

Guia de preparação DOP-C02

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

6 capítulos por domínio	**36** questões comentadas	**75%** 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 DOP-C02: o guia explica o que cai e onde estão as armadilhas; as provas treinam o ritmo e mostram o que você ainda não domina.

Quando	O que fazer
Semana 1	Faça o teste diagnóstico (página seguinte) e leia os capítulos 1 e 2. No fim da semana, faça a Prova 1 do curso no tempo do exame.
Semana 2	Leia os demais capítulos, começando pelos domínios em que você errou mais no diagnóstico. Faça as provas 2 e 3.
Semana 3	Releia os alertas de pegadinha e as tabelas de comparação. Faça as provas 4 e 5 e revise cada erro pela explicação.
Semana 4	Estude a folha de cola, faça a Prova 6 e refaça as provas abaixo de 80%. Agende o exame quando estiver acima de 80% em todas (a nota de corte é 75%).

Mapa do exame

Cada prova do curso tem 75 questões e 180 minutos, no formato do exame. Referência de aprovação: 75%.

Domínio (capítulo)	Peso	Neste guia
1. Domínio 1: Automação do SDLC	22%	18 conceitos · 14 pegadinhas
2. Domínio 2: Gerenciamento de configuração e IaC	17%	18 conceitos · 14 pegadinhas
3. Domínio 3: Soluções resilientes na nuvem	15%	18 conceitos · 14 pegadinhas
4. Domínio 4: Monitoramento e registro em log	15%	18 conceitos · 14 pegadinhas
5. Domínio 5: Resposta a incidentes e eventos	14%	16 conceitos · 14 pegadinhas
6. Domínio 6: Segurança e conformidade	17%	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: Automação do SDLC

Uma empresa de telecom mantém um pipeline do AWS CodePipeline em us-east-1, criado por um template do CloudFormation com a propriedade ArtifactStore apontando para um bucket S3 criptografado com uma chave KMS do cliente. Para atender clientes europeus, a equipe adicionou ao estágio de produção uma action de deploy do CloudFormation com Region: eu-west-1, e a atualização da stack falhou com um erro de validação da estrutura do pipeline. A equipe quer manter um só pipeline implantando nas duas Regiões. Qual mudança no template resolve o problema?

A. Substituir ArtifactStore por ArtifactStores, com um bucket de artefatos e a chave KMS correspondente para us-east-1 e outro bucket e chave KMS para eu-west-1
B. Trocar a chave do bucket de artefatos por uma chave KMS multi-Região com réplica em eu-west-1, mantendo a propriedade ArtifactStore com o bucket de us-east-1
C. Criar um segundo pipeline em eu-west-1, acionado por uma regra do EventBridge quando o pipeline de us-east-1 concluir o estágio de homologação
D. Ativar a replicação entre Regiões do S3 do bucket de artefatos para um bucket em eu-west-1 e informar esse bucket na configuração da nova action de deploy

D2 · Domínio 1: Automação do SDLC

Uma empresa de telecom adota GitOps para uma aplicação em Amazon EKS. O código fica em um repositório no GitHub conectado à AWS por AWS CodeConnections, e os manifests Helm de cada ambiente ficam em um repositório de configuração separado, observado pelo Argo CD. A cada pull request, os testes automatizados devem rodar; a cada merge na branch main, o fluxo deve gerar uma imagem versionada em um registro privado e atualizar o ambiente pelo Git, de forma que o repositório de configuração continue sendo a fonte da verdade do cluster. Como projetar esse fluxo?

A. Usar um pipeline V2 com origem no GitHub e CodeBuild para testar e gerar a imagem, publicá-la no ECR com tag imutável e fazer commit da nova tag no repositório de configuração
B. Configurar o Argo CD Image Updater para acompanhar a tag latest no Amazon ECR, gravando a nova imagem direto nos parâmetros da aplicação do Argo CD, e manter o CodeBuild para os testes
C. Usar um pipeline V2 com origem no GitHub e CodeBuild para testar e gerar a imagem, enviá-la ao Amazon ECR e aplicar o chart no cluster com helm upgrade pelo CodeBuild
D. Usar a ação de implantação do Amazon EKS no CodePipeline para aplicar os manifests Helm do repositório da aplicação a cada merge, deixando o Argo CD monitorar o estado do cluster

D3 · Domínio 1: Automação do SDLC

Você está implantando um fluxo GitOps para uma aplicação de microsserviços no Amazon EKS. O time quer que toda a configuração de ambientes e as implantações venham de uma única fonte de verdade no Git, com reconciliação automática do estado desejado. Qual abordagem segue MELHOR os princípios de GitOps na AWS?

A. Usar StackSets do CloudFormation para aplicar os manifests em vários clusters EKS, a partir de templates versionados em um bucket S3
B. Guardar os manifests no GitHub e fazer o CodePipeline acionar o CodeBuild a cada commit, aplicando-os no EKS com comandos kubectl apply
C. Manter os manifests em um repositório Git e aplicá-los com Terraform por um projeto do CodeBuild a cada merge na branch principal
D. Usar um repositório Git como fonte de verdade dos manifests e um operador GitOps no cluster, como Flux ou Argo CD, que reconcilia o estado continuamente

D4 · Domínio 2: Gerenciamento de configuração e IaC

Uma seguradora usa o AWS Control Tower e cria contas pelo Account Factory no console. Dependendo do tipo de conta, analytics ou web, cada conta precisa nascer com recursos padrão diferentes, como roles, buckets de log e parâmetros, descritos em templates do CloudFormation que a equipe já mantém. O tipo de pacote é escolhido por quem cria a conta, independentemente da OU, e as contas antigas também devem poder receber o pacote. A equipe não quer construir nem manter pipelines de customização. Qual solução atende a esses requisitos?

A. Adotar o Account Factory for Terraform (AFT) e reescrever os templates do CloudFormation como customizações por conta, aplicadas pelo pipeline do AFT quando a conta é criada
B. Implantar o Customizations for AWS Control Tower (CfCT) com um manifest.yaml que associa cada template a uma OU e aplicar os templates no evento de ciclo de vida CreateManagedAccount
C. Criar stack sets service-managed com auto-deployment em OUs específicas por tipo de pacote e mover cada conta nova para a OU correspondente logo depois da criação no Account Factory
D. Publicar cada template como produto do Service Catalog numa conta hub com a role AWSControlTowerBlueprintAccess e escolher o blueprint do Account Factory Customization na conta

D5 · Domínio 2: Gerenciamento de configuração e IaC

Uma empresa de educação tem 150 contas num AWS Organizations e precisa, toda noite, remover snapshots EBS manuais com mais de 90 dias em cada conta e Região usada, inclusive nas contas criadas depois, gerando um relatório central. O time de plataforma quer uma função Lambda em Python (boto3) rodando na conta de automação, e não na conta de gerenciamento, com menor privilégio e sem manter uma lista de contas à mão. Qual desenho atende a esses requisitos?

A. Criar em cada conta um usuário IAM com access keys guardadas no Secrets Manager da conta de automação e fazer a função percorrer os segredos, autenticando-se em cada conta para apagar os snapshots
B. Rodar a função na conta de automação lendo os IDs das contas de uma variável de ambiente atualizada pelo pipeline e assumindo em cada conta a OrganizationAccountAccessRole para apagar os snapshots
C. Criar na conta de automação uma política do Amazon Data Lifecycle Manager com alvo nas contas da organização, para que ela apague os snapshots manuais com mais de 90 dias
D. Delegar à conta de automação a leitura do Organizations, chamar ListAccounts com paginação e assumir em cada conta uma role mínima criada por StackSets service-managed com auto-deployment

D6 · Domínio 2: Gerenciamento de configuração e IaC

Uma empresa de SaaS mantém um template do AWS SAM com 40 funções Lambda, e cada função repete Runtime, Timeout, Tracing e VpcConfig. Os desenvolvedores implantam em dev, homologação e produção digitando comandos sam deploy longos, com nome de stack, Região, bucket e parameter overrides diferentes por ambiente, e erros de digitação já publicaram parâmetros de dev em produção. A empresa quer padronizar as funções e os comandos por ambiente, versionando tudo no repositório e com o MENOR esforço de manutenção. Qual solução atende a esses requisitos?

A. Manter três templates SAM, um por ambiente, com as propriedades fixas em cada função, e um script de shell que escolhe o template e os parâmetros a partir do nome da branch do repositório
B. Usar a seção Mappings com um mapa por ambiente para Runtime, Timeout e VpcConfig e Fn::FindInMap em cada função, informando o ambiente por uma variável de shell
C. Declarar Runtime, Timeout, Tracing e VpcConfig na seção Globals e criar no samconfig.toml as seções dev, hml e prod com os parâmetros de deploy, usando sam deploy --config-env
D. Transformar as propriedades comuns em parâmetros do template e usar o samconfig.toml com a seção default, executando sam deploy --guided a cada implantação

D7 · Domínio 3: Soluções resilientes na nuvem

Uma operadora de planos de saúde precisa comprovar a cada trimestre que sua aplicação Java em EC2 sobrevive à falha de AZ do banco RDS for PostgreSQL, implantado como instância Multi-AZ. No último teste, o RDS concluiu a troca para a standby em cerca de dois minutos, mas a aplicação continuou com erros de conexão por mais de 15 minutos, até que os servidores foram reiniciados. Os servidores usam o endpoint DNS da instância na string de conexão. Qual forma de executar o teste e qual ajuste atendem ao requisito de recuperação automática?

A. Executar reboot-db-instance com --force-failover e reduzir o tempo de cache de DNS da JVM, fazendo a aplicação descartar e reabrir conexões ao detectar erro
B. Parar a instância primária pelo console do RDS para forçar a promoção da standby e, ao fim do teste, apontar a aplicação para o endpoint próprio da standby promovida
C. Promover a standby a instância independente com promote-read-replica durante o teste e atualizar a string de conexão da aplicação com o novo endpoint
D. Executar reboot-db-instance com --force-failover e trocar na aplicação o endpoint DNS pelo IP privado da primária, para eliminar a resolução DNS durante a troca

D8 · Domínio 3: Soluções resilientes na nuvem

Uma aplicação em EC2 com banco MySQL no Amazon RDS precisa de RTO de 15 minutos e RPO de cerca de 1 minuto em caso de perda da região. O orçamento não comporta uma arquitetura ativo-ativo. Qual estratégia de recuperação de desastres atende aos requisitos com o menor custo?

A. Warm standby: réplica do RDS entre regiões e uma cópia reduzida da aplicação já rodando, escalada no failover pelo Route 53
B. Backup and restore: copiar snapshots do RDS e AMIs para a outra região a cada hora e recriar o ambiente pelo CloudFormation
C. Pilot light: manter só a réplica do RDS entre regiões e AMIs prontas, criando toda a camada de aplicação durante o failover
D. Multi-site ativo-ativo: aplicação em escala total nas duas regiões, com escrita nos dois bancos e roteamento por latência

D9 · Domínio 3: Soluções resilientes na nuvem

Uma empresa global roda uma aplicação em Amazon EC2 atrás de um Application Load Balancer em cada uma de duas Regiões, com registros de failover do Amazon Route 53. Em um teste de resiliência, uma falha parcial fez o ALB primário devolver muitos erros 5xx enquanto as instâncias continuavam passando no health check do target group, e o Route 53 não moveu o tráfego para a Região secundária. O time quer detectar esse tipo de falha e fazer o failover sem intervenção humana. Qual abordagem é MAIS adequada?

A. Colocar o AWS Global Accelerator na frente de um ALB central em uma Região, usando os health checks do acelerador para tirar o tráfego quando o ALB parar de responder
B. Ativar Evaluate Target Health nos registros de alias que apontam para os ALBs, sem criar health checks próprios do Route 53 para a aplicação em cada Região
C. Associar aos registros de failover health checks do Route 53 baseados em um endpoint de saúde profundo da aplicação ou em um alarme do CloudWatch sobre os 5XX do ALB
D. Trocar o failover por registros multivalue answer com os endereços IP dos ALBs, confiando nos health checks do target group para tirar as instâncias com falha

D10 · Domínio 4: Monitoramento e registro em log

Uma empresa de telecom envia os VPC Flow Logs de 30 VPCs para um bucket S3 no formato de texto padrão, compactado, com prefixos por dia, e a equipe de segurança consulta no Athena janelas de 1 a 2 horas durante investigações. As consultas escaneiam o dia inteiro, levam minutos e custam caro. A equipe quer reduzir o volume escaneado e o tempo das consultas sem criar jobs de ETL nem infraestrutura de streaming. Qual solução atende a esses requisitos?

A. Mudar o destino dos flow logs para o CloudWatch Logs e substituir as consultas do Athena por consultas do Logs Insights com filtros de tempo de 1 a 2 horas
B. Criar novos flow logs para o S3 com formato de arquivo Parquet, prefixos compatíveis com Hive e partições por hora, e criar as tabelas do Athena sobre esses prefixos com projeção de partições
C. Enviar os flow logs para um Amazon Data Firehose com conversão de formato para Parquet e particionamento dinâmico por hora, com buffer de 5 minutos e prefixo por conta, gravando no bucket consultado pelo Athena
D. Criar um job do AWS Glue agendado a cada hora que converte os arquivos de texto em Parquet e grava partições por hora, atualizando o Glue Data Catalog no fim de cada execução

D11 · Domínio 4: Monitoramento e registro em log

Uma rede varejista opera 15 clusters do Amazon ECS, com tarefas no EC2 e no Fargate, e usa o Container Insights padrão, que mostra métricas de cluster e de serviço. Em incidentes de falta de memória, a equipe não consegue identificar qual tarefa ou qual contêiner está consumindo mais memória. Ela quer métricas por tarefa e por contêiner, com dashboards prontos, sem instalar agentes ou sidecars e valendo também para os clusters que forem criados depois. Qual solução atende a esses requisitos com o MENOR esforço?

A. Manter o Container Insights padrão e adicionar um sidecar do CloudWatch agent com StatsD em cada task definition para publicar a memória usada por contêiner
B. Implantar o ADOT Collector como serviço daemon nos clusters EC2 e como sidecar no Fargate, exportando as métricas por contêiner para o Managed Service for Prometheus
C. Definir a configuração de conta containerInsights como enhanced para os clusters novos e atualizar os 15 clusters atuais com update-cluster-settings para enhanced
D. Ativar o CloudWatch Application Signals nos serviços dos clusters, que coleta CPU e memória por contêiner e monta os dashboards de infraestrutura das tarefas

D12 · Domínio 4: Monitoramento e registro em log

Uma equipe de confiabilidade investiga incidentes intermitentes em uma aplicação que roda no Amazon ECS com tasks no AWS Fargate. Quando um contêiner é reiniciado, os logs locais são perdidos, o que dificulta a análise de causa raiz. A equipe quer centralizar os logs de aplicação de todas as tasks, consultá-los lado a lado com as métricas e os alarmes que já existem no Amazon CloudWatch durante um incidente e reduzir ao máximo o esforço operacional contínuo. Qual abordagem atende melhor a esses requisitos?

A. Executar uma instância EC2 dedicada com Fluentd recebendo os logs das tasks por syslog remoto e enviando-os a um bucket S3 para análise posterior
B. Configurar o driver de log awslogs nas task definitions para enviar stdout e stderr a log groups do CloudWatch Logs e consultá-los junto das métricas e alarmes
C. Habilitar o Container Insights no cluster e investigar os incidentes pelas métricas de desempenho das tasks, como CPU, memória e reinicializações de contêineres
D. Adicionar às tasks um sidecar FireLens com Fluent Bit que envia os logs a um bucket S3 pelo Amazon Data Firehose e consultá-los com o Amazon Athena nos incidentes

01

CAPÍTULO 1 · 22% DA PROVA

Domínio 1: Automação do SDLC

Este é o domínio de maior peso (22%) e o que mais exige detalhe: CodePipeline, CodeBuild, CodeDeploy, CodeArtifact, ECR e as estratégias de implantação de ECS, Lambda, EC2, Elastic Beanstalk e EKS. Quase sempre o enunciado é um cenário longo com uma restrição decisiva escondida no meio: sem adicionar o CodeDeploy, sem reduzir capacidade, sem dobrar a frota, sem gerenciar runner, com rollback automático por alarme. Onde mais erro é no hook certo do ciclo de implantação (AfterInstall x AfterAllowTestTraffic, PRE_SCALE_UP x POST_TEST_TRAFFIC_SHIFT), na diferença entre canary e linear e nas permissões entre contas (artifact store com chave KMS do cliente). Preciso ler primeiro o requisito de teste, de tráfego e de reversão e só depois olhar as alternativas.

Neste capítulo

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

Tópicos-chave

Blue/green nativo do ECS

Com o deployment controller ECS, o serviço usa as estratégias rolling, blue/green, linear e canary sem precisar do CodeDeploy. O blue/green passa pelos estágios PRE_SCALE_UP, POST_SCALE_UP, TEST_TRAFFIC_SHIFT, POST_TEST_TRAFFIC_SHIFT, PRODUCTION_TRAFFIC_SHIFT, BAKE_TIME e CLEAN_UP, e hooks em Lambda aprovam ou reprovam cada estágio. Testar a versão nova com tráfego de teste (regra de teste no listener) é hook no POST_TEST_TRAFFIC_SHIFT; alarmes e falha de hook acionam o rollback.

Blue/green do ECS com CodeDeploy

Exige deployment controller CODE_DEPLOY, dois target groups, listener de produção e listener de teste. O hook AfterAllowTestTraffic roda os testes pelo listener de teste antes de a produção mudar. No CodePipeline, a action CodeDeployToECS lê imageDetail.json, taskdef.json e appspec.yaml; a action ECS padrão lê imagedefinitions.json.

Hooks do AppSpec em EC2/on-premises

A ordem dos hooks é ApplicationStop, BeforeInstall, AfterInstall, ApplicationStart e ValidateService; no ciclo completo, DownloadBundle vem antes de BeforeInstall e Install antes de AfterInstall. Validar a aplicação já iniciada é ValidateService com script que sai com código diferente de zero na falha. Falha em DownloadBundle aponta para o instance profile (S3 e KMS) e o agente na instância, não para a role de serviço do CodeDeploy.

Hooks de Lambda no CodeDeploy/SAM

Lambda só tem BeforeAllowTraffic (PreTraffic no SAM), que testa a versão nova antes de qualquer tráfego, e AfterAllowTraffic (PostTraffic), que valida depois do deslocamento total. Lambda não tem BeforeInstall. O SAM DeploymentPreference coloca o CodeDeploy no alias e reverte sozinho por alarme.

Canary, Linear e AllAtOnce

Canary envia uma fatia, espera e manda o resto de uma vez (Canary10Percent5Minutes). Linear move incrementos iguais a cada intervalo (Linear10PercentEvery1Minutes = 10% por minuto até 100%). Reverter por métrica de negócio exige alarme associado ao deployment group; no ECS nativo, 5% e depois 100% é canary e vários passos iguais é linear.

Minimum healthy hosts e rollback do CodeDeploy

Nunca abaixo de X% da frota em in-place é configuração personalizada com FLEET_PERCENT = X; HOST_COUNT é número absoluto de instâncias. HEALTH_CONSTRAINTS indica falhas demais para o mínimo saudável: olhe o hook que falhou e o log do agente antes de mexer na configuração. Rollback do CodeDeploy é uma nova implantação da última revisão boa, com outro deployment ID.

file_exists_behavior

O padrão DISALLOW gera o erro 'a specified file already exists at this location' quando há arquivo não implantado pelo CodeDeploy no destino. OVERWRITE faz a revisão vencer; RETAIN mantém o arquivo local.

Modos de execução do CodePipeline

QUEUED processa cada execução em ordem, uma por vez (cada commit precisa passar). PARALLEL roda execuções independentes (por branch). SUPERSEDED deixa a execução mais nova substituir a que espera entre estágios, mesmo com aprovação manual no caminho.

Pipeline tipo V2: triggers, variáveis, condições e rollback

Triggers V2 filtram por branch, caminho de arquivo e tag (monorepo) e aceitam eventos de pull request. Variável no nível do pipeline é informada ao iniciar a execução e lida como #{variables.NOME}; valor gerado no build sai por exported-variables com namespace e é lido como #{Namespace.Variavel}. Condição beforeEntry com regra CloudWatchAlarm e result FAIL bloqueia o estágio enquanto o alarme está em ALARM; rollback de estágio (onFailure ROLLBACK ou RollbackStage) volta à última execução boa sem rebuild.

Pipeline entre contas e Regiões

Entre contas, troque a chave aws/s3 do artifact store por uma chave KMS gerenciada pelo cliente com key policy liberando as contas de destino e use RoleArn na action com role da conta de destino. Action em outra Região exige um bucket de artefatos por Região (artifactStores) com a chave KMS daquela Região; o próprio CodePipeline copia os artefatos.

buildspec do CodeBuild

post_build roda mesmo com falha na fase build: para não publicar, use on-failure: ABORT ou teste CODEBUILD_BUILD_SUCCEEDING = 1. A seção reports publica testes e cobertura no console; secondary-artifacts entrega várias saídas com identificadores iguais aos output artifacts. Segredo vem de env/secrets-manager (ou variável SECRETS_MANAGER) com GetSecretValue na service role, nunca PLAINTEXT.

Ambiente e computação do CodeBuild

'Cannot connect to the Docker daemon' = falta o modo privilegiado. CodeBuild em VPC usa sub-redes privadas com NAT ou endpoints, pois não recebe IP público. Computação Lambda serve a builds curtos sem Docker, VPC ou cache; fleet de capacidade reservada atende carga constante com provisionamento lento; runner hospedado no CodeBuild atende GitHub Actions com acesso à VPC sem gerenciar runner.

AWS CodeArtifact

Repositório de pacotes (npm, PyPI, Maven) com conexão externa como upstream, guardando cópia local e pacotes privados. O token dura no máximo 12 h: E401 depois de horas é token expirado, faça o login no pre_build de cada build. Entre contas: token pela política do domínio, leitura e publicação pela política do repositório.

Amazon ECR no pipeline

Pull through cache resolve limite de pull do Docker Hub, com segredo de prefixo ecr-pullthroughcache/ no Secrets Manager. Tags imutáveis garantem promover exatamente o que foi testado; lifecycle policy limpa imagens untagged por idade e imagens por prefixo de tag, por contagem ou idade. Replicação do registro com registry policy no destino leva imagens a outras contas e Regiões; política do repositório com aws:PrincipalOrgID libera pull para a organização.

Lambda: versões, aliases e imagens

Canary sem CodeDeploy é alias com AdditionalVersionWeights entre duas versões publicadas, nunca $LATEST. Lambda com imagem de contêiner não segue a tag: cada deploy precisa de update-function-code, de preferência com o digest. Layer compartilhado com a organização usa add-layer-version-permission com --principal * e --organization-id, repetido a cada nova versão.

Políticas de implantação do Elastic Beanstalk

Rolling atualiza in-place e reduz capacidade; Rolling with additional batch mantém a capacidade total sem dobrar a frota. Immutable lança um ASG temporário com instâncias novas, não toca nas atuais e tem reversão simples. Traffic splitting faz canary com porcentagem (NewVersionPercent + EvaluationTime).

Rolling update do ECS e do EKS

ECS sem reduzir capacidade = minimumHealthyPercent 100 + maximumPercent 200 (exige capacidade extra); 100/100 trava a implantação. Rolling update preso em IN_PROGRESS se resolve com deploymentCircuitBreaker com rollback true. No EKS, zero downtime é maxUnavailable 0 + maxSurge maior que 0 + readinessProbe que reflete o pod pronto.

GitOps e EC2 Image Builder

GitOps tem Git como fonte da verdade e um operador (Argo CD ou Flux) que puxa e reconcilia; o CI gera a imagem com tag imutável e faz commit no repositório de configuração. EC2 Image Builder gera golden AMI ou imagem de contêiner testada, distribui multi-Região/multi-conta (distribution configuration), reconstrói só com EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE e limpa AMIs e snapshots com lifecycle policy.

Macetes para memorizar

💡 Ordem dos hooks do AppSpec (EC2)

Para, Antes, Depois, Começa, Valida: ApplicationStop, BeforeInstall, AfterInstall, ApplicationStart, ValidateService.

💡 Canary x Linear

Canary = canarinho na mina: uma fatia vai na frente, depois todos. Linear = escada: degraus iguais.

💡 Arquivos do deploy ECS

Padrão = um arquivo (imagedefinitions). Blue/green com CodeDeploy = trio (imageDetail, taskdef, appspec).

💡 Modos de execução

QUEUED = fila do banco (todos passam). PARALLEL = cada branch na sua pista. SUPERSEDED = só o mais novo vale.

💡 Pipeline entre contas

Conta cruzada pede CMK + RoleArn: chave do cliente, role do destino.

💡 post_build

post_build roda até no velório do build: condicione a publicação a CODEBUILD_BUILD_SUCCEEDING.

💡 Onde fica cada teste

SAST e unitário no build; DAST, integração e UI depois do deploy em teste, antes da aprovação.

Tabela de memorização: não confunda

Quando é...	...e quando é
Canary Uma fatia de tráfego (ex.: 10% ou 5%), espera de observação e depois o resto de uma vez.	**Linear** Incrementos iguais a cada intervalo até 100% (ex.: 10% por minuto).
imagedefinitions.json Action de deploy ECS padrão (rolling) no CodePipeline.	**imageDetail.json + taskdef.json + appspec.yaml** Action CodeDeployToECS (blue/green com CodeDeploy); o imagedefinitions.json não aceita taskDefinitionArn nem containerPort.
Rolling with additional batch Manter capacidade total durante o deploy sem dobrar a frota.	**Immutable (Beanstalk)** Não tocar nas instâncias atuais e ter reversão simples, aceitando dobrar a frota temporariamente.
CodeArtifact com conexão externa Pacotes de linguagem (npm, PyPI, Maven) com cópia interna e pacotes privados.	**Pull through cache do ECR** Imagens de contêiner de registros upstream como o Docker Hub, em cache e sincronizadas.
BeforeAllowTraffic (PreTraffic) Testar a versão nova da Lambda antes de qualquer tráfego.	**AfterAllowTraffic (PostTraffic)** Validar a Lambda depois que todo o tráfego foi deslocado.
QUEUED Cada commit precisa passar por todos os estágios, em ordem e um por vez.	**SUPERSEDED** Só a execução mais nova importa; a anterior pode ser substituída entre estágios.

Variável no nível do pipeline (#{variables.NOME}) Valor informado ao iniciar a execução e usado por várias actions.	**exported-variables com namespace (#{Namespace.Variavel})** Valor gerado durante o build e passado para actions seguintes.

Números que caem

Item	Valor
Peso do domínio	**22%**
Validade máxima do token do CodeArtifact	**12 h**
ECS sem reduzir capacidade	**minimumHealthyPercent 100 + maximumPercent 200**
Linear10PercentEvery1Minutes	**10% do tráfego por minuto até 100%**
Deregistration delay padrão do target group	**300 segundos**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Usar o rolling update do ECS com circuit breaker e alarmes quando o requisito pede tráfego de teste isolado antes da produção.
Como não cair: Rolling substitui tarefas no mesmo target group e não tem estágio de teste; tráfego de teste exige blue/green (nativo ou com CodeDeploy).

⚠ **Alerta de prova:** Colocar o hook de testes de API no PRE_SCALE_UP do blue/green nativo do ECS.
Como não cair: No PRE_SCALE_UP a revisão green ainda não subiu; testes com tráfego de teste vão no POST_TEST_TRAFFIC_SHIFT.

⚠ **Alerta de prova:** Trocar o controller para CODE_DEPLOY e usar AfterAllowTestTraffic quando a equipe não quer adicionar o CodeDeploy.
Como não cair: Atende tecnicamente, mas viola a restrição; o controlador ECS já faz blue/green com hooks, tráfego de teste, bake time e rollback.

⚠ **Alerta de prova:** Rodar os testes no hook AfterInstall apontando para o listener de produção.
Como não cair: Em AfterInstall a task set substituta existe, mas a produção ainda vai para a original; teste é AfterAllowTestTraffic pelo listener de teste.

⚠ **Alerta de prova:** Definir maximumPercent 105 para mandar 5% do tráfego à versão nova e esperar 15 minutos.
Como não cair: maximumPercent controla capacidade, não porcentagem de tráfego nem tempo; para isso use canary.

⚠ **Alerta de prova:** Manter o alias da Lambda em $LATEST e criar outro alias com peso de canary.
Como não cair: $LATEST é mutável; o peso precisa ficar entre duas versões publicadas (AdditionalVersionWeights).

⚠ **Alerta de prova:** Manter a chave aws/s3 no artifact store e só liberar a leitura no bucket policy para as outras contas.
Como não cair: A política de uma chave gerenciada pela AWS não pode ser alterada; as contas de destino não decifram. Use chave KMS gerenciada pelo cliente.

⚠ **Alerta de prova:** Usar condição onSuccess com regra CloudWatchAlarm para impedir o deploy em produção durante um alarme.
Como não cair: onSuccess é avaliada depois das actions do estágio, quando o deploy já ocorreu; bloquear a entrada é beforeEntry com result FAIL.

⚠ **Alerta de prova:** Retornar statusCode 200 ou 500 no handler da Lambda invocada pelo CodePipeline.
Como não cair: O pipeline ignora o retorno; a função precisa chamar PutJobSuccessResult ou PutJobFailureResult com o id do job, senão a action fica em In progress.

⚠ **Alerta de prova:** Configurar minimum healthy hosts HOST_COUNT 75 querendo dizer 75% da frota.
Como não cair: HOST_COUNT é número absoluto de instâncias; porcentagem da frota é FLEET_PERCENT.

⚠ **Alerta de prova:** Subir o dockerd à mão na fase install do buildspec para contornar o erro de Docker daemon.
Como não cair: O daemon só sobe com o modo privilegiado do projeto; sem ele o erro continua.

⚠ **Alerta de prova:** Colocar o CodeBuild em sub-rede pública com internet gateway para alcançar a internet.
Como não cair: O CodeBuild não atribui IP público às interfaces que cria na VPC; use sub-redes privadas com NAT ou endpoints.

⚠ **Alerta de prova:** Usar file_exists_behavior: RETAIN quando a revisão deve substituir os arquivos existentes.
Como não cair: RETAIN mantém a cópia local; a revisão vencer é OVERWRITE.

⚠ **Alerta de prova:** Usar a action InspectorScan em SourceCodeScan como SAST do código próprio.
Como não cair: Ela gera SBOM e avalia componentes de código aberto; SAST + SCA + IaC em repositórios GitHub/GitLab é Inspector code security (CodeGuru Security foi descontinuado).

Como cai na prova

📝 **Como cai:** Cenário de implantação em ECS/Lambda/EC2 com requisito de testar com tráfego de teste, reverter por alarme e observar depois da troca.
Como resolver: 1) Veja se o CodeDeploy é permitido. 2) Localize o momento do teste (antes de qualquer tráfego, com tráfego de teste, depois da troca). 3) Escolha o hook daquele momento. 4) Confirme alarme + bake time para o rollback.

📝 **Como cai:** Pipeline entre contas ou Regiões falhando com AccessDenied ou kms:Decrypt.
Como resolver: Procure chave aws/s3 no artifact store: troque por CMK compartilhada, adicione RoleArn com role na conta de destino e, para outra Região, artifactStores com bucket e chave da Região.

📝 **Como cai:** Requisito numérico de capacidade ou tráfego (nunca abaixo de X%, 10% por minuto, 5% e depois o resto).
Como resolver: Traduza para o parâmetro: FLEET_PERCENT, Linear, Canary, minimumHealthyPercent/maximumPercent. Desconfie de parâmetros de capacidade usados como se fossem tráfego.

📝 **Como cai:** Build falhando com mensagem específica (Docker daemon, E401, toomanyrequests, DownloadBundle).
Como resolver: Associe a mensagem à causa: modo privilegiado, token do CodeArtifact expirado, pull through cache, instance profile e agente.

📝 **Como cai:** Escolher entre várias soluções que funcionam, com 'menor esforço operacional'.
Como resolver: Descarte as que exigem código próprio (Lambda de polling, scripts) ou servidores próprios; prefira o recurso nativo gerenciado que atende a todas as restrições.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
toomanyrequests: You have reached your pull rate limit no build	**Pull through cache do ECR com segredo ecr-pullthroughcache/ no Secrets Manager.**
Ligar funcionalidade aos poucos sem novo deploy, com rollback por alarme	**Feature flag no AWS AppConfig + estratégia de implantação com bake time.**

Teste de UI roteirizado com navegador, capturas e HAR, reaproveitado como monitor da produção	**Canary do CloudWatch Synthetics.**
Segredo no template do CloudFormation versionado no Git	**Dynamic reference ao Secrets Manager; NoEcho só esconde a exibição.**
Mesmo template em vários ambientes no CodePipeline	**Um arquivo de configuração por ambiente no TemplateConfiguration.**
Validar CloudFormation antes do merge sem criar recursos	**cfn-lint + cfn-guard como verificação obrigatória do pull request.**
CDK Pipelines implantando em outra conta	**cdk bootstrap na conta de destino com --trust (conta do pipeline) e --cloudformation-execution-policies.**
Reconstruir a aplicação quando a imagem base muda	**Action de origem do ECR (repositório e tag) como segunda origem do pipeline.**
Pipeline falha ao usar a conexão do GitHub	**Falta codeconnections:UseConnection na role de serviço.**
Testar mudança de configuração do CloudFront com parte do tráfego real	**Staging distribution + continuous deployment policy.**
BlockTraffic demorado no CodeDeploy com requisições curtas	**Deregistration delay do target group alto demais; ajuste ao tempo real das requisições.**
Trocar AMI no mesmo ASG em lotes, com pausa e rollback	**Instance refresh com checkpoints, skip matching e alarmes.**
Bloquear deploy por CVE descoberto depois do build	**Inspector com varredura contínua do ECR + condição de estágio no pipeline.**
Execuções duplicadas do pipeline após criar regra de evento	**PollForSourceChanges omitido vale true; desligue o polling.**

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

- Testar com tráfego de teste: POST_TEST_TRAFFIC_SHIFT (ECS nativo) ou AfterAllowTestTraffic (CodeDeploy); nunca AfterInstall na produção.
- Artifact store entre contas exige chave KMS gerenciada pelo cliente; aws/s3 não é compartilhável.
- post_build roda mesmo se o build falhar; on-failure: ABORT ou CODEBUILD_BUILD_SUCCEEDING.
- Rollback do CodeDeploy é uma nova implantação com outro deployment ID.
- Promova os artefatos da execução (tag imutável), não busque o branch de novo em cada estágio.

Essenciais para a prova (checklist)

- ☐ Sei citar os estágios do blue/green nativo do ECS e em qual deles roda o teste com tráfego de teste.

- ☐ Sei a ordem dos hooks do AppSpec para EC2 e os dois hooks de Lambda.
- ☐ Sei diferenciar Canary, Linear e AllAtOnce e onde ligar o rollback por alarme.
- ☐ Sei montar um pipeline entre contas e entre Regiões (CMK, RoleArn, artifactStores).
- ☐ Sei usar triggers V2, variáveis de pipeline, namespaces, condições beforeEntry e rollback de estágio.
- ☐ Sei escolher QUEUED, PARALLEL ou SUPERSEDED pelo requisito.
- ☐ Sei resolver erros de CodeBuild: modo privilegiado, VPC, post_build, reports e segredos.
- ☐ Sei quando usar CodeArtifact, pull through cache do ECR e replicação do registro.
- ☐ Sei escolher a política de implantação do Beanstalk certa para capacidade e reversão.
- ☐ Sei configurar canary de Lambda sem CodeDeploy e com SAM DeploymentPreference.

Questões de revisão do capítulo

R1.1 · Domínio 1: Automação do SDLC

Uma rede de farmácias tem um pipeline no AWS CodePipeline que implanta o site de e-commerce em homologação e, depois de uma aprovação, em produção. Os testes de UI (login, busca, carrinho e checkout) são feitos por uma pessoa clicando no site, o que atrasa as entregas. A equipe quer testes de UI automatizados em navegador headless, com capturas de tela e HAR de cada passo guardados no S3, executados logo após o deploy em homologação e reprovando o pipeline se algum passo falhar; o mesmo script deve depois monitorar a produção a cada 5 minutos. Qual solução atende a esses requisitos?

A. Criar um teste de carga com a solução Distributed Load Testing on AWS acionado pelo pipeline, reprovando o estágio quando a latência p95 passar do limite combinado
B. Ativar o CloudWatch RUM em homologação e criar uma condição no estágio de produção sobre o alarme de erros de JavaScript do app monitor, bloqueando a entrada no estágio de produção
C. Adicionar uma action de teste do CodeBuild que roda os testes unitários do front-end com Jest e cobertura de código, publicando os resultados num report group
D. Criar um canary do CloudWatch Synthetics com script de navegador para o fluxo e uma action no pipeline que inicia o canary após o deploy e reprova o estágio se a execução falhar

R1.2 · Domínio 1: Automação do SDLC

A equipe de segurança de um grupo de varejo quer fornecer aos times de produto uma imagem base de contêiner endurecida, com pacotes atualizados, agentes obrigatórios e testes de conformidade, para ser usada no FROM dos Dockerfiles. A imagem deve ser reconstruída toda semana e quando os componentes mudarem, testada antes de publicar e disponibilizada no Amazon ECR de três Regiões. A equipe já mantém componentes do EC2 Image Builder para endurecer AMIs e não quer manter scripts de build próprios. Qual solução atende a esses requisitos com o MENOR esforço operacional?

A. Criar uma receita de contêiner do EC2 Image Builder com a imagem pai, componentes de build e de teste e repositório de destino no ECR, num pipeline agendado com distribuição para as Regiões
B. Criar um projeto do AWS CodeBuild agendado pelo EventBridge que executa docker build com um Dockerfile de endurecimento e envia a imagem a um repositório do ECR com replicação entre Regiões configurada no registro
C. Configurar uma regra de pull through cache do ECR para a imagem pública do sistema operacional e aplicar os componentes de endurecimento nos Dockerfiles de cada time
D. Criar uma receita de imagem (AMI) do EC2 Image Builder com os componentes atuais e exportar a AMI resultante como imagem de contêiner para o ECR em cada Região

R1.3 · Domínio 1: Automação do SDLC

Os builds no AWS CodeBuild de vários times baixam pacotes npm públicos direto da internet e às vezes falham quando o registro público fica lento. A empresa também quer publicar pacotes internos e controlar quais versões os builds usam. Qual solução atende com menor esforço?

A. Criar uma imagem de build no Amazon ECR com os pacotes já instalados e reconstruí-la toda semana por um pipeline próprio, usando-a como imagem de cada projeto do CodeBuild
B. Criar no CodeArtifact um repositório com conexão externa ao npmjs como upstream do repositório dos builds, publicar nele os pacotes internos e usar aws codeartifact login
C. Ativar o cache do CodeBuild no Amazon S3 para a pasta node_modules e publicar os pacotes internos como artefatos do build em outro bucket compartilhado entre os times
D. Guardar os pacotes .tgz das dependências em um bucket versionado do Amazon S3 e rodar o npm install apontando para esses arquivos na fase install de cada projeto

R1.4 · Domínio 1: Automação do SDLC

Uma plataforma de streaming roda serviços em contêineres no Amazon ECS e funções AWS Lambda que já leem configuração pelo agente e pela extensão do AWS AppConfig. O time de produto quer ligar um novo recomendador para uma pequena parte da frota, ampliar a exposição aos poucos, voltar sozinho ao estado anterior se o alarme de erros do CloudWatch disparar e desligar a funcionalidade em segundos se necessário, tudo sem nova implantação de código. Qual solução atende a esses requisitos?

A. Criar uma feature flag no AWS AppConfig e implantá-la com uma estratégia linear com bake time, associando o alarme de erros ao ambiente
B. Publicar uma versão da função com a funcionalidade ligada e dar a ela 5% de peso no alias, ampliando o peso aos poucos com o AWS CodeDeploy
C. Usar a implantação contínua do Amazon CloudFront, com uma distribuição de staging recebendo 5% do tráfego com a funcionalidade ligada
D. Guardar a flag em um parâmetro do Parameter Store lido pela aplicação na inicialização e alterar o valor dele para ligar ou desligar a funcionalidade

Respostas comentadas

R1.1. D) Criar um canary do CloudWatch Synthetics com script de navegador para o fluxo e uma action no pipeline que inicia o canary após o deploy e reprova o estágio se a execução falhar
Por quê: O CloudWatch Synthetics executa canaries, scripts em Node.js com Puppeteer ou em Python com Selenium num navegador headless, que guardam capturas de tela, arquivos HAR e logs num bucket S3 e publicam métricas como SuccessPercent. Um canary pode rodar uma vez ou num agendamento; no pipeline, uma action que chama StartCanary e lê GetCanaryRuns transforma o canary num teste de UI pós-deploy.
✗ A) Incorreta. Teste de carga mede desempenho sob volume; não verifica se cada passo do fluxo de UI funciona, e não serviria para o monitoramento contínuo da produção.
✗ B) Incorreta. O RUM depende de tráfego de usuários reais, que não existe em homologação logo após o deploy; ele observa a experiência, não executa um roteiro de teste.
✗ C) Incorreta. Testes unitários com Jest validam componentes isolados durante o build; não exercitam o fluxo do site implantado no navegador nem geram capturas de cada passo.

> **Dica de prova:** Teste de UI roteirizado com navegador, capturas e HAR, reaproveitável como monitor da produção: canary do CloudWatch Synthetics.

R1.2. A) Criar uma receita de contêiner do EC2 Image Builder com a imagem pai, componentes de build e de teste e repositório de destino no ECR, num pipeline agendado com distribuição para as Regiões

Por quê: O EC2 Image Builder cria e mantém AMIs (receitas de imagem) e imagens de contêiner (receitas de contêiner). A receita de contêiner combina imagem pai (Docker Hub, ECR ou imagens gerenciadas), componentes de build e teste e repositório de destino no ECR; o pipeline executa por agendamento ou quando dependências mudam, e a distribuição envia a imagem para outras Regiões.

✗ B) Incorreta. Funciona, mas a equipe teria de manter o Dockerfile e os scripts de endurecimento e teste em paralelo aos componentes do Image Builder que já existem, o que aumenta o esforço.

✗ C) Incorreta. O pull through cache só mantém cópias de imagens públicas no ECR, sem endurecimento nem testes, e espalhar o endurecimento pelos Dockerfiles dos times foge do controle centralizado.

✗ D) Incorreta. Uma receita de imagem do Image Builder gera AMIs; imagens Docker para o ECR exigem uma receita de contêiner, e não existe exportação de AMI como imagem de contêiner.

> **Dica de prova:** Imagem de contêiner endurecida, testada e publicada no ECR, reaproveitando componentes = receita de contêiner do EC2 Image Builder.

R1.3. B) Criar no CodeArtifact um repositório com conexão externa ao npmjs como upstream do repositório dos builds, publicar nele os pacotes internos e usar aws codeartifact login

Por quê: O AWS CodeArtifact é um repositório de artefatos gerenciado: um repositório com conexão externa ao npmjs, colocado como upstream do repositório usado pelos builds, busca e guarda os pacotes públicos na primeira requisição, enquanto o repositório dos builds hospeda os pacotes internos. O comando aws codeartifact login gera o token e configura o npm para usar o endpoint.

✗ A) Incorreta. Uma imagem com pacotes fixos fica desatualizada entre as reconstruções e não resolve a publicação de pacotes internos.

✗ C) Incorreta. O cache acelera builds que encontram o cache válido, mas não cria um registro para pacotes internos nem controla as versões consumidas.

✗ D) Incorreta. Manter .tgz no S3 não resolve as dependências transitivas nem oferece um registro npm para publicar e versionar os pacotes internos.

> **Dica de prova:** Dependências de build (npm, PyPI, Maven) com cópia local e pacotes privados na AWS: CodeArtifact com conexão externa como upstream.

R1.4. A) Criar uma feature flag no AWS AppConfig e implantá-la com uma estratégia linear com bake time, associando o alarme de erros ao ambiente

Por quê: O AWS AppConfig separa a liberação de funcionalidades da implantação de código: feature flags ficam em um perfil de configuração, e cada mudança é implantada com uma estratégia (linear ou exponencial, com bake time). Alarmes do CloudWatch associados ao ambiente fazem o AppConfig reverter a configuração automaticamente durante a implantação ou o bake time, e as aplicações recebem o novo valor pelo agente ou pela extensão, sem reiniciar.

✗ B) Incorreta. O deslocamento por pesos no alias exige publicar código novo a cada mudança, e não alcança os serviços do ECS; a funcionalidade não pode ser desligada sem outra implantação.

✗ C) Incorreta. A implantação contínua do CloudFront testa mudanças de configuração da distribuição com parte do tráfego, não liga funcionalidades dentro do código, e não cobre as chamadas internas das funções e dos serviços.

✗ D) Incorreta. Lido só na inicialização, o parâmetro exige reiniciar os serviços para valer, e não há liberação gradual nem volta automática por alarme.

> **Dica de prova:** Ligar funcionalidade aos poucos sem novo deploy e com rollback por alarme: feature flag no AppConfig + estratégia de implantação com bake time.

02

CAPÍTULO 2 · 17% DA PROVA

Domínio 2: Gerenciamento de configuração e IaC

Aqui preciso dominar CloudFormation a fundo (StackSets, políticas de atributo, change sets, exports, custom resources, drift), CDK e SAM, o Systems Manager (State Manager, Patch Manager, Automation, Run Command, Inventory) e a governança multi-conta com Organizations, Control Tower e Service Catalog. Vale 17% e cai como cenário de escala: dezenas de contas, OUs, contas novas que precisam nascer prontas. Erro comum é escolher a ferramenta de detecção quando o enunciado pede prevenção (ou o contrário) e confundir DeletionPolicy com UpdateReplacePolicy. Também preciso saber qual role falta em cada conta quando o StackSets ou o Automation falha.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 6 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

StackSets service-managed x self-managed

Service-managed com auto-deployment implanta sozinho em contas novas da OU, sem roles manuais. Self-managed exige AWSCloudFormationStackSetExecutionRole (e confiança) em cada conta de destino e não segue contas novas. Fora da conta de gerenciamento, use administrador delegado com --call-as DELEGATED_ADMIN.

Tolerância a falhas e concorrência do StackSets

A tolerância a falhas é por Região; em SEQUENTIAL, estourar a tolerância interrompe as Regiões seguintes. STRICT limita a simultaneidade a tolerância + 1. Para excluir algumas contas de uma OU sem movê-las, use AccountFilterType DIFFERENCE.

DeletionPolicy e UpdateReplacePolicy

DeletionPolicy protege o recurso quando ele sai da stack ou a stack é excluída; UpdateReplacePolicy protege quando uma atualização substitui o recurso. Importar recurso existente exige change set IMPORT com DeletionPolicy em cada recurso.

Stack policy, termination protection e rollback triggers

Stack policy nega ações como Update:Replace em recursos críticos, mas não reverte nada e não bloqueia mudança feita pelo console fora do CloudFormation. Termination protection na raiz + negar UpdateTerminationProtection e DeleteStack por SCP/IAM impede excluir stack crítica. Reverter por métrica depois do deploy é rollback triggers + monitoring time.

Exports e referências entre stacks

Um export não pode mudar nem sair enquanto estiver importado ('Export cannot be updated as it is in use'): crie um export novo, migre os importadores e remova o antigo. Para não travar a stack produtora, publique o valor no Parameter Store e consuma com resolve:ssm.

Custom resource e cfn-signal

Custom resource travado por cerca de uma hora significa que a função não respondeu no ResponseURL (cfnresponse), muitas vezes no Delete ou numa exceção. Esperar o bootstrap da EC2 antes do CREATE_COMPLETE é CreationPolicy + cfn-signal; sem sinal no Timeout, a criação falha e faz rollback.

UpdatePolicy de Auto Scaling group

AutoScalingReplacingUpdate cria um grupo novo (imutável, rollback rápido); AutoScalingRollingUpdate troca em lotes mantendo o grupo. UpdatePolicy vale para atualizações de Auto Scaling group, não para instância EC2 avulsa.

Novidades do CloudFormation cobradas

Fn::ForEach, Fn::Length e Fn::ToJsonString exigem Transform AWS::LanguageExtensions (a transformação do SAM não habilita). Stack refactoring move recursos entre stacks sem recriar; IaC generator escaneia recursos criados pelo console e gera template para importação; Git sync atualiza a stack a cada commit sem pipeline.

CDK e SAM

CDK Aspects aplicam regra a todos os constructs, inclusive de bibliotecas; cdk diff mostra o que vai mudar; cdk.context.json deve ser versionado (atualize com cdk context --reset). No SAM, Globals evita repetição e samconfig.toml com --config-env separa parâmetros por ambiente; sam sync é para desenvolvimento, produção usa sam deploy ou pipeline.

Controles preventivos, detectivos e proativos

Preventivo bloqueia a ação (SCP/RCP); detectivo acha o que já existe (AWS Config); proativo barra o template antes de criar (Hooks do CloudFormation, cfn-guard no pipeline). SCP limita principais da organização; RCP limita recursos da organização, inclusive contra principais externos.

Control Tower: AFC, CfCT e AFT

Conta nascendo com pacote escolhido na criação, sem pipeline = Account Factory Customization com blueprint. Templates CloudFormation e SCPs por OU via manifesto e pipeline = CfCT. Requisito de Terraform = AFT. Muitas contas existentes de uma OU entram com Register OU; SCP gerenciada alterada fora do Control Tower se corrige com Re-register da OU.

Systems Manager: quem faz o quê

Patches = Patch Manager (patch policy do Quick Setup na organização). Estado desejado que se corrige sozinho = associação do State Manager. Mudança com aprovação e trilha passo a passo = runbook do Automation com aws:approve e executeScript. Janela com início, duração, cutoff e ordem de tarefas = maintenance window.

Patch baseline e patch policy

Aprovar patches N dias depois do lançamento, bloquear uma KB e definir severidade de conformidade exige baseline personalizada. Com patch policy em uso, vale o baseline da política e patch group não conta; fora dela, vale o patch group e, sem grupo, o baseline padrão.

Nós gerenciados e agentes

Servidor fora da AWS entra por hybrid activation + SSM Agent (nós mi-). EC2 gerenciado sem instance profile = Default Host Management Configuration (exige IMDSv2). Agente de terceiros em frota mista com versões = Distributor + State Manager com AWS-ConfigureAWSPackage; recipes Chef sem servidor = AWS-ApplyChefRecipes.

Automation e Run Command multi-conta

Automation multi-conta usa AdministrationRole na conta central, ExecutionRole em cada conta alvo, administrador delegado para OUs e runbook compartilhado (pelo AWS RAM na mesma Região). Rate control do Run Command: max errors vale para novos envios, não para invocações já em andamento. Saída completa vai para S3 e/ou CloudWatch Logs.

Service Catalog

Times implantam só templates aprovados, sem editar código e sem permissão ampla: portfólio compartilhado com template constraint e launch constraint. TagOptions padronizam valores de tags; template corrigido vira nova versão do mesmo produto + Update provisioned product.

Políticas do Organizations

Tag policies padronizam chave e valores (enforced_for para bloquear). Backup policy impõe plano que a conta não muda. Declarative policy fixa configuração de serviço para a organização inteira. Conta criada pelo Organizations tem OrganizationAccountAccessRole; conta convidada não tem.

Acesso root centralizado

Root sem login nas contas-membro = gerenciamento centralizado do acesso root. Bucket policy que trancou todos, sem senha root, se resolve com privileged root action (sts:AssumeRoot com S3UnlockBucketPolicy).

Macetes para memorizar

💡 Políticas de atributo

Deletion = morte da stack; UpdateReplace = troca na atualização.

💡 Tipos de controle

Preventivo trava a porta (SCP), proativo revista na entrada (Hooks/Guard), detectivo procura dentro de casa (Config).

💡 Control Tower

CfCT = CloudFormation; AFT = Terraform (o T de Terraform); AFC = blueprint na hora da criação.

💡 Systems Manager

Patch = Patch Manager; Estado = State Manager; Aprovação = Automation; Agora = Run Command; Horário = maintenance window.

💡 Export em uso

Export importado é concreto seco: não muda. Crie outro, migre, apague.

💡 Contas do Organizations

Nasceu na organização = tem OrganizationAccountAccessRole; foi convidada = crie a role.

Tabela de memorização: não confunda

Quando é...	...e quando é
DeletionPolicy Proteger o recurso quando a stack é excluída ou ele sai do template.	**UpdateReplacePolicy** Proteger o recurso quando uma atualização exige substituição.
StackSets service-managed Contas da organização por OU, com auto-deployment em contas novas.	**StackSets self-managed** Contas fora do modelo de OU, com roles de administração e execução criadas à mão.
CfCT Control Tower com templates CloudFormation e SCPs por OU.	**AFT** Control Tower com Terraform para provisionar e customizar contas.

Associação do State Manager Configuração que precisa ser reaplicada e corrigida sozinha ao longo do tempo.	**Run Command** Ação pontual agora, como patch de emergência com rate control.
Stack policy Impedir atualização ou substituição de recursos protegidos.	**Rollback triggers** Reverter a atualização quando um alarme dispara no monitoring time.
AutoScalingReplacingUpdate Imutável: grupo novo e rollback rápido.	**AutoScalingRollingUpdate** Lotes no mesmo grupo.

Números que caem

Item	Valor
Peso do domínio	**17%**
Simultaneidade no modo STRICT do StackSets	**Tolerância a falhas + 1**
Re-register da OU x atualização da landing zone	**Landing zone inteira só se a OU passar de 1.000 contas**
Custom resource sem resposta no ResponseURL	**Stack travada por cerca de uma hora**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Usar stack policy negando Update:Replace achando que ela reverte a atualização se algo der errado.
Como não cair: Stack policy só impede atualizar recursos protegidos; reverter por métrica é rollback triggers com monitoring time.

⚠ **Alerta de prova:** Ativar uma 'substituição forçada de exportações' para mudar um export em uso.
Como não cair: Essa opção não existe; crie um export novo, migre os importadores e remova o antigo, ou use Parameter Store com resolve:ssm.

⚠ **Alerta de prova:** Usar stack sets self-managed e um runbook para adicionar cada conta nova.
Como não cair: Self-managed não segue contas novas; service-managed com auto-deployment faz isso sem código.

⚠ **Alerta de prova:** Manter a AWS-DefaultPatchBaseline e passar a KB como exclusão no AWS-RunPatchBaseline.
Como não cair: A baseline predefinida não tem a regra de N dias nem rejeição de KB; crie baseline personalizada.

⚠ **Alerta de prova:** Criar associação do State Manager na conta de gerenciamento esperando alcançar as contas-membro.
Como não cair: Associações valem só para a conta e Região onde são criadas; na organização use patch policy do Quick Setup ou Automation multi-conta.

⚠ **Alerta de prova:** Usar UpdatePolicy AutoScalingRollingUpdate numa instância EC2 avulsa para esperar o bootstrap.
Como não cair: UpdatePolicy é para Auto Scaling group; instância avulsa usa CreationPolicy + cfn-signal.

⚠ **Alerta de prova:** Instalar o SSM Agent em servidores locais com instance profile AmazonSSMManagedInstanceCore.
Como não cair: Instance profile e metadados só existem em EC2; servidor local usa hybrid activation.

⚠ **Alerta de prova:** Mover recursos compartilhados para a conta de gerenciamento para ficarem protegidos pelas SCPs da raiz.
Como não cair: SCP não afeta a conta de gerenciamento; não rode cargas nela.

⚠ **Alerta de prova:** Fazer reset da landing zone para corrigir uma SCP gerenciada alterada.
Como não cair: Reset é para drift grave; SCP alterada se corrige com Re-register da OU.

⚠ **Alerta de prova:** Esperar que a drift detection reaplique o template do repositório.
Como não cair: Drift detection só detecta; detecção contínua é a regra cloudformation-stack-drift-detection-check do Config.

⚠ **Alerta de prova:** Declarar só Transform AWS::Serverless-2016-10-31 para usar Fn::ForEach.
Como não cair: O loop exige também AWS::LanguageExtensions.

⚠ **Alerta de prova:** Usar o padrão .sync no Step Functions para dispensar permissões IAM na role.
Como não cair: .sync só espera o término do job; a role sempre precisa das ações chamadas.

⚠ **Alerta de prova:** Editar as roles AWSReservedSSO direto pelo IAM, conta por conta.
Como não cair: Essas roles são gerenciadas pelo Identity Center; mude o permission set.

⚠ **Alerta de prova:** Fazer Retain + remover + resource import para mover recursos entre stacks quando o requisito é menor esforço.
Como não cair: Funciona, mas é o caminho antigo; hoje use stack refactoring.

Como cai na prova

📝 **Como cai:** Dezenas de contas e OUs; contas novas precisam receber baseline sem ação manual.
Como resolver: Procure StackSets service-managed com auto-deployment, ou CfCT/AFT se houver Control Tower; descarte self-managed, scripts e Lambdas agendadas.

📝 **Como cai:** Requisitos misturando bloquear, detectar o existente e barrar antes de criar.
Como resolver: Mapeie cada requisito: bloquear = SCP/RCP; existente = Config; antes de criar = Hooks/cfn-guard. Alternativas que trocam a ordem estão erradas.

📝 **Como cai:** Stack travada ou recurso crítico em risco de recriação.
Como resolver: Identifique o estado: export em uso, custom resource sem resposta, falta de sinal. Para proteção, combine stack policy negando Update:Replace + UpdateReplacePolicy + DeletionPolicy.

📝 **Como cai:** Patch e configuração em frota grande (EC2 e on-premises).
Como resolver: Patch na organização = patch policy do Quick Setup; regras de aprovação = baseline personalizada; estado contínuo = State Manager; on-premises = hybrid activation.

📝 **Como cai:** Templates duplicados por produto ou ambiente.
Como resolver: Prefira módulos ou nested stacks versionados + parâmetros por ambiente; template monolítico com condições é distrator.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
'should have AWSCloudFormationStackSetExecutionRole'	**Falta a role de execução (ou a confiança) na conta de destino.**
Configurar a conta nova do Organizations assim que estiver pronta	**Evento CreateAccountResult com SUCCEEDED, não a chamada CreateAccount.**
Tarefa em todas as contas da organização	**ListAccounts + AssumeRole numa role mínima distribuída por StackSets com auto-deployment.**
Unrecognized resource types com tipo de terceiro	**Ativar a extensão pública em cada conta e Região.**
Prévia do impacto (inclusive substituição) antes de aplicar	**Change set + aprovação manual (CHANGE_SET_REPLACE, aprovação, CHANGE_SET_EXECUTE).**
Espera humana longa no meio do fluxo	**Step Functions Standard com .waitForTaskToken; Retry com BackoffRate e Catch.**

Lambda que só chama uma API dentro do Step Functions	**Integração aws-sdk direta.**
Configuração por ambiente e serviço no Parameter Store	**Nomes hierárquicos + GetParametersByPath + IAM por caminho.**
Ajustes frequentes de configuração com validação e rollback rápido	**AWS AppConfig; a infraestrutura de base continua em IaC.**
Inventário de software consultável por SQL	**Inventory com resource data sync para o S3 e Athena.**
Impedir licença BYOL além do contrato	**Self-managed license do License Manager com hard limit associada à AMI.**
Permission set falhou só em algumas contas	**A política gerenciada pelo cliente referenciada não existe nelas; crie com StackSets.**
Regra do Config sob medida sem Lambda	**Custom policy em Guard.**
Config caro por recursos efêmeros	**Gravar só os tipos necessários e frequência diária onde tempo real não é exigido.**

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

- Stack policy não faz rollback e não bloqueia mudança pelo console.
- Export importado não muda; Parameter Store com resolve:ssm evita o acoplamento.
- Associação do State Manager vale só na conta e Região onde foi criada.
- Service-managed + auto-deployment é o caminho para contas novas sem esforço.

Essenciais para a prova (checklist)

- ☐ Sei diferenciar StackSets service-managed e self-managed e as roles de cada um.
- ☐ Sei usar DeletionPolicy, UpdateReplacePolicy, stack policy e termination protection juntas.
- ☐ Sei resolver export em uso, custom resource travado e falta de cfn-signal.
- ☐ Sei quando usar AFC, CfCT, AFT, Register OU e Re-register.
- ☐ Sei classificar controles em preventivo, detectivo e proativo.
- ☐ Sei escolher entre Patch Manager, State Manager, Automation, Run Command e maintenance window.
- ☐ Sei configurar baseline personalizada e a regra de precedência da patch policy.
- ☐ Sei usar Service Catalog com constraints e TagOptions.
- ☐ Sei quando usar CDK Aspects, cdk diff, Globals e samconfig.toml.

Questões de revisão do capítulo

R2.1 · Domínio 2: Gerenciamento de configuração e IaC

Um banco usa o Session Manager em 60 contas da organização e em duas Regiões para o acesso administrativo às instâncias, sem SSH. A auditoria passou a exigir que cada sessão seja gravada, criptografada com uma chave KMS, num bucket S3 central e num log group do CloudWatch Logs com streaming, e que nenhuma conta altere essa configuração localmente. Hoje as preferências do Session Manager estão no padrão em cada conta, sem gravação. Qual abordagem padroniza a configuração com o MENOR esforço operacional, incluindo as contas criadas no futuro?

A. Criar um organization trail do CloudTrail com data events do Session Manager, que grava a entrada e a saída de cada sessão no bucket central e no CloudWatch Logs
B. Implantar por StackSets service-managed com auto-deployment o documento SSM-SessionManagerRunShell com s3BucketName, cloudWatchLogGroupName, criptografia e streaming, negando por SCP a alteração dele
C. Criar em cada conta uma associação do State Manager que instala o CloudWatch agent e envia /var/log/secure e o histórico do shell das instâncias para o log group central da auditoria
D. Configurar a Default Host Management Configuration em cada conta e Região, para que as instâncias passem a gravar as sessões no bucket central e no log group com a role padrão do serviço e a chave KMS gerenciada da conta

R2.2 · Domínio 2: Gerenciamento de configuração e IaC

A segurança de uma empresa de energia com 90 contas no AWS Organizations quer impedir que qualquer conta, inclusive as futuras, torne públicas AMIs ou snapshots do EBS. O controle deve valer independentemente das permissões IAM de quem faz a chamada, inclusive para service-linked roles, e continuar valendo se o EC2 lançar novas APIs de compartilhamento. A equipe também quer um relatório do estado atual desses atributos nas contas. Qual solução atende a esses requisitos?

A. Implantar em cada conta a regra do AWS Config ebs-snapshot-public-restorable-check com remediação automática que remove o compartilhamento público do snapshot
B. Anexar à raiz da organização uma declarative policy de EC2 com image block public access e snapshot block public access habilitados
C. Habilitar o block public access do S3 no nível da organização, que também cobre AMIs e snapshots do EBS por serem armazenados no Amazon S3
D. Anexar uma SCP que nega ec2:ModifyImageAttribute e ec2:ModifySnapshotAttribute quando a launch permission ou a permissão de criação de volume incluir o grupo all

R2.3 · Domínio 2: Gerenciamento de configuração e IaC

Uma empresa de streaming de mídia ativou o AWS Config com gravação contínua de todos os tipos de recurso em 30 contas. A conta de processamento de vídeo cria e destrói milhares de instâncias Spot, volumes EBS e interfaces de rede por hora, e o custo de itens de configuração dessa conta passou a superar o de vários serviços. A conformidade só exige histórico em tempo real das mudanças de security groups, buckets S3 e roles IAM, e um retrato diário das instâncias EC2 é suficiente. Qual configuração do gravador dessa conta atende a esses requisitos com o menor custo?

A. Manter a gravação de cada tipo de recurso suportado e mudar a frequência do gravador para diária, de modo que cada recurso efêmero gere no máximo um item de configuração por dia na conta de processamento
B. Excluir da gravação as instâncias, os volumes e as interfaces de rede e usar um trail do CloudTrail para montar o retrato diário das instâncias EC2 a partir dos eventos RunInstances
C. Desligar o gravador na conta de processamento e usar regras periódicas do AWS Config, que avaliam os recursos a cada 24 horas sem gerar itens de configuração nem custos de gravação
D. Usar a estratégia de gravação de tipos específicos com security groups, buckets S3, roles IAM e instâncias EC2, frequência contínua como padrão e uma substituição de gravação diária para AWS::EC2::Instance

R2.4 · Domínio 2: Gerenciamento de configuração e IaC

Times de uma empresa multi-conta reclamam que pequenos ajustes de configuração de aplicação em produção exigem mudar código Terraform, revisar pull request e rodar o pipeline inteiro. O time de segurança exige rastreabilidade de quem alterou o quê e quando, aderência a padrões corporativos de configuração e reversão rápida de alterações problemáticas sem recriar a infraestrutura. Qual abordagem atende melhor a esses requisitos, mantendo os benefícios de IaC?

A. Manter infraestrutura e configuração em Terraform com workspaces por ambiente e passar os ajustes de parâmetros pelo mesmo pipeline com revisão de pull request
B. Migrar a gestão de configuração para StackSets do CloudFormation e permitir alterações diretas pelo console do CloudFormation quando o ajuste for urgente
C. Guardar arquivos YAML de configuração em um bucket versionado do S3 e usar uma associação do State Manager que aplica a versão mais recente nas instâncias
D. Manter a infraestrutura base em Terraform e mover as configurações de aplicação para o AWS AppConfig e o Parameter Store, com validadores e rollback por alarme

Respostas comentadas

R2.1. B) Implantar por StackSets service-managed com auto-deployment o documento SSM-SessionManagerRunShell com s3BucketName, cloudWatchLogGroupName, criptografia e streaming, negando por SCP a alteração dele

Por quê: As preferências do Session Manager (s3BucketName, s3EncryptionEnabled, cloudWatchLogGroupName, cloudWatchEncryptionEnabled, cloudWatchStreamingEnabled, kmsKeyId, idleSessionTimeout) são guardadas num documento do tipo Session chamado SSM-SessionManagerRunShell, por conta e Região. Por ser um documento, ele pode ser criado por IaC e distribuído com StackSets; as roles das instâncias precisam de permissão para gravar no bucket e no log group e para usar a chave KMS.

✗ A) Incorreta. O CloudTrail registra as chamadas de API como StartSession e TerminateSession, não o conteúdo digitado e exibido na sessão; a gravação da sessão é definida nas preferências do Session Manager.

✗ C) Incorreta. Os logs do sistema operacional não equivalem à gravação da sessão com entrada e saída, e cada conta continuaria podendo mudar as preferências do Session Manager.

✗ D) Incorreta. A Default Host Management Configuration dá às instâncias uma role padrão para o SSM Agent se comunicar com o serviço; ela não define para onde as sessões são gravadas.

> **Dica de prova:** Padronizar a gravação de sessões em escala: documento SSM-SessionManagerRunShell por StackSets. O CloudTrail registra o início e o fim da sessão, não o conteúdo.

R2.2. B) Anexar à raiz da organização uma declarative policy de EC2 com image block public access e snapshot block public access habilitados

Por quê: Declarative policies do Organizations (EC2: image block public access, snapshot block public access, VPC block public access, serial console, IMDS, allowed images) fixam a configuração do serviço em escala. Diferem das SCPs e RCPs, que limitam permissões de API por principal ou recurso.

✗ A) Incorreta. É um controle detectivo que corrige depois que o snapshot já ficou público; não impede a ação nem cobre AMIs.

✗ C) Incorreta. O block public access do S3 vale para buckets e objetos do S3; AMIs e snapshots têm controles próprios no EC2.

✗ D) Incorreta. SCPs controlam APIs por principal, não se aplicam a service-linked roles e não acompanham APIs novas; além disso, não há chave de condição para inspecionar o grupo all nessas chamadas.

> **Dica de prova:** Configuração fixa do serviço para a organização inteira, inclusive service-linked roles: declarative policy.

R2.3. D) Usar a estratégia de gravação de tipos específicos com security groups, buckets S3, roles IAM e instâncias EC2, frequência contínua como padrão e uma substituição de gravação diária para AWS::EC2::Instance

Por quê: O custo do AWS Config acompanha o número de itens de configuração gravados. O gravador permite escolher a estratégia (todos os tipos com exclusões ou tipos específicos) e a frequência de gravação, contínua ou diária, inclusive com substituições por tipo de recurso. Em contas com recursos efêmeros, gravar só o necessário e usar gravação diária onde o tempo real não é exigido reduz bastante o custo.

✗ A) Incorreta. A frequência diária para todos os tipos reduz o volume, mas tira o histórico em tempo real dos security groups, buckets e roles exigido pela conformidade, e continua gravando recursos que ninguém precisa.

✗ B) Incorreta. Eventos do CloudTrail registram chamadas de API, e não o estado de configuração das instâncias; o retrato diário exigido continua dependendo do AWS Config gravando o tipo AWS::EC2::Instance.

✗ C) Incorreta. Sem o gravador não há histórico de configuração nem avaliação das regras sobre os recursos; regras periódicas não substituem os itens de configuração exigidos pela auditoria.

> **Dica de prova:** Config caro por recursos efêmeros: grave só os tipos necessários e use frequência diária onde tempo real não é exigido.

R2.4. D) Manter a infraestrutura base em Terraform e mover as configurações de aplicação para o AWS AppConfig e o Parameter Store, com validadores e rollback por alarme

Por quê: Separar domínios de mudança acelera ajustes sem perder controle: a infraestrutura de base continua em IaC, e a configuração de aplicação vai para o AWS AppConfig, que guarda versões, aplica validadores antes do rollout, usa estratégias de implantação graduais com bake time e faz rollback automático por alarme do CloudWatch. As alterações ficam registradas no CloudTrail.

✗ A) Incorreta. Mantém o atrito que motivou a mudança: cada ajuste pequeno continua exigindo o ciclo completo de IaC, sem reversão rápida em runtime.

✗ B) Incorreta. StackSets não são uma ferramenta de configuração de aplicação, e mudanças diretas pelo console enfraquecem a rastreabilidade e geram drift.

✗ C) Incorreta. O bucket guarda versões e o State Manager aplica a configuração, mas não há validação antes do rollout, implantação gradual nem reversão automática.

> **Dica de prova:** Ajustes frequentes de configuração com validação e rollback rápido: AWS AppConfig; infraestrutura de base continua em IaC.

03

CAPÍTULO 3 · 15% DA PROVA

Domínio 3: Soluções resilientes na nuvem

Neste domínio (15%) eu preciso escolher a estratégia de alta disponibilidade e de recuperação de desastres que atende ao RTO e ao RPO pedidos pelo menor custo, e saber escalar direito (Auto Scaling, ECS, EKS, Lambda, DynamoDB, Aurora). A prova cobra muito a diferença entre falha de AZ, falha de Região e erro lógico (dados apagados), porque cada uma tem remédio diferente. Onde mais erro é achar que Multi-AZ ou réplica protegem contra exclusão acidental e escolher métrica de escala errada, como CPU num serviço que espera E/S. Também preciso lembrar dos recursos novos de resiliência: ARC (zonal shift e Region switch), FIS, Resilience Hub e restore testing do AWS Backup.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 7 macetes · 13 gatilhos · 4 questões comentadas

Tópicos-chave

Estratégias de DR

Em ordem de custo e velocidade: backup and restore, pilot light, warm standby e multi-site. Quanto mais já roda na Região de DR, menor o RTO e maior o custo. RPO quase zero e RTO de até 1 hora sem pagar ativo-ativo pede pilot light ou warm standby com dados replicados continuamente.

Erro lógico x falha de infraestrutura

Multi-AZ e réplicas replicam o erro (dados apagados). Erro lógico no RDS com RPO de minutos se resolve com PITR dos backups automatizados, que cria instância nova. No Aurora MySQL, backtrack desfaz no próprio cluster em minutos, até 72 h; Aurora PostgreSQL não tem backtrack.

RDS Multi-AZ, Multi-AZ DB cluster e réplica entre Regiões

Falha de AZ com RTO de minutos = Multi-AZ (instância). Multi-AZ DB cluster dá duas standbys legíveis e failover mais rápido (MySQL/PostgreSQL). Multi-AZ não cruza Região: entre Regiões use réplica + promoção automatizada + CNAME no Route 53, com RPO igual ao atraso de replicação.

Aurora Global Database

Switchover é para teste de DR e volta sem perda de dados; failover (allow-data-loss) é quando a Região primária está fora. Leituras saturadas no reader endpoint se resolvem com Aurora Auto Scaling de réplicas por CPU ou conexões.

DynamoDB resiliente

Global tables replicam entre Regiões; MRSC dá RPO zero e leitura forte em qualquer Região (três Regiões; sem transações e TTL). PITR volta ao segundo exato; AWS Backup acrescenta cópia entre Regiões/contas e Vault Lock. Picos súbitos e imprevisíveis = on-demand; tráfego previsível = provisionado com auto scaling.

Route 53, Global Accelerator e ARC

Failover regional por erro de aplicação = health check em endpoint profundo ou em alarme do CloudWatch (Evaluate Target Health só olha os destinos); endpoint privado = health check por alarme; vários sinais = health check calculado. IP estático, TCP/UDP e failover sem DNS = Global Accelerator (traffic dial drena Região). Problema em uma AZ = zonal shift do ARC; failover multi-Região de vários serviços sem depender da Região afetada = Region switch do ARC.

Estabilidade estática

Sobreviver à perda de uma AZ sem degradação exige capacidade pré-provisionada nas AZs restantes; mais AZs significam menos excedente. Antes de confiar no failover, confira IPs por sub-rede, ENIs por tipo de instância e capacidade reservada, e teste com o FIS. Um NAT gateway zonal para várias AZs é ponto único de falha: NAT gateway regional ou um NAT zonal por AZ.

EC2 Auto Scaling avançado

Default instance warmup evita escala exagerada enquanto instâncias aquecem; warm pool resolve inicialização lenta. Pico cíclico = predictive scaling; evento com data marcada = scheduled action; desvio = target tracking. Aplicação parada com SO saudável exige health check do ELB no grupo.

Métrica de escala certa

Serviço que espera E/S com CPU baixa e latência alta escala por ALBRequestCountPerTarget (ou backlog/latência). Worker de fila SQS escala por backlog por instância (metric math), não por CPU nem só pelo tamanho da fila.

Load balancer resiliente

Slow start (só com round robin) evita sobrecarregar instância nova; deregistration delay maior que a requisição mais longa evita corte no scale-in. NLB tem cross-zone desligado por padrão (ALB ligado); alvos desiguais por AZ pedem ligar. Sessão perdida no scale-in = sessão fora da instância (ElastiCache/DynamoDB), sem sticky sessions.

Lambda: concorrência e SnapStart

Concorrência reservada limita uma função e protege as outras sem custo; concorrência provisionada elimina cold start (paga), com scheduled scaling para pico previsível ou target tracking em ProvisionedConcurrencyUtilization. SnapStart só em versão publicada/alias, sem unicidade gerada na inicialização e não junta com provisioned concurrency.

SQS + Lambda e desacoplamento

Proteger o downstream = maximum concurrency no event source mapping, não reserved concurrency. Visibility timeout de 6x o timeout da função. Dependência que não escala junto com a entrada pede uma fila no meio; mesmo evento para vários consumidores = SNS + uma fila SQS por consumidor, cada uma com DLQ.

Retry storm

Tentativas sincronizadas de vários clientes derrubam o serviço. Remédio: poucas tentativas, backoff exponencial, jitter e idempotência.

AWS Backup e imutabilidade

Vault Lock em compliance impede apagar backups, nem o root consegue; governance pode ser removido por IAM. Recuperar com a conta da aplicação comprometida exige cópias em outra conta, em cofre imutável (logically air-gapped vault ou Vault Lock). Provar que restaura no RTO = restore testing com validação automatizada.

Armazenamento resiliente

EBS é de uma AZ só; arquivos compartilhados entre AZs com POSIX = EFS Regional; perda de Região = replicação do EFS (réplica só leitura até excluir a replicação). S3 com RPO previsível entre Regiões = CRR com Replication Time Control; endpoint único com failover ativo-passivo = Multi-Region Access Point com failover controls. Snapshots EBS por tag com cópia entre Regiões sem código = Data Lifecycle Manager.

ECS e EKS resilientes

Tarefa doente recebendo tráfego pede health check de aplicação com intervalo e limiar curtos (grace period alto piora). Tarefas em PROVISIONING esperando instância: targetCapacity abaixo de 100 dá folga. No EKS, pods pendentes com tipos variados, Spot e consolidação = Karpenter; pods curtos sem gerenciar nós = Fargate (GPU ou DaemonSet = node groups).

Elastic Disaster Recovery e Resilience Hub

Elastic Disaster Recovery replica servidores; testar o DR sem afetar produção nem replicação é um drill. Resilience Hub avalia aplicações contra metas de RTO/RPO e gera recomendações. Plano de DR repetível com aprovação = runbook do Automation (createStack + approve + executeAwsApi).

Disponibilidade composta

Componentes em série multiplicam: 99,5% x 99,5% dá cerca de 99%. Em paralelo independentes: 1 - (1 - A)^n. Cotas também são risco: alerta em % da cota que acompanha aumentos = métrica AWS/Usage + SERVICE_QUOTA() em metric math.

Macetes para memorizar

Estratégias de DR

B-P-W-M (Backup, Pilot, Warm, Multi): cada degrau liga mais coisa na Região de DR, reduz o RTO e aumenta a conta.

Erro lógico

Réplica copia até o erro: dado apagado volta com PITR ou backtrack, nunca com Multi-AZ.

Escala de Auto Scaling

Ciclo = predictive; calendário = scheduled; surpresa = target tracking.

Concorrência Lambda

Reservada = cerca (grátis); provisionada = forno pré-aquecido (pago).

Vault Lock

Compliance = cofre soldado; governance = cofre com chave do IAM.

Cross-zone

ALB já vem ligado; NLB vem desligado.

Disponibilidade

Série multiplica as disponibilidades e piora; paralelo multiplica as chances de falha (1 - (1 - A)^n) e melhora.

Tabela de memorização: não confunda

Quando é...	...e quando é
Pilot light / warm standby RTO de minutos a 1 hora com custo controlado.	**Multi-site ativo-ativo** RTO quase zero aceitando pagar ambiente completo nas duas Regiões.
Backtrack (Aurora MySQL) Desfazer erro lógico em minutos no próprio cluster, até 72 h.	**PITR** Restaurar para um instante em cluster/instância nova, levando mais tempo.
Concorrência reservada Limitar uma função e proteger as outras, sem custo.	**Concorrência provisionada** Eliminar cold start, pagando pelos ambientes prontos.

Zonal shift do ARC Problema isolado em uma AZ.	**Failover do Route 53 / Region switch** Recuperação entre Regiões.
Data Lifecycle Manager Snapshots EBS e AMIs por tag, com retenção e cópia entre Regiões.	**AWS Backup** Vários serviços, cofres, cópia entre contas e Vault Lock.
Write-through Cache com dados velhos: grava no cache junto com o banco.	**Lazy loading com TTL** Cache cheio de itens não lidos: só carrega o que é pedido e expira.
ElastiCache Multi-AZ Cache de sessão sobrevivendo à perda de uma AZ.	**Global Datastore** Cache em outra Região.

Números que caem

Item	Valor
Peso do domínio	**15%**
Janela máxima do backtrack do Aurora MySQL	**Até 72 h**
Visibility timeout com Lambda	**6x o timeout da função**
Global table MRSC do DynamoDB	**Três Regiões**
Disponibilidade em série	**99,5% x 99,5% dá cerca de 99%**
Disponibilidade em paralelo independente	**1 - (1 - A)^n**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Escolher Multi-AZ ou réplica de leitura para recuperar dados apagados por erro.
Como não cair: Eles replicam o erro; use PITR, backtrack ou backup.

⚠ **Alerta de prova:** Estender o Multi-AZ do RDS para outra Região.
Como não cair: Multi-AZ funciona dentro de uma Região; entre Regiões é réplica + promoção + DNS.

⚠ **Alerta de prova:** Promover réplica entre Regiões de um cluster Aurora com backtrack ativado.
Como não cair: Clusters com backtrack não podem ter réplica entre Regiões, e o atraso de replicação não desfaz erro lógico.

⚠ **Alerta de prova:** Reanexar o volume EBS em outra AZ num ASG de uma instância.
Como não cair: EBS pertence a uma AZ; dados compartilhados entre AZs pedem EFS.

⚠ **Alerta de prova:** Trocar o failover por registros multivalue com IPs dos ALBs.
Como não cair: IPs do ALB mudam, multivalue não define primário e o target group continua vendo as instâncias saudáveis; use failover com health check profundo.

⚠ **Alerta de prova:** Usar S3 Transfer Acceleration e trocar o endpoint no SDK para failover.
Como não cair: Transfer Acceleration só acelera transferência; failover transparente é Multi-Region Access Point.

⚠ **Alerta de prova:** Limitar a função com reserved concurrency baixa para proteger o RDS numa fila SQS.
Como não cair: Gera throttling, mensagens voltam à fila e podem ir para a DLQ; use maximum concurrency no event source mapping.

⚠ **Alerta de prova:** Ativar Fast Snapshot Restore para ter snapshots nas duas Regiões.
Como não cair: FSR só acelera restauração de snapshots existentes; cópia entre Regiões é DLM ou AWS Backup.

⚠ **Alerta de prova:** Comprar capacidade reservada do DynamoDB para picos imprevisíveis.
Como não cair: É compromisso de pagamento constante; picos súbitos com ociosidade pedem on-demand.

⚠ **Alerta de prova:** Ativar sticky sessions para não perder sessão com o cache em uma AZ.
Como não cair: Afinidade não tira o ponto único de falha; use ElastiCache Multi-AZ com failover.

⚠ **Alerta de prova:** Aceitar Vault Lock e relatórios de jobs como prova de que o backup restaura.
Como não cair: Nenhum dos dois testa restauração; use restore testing do AWS Backup.

⚠ **Alerta de prova:** Converter a fila em FIFO com um MessageGroupId por serviço para entregar o pedido a três serviços.
Como não cair: Cada mensagem da fila vai a um só consumidor; fan-out é SNS + uma fila por consumidor.

⚠ **Alerta de prova:** Criar o plano de backup na conta de cópias apontando recursos da produção por ARN.
Como não cair: O plano protege recursos da própria conta; copie da produção para um cofre de outra conta (o vault padrão não serve de destino entre contas).

⚠ **Alerta de prova:** Usar scheduled actions para picos imprevisíveis.
Como não cair: Ações agendadas servem a eventos com data marcada; picos imprevisíveis pedem target tracking com warmup.

Como cai na prova

📝 **Como cai:** RTO e RPO numéricos e restrição de custo.
Como resolver: Identifique se a falha é de AZ, de Região ou lógica. Depois escolha o degrau de DR mais barato que cumpre o RTO; descarte multi-site se o custo for restrição.

📝 **Como cai:** Escala que não acompanha a carga (latência alta, fila crescendo, instâncias aquecendo).
Como resolver: Troque a métrica (requisições por alvo, backlog por instância), ajuste warmup/warm pool e só então pense em capacidade.

📝 **Como cai:** Failover entre Regiões automático sem mudar a aplicação.
Como resolver: DNS com failover e health check profundo, Global Accelerator, Multi-Region Access Point ou Region switch do ARC, conforme o protocolo e o serviço.

📝 **Como cai:** Backup que precisa sobreviver a conta comprometida ou provar restauração.
Como resolver: Cópia em outra conta com cofre imutável; restore testing para provar o RTO.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Esgotamento de conexões do banco em picos, sem mudar código	**RDS Proxy (monitore DatabaseConnectionsCurrentlySessionPinned).**
Aplicação demora a voltar após failover Multi-AZ	**Cache de DNS (JVM) e conexões antigas; teste com reboot com failover.**
Milhões de objetos do S3 processados em paralelo	**Map distribuído do Step Functions (ItemReader, MaxConcurrency, ResultWriter).**
Banco do Beanstalk em produção	**Banco fora do ambiente (decoupled com Retain), aplicando a deletion policy antes.**
Origem sobrecarregada pelo CloudFront	**Origin Shield.**
Leituras repetidas no API Gateway / cliente abusivo	**Cache de estágio / usage plan com API key.**

Base On-Demand + Spot com menos interrupções	**OnDemandBaseCapacity + capacity-optimized + vários tipos.**
Testar resiliência com falhas controladas	**Fault Injection Service com stop conditions por alarme.**
Falha de nó no EKS sem impacto	**Diversidade de instâncias + PodDisruptionBudgets + readiness probes + RollingUpdate conservador.**
DR regional que depende de segredos e arquivos	**Replicar dados, arquivos e segredos (réplica do RDS, replicação do EFS, segredos replicados).**
Ninguém, nem o root, pode apagar o backup	**Vault Lock em modo compliance.**
Avaliar aplicações contra metas de RTO/RPO	**AWS Resilience Hub.**
RPO previsível para objetos entre Regiões	**S3 CRR com Replication Time Control.**

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

- Multi-AZ não cobre falha de Região nem erro lógico.
- DR regional não pode depender da Região que caiu: replique dados, arquivos e segredos.
- SQS + Lambda: maximum concurrency no event source mapping e visibility timeout de 6x.
- Escale pela métrica que reflete o gargalo, não por CPU por hábito.

Essenciais para a prova (checklist)

- ☐ Sei ordenar backup and restore, pilot light, warm standby e multi-site por custo e RTO.
- ☐ Sei escolher entre PITR, backtrack, Multi-AZ, Multi-AZ DB cluster e réplica entre Regiões.
- ☐ Sei a diferença entre switchover e failover no Aurora Global Database.
- ☐ Sei configurar health checks do Route 53 para endpoint privado e vários sinais.
- ☐ Sei usar warmup, warm pool, predictive e scheduled scaling.
- ☐ Sei proteger o downstream de uma fila SQS com Lambda.
- ☐ Sei usar Vault Lock, cofres em outra conta e restore testing.
- ☐ Sei quando usar zonal shift, Region switch, Global Accelerator e FIS.
- ☐ Sei calcular disponibilidade em série e em paralelo.

Questões de revisão do capítulo

R3.1 · Domínio 3: Soluções resilientes na nuvem

O serviço de cotações de uma corretora roda em instâncias EC2 num Auto Scaling group atrás de um ALB, com o target group usando o algoritmo least outstanding requests. A JVM leva cerca de 2 minutos para aquecer, e logo após cada scale-out as instâncias novas recebem uma fatia cheia de requisições, gerando picos de latência e 5xx. No scale-in, exportações que demoram até 60 segundos são cortadas com 502, porque o deregistration delay está em 5 segundos. O time quer resolver os dois problemas sem mudar o código. Qual configuração atende a esses requisitos?

A. Ativar o slow start de 120 segundos no target group mantendo least outstanding requests e aumentar o health check grace period do Auto Scaling group para 120 segundos
B. Adicionar um lifecycle hook de terminação de 60 segundos e trocar o algoritmo para weighted random com anomaly mitigation, que reduz o tráfego enviado às instâncias novas enquanto elas aquecem
C. Trocar o algoritmo do target group para round robin, ativar slow start de cerca de 120 segundos e aumentar o deregistration delay para cerca de 90 segundos
D. Configurar o default instance warmup do Auto Scaling group em 120 segundos e ativar sticky sessions no target group, para que as conexões em andamento terminem na mesma instância durante o scale-in

R3.2 · Domínio 3: Soluções resilientes na nuvem

Uma fintech mantém um banco Amazon RDS for MySQL com a retenção de backups automatizados em 0 e snapshots manuais diários, feitos às 2h por uma automação e copiados para outra Região, o que já atende ao plano de desastre regional. Ontem, às 15h40, um job com bug apagou milhares de registros de clientes, e a restauração pelo último snapshot perderia mais de 13 horas de transações. Para esse tipo de corrupção lógica, a empresa quer RPO de poucos minutos dentro da Região, sem o custo de uma segunda instância permanente. Qual solução atende a esse requisito?

A. Criar uma réplica de leitura na mesma Região e promovê-la quando um job apagar dados, já que a réplica mantém os registros que existiam antes do problema acontecer na instância primária do banco
B. Ativar os backups automatizados com retenção de 14 dias e, num incidente, restaurar com restore-db-instance-to-point-in-time para o instante anterior ao job, numa nova instância
C. Aumentar a frequência dos snapshots manuais para cada hora com o EventBridge Scheduler, copiando cada snapshot para a outra Região para manter o plano de desastre
D. Converter a instância para Multi-AZ, para que a standby síncrona em outra AZ guarde uma cópia dos dados que possa ser promovida quando houver uma corrupção lógica

R3.3 · Domínio 3: Soluções resilientes na nuvem

Uma editora mantém num sistema de arquivos Amazon EFS os arquivos editoriais usados por instâncias EC2 em várias zonas de disponibilidade de us-east-1. Hoje o sistema de arquivos usa a classe One Zone, e uma falha na zona deixaria o conteúdo indisponível. A empresa quer resistir à perda de uma zona sem intervenção e também ter uma cópia em us-west-2 para desastre regional, com RPO de cerca de 15 minutos na maior parte do tempo e failover que deixe a cópia gravável. Qual solução atende a esses requisitos?

A. Manter o sistema de arquivos One Zone e criar um plano do AWS Backup com backups a cada hora e cópia para us-west-2, restaurando um sistema Regional no desastre
B. Migrar os dados para um sistema de arquivos EFS Regional e montar o mesmo sistema de arquivos nas instâncias de us-west-2 por VPC peering entre as Regiões, usando o mesmo sistema de arquivos nas duas Regiões
C. Migrar os dados para um sistema de arquivos EFS Regional e criar uma replicação do EFS para us-west-2; no desastre, excluir a configuração de replicação para tornar a réplica gravável
D. Criar uma replicação do EFS One Zone para outro sistema One Zone numa segunda zona de us-east-1 e usar a mesma replicação para us-west-2 com failover pelo Route 53

R3.4 · Domínio 3: Soluções resilientes na nuvem

Instâncias de um Auto Scaling group de uma aplicação de varejo continuam em execução quando o processo da aplicação trava, porque o sistema operacional segue respondendo às verificações de status do EC2. O Application Load Balancer já marca essas instâncias como unhealthy no target group, mas o Auto Scaling group não as troca. Qual mudança faz o Auto Scaling substituí-las automaticamente?

A. Aumentar o health check grace period configurado no Auto Scaling group
B. Criar um alarme de StatusCheckFailed_System com a ação de recuperação do EC2
C. Adicionar um lifecycle hook de término ao Auto Scaling group da aplicação
D. Ativar as verificações de saúde do ELB no Auto Scaling group da aplicação

Respostas comentadas

R3.1. C) Trocar o algoritmo do target group para round robin, ativar slow start de cerca de 120 segundos e aumentar o deregistration delay para cerca de 90 segundos

Por quê: O slow start do target group faz alvos recém-registrados e saudáveis receberem tráfego crescente durante a duração configurada (30 a 900 segundos), mas não pode ser usado com os algoritmos least outstanding requests nem weighted random. O deregistration delay (padrão de 300 segundos) mantém o alvo em draining para concluir requisições em andamento; se a conexão for cortada antes, o cliente recebe erro 5xx.

✗ A) Incorreta. O slow start não pode ser ativado com least outstanding requests, e o grace period só adia a avaliação de saúde pelo Auto Scaling, sem mudar o corte das conexões no scale-in.

✗ B) Incorreta. Weighted random também não aceita slow start, e a anomaly mitigation reage a erros já ocorridos; o lifecycle hook atrasa a terminação, mas o ALB corta as conexões após os 5 segundos de deregistration delay.

✗ D) Incorreta. O instance warmup afeta as métricas usadas pelas políticas de escala, não a quantidade de requisições enviadas pelo ALB, e sticky sessions não impedem o corte após o deregistration delay.

> **Dica de prova:** Instância nova sobrecarregada: slow start (só com round robin). Requisições cortadas no scale-in: deregistration delay maior que a requisição mais longa.

R3.2. B) Ativar os backups automatizados com retenção de 14 dias e, num incidente, restaurar com restore-db-instance-to-point-in-time para o instante anterior ao job, numa nova instância

Por quê: Snapshots (manuais, ou copiados entre Regiões) dão RPO igual ao intervalo entre eles. Os backups automatizados do RDS combinam snapshot diário com logs de transação e permitem point-in-time recovery (PITR) dentro da retenção, de 1 a 35 dias, com RPO de minutos para erros lógicos. Multi-AZ e réplicas protegem contra falha de infraestrutura, não contra dados apagados por engano.

✗ A) Incorreta. A réplica aplica as mesmas exclusões em segundos; além disso, é uma instância permanente, o que a empresa não quer pagar.

✗ C) Incorreta. Snapshots a cada hora ainda deixariam RPO de até uma hora, longe dos poucos minutos exigidos, e multiplicam o custo das cópias entre Regiões.

✗ D) Incorreta. A standby do Multi-AZ replica de forma síncrona inclusive as exclusões feitas pelo job, então não protege contra corrupção lógica, e é uma segunda instância permanente.

> **Dica de prova:** Erro lógico (dados apagados) com RPO de minutos no RDS: PITR dos backups automatizados. Multi-AZ e réplica replicam o erro.

R3.3. C) Migrar os dados para um sistema de arquivos EFS Regional e criar uma replicação do EFS para us-west-2; no desastre, excluir a configuração de replicação para tornar a réplica gravável

Por quê: Sistemas EFS Regional guardam dados em várias zonas de uma Região, enquanto One Zone fica em uma só. Para desastre regional, a replicação do EFS mantém uma réplica somente leitura em outra Região (RPO de 15 minutos para a maioria dos sistemas, monitorado pela métrica TimeSinceLastSync). No failover, exclui-se a configuração de replicação e a réplica passa a aceitar escrita; o failback recria a replicação no sentido desejado.

✗ A) Incorreta. Backups a cada hora não atingem o RPO de cerca de 15 minutos, a perda da zona ainda derruba o acesso até uma restauração, e restaurar exige intervenção.

✗ B) Incorreta. Montar o sistema de arquivos a partir de outra Região não cria uma cópia independente: um desastre em us-east-1 deixaria os dados indisponíveis nas duas Regiões.

✗ D) Incorreta. Uma configuração de replicação tem um único destino e não resolve sozinha a alta disponibilidade entre zonas; o Route 53 não faz failover de pontos de montagem do EFS.

> **Dica de prova:** EFS: perda de AZ = classe Regional; perda de Região = replicação do EFS (réplica só leitura até excluir a replicação).

R3.4. D) Ativar as verificações de saúde do ELB no Auto Scaling group da aplicação

Por quê: Por padrão, o EC2 Auto Scaling usa só as verificações de status do EC2, que olham a instância e o host, não a aplicação. Ao ativar as verificações de saúde do ELB, o grupo passa a tratar como unhealthy a instância que falha no health check do target group e a substitui.

✗ A) Incorreta. O grace period só adia a primeira verificação de saúde de uma instância nova; não faz o grupo considerar a saúde vista pelo load balancer.

✗ B) Incorreta. A ação de recuperação reage a falhas do hardware ou do host subjacente; com o sistema operacional saudável, o alarme não dispara para uma aplicação travada.

✗ C) Incorreta. O lifecycle hook de término permite ações quando uma instância já vai ser encerrada; ele não decide que uma instância está com problema.

> **Dica de prova:** Aplicação parada com SO saudável: verificação de saúde do ELB no Auto Scaling group.

04

CAPÍTULO 4 · 15% DA PROVA

Domínio 4: Monitoramento e registro em log

Aqui eu preciso saber coletar, centralizar e transformar métricas, logs e traces (CloudWatch, CloudWatch Logs, X-Ray/ADOT, Container Insights, Prometheus gerenciado) e montar alarmes que não gerem ruído. Vale 15% e o enunciado costuma pedir o mínimo de esforço: sem agentes, sem código, sem operar cluster, sem criar job de ETL. Onde mais erro é achar que metric filter tem efeito retroativo, escolher o status check errado para recover e confundir cross-account observability (ver) com centralização de logs (copiar). Também preciso conhecer os detalhes de alarmes: treatMissingData, M de N, alta resolução e anomaly detection.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 6 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Metric filter

Transforma log em métrica: valor = $.campo, dimensão = $.campo de baixa cardinalidade e sem default value quando há dimensão. Não tem efeito retroativo: metric filter novo não tem histórico, o passado se consulta no Logs Insights. Alerta por contagem de eventos do CloudTrail num período = trail no CloudWatch Logs + metric filter + alarme.

CloudWatch Logs Insights

Consulta logs, inclusive o passado. Top N por grupo = stats com by campo, sort desc e limit N; percentil = pct(campo, 95). Buscas lentas por ID melhoram com field index policy, que vale só para eventos ingeridos depois.

Recursos novos do CloudWatch Logs

Live Tail mostra logs chegando agora com filtro e destaque. Transformer padroniza campos na ingestão sem mudar a aplicação. Data protection policy mascara PII sem mudar código (só quem tem logs:Unmask vê). Anomaly detector acha erros novos sem um filtro por padrão. Regras de centralização copiam logs de todas as contas e Regiões para uma conta central sem subscription filters.

Classes de log e retenção

Classe Standard suporta metric filter; Infrequent Access reduz custo de ingestão para logs volumosos e pouco consultados, com menos recursos e sem indexar campos. Logs quentes ficam no CloudWatch Logs com retenção curta; logs frios vão ao S3 por subscription filter com Firehose, lifecycle e Athena.

Subscription filter x export task

Fluxo contínuo de logs = subscription filter (para Firehose, Lambda ou outra conta). Histórico pontual = export task. Logs de várias contas numa só podem usar subscription filters entre contas; Config e CloudTrail não substituem logs de aplicação.

Cross-account observability (OAM)

Painel único de métricas, logs e traces de várias contas, sem copiar dados. Se o requisito é só visualizar, é OAM; se é copiar os logs para uma conta central, são regras de centralização ou subscription filters.

Alarmes: dados ausentes, M de N e resolução

Heartbeat que parou = treatMissingData breaching; métrica que só existe quando há erro = notBreaching. Reduzir alarmes de picos isolados = DatapointsToAlarm menor que EvaluationPeriods (M de N). Alarme de 10 ou 30 segundos só em métrica personalizada de alta resolução (StorageResolution 1).

Alarme composto e anomaly detection

Dezenas de páginas pelo mesmo incidente = alarme composto notificando e alarmes filhos sem ação. Latência com padrão que varia ao longo do dia = anomaly detection em percentil (p95/p99), não limite fixo na média; avaliar um só ponto continua ruidoso.

Metric math

Taxa de erro sem divisão por zero: IF(requisições > 0, 100 * erros / requisições, 0). Alerta em % da cota = AWS/Usage + SERVICE_QUOTA(). Alarme sem dados em métrica personalizada: confira se a combinação exata de dimensões é publicada.

Status checks e ações de EC2

Recover é para StatusCheckFailed_System (host); Reboot para StatusCheckFailed_Instance (sistema operacional). O status check não enxerga processo da aplicação: para saber se um processo está vivo, use procstat do CloudWatch agent e alarme em procstat_lookup_pid_count abaixo de 1.

CloudWatch agent

Configuração no Parameter Store deve ter nome iniciado por AmazonCloudWatch- para a política gerenciada funcionar. Aplicação que já emite StatsD ou collectd usa a seção statsd ou collectd sem mudar código. Auto Scaling que demora a reagir com CPU: confira se o monitoramento detalhado (1 min) está ligado.

Embedded Metric Format (EMF)

Cada combinação de dimensões declarada no _aws vira métrica personalizada cobrada. Custo alto se resolve declarando menos métricas e dimensões; o detalhe fica no log para o Logs Insights.

Tracing com X-Ray e ADOT

Instrumentação nova usa ADOT (OpenTelemetry), não o SDK do X-Ray. Serverless: tracing ativo na Lambda e no Step Functions e trace map do CloudWatch. Sampling rule com reservoir (por segundo) e rate (percentual) reduz custo e garante amostra mínima; alarme sobre subconjunto de traces = grupo do X-Ray com filter expression.

Observabilidade de contêineres

ECS por tarefa e contêiner sem agentes = Container Insights com observabilidade aprimorada. Logs de Fargate com mínimo esforço = driver awslogs (FireLens só quando o destino é outro). EKS com pouco esforço = add-on CloudWatch Observability; logs de pods = Fluent Bit em DaemonSet; alta cardinalidade sem operar Prometheus = Managed Service for Prometheus + Managed Grafana.

Monitoramento de experiência do usuário

Usuários reais no navegador = CloudWatch RUM. Roteiro agendado de teste = Synthetics. Impacto da internet por cidade e ISP sem instrumentar o cliente = Internet Monitor. Alertar pelo consumo do orçamento de erros = SLO do Application Signals + alarme de burn rate.

Logs no S3 consultados pelo Athena

Flow logs baratos no Athena = entrega no S3 em Parquet, prefixo Hive e partição por hora. Logs do ALB/CloudTrail sem manter partições = partition projection por data. Quem apagou objeto do S3 = evento de dados do CloudTrail (seletor avançado, readOnly false) + Athena.

Contributor Insights e metric streams

Top N de clientes, IPs ou chaves quentes = Contributor Insights. Métricas para parceiro sem polling = metric stream + Data Firehose (mesma conta e Região). Painel interativo para a diretoria sobre dados do Athena com e-mail agendado = Amazon Quick Sight.

Health checks e métricas de ALB/Route 53

Target_5XX é da aplicação; ELB_5XX é do balanceador. ResponseCodeMismatch = ajustar caminho e matcher (prefira endpoint de saúde real). Página de erro com HTTP 200 = string matching no health check do Route 53; as métricas do Route 53 ficam em us-east-1.

Macetes para memorizar

Recover x Reboot
System = Servidor físico = recover; Instance = Interior (SO) = reboot.

Metric filter
Metric filter olha para frente; Logs Insights olha para trás.

Dados ausentes
Silêncio de heartbeat é grito (breaching); silêncio de erro é paz (notBreaching).

Ver x copiar logs
OAM = janela para as contas; centralização = mudança dos logs para a conta central.

Experiência do usuário
RUM = gente real; Synthetics = robô agendado; Internet Monitor = estrada até a AWS.

Região fixa
Alarme de health check do Route 53 e regra do EventBridge para eventos do Trusted Advisor: sempre em us-east-1.

Tabela de memorização: não confunda

Quando é...	...e quando é
Metric filter Métrica e alarme contínuos a partir de agora.	**Logs Insights** Investigar o passado ou consultas ad hoc.
Cross-account observability Só visualizar dados de várias contas num painel.	**Regras de centralização do CloudWatch Logs** Copiar logs de todas as contas e Regiões para uma conta central.
Recover StatusCheckFailed_System (problema no host).	**Reboot** StatusCheckFailed_Instance (sistema operacional).
treatMissingData breaching Heartbeat: ausência de dado é problema.	**treatMissingData notBreaching** Métrica que só aparece quando há erro.
CloudWatch RUM Experiência de usuários reais no navegador.	**CloudWatch Synthetics** Roteiro agendado simulando o usuário.
Driver awslogs Logs do Fargate para o CloudWatch Logs com mínimo esforço.	**FireLens** Logs para outro destino.

Números que caem

Item	Valor
Peso do domínio	**15%**

Monitoramento detalhado de EC2	**1 minuto**
Alarme de alta resolução	**10 ou 30 segundos, com StorageResolution 1**
Região das métricas do Route 53 e eventos do Trusted Advisor	**us-east-1**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Colocar default value 0 no metric filter para preencher o histórico da madrugada.
Como não cair: O valor padrão só vale a partir da criação do filtro; o passado se consulta no Logs Insights.

⚠ **Alerta de prova:** Criar o alarme do health check do Route 53 na Região da aplicação.
Como não cair: As métricas do Route 53 são publicadas em us-east-1.

⚠ **Alerta de prova:** Alarme em StatusCheckFailed_Instance para detectar processo de aplicação parado.
Como não cair: O status check avalia SO e rede; processo vivo é procstat do CloudWatch agent.

⚠ **Alerta de prova:** Metric filter contando erros para detectar processo travado.
Como não cair: Processo travado não escreve log; use heartbeat com treatMissingData breaching ou procstat.

⚠ **Alerta de prova:** Principal logs.amazonaws.com sem Região na política da CMK do log group.
Como não cair: É o principal regional logs.REGIÃO.amazonaws.com com condição kms:EncryptionContext:aws:logs:arn.

⚠ **Alerta de prova:** Publicar métricas EMF com dimensões por pod, rota e tenant.
Como não cair: Cada combinação vira métrica cobrada; alta cardinalidade fica no log ou no Prometheus gerenciado.

⚠ **Alerta de prova:** Aumentar o default cooldown para o Auto Scaling reagir melhor.
Como não cair: Cooldown maior atrasa a escala; o problema costuma ser a falta do monitoramento detalhado (1 min).

⚠ **Alerta de prova:** Trocar o health check do target group do ALB para TCP.
Como não cair: ALB usa HTTP/HTTPS e TCP esconderia falhas da aplicação; ajuste caminho e matcher.

⚠ **Alerta de prova:** Mover log groups para Infrequent Access para acelerar buscas por ID.
Como não cair: A classe reduz custo e não indexa campos; use field index policy.

⚠ **Alerta de prova:** Anomaly detection avaliando um único ponto para reduzir ruído.
Como não cair: Um ponto fora da banda continua disparando; combine com M de N.

⚠ **Alerta de prova:** Sampling rule com reservoir alto para gerar métrica de falhas.
Como não cair: Sampling só define quanto registrar; alarme sobre falhas é grupo do X-Ray com filter expression.

⚠ **Alerta de prova:** Ativar eventos de dados de todos os buckets e procurar DeleteObject no histórico de eventos.
Como não cair: O histórico mostra só eventos de gerenciamento; use seletor avançado nos buckets certos e Athena.

⚠ **Alerta de prova:** Criar job do Glue para converter flow logs em Parquet.
Como não cair: A entrega dos flow logs já aceita Parquet, prefixo Hive e partição por hora.

⚠ **Alerta de prova:** Usar max e min no Logs Insights quando o pedido é média ou p95.
Como não cair: Extremos são sensíveis a outliers; use avg e pct(campo, 95).

Como cai na prova

📝 **Como cai:** Centralizar logs/métricas/traces de muitas contas com o menor esforço.
Como resolver: Pergunte: só ver ou copiar? Ver = OAM; copiar = regras de centralização ou subscription filters entre contas. Descarte pipelines Kinesis + Lambda feitos à mão.

📝 **Como cai:** Alarme com falsos positivos ou que não dispara.
Como resolver: Verifique M de N, treatMissingData, dimensões publicadas, anomaly detection em percentil e resolução da métrica.

📝 **Como cai:** Gerar métrica a partir de log ou investigar incidente que já aconteceu.
Como resolver: Futuro = metric filter; passado = Logs Insights. Contagem de eventos do CloudTrail = trail no CloudWatch Logs.

📝 **Como cai:** Observabilidade de ECS/EKS/Lambda sem mudar código.
Como resolver: Container Insights aprimorado, add-on CloudWatch Observability, Lambda Insights, ADOT; prefira a opção gerenciada a operar Prometheus e Grafana.

📝 **Como cai:** Custo alto de logs ou de consultas.
Como resolver: Classe Infrequent Access, retenção curta + S3 com Athena em Parquet e particionado, menos dimensões no EMF, sampling no X-Ray.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Dashboard que troca conta, Região ou recurso por menu	**Dashboard variables.**
Alarme chamando Lambda sem SNS	**Ação Lambda no alarme + resource policy para lambda.alarms.cloudwatch.amazonaws.com.**
Latência da aplicação atrás de ALB	**TargetResponseTime (e HTTPCode_Target_5XX_Count), com vários períodos de avaliação.**
Dados velhos na réplica	**ReplicaLag.**
Tarefas ECS em PROVISIONING com CapacityProviderReservation acima de 100 e ASG no máximo	**Aumentar o MaxSize do ASG.**
RDS sem espaço com crescimento imprevisível	**Storage autoscaling com MaxAllocatedStorage + alarme de FreeStorageSpace.**
Aviso antecipado de manutenção programada da AWS	**AWS Health no EventBridge (scheduledChange).**
Evento do Trusted Advisor	**Regra do EventBridge em us-east-1.**
Algo precisa rodar antes de a instância sair do ASG	**Lifecycle hook de término + EventBridge + complete-lifecycle-action.**
Vários consumidores disputando leitura no Kinesis	**Enhanced fan-out (mais shards resolvem escrita).**
Alertar só crash de tarefa do ECS	**lastStatus STOPPED + stopCode EssentialContainerExited + group do serviço.**
Throttling no GSI com a tabela folgada	**O GSI tem capacidade e auto scaling próprios.**
Status de pipelines de várias contas em um lugar	**Eventos do CodePipeline no EventBridge + event bus central com resource policy.**

Arquivo de log chegou no S3 e precisa ir a outro serviço sem polling	**S3 Event Notification + Lambda.**

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

- Metric filter não é retroativo.
- Recover para status check de sistema; reboot para status check de instância.
- Instrumentação nova de tracing é ADOT, não o SDK do X-Ray.
- Ver dados de várias contas é OAM; copiar logs é centralização ou subscription filter.

Essenciais para a prova (checklist)

- ☐ Sei criar metric filter com valor e dimensão a partir de JSON.
- ☐ Sei escrever consulta de Top N e percentil no Logs Insights.
- ☐ Sei configurar treatMissingData, M de N, alarme composto e anomaly detection.
- ☐ Sei diferenciar OAM, regras de centralização, subscription filter e export task.
- ☐ Sei escolher entre Container Insights, CloudWatch Observability, Prometheus gerenciado e ADOT.
- ☐ Sei usar procstat, statsd e collectd no CloudWatch agent.
- ☐ Sei quando usar RUM, Synthetics, Internet Monitor e Application Signals.
- ☐ Sei montar consultas baratas no Athena (Parquet, partição, partition projection).
- ☐ Sei usar data protection policy, transformer, Live Tail e field index.

Questões de revisão do capítulo

R4.1 · Domínio 4: Monitoramento e registro em log

Durante a implantação canary de um serviço no Amazon ECS, a SRE de uma healthtech quer acompanhar em tempo real os logs das tarefas novas no log group do serviço, que usa a classe Standard. Ela quer ver só as linhas com ERROR, destacar as que contêm timeout e encerrar o acompanhamento ao fim dos cerca de 20 minutos da implantação, sem criar metric filters, subscription filters ou exportações. Qual solução atende a esses requisitos com o MENOR esforço?

A. Ativar o log anomaly detection no log group do serviço, que exibe em tempo real as linhas com ERROR e destaca as que contêm timeout durante a implantação
B. Executar a cada minuto uma consulta do CloudWatch Logs Insights que filtra ERROR nos últimos 5 minutos e salvar a consulta num dashboard do serviço com atualização automática
C. Iniciar uma sessão do CloudWatch Logs Live Tail no log group do serviço, com filter pattern ERROR e destaque para timeout, encerrando a sessão ao fim da implantação
D. Criar uma subscription filter do log group para uma função Lambda que repassa as linhas com ERROR a um canal de chat e removê-la ao final da implantação canary

R4.2 · Domínio 4: Monitoramento e registro em log

O sistema de faturamento de uma operadora de telecom roda numa única instância EC2 com volumes EBS, porque a licença do fornecedor está vinculada ao ID da instância e ao endereço IP privado, que também está liberado nos firewalls de parceiros. Duas vezes neste ano a instância ficou inacessível por falha no host físico, e a recuperação dependeu de alguém parar e iniciar a instância de madrugada. A equipe quer que esse tipo de falha seja corrigido sem intervenção, mantendo o ID da instância, o IP privado e o Elastic IP, e com aviso ao time. Qual configuração atende a esses requisitos?

A. Colocar a instância num Auto Scaling group com mínimo e máximo 1 e health check do EC2, para que uma instância nova seja lançada pelo launch template quando o host falhar
B. Criar um alarme do CloudWatch sobre StatusCheckFailed_Instance com a ação Recover e notificação SNS, porque essa métrica indica que a instância perdeu acesso ao host
C. Criar um alarme do CloudWatch sobre StatusCheckFailed_System com a ação Recover e notificação SNS, avaliando dois períodos consecutivos de 1 minuto
D. Criar um alarme do CloudWatch sobre StatusCheckFailed_System com a ação Reboot e notificação SNS, para a instância reiniciar com o mesmo ID

R4.3 · Domínio 4: Monitoramento e registro em log

Um e-commerce tem alarmes do CloudWatch para latência do ALB, erros 5XX, CPU do banco RDS, profundidade da fila do SQS e erros das funções Lambda, cada um notificando o plantão por um tópico do SNS. Quando o banco degrada, todos entram em ALARM juntos e o plantão recebe dezenas de páginas pelo mesmo incidente. A equipe quer uma única página quando a combinação indicar impacto ao cliente, sem perder os alarmes individuais para o diagnóstico. Qual solução atende a esses requisitos?

A. Adicionar uma filter policy na assinatura do SNS do plantão para entregar as mensagens de alarme marcadas com o atributo de severidade alta
B. Criar um alarme sobre uma expressão de metric math que soma os erros e a latência dos componentes, removendo os alarmes individuais que existem hoje para cada um deles
C. Trocar os limites estáticos de cada alarme por bandas de anomaly detection, mantendo a notificação de cada um deles no tópico do SNS do plantão
D. Criar um alarme composto com regra como ALARM(latencia) AND (ALARM(erros5xx) OR ALARM(fila)) notificando o plantão e retirar as ações dos alarmes filhos

R4.4 · Domínio 4: Monitoramento e registro em log

Uma empresa de saúde precisa criptografar os log groups de uma aplicação no CloudWatch Logs com uma chave do AWS KMS gerenciada pelo cliente, em us-east-1, e garantir que a chave só possa ser usada para os log groups daquela conta. A equipe criou uma chave simétrica com a política padrão (só a declaração da raiz da conta) e executou aws logs associate-kms-key no log group /app/prontuario, recebendo AccessDeniedException informando que a chave não existe ou não pode ser usada com o ARN do log group. Qual alteração resolve o erro e atende ao requisito de restrição?

A. Criar uma chave multi-Região, replicá-la para as Regiões em que há log groups e manter na política só a declaração da raiz da conta, que já delega permissões ao IAM
B. Conceder kms:Encrypt, kms:Decrypt e kms:GenerateDataKey na política da chave à role de execução da aplicação que grava os logs no log group, com a condição kms:ViaService de logs
C. Adicionar à política da chave uma declaração para o principal logs.us-east-1.amazonaws.com com as ações de criptografia e a condição kms:EncryptionContext:aws:logs:arn da conta
D. Adicionar à política da chave uma declaração para o principal de serviço logs.amazonaws.com, sem Região, com a condição aws:SourceAccount igual ao ID da conta

Respostas comentadas

R4.1. C) Iniciar uma sessão do CloudWatch Logs Live Tail no log group do serviço, com filter pattern ERROR e destaque para timeout, encerrando a sessão ao fim da implantação

Por quê: CloudWatch Logs Live Tail: sessão interativa, no console, CLI (start-live-tail) ou API, que transmite os eventos de log assim que são ingeridos, com filter patterns, filtros por log stream e destaque de termos. Uma sessão dura até três horas, é cobrada por minuto e vale para log groups da classe Standard. É indicado para troubleshooting e acompanhamento de implantações.

✗ A) Incorreta. O log anomaly detection identifica padrões anômalos ao longo do tempo. Não é um visualizador de logs em tempo real com filtro e destaque.

✗ B) Incorreta. Consultas repetidas não são um fluxo em tempo real, e o dashboard é mais trabalho do que o necessário para um acompanhamento pontual.

✗ D) Incorreta. Contraria o requisito de não criar subscription filters e exige uma função a escrever e remover.

> **Dica de prova:** Ver logs chegando agora, com filtro e destaque, sem criar nada = CloudWatch Logs Live Tail.

R4.2. C) Criar um alarme do CloudWatch sobre StatusCheckFailed_System com a ação Recover e notificação SNS, avaliando dois períodos consecutivos de 1 minuto

Por quê: O status check de sistema (StatusCheckFailed_System) detecta problemas que exigem a AWS para reparar, como perda de energia ou de rede no host. Um alarme do CloudWatch com a ação Recover migra a instância para hardware novo mantendo ID, IPs privados, Elastic IP e metadados; dados em memória e em instance store se perdem. Para falhas do status check de instância, a ação recomendada é o reboot.

✗ A) Incorreta. O Auto Scaling substituiria a instância por outra, com novo ID e novo IP privado, o que quebra a licença e as regras de firewall dos parceiros.

✗ B) Incorreta. A ação Recover só pode ser usada com StatusCheckFailed_System; StatusCheckFailed_Instance indica problema no sistema operacional ou na configuração da instância, para o qual a ação indicada é o reboot.

✗ D) Incorreta. O reboot mantém a instância no mesmo host físico, então não resolve uma falha do hardware subjacente; é a ação indicada para falhas do status check de instância.

> **Dica de prova:** Recover só com StatusCheckFailed_System (host); Reboot para StatusCheckFailed_Instance (sistema operacional).

R4.3. D) Criar um alarme composto com regra como ALARM(latencia) AND (ALARM(erros5xx) OR ALARM(fila)) notificando o plantão e retirar as ações dos alarmes filhos

Por quê: Alarmes compostos combinam o estado de outros alarmes em uma regra lógica e reduzem a fadiga de alertas: só o composto tem ação de notificação, enquanto os alarmes filhos seguem registrando o estado de cada componente. O recurso de supressão de ações do alarme composto também pode silenciar notificações durante janelas conhecidas, como manutenção.

✗ A) Incorreta. As notificações de alarme do CloudWatch não carregam um atributo de severidade definido pela equipe para a filter policy usar, e cada alarme de alta severidade continuaria gerando uma página separada.

✗ B) Incorreta. Somar métricas de naturezas diferentes produz um sinal sem significado claro, e remover os alarmes individuais elimina a visibilidade por componente pedida para o diagnóstico.

✗ C) Incorreta. A anomaly detection melhora o limite de cada alarme, mas eles continuam notificando um a um; em um incidente real todos disparam e o volume de páginas não muda.

> **Dica de prova:** Dezenas de páginas pelo mesmo incidente: alarme composto notificando e alarmes filhos sem ação.

R4.4. C) Adicionar à política da chave uma declaração para o principal logs.us-east-1.amazonaws.com com as ações de criptografia e a condição kms:EncryptionContext:aws:logs:arn da conta

Por quê: Para criptografar um log group com chave do KMS gerenciada pelo cliente, a política da chave deve autorizar o principal de serviço logs.REGIÃO.amazonaws.com, e a boa prática é restringir pelo contexto de criptografia kms:EncryptionContext:aws:logs:arn, que o CloudWatch Logs preenche com o ARN do log group. Só chaves simétricas são suportadas, e desativar a chave torna os logs ilegíveis.

✗ A) Incorreta. A declaração da raiz habilita políticas IAM da conta, mas o CloudWatch Logs é um principal de serviço que precisa ser autorizado explicitamente na política da chave; uma chave multi-Região não muda isso.

✗ B) Incorreta. Quem criptografa os dados do log group é o próprio CloudWatch Logs, que precisa de permissão na política da chave; permissões para a role de execução não substituem a declaração para o serviço.

✗ D) Incorreta. A documentação exige o principal de serviço regional, como logs.us-east-1.amazonaws.com, na mesma Região da chave; o principal sem Região não autoriza o CloudWatch Logs a usar a chave.

> **Dica de prova:** Log group com CMK: política da chave para logs.REGIÃO.amazonaws.com + condição kms:EncryptionContext:aws:logs:arn.

05

CAPÍTULO 5 · 14% DA PROVA

Domínio 5: Resposta a incidentes e eventos

Este domínio (14%) mede se eu sei reagir automaticamente a eventos e diagnosticar falhas: EventBridge, SNS, SQS, Step Functions, Systems Manager Automation e o troubleshooting de CloudFormation, CodeDeploy, ECS, EKS, Lambda e Auto Scaling. As questões trazem uma mensagem de erro ou um sintoma e pedem a causa e a correção com menor esforço. Onde mais erro é confundir execution role com task role, DLQ com destino on-failure e status check de sistema com o de instância. Também caio em soluções com polling por Lambda quando existe evento nativo no EventBridge.

Neste capítulo

16 tópicos-chave · 14 pegadinhas · 6 macetes · 10 gatilhos · 4 questões comentadas

Tópicos-chave

EventBridge como centro da resposta

Reagir em segundos a uma chamada de API = regra para AWS API Call via CloudTrail; Config periódico e metric filter são mais lentos. Várias reações a um alarme com filtro pelo estado anterior = regra em CloudWatch Alarm State Change. Input transformer (InputPathsMap + InputTemplate) formata a mensagem sem código.

Permissões e entrega do EventBridge

Regra que dispara (TriggeredRules) mas não roda a Lambda = falta permissão de recurso para events.amazonaws.com; a role de execução da função não resolve. Eventos não entregues vão para a DLQ do alvo. Replay do archive filtrando a regra reprocessa eventos para um só consumidor.

Lambda assíncrona: idade, tentativas e destinos

MaximumEventAgeInSeconds descarta evento velho; MaximumRetryAttempts 0 não repete após erro. Falha assíncrona com contexto e stack trace vai para o destino on-failure, não para a DLQ. Duplicatas em fila padrão ou Lambda assíncrona pedem idempotência no consumidor (DynamoDB com escrita condicional e TTL).

SQS + Lambda em incidentes

Uma mensagem ruim fazendo o lote voltar = ReportBatchItemFailures com batchItemFailures. Duplicidade = visibility timeout de 6x o timeout da função (+ batching window). DLQ redrive devolve mensagens à fila de origem com velocidade controlada após a correção; ordem por entidade com paralelismo = FIFO com MessageGroupId = ID da entidade.

Kinesis + Lambda em incidentes

IteratorAge alto com função lenta e sem erros = ParallelizationFactor ou mais shards; disputa de leitura = enhanced fan-out. Shard travado por registro ruim = bisect on error + limite de tentativas/idade + destino on-failure no event source mapping. Vários consumidores com ordem e reprocessamento = Kinesis com retenção e checkpoints.

SNS: filtros e falhas de entrega

Filter policy na assinatura entrega a cada fila só o que ela quer; escopo MessageAttributes avalia só atributos, escopo MessageBody avalia o corpo JSON. Falha de entrega sem rastro = delivery status logging + DLQ do SQS na assinatura.

Step Functions na resposta a incidentes

Erro transitório = Retry com backoff; erro definitivo ou tentativas esgotadas = Catch para o estado de compensação, com ResultPath preservando a entrada. Resposta com várias etapas, retry e histórico = EventBridge + Step Functions. Fila para Step Functions com filtro e enriquecimento sem código = EventBridge Pipes.

Troubleshooting do CloudFormation

UPDATE_ROLLBACK_FAILED: corrija a causa e use continue-update-rollback, com --resources-to-skip se o recurso não volta; cancel-update-stack só em UPDATE_IN_PROGRESS. DELETE_FAILED por bucket não vazio com dados a preservar: repita a exclusão com RetainResources (a stack não aceita atualização nesse estado). 'Failed to receive resource signal' em sub-rede isolada = endpoint de interface do CloudFormation ou NAT; erro antes do sinal = cloud-init-output.log.

Troubleshooting do ECS

Erro antes de o contêiner subir (pull da imagem, segredo, logs) = task execution role; erro do código chamando a AWS = task role. Segredo vai na seção secrets, não em environment. Tarefa morta por health check do ELB durante o aquecimento = health check grace period do serviço; shell auditado sem SSH = ECS Exec (nova implantação + ssmmessages na task role).

Troubleshooting do EKS

Pod Pending com Insufficient cpu = falta capacidade requisitável (maxSize do node group e requests). Pod sem IP com nó ocioso = limite de IP por ENI, use prefix delegation; sub-rede sem IP = sub-redes maiores ou CIDR secundário. exec format error em Graviton = imagem sem arm64, publique manifest list multiarquitetura.

Troubleshooting de ALB e health check

502 = alvo fechou/resetou a conexão (keep-alive do alvo menor que o idle timeout); 504 = alvo lento demais para o idle timeout. ResponseCodeMismatch = caminho ou matcher errado; Timeout = rede, security group ou aplicação que não responde.

Troubleshooting de EC2 e Auto Scaling

Status check de instância = reboot (e EC2Rescue se persistir); status check de sistema = recover. InsufficientInstanceCapacity = diversifique tipos; VcpuLimitExceeded = cota. Instâncias novas encerradas em ciclo com health check do ELB = grace period curto (0 pela CLI) ou falta de lifecycle hook; health check próprio da aplicação = SetInstanceHealth Unhealthy.

Troubleshooting de deploy

ScriptFailed no CodeDeploy = ler o deployments log do agente na instância (ou no CloudWatch Logs). Beanstalk quebrado após deploy: reimplante a versão anterior e investigue pelos bundle logs (eb-engine.log). Build sem pista: log e fase que falhou, depois codebuild-breakpoint com sessão do Session Manager.

Resposta automática a segurança

Access key exposta = evento AWS_RISK_CREDENTIALS_EXPOSED do Health + EventBridge + automação que inativa a chave. Evidência forense em EC2 = isolar por security group e tirar snapshots na hora; desligar a instância perde evidência. Pod comprometido = GuardDuty (EKS Protection + Runtime Monitoring) + EventBridge + Automation com aws:approve.

Resposta multi-conta

Incidente multi-conta com correção automática e registro central = EventBridge + runbooks do Automation + OpsCenter. Eventos do AWS Health de toda a organização = visão organizacional + administrador delegado + EventBridge. Achados de várias contas com resposta automática = Security Hub CSPM + EventBridge + Automation.

Investigar quem fez o quê

Mudança recente sem trail configurado = histórico de eventos do CloudTrail (90 dias, por Região). Quem fez kubectl edit no EKS = logs de audit e authenticator do plano de controle. Mudança de configuração com estado anterior e novo = evento Configuration Item Change do Config no EventBridge.

Macetes para memorizar

Roles do ECS

Execution = antes de nascer (pull, logs, segredo); Task = depois de nascer (código).

Códigos do ALB

502 = alvo bateu o telefone; 504 = alvo não atendeu a tempo.

Falhas assíncronas

DLQ guarda o corpo; on-failure guarda o corpo e a autópsia (contexto e stack trace).

Step Functions

Retry tenta de novo; Catch desiste com elegância e compensa.

Estados do CloudFormation

UPDATE_ROLLBACK_FAILED = continue-update-rollback; DELETE_FAILED = excluir de novo com RetainResources; UPDATE_IN_PROGRESS = cancel-update-stack.

Capacidade x cota

Insufficient = falta na prateleira da AZ; Limit = falta no seu cartão (cota).

Tabela de memorização: não confunda

Quando é...	...e quando é
Task execution role Agente do ECS/Fargate: pull da imagem, envio de logs, busca de segredos.	**Task role** O código da aplicação chamando serviços AWS.
Destino on-failure Guardar a falha assíncrona com contexto e stack trace.	**DLQ** Guardar só o evento ou a mensagem que não foi processada.
EventBridge (uma regra por destino) Consumidores com regras diferentes, alvos AWS variados e SaaS por HTTP; filtros sobrepostos do S3.	**SNS com filter policy** Fan-out simples para SQS, Lambda ou HTTP.
Retry Erro transitório, com backoff.	**Catch** Erro definitivo ou tentativas esgotadas, levando à compensação.
HTTP 502 no ALB Alvo fechou ou resetou a conexão.	**HTTP 504 no ALB** Alvo não respondeu dentro do idle timeout.
InsufficientInstanceCapacity Falta temporária de capacidade do tipo na AZ: diversifique tipos.	**VcpuLimitExceeded** Cota da conta atingida: peça aumento.

Números que caem

Item	Valor
Peso do domínio	**14%**

Histórico de eventos do CloudTrail	**90 dias, por Região**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Criar Lambda agendada que faz polling de ListPipelineExecutions e posta no Slack.
Como não cair: Use EventBridge (estado FAILED) + SNS + Amazon Q Developer in chat applications, sem código nem token.

⚠ **Alerta de prova:** Adicionar lambda:InvokeFunction à role de execução para o EventBridge invocar a função.
Como não cair: Quem pode invocar é definido na política de recurso da função, para events.amazonaws.com.

⚠ **Alerta de prova:** Reduzir o maxReceiveCount para 1 para isolar a mensagem ruim do lote.
Como não cair: O lote inteiro iria para a DLQ, inclusive as válidas; use ReportBatchItemFailures.

⚠ **Alerta de prova:** Reduzir o visibility timeout para as falhas voltarem mais rápido.
Como não cair: Mensagens reaparecem antes e aumentam as duplicações; use 6x o timeout da função.

⚠ **Alerta de prova:** Provisioned concurrency para acabar com throttles quando ConcurrentExecutions está travada na reserva.
Como não cair: A provisionada fica limitada pela reservada; aumente a concorrência reservada.

⚠ **Alerta de prova:** Ativar o circuit breaker para substituir tarefas em execução com conexões quebradas.
Como não cair: O circuit breaker só age durante uma implantação que não estabiliza.

⚠ **Alerta de prova:** Adicionar DeletionPolicy Retain e atualizar uma stack em DELETE_FAILED.
Como não cair: Nesse estado a stack só aceita nova exclusão; use RetainResources.

⚠ **Alerta de prova:** Excluir e recriar a stack em UPDATE_ROLLBACK_FAILED.
Como não cair: Corrija a causa e use continue-update-rollback (com --resources-to-skip se preciso).

⚠ **Alerta de prova:** Colocar nodeSelector amd64 para resolver exec format error em nós Graviton.
Como não cair: Isso impede usar os nós arm64; publique imagem multiarquitetura.

⚠ **Alerta de prova:** Mover a referência do segredo de secrets para environment com o ARN.
Como não cair: Em environment o contêiner recebe o texto do ARN; injeção do valor exige secrets.

⚠ **Alerta de prova:** Fazer replay do archive sem filtrar regras.
Como não cair: Todos os consumidores recebem de novo; filtre pela regra do consumidor afetado.

⚠ **Alerta de prova:** Filter policy com escopo MessageAttributes para filtrar campo do corpo JSON.
Como não cair: Esse escopo só lê atributos; use escopo MessageBody.

⚠ **Alerta de prova:** Pedir aumento de cota ao ver InsufficientInstanceCapacity.
Como não cair: Isso é falta de capacidade na AZ; cota gera VcpuLimitExceeded.

⚠ **Alerta de prova:** Configurar DLQ da invocação assíncrona para o registro ruim do Kinesis.
Como não cair: Event source mapping de stream usa bisect on error, limites e destino on-failure próprios.

Como cai na prova

📝 **Como cai:** Mensagem de erro literal (UPDATE_ROLLBACK_FAILED, exec format error, InsufficientInstanceCapacity, ResponseCodeMismatch, 502).
Como resolver: Associe a mensagem à causa raiz conhecida antes de ler as alternativas; descarte as que mudam capacidade ou tempo sem atacar a causa.

📝 **Como cai:** Resposta automática a evento de segurança ou operação, com menor esforço.
Como resolver: Procure o evento nativo (Health, GuardDuty, Config, CloudTrail, Alarm State Change) no EventBridge e um alvo gerenciado (Automation, Step Functions, SNS). Descarte polling por Lambda.

📝 **Como cai:** Mensagens duplicadas, perdidas ou travando a fila/stream.
Como resolver: Identifique a origem: visibility timeout, lote inteiro falhando, registro ruim no shard, falta de idempotência; aplique o recurso específico (6x, ReportBatchItemFailures, bisect, escrita condicional).

📝 **Como cai:** Contêiner que não sobe ou não acessa um serviço.
Como resolver: Antes de subir = execution role, rede (endpoints/NAT) e arquitetura da imagem; depois de subir = task role e configuração da aplicação.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Acessar banco privado sem bastion	**Session Manager com port forwarding para host remoto.**
Reagir a failover do RDS automaticamente	**Regra do EventBridge para RDS DB Instance Event (categoria failover).**
Notificação do S3 some com destino criptografado	**Política da chave KMS gerenciada pelo cliente; prefixos sobrepostos = EventBridge.**
Mesmo evento do S3 para vários destinos com filtros sobrepostos	**EventBridge habilitado no bucket + uma regra por destino.**
Rollback de instance refresh indisponível	**Grupo em $Latest/$Default ou AMI por alias do SSM; use versão numerada + desired configuration.**
Patch de emergência fora da janela	**Run Command + AWS-RunPatchBaseline Install + rate control.**
AccessDenied em sts:GetServiceBearerToken no login do CodeArtifact	**Falta essa ação na role, com a condição sts:AWSServiceName.**
Incidente durante canary	**Primeiro reverter (rollback automático por alarme), depois analisar com CloudWatch e X-Ray.**
Trace com subsegmento externo ocupando o timeout	**Timeout e retry com backoff no cliente HTTP, não mais tempo na função.**
Detecção e resposta multi-conta como código	**Cross-account observability + alarmes, EventBridge e Automation em CloudFormation.**

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

- Execution role atua antes do contêiner subir; task role atua no código.
- EventBridge invoca Lambda pela política de recurso da função, não pela role de execução.
- Em incidente durante deploy, reverter primeiro e analisar depois.
- Evidência forense: isolar e fazer snapshot, nunca desligar a instância.

Essenciais para a prova (checklist)

- ☐ Sei construir regras do EventBridge para API Call, Alarm State Change, Health e Config.
- ☐ Sei usar input transformer, archive e replay filtrado.
- ☐ Sei tratar falhas em SQS, Kinesis e Lambda assíncrona com o recurso certo.
- ☐ Sei usar Retry e Catch no Step Functions.
- ☐ Sei sair de UPDATE_ROLLBACK_FAILED e DELETE_FAILED.
- ☐ Sei diagnosticar 502, 504, ResponseCodeMismatch e Timeout.

- ☐ Sei separar execution role e task role no ECS.
- ☐ Sei resolver Pending, falta de IP e exec format error no EKS.
- ☐ Sei montar resposta automática a credencial exposta e a instância comprometida.

Questões de revisão do capítulo

R5.1 · Domínio 5: Resposta a incidentes e eventos

Um desenvolvedor de uma startup de mídia publicou por engano, num repositório público, a access key de um usuário IAM. A AWS detectou a exposição e o AWS Health gerou na conta o evento AWS_RISK_CREDENTIALS_EXPOSED. A segurança quer que, em casos assim, a chave exposta se torne inativa em segundos, a equipe seja avisada e a atividade recente da chave seja levantada, sem depender de alguém acompanhando o AWS Health Dashboard. Qual solução atende a esses requisitos?

A. Criar uma regra do EventBridge para o evento do Health com alvo num tópico SNS assinado pelo plantão, que troca a chave pelo console do IAM assim que receber o aviso da exposição
B. Ativar o GuardDuty e criar uma regra do EventBridge para achados UnauthorizedAccess:IAMUser que aciona a Lambda de contenção quando a chave exposta for usada a partir de endereços fora da AWS
C. Criar uma regra do AWS Config access-keys-rotated com maxAccessKeyAge de 1 dia e remediação automática que exclui as chaves mais antigas que o limite
D. Criar uma regra do EventBridge para o eventTypeCode AWS_RISK_CREDENTIALS_EXPOSED que aciona um fluxo do Step Functions para marcar a chave como Inactive, consultar o CloudTrail e avisar pelo SNS

R5.2 · Domínio 5: Resposta a incidentes e eventos

Uma fintech roda serviços de pagamento no Amazon ECS em 20 contas e duas Regiões. Incidentes de latência e falhas intermitentes passaram despercebidos por horas porque os alarmes do CloudWatch não eram padronizados. A equipe de SRE precisa consolidar métricas e eventos de todas as contas, definir alarmes e respostas automáticas como código versionado, correlacionar sinais de vários serviços e orquestrar playbooks de resposta, com pouco esforço operacional e boa auditoria. Qual abordagem atende melhor a esses requisitos?

A. Usar StackSets para criar alarmes padronizados em cada conta e Região, com notificações a um tópico central do SNS que aciona uma Lambda escrita para cada serviço, sem métricas consolidadas
B. Usar a CloudWatch cross-account observability com uma conta de monitoramento e definir em CloudFormation, versionados em Git, os alarmes, as regras do EventBridge e os runbooks do Automation
C. Instalar o agente do CloudWatch nas tarefas do ECS enviando métricas e logs a uma conta central, montar dashboards por serviço e criar alarmes com notificação por um tópico do SNS
D. Espelhar os logs de cada conta por subscription filters para uma conta de incidentes, processá-los com o Glue e rodar consultas agendadas do Athena que acionam Lambdas de resposta

R5.3 · Domínio 5: Resposta a incidentes e eventos

A API de cotações de uma corretora usa o Amazon API Gateway com integração síncrona a uma função Lambda. Na abertura do pregão, cerca de 3% das chamadas retornam erro aos clientes, a métrica Throttles da função sobe e ConcurrentExecutions fica travada em 50. A cota de concorrência da conta na Região é de 1.000, e as demais funções usam menos de 200 no mesmo horário. A equipe descobriu que, meses atrás, alguém configurou concorrência reservada de 50 nessa função. O que resolve o problema com o MENOR esforço, sem afetar as demais funções?

A. Aumentar a concorrência reservada da função para um valor compatível com o pico observado, por exemplo 400, mantendo a soma das reservas dentro da cota da conta e acompanhando Throttles
B. Solicitar o aumento da cota de concorrência da conta para 3.000, já que o throttling ocorre porque a conta inteira atingiu o limite regional durante a abertura do pregão
C. Ativar o throttling de estágio do API Gateway com limite de 50 requisições por segundo, para que as chamadas excedentes sejam rejeitadas antes de chegarem à função
D. Configurar provisioned concurrency de 50 no alias da função para eliminar os cold starts, já que os throttles são causados pelo tempo de inicialização dos ambientes de execução

R5.4 · Domínio 5: Resposta a incidentes e eventos

Uma empresa de logística tem alarmes do CloudWatch com o prefixo prod-critico- em várias aplicações. Quando um deles passa de OK para ALARM, a empresa quer iniciar uma máquina de estados do Step Functions que coleta diagnósticos, enviar o evento ao sistema de ITSM de um parceiro por HTTPS com autenticação por API key e publicar num tópico SNS. Nada deve disparar quando o alarme vai de INSUFFICIENT_DATA para ALARM, e a inclusão de novos destinos não pode exigir alterar cada alarme. Qual solução atende a esses requisitos?

A. Criar uma regra do EventBridge para CloudWatch Alarm State Change com detail.alarmName por prefixo prod-critico-, detail.state.value ALARM e detail.previousState.value OK, com alvos na máquina de estados, numa API destination e no tópico SNS
B. Configurar em cada alarme as ações de ALARM para a máquina de estados do Step Functions, para o endpoint HTTPS do parceiro e para o tópico SNS, com a opção de não acionar a partir de INSUFFICIENT_DATA
C. Criar um metric stream dos namespaces das aplicações para um Amazon Data Firehose com destino HTTP do parceiro e uma regra que inicia a máquina de estados quando o stream recebe valores acima do limite configurado em cada alarme
D. Criar um alarme composto com o estado ALARM de cada alarme prod-critico- e ação para um tópico SNS com assinatura HTTPS do parceiro e uma função Lambda que inicia a máquina de estados

Respostas comentadas

R5.1. D) Criar uma regra do EventBridge para o eventTypeCode AWS_RISK_CREDENTIALS_EXPOSED que aciona um fluxo do Step Functions para marcar a chave como Inactive, consultar o CloudTrail e avisar pelo SNS

Por quê: O AWS Health publica eventos específicos da conta no EventBridge (source aws.health), inclusive AWS_RISK_CREDENTIALS_EXPOSED, gerado quando a AWS encontra uma access key da conta exposta publicamente. Regras do EventBridge sobre o eventTypeCode permitem resposta automática com Lambda ou Step Functions: inativar a chave com UpdateAccessKey, levantar a atividade no histórico de eventos do CloudTrail e notificar por SNS. Depois, a chave deve ser excluída e substituída por credenciais temporárias.

✗ A) Incorreta. A notificação chega rápido, mas a contenção depende de uma pessoa agir no console, o que não atende ao requisito de tornar a chave inativa em segundos.

✗ B) Incorreta. O GuardDuty gera achados quando a credencial é usada de forma suspeita, ou seja, depois do abuso; o evento do Health permite agir assim que a exposição é detectada.

✗ C) Incorreta. A regra trata a idade da chave, não a exposição, e excluiria chaves válidas de outros usuários sem levantar a atividade da chave exposta.

> **Dica de prova:** Access key exposta: evento AWS_RISK_CREDENTIALS_EXPOSED do Health, regra do EventBridge e automação que inativa a chave (UpdateAccessKey com Status Inactive).

R5.2. B) Usar a CloudWatch cross-account observability com uma conta de monitoramento e definir em CloudFormation, versionados em Git, os alarmes, as regras do EventBridge e os runbooks do Automation

Por quê: A cross-account observability (Observability Access Manager) liga as contas de origem a uma conta de monitoramento, onde é possível ver e correlacionar métricas, logs e traces e criar alarmes sobre dados de outras contas. Com alarmes, regras do EventBridge e runbooks do Systems Manager Automation definidos em CloudFormation e versionados em Git, a detecção e a resposta ficam padronizadas, revisáveis e auditáveis.

✗ A) Incorreta. Os alarmes ficam padronizados, mas cada conta só vê as próprias métricas, e Lambdas isoladas por serviço não correlacionam sinais nem orquestram um playbook.

✗ C) Incorreta. Centraliza parte dos dados, mas os dashboards e alarmes ficam fora do código e a resposta para na notificação, sem playbooks automáticos.

✗ D) Incorreta. O processamento em lote com Glue e consultas agendadas atrasa a detecção; serve à análise forense, não à resposta quase em tempo real.

> **Dica de prova:** Detecção e resposta multi-conta como código: cross-account observability + alarmes, EventBridge e Automation em CloudFormation.

R5.3. A) Aumentar a concorrência reservada da função para um valor compatível com o pico observado, por exemplo 400, mantendo a soma das reservas dentro da cota da conta e acompanhando Throttles

Por quê: Na Lambda, a métrica Throttles indica invocações rejeitadas por falta de concorrência. A causa pode ser a cota da conta na Região ou a concorrência reservada da própria função, que reserva capacidade e também limita o máximo. ConcurrentExecutions estabilizada exatamente no valor da reserva é o sinal típico de que o teto é a reserva, e não a conta.

✗ B) Incorreta. A conta não chegou ao limite: as outras funções usam menos de 200 e esta está travada em 50 pela reserva; aumentar a cota não muda o teto da função.

✗ C) Incorreta. Limitar no API Gateway só troca o erro da Lambda por um 429 do API Gateway; os clientes continuam recebendo erros.

✗ D) Incorreta. Provisioned concurrency reduz a latência de inicialização, mas fica limitada pela concorrência reservada; com teto de 50, os throttles continuam.

> **Dica de prova:** ConcurrentExecutions travada no valor da reserva + Throttles: aumente a concorrência reservada da função.

R5.4. A) Criar uma regra do EventBridge para CloudWatch Alarm State Change com detail.alarmName por prefixo prod-critico-, detail.state.value ALARM e detail.previousState.value OK, com alvos na máquina de estados, numa API destination e no tópico SNS

Por quê: O CloudWatch publica no EventBridge o evento CloudWatch Alarm State Change a cada transição, com detail.alarmName, detail.state.value e detail.previousState.value. Uma regra com padrão por prefixo e por transição dispara vários alvos (Step Functions, SNS, SQS, Lambda, API destinations e outros) sem tocar nos alarmes, e novos destinos entram só na regra.

✗ B) Incorreta. Ações de alarme aceitam SNS, Lambda, Auto Scaling, EC2 e Systems Manager, mas não Step Functions nem endpoints HTTPS diretos, e não há opção para filtrar o estado anterior.

✗ C) Incorreta. Metric streams entregam valores de métricas, não mudanças de estado de alarme, e a regra não tem como reagir aos dados do stream.

✗ D) Incorreta. O alarme composto precisa ser editado a cada alarme novo, acrescenta código na função Lambda e não distingue a transição a partir de INSUFFICIENT_DATA.

> **Dica de prova:** Várias reações a um alarme, com filtro pelo estado anterior: regra do EventBridge em CloudWatch Alarm State Change.

06

CAPÍTULO 6 · 17% DA PROVA

Domínio 6: Segurança e conformidade

Aqui preciso dominar identidade e acesso em escala (IAM, Identity Center, roles entre contas, ABAC, permissions boundary), criptografia com KMS, segredos, proteção de rede e de borda, e os serviços de detecção e conformidade (Config, GuardDuty, Inspector, Security Hub CSPM, Macie, Access Analyzer, CloudTrail). Vale 17% e o enunciado quase sempre separa prevenir, detectar e corrigir. Onde mais erro é escolher ferramenta de detecção quando o pedido é impedir, esquecer que SCP não vale para a conta de gerenciamento e confundir Object Lock (impede mudança) com integridade do CloudTrail (prova que não mudou). Também caio em soluções com chave aws/s3 quando é preciso compartilhar entre contas.

Neste capítulo
18 tópicos-chave · 14 pegadinhas · 6 macetes · 13 gatilhos · 4 questões comentadas

Tópicos-chave

SCP, RCP e conta de gerenciamento

SCP é a restrição que nem o admin da conta pode afrouxar; não vale para a conta de gerenciamento nem para service-linked roles. Forçar que todo deploy passe pelo pipeline aprovado = SCP na OU liberando só as roles do pipeline. Config e Service Catalog não bloqueiam caminhos alternativos.

Prevenir x detectar x corrigir

Impedir a criação = SCP. Achar o que já existe e corrigir = regra do AWS Config com remediação por SSM Automation (ou conformance pack). Para exigir KMS, a regra precisa olhar a chave, não só se há criptografia; drift de stacks em escala = cloudformation-stack-drift-detection-check (exige cloudformationRoleArn).

Identity Center e federação

Funcionários em várias contas = IAM Identity Center; IdP externo com usuários e grupos sincronizados = SAML para login e SCIM para provisionamento. Acesso por atributo sem um permission set por time = attributes for access control + aws:PrincipalTag igual a aws:ResourceTag. Clientes do app com credenciais AWS = Cognito identity pool.

Credenciais sem chave de acesso

GitHub Actions = provedor OIDC no IAM + role com condição em sub e aud (sem sub, outros repositórios assumem). Servidor fora da AWS com certificado X.509 = IAM Roles Anywhere. Menor privilégio por pod no EKS = Pod Identity (ou IRSA) + IMDSv2 com hop limit 1 + RBAC mínimo + Network Policies.

Trust policy, session policy e PassRole

Restringir quem assume entre contas = aws:PrincipalArn na trust policy; MFA para humanos = aws:MultiFactorAuthPresent, nunca na declaração usada por roles de serviço. Session policy no AssumeRole restringe a sessão (interseção com a política da role). PassRole com menor privilégio = Resource com o ARN da role + iam:PassedToService.

Permissions boundary e delegação

Delegar criação de roles sem escalada = condição iam:PermissionsBoundary + negar DeleteRolePermissionsBoundary e edição da política de limite. Delegar IAM por time numa conta compartilhada = Resource por path do IAM + iam:PermissionsBoundary. ABAC (aws:ResourceTag igual a aws:PrincipalTag) evita reescrever política a cada recurso.

IAM Access Analyzer

Analyzer de acesso não utilizado acha roles, chaves e permissões sem uso na organização. Policy generation cria política de menor privilégio a partir do CloudTrail. CheckNoNewAccess barra no pipeline política que amplia permissões; analyzer com zona = organização detecta bucket público.

KMS: políticas, grants e rotação

Chave da empresa com controle de quem decifra entre contas = chave gerenciada pelo cliente com key policy entre contas; aws/s3 não é compartilhável. Acesso temporário e programático sem mexer na key policy = grant (CreateGrant + grant token + RetireGrant). Chave assimétrica, HMAC ou em custom key store tem rotação manual trocando o alias; dado cifrado no cliente e replicado = chave multi-Região criada assim desde o início.

Criptografia no S3 e no EBS

Exigir chave KMS específica sem quebrar uploads sem cabeçalho = criptografia padrão + Deny com StringNotEqualsIfExists; TLS = aws:SecureTransport. Objetos de outras contas inacessíveis ao dono = Object Ownership Bucket owner enforced. EBS criptografado por padrão é por conta e Região; em escala, regra do Config + remediação.

Segredos

Segredo com rotação automática, auditoria ou réplica em outra Região = Secrets Manager; configuração e segredo estático de baixo custo = SecureString do Parameter Store. Senha mestre do RDS com rotação sem Lambda = ManageMasterUserPassword. No EKS, Secrets Manager + ASCP (CSI driver) + IRSA ou Pod Identity; Secret do Kubernetes em Base64 não é criptografia.

Auditoria com CloudTrail

Trilha imutável = trail organizacional + S3 Object Lock em modo compliance (MFA Delete não é WORM). Provar que os logs não foram alterados = log file integrity validation (digest files) + validate-logs. Trail com SSE-KMS exige GenerateDataKey* para cloudtrail.amazonaws.com com SourceArn do trail; volume anormal de chamadas = CloudTrail Insights; auditar só alguns buckets = data events com seletores avançados.

Detecção de ameaças e vulnerabilidades

GuardDuty detecta ameaças: Runtime Monitoring para contêiner em execução, Malware Protection for S3 para uploads, InstanceCredentialExfiltration.OutsideAWS para credencial de EC2 usada de fora. Inspector cuida de vulnerabilidades (imagem no ECR, code security em pull request); suppression rule só esconde e só a administradora delegada cria. Linha do tempo de incidente = Amazon Detective.

Security Hub CSPM e Macie

Security Hub CSPM consolida achados; padrões e controles na organização inteira, multi-Região e com exceções por OU = configuração central com políticas de configuração; suprimir ou reclassificar achados sem código = automation rules. Macie com descoberta automatizada + administrador delegado dá visibilidade barata de dados sensíveis; formato próprio = custom data identifier, falso positivo = allow list.

Proteção de borda e rede

Raspagem e bots = WAF com rate-based rule + Bot Control; Shield Advanced é para DDoS (health check do Route 53 associado + proactive engagement). Mesma proteção WAF em toda a organização = Firewall Manager. Saída restrita a domínios aprovados com inspeção de SNI e Host = Network Firewall com domain list ALLOWLIST. ALB só pelo CloudFront = prefix list gerenciada origin-facing no security group.

Rede privada e acesso a instâncias

NACL personalizada que quebrou tráfego = falta regra de portas efêmeras para o retorno. Camada a camada com Auto Scaling = regra de SG referenciando o SG de origem. ECR sem NAT = ecr.api + ecr.dkr + gateway S3, com logs e STS em endpoints próprios. Bastion e SSH viram Session Manager com logging de sessão.

Certificados e TLS

ACM com validação por DNS renova sozinho; certificado importado não renova, monitore DaysToExpiry. mTLS interno sem operar CA = AWS Private CA (com cert-manager e aws-privateca-issuer no EKS); certificado de CA privada não é aceito por navegadores públicos. Listener HTTPS sem SslPolicy via IaC fica com ELBSecurityPolicy-2016-08 (TLS 1.0): use TLS13-1-2 e redirect HTTP_301 na porta 80.

Supply chain e segregação de funções

Só código assinado na Lambda = AWS Signer + code signing config em Enforce (Warn só avisa). No EKS, AWS Signer + política de admissão. Ninguém aprova e implanta sozinho = aprovação manual restrita a revisores + role de deploy assumível só pelo serviço; SoD multi-conta combina SCPs, trail organizacional com Object Lock e permission sets temporários.

Execution role x task role e invocação de Lambda

Execution role é o que o ECS faz para iniciar a tarefa; task role é o que o código faz depois. Serviço invocando Lambda = principal de serviço na política da função + aws:SourceArn/aws:SourceAccount, nunca a raiz da conta.

Macetes para memorizar

💡 Prevenir x detectar

SCP é muro; Config é câmera com faxineiro (remediação).

💡 Imutável x íntegro

Object Lock tranca a gaveta; digest file é o lacre que prova que ninguém abriu.

💡 Serviços de segurança

GuardDuty = guarda (ameaça); Inspector = inspetor de vulnerabilidade; Macie = farejador de dado sensível; Security Hub = central que junta tudo; Detective = investigador.

💡 Segredos

Gira sozinho ou viaja de Região = Secrets Manager; parado e barato = SecureString.

💡 OIDC do GitHub

Sem sub na condição, qualquer repositório entra pela porta.

💡 SCP

SCP manda em todos, menos no chefe (conta de gerenciamento) e nos robôs (service-linked roles).

Tabela de memorização: não confunda

Quando é...	...e quando é
SCP Impedir a ação em todas as contas, inclusive contra admin local.	**AWS Config** Detectar o que já existe, guardar histórico e corrigir com remediação.
S3 Object Lock (compliance) Impedir que logs sejam alterados ou apagados.	**Log file integrity validation** Provar que os logs do CloudTrail não foram alterados.
Secrets Manager Rotação automática, réplica entre Regiões, auditoria.	**Parameter Store SecureString** Configuração ou segredo estático com custo baixo.
GuardDuty Ameaças e comportamento malicioso (runtime, credenciais, malware).	**Inspector** Vulnerabilidades em imagens, instâncias e código.

WAF (rate-based + Bot Control) Raspagem, bots e abuso na camada de aplicação.	**Shield Advanced** DDoS com apoio do SRT.
Grant do KMS Acesso temporário e programático sem alterar a política.	**Key policy** Controle permanente de quem usa a chave, inclusive entre contas.
Network Firewall Saída da VPC restrita a domínios (SNI/Host).	**WAF** Proteção da entrada de aplicações web.

Números que caem

Item	Valor
Peso do domínio	**17%**
Hop limit do IMDSv2 nos nós do EKS para bloquear pods	**1**
Política padrão do listener HTTPS sem SslPolicy via IaC	**ELBSecurityPolicy-2016-08 (TLS 1.0)**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Emitir certificados do AWS Private CA para o ALB público.
Como não cair: CA privada não é confiável para navegadores públicos; use ACM público com validação por DNS.

⚠ **Alerta de prova:** Alterar a key policy com aws:CurrentTime para dar acesso temporário a um job.
Como não cair: Mexe na política que exige aprovação; acesso temporário programático é grant.

⚠ **Alerta de prova:** Usar analyzer de acesso externo para gerar a política de menor privilégio.
Como não cair: Ele avalia acesso de fora da zona de confiança; geração de política usa o CloudTrail (policy generation).

⚠ **Alerta de prova:** Criar filtros de supressão no GuardDuty para elevar a severidade de achados.
Como não cair: Supressão arquiva achados; reclassificar no Security Hub CSPM é automation rule.

⚠ **Alerta de prova:** Criar uma role padrão em cada conta para impedir políticas amplas.
Como não cair: Admin local cria outras roles; restrição que ninguém afrouxa é SCP.

⚠ **Alerta de prova:** NACL liberando os IPs atuais dos domínios aprovados.
Como não cair: NACL filtra IP e porta e os IPs mudam; use Network Firewall com domain list.

⚠ **Alerta de prova:** Liberar lambda:InvokeFunction para a raiz da conta para o SNS invocar a função.
Como não cair: Quem invoca é o principal de serviço; use sns.amazonaws.com com aws:SourceArn.

⚠ **Alerta de prova:** Role de administrador compartilhada com MFA como controle de menor privilégio.
Como não cair: MFA reforça autenticação, mas mantém excesso de permissão e perde a rastreabilidade por equipe.

⚠ **Alerta de prova:** Lambda que atualiza IP set do WAF a partir dos logs do ALB.
Como não cair: Rate-based rule faz isso nativamente e acompanha robôs que trocam de IP.

⚠ **Alerta de prova:** Hashes próprios num DynamoDB como prova de integridade dos logs.
Como não cair: Não é prova independente; use log file integrity validation do CloudTrail.

⚠ **Alerta de prova:** Endpoints de interface sem DNS privado mantendo os nomes públicos padrão.
Como não cair: Os nomes públicos continuam resolvendo para IPs inalcançáveis; habilite DNS privado.

⚠ **Alerta de prova:** Inverter as permissões entre task role e execution role.
Como não cair: Pull do ECR, logs e segredos são da execution role; S3, SQS e chamadas do código são da task role.

⚠ **Alerta de prova:** Regras de NACL nas portas 443 e 8080 para as respostas.
Como não cair: Respostas vão para portas efêmeras de quem abriu a conexão.

⚠ **Alerta de prova:** Guardar segredos criptografados no Git com git-crypt e auditar o uso da chave.
Como não cair: Segredo em Git amplia exposição (clones, forks); use Secrets Manager.

Como cai na prova

📝 **Como cai:** Requisito de impedir algo em todas as contas, inclusive contas novas e admins locais.
Como resolver: SCP (ou RCP para recursos) na OU certa; descarte roles padrão, Config e Service Catalog, que não bloqueiam.

📝 **Como cai:** Detectar desvio, corrigir automaticamente e manter histórico de conformidade.
Como resolver: AWS Config (regra ou conformance pack) com remediação por SSM Automation; GuardDuty é ameaça e Security Hub CSPM só consolida.

📝 **Como cai:** Acesso entre contas, a chave KMS ou a segredos com menor privilégio.
Como resolver: Verifique trust policy (PrincipalArn), key policy com chave do cliente, grant para acesso temporário e condições de origem (SourceArn/SourceAccount).

📝 **Como cai:** Auditoria e prova para regulador.
Como resolver: Trail organizacional + Object Lock compliance para imutabilidade e log file integrity validation para prova de não alteração.

📝 **Como cai:** Credenciais em CI/CD, servidores externos ou pods.
Como resolver: Elimine access keys: OIDC com sub e aud, IAM Roles Anywhere, Pod Identity/IRSA com IMDSv2 hop limit 1.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Roles, chaves e permissões sem uso em toda a organização	**Analyzer de acesso não utilizado do Access Analyzer.**
Alerta de uso do root em toda a organização	**Regra do EventBridge (userIdentity.type Root) por StackSets service-managed + barramento central + SNS.**
Bucket compartilhado com política gigante	**Um access point por time e bucket policy delegando aos access points.**
Residência de dados diferente por OU no Control Tower	**Region deny aplicado à OU (CT.MULTISERVICE.PV.1).**
HSM dedicado com integração nativa a S3/EBS	**KMS custom key store sobre CloudHSM.**
Alarme sobre conexões rejeitadas na VPC	**Flow log no CloudWatch Logs + metric filter (action=REJECT) + alarme.**
Logs do WAF	**CloudWatch Logs, S3 ou Firehose com prefixo aws-waf-logs-; redacted fields e logging filter.**
Pipeline multi-conta regulado	**Conta de ferramentas + chave KMS do cliente + Secrets Manager + trail organizacional com Object Lock.**
Cobertura do GuardDuty em contas novas	**Administrador delegado + auto-enable.**
Ameaça dentro do contêiner em execução	**GuardDuty Runtime Monitoring com agente automatizado.**
Mascarar ou achar dados sensíveis no S3 da organização	**Macie com descoberta automatizada + administrador delegado.**

Operar serviços de segurança fora da conta de gerenciamento	**Trusted access e administrador delegado por serviço.**
Segredo ou vulnerabilidade no código antes do merge, sem scanner próprio	**Inspector code security com gatilho de pull request.**

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

- SCP não vale para a conta de gerenciamento nem para service-linked roles.
- Chave aws/s3 não é compartilhável entre contas; use chave gerenciada pelo cliente.
- Object Lock impede mudança; integrity validation prova que não houve mudança.
- Supressão (Inspector, GuardDuty, Security Hub) esconde achados; não corrige nada.
- OIDC do GitHub sempre com condição em sub e aud.

Essenciais para a prova (checklist)

- ☐ Sei escolher entre SCP, RCP, Config e Hooks para cada requisito.
- ☐ Sei configurar Identity Center com SAML, SCIM e ABAC.
- ☐ Sei eliminar access keys com OIDC, Roles Anywhere e Pod Identity.
- ☐ Sei usar permissions boundary, session policy e iam:PassedToService.
- ☐ Sei compartilhar chaves KMS entre contas e usar grants.
- ☐ Sei montar trilha de auditoria imutável e verificável.
- ☐ Sei diferenciar GuardDuty, Inspector, Macie, Security Hub CSPM e Detective.
- ☐ Sei proteger borda e saída com WAF, Shield Advanced, Firewall Manager e Network Firewall.
- ☐ Sei escolher Secrets Manager ou SecureString pelo requisito.
- ☐ Sei garantir só código e imagens assinados com AWS Signer.

Questões de revisão do capítulo

R6.1 · Domínio 6: Segurança e conformidade

Uma empresa de mídia teve duas indisponibilidades no último ano por certificados vencidos. No primeiro caso, um certificado emitido pelo ACM com validação por e-mail não renovou porque ninguém aprovou a mensagem de renovação. No segundo, um certificado de uma CA externa, importado no ACM e usado num ALB, venceu sem aviso. A empresa quer que os certificados emitidos pelo ACM renovem sem ação humana enquanto estiverem em uso e quer ser avisada com 30 dias de antecedência sobre os importados, que continuam vindo da CA externa. Qual solução atende a esses requisitos?

A. Reimportar os certificados externos no ACM marcando a renovação gerenciada e manter a validação por e-mail nos emitidos pelo ACM, com a mensagem de aprovação redirecionada a uma lista monitorada
B. Ativar a regra gerenciada acm-certificate-expiration-check do AWS Config nos certificados emitidos pelo ACM, que dispara a renovação quando faltam 30 dias, e manter a validação por e-mail
C. Migrar os certificados importados para o AWS Private CA e emitir a partir dele certificados para o ALB público, que passam a renovar sozinhos pelo ACM e são aceitos pelos navegadores dos clientes sem validação de domínio
D. Reemitir os certificados do ACM com validação por DNS, mantendo os registros CNAME no Route 53, e criar alarmes do CloudWatch sobre a métrica DaysToExpiry dos importados com limite de 30 dias e ação num tópico SNS

R6.2 · Domínio 6: Segurança e conformidade

Uma empresa de mídia tem uma role DeployProd em cada conta de produção. Hoje a trust policy confia em arn:aws:iam::111111111111:root, a conta de ferramentas, e qualquer principal daquela conta com permissão sts:AssumeRole consegue assumi-la. A segurança quer que a role do CodePipeline e a role do CodeBuild de deploy assumam a DeployProd sem interação, e que os engenheiros de plantão, usuários IAM do grupo de emergência da conta de ferramentas, só consigam assumi-la com MFA. Como a trust policy deve ficar?

A. Manter o principal da conta 111111111111 e adicionar uma SCP na conta de ferramentas que nega sts:AssumeRole quando aws:MultiFactorAuthPresent não é true
B. Manter o principal da conta em duas declarações: uma com aws:PrincipalArn nas roles do pipeline e do build, e outra com aws:PrincipalArn nos usuários de plantão e aws:MultiFactorAuthPresent true
C. Manter o principal da conta com a condição aws:SourceAccount igual a 111111111111 e aws:MultiFactorAuthPresent igual a true na declaração
D. Trocar o principal pelos ARNs das roles do pipeline e do build e dos usuários de plantão, e anexar à DeployProd um permissions boundary que exige aws:MultiFactorAuthPresent nas sessões

R6.3 · Domínio 6: Segurança e conformidade

Um grupo financeiro tem 120 contas no AWS Organizations, em quatro Regiões, e uma conta de segurança já registrada como administrador delegado do AWS Security Hub CSPM. Hoje cada conta ativa padrões e controles por conta própria: há contas sem o padrão AWS Foundational Security Best Practices e Regiões sem o serviço. A segurança quer o FSBP ativo nas contas e Regiões, inclusive em contas novas, com exceções por OU para controles que não se aplicam, e que as contas-membro não consigam mudar isso. Qual solução atende com o MENOR esforço operacional?

A. Aplicar uma SCP que nega securityhub:BatchDisableStandards e securityhub:UpdateStandardsControl nas contas e pedir a cada time que ative o FSBP nas Regiões
B. Ativar a configuração central na conta administradora, com Região inicial e Regiões vinculadas, e criar políticas de configuração com o FSBP associadas à raiz e às OUs com exceções
C. Criar uma regra do AWS Config em cada conta que verifica se o FSBP está ativo e uma remediação automática que chama BatchEnableStandards quando a regra marcar NON_COMPLIANT
D. Ativar o auto-enable de contas novas na configuração local do administrador delegado em cada uma das Regiões e implantar por StackSets um recurso AWS::SecurityHub::Standard com o padrão FSBP

R6.4 · Domínio 6: Segurança e conformidade

Uma operadora de telecom tem 70 contas no AWS Organizations e um organization trail ativo. A segurança precisa ser alertada em até poucos minutos, por um tópico SNS da conta de segurança, sempre que o usuário root de qualquer conta, inclusive das que forem criadas, fizer login no console ou chamar uma API. A organização não usa o Amazon GuardDuty, e o orçamento não prevê ativá-lo agora. Qual solução atende a esses requisitos com o MENOR esforço operacional?

A. Implantar a regra gerenciada root-account-mfa-enabled do AWS Config por um conformance pack organizacional e notificar pelo tópico SNS da conta de segurança cada conta que ficar não conforme com a regra
B. Ativar na organização um analisador de acesso não utilizado do IAM Access Analyzer e notificar pelo SNS quando ele apontar uso recente das credenciais do usuário root em uma das contas-membro
C. Criar na conta de gerenciamento uma regra do EventBridge com detail.userIdentity.type igual a Root, já que o organization trail faz os eventos de cada conta da organização chegarem ao barramento padrão da conta de gerenciamento
D. Implantar por StackSets service-managed com auto-deployment, nas Regiões usadas, uma regra do EventBridge com detail.userIdentity.type igual a Root que envia os eventos ao barramento da conta de segurança, onde outra regra publica no SNS

Respostas comentadas

R6.1. D) Reemitir os certificados do ACM com validação por DNS, mantendo os registros CNAME no Route 53, e criar alarmes do CloudWatch sobre a métrica DaysToExpiry dos importados com limite de 30 dias e ação num tópico SNS

Por quê: O ACM renova automaticamente os certificados que emite quando estão em uso e o domínio pode ser revalidado; com validação por DNS, o registro CNAME mantido permite renovação sem intervenção. Certificados importados não são renovados pelo ACM, e para eles o ACM publica a métrica DaysToExpiry no CloudWatch e eventos de proximidade do vencimento no EventBridge, usados para alertar.

✗ A) Incorreta. O ACM não renova certificados importados, e a validação por e-mail continua dependendo de alguém aprovar cada renovação.

✗ B) Incorreta. A regra do Config apenas avalia a proximidade do vencimento e marca a conformidade; ela não renova certificados nem resolve a dependência da aprovação por e-mail.

✗ C) Incorreta. Certificados de uma CA privada não são confiáveis para navegadores públicos, então não servem para um ALB acessado por clientes na internet.

> **Dica de prova:** ACM: validação por DNS para renovação automática; importado não renova, monitore DaysToExpiry.

R6.2. B) Manter o principal da conta em duas declarações: uma com aws:PrincipalArn nas roles do pipeline e do build, e outra com aws:PrincipalArn nos usuários de plantão e aws:MultiFactorAuthPresent true

Por quê: Quem pode assumir uma role é decidido pela trust policy (a política de recurso da role) junto com a política de identidade do chamador. Confiar na raiz da conta delega a decisão aos administradores daquela conta; condições como aws:PrincipalArn restringem a roles ou usuários específicos, e aws:MultiFactorAuthPresent exige que a sessão do chamador tenha sido autenticada com MFA.

✗ A) Incorreta. A SCP valeria para os principais da conta de ferramentas, inclusive as roles do pipeline e do build, que não usam MFA; o deploy automatizado pararia, e a trust policy continuaria aberta à conta inteira.

✗ C) Incorreta. aws:SourceAccount serve quando um serviço da AWS age em nome de um recurso; ele não identifica qual role ou usuário assume a DeployProd, e exigir MFA na única declaração bloquearia as roles do pipeline e do build.

✗ D) Incorreta. Um permissions boundary limita as permissões máximas das políticas de identidade da DeployProd depois de assumida; ele não participa da decisão de quem pode assumir a role nem impõe MFA na chamada AssumeRole.

> **Dica de prova:** Restringir quem assume entre contas: aws:PrincipalArn na trust policy. MFA para humanos: aws:MultiFactorAuthPresent, nunca na declaração usada por roles de serviço.

R6.3. B) Ativar a configuração central na conta administradora, com Região inicial e Regiões vinculadas, e criar políticas de configuração com o FSBP associadas à raiz e às OUs com exceções

Por quê: Na configuração central do Security Hub CSPM, o administrador delegado escolhe uma Região inicial (home Region) e Regiões vinculadas e cria políticas de configuração que definem se o serviço fica ativo, quais padrões e quais controles, com parâmetros. As políticas são associadas à raiz, a OUs ou a contas, com herança; contas novas da OU recebem a política automaticamente, e contas gerenciadas centralmente não mudam essas configurações.

✗ A) Incorreta. A SCP impede desligar, mas não ativa o padrão; depender de cada time deixa lacunas, e contas novas continuariam sem o FSBP.

✗ C) Incorreta. Cria uma automação própria em cada conta e Região, que corrige depois do fato e não gerencia exceções de controles por OU.

✗ D) Incorreta. A configuração local precisa ser repetida por Região e não impede que as contas alterem padrões e controles; o StackSet também não trata exceções de controles por OU de forma central.

> **Dica de prova:** Padrões e controles do Security Hub CSPM na organização inteira, multi-Região e com exceções por OU: configuração central com políticas de configuração.

R6.4. D) Implantar por StackSets service-managed com auto-deployment, nas Regiões usadas, uma regra do EventBridge com detail.userIdentity.type igual a Root que envia os eventos ao barramento da conta de segurança, onde outra regra publica no SNS

Por quê: O CloudTrail entrega ao barramento padrão do EventBridge de cada conta e Região os eventos de API e de login no console, e regras com padrão sobre detail.userIdentity.type Root detectam o uso do root. Para cobrir a organização inclusive contas futuras, implante as regras com StackSets service-managed e auto-deployment e centralize os eventos num barramento da conta de segurança. A boa prática é ainda centralizar o gerenciamento do root e remover credenciais root das contas-membro.

✗ A) Incorreta. A regra verifica se o root tem MFA ativado, e não se ele foi usado; não haveria alerta quando o root fizesse login ou chamasse uma API.

✗ B) Incorreta. O analisador de acesso não utilizado reporta permissões e credenciais sem uso por período; ele não gera alertas de uso do root em minutos.

✗ C) Incorreta. O organization trail entrega arquivos de log no S3 e, se configurado, no CloudWatch Logs; ele não encaminha os eventos de cada conta ao barramento do EventBridge da conta de gerenciamento.

> **Dica de prova:** Alerta de uso do root em toda a organização: regra do EventBridge (userIdentity.type Root) por StackSets service-managed + barramento central + SNS.

DIA DA PROVA

Estratégia de prova

A DOP-C02 tem 75 questões em 180 minutos (65 pontuadas e 10 não pontuadas, sem identificação), de múltipla escolha e múltipla resposta, com pontuação de 100 a 1000 e mínimo de 750 no modelo compensatório: não é preciso passar em cada domínio, só no total. Os enunciados são cenários longos de empresas com várias contas, pipelines e serviços, e a resposta certa quase sempre é o recurso nativo e gerenciado que atende a todas as restrições ao mesmo tempo. Os distratores são plausíveis: muitos funcionam tecnicamente, mas violam uma restrição (adicionam um serviço proibido, exigem código próprio, dobram a frota, mudam a aplicação). Expressões como menor esforço operacional, sem gerenciar servidores e sem alterar o código decidem a questão. Questão em branco conta como erro e não há penalidade por chute.

Como responder

✓ Leia a última frase primeiro (o que pedem e se é Escolha DUAS) e depois sublinhe as restrições do cenário.

✓ Classifique o requisito antes de olhar as alternativas: prevenir, detectar, corrigir, implantar, reverter ou recuperar.

✓ Elimine alternativas com Lambda de polling, scripts próprios ou servidores autogerenciados quando existe recurso nativo.

✓ Desconfie de alternativas que 'atendem tecnicamente' mas acrescentam o serviço que a empresa disse que não quer.

✓ Em erros literais (UPDATE_ROLLBACK_FAILED, 502, E401, exec format error), associe a mensagem à causa antes de ler as opções.

✓ Em RTO/RPO, identifique primeiro o tipo de falha (AZ, Região ou erro lógico) e depois o degrau de DR mais barato.

✓ São 75 questões em 180 minutos: controle o tempo para chegar ao fim e nunca deixe questão em branco, porque em branco conta como erro e chute não é penalizado.

✓ Priorize o estudo pelo peso: Automação do SDLC (22%) e depois IaC e Segurança (17% cada).

Erros que derrubam candidatos

✗ Escolher ferramenta de detecção (Config, GuardDuty) quando o enunciado pede impedir (SCP).

✗ Trocar o hook do ciclo de implantação, testando em AfterInstall ou PRE_SCALE_UP em vez do estágio com tráfego de teste.

✗ Achar que Multi-AZ ou réplica recuperam dados apagados por erro lógico.

✗ Usar a chave aws/s3 em cenários entre contas, onde só a chave gerenciada pelo cliente funciona.

✗ Confundir task execution role com task role no ECS.

✗ Acreditar que metric filter ou default value preenchem o passado.

✗ Ignorar que SCP não vale para a conta de gerenciamento.

Na semana da prova

- ☐ Refaça as provas simuladas e revise cada alternativa errada, não só as questões que errou.
- ☐ Releia as seções de pegadinhas e comparações dos domínios 1, 2 e 6, que somam mais da metade da prova.
- ☐ Revise as listas de hooks, estágios do blue/green e modos de execução do CodePipeline até saber de cor.
- ☐ Treine o ritmo num simulado completo de 75 questões em 180 minutos.
- ☐ Na véspera, revise só os blocos de lembrar e números e durma bem.

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: Automação do SDLC

‣ toomanyrequests: You have reached your pull rate limit no build → **Pull through cache do ECR com segredo ecr-pullthroughcache/ no Secrets Manager.**

‣ Ligar funcionalidade aos poucos sem novo deploy, com rollback por alarme → **Feature flag no AWS AppConfig + estratégia de implantação com bake time.**

‣ Teste de UI roteirizado com navegador, capturas e HAR, reaproveitado como monitor da produção → **Canary do CloudWatch Synthetics.**

‣ Segredo no template do CloudFormation versionado no Git → **Dynamic reference ao Secrets Manager; NoEcho só esconde a exibição.**

‣ Mesmo template em vários ambientes no CodePipeline → **Um arquivo de configuração por ambiente no TemplateConfiguration.**

⚠ Rolling substitui tarefas no mesmo target group e não tem estágio de teste; tráfego de teste exige blue/green (nativo ou com CodeDeploy).

⚠ No PRE_SCALE_UP a revisão green ainda não subiu; testes com tráfego de teste vão no POST_TEST_TRAFFIC_SHIFT.

⚠ Atende tecnicamente, mas viola a restrição; o controlador ECS já faz blue/green com hooks, tráfego de teste, bake time e rollback.

2. Domínio 2: Gerenciamento de configuração e IaC

‣ 'should have AWSCloudFormationStackSetExecutionRole' → **Falta a role de execução (ou a confiança) na conta de destino.**

‣ Configurar a conta nova do Organizations assim que estiver pronta → **Evento CreateAccountResult com SUCCEEDED, não a chamada CreateAccount.**

‣ Tarefa em todas as contas da organização → **ListAccounts + AssumeRole numa role mínima distribuída por StackSets com auto-deployment.**

‣ Unrecognized resource types com tipo de terceiro → **Ativar a extensão pública em cada conta e Região.**

‣ Prévia do impacto (inclusive substituição) antes de aplicar → **Change set + aprovação manual (CHANGE_SET_REPLACE, aprovação, CHANGE_SET_EXECUTE).**

⚠ Stack policy só impede atualizar recursos protegidos; reverter por métrica é rollback triggers com monitoring time.

⚠ Essa opção não existe; crie um export novo, migre os importadores e remova o antigo, ou use Parameter Store com resolve:ssm.

⚠ Self-managed não segue contas novas; service-managed com auto-deployment faz isso sem código.

3. Domínio 3: Soluções resilientes na nuvem

‣ Esgotamento de conexões do banco em picos, sem mudar código → **RDS Proxy (monitore DatabaseConnectionsCurrentlySessionPinned).**

‣ Aplicação demora a voltar após failover Multi-AZ → **Cache de DNS (JVM) e conexões antigas; teste com reboot com failover.**

‣ Milhões de objetos do S3 processados em paralelo → **Map distribuído do Step Functions (ItemReader, MaxConcurrency, ResultWriter).**

‣ Banco do Beanstalk em produção → **Banco fora do ambiente (decoupled com Retain), aplicando a deletion policy antes.**

‣ Origem sobrecarregada pelo CloudFront → **Origin Shield.**

⚠ Eles replicam o erro; use PITR, backtrack ou backup.

⚠ Multi-AZ funciona dentro de uma Região; entre Regiões é réplica + promoção + DNS.

⚠ Clusters com backtrack não podem ter réplica entre Regiões, e o atraso de replicação não desfaz erro lógico.

4. Domínio 4: Monitoramento e registro em log

‣ Dashboard que troca conta, Região ou recurso por menu → **Dashboard variables.**

‣ Alarme chamando Lambda sem SNS → **Ação Lambda no alarme + resource policy para lambda.alarms.cloudwatch.amazonaws.com.**

‣ Latência da aplicação atrás de ALB → **TargetResponseTime (e HTTPCode_Target_5XX_Count), com vários períodos de avaliação.**

‣ Dados velhos na réplica → **ReplicaLag.**

‣ Tarefas ECS em PROVISIONING com CapacityProviderReservation acima de 100 e ASG no máximo → **Aumentar o MaxSize do ASG.**

⚠ O valor padrão só vale a partir da criação do filtro; o passado se consulta no Logs Insights.

⚠ As métricas do Route 53 são publicadas em us-east-1.

⚠ O status check avalia SO e rede; processo vivo é procstat do CloudWatch agent.

5. Domínio 5: Resposta a incidentes e eventos

‣ Acessar banco privado sem bastion → **Session Manager com port forwarding para host remoto.**

‣ Reagir a failover do RDS automaticamente → **Regra do EventBridge para RDS DB Instance Event (categoria failover).**

‣ Notificação do S3 some com destino criptografado → **Política da chave KMS gerenciada pelo cliente; prefixos sobrepostos = EventBridge.**

‣ Mesmo evento do S3 para vários destinos com filtros sobrepostos → **EventBridge habilitado no bucket + uma regra por destino.**

‣ Rollback de instance refresh indisponível → **Grupo em $Latest/$Default ou AMI por alias do SSM; use versão numerada + desired configuration.**

⚠ Use EventBridge (estado FAILED) + SNS + Amazon Q Developer in chat applications, sem código nem token.

⚠ Quem pode invocar é definido na política de recurso da função, para events.amazonaws.com.

⚠ O lote inteiro iria para a DLQ, inclusive as válidas; use ReportBatchItemFailures.

6. Domínio 6: Segurança e conformidade

‣ Roles, chaves e permissões sem uso em toda a organização → **Analyzer de acesso não utilizado do Access Analyzer.**

‣ Alerta de uso do root em toda a organização → **Regra do EventBridge (userIdentity.type Root) por StackSets service-managed + barramento central + SNS.**

‣ Bucket compartilhado com política gigante → **Um access point por time e bucket policy delegando aos access points.**

‣ Residência de dados diferente por OU no Control Tower → **Region deny aplicado à OU (CT.MULTISERVICE.PV.1).**

‣ HSM dedicado com integração nativa a S3/EBS → **KMS custom key store sobre CloudHSM.**

⚠ CA privada não é confiável para navegadores públicos; use ACM público com validação por DNS.

⚠ Mexe na política que exige aprovação; acesso temporário programático é grant.

⚠ Ele avalia acesso de fora da zona de confiança; geração de política usa o CloudTrail (policy generation).

GABARITO

Respostas do diagnóstico

D1. A) Substituir ArtifactStore por ArtifactStores, com um bucket de artefatos e a chave KMS correspondente para us-east-1 e outro bucket e chave KMS para eu-west-1 → revise o capítulo 1

Por quê: Pipelines com actions entre Regiões exigem um artifact store em cada Região onde há actions, e o CodePipeline copia os artefatos entre eles. Pelo console, os buckets padrão são criados automaticamente; pela CLI, CloudFormation ou SDK, é preciso declarar artifactStores com um bucket (e, se houver, a chave KMS) por Região, no lugar do artifactStore de Região única.

✗ B) Incorreta. A chave multi-Região não substitui o bucket de artefatos exigido em cada Região com actions; o pipeline continua com uma única artifact store.

✗ C) Incorreta. Funciona, mas cria e mantém dois pipelines e uma integração entre eles, contrariando o requisito de manter um só pipeline.

✗ D) Incorreta. O CodePipeline copia os artefatos entre os buckets das Regiões por conta própria; a action não aceita um bucket na configuração, e o pipeline continuaria sem artifact store declarado para eu-west-1.

> **Dica de prova:** Action em outra Região = um bucket de artefatos por Região (artifactStores), com a chave KMS daquela Região.

D2. A) Usar um pipeline V2 com origem no GitHub e CodeBuild para testar e gerar a imagem, publicá-la no ECR com tag imutável e fazer commit da nova tag no repositório de configuração → revise o capítulo 1

Por quê: No GitOps, o estado desejado do cluster fica em um repositório Git de configuração e um operador como o Argo CD (inclusive a capability gerenciada do EKS) reconcilia continuamente o cluster com esse repositório. O pipeline de CI testa, gera a imagem com tag imutável no ECR e promove a versão por commit nos manifests, nunca aplicando mudanças direto no cluster.

✗ B) Incorreta. A tag latest é mutável e dificulta saber qual versão está em produção, e gravar a imagem direto nos parâmetros da aplicação tira a mudança do repositório de configuração.

✗ C) Incorreta. O helm upgrade direto no cluster é uma mudança imperativa que não passa pelo repositório de configuração; o Argo CD veria divergência em relação ao Git e o histórico do ambiente deixaria de estar no Git.

✗ D) Incorreta. A ação de implantação do EKS aplica os manifests diretamente no cluster pelo pipeline, o que quebra o modelo GitOps em que o operador aplica o que está no repositório de configuração.

> **Dica de prova:** GitOps: CI gera a imagem com tag imutável e faz commit no repositório de configuração; quem aplica no cluster é o operador (Argo CD ou Flux).

D3. D) Usar um repositório Git como fonte de verdade dos manifests e um operador GitOps no cluster, como Flux ou Argo CD, que reconcilia o estado continuamente → revise o capítulo 1

Por quê: GitOps usa o modelo pull: o estado desejado fica versionado no Git e um operador dentro do cluster (Flux ou Argo CD, inclusive como capability gerenciada do EKS) compara continuamente o estado real com o repositório e corrige os desvios. Pipelines que fazem push com kubectl ou Terraform aplicam a mudança uma vez, mas não mantêm o cluster convergindo para o Git.

✗ A) Incorreta. O bucket S3 não é a fonte de verdade no Git pedida, e os StackSets implantam stacks do CloudFormation, sem reconciliar continuamente o estado dos objetos do Kubernetes.

✗ B) Incorreta. Também é um fluxo push: o pipeline aplica mudanças, mas nada reconcilia o cluster com o Git se alguém alterar o estado fora dele.

✗ C) Incorreta. É um fluxo push: o Terraform aplica as mudanças no merge, mas nada reconcilia o cluster com o Git se alguém alterar o estado diretamente entre uma execução e outra.

> **Dica de prova:** GitOps = Git como fonte da verdade + operador que puxa e reconcilia. Pipeline que só faz push no cluster não corrige desvios.

D4. D) Publicar cada template como produto do Service Catalog numa conta hub com a role AWSControlTowerBlueprintAccess e escolher o blueprint do Account Factory Customization na conta → revise o capítulo 2

Por quê: Account Factory Customization (AFC): blueprints são produtos do Service Catalog (CloudFormation ou Terraform pelo tipo External) guardados numa conta hub, na Região de origem da landing zone. Ao criar, inscrever ou atualizar uma conta, escolhe-se um blueprint (um por conta), que o Control Tower implanta por um stack set usando a role AWSControlTowerBlueprintAccess, sem pipeline próprio.

✗ A) Incorreta. O AFT é baseado em Terraform e em um pipeline próprio, exigiria reescrever os templates existentes e aumentaria a operação.

✗ B) Incorreta. O CfCT funciona, mas é um pipeline que a equipe implanta e mantém, e o manifesto associa os templates a OUs, não ao tipo de pacote escolhido na criação da conta.

✗ C) Incorreta. Stack sets com auto-deployment seguem a OU, o que obriga a usar a OU como tipo de pacote e a mover contas, justamente o que a empresa não quer.

> **Dica de prova:** Conta nascer pronta com pacote escolhido na criação e sem pipeline = AFC com blueprint. Por OU com manifesto e pipeline = CfCT. Terraform = AFT.

D5. D) Delegar à conta de automação a leitura do Organizations, chamar ListAccounts com paginação e assumir em cada conta uma role mínima criada por StackSets service-managed com auto-deployment → revise o capítulo 2

Por quê: Automação multi-conta com SDK segue um padrão: descobrir as contas pela API do Organizations (ListAccounts, paginado), assumir em cada conta uma role de confiança restrita ao principal da automação e executar a tarefa com credenciais temporárias. A política de delegação do Organizations permite que uma conta membro chame as APIs de leitura, evitando cargas na conta de gerenciamento, e StackSets service-managed com auto-deployment garantem a role em contas futuras.

✗ A) Incorreta. Access keys de longa duração em cada conta são credenciais permanentes para proteger e rotacionar, o oposto do uso de roles com credenciais temporárias do STS.

✗ B) Incorreta. A lista de contas mantida no pipeline deixa de fora contas novas, e a OrganizationAccountAccessRole tem permissões administrativas e não existe em contas convidadas, o que viola o menor privilégio.

✗ C) Incorreta. O Data Lifecycle Manager gerencia o ciclo de vida dos snapshots criados pelas próprias políticas; ele não apaga snapshots manuais criados fora dele em outras contas.

> **Dica de prova:** Tarefa em cada conta: ListAccounts + AssumeRole numa role mínima distribuída por StackSets com auto-deployment.

D6. C) Declarar Runtime, Timeout, Tracing e VpcConfig na seção Globals e criar no samconfig.toml as seções dev, hml e prod com os parâmetros de deploy, usando sam deploy --config-env → revise o capítulo 2

Por quê: No AWS SAM, a seção Globals evita repetir propriedades comuns de AWS::Serverless::Function, Api e outros recursos, e cada recurso pode sobrescrever o que precisar. O arquivo de configuração da SAM CLI (samconfig.toml) organiza parâmetros por ambiente e por comando, e --config-env seleciona o ambiente, eliminando comandos longos digitados à mão.

✗ A) Incorreta. Triplica o template e as alterações, criando divergência entre ambientes, e o script de shell vira mais um componente para manter.

✗ B) Incorreta. Cada função continua precisando de Fn::FindInMap para cada propriedade, o que mantém a repetição, e o nome da stack, a Região e o bucket por ambiente continuam fora do controle de versão.

✗ D) Incorreta. O modo --guided é interativo e depende de respostas digitadas a cada deploy, que é justamente a fonte dos erros, e cada função ainda repete as referências aos parâmetros.

> **Dica de prova:** SAM: propriedade repetida em funções = Globals; parâmetros de deploy por ambiente = samconfig.toml + --config-env.

D7. A) Executar reboot-db-instance com --force-failover e reduzir o tempo de cache de DNS da JVM, fazendo a aplicação descartar e reabrir conexões ao detectar erro → revise o capítulo 3

Por quê: No failover de uma instância RDS Multi-AZ, o endpoint continua o mesmo e o RDS atualiza o registro DNS para apontar para a standby; os clientes precisam fechar e reabrir conexões e resolver o nome de novo. Para testar, use o reboot com failover (--force-failover), recomendado para simular a falha da instância.

✗ B) Incorreta. Parar uma instância Multi-AZ não provoca failover para a standby, e a standby de uma instância Multi-AZ não tem endpoint próprio para receber conexões.

✗ C) Incorreta. A standby de uma instância Multi-AZ não é uma réplica de leitura e não pode ser promovida com promote-read-replica; além disso, trocar a string de conexão não é recuperação automática.

✗ D) Incorreta. O reboot com failover está certo, mas o IP muda quando a standby assume; fixar o IP faz a aplicação continuar apontando para o servidor antigo depois do failover.

> **Dica de prova:** Teste de failover Multi-AZ: reboot com failover. Aplicação lenta para voltar: cache de DNS (JVM) e conexões antigas.

D8. A) Warm standby: réplica do RDS entre regiões e uma cópia reduzida da aplicação já rodando, escalada no failover pelo Route 53 → revise o capítulo 3

Por quê: Pelo guia de recuperação de desastres da AWS, warm standby mantém uma versão reduzida e funcional do ambiente na região de recuperação, com dados replicados. Isso entrega RTO de minutos e RPO de segundos a um custo menor que o ativo-ativo.

✗ B) Incorreta. RPO de até uma hora e restauração lenta não atendem aos requisitos.

✗ C) Incorreta. O RPO é atendido pela réplica, mas criar e aquecer toda a camada de aplicação costuma levar dezenas de minutos, arriscando o RTO de 15 minutos.

✗ D) Incorreta. Atende com folga, mas é a opção de maior custo, fora do orçamento.

> **Dica de prova:** Ordem de custo e velocidade: backup and restore, pilot light, warm standby, multi-site. Quanto mais já roda na região de DR, menor o RTO e maior o custo.

D9. C) Associar aos registros de failover health checks do Route 53 baseados em um endpoint de saúde profundo da aplicação ou em um alarme do CloudWatch sobre os 5XX do ALB → revise o capítulo 3

Por quê: O failover do Route 53 depende do health check associado ao registro primário. Para pegar falhas que não derrubam as instâncias, o health check precisa medir o que o usuário vê: um endpoint de saúde que exercita as dependências ou um health check baseado no estado de um alarme do CloudWatch sobre os erros 5XX do ALB. Com TTL baixo, os clientes passam rapidamente para a Região secundária.

✗ A) Incorreta. Um ALB central em uma única Região elimina a redundância regional, e o acelerador só reage quando o endpoint para de responder ao seu health check.

✗ B) Incorreta. Com Evaluate Target Health, o Route 53 considera o ALB saudável enquanto houver destinos saudáveis nos target groups; como as instâncias passam no health check, a falha de 5xx não é detectada.

✗ D) Incorreta. Os IPs do ALB mudam ao longo do tempo, multivalue não define primário e secundário, e o health check do target group continua vendo as instâncias como saudáveis.

> **Dica de prova:** Failover regional por erro de aplicação: health check do Route 53 em endpoint profundo ou em alarme do CloudWatch. Evaluate Target Health só olha os destinos.

D10. B) Criar novos flow logs para o S3 com formato de arquivo Parquet, prefixos compatíveis com Hive e partições por hora, e criar as tabelas do Athena sobre esses prefixos com projeção de partições → revise o capítulo 4

Por quê: Flow logs entregues no S3 podem usar o formato Parquet, prefixos compatíveis com Hive e partições por hora, opções que reduzem o volume lido pelo Athena e o tempo das consultas em janelas curtas. Essas opções são definidas na criação do flow log, então para mudá-las é preciso criar um flow log novo. A projeção de partições do Athena evita manter partições manualmente.

✗ A) Incorreta. Troca o destino e a ferramenta de análise, com custo de ingestão no CloudWatch Logs, em vez de otimizar as consultas que a equipe já faz no Athena.

✗ C) Incorreta. Funciona, mas acrescenta a infraestrutura de streaming que a equipe quer evitar, já que a entrega direta no S3 tem essas opções.

✗ D) Incorreta. Resolve o formato, mas cria exatamente o job de ETL que a equipe não quer operar.

> **Dica de prova:** Flow logs caros no Athena: entrega no S3 em Parquet, prefixo Hive e partição por hora.

D11. C) Definir a configuração de conta containerInsights como enhanced para os clusters novos e atualizar os 15 clusters atuais com update-cluster-settings para enhanced → revise o capítulo 4

Por quê: Container Insights com observabilidade aprimorada para ECS: além das métricas de cluster e serviço, coleta métricas de tarefa e de contêiner (CPU, memória, rede e armazenamento) no EC2 e no Fargate, com dashboards prontos e drill-down. É ativado pela configuração containerInsights com valor enhanced, na conta (padrão para clusters novos) ou em cada cluster.

✗ A) Incorreta. Exige alterar cada task definition e manter sidecars, o que a empresa quer evitar.

✗ B) Incorreta. Funciona, mas exige implantar e operar coletores em cada cluster e montar os próprios dashboards.

✗ D) Incorreta. O Application Signals foca em métricas de aplicação, como latência, erros e SLOs. As métricas de infraestrutura por tarefa e contêiner vêm do Container Insights.

> **Dica de prova:** Métricas por tarefa e contêiner no ECS sem agentes = Container Insights com observabilidade aprimorada (enhanced).

D12. B) Configurar o driver de log awslogs nas task definitions para enviar stdout e stderr a log groups do CloudWatch Logs e consultá-los junto das métricas e alarmes → revise o capítulo 4

Por quê: No ECS com Fargate, o driver de log awslogs na task definition envia stdout e stderr de cada contêiner para um log group do CloudWatch Logs sem infraestrutura adicional. Os logs ficam preservados após reinícios e podem ser consultados com o CloudWatch Logs Insights e correlacionados com métricas e alarmes no mesmo serviço; o FireLens é a opção quando é preciso rotear logs para outros destinos.

✗ A) Incorreta. Uma instância dedicada com Fluentd é infraestrutura própria para operar, com patching e disponibilidade, e os logs no S3 não ficam ao lado das métricas e alarmes do CloudWatch.

✗ C) Incorreta. O Container Insights traz métricas e eventos de desempenho das tasks, mas não guarda os logs de aplicação necessários para entender o que o código fez antes da falha.

✗ D) Incorreta. O FireLens funciona, mas mandar os logs só ao S3 tira a consulta do CloudWatch, onde estão as métricas e os alarmes, e acrescenta sidecar, stream do Firehose e tabelas do Athena para manter.

> **Dica de prova:** Logs de tasks Fargate centralizados com o mínimo de esforço: driver awslogs para o CloudWatch Logs. FireLens só quando o destino é outro.

Agora, treine no formato real

Faça as 6 provas completas do curso DOP-C02: 450 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.