenos sem mudar a ordem dos registros
C. Sort com id_antena e depois numero_origem, ordenando hierarquicamente os arquivos da partição
D. Z-order nas duas colunas, intercalando os valores para agrupar dados próximos em cada arquivo

R2.2 · Domínio 2: Gerenciamento do armazenamento de dados

Uma varejista quer catalogar no AWS Glue Data Catalog uma tabela do Amazon DynamoDB de produção, com capacidade provisionada e 2 bilhões de itens, cujo esquema é estável. Um teste de crawler levou horas e causou throttling na aplicação. Como configurar o crawler para catalogar a tabela com o menor impacto?

A. Ativar a amostragem de dados e definir um scanning rate baixo da capacidade de leitura
B. Apontar o crawler para o DynamoDB Streams da tabela em vez de ler os itens armazenados
C. Aumentar o scanning rate para 1,5 para que o crawler termine a leitura em menos tempo
D. Mudar a tabela para o modo on-demand para que o crawler leia sem limite de capacidade

R2.3 · Domínio 2: Gerenciamento do armazenamento de dados

Um crawler do AWS Glue aponta para s3://telemetria/veiculos/, onde arquivos JSON ficam em pastas ano=2024, ano=2025 e ano=2026. Como os campos mudaram um pouco entre os anos, o crawler criou uma tabela por pasta, e os analistas querem uma única tabela particionada por ano com todas as colunas. Qual configuração resolve?

A. Configurar exclude patterns para que o crawler deixe de lado as pastas dos anos mais antigos
B. Marcar a opção Create a single schema for each S3 path nas opções avançadas do crawler
C. Adicionar um classificador grok customizado que force um esquema comum nos arquivos JSON
D. Mudar a política de mudança de esquema para só registrar em log, sem atualizar a tabela

R2.4 · Domínio 2: Gerenciamento do armazenamento de dados

Uma holding mantém o data lake e o AWS Glue Data Catalog em uma conta central. Analistas de uma conta de marketing precisam consultar essas tabelas no Amazon Athena da própria conta, referenciando-as como catalogo.banco.tabela, sem copiar metadados. As políticas de IAM e a política de recurso do catálogo já estão configuradas. O que falta fazer na conta de marketing?

A. Executar na conta de marketing um crawler apontando para os buckets da conta central do lake
B. Copiar o catálogo com scripts de exportação e importação de metadados entre as duas contas
C. Criar um conector do Athena Federated Query em Lambda para ler o catálogo da outra conta
D. Registrar no Athena uma fonte de dados do tipo Glue Data Catalog com o ID da conta central

Respostas comentadas

R2.1. D) Z-order nas duas colunas, intercalando os valores para agrupar dados próximos em cada arquivo

Por quê: Os otimizadores do AWS Glue Data Catalog para Iceberg oferecem as estratégias binpack, sort e Z-order. O Z-order intercala os bits dos valores de várias colunas e mantém próximos os registros semelhantes em todas elas, indicado para filtros multidimensionais de peso igual.

✗ A) Incorreta. Colunas de alta cardinalidade como partição criam milhões de partições e arquivos minúsculos.
✗ B) Incorreta. O binpack só junta arquivos pequenos; não organiza os dados para os filtros.
✗ C) Incorreta. A ordenação hierárquica favorece a primeira coluna; filtros só pela segunda continuam lendo muitos arquivos.

> **Dica de prova:** Filtros em várias colunas com a mesma importância numa tabela Iceberg = compactação Z-order.

R2.2. A) Ativar a amostragem de dados e definir um scanning rate baixo da capacidade de leitura

Por quê: Para fontes DynamoDB, o crawler do AWS Glue oferece amostragem de dados (lê uma amostra em vez da tabela inteira) e scanning rate, o percentual de RCUs provisionadas que ele pode usar (0,1 a 1,5; padrão de 0,5 em tabelas provisionadas). A recomendação é usar o crawler com tabelas em modo provisionado.

✗ B) Incorreta. O crawler não usa DynamoDB Streams como fonte de catalogação.
✗ C) Incorreta. Um scanning rate maior consome mais capacidade de leitura e agrava o throttling.
✗ D) Incorreta. A documentação recomenda não usar crawlers com tabelas on-demand, e a leitura completa continuaria disputando a tabela com a aplicação.

> **Dica de prova:** Crawler em tabela DynamoDB de produção = amostragem + scanning rate baixo.

R2.3. B) Marcar a opção Create a single schema for each S3 path nas opções avançadas do crawler

Por quê: Por padrão, o crawler considera compatibilidade dos dados e similaridade dos esquemas e pode criar uma tabela por pasta. A opção Create a single schema for each S3 path (TableGroupingPolicy CombineCompatibleSchemas) gera uma tabela única, com as colunas combinadas, quando os dados são compatíveis.

✗ A) Incorreta. Excluir pastas tiraria anos da tabela, o que os analistas não querem.

✗ C) Incorreta. Grok serve para formatos de texto sem estrutura conhecida; os arquivos já são JSON, e o problema é o agrupamento em tabelas.

✗ D) Incorreta. A política de mudança de esquema controla atualizações de tabelas existentes; não junta pastas em uma tabela.

> **Dica de prova:** Crawler criando uma tabela por pasta com esquemas parecidos = Create a single schema for each S3 path.

R2.4. D) Registrar no Athena uma fonte de dados do tipo Glue Data Catalog com o ID da conta central

Por quê: Com as permissões de IAM e a política de recurso do AWS Glue configuradas, a conta consumidora registra o catálogo da conta dona no Athena como um DataCatalog do tipo GLUE com o parâmetro catalog-id. Depois, as consultas referenciam o catálogo pelo nome registrado.

✗ A) Incorreta. O crawler criaria um catálogo duplicado na conta de marketing, contrariando o requisito.

✗ B) Incorreta. Copiar metadados cria outra versão a manter sincronizada, exatamente o que se quer evitar.

✗ C) Incorreta. Conectores de Federated Query servem para fontes fora do catálogo do Glue, como bancos e APIs.

> **Dica de prova:** Athena consultando o Glue Data Catalog de outra conta = registrar data source GLUE com catalog-id da conta dona.

03

CAPÍTULO 3 · 22% DA PROVA

Domínio 3: Operações e suporte de dados

Este domínio (22%) mede se eu sei operar o pipeline no dia a dia: monitorar, alertar, investigar falhas, garantir qualidade dos dados e analisar com SQL. Cai muito SQL no Athena e no Redshift (funções de janela, UNNEST, ROLLUP, PIVOT, MERGE, TRY_CAST, funções aproximadas) e muita operação de CloudWatch Logs, Step Functions, MWAA, EMR e Glue. As alternativas erradas costumam ser recursos reais aplicados ao problema errado: VACUUM no lugar de ANALYZE, SQA no lugar de concurrency scaling, metric filter no lugar de subscription filter. Também confundo fácil Glue Data Quality (rulesets, ações, resultados por linha) com DataBrew (profile job, receita). A pergunta que me guia: o que exatamente o sintoma revela e qual ferramenta enxerga esse dado?

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

Tópicos-chave

Otimização e custo no Athena

Query exhausted resources: LIMIT no ORDER BY, funções aproximadas como approx_distinct e menos dados lidos. Workgroup com limite de dados escaneados por consulta barra consultas caras; capacity reservation dá DPUs dedicadas para concorrência previsível; query result reuse reaproveita consultas idênticas sobre dados que mudam pouco. Notebooks do Athena for Apache Spark controlam custo com session idle timeout e MaxConcurrentDpus.

SQL do Athena

CROSS JOIN UNNEST transforma array em linhas; ARRAY_AGG faz o inverso. TRY_CAST ... IS NULL encontra valores que não convertem sem falhar a consulta. TABLESAMPLE SYSTEM dá amostra rápida (BERNOULLI é mais uniforme, mas lê tudo). Prepared statements com EXECUTE USING evitam injeção de SQL.

Funções de janela e agregações

Acumulado: SUM OVER (ORDER BY dia ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW). Média móvel de 7 dias: AVG OVER (ORDER BY dia ROWS BETWEEN 6 PRECEDING AND CURRENT ROW). Diferença para o dia anterior: valor - LAG(valor) OVER (PARTITION BY loja ORDER BY dia). Deduplicar ficando com o mais recente: ROW_NUMBER() OVER (PARTITION BY chave ORDER BY data DESC) = 1.

Agregações especiais no Redshift

GROUP BY ROLLUP gera subtotais hierárquicos e total geral; PIVOT transforma linhas em colunas (UNPIVOT faz o inverso); MEDIAN dá o valor típico com outliers, ao contrário da média; CTE recursiva (WITH RECURSIVE) percorre hierarquias de profundidade variável. MERGE faz upsert da staging para a tabela final.

Manutenção e desempenho do Redshift

Estimativas de linhas erradas ou estatísticas ausentes: ANALYZE; espaço e ordenação: VACUUM. Picos de concorrência com fila: concurrency scaling; consultas curtas presas atrás de longas: short query acceleration; consultas descontroladas: query monitoring rules com log, hop ou abort. Uso esporádico: Redshift Serverless; uso constante e alto: provisionado com nós reservados.

Recursos de SQL e auditoria no Redshift

Stored procedure chamada com CALL e agendada no query scheduler executa várias etapas com controle de erro. Late-binding view (WITH NO SCHEMA BINDING) não quebra quando a tabela externa é recriada. Texto das consultas e conexões para auditoria: enable_user_activity_logging = true no parameter group.

CloudWatch Logs e alarmes

Alertar sobre um texto nos logs: metric filter + alarme com ação em tópico do SNS (alarme não lê texto). Enviar eventos em tempo quase real: subscription filter para Firehose, Kinesis, Lambda ou OpenSearch. Investigar horas recentes em vários log groups: Logs Insights (stats count(*) by bin(5m)). Período passado em arquivo: export task para o S3. Custo crescendo: política de retenção, pois o padrão é nunca expirar.

Alarmes e notificações

Alarme ruidoso por picos isolados: M de N datapoints to alarm (ex.: 3 de 5). Input transformer no alvo do EventBridge formata o evento sem código. Retry policy + DLQ do SQS no alvo evitam perder eventos quando o destino cai. Alertas no Slack ou Teams sem código: Amazon Q Developer in chat applications assinando o tópico do SNS.

Operação do Step Functions

Express não tem histórico nativo: habilite logging no CloudWatch Logs (nível ERROR ou ALL). Standard guarda histórico por 90 dias; para mais tempo, logging ALL com dados de execução e retenção no log group. Redrive retoma uma execução Standard a partir do estado que falhou. O tipo do workflow não pode ser alterado depois de criado.

Operação do Amazon MWAA

DAG que não aparece: logs de DAG processing no CloudWatch (erro de importação). Biblioteca faltando: requirements.txt no S3 e atualização do ambiente; código próprio: plugins.zip. Sensores esperando horas e ocupando workers: modo deferrable, que delega a espera ao triggerer. Tarefas presas em queued: mais workers ou classe maior. DAG que depende de dado atualizado por outra: datasets do Airflow.

Operação do Amazon EMR

Container killed by YARN for exceeding memory limits: aumentar spark.executor.memoryOverhead. Falta de capacidade do tipo de instância: instance fleets com vários tipos e várias sub-redes ou AZs. Cluster esquecido ligado: auto-termination por ociosidade. Enviar trabalhos a um cluster compartilhado: steps com ActionOnFailure CANCEL_AND_WAIT (TERMINATE_CLUSTER derruba o cluster).

Operação de jobs do Glue

No space left on device no shuffle: shuffle no S3 (--write-shuffle-files-to-s3) ou workers maiores. Achar a linha que falhou: job run insights (--enable-job-insights). Skew em junção: Glue 4.0 com AQE; chave muito quente: salting. ConcurrentRunsExceededException: aumentar Maximum concurrency (padrão 1). Avisar que o job está demorando sem matar: delay notification threshold + EventBridge + SNS.

AWS Glue Data Quality

Recomendação de regras dá ponto de partida baseado nos dados. Evaluate Data Quality com a ação Fail job without loading to target data impede a carga quando a regra falha (a ação padrão None carrega mesmo assim). rowLevelOutcomes separa as linhas ruins para quarentena sem falhar o job. Regras dinâmicas do DQDL (last, avg) acompanham o histórico em vez de limites fixos.

DataBrew, Data Wrangler e amostragem

Profile job do DataBrew mostra distribuição, ausentes e distintos sem código; ruleset associado valida a cada execução; receita publicada vira jobs de receita para outros datasets; dataset com caminho parametrizado pega o arquivo das últimas 24 horas. Data Wrangler gera o relatório Data Quality and Insights para ML. Amostra que preserva proporção de grupos: amostragem estratificada.

Investigação com CloudTrail e outros

Sem trilha, quem fez o quê nos últimos 90 dias: Event history. DeleteObject ou GetObject não aparece: data events do S3 não estavam habilitados. Throttling no DynamoDB abaixo da capacidade total: chave quente, descoberta com Contributor Insights. Provar que origem e destino batem no DMS: data validation (premigration assessment é antes de migrar).

Lambda, SDK e APIs em operação

Invocação assíncrona que falha: destino on-failure ou DLQ (bisect é só para streams). Lambda acionada pelo S3 gravando no mesmo bucket: loop recursivo, separe bucket ou prefixo de saída. Throttling de API pelo SDK: retry mode adaptive ou standard com backoff. Listagem parando em 1.000 itens: paginator. Athena via SDK: StartQueryExecution, GetQueryExecution até estado final, GetQueryResults paginado.

Visualização e APIs de dados

Dashboard lento sobre Athena com dados diários: importar no SPICE com refresh agendado. Cada gerente vê só sua região: row-level security no dataset do Quick Sight. API Gateway: usage plans com chave de API dão limites e cotas por cliente; cache do estágio reduz chamadas repetidas ao backend. Busca de texto livre e painéis em logs: OpenSearch Service; consulta pontual no S3: Athena.

Macetes para memorizar

Redshift: ANALYZE x VACUUM

ANALYZE Avalia estatísticas; VACUUM Varre espaço.

Redshift: filas

Muita gente = concurrency scaling. Curta presa atrás de longa = SQA. Consulta louca = QMR abort.

CloudWatch Logs

Contar = metric filter. Mandar = subscription filter. Investigar = Logs Insights. Arquivar passado = export.

Janelas SQL

UNBOUNDED = acumula tudo; 6 PRECEDING = média de 7; LAG olha para trás, LEAD para frente; ROW_NUMBER = 1 deduplica.

Glue Data Quality

Recomendar para começar, Fail without loading para barrar, rowLevelOutcomes para separar, regra dinâmica para acompanhar o histórico.

Arrays no Athena

UNNEST abre o array em linhas; ARRAY_AGG fecha linhas em array.

Falhas da Lambda

Assíncrona = on-failure ou DLQ. Stream = bisect + on-failure. Síncrona = quem chamou trata.

Tabela de memorização: não confunda

Quando é...	...e quando é
ANALYZE Estatísticas ausentes ou estimativas de linhas erradas no plano do otimizador.	**VACUUM** Recuperar espaço de linhas excluídas e reordenar dados; não atualiza estatísticas.
Concurrency scaling Centenas de consultas simultâneas em pico, com fila; adiciona capacidade temporária.	**Short query acceleration** Consultas curtas presas atrás de relatórios longos; prioriza as curtas sem aumentar capacidade.
Metric filter + alarme Contar ocorrências de um texto e notificar ao passar de um limite.	**Subscription filter** Entregar o conteúdo dos eventos em tempo quase real para Firehose, Kinesis, Lambda ou OpenSearch.
Logs Insights Investigar rapidamente horas recentes em vários log groups com consultas.	**Export task para o S3** Entregar todos os eventos de um período passado em arquivos para ferramentas próprias.
Workgroup com limite de dados escaneados Barrar consultas caras de um grupo de usuários.	**Capacity reservation** Garantir DPUs dedicadas para concorrência previsível em um workgroup.
Glue Data Quality Regras DQDL em jobs e no catálogo, com ações de falha, resultados por linha e regras dinâmicas.	**Glue DataBrew** Perfil exploratório sem código, receitas visuais e rulesets avaliados pelo profile job.
Destino on-failure / DLQ Invocação assíncrona da Lambda (ex.: notificação do S3) que falha após as tentativas.	**BisectBatchOnFunctionError** Event source mapping de streams (Kinesis, DynamoDB Streams) com registro venenoso no lote.

Números que caem

Item	Valor
Histórico de execuções Standard do Step Functions	**90 dias (a cota pode ser reduzida para 30, não ampliada)**
Novas tentativas de invocação assíncrona da Lambda	**de 0 a 2**
Chaves por resposta de list_objects_v2	**máximo de 1.000**
Maximum concurrency padrão de um job do Glue	**1**
Partições por CTAS ou INSERT INTO no Athena	**até 100 (complete o histórico com INSERT INTO em blocos)**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Rodar VACUUM quando o plano mostra estimativas de linhas erradas.
Como não cair: Estatísticas ausentes se corrigem com ANALYZE; VACUUM cuida de espaço e ordenação.

⚠ **Alerta de prova:** Criar alarme do CloudWatch direto sobre o log group procurando um texto.
Como não cair: Alarmes avaliam métricas; primeiro crie um metric filter com o padrão, depois o alarme sobre a métrica.

⚠ **Alerta de prova:** Criar subscription filter direto para o SNS ou para reprocessar o histórico.
Como não cair: Subscription filter entrega a Kinesis, Firehose, Lambda ou OpenSearch e só eventos novos; histórico sai com export task.

⚠ **Alerta de prova:** Mudar a state machine Express para Standard nas configurações para ganhar histórico.
Como não cair: O tipo é imutável depois da criação; em Express, habilite logging no CloudWatch Logs.

⚠ **Alerta de prova:** Pedir aumento da retenção do histórico de execuções Standard para 1 ano.
Como não cair: O histórico fica 90 dias (a cota só pode ser reduzida); para guardar mais, logging ALL com retenção no log group.

⚠ **Alerta de prova:** Reduzir o poke_interval ou adicionar schedulers para liberar workers presos por sensores.
Como não cair: No modo poke o sensor ocupa o slot o tempo todo; use modo deferrable (ou reschedule).

⚠ **Alerta de prova:** Instalar biblioteca no MWAA por SSH ou com BashOperator.
Como não cair: O MWAA não dá SSH e a importação falha na análise da DAG, antes das tarefas; use requirements.txt e atualize o ambiente.

⚠ **Alerta de prova:** Usar a ação padrão do Evaluate Data Quality esperando que a carga seja bloqueada.
Como não cair: Com a ação None, o job continua e carrega; para impedir, Fail job without loading to target data.

⚠ **Alerta de prova:** Usar o resultado agregado do ruleset ou a regra RowCount para separar linhas ruins.
Como não cair: O agregado vale para o dataset inteiro; separar linhas é rowLevelOutcomes com filtro.

⚠ **Alerta de prova:** Configurar bisect on error para uma Lambda invocada por notificações do S3.
Como não cair: Notificação do S3 é invocação assíncrona, sem event source mapping; use destino on-failure ou DLQ. Novas tentativas assíncronas vão de 0 a 2.

⚠ **Alerta de prova:** Passar MaxKeys=250000 ou aumentar retries quando a listagem do S3 para em 1.000 objetos.
Como não cair: MaxKeys só reduz a página; a resposta vem paginada e o paginator segue o ContinuationToken.

⚠ **Alerta de prova:** Usar TABLESAMPLE BERNOULLI ou WHERE rand() para economizar leitura no Athena.
Como não cair: Os dois avaliam linha a linha e leem a tabela toda; para economizar leitura, TABLESAMPLE SYSTEM.

⚠ **Alerta de prova:** Usar ActionOnFailure TERMINATE_CLUSTER em steps de um cluster compartilhado.
Como não cair: Isso derruba o cluster no primeiro erro; use CANCEL_AND_WAIT.

⚠ **Alerta de prova:** Consultar um event data store do CloudTrail Lake recém-criado para achar quem excluiu algo ontem.
Como não cair: O event data store só guarda eventos a partir da criação (ou importação); sem trilha, use o Event history (90 dias).

Como cai na prova

📝 **Como cai:** Mensagem de erro literal no enunciado (YARN memory, No space left on device, ConcurrentRunsExceeded, HIVE_TOO_MANY_OPEN_PARTITIONS, erro 1023).
Como resolver: Associe a mensagem ao recurso específico e descarte o aumento genérico de capacidade. Esses cenários premiam quem conhece o parâmetro exato.

📝 **Como cai:** Pede uma consulta SQL e as alternativas são variações de funções de janela ou agregação.
Como resolver: Confira três coisas: PARTITION BY certo (por entidade, não por data), ORDER BY presente e o frame (UNBOUNDED para acumulado, N-1 PRECEDING para média móvel).

📝 **Como cai:** Queremos ser avisados quando algo acontecer nos logs ou no pipeline, sem código.
Como resolver: Defina se é contagem (metric filter + alarme + SNS), conteúdo em tempo real (subscription filter) ou evento de serviço (regra do EventBridge com input transformer).

📝 **Como cai:** Problema de desempenho no Redshift ou Athena com várias otimizações reais.
Como resolver: Leia o sintoma: fila em pico, consulta curta presa, estimativa errada, consulta repetida, custo por TB escaneado. Cada sintoma tem um recurso; descarte os que atacam outra causa.

📝 **Como cai:** Qualidade de dados: impedir carga, separar registros ruins ou descobrir quais regras escrever.
Como resolver: Barrar = Fail job without loading; separar = rowLevelOutcomes; descobrir = recomendação de regras; limite que muda = regra dinâmica; explorar sem código = profile job do DataBrew.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Query exhausted resources at this scale factor	**approx_distinct, LIMIT no ORDER BY e menos dados lidos**
Container killed by YARN for exceeding memory limits	**Aumentar spark.executor.memoryOverhead**
No space left on device no shuffle do Glue	**Shuffle no S3 ou workers maiores**
ConcurrentRunsExceededException	**Aumentar Maximum concurrency do job**
Uma tarefa roda por horas porque uma chave concentra os dados	**Data skew: AQE no Glue 4.0 ou salting**
DAG nova não aparece na interface do Airflow	**Logs de DAG processing no CloudWatch**
Execução Standard falhou no estado 10 de 12	**Redrive**
Alarme dispara por picos de um minuto que se resolvem sozinhos	**M de N datapoints to alarm**
Formatar o e-mail do evento do EventBridge sem código	**Input transformer**
Mesmas consultas repetidas sobre dados que mudam uma vez por dia	**Query result reuse (no Quick Sight: SPICE com refresh)**
Valores do usuário concatenados em consultas do Athena	**Prepared statements com EXECUTE USING**
Não aparece DeleteObject no CloudTrail	**Data events do S3 não habilitados**
Throttling no DynamoDB abaixo da capacidade total	**Chave quente; Contributor Insights**
Limites e cotas diferentes por parceiro em uma API	**Usage plans com chaves de API**

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

- Alarme do CloudWatch não lê texto de log: sempre metric filter antes.
- O tipo da state machine (Standard ou Express) não muda depois de criado.
- A ação padrão do Evaluate Data Quality (None) não impede a carga.
- Sensor em modo poke prende o worker; deferrable libera.
- Retenção padrão de log group é nunca expirar: defina retenção para conter custo.

Essenciais para a prova (checklist)

- ☐ Sei escrever acumulado, média móvel, LAG e deduplicação com ROW_NUMBER.
- ☐ Sei usar UNNEST, TRY_CAST, TABLESAMPLE, ROLLUP, PIVOT, MEDIAN e MERGE.
- ☐ Sei quando aplicar ANALYZE, VACUUM, concurrency scaling, SQA e query monitoring rules.
- ☐ Sei diferenciar metric filter, subscription filter, Logs Insights e export task.
- ☐ Sei configurar logging e redrive no Step Functions.
- ☐ Sei diagnosticar DAG que não aparece, biblioteca faltando e tarefas em queued no MWAA.
- ☐ Sei resolver erros de memória, capacidade e disco no EMR e no Glue.
- ☐ Sei usar as ações e saídas do Glue Data Quality e os recursos do DataBrew.
- ☐ Sei tratar falhas de invocação assíncrona e loop recursivo na Lambda.
- ☐ Sei controlar custo e concorrência no Athena com workgroups, capacity reservation e result reuse.

Questões de revisão do capítulo

R3.1 · Domínio 3: Operações e suporte de dados

Um job do AWS Glue com muitas junções e agregações sobre 3 TB falha nos estágios de shuffle com o erro No space left on device nos executores. A equipe não quer reescrever a lógica do job. Qual ajuste resolve de forma mais direta?

A. Reduzir o número de workers para diminuir a quantidade de arquivos de shuffle
B. Mudar o tipo de worker para G.025X, o mais econômico para jobs em lote
C. Habilitar o shuffle no Amazon S3 com o parâmetro --write-shuffle-files-to-s3
D. Habilitar job bookmarks para processar os dados em partes a cada execução

R3.2 · Domínio 3: Operações e suporte de dados

A equipe de operações quer buscar texto livre e filtrar em segundos os logs de aplicação de dezenas de microsserviços dos últimos 30 dias, com painéis interativos e alertas, e ingestão contínua a partir de várias fontes. Qual serviço é o mais adequado como plataforma de análise desses logs?

A. Amazon Athena sobre arquivos de log compactados em CSV
B. Amazon OpenSearch Service com o OpenSearch Dashboards
C. Amazon DynamoDB com um índice secundário por campo
D. Amazon Redshift provisionado com tabelas por microsserviço

R3.3 · Domínio 3: Operações e suporte de dados

Um workflow Standard do AWS Step Functions carrega dados regulatórios todos os dias. A auditoria exige guardar por 1 ano, para cada execução, a entrada e a saída de cada estado, para provar quais arquivos foram processados e com quais parâmetros. A equipe percebeu que o histórico de execuções não atende. O que configurar?

A. Data events do AWS CloudTrail para as chamadas StartExecution da state machine diária
B. Logging da state machine no nível ALL com dados de execução e log group com retenção de 1 ano
C. Pedido de aumento da retenção do histórico de execuções para 365 dias na cota do Step Functions
D. Métricas ExecutionsSucceeded e ExecutionTime com painel no CloudWatch por um ano

R3.4 · Domínio 3: Operações e suporte de dados

Um job do AWS Glue grava novas partições no Amazon S3 em horários variáveis. Depois de cada execução bem-sucedida, um crawler precisa ser iniciado para atualizar o Data Catalog. A equipe não quer manter um orquestrador só para isso. Qual é a solução com menos componentes?

A. Agendamento do crawler a cada 15 minutos para sempre pegar as partições novas gravadas pelo job
B. Notificação de eventos do S3 para cada objeto gravado que inicia o crawler por uma função Lambda
C. Regra do EventBridge para Glue Job State Change com estado SUCCEEDED que inicia o crawler
D. Ambiente do Amazon MWAA com uma DAG que executa o job e depois o crawler, em sequência fixa

Respostas comentadas

R3.1. C) Habilitar o shuffle no Amazon S3 com o parâmetro --write-shuffle-files-to-s3

Por quê: Jobs Spark gravam dados de shuffle no disco local. Quando o volume excede esse disco, surge No space left on device. O AWS Glue oferece shuffle no Amazon S3 (--write-shuffle-files-to-s3, e o Cloud Shuffle Storage Plugin no Glue 3.0 ou posterior); outra opção é usar workers maiores, como G.2X, com mais disco.

✗ A) Incorreta. Menos workers concentram mais dados por worker e pioram a falta de disco local.
✗ B) Incorreta. O G.025X é voltado a jobs de streaming de baixo volume e tem ainda menos recursos.
✗ D) Incorreta. Bookmarks controlam leitura incremental; o job ainda faria os mesmos shuffles grandes em cada execução.

> **Dica de prova:** Glue sem espaço em disco no shuffle: shuffle no S3 ou workers maiores.

R3.2. B) Amazon OpenSearch Service com o OpenSearch Dashboards

Por quê: O Amazon OpenSearch Service é usado para análise de logs e observabilidade: recebe dados por Firehose, OpenSearch Ingestion ou subscription filters do CloudWatch Logs, indexa para busca e agregação e oferece Dashboards e alertas. Para consultas pontuais e baratas sobre logs no S3, o Athena é a alternativa.

✗ A) Incorreta. O Athena consulta logs sob demanda, mas não oferece busca interativa em segundos nem painéis nativos.

✗ C) Incorreta. O DynamoDB é um banco chave-valor e não oferece busca de texto livre nem agregações flexíveis.

✗ D) Incorreta. O Redshift é voltado a SQL analítico sobre dados estruturados, não a busca de texto livre em logs.

> **Dica de prova:** Busca de texto e painéis interativos em logs: OpenSearch Service. Consulta pontual no S3: Athena.

R3.3. B) Logging da state machine no nível ALL com dados de execução e log group com retenção de 1 ano

Por quê: O Step Functions mantém o histórico de execuções Standard por 90 dias. Para auditoria e rastreabilidade de longo prazo, habilita-se o logging da state machine no CloudWatch Logs (níveis ALL, ERROR, FATAL ou OFF), com a opção de incluir os dados de execução (entradas e saídas), e define-se a retenção do log group conforme a exigência.

✗ A) Incorreta. O CloudTrail registra quem iniciou a execução, mas não a entrada e a saída de cada estado.

✗ C) Incorreta. O histórico de execuções Standard fica disponível por 90 dias; a cota pode ser reduzida para 30 dias, não ampliada para um ano.

✗ D) Incorreta. Métricas agregadas não mostram os dados processados em cada execução.

> **Dica de prova:** Rastreabilidade de execuções por mais de 90 dias: logging da state machine no CloudWatch Logs (ALL + dados de execução) com retenção.

R3.4. C) Regra do EventBridge para Glue Job State Change com estado SUCCEEDED que inicia o crawler

Por quê: O AWS Glue emite eventos no Amazon EventBridge, como Glue Job State Change (SUCCEEDED, FAILED, TIMEOUT) e Glue Crawler State Change. Regras com padrões de evento acionam alvos (Lambda, Step Functions, SNS) de forma reativa, sem polling nem orquestrador dedicado.

✗ A) Incorreta. Rodar o crawler a cada 15 minutos gera execuções inúteis e ainda deixa o catálogo atrasado em relação ao fim do job.

✗ B) Incorreta. O job grava muitos objetos; uma invocação por objeto iniciaria o crawler várias vezes e falharia por ele já estar em execução.

✗ D) Incorreta. O MWAA funcionaria, mas é justamente o orquestrador dedicado que a equipe quer evitar.

> **Dica de prova:** Reagir ao fim de um job do Glue: regra do EventBridge para Glue Job State Change.

04

CAPÍTULO 4 · 18% DA PROVA

Domínio 4: Segurança e governança de dados

É o domínio de menor peso (18%), mas os cenários são traiçoeiros porque todas as alternativas parecem seguras. Preciso dominar IAM (menor privilégio, ABAC, RBAC, permissions boundary), KMS (key policy, acesso entre contas, envelope encryption), rede (endpoints, security groups), Lake Formation, permissões no Redshift, Macie, CloudTrail, Config e a governança do SageMaker Unified Studio. A prova separa bem detectar de impedir (Config x SCP), quem chamou de como estava (CloudTrail x Config) e filtrar linhas de mascarar colunas (RLS x dynamic data masking). Onde mais erro é no acesso entre contas: quase sempre são duas pontas, a política na conta dona do recurso e a política IAM na conta que acessa. CloudTrail Lake está fechado a novos clientes desde 31/05/2026, então só vale como resposta para quem já o usa.

Neste capítulo

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

Tópicos-chave

Menor privilégio no IAM

Política gerenciada ampla (AmazonS3FullAccess) se troca por política customizada com ações mínimas e ARNs de prefixo. Para reduzir uma role ampla ao que foi usado, a geração de política do IAM Access Analyzer usa a atividade do CloudTrail. Credenciais em código ou variável de ambiente se substituem pela execution role do serviço.

ABAC x RBAC, grupos e variáveis

Muitas equipes e recursos que mudam sempre: ABAC comparando aws:PrincipalTag com a tag do recurso. Poucas funções estáveis: RBAC com role ou permission set por função. Mesmas permissões para vários usuários IAM: grupo IAM. Uma política para muitos usuários, cada um com seu prefixo: variável ${aws:username} no ARN e na condição s3:prefix.

Delegação segura

Permissions boundary obrigatória deixa engenheiros criarem roles sem escalar privilégios. Service role do CloudFormation passada com iam:PassRole tira de quem implanta as permissões dos recursos. Acessar outra conta pela CLI sem chaves: perfil com role_arn + source_profile.

SCPs e AWS Organizations

SCP com aws:RequestedRegion (exceto serviços globais) restringe regiões em toda a organização, inclusive para administradores das contas. SCP negando s3:PutReplicationConfiguration e backup:StartCopyJob impede cópias para outras regiões. Trilha da organização criada na conta de gerenciamento registra todas as contas, incluindo as futuras, sem poder ser desativada localmente.

AWS KMS: key policy e acesso entre contas

AccessDenied no KMS com IAM correto: a key policy precisa permitir o uso pela conta ou pela role. Ler SSE-KMS de outra conta: bucket policy + key policy na conta dona + política IAM com kms:Decrypt na conta que lê. Chaves gerenciadas pela AWS (aws/s3, aws/rds) não têm key policy alterável nem servem para outra conta: use chave gerenciada pelo cliente.

Criptografia de dados grandes e em escala

A API Encrypt aceita só até 4 KB: dados grandes usam envelope encryption com GenerateDataKey e cifragem local. Milhões de objetos com SSE-KMS gerando custo e throttling: S3 Bucket Keys. Nem a AWS pode ver o dado em claro: criptografia do lado do cliente (S3 Encryption Client) antes do upload; SSE-KMS, SSE-S3 e SSE-C recebem o dado em claro.

Proteções do S3

Exigir HTTPS: bucket policy com Deny quando aws:SecureTransport for false. Bucket compartilhado com política enorme: um S3 Access Point por equipe, com política e origem de rede próprias. Impedir exfiltração para buckets de terceiros: política do VPC endpoint permitindo só aws:ResourceOrgID da organização.

Endpoints e rede

Sub-rede privada acessando S3 ou DynamoDB sem custo extra: gateway endpoint (existe só para esses dois). Secrets Manager, KMS ou STS sem NAT: interface endpoint com DNS privado. Recursos com IP variável: regra de entrada referenciando o security group de origem. Conexão JDBC do Glue em VPC: regra de entrada autorreferenciada para todo TCP no security group.

Secrets Manager x Parameter Store

Credencial que muda e precisa de rotação: Secrets Manager, referenciado na conexão do Glue. Configuração comum não secreta: Parameter Store String (nível standard). Senha master do RDS sem ninguém conhecer: gerenciamento pelo RDS no Secrets Manager. Rotação sem janela de falha: usuários alternados com segredo de superusuário separado; segredo lido em toda invocação da Lambda: extensão Parameters and Secrets com cache.

AWS Lake Formation

Data filters restringem linhas e colunas da mesma tabela (segurança em nível de célula). LF-Tags (TBAC) escalam grants para milhares de tabelas por classificação. Resource link + grants na conta consumidora tornam visível um banco compartilhado. Hybrid access mode com opt-in permite adotar o Lake Formation sem quebrar quem usa IAM; AccessDenied no Athena com IAM correto indica falta de grant no Lake Formation (e lakeformation:GetDataAccess na política IAM).

Controle de acesso no Redshift

Esconder colunas sem view: GRANT SELECT por coluna (REVOKE de coluna não anula SELECT na tabela). Mesma coluna exibida diferente por perfil: dynamic data masking; filtrar linhas: row-level security. Muitos usuários com o mesmo conjunto de privilégios: role de banco. Tabelas futuras: ALTER DEFAULT PRIVILEGES. Compartilhar só um recorte: view no datashare.

Amazon Macie e PII no pipeline

Visão contínua e barata de onde há PII: descoberta automatizada por amostras; análise completa: job direcionado. Formato próprio da empresa: custom data identifier (regex + palavras-chave); allow list diz o que ignorar. Mascarar PII durante o ETL: Detect Sensitive Data do Glue. Integrar achados ao Lake Formation: EventBridge + Lambda aplicando LF-Tags.

Pseudonimização

Esconder o identificador mantendo junções: hash SHA-256 com salt secreto guardado à parte. No DataBrew, DETERMINISTIC_ENCRYPT mantém junções e permite reverter com a chave; CRYPTOGRAPHIC_HASH preserva junções sem reversão. Números aleatórios por linha e truncamento quebram as junções.

Auditoria com CloudTrail

Provar que o log não foi adulterado: validação de integridade com arquivos digest assinados. SQL sobre anos de logs no S3: Athena com partition projection por data. Event history cobre 90 dias de eventos de gerenciamento. CloudTrail Lake permite SQL em anos de eventos e federação do event data store para o Athena, mas está fechado a novos clientes desde 31/05/2026.

AWS Config, Flow Logs e logs

Como estava a configuração em cada momento: histórico do AWS Config (CloudTrail mostra quem chamou a API). Detectar e corrigir desvio: regras gerenciadas do Config + remediação automática com SSM Automation. O que trafegou na rede: VPC Flow Logs consultados no Athena. Logs com chave da empresa: chave KMS gerenciada pelo cliente no log group; centralizar logs de várias contas: subscription filter para destination na conta central.

SageMaker Unified Studio e Catalog

Domínio (organização) > domain units (áreas com administração delegada) > projetos (equipes com membros e recursos). Login corporativo: IAM Identity Center federado ao IdP, com grupos SSO no domínio. Ativo só no inventário não aparece na busca: falta publicar. Acesso a ativo de outro projeto: assinatura pelo projeto consumidor e aprovação do produtor.

Residência de dados e arquiteturas de governança

A AWS não move conteúdo para fora da região escolhida; a residência é quebrada por recursos que o cliente configura (CRR, global tables, cópias de backup ou snapshot). Multi-AZ, chave KMS regional e gateway endpoint mantêm os dados na região. Bedrock com residência: inference profile geográfico. Domínios donos dos dados com governança central: data mesh; dados nos produtores e catálogo no centro: hub central com Lake Formation.

Modelo de responsabilidade compartilhada

Em serviços serverless e gerenciados (Glue, MSK, RDS), a AWS cuida de hardware, data centers e patches do ambiente. O cliente continua responsável por dados, permissões da role, acesso a origem e destino e configuração de criptografia. Para reduzir patching, prefira o serviço gerenciado (MSK) a rodar o software em EC2.

Macetes para memorizar

💡 Acesso entre contas

Duas portas para abrir: a do dono (bucket policy ou key policy) e a de quem entra (política IAM).

💡 Impedir x detectar

SCP é cadeado, Config é câmera: um impede, o outro grava e avisa.

💡 Quem x como

CloudTrail responde QUEM fez; Config responde COMO estava; Flow Logs responde O QUE trafegou.

💡 Endpoints

Gateway só para dois: S3 e DynamoDB, sem custo e pela tabela de rotas. Os demais serviços usam interface (PrivateLink), cobrada por hora e por dados.

💡 Redshift: proteger dados

Linha = RLS. Coluna inteira = GRANT por coluna. Pedaço da coluna = masking. Futuras = DEFAULT PRIVILEGES.

💡 Lake Formation

Tag para escala, filtro para célula, link para outra conta, híbrido para migrar sem quebrar.

💡 Macie

Custom identifier = o que procurar; allow list = o que ignorar; descoberta automatizada = onde olhar sempre.

Tabela de memorização: não confunda

Quando é...	...e quando é
ABAC Muitas equipes e recursos que surgem o tempo todo; uma política compara tags do principal e do recurso.	**RBAC** Poucas funções estáveis com o mesmo conjunto de permissões para todas as pessoas da função.

Row-level security Restringir quais linhas cada perfil enxerga.	**Dynamic data masking** Mostrar a mesma coluna de forma diferente por perfil (ex.: só os 4 últimos dígitos), sem cópia.
AWS Config Como estava a configuração de um recurso em cada momento; detectar e remediar desvio.	**AWS CloudTrail** Quem chamou qual API e quando; integridade dos logs com arquivos digest.
SCP Impedir antes que a ação aconteça, inclusive para administradores das contas-membro.	**Regra do AWS Config** Detectar depois que o recurso foi criado ou alterado (e corrigir com remediação).
Gateway endpoint S3 e DynamoDB, sem custo extra, associado às tabelas de rotas.	**Interface endpoint** Demais serviços (Secrets Manager, KMS, STS) e acesso a partir de redes locais (também existe para o S3), com DNS privado; cobrado por hora e por dados.
Secrets Manager Credenciais que exigem rotação gerenciada.	**Parameter Store** Configuração comum não secreta (nomes de buckets, URLs, flags).
LF-Tags Escalar permissões para muitas tabelas por classificação ou domínio.	**Data filters** Restringir linhas e colunas dentro de uma tabela.

Números que caem

Item	Valor
Payload máximo da API Encrypt do KMS	**4 KB (não ajustável)**
Event history do CloudTrail	**90 dias de eventos de gerenciamento**
CloudTrail Lake	**fechado a novos clientes desde 31/05/2026**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Usar regra do AWS Config para impedir a criação de recursos fora das regiões aprovadas.
Como não cair: Config detecta depois; impedir em toda a organização é SCP com aws:RequestedRegion.

⚠ **Alerta de prova:** Compartilhar a chave aws/s3 ou aws/rds com outra conta.
Como não cair: A key policy de chaves gerenciadas pela AWS não pode ser alterada; copie ou criptografe com chave gerenciada pelo cliente liberada na key policy.

⚠ **Alerta de prova:** Ampliar a política IAM para kms:* quando o Decrypt dá AccessDenied.
Como não cair: Se a key policy não autoriza a conta ou a role, nenhuma política IAM resolve; corrija a key policy.

⚠ **Alerta de prova:** Usar row-level security para esconder ou mascarar colunas no Redshift.
Como não cair: RLS filtra linhas. Esconder coluna inteira: GRANT por coluna. Mostrar parcial: dynamic data masking.

⚠ **Alerta de prova:** Dar GRANT SELECT na tabela e depois REVOKE nas colunas sensíveis.
Como não cair: O REVOKE de coluna não retira o acesso dado no nível da tabela; conceda SELECT só nas colunas permitidas.

⚠ **Alerta de prova:** Rodar GRANT SELECT ON ALL TABLES de novo para incluir as tabelas criadas toda semana.
Como não cair: ON ALL TABLES vale só para as existentes; para tabelas futuras use ALTER DEFAULT PRIVILEGES.

⚠ **Alerta de prova:** Aplicar LF-Tag nas linhas da tabela ou usar resource link para filtrar dados.
Como não cair: LF-Tags vão em bancos, tabelas e colunas; resource links são ponteiros. Filtrar linhas e colunas é data filter.

⚠ **Alerta de prova:** Remover IAMAllowedPrincipals de todas as tabelas para começar a usar o Lake Formation.
Como não cair: Isso faz o Lake Formation valer para todos e quebra jobs antigos sem grants; use hybrid access mode com opt-in.

⚠ **Alerta de prova:** Criar allow list ou regra de supressão no Macie para detectar a matrícula interna.
Como não cair: Allow list indica o que ignorar e supressão arquiva achados; detectar formato próprio é custom data identifier.

⚠ **Alerta de prova:** Guardar as chaves de acesso no Secrets Manager ou criptografá-las com KMS na variável de ambiente da Lambda.
Como não cair: Isso mantém credenciais de longo prazo desnecessárias; dê as permissões à execution role e remova as chaves.

⚠ **Alerta de prova:** Considerar Multi-AZ, chave KMS regional ou gateway endpoint como violação de residência de dados.
Como não cair: Tudo isso fica na mesma região; violam residência CRR, global tables e cópias de backup ou snapshot para outra região.

⚠ **Alerta de prova:** Usar SSE-KMS ou SSE-C quando o parceiro exige que nem a AWS veja o dado em claro.
Como não cair: Nos dois o S3 recebe o conteúdo em claro para criptografar no servidor; só a criptografia do lado do cliente atende.

⚠ **Alerta de prova:** Usar o inference profile global do Bedrock para ganhar vazão com dados que não podem sair da UE.
Como não cair: O perfil global processa em qualquer região comercial; com residência, use o inference profile geográfico.

⚠ **Alerta de prova:** Proteger uma API só para VPCs com API regional e WAF ou chave de API.
Como não cair: A API regional segue pública e chave de API não é autorização; use API privada com resource policy por aws:SourceVpce.

Como cai na prova

📝 **Como cai:** AccessDenied mesmo com a política IAM correta.
Como resolver: Procure a outra camada de autorização: key policy do KMS, grant do Lake Formation, resource-based policy da Lambda, bucket policy. O erro quase nunca se resolve ampliando a política IAM.

📝 **Como cai:** Requisito de impedir algo em toda a organização, inclusive para administradores das contas.
Como resolver: Escolha SCP. Descarte regras do Config (detectam depois), políticas IAM locais (podem ser alteradas) e ferramentas de descoberta como Macie.

📝 **Como cai:** Multiple response sobre acesso entre contas ou configuração em duas partes.
Como resolver: Marque uma ação na conta dona do recurso e outra na conta que acessa; desconfie de opções que trocam a chave para aws/ ou dão permissões amplas.

📝 **Como cai:** Proteger dados sensíveis mantendo a utilidade (junções, visualização parcial, reversão por um time).
Como resolver: Defina o que precisa sobreviver: junção sem reversão = hash com salt; junção com reversão = criptografia determinística; ver parte = masking; esconder linhas = RLS ou data filter.

📝 **Como cai:** Governança no SageMaker Unified Studio (colaboração, delegação, descoberta, acesso).
Como resolver: Equipe colaborando = projeto; área com admins próprios = domain unit; ninguém acha o ativo = publicar; acesso a ativo de outro projeto = assinatura com aprovação; login corporativo = IAM Identity Center.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Reduzir role ampla ao que foi realmente usado	**Geração de política do IAM Access Analyzer**
Desenvolvedores criam roles sem poder escalar privilégios	**Permissions boundary obrigatória**
40 equipes, novas surgindo todo mês, acesso por tag	**ABAC com aws:PrincipalTag**
Arquivo de 200 MB com KMS falhando no Encrypt	**Envelope encryption com GenerateDataKey**
Throttling ou custo alto do KMS com milhões de objetos	**S3 Bucket Keys**
Ler objetos SSE-KMS de outra conta	**Key policy na conta dona + kms:Decrypt na política IAM da conta que lê**
Lambda em sub-rede privada sem NAT lendo segredo	**Interface endpoint do Secrets Manager com DNS privado**
Analistas do Sul só veem regiao = SUL e sem a coluna cpf	**Data filter do Lake Formation**
1.200 tabelas e novas toda semana, acesso por classificação	**LF-Tags (TBAC)**
Banco compartilhado aparece para o admin, mas não para os analistas	**Resource link + grants na conta consumidora**
Provar que logs de auditoria não foram alterados	**Validação de integridade do CloudTrail (digest)**
Linha do tempo das mudanças de configuração de um bucket	**Histórico de configuração do AWS Config**
Licenciar e receber dados de terceiros na fatura da AWS	**AWS Data Exchange (API de terceiro: Data Exchange for APIs)**
Ativo existe no projeto, mas ninguém o encontra no catálogo	**Publicar o ativo do inventário no catálogo**

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

- Chaves gerenciadas pela AWS (aws/...) nunca servem para acesso entre contas.
- Data events do S3 não vêm habilitados por padrão em uma trilha.
- No serverless, criptografia, IAM e dados continuam sendo responsabilidade do cliente.

Essenciais para a prova (checklist)

- ☐ Sei montar acesso entre contas a objetos SSE-KMS (bucket policy, key policy e IAM).
- ☐ Sei quando usar ABAC, RBAC, grupos IAM, variáveis de política e permissions boundary.
- ☐ Sei restringir regiões e cópias de dados com SCP e diferenciar isso do AWS Config.
- ☐ Sei escolher entre gateway endpoint, interface endpoint, NAT gateway e política do VPC endpoint.
- ☐ Sei quando usar Secrets Manager, Parameter Store, rotação gerenciada pelo RDS e usuários alternados.
- ☐ Sei aplicar data filters, LF-Tags, resource links e hybrid access mode no Lake Formation.
- ☐ Sei aplicar GRANT por coluna, RLS, dynamic data masking, roles e ALTER DEFAULT PRIVILEGES no Redshift.
- ☐ Sei configurar o Macie (descoberta automatizada, custom data identifiers, allow lists) e o Detect Sensitive Data do Glue.
- ☐ Sei usar CloudTrail (trilha da organização, data events, integridade) e Config (histórico, remediação).
- ☐ Sei organizar domínio, domain units, projetos, publicação e assinatura no SageMaker Unified Studio.

Questões de revisão do capítulo

R4.1 · Domínio 4: Segurança e governança de dados

Vários jobs do AWS Glue usam uma conexão JDBC com usuário e senha digitados nas propriedades da conexão. A equipe de segurança passou a rotacionar a senha do banco a cada 30 dias, e os jobs falham depois de cada rotação até alguém atualizar a conexão. Como eliminar o problema?

A. Guardar a senha em um parâmetro do job, atualizado pelo pipeline de CI/CD a cada mês
B. Gravar a senha criptografada com o AWS KMS no script e descriptografá-la no início
C. Configurar a conexão para obter as credenciais de um segredo do AWS Secrets Manager
D. Usar um usuário IAM com chaves de acesso de longo prazo gravadas na conexão JDBC

R4.2 · Domínio 4: Segurança e governança de dados

Uma role usada por um pipeline de ingestão foi criada há um ano com permissões amplas para não falhar. A segurança quer substituí-la por uma política com apenas as ações e recursos que o pipeline realmente usou nos últimos 90 dias, sem levantar tudo à mão. Qual recurso da AWS ajuda?

A. Geração de política do IAM Access Analyzer com base na atividade do CloudTrail
B. Amazon GuardDuty analisando as chamadas de API feitas com a role do pipeline de ingestão
C. AWS Config com a regra que verifica políticas com Action e Resource coringa
D. Simulador de políticas do IAM testando cada ação do pipeline uma por uma

R4.3 · Domínio 4: Segurança e governança de dados

Cinquenta jobs de dados leem parâmetros de configuração não secretos, como nomes de buckets, URLs de APIs internas e flags por ambiente. Esses valores mudam poucas vezes por ano, não precisam de rotação e a empresa quer o menor custo, com versionamento e controle por IAM. Onde guardar?

A. Parâmetros do tipo String no Systems Manager Parameter Store (nível standard)
B. Tags de recurso aplicadas aos buckets e lidas pelos jobs com a API
C. Variáveis gravadas no código-fonte de cada job em um repositório Git
D. Segredos no AWS Secrets Manager com rotação automática configurada a cada 30 dias

R4.4 · Domínio 4: Segurança e governança de dados

Um bucket compartilhado do data lake atende 15 equipes, e a bucket policy passou de 300 linhas, perto do limite de tamanho e difícil de revisar. Cada equipe acessa prefixos diferentes, algumas só a partir de VPCs específicas. Como simplificar o controle de acesso?

A. Dividir a bucket policy em várias declarações menores dentro da mesma política
B. Criar uma role IAM por equipe e retirar a bucket policy do bucket compartilhado
C. Ativar ACLs de objeto e conceder acesso por equipe em cada objeto do bucket
D. Criar um S3 Access Point por equipe, com política e origem de rede próprias

Respostas comentadas

R4.1. C) Configurar a conexão para obter as credenciais de um segredo do AWS Secrets Manager
Por quê: Em vez de usuário e senha fixos, as conexões do Glue aceitam credenciais guardadas no Secrets Manager. Como o segredo é lido em tempo de execução, a rotação automática do Secrets Manager não quebra os jobs. A role do job precisa de secretsmanager:GetSecretValue no segredo (e kms:Decrypt se ele usar chave gerenciada pelo cliente).
✗ A) Incorreta. Continua dependendo de um processo manual sincronizado com a rotação e deixa a senha visível nos parâmetros do job.
✗ B) Incorreta. A senha continua fixa no script; a cada rotação seria preciso criptografar e publicar o código de novo.
✗ D) Incorreta. Chaves de acesso de longo prazo são uma má prática e não resolvem a autenticação no banco com a senha rotacionada.

> **Dica de prova:** Credencial de banco que muda: referencie um segredo do Secrets Manager na conexão, nunca a senha fixa.

R4.2. A) Geração de política do IAM Access Analyzer com base na atividade do CloudTrail

Por quê: O IAM Access Analyzer gera políticas com base na atividade de acesso registrada pelo CloudTrail (trilha) de uma role ou usuário em um período. Também oferece o último acesso a serviços, validação de políticas e verificação de acesso externo e não utilizado, apoiando o refinamento contínuo do menor privilégio.

✗ B) Incorreta. O GuardDuty detecta ameaças, não produz políticas de menor privilégio.

✗ C) Incorreta. A regra do Config aponta políticas amplas, mas não gera a política mínima.

✗ D) Incorreta. O simulador testa políticas existentes, mas exigiria saber de antemão todas as ações usadas.

> **Dica de prova:** Reduzir uma role ampla ao que foi usado: geração de política do IAM Access Analyzer (atividade do CloudTrail).

R4.3. A) Parâmetros do tipo String no Systems Manager Parameter Store (nível standard)

Por quê: Parameter Store e Secrets Manager guardam valores com controle por IAM. O Parameter Store (String, StringList e SecureString) é adequado para configurações e segredos simples, com nível standard gratuito; o Secrets Manager adiciona rotação gerenciada, replicação entre regiões e integração nativa com bancos, com custo por segredo.

✗ B) Incorreta. Tags servem para organizar recursos e custos, não como repositório central de configuração.

✗ C) Incorreta. Valores no código exigem nova implantação a cada mudança e se espalham por 50 jobs.

✗ D) Incorreta. O Secrets Manager é cobrado por segredo e é indicado para credenciais que precisam de rotação, não para configurações comuns.

> **Dica de prova:** Configuração comum: Parameter Store. Credencial com rotação: Secrets Manager.

R4.4. D) Criar um S3 Access Point por equipe, com política e origem de rede próprias

Por quê: Amazon S3 Access Points simplificam o acesso a buckets compartilhados: cada access point tem política própria (por exemplo, restrita a um prefixo e a ações específicas) e pode aceitar só tráfego de uma VPC. A bucket policy delega ao controle dos access points da conta, evitando uma política gigante.

✗ A) Incorreta. Reorganizar as declarações não reduz o tamanho total nem a complexidade da política única.

✗ B) Incorreta. Roles ajudam, mas sem política no recurso fica difícil impor restrições de rede por equipe no bucket.

✗ C) Incorreta. ACLs são desaconselhadas (desabilitadas por padrão) e muito mais difíceis de gerenciar.

> **Dica de prova:** Bucket compartilhado com política enorme: um S3 Access Point por aplicação ou equipe.

DIA DA PROVA

Estratégia de prova

A DEA-C01 é uma prova de cenários: cada enunciado descreve uma empresa, um sintoma ou requisito e uma restrição decisiva, como menor esforço operacional, sem código, sem copiar dados, sem reescrever, menor custo ou sem instalar nada. As alternativas erradas são serviços e parâmetros reais, plausíveis, mas que resolvem outro problema ou ferem a restrição; muitas vezes uma delas funciona, porém com Lambda de cola ou processo manual. A prova valoriza o recurso nativo e gerenciado do serviço e o parâmetro exato para o sintoma. São 65 questões, das quais 15 não pontuadas e não identificadas, então trate todas como valendo.

Como responder

✓ Administre o tempo: são 65 questões em 130 minutos, cerca de 2 minutos cada; não se prenda a uma questão difícil: elimine o que puder, fique com a melhor restante e siga em frente.

✓ Leia primeiro a última frase do enunciado e grife a restrição (menor esforço operacional, sem código, menor custo, sem copiar, sem reescrever); ela costuma eliminar duas alternativas.

✓ Nos itens de múltipla resposta, conte quantas o enunciado pede e procure peças que se completam (ex.: política na conta dona + política na conta que acessa; NAT gateway + rota).

✓ Quando uma alternativa funciona mas exige escrever e manter uma Lambda, procure se existe recurso nativo que faz o mesmo; se existir, ele é a resposta.

✓ Separe o sintoma da causa antes de olhar as opções: throttling em um shard só, uma task lenta, fila em pico, estatística ausente. Cada causa tem um recurso específico.

✓ Desconfie de aumentar capacidade genericamente (mais workers, mais memória, mais shards) quando o enunciado aponta problema de forma de leitura, distribuição ou configuração.

✓ Não deixe nenhuma em branco: questão não respondida conta como errada e não há penalidade por chute.

✓ Distribua a revisão pelos pesos: Ingestão e transformação 34%, Armazenamento 26%, Operações 22%, Segurança e governança 18%; a pontuação é compensatória e o mínimo é 720 na escala de 100 a 1000.

Erros que derrubam candidatos

✗ Escolher a alternativa que funciona tecnicamente, ignorando a restrição de menor esforço operacional ou de não usar código.

✗ Confundir recursos de nomes parecidos: concorrência reservada x provisionada, GSI x LSI, metric filter x subscription filter, ANALYZE x VACUUM, Object Lock compliance x governance.

✗ Achar que ampliar a política IAM resolve AccessDenied quando o bloqueio está na key policy do KMS ou no Lake Formation.

✗ Tratar detecção como prevenção: Config, Macie e GuardDuty não impedem nada; quem impede é SCP, política ou bloqueio nativo.

✗ Esquecer limites fixos que derrubam alternativas: Lambda em 15 minutos, Express em 5 minutos, Encrypt do KMS em 4 KB, gateway endpoint só para S3 e DynamoDB.

✗ Marcar recursos que saíram ou estão fechados a novos clientes (CloudTrail Lake, Snowball Edge) para clientes novos.

✗ Escolher soluções que copiam ou reescrevem dados quando o enunciado pede consulta no lugar, como CTAS em Iceberg ou cópia de catálogo.

Na semana da prova

- ☐ Refaça as provas simuladas e, em cada erro, escreva o par certo x errado que você confundiu; revise essa lista todo dia.
- ☐ Revise os números que eliminam alternativas: 15 minutos da Lambda, 5 minutos do Express, 4 KB do KMS, 1 MB/s ou 1.000 registros/s por shard, 90 e 180 dias mínimos do Glacier, 128 KB do Lifecycle.
- ☐ Repasse os temas novos da versão 1.1 do guia: Apache Iceberg e S3 Tables, índices vetoriais (HNSW, IVF), Bedrock Knowledge Bases, SageMaker Unified Studio e SageMaker Catalog.
- ☐ Pratique leitura de enunciado: em cada cenário, identifique sintoma, causa e restrição antes de olhar as alternativas.
- ☐ Na véspera, revise só as seções lembrar e macetes de cada domínio.

REVISÃO FINAL

Folha de cola

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

1. Domínio 1: Ingestão e transformação de dados

‣ Um shard com throttling e os outros ociosos → **Chave de partição de alta cardinalidade**

‣ Vários consumidores no mesmo stream ficando atrasados → **Enhanced fan-out**

‣ Registros de ~100 bytes estourando o limite de registros por shard → **Agregação com a KPL**

‣ Job do Glue reprocessa todos os arquivos a cada execução → **Job bookmarks + transformation_ctx + job.commit**

‣ Centenas de milhares de arquivos de poucos KB deixando o Glue lento → **groupFiles inPartition + groupSize (no EMR: S3DistCp --groupBy)**

⚠ Enhanced fan-out só amplia a leitura dos consumidores. Escrita se resolve com chave de alta cardinalidade, mais shards, on-demand ou agregação da KPL.

⚠ Com a mesma chave quente, os registros continuam caindo no mesmo shard; o conserto é a chave de alta cardinalidade.

⚠ Bookmarks só controlam o que já foi processado entre execuções. Paralelismo JDBC é hashfield/hashpartitions; malformados é errorsAsDynamicFrame.

2. Domínio 2: Gerenciamento do armazenamento de dados

‣ Acesso trimestral com abertura imediata e menor custo → **S3 Glacier Instant Retrieval**

‣ Retenção de anos, acesso raríssimo, horas de espera aceitáveis → **S3 Glacier Deep Archive**

‣ Fatura do S3 maior que a soma dos objetos listados → **AbortIncompleteMultipartUpload no Lifecycle**

‣ Nem o usuário root pode apagar por 5 anos → **Object Lock modo compliance**

‣ Milhares de partições previsíveis por data e hora no Athena → **Partition projection**

⚠ MSCK só reconhece pastas chave=valor de tabelas Hive. Layout sem esse padrão: ALTER TABLE ADD PARTITION com LOCATION. Iceberg: nada de MSCK.

⚠ Isso cria milhões de pastas e arquivos minúsculos. Partição para baixa cardinalidade; bucketing ou Z-order para a alta.

⚠ LSI só nasce com a tabela e mantém a mesma partition key; em tabela existente use GSI.

3. Domínio 3: Operações e suporte de dados

‣ Query exhausted resources at this scale factor → **approx_distinct, LIMIT no ORDER BY e menos dados lidos**

‣ Container killed by YARN for exceeding memory limits → **Aumentar spark.executor.memoryOverhead**

‣ No space left on device no shuffle do Glue → **Shuffle no S3 ou workers maiores**

‣ ConcurrentRunsExceededException → **Aumentar Maximum concurrency do job**

‣ Uma tarefa roda por horas porque uma chave concentra os dados → **Data skew: AQE no Glue 4.0 ou salting**

⚠ Estatísticas ausentes se corrigem com ANALYZE; VACUUM cuida de espaço e ordenação.

⚠ Alarmes avaliam métricas; primeiro crie um metric filter com o padrão, depois o alarme sobre a métrica.

⚠ Subscription filter entrega a Kinesis, Firehose, Lambda ou OpenSearch e só eventos novos; histórico sai com export task.

4. Domínio 4: Segurança e governança de dados

‣ Reduzir role ampla ao que foi realmente usado → **Geração de política do IAM Access Analyzer**

‣ Desenvolvedores criam roles sem poder escalar privilégios → **Permissions boundary obrigatória**

‣ 40 equipes, novas surgindo todo mês, acesso por tag → **ABAC com aws:PrincipalTag**

‣ Arquivo de 200 MB com KMS falhando no Encrypt → **Envelope encryption com GenerateDataKey**

‣ Throttling ou custo alto do KMS com milhões de objetos → **S3 Bucket Keys**

⚠ Config detecta depois; impedir em toda a organização é SCP com aws:RequestedRegion.

⚠ A key policy de chaves gerenciadas pela AWS não pode ser alterada; copie ou criptografe com chave gerenciada pelo cliente liberada na key policy.

⚠ Se a key policy não autoriza a conta ou a role, nenhuma política IAM resolve; corrija a key policy.

GABARITO

Respostas do diagnóstico

D1. D) Na pasta processing-failed do bucket de destino, com o código e a mensagem de erro de cada registro → revise o capítulo 1

Por quê: Na transformação de dados do Amazon Data Firehose, registros com resultado ProcessingFailed ou cuja invocação falhou após as tentativas são entregues no S3 na pasta processing-failed (ou no prefixo de erro configurado), com metadados do erro, permitindo análise e reprocessamento.

✗ A) Incorreta. O Firehose não oferece reenvio de registros com falha pelo console a partir do buffer.
✗ B) Incorreta. O CloudTrail registra chamadas de API, não o conteúdo de registros de dados com falha.
✗ C) Incorreta. O Firehose invoca a função de forma síncrona; a DLQ de invocações assíncronas não recebe esses registros.

> **Dica de prova:** Registros sumindo no Firehose com Lambda de transformação: olhar a pasta processing-failed no S3.

D2. A) Usar a pandas API on Spark (pyspark.pandas), que mantém a sintaxe do pandas com execução distribuída → revise o capítulo 1

Por quê: Na escolha de linguagem e biblioteca, pandas é eficiente em um único nó, enquanto PySpark distribui o processamento. A pandas API on Spark (pyspark.pandas), disponível no AWS Glue 4.0, permite escalar código pandas para grandes volumes com pouca alteração.

✗ B) Incorreta. Reescrever em Scala com RDDs exige aprender outra linguagem e API, o oposto do requisito.
✗ C) Incorreta. Um job Python shell roda em um único nó com memória limitada; o problema de memória continua.
✗ D) Incorreta. Processar em lotes na Lambda exige reescrever a lógica e quebra operações que dependem da base inteira.

> **Dica de prova:** Código pandas sem memória para o volume total: pandas API on Spark no Glue 4.0+.

D3. A) Ativar o auto scaling do job e definir 60 como número máximo de workers → revise o capítulo 1

Por quê: AWS Glue Auto Scaling, disponível a partir do Glue 3.0, ajusta dinamicamente o número de workers durante a execução de jobs Spark, até o máximo definido. Isso elimina a necessidade de dimensionar manualmente e reduz custo quando a carga varia entre execuções ou entre estágios.

✗ B) Incorreta. Flex reduz o preço por DPU-hora, mas o job continua reservando 60 workers nos dias de pouco volume.
✗ C) Incorreta. Isso muda o requisito de desempenho das segundas-feiras em vez de ajustar a capacidade ao volume.
✗ D) Incorreta. G.025X é destinado a jobs de streaming de baixo volume e não comporta o pico das segundas-feiras.

> **Dica de prova:** Job do Glue com carga variável e workers ociosos: auto scaling com máximo de workers.

D4. B) S3 Glacier Instant Retrieval, com recuperação em milissegundos e mínimo de 90 dias → revise o capítulo 2

Por quê: O S3 Glacier Instant Retrieval atende dados de longa duração acessados raramente, cerca de uma vez por trimestre, com recuperação em milissegundos. Tem duração mínima de cobrança de 90 dias, armazenamento mais barato que o Standard-IA e custo de recuperação maior.

✗ A) Incorreta. Atende à latência, mas o armazenamento custa mais que o Glacier Instant Retrieval para acesso trimestral.

✗ C) Incorreta. Mesmo a recuperação expedita leva minutos, o que não atende à abertura imediata do exame.

✗ D) Incorreta. O Deep Archive leva horas para restaurar os objetos.

> **Dica de prova:** Acesso trimestral + leitura em milissegundos + menor custo = S3 Glacier Instant Retrieval.

D5. B) Usar o método IVF do faiss com product quantization (PQ), treinando o modelo antes → revise o capítulo 2

Por quê: No OpenSearch k-NN, HNSW oferece alto recall com grande consumo de memória. O IVF (faiss) agrupa vetores em listas com k-means, exige treino e usa menos memória; com product quantization, a compressão reduz drasticamente a memória em troca de perda de recall.

✗ A) Incorreta. A busca exata compara a consulta com todos os vetores; com 2 bilhões, a latência e o custo de CPU seriam inviáveis.

✗ C) Incorreta. Aumentar m cria mais conexões por nó, o que aumenta a memória do grafo HNSW.

✗ D) Incorreta. O ef_search é parâmetro de busca; não reduz a memória necessária para manter o grafo.

> **Dica de prova:** Bilhões de vetores, memória limitada e recall menor aceitável = IVF + PQ (faiss, com treino).

D6. B) Criar um conector SFTP do AWS Transfer Family e agendar a recuperação com StartFileTransfer → revise o capítulo 2

Por quê: O AWS Transfer Family oferece servidores gerenciados (para receber arquivos) e conectores (para a AWS atuar como cliente). O conector SFTP envia arquivos do S3 para servidores SFTP remotos ou recupera arquivos deles para o S3, sem infraestrutura própria, e pode ser acionado por agendamento.

✗ A) Incorreta. Notificações de evento do S3 reagem a mudanças no próprio bucket; não monitoram servidores externos.

✗ C) Incorreta. Um servidor SFTP recebe arquivos enviados pelo parceiro; aqui é a cooperativa que precisa buscá-los.

✗ D) Incorreta. O Firehose recebe dados por Direct PUT, Kinesis Data Streams ou MSK; não lê servidores SFTP.

> **Dica de prova:** Buscar arquivos no SFTP de um parceiro = conector SFTP do Transfer Family. Parceiro enviando = servidor SFTP.

D7. A) Nos logs de processamento de DAGs do ambiente no CloudWatch Logs → revise o capítulo 3

Por quê: O Amazon MWAA envia ao CloudWatch Logs, conforme a configuração, os logs de processamento de DAGs, do scheduler, das tarefas, do web server e dos workers. DAGs que não aparecem costumam ter erros de importação (dependência ausente, sintaxe), registrados no log de processamento de DAGs.

✗ B) Incorreta. Se a DAG não foi carregada, ela não tem execuções nem logs de tarefa.

✗ C) Incorreta. O consumo de CPU não indica erro de análise de um arquivo de DAG específico.

✗ D) Incorreta. Saber que o arquivo foi enviado não explica por que o Airflow não conseguiu carregá-lo.

> **Dica de prova:** DAG não aparece no MWAA: veja os logs de DAG processing (erro de importação) no CloudWatch.

D8. C) CROSS JOIN UNNEST(itens) AS t(produto) na cláusula FROM → revise o capítulo 3

Por quê: No Athena (Trino), arrays aninhados são achatados com UNNEST na cláusula FROM, em geral com CROSS JOIN, gerando uma linha por elemento. Para mapas, UNNEST devolve chave e valor. Depois, um GROUP BY por produto conta as ocorrências.

✗ A) Incorreta. CARDINALITY devolve o tamanho do array, não os produtos.

✗ B) Incorreta. ARRAY_AGG faz o contrário: junta valores de várias linhas em um array.

✗ D) Incorreta. LATERAL VIEW explode é sintaxe do Hive, não das consultas DML do Athena; e converter o array em texto é desnecessário.

> **Dica de prova:** Array em várias linhas no Athena: CROSS JOIN UNNEST. Juntar linhas em array: ARRAY_AGG.

D9. D) Pesquisar o evento DeleteTable no Event history do CloudTrail no console → revise o capítulo 3

Por quê: O Event history do CloudTrail registra, sem configuração, os eventos de gerenciamento dos últimos 90 dias em cada região. DeleteTable é uma chamada de API de gerenciamento do Glue, então aparece com a identidade que a fez. Para guardar mais tempo ou registrar data events, é preciso uma trilha ou o CloudTrail Lake.

✗ A) Incorreta. Os logs do Glue no CloudWatch são de execução de jobs e crawlers; chamadas de API do Data Catalog ficam no CloudTrail.

✗ B) Incorreta. Um event data store só passa a guardar eventos a partir da criação (ou de uma importação configurada); não traz automaticamente o evento de ontem.

✗ C) Incorreta. As versões da tabela registram mudanças de esquema, não quem chamou a exclusão; com a tabela excluída, nem há onde consultar.

> **Dica de prova:** Quem fez o quê nos últimos 90 dias, sem trilha: Event history do CloudTrail.

D10. D) Habilitar o gerenciamento da senha master pelo RDS no AWS Secrets Manager → revise o capítulo 4

Por quê: O Amazon RDS pode gerenciar a senha do usuário master no Secrets Manager: a senha é gerada pelo serviço, armazenada em um segredo gerenciado pelo RDS e rotacionada automaticamente. As aplicações e os pipelines leem o segredo em tempo de execução.

✗ A) Incorreta. O Parameter Store guarda o valor criptografado, mas não oferece rotação gerenciada da senha do banco.
✗ B) Incorreta. Funciona, mas exige implementar e manter a função de rotação, que a equipe quer evitar.
✗ C) Incorreta. NoEcho só esconde o valor na exibição do CloudFormation; alguém ainda define a senha e não há rotação.

> **Dica de prova:** Senha master do RDS sem ninguém conhecer e com rotação automática: gerenciamento pelo RDS no Secrets Manager.

D11. B) Criar um projeto para o caso de uso e adicionar os integrantes como membros do projeto → revise o capítulo 4

Por quê: No SageMaker Unified Studio, o domínio agrupa usuários e recursos da organização, e o projeto é o limite onde um grupo colabora em um caso de uso. Quem cria o projeto (ou o dono) adiciona membros; os recursos e ferramentas disponíveis vêm do project profile usado na criação. Só o que for publicado no catálogo fica visível para o restante do domínio.

✗ A) Incorreta. Publicar no catálogo torna os ativos visíveis para descoberta por todo o domínio, o contrário do pedido.
✗ C) Incorreta. Acesso a todos os buckets do domínio expõe dados de outras equipes e ignora o controle por projeto.
✗ D) Incorreta. Domínios separados por pessoa impedem justamente a colaboração e multiplicam a administração.

> **Dica de prova:** Unified Studio: domínio (organização) > domain units (áreas) > projetos (equipes com membros e recursos).

D12. A) GRANT SELECT nas 23 colunas permitidas da tabela funcionarios ao grupo de RH analítico → revise o capítulo 4

Por quê: O controle de acesso por coluna do Redshift permite conceder SELECT e UPDATE em colunas específicas (GRANT SELECT (col1, col2) ON tabela TO ...). É a forma nativa de esconder colunas sem criar views. Para mostrar valores mascarados em vez de ocultar a coluna, usa-se dynamic data masking.

✗ B) Incorreta. Row-level security filtra linhas, não colunas; mascarar valores é função do dynamic data masking.
✗ C) Incorreta. Com SELECT na tabela inteira concedido, o REVOKE de colunas não retira o acesso dado no nível da tabela.
✗ D) Incorreta. Isso não daria acesso às demais colunas da tabela funcionarios, que é a tabela de interesse.

> **Dica de prova:** Esconder colunas de um grupo no Redshift sem view: GRANT SELECT por coluna.

Agora, treine no formato real

Faça as 6 provas completas do curso DEA-C01: 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.