Aula 13 – Observabilidade: Logs, Métricas e Tracing

Imagine que você está dirigindo um carro de última geração, superpotente, mas sem nenhum painel de controle. Você não saberia a velocidade, o nível de combustível, a temperatura do motor ou se há alguma falha. É exatamente essa a sensação de gerenciar um sistema de computação serverless complexo sem observabilidade. Em um mundo onde as aplicações são distribuídas em dezenas ou centenas de funções e microsserviços, entender o que está acontecendo "por baixo do capô" não é apenas útil, é essencial para a sobrevivência do seu projeto.

Nesta aula, vamos desvendar os mistérios da observabilidade, um pilar fundamental para qualquer arquiteto ou desenvolvedor que trabalha com tecnologias emergentes na nuvem. Nosso objetivo é que, ao final, você seja capaz de compreender a importância dos logs, métricas e tracing, e como utilizá-los para diagnosticar problemas, otimizar performance e garantir a saúde de suas aplicações serverless. Prepare-se para transformar a complexidade em clareza, aprendendo a usar ferramentas poderosas como Amazon CloudWatch Logs, CloudWatch Metrics e AWS X-Ray para ter uma visão completa do seu sistema.

O Desafio da Visibilidade em Sistemas Distribuídos

No universo da computação em nuvem, especialmente com a ascensão de arquiteturas serverless e de microsserviços, a forma como construímos e operamos aplicações mudou drasticamente. Não temos mais um servidor monolítico onde podemos simplesmente fazer login e verificar arquivos de log. Agora, uma única requisição pode atravessar dezenas de serviços, funções e bancos de dados, cada um executando em seu próprio ambiente isolado. Essa descentralização traz agilidade e escalabilidade, mas também um desafio monumental: como saber o que realmente está acontecendo?

Pense em uma cidade grande e movimentada, com milhares de ruas, prédios e veículos. Se um problema acontece em um cruzamento específico, como você o identificaria rapidamente sem um sistema de monitoramento de tráfego ou câmeras de segurança? Em um sistema distribuído, cada função Lambda, cada API Gateway, cada banco de dados DynamoDB é como um ponto nessa cidade. Sem uma forma de "observar" o fluxo e o comportamento de cada um, diagnosticar uma falha ou entender um gargalo de performance pode se tornar uma tarefa quase impossível, consumindo horas preciosas e impactando a experiência do usuário.

> **É nesse cenário que a observabilidade se torna a bússola que nos guia.** Ela não é apenas sobre coletar dados, mas sobre entender o estado interno de um sistema a partir de seus dados externos. Os três pilares que nos permitem alcançar essa compreensão são os Logs, as Métricas e o Tracing, cada um oferecendo uma perspectiva única e complementar sobre a saúde e o comportamento da nossa aplicação.

Logs: O Diário de Bordo Detalhado do Sistema

Registros Textuais

Eventos que ocorrem durante a execução do software

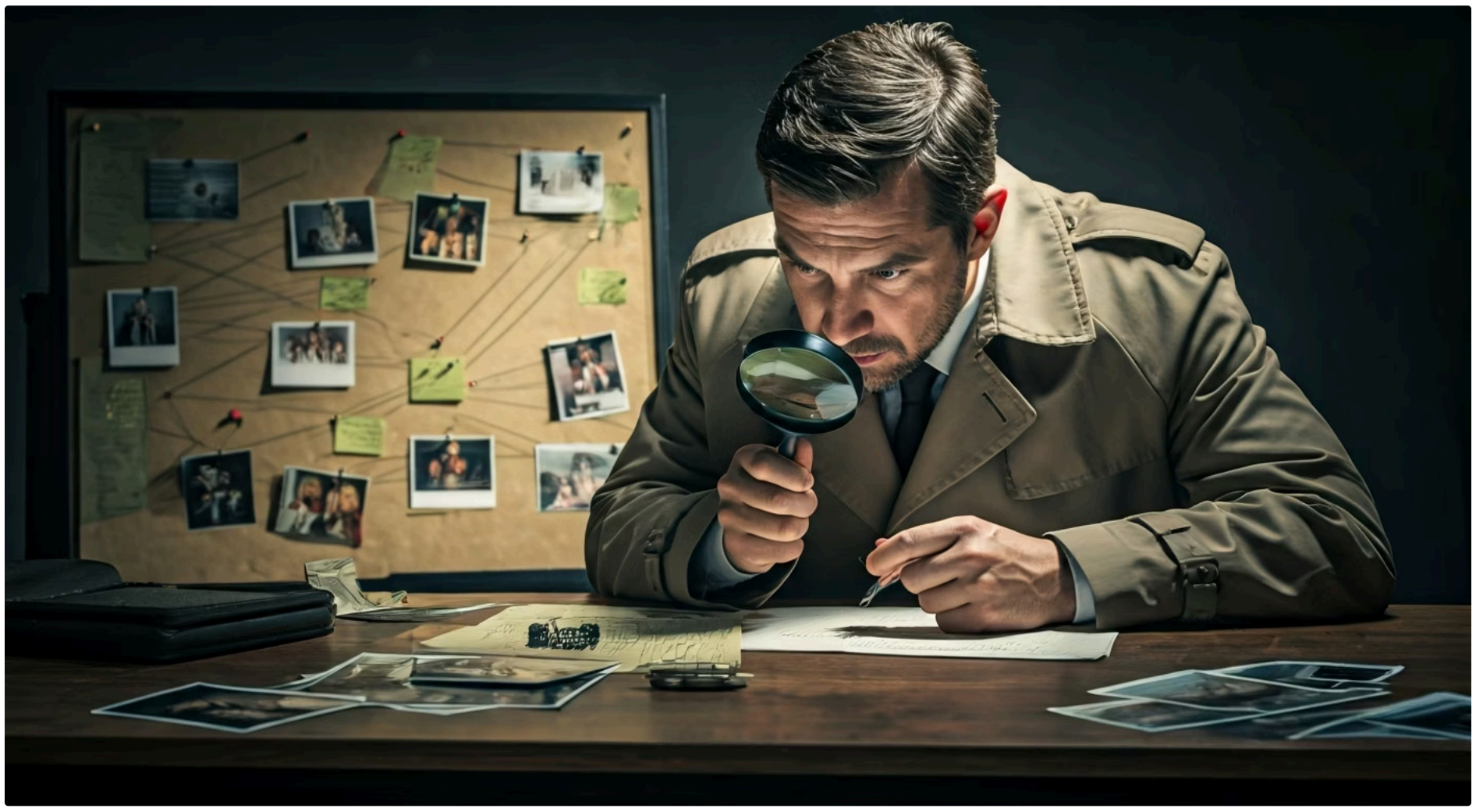
Contexto Temporal

Informações sobre o que aconteceu e quando

Detalhes Cruciais

Stack traces, erros e avisos completos

Quando algo inesperado acontece em uma aplicação, ou mesmo quando tudo está funcionando perfeitamente, a primeira coisa que geralmente procuramos é o "diário de bordo" do sistema: os logs. Eles são registros textuais de eventos que ocorrem durante a execução de um software, contendo informações cruciais sobre o que aconteceu, quando, e em que contexto. Em um ambiente serverless, cada invocação de uma função Lambda, cada requisição a uma API Gateway, cada operação em um banco de dados pode gerar logs.



Imagine que você é um detetive investigando um caso complexo. Você não se contentaria apenas com um resumo; você precisaria de cada detalhe, cada pista, cada depoimento. Os logs são esses depoimentos detalhados. Eles podem registrar desde mensagens informativas sobre o início e fim de uma operação, até avisos sobre condições anormais e, crucialmente, os erros completos com seus respectivos rastreamentos de pilha (stack traces). Sem logs bem estruturados e acessíveis, diagnosticar a causa raiz de um problema em um sistema distribuído seria como procurar uma agulha em um palheiro, às cegas.

> A importância dos logs se amplifica em arquiteturas serverless, onde as funções são efêmeras e podem ser invocadas milhares de vezes por segundo. Cada invocação é uma nova "instância" que executa e desaparece. Os logs são a única memória persistente do que ocorreu em cada uma dessas execuções.

Por exemplo, se uma função Lambda falha ao processar um item de uma fila, o log pode conter o ID do item, o erro específico e o estado da função naquele momento, permitindo que você reproduza o problema e encontre a solução.

Coletando e Analisando Logs com Amazon CloudWatch Logs

Gerenciar o volume massivo de logs gerados por um sistema serverless pode ser um desafio por si só. É aqui que ferramentas especializadas, como o Amazon CloudWatch Logs, se tornam indispensáveis. O CloudWatch Logs atua como um serviço centralizado para coletar, monitorar, armazenar e analisar logs de diversas fontes na AWS, incluindo funções Lambda, contêineres, instâncias EC2 e muito mais. Ele transforma o caos de logs dispersos em um repositório organizado e pesquisável.

01

Agregação Centralizada

Logs de múltiplas fontes reunidos em grupos de logs organizados

02

Filtragem Poderosa

Capacidades avançadas de pesquisa e filtragem em milhões de linhas

03

CloudWatch Logs Insights

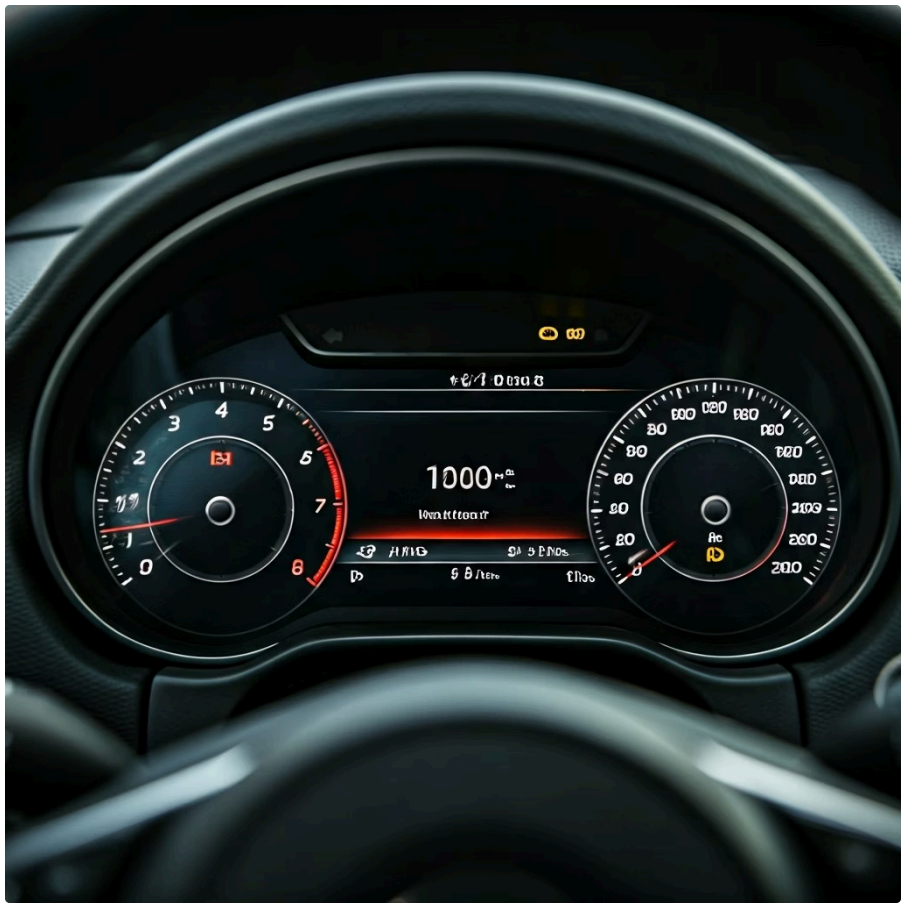
Ferramenta de consulta interativa para análise profunda

Ao invés de ter que acessar cada recurso individualmente para ver seus logs, o CloudWatch Logs agrega tudo em "grupos de logs", que são coleções lógicas de streams de logs. Cada invocação de uma função Lambda, por exemplo, envia seus logs para um stream dentro de um grupo de logs específico daquela função. Isso permite que você visualize todos os logs de uma função em um único lugar, facilitando a depuração e a auditoria. Além disso, o serviço oferece poderosas capacidades de filtragem e pesquisa, permitindo que você encontre rapidamente mensagens específicas, erros ou padrões dentro de milhões de linhas de log.

Exemplo Prático

Considere um cenário onde sua aplicação serverless está apresentando falhas intermitentes. Você pode usar o CloudWatch Logs Insights, uma ferramenta de consulta interativa, para buscar por mensagens de erro ou exceções em todos os seus grupos de logs de Lambda. Por exemplo, uma consulta simples pode filtrar logs que contenham a palavra "ERROR" ou "Exception", e até mesmo extrair campos específicos dos logs estruturados (como JSON) para análise. Essa capacidade de mergulhar nos detalhes dos logs é crucial para identificar a causa raiz de problemas complexos e garantir a estabilidade da sua aplicação.

Métricas: Os Números que Contam uma História



Enquanto os logs nos dão a história detalhada de cada evento, as métricas nos oferecem uma visão panorâmica e quantitativa do comportamento do sistema ao longo do tempo. Elas são pontos de dados numéricos coletados em intervalos regulares, representando o desempenho e a saúde de um componente ou da aplicação como um todo. Se os logs são o diário do detetive, as métricas são os sinais vitais do paciente: batimentos cardíacos, temperatura, pressão arterial.

Em um ambiente serverless, métricas são essenciais para entender a performance e a utilização dos recursos sem precisar ler cada log individualmente. Por exemplo, saber o número total de invocações de uma função Lambda, a taxa de erros, a duração média de execução ou o número de vezes que uma função foi "throttled" (limitada por exceder cotas) são informações vitais. Esses números nos permitem identificar tendências, detectar anomalias e tomar decisões proativas antes que pequenos problemas se transformem em grandes crises.

Visão Panorâmica

Comportamento do sistema ao longo do tempo em números

Identificação de Tendências

Detecte padrões e anomalias antes que se tornem problemas

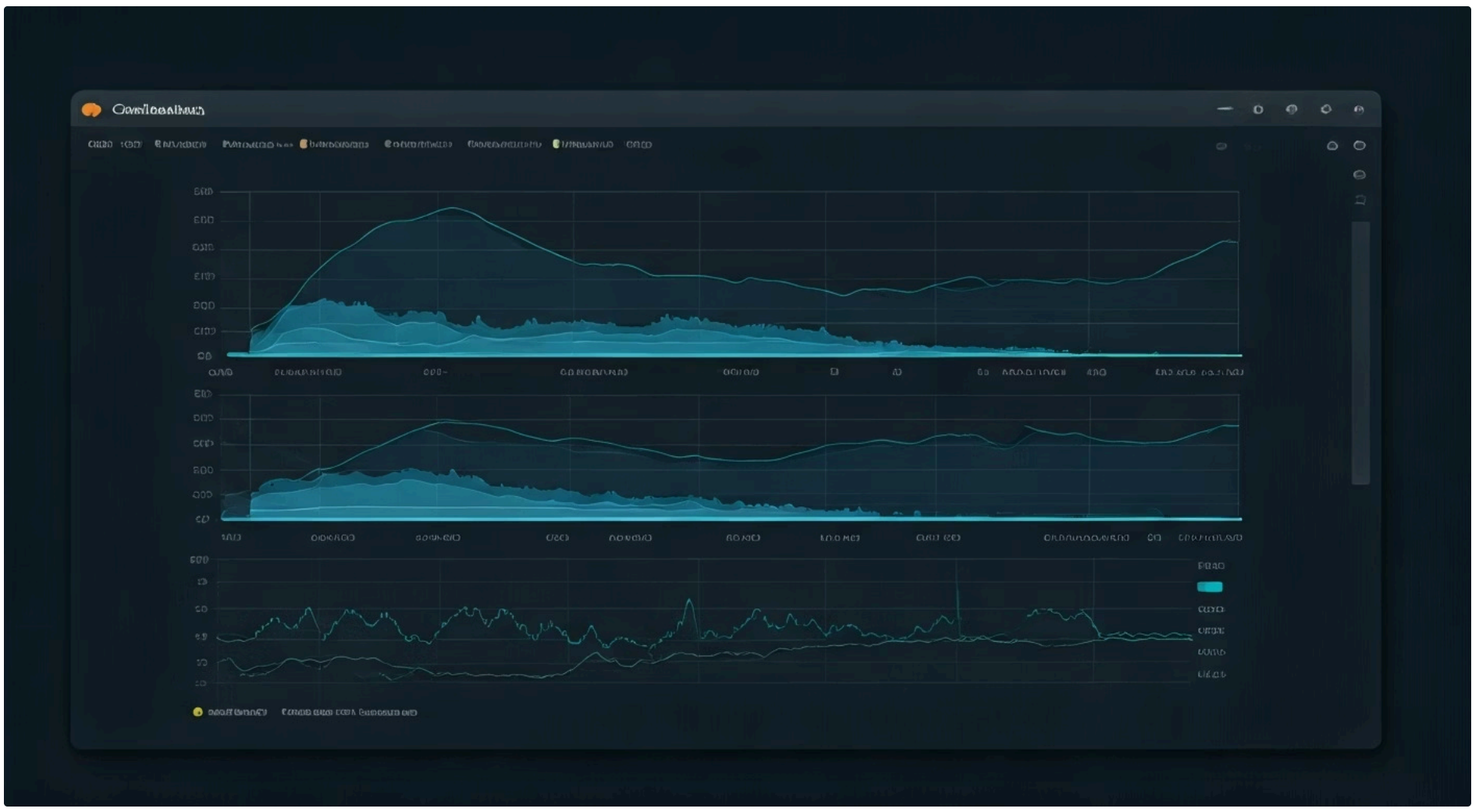
Decisões Proativas

Tome ações baseadas em dados quantitativos

> Imagine o painel de instrumentos de um carro: você não precisa saber cada explosão no motor (o log detalhado) para saber que está ficando sem combustível ou que a temperatura está alta. O velocímetro, o medidor de combustível e o termômetro (as métricas) fornecem informações rápidas e acionáveis. Da mesma forma, as métricas em serverless permitem que você veja rapidamente se sua aplicação está sob estresse, se há um pico de erros ou se a latência está aumentando, tudo isso sem precisar mergulhar nos logs, que seriam como tentar entender o motor do carro em tempo real.

Monitorando a Saúde da Aplicação com Amazon CloudWatch Metrics

Para transformar esses pontos de dados numéricos em inteligência acionável, precisamos de uma ferramenta que colete, agregue e visualize essas métricas de forma eficaz. O Amazon CloudWatch Metrics é o serviço da AWS projetado exatamente para isso. Ele coleta métricas de praticamente todos os serviços AWS, incluindo Lambda, API Gateway, DynamoDB, e muitos outros, permitindo que você crie dashboards personalizados e configure alarmes para ser notificado sobre condições anormais.



Invocations

Quantas vezes a função foi chamada

Errors

Quantas invocações resultaram em erro

Duration

Tempo que a função levou para executar

Throttles

Invocações rejeitadas por exceder limites

Com o CloudWatch Metrics, você pode monitorar a saúde de suas funções Lambda através de métricas padrão como Invocations (quantas vezes a função foi chamada), Errors (quantas invocações resultaram em erro), Duration (o tempo que a função levou para executar) e Throttles (quantas invocações foram rejeitadas por excederem os limites de concorrência). Essas métricas são automaticamente coletadas e disponibilizadas, sem a necessidade de qualquer configuração adicional em suas funções.

Exemplo Prático de Alarme

Um exemplo prático seria criar um alarme no CloudWatch que dispare sempre que a métrica Errors de uma função Lambda específica exceder um determinado limite (por exemplo, 5 erros em 5 minutos). Quando esse alarme é ativado, ele pode enviar uma notificação para um tópico SNS, que por sua vez pode enviar um e-mail, uma mensagem para um chat de equipe ou até mesmo acionar uma função Lambda para tentar uma correção automática. Isso permite uma resposta rápida a problemas, minimizando o impacto nos usuários e no negócio.

Tracing: A Jornada Completa de uma Requisição

Em sistemas distribuídos, uma única requisição de um usuário pode desencadear uma cascata de chamadas entre diferentes serviços, funções e bancos de dados. Se um usuário relata que "a aplicação está lenta", como você descobre qual parte dessa complexa cadeia de eventos é a responsável pelo gargalo? É aqui que o tracing entra em cena, oferecendo uma visão ponta a ponta da jornada de uma requisição através de todos os componentes do seu sistema.

O tracing é a prática de rastrear o caminho completo de uma requisição ou transação à medida que ela se propaga por múltiplos serviços. Ele nos permite visualizar a sequência de operações, o tempo gasto em cada etapa e identificar dependências entre os serviços. Imagine que você enviou um pacote por uma transportadora. Você não quer apenas saber se ele chegou (métrica) ou ver o diário de bordo de cada caminhão (logs). Você quer rastrear o pacote em cada etapa: coleta, centro de distribuição A, centro de distribuição B, entrega. O tracing faz exatamente isso para suas requisições.

Span Inicial

Requisição do usuário

Span Intermediário

Processamento em serviços

Span de Dados

Consulta ao banco

Span Final

Resposta ao usuário

Cada etapa da jornada de uma requisição é chamada de "span", e um conjunto de spans interconectados que representam a requisição completa é chamado de "trace". Ao analisar um trace, você pode ver exatamente qual serviço chamou qual, quanto tempo cada serviço levou para responder e se houve algum erro em alguma etapa específica. Essa capacidade de visualizar o fluxo completo é inestimável para otimizar a performance, depurar problemas de latência e entender o comportamento real do seu sistema distribuído.

Rastreamento de Requisições Ponta a Ponta com AWS X-Ray

Para implementar o tracing de forma eficaz em um ambiente AWS, o AWS X-Ray é a ferramenta ideal. O X-Ray é um serviço que ajuda desenvolvedores a analisar e depurar aplicações distribuídas, como aquelas construídas com microsserviços e funções serverless. Com ele, você pode entender como sua aplicação e seus serviços subjacentes estão performando, e identificar a causa raiz de problemas de performance e erros.

Instrumentação

Habilite o tracing ativo nas funções Lambda ou use o X-Ray SDK

Coleta de Dados

X-Ray coleta informações sobre chamadas entre serviços

Mapa de Serviço

Visualize conexões e latência média de cada componente

Identificação de Gargalos

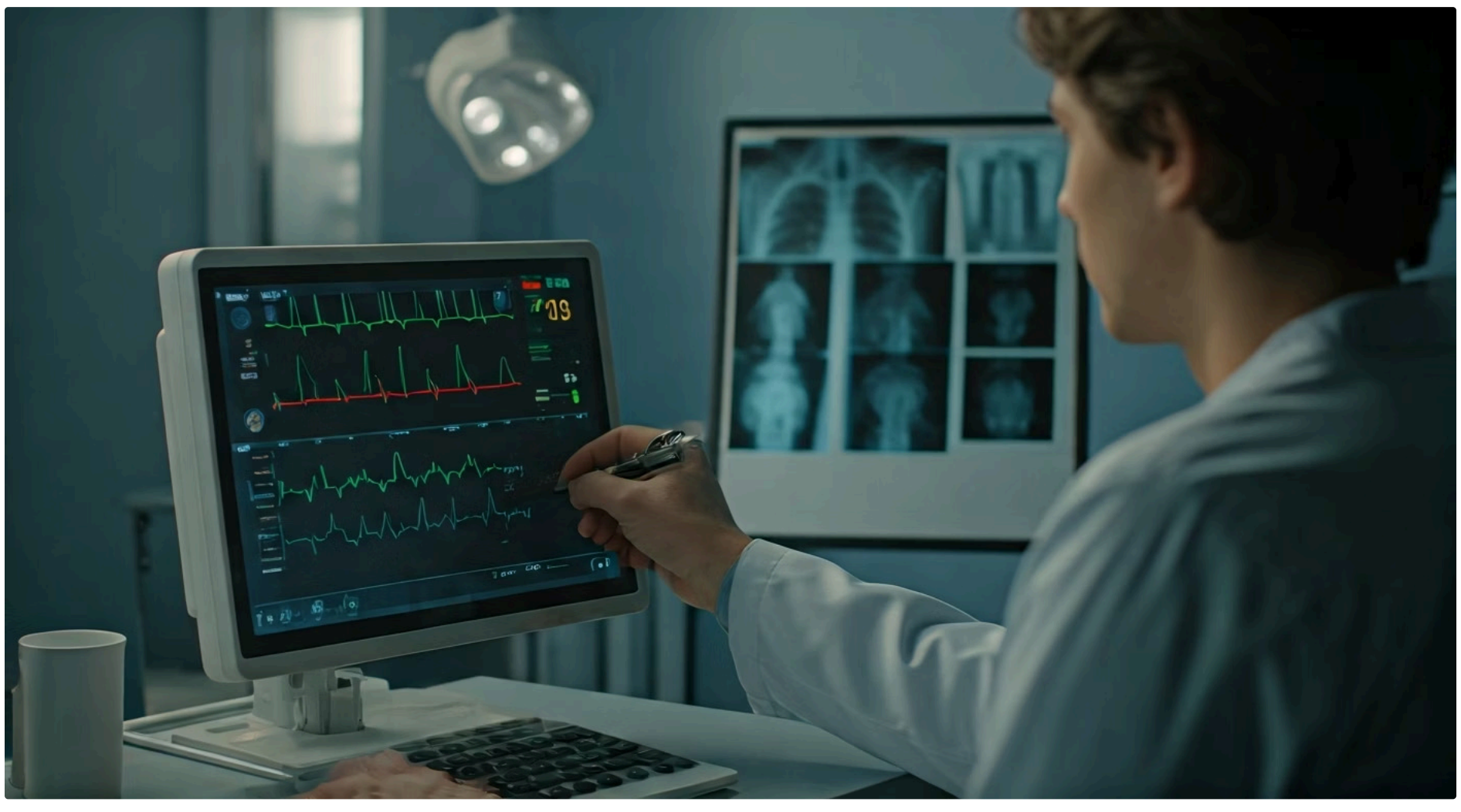
Descubra qual segmento contribui mais para a latência total

O X-Ray funciona coletando dados sobre as requisições que sua aplicação atende. Para isso, você precisa instrumentar seu código ou configurar os serviços AWS para enviar dados de trace. Por exemplo, para funções Lambda, você pode simplesmente habilitar o tracing ativo, e o X-Ray SDK (Software Development Kit) será automaticamente injetado para coletar informações sobre as chamadas de saída da sua função para outros serviços AWS. Ele então visualiza esses dados em um "mapa de serviço", que mostra as conexões entre seus serviços e a latência média de cada um.

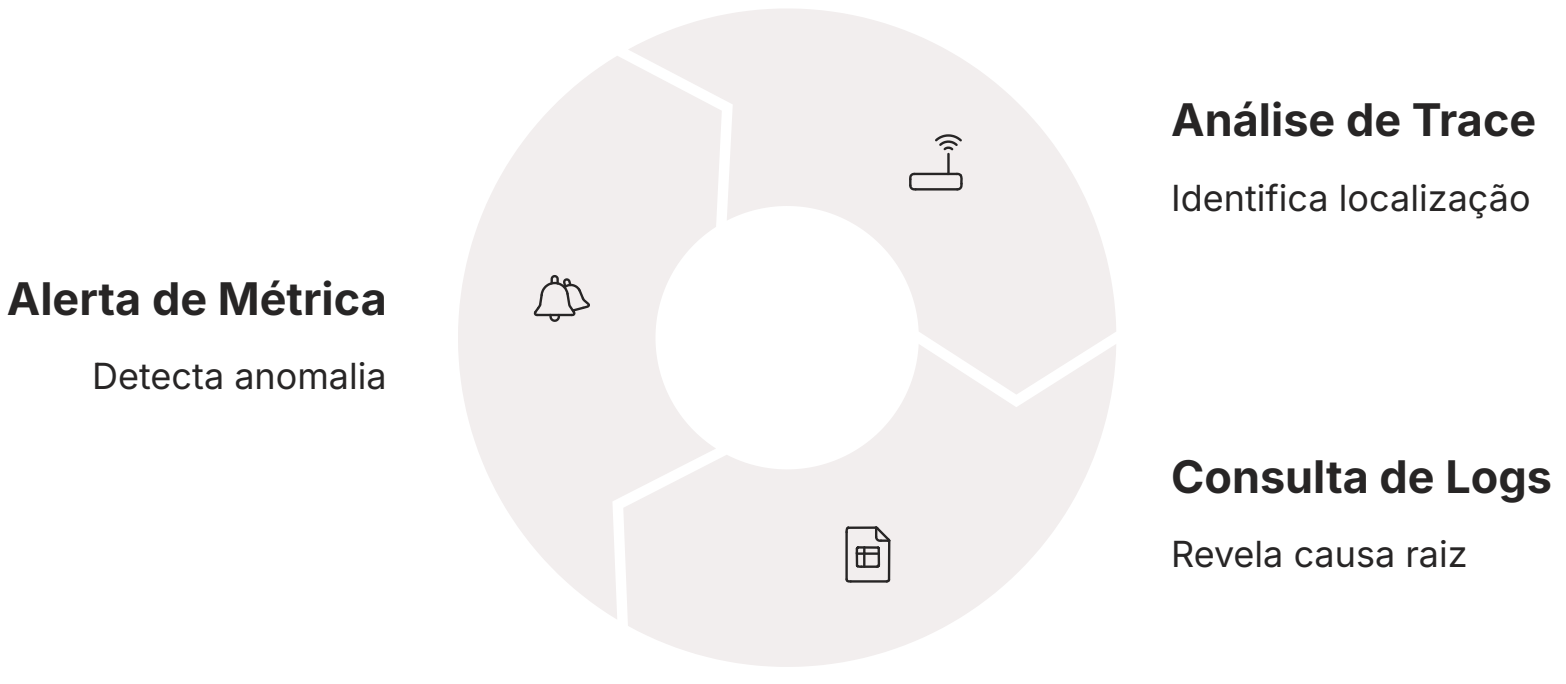
> Considere uma aplicação que usa API Gateway, uma função Lambda e um banco de dados DynamoDB. Quando um usuário faz uma requisição via API Gateway, o X-Ray pode rastrear essa requisição através do Gateway, da função Lambda (incluindo chamadas internas para o DynamoDB) e de volta ao usuário. Se a requisição estiver lenta, o mapa de serviço do X-Ray mostrará qual segmento (span) está contribuindo mais para a latência total, permitindo que você otimize especificamente essa parte do seu sistema. Isso é muito mais eficiente do que tentar adivinhar onde está o problema apenas com logs e métricas isoladas.

Integrando os Pilares da Observabilidade

Até agora, exploramos os logs, métricas e tracing como entidades separadas, cada uma com sua função específica. No entanto, o verdadeiro poder da observabilidade reside na integração e na complementaridade desses três pilares. Eles não são alternativas um ao outro, mas sim peças de um quebra-cabeça que, quando montado, oferece uma imagem completa e tridimensional do estado do seu sistema.



Pense em um médico avaliando a saúde de um paciente. Ele não se baseia apenas no histórico médico (os logs detalhados), nem apenas nos sinais vitais atuais (as métricas), nem apenas em exames específicos como um raio-X (o tracing). Ele usa todas essas informações em conjunto para formar um diagnóstico preciso. Da mesma forma, em um sistema serverless, você pode começar com um alerta de métrica (por exemplo, um pico de erros na função Lambda), mergulhar nos traces para identificar a requisição específica que falhou e, finalmente, consultar os logs detalhados dessa requisição para entender a causa raiz do erro.

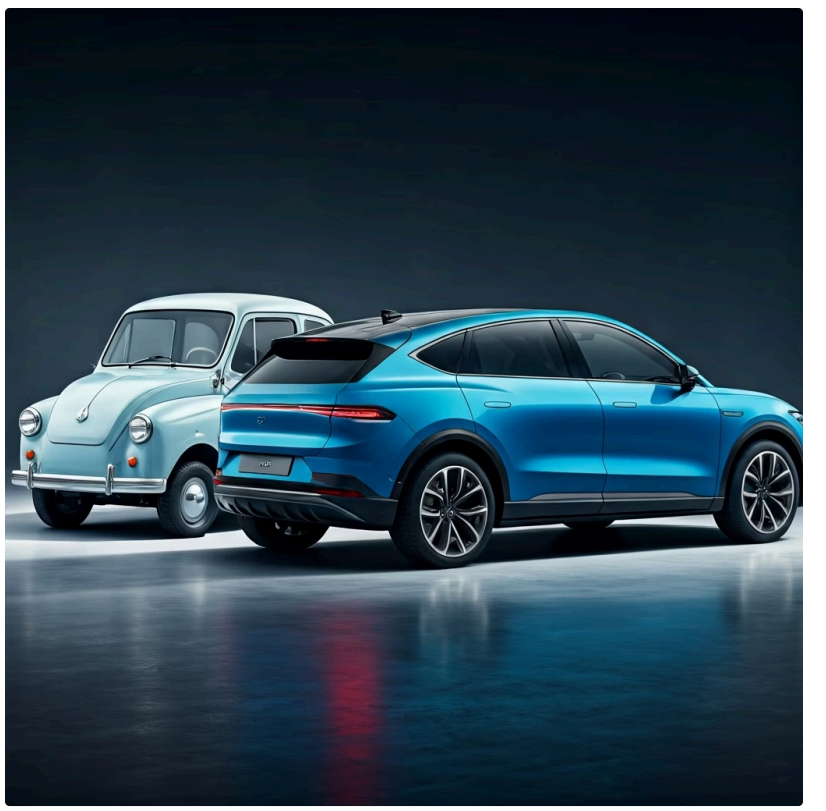


Essa abordagem integrada permite uma depuração mais rápida e eficiente. Um alerta de métrica pode indicar um problema, o trace pode mostrar onde ele ocorreu na cadeia de serviços, e os logs fornecem o "porquê" detalhado. Sem um desses pilares, sua capacidade de entender e resolver problemas seria severamente limitada. A observabilidade eficaz é a arte de combinar essas diferentes perspectivas para obter uma compreensão holística do seu sistema.

Conceito	Âmbito/Aplicação	Base/Origem	Exemplo
Logs	Eventos discretos, detalhes de execução	Mensagens textuais, JSON	Erros, avisos, informações de depuração
Métricas	Agregações numéricas, tendências	Pontos de dados numéricos	Taxa de erros, latência, uso de CPU
Tracing	Fluxo de requisições, dependências	Spans interconectados	Caminho completo de uma requisição entre serviços

Evolução do FaaS e Serverless Containers

O cenário serverless está em constante evolução, e com ele, as demandas por observabilidade também se tornam mais sofisticadas. O Function-as-a-Service (FaaS), como o AWS Lambda, que começou com funções de curta duração e sem estado, está se tornando cada vez mais robusto. Hoje, vemos suporte a tempos de execução mais longos, gerenciamento de estado através de integrações com outros serviços e até mesmo a capacidade de executar funções em ambientes de contêineres.



Essa evolução significa que as funções serverless podem agora lidar com cargas de trabalho mais complexas e de maior duração, o que, por sua vez, exige uma observabilidade mais granular e contínua. Não basta apenas saber se a função falhou; precisamos entender o que aconteceu durante todo o seu ciclo de vida estendido. Além disso, a ascensão dos "Serverless Containers" é uma tendência notável. Tecnologias como AWS Fargate e Google Cloud Run combinam a simplicidade operacional do serverless com a flexibilidade e portabilidade dos contêineres.

Tempos de Execução Estendidos

Funções que rodam por períodos mais longos

Gerenciamento de Estado

Integração com serviços para persistência

Serverless Containers

Flexibilidade de contêineres com simplicidade serverless

> Imagine a evolução dos carros elétricos: de modelos básicos com autonomia limitada, eles se transformaram em SUVs de alta performance com baterias de longa duração e recursos avançados. Da mesma forma, o serverless está amadurecendo, permitindo que desenvolvedores construam aplicações mais poderosas sem se preocupar com a infraestrutura subjacente. No entanto, essa maior capacidade traz consigo a necessidade de ferramentas de observabilidade que possam acompanhar a complexidade crescente, garantindo que a simplicidade do desenvolvimento não se traduza em cegueira operacional.

Infraestrutura como Código (IaC) para Observabilidade

Gerenciar recursos de observabilidade manualmente, especialmente em ambientes serverless que podem ter centenas de funções e microsserviços, é uma receita para o caos. A Infraestrutura como Código (IaC) surge como a solução para esse desafio, permitindo que você defina e provisione seus recursos de infraestrutura – incluindo aqueles relacionados à observabilidade – usando arquivos de configuração versionados, em vez de configurações manuais através de consoles.

1	2	3
Automação	**Repetibilidade**	**Versionamento**
Provisionamento automático de recursos	Configuração consistente em todos os ambientes	Histórico completo de alterações

Ferramentas como Serverless Framework e AWS SAM (Serverless Application Model) são padrões de mercado que permitem descrever sua aplicação serverless e seus recursos de observabilidade adjacentes em um único arquivo de configuração. Por exemplo, você pode definir que sua função Lambda deve ter o tracing do X-Ray habilitado, que seus logs devem ser enviados para um grupo de logs específico no CloudWatch e que um alarme deve ser criado para monitorar a taxa de erros, tudo isso dentro do mesmo arquivo YAML ou JSON.

A grande vantagem do IaC é a automação, repetibilidade e versionamento. Você pode ter certeza de que seus recursos de observabilidade serão configurados de forma consistente em todos os ambientes (desenvolvimento, teste, produção). Além disso, ao versionar seus arquivos IaC, você tem um histórico de todas as alterações, facilitando a auditoria e o rollback se algo der errado. Isso transforma a observabilidade de uma tarefa manual e propensa a erros em um processo automatizado e parte integrante do ciclo de vida de desenvolvimento da sua aplicação.

Melhores Práticas em Observabilidade Serverless

Compreender os pilares da observabilidade é o primeiro passo; aplicá-los de forma eficaz é o segundo. Para extrair o máximo valor dos logs, métricas e tracing em um ambiente serverless, algumas melhores práticas são essenciais. Elas garantem que você não apenas colete dados, mas que esses dados sejam úteis, acionáveis e não se tornem um fardo.

Logging Estruturado

Use formatos como JSON com campos importantes: requestId, userId, transactionId, level (INFO, WARN, ERROR) e message. Isso torna a pesquisa no CloudWatch Logs Insights muito mais eficiente.

Métricas Personalizadas

Crie métricas para eventos de negócio importantes, além das métricas padrão. Por exemplo, "pedidos processados com sucesso" ou "usuários registrados" fornecem insights valiosos sobre o desempenho do negócio.

Amostragem Inteligente

Configure regras de amostragem para rastrear uma porcentagem das requisições ou todas que excedam um limite de latência, evitando custos excessivos com X-Ray.

Dashboards e Alertas

Crie painéis no CloudWatch que consolidem métricas importantes e configure alarmes que notifiquem sua equipe antes que problemas afetem os usuários.

> Construir uma casa com bons alicerces e sistemas de segurança integrados desde o projeto é sempre mais eficaz do que tentar consertar tudo depois que a tempestade chega.

Desafios Comuns e Como Superá-los

Apesar de seus imensos benefícios, a implementação da observabilidade em sistemas serverless não está isenta de desafios. Um dos mais proeminentes é o **custo associado ao volume de dados**. Logs e traces podem gerar uma quantidade enorme de dados, e o armazenamento e processamento desses dados podem se tornar caros se não forem gerenciados de forma eficiente. Outro desafio é a **correlação de eventos** em sistemas altamente distribuídos, onde uma única transação pode gerar logs e métricas em dezenas de serviços diferentes, dificultando a montagem da história completa.

Desafio: Custo de Dados Volume massivo de logs e traces	**Solução: Políticas de Retenção** Defina quanto tempo os logs devem ser armazenados
Solução: Filtragem de Logs Envie apenas logs relevantes ao CloudWatch	**Solução: Amostragem X-Ray** Controle custos sem perder visibilidade crítica

---

Desafio: Correlação Eventos dispersos em múltiplos serviços	**Solução: IDs de Correlação** Gere um ID único no início da requisição
Propagação Passe o ID para todos os serviços downstream	**Filtragem Unificada** Agrupe todos os eventos de uma requisição

Para superar o desafio do custo, implemente **políticas de retenção de logs** no CloudWatch Logs, definindo por quanto tempo os logs devem ser armazenados. Além disso, utilize a **filtragem de logs** para enviar apenas os logs mais relevantes para o CloudWatch, descartando informações excessivamente verbosas que não agregam valor. Para o tracing, como mencionado, a **amostragem inteligente** no X-Ray é crucial para controlar os custos sem perder a visibilidade sobre os problemas críticos.

Em relação à correlação de eventos, a adoção de **IDs de correlação** é uma prática poderosa. Ao iniciar uma requisição, gere um ID único e passe-o para todos os serviços downstream. Inclua esse ID em todos os logs e spans de trace. Isso permite que você filtre e agrupe todos os eventos relacionados a uma única requisição, independentemente de quantos serviços ela atravessou. Lembre-se, a observabilidade é um investimento na resiliência e na eficiência operacional do seu sistema, não apenas um custo. Com as estratégias certas, você pode maximizar seu retorno.

Tendências Futuras em Observabilidade

O campo da observabilidade está em constante evolução, impulsionado pela crescente complexidade das arquiteturas de nuvem e pela necessidade de insights mais profundos e automatizados. Uma das tendências mais significativas é a **AIOps (Inteligência Artificial para Operações)**, que busca aplicar machine learning e inteligência artificial aos dados de observabilidade (logs, métricas, traces) para detectar anomalias, prever problemas e até mesmo sugerir ou executar ações corretivas automaticamente.

AIOps

IA aplicada a logs, métricas e traces para detecção automática de anomalias

Observabilidade como Código

Configuração de monitoramento integrada ao repositório

1 2 3

OpenTelemetry

Padrões abertos para instrumentação agnóstica de plataforma

Outra tendência importante é a ascensão de padrões abertos, como o **OpenTelemetry**. Este projeto visa fornecer um conjunto unificado de APIs, SDKs e ferramentas para instrumentar, gerar, coletar e exportar dados de telemetria (logs, métricas e traces) de forma agnóstica à plataforma. Isso significa que você pode instrumentar sua aplicação uma vez e enviar os dados para diferentes backends de observabilidade, evitando o aprisionamento tecnológico e promovendo a interoperabilidade.

Do Reativo ao Proativo

- Diagnóstico manual de problemas
- Análise de dados históricos
- Resposta após incidentes

Para Preditivo e Inteligente

- Detecção automática de anomalias
- Previsão de problemas futuros
- Ações corretivas automatizadas

> Imagine a transição de um mapa de papel estático para um sistema GPS com inteligência artificial que não apenas mostra o caminho, mas também prevê o trânsito, sugere rotas alternativas em tempo real e até mesmo alerta sobre possíveis problemas mecânicos no seu veículo. É para essa direção que a observabilidade está caminhando: de uma ferramenta reativa de diagnóstico para um sistema proativo e preditivo que garante a saúde e a performance contínua das aplicações. A observabilidade como código, onde a configuração de monitoramento é parte integrante do repositório de código da aplicação, também se tornará cada vez mais comum, garantindo que a observabilidade seja um cidadão de primeira classe no ciclo de desenvolvimento.

Consolidação e Próximos Passos

Nesta aula, mergulhamos no mundo da observabilidade, um pilar indispensável para o sucesso de qualquer aplicação serverless moderna. Vimos como os logs nos fornecem o diário detalhado dos eventos, as métricas nos dão uma visão quantitativa da saúde e performance, e o tracing nos permite seguir a jornada completa de uma requisição através de sistemas distribuídos. Exploramos ferramentas AWS como CloudWatch Logs, CloudWatch Metrics e AWS X-Ray, e discutimos a importância da integração desses pilares, as melhores práticas e as tendências futuras.

Em prática

Comece aplicando logging estruturado em suas funções Lambda. Configure métricas e alarmes básicos no CloudWatch para monitorar erros e latência. Habilite o AWS X-Ray para suas APIs e funções críticas para visualizar o fluxo das requisições. Use IaC para gerenciar esses recursos de observabilidade desde o início do projeto.

Autoavaliação

1

Qual dos pilares da observabilidade é mais adequado para fornecer um histórico detalhado de eventos e mensagens de erro específicas de uma única execução de uma função serverless?

a) Métricas
b) Tracing
c) Logs
d) Alertas

2

Um desenvolvedor percebe que sua função Lambda está com um alto número de invocações rejeitadas devido a limites de concorrência. Qual métrica do Amazon CloudWatch seria a mais relevante para monitorar essa situação?

a) Invocations
b) Errors
c) Duration
d) Throttles

3

Para identificar qual serviço em uma cadeia de microsserviços está causando um gargalo de latência em uma requisição ponta a ponta, qual ferramenta da AWS seria a mais eficaz?

a) Amazon S3
b) AWS X-Ray
c) Amazon DynamoDB
d) AWS IAM

4

Qual das seguintes afirmações melhor descreve a importância da Infraestrutura como Código (IaC) no contexto da observabilidade serverless?

a) IaC elimina a necessidade de logs e métricas
b) IaC permite definir e provisionar recursos de observabilidade de forma automatizada e versionada
c) IaC é usado exclusivamente para gerenciar a segurança
d) IaC é uma ferramenta para visualizar dashboards

Gabarito

1. c) Logs; 2. d) Throttles; 3. b) AWS X-Ray; 4. b) IaC permite definir e provisionar recursos de observabilidade de forma automatizada e versionada.

Questão Dissertativa

> Descreva como os três pilares da observabilidade (Logs, Métricas e Tracing) se complementam para fornecer uma visão holística da saúde e desempenho de uma aplicação serverless.

Próxima Aula

Aula 14: Exploraremos a "Orquestração de Workflows com AWS Step Functions", aprendendo a coordenar múltiplas funções serverless e outros serviços AWS em fluxos de trabalho complexos e resilientes.

Recursos Adicionais

- **Documentação AWS CloudWatch:** Para aprofundar nos detalhes de logs e métricas.
- **Documentação AWS X-Ray:** Para entender a instrumentação e análise de traces.
- **Artigos sobre OpenTelemetry:** Para explorar padrões abertos de observabilidade.

NOTA IMPORTANTE: As informações técnicas desta aula estão atualizadas até 2025. Consulte sempre fontes oficiais da AWS para verificar alterações e as últimas funcionalidades.