STATUS: OPERACIONAL

Menos tempo parado.
Mais controlo.
Zero surpresas.

Como uma plataforma serverless com IA reduz o tempo de resposta a incidentes de horas para minutos, corta custos operacionais e mantém sempre uma pessoa no comando das decisões críticas.

CASE STUDY TÉCNICO · AI OPS SERVERLESS PLATFORM

AWS · Terraform · Python 3.12 · Step Functions · Claude API · GitHub Actions

PATRICIO LUMBE · CLOUD ENGINEER

ÍNDICE

Neste ebook

> **Nota de leitura:** este documento assume zero contexto prévio. Cada secção pode ser lida isoladamente, por isso sinta-se à vontade para ir direto ao capítulo que mais lhe interessa.

01 · O PONTO DE PARTIDA

Quando algo falha às 3 da manhã

Normalmente, alguém tem de notar o alarme, vasculhar logs para perceber o que aconteceu, decidir uma correção e executá-la manualmente. Este projeto automatiza a deteção, o diagnóstico e a explicação, e só espera que uma pessoa diga "sim, avança" antes de tocar em qualquer coisa.

O **AI Ops Serverless Platform** é um sistema de resposta a incidentes construído inteiramente em AWS, que combina monitorização em tempo real com um modelo de IA (Claude, da Anthropic) para ler alarmes, escrever uma explicação em linguagem simples sobre a causa provável e sugerir uma ação de correção, sem nunca decidir sozinho o que mudar na infraestrutura.

> **A garantia central do sistema:** a IA nunca tem as chaves. Ela só pode sugerir ações de uma lista curta e previamente aprovada, e o alvo real da ação vem sempre de uma tabela definida em Terraform, nunca do modelo. Um alarme desconhecido? O sistema regista e espera.

Porque é que isto importa

A resposta manual a incidentes costuma seguir sempre o mesmo padrão: 30 minutos até alguém notar, 60 minutos a investigar logs, 30 minutos a aplicar a correção, o que soma quase 2 horas, normalmente fora do horário de trabalho. Este sistema reduz esse ciclo para minutos, mantendo sempre uma pessoa como decisor final em qualquer passo que altere infraestrutura.

02 · IMPACTO NO NEGÓCIO

O que isto significa para a sua empresa

Por trás da arquitetura técnica está uma pergunta simples: como é que isto poupa tempo, dinheiro e risco a uma empresa real? Aqui estão os números e as razões.

2 a 7 min

tempo médio até à recuperação de um incidente. Antes, o processo manual demorava perto de 2 horas.

$0,60

custo mensal da plataforma em repouso, sem incidentes. Escala apenas com o uso real.

100%

das ações de correção passam por aprovação humana explícita antes de serem executadas.

24 / 7

cobertura de monitorização, sem precisar de uma equipa de plantão dedicada a vigiar alarmes.

Financeiro CUSTO

Cada minuto de indisponibilidade custa dinheiro, seja em vendas perdidas, em produtividade parada ou em reputação junto dos clientes. Reduzir o tempo médio de resposta de quase duas horas para poucos minutos tem impacto direto nessa conta. Ao mesmo tempo, a própria plataforma custa menos de um dólar por mês quando não há incidentes, e escala apenas com o volume real de problemas, sem exigir uma equipa dedicada a vigiar alarmes fora de horas.

Performance e operação VELOCIDADE

A equipa deixa de perder tempo a vasculhar logs manualmente. O sistema entrega a causa provável e uma ação sugerida em menos de um minuto, escrita em linguagem simples que qualquer pessoa da equipa consegue entender, não só quem está de plantão naquele momento. Isto reduz a fadiga causada por excesso de alertas e liberta os engenheiros para trabalho de maior valor.

Segurança CONTROLO

A IA nunca tem acesso direto à infraestrutura. Cada ação possível vem de uma lista definida em código, revista e aprovada pela equipa com antecedência, e nenhuma correção corre sem uma pessoa clicar em aprovar. A velocidade da automação nunca é trocada pelo controlo que uma empresa precisa de manter sobre os seus sistemas críticos.

Disponibilidade e resiliência CONTINUIDADE

A monitorização corre 24 horas por dia, sem depender de alguém estar acordado ou disponível. A arquitetura é serverless, o que elimina um servidor único cuja falha derrubaria todo o sistema. E cada decisão, aprovação e execução fica registada, criando um histórico que serve tanto para auditoria como para melhoria contínua.

> Para uma empresa, isto traduz-se num sistema de resposta a incidentes mais barato de operar, mais rápido a recuperar, mais seguro de confiar e mais simples de auditar, sem exigir uma equipa de operações permanentemente de vigia.

03 · FLUXO OPERACIONAL

Do alarme à recuperação

Um incidente típico percorre sete passos, com uma única pausa deliberada: a aprovação humana.

01 **CloudWatch deteta a anomalia** `~60s`
Um alarme dispara com base em métricas da aplicação (ECS Fargate, Lambda, API Gateway).

02 **EventBridge encaminha o evento**
O alarme é roteado automaticamente para o fluxo de trabalho correto.

03 **Step Functions orquestra a resposta**
Uma máquina de estados coordena cada etapa seguinte, com histórico completo de execução.

04 **Claude analisa o alarme** `~30s`
A função `anomaly_analyser` pede à IA para explicar a causa provável e sugerir uma ação, em modo apenas leitura.

05 **Notificação no Slack e aprovação no Mission Control** `1 a 5 min`
Uma pessoa lê a explicação e aprova ou rejeita a ação sugerida no dashboard.

06 **Remediação executada** `~30s`
Só depois de aprovada, a função `remediator` aplica a ação, sempre a partir da lista permitida em Terraform.

✓ **Serviço recuperado**
Tempo total do alarme à recuperação: **2 a 7 minutos.**

> **Custo zero enquanto espera:** o Step Functions pausa a execução a $0 enquanto aguarda a decisão humana, graças ao padrão `waitForTaskToken`. Não há computação a correr, nem risco a acumular, enquanto ninguém decidiu nada.

— 04 · SOB O CAPÔ

Arquitetura completa

Cinco funções Lambda, cada uma com uma única responsabilidade e permissões mínimas. Nenhuma função acumula mais poder do que precisa.

FUNÇÃO	RESPONSABILIDADE	MODIFICA INFRAESTRUTURA?
`anomaly_analyser`	Lê o alarme, consulta o Claude, valida contra as regras definidas	Não (apenas leitura)
`remediator`	Executa a ação aprovada (por exemplo, reiniciar um serviço)	Sim, apenas ações previamente aprovadas
`runbook_assistant`	Responde a perguntas operacionais lendo documentação Markdown	Não (apenas leitura)
`cost_reporter`	Envia um resumo diário de custos	Não (apenas leitura)
`mission_control`	Dashboard web para aprovações e histórico	Não (apenas interface)

Porque Step Functions, e não uma única função

Uma função Lambda isolada não serve para resposta a incidentes por três razões: funções Lambda têm um limite de 15 minutos, mas humanos precisam de mais tempo para aprovar; se algo falha a meio do processo, não há forma de saber em que passo; e não há como pausar sem estar a pagar computação enquanto se espera.

Step Functions resolve isto

- Pausa a custo zero enquanto espera aprovação humana
- Cada passo é repetido de forma independente se falhar
- Histórico de execução completo, guardado 90 dias
- Desenhado para fluxos longos com pontos de decisão humana

Infraestrutura como código

Quase toda a infraestrutura é gerida por 8 módulos Terraform independentes: IAM, Lambda, Step Functions, API Gateway, EventBridge, ECS, monitorização e Mission Control, todos versionados e reproduzíveis com um único `terraform apply`.

— 05 · GOVERNANÇA DE IA

Seis camadas entre a IA e a infraestrutura

O princípio de desenho é simples: a IA sugere, uma pessoa decide, e o código, nunca o modelo, determina o que pode ser alterado.

1. **A IA nunca escolhe o alvo**
   Os alvos possíveis de qualquer ação vêm sempre de uma tabela definida em Terraform, nunca de texto gerado pelo modelo.

2. **Lista de ações permitidas**
   Só um conjunto curto e previamente aprovado de ações pode ser executado; tudo o resto é recusado por padrão.

3. **Aprovação humana obrigatória**
   Nenhuma ação corre sem um clique explícito de aprovação de uma pessoa no Mission Control.

4. **Validação redundante**
   A função de remediação revalida a lista de ações permitidas de forma independente, sem atalhos herdados de passos anteriores.

5. **Seguro por padrão**
   Um alarme desconhecido ou fora do padrão esperado nunca resulta numa ação, apenas em registo e espera.

6. **Auditoria completa**
   Cada decisão, do alarme à aprovação até à execução, fica registada no CloudWatch e no histórico do Step Functions.

06 · O QUE A PLATAFORMA FAZ

Funcionalidades principais

Análise de incidentes com IA

O Claude lê alarmes e explica a causa provável em linguagem simples, sem jargão técnico.

Aprovação humana obrigatória

Nada é executado até uma pessoa clicar em aprovar no dashboard.

Remediação determinística

Ações vêm sempre de uma lista definida em Terraform, nunca inventadas pelo modelo.

Espera a custo zero

O Step Functions pausa sem custo enquanto aguarda a decisão humana.

Auditoria completa

Todas as decisões ficam registadas no CloudWatch e no histórico do Step Functions.

Serverless e económico

Cerca de $0,60 por mês em repouso, escalando apenas com o volume real de incidentes.

Assistente de runbooks (RAG)

Responde a perguntas operacionais lendo a documentação Markdown da própria equipa.

Dashboard Mission Control

Interface web para aprovações, histórico de incidentes e perguntas e respostas.

07 · FERRAMENTAS UTILIZADAS

Stack tecnológico

CAMADA	TECNOLOGIA
Computação	AWS Lambda, ECS Fargate, API Gateway
Orquestração	Step Functions, EventBridge
Monitorização	CloudWatch (Logs, Métricas, Alarmes)
Inteligência Artificial	Amazon Bedrock (Claude Sonnet 4.5)
Infraestrutura	Terraform, 8 módulos
Armazenamento	S3, Secrets Manager

Linguagem	Python 3.12
CI/CD	GitHub Actions
Formato de runbooks	Markdown com recuperação RAG

Pipeline de integração contínua

Cada push para `main` ou `develop` corre automaticamente: testes unitários em Python (pytest), validação de formato Terraform, build e push da imagem Docker para o ECR, aplicação do Terraform e deployment forçado no ECS.

— 08 · PROBLEMAS REAIS, SOLUÇÕES REAIS

Desafios de engenharia resolvidos

DESAFIO	SOLUÇÃO
Esperar por uma pessoa sem pagar computação	`waitForTaskToken`
Manter a remediação por IA segura	Lista determinística definida em Terraform; o modelo nunca nomeia alvos
Evitar alterações acidentais na infraestrutura	A Lambda revalida a lista de ações permitidas de forma independente
Cobrir alarmes 24 horas por dia sem equipa de plantão	Deteção e análise totalmente automatizadas; a pessoa só aprova ou rejeita
Explicar incidentes a quem não é engenheiro	Resumos em linguagem simples gerados pelo Claude, publicados no Slack
Partilhar conhecimento operacional	RAG sobre os runbooks Markdown da equipa, guardados no S3

— 09 · EFICIÊNCIA

Custos e eficiência

Arquitetura serverless significa pagar apenas pelo que é realmente usado.

CENÁRIO	CUSTO MENSAL
Repouso (sem tráfego, sem alarmes)	~$0,60
Carga de exemplo (10 incidentes por dia)	~$3,00
Com a app de demonstração a correr 24 horas por dia	~$9,60

A aplicação de demonstração (ECS Fargate) existe apenas para mostrar o sistema a funcionar ao vivo. Removê-la baixa o custo para menos de $1 por mês.

— 10 · UM INCIDENTE REAL, DO INÍCIO AO FIM

Plataforma em ação

Para tornar tudo isto concreto, aqui está uma execução real do ambiente de demonstração, do serviço saudável até ao incidente resolvido com aprovação humana.

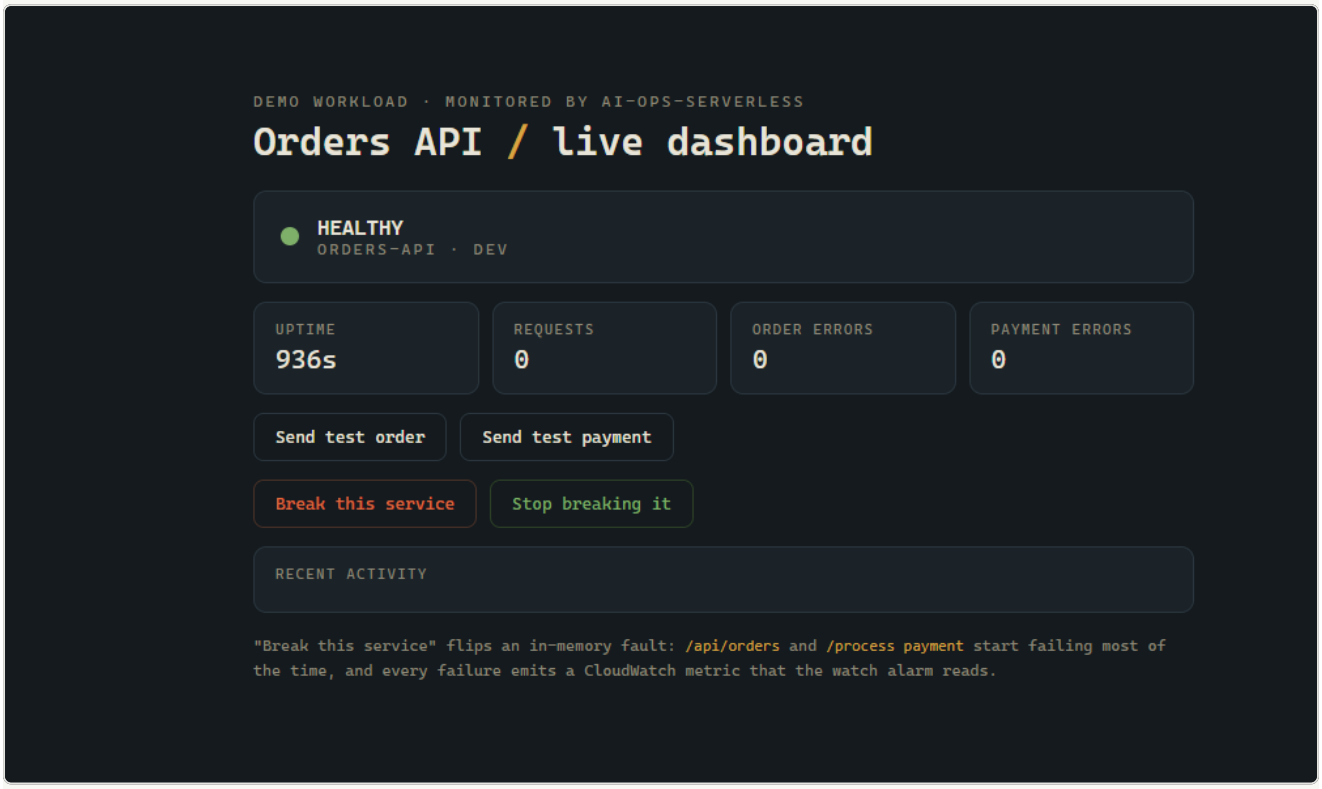

FIG. 1 Orders API saudável, monitorizada continuamente pela plataforma. É este o estado normal que o CloudWatch observa a cada momento.

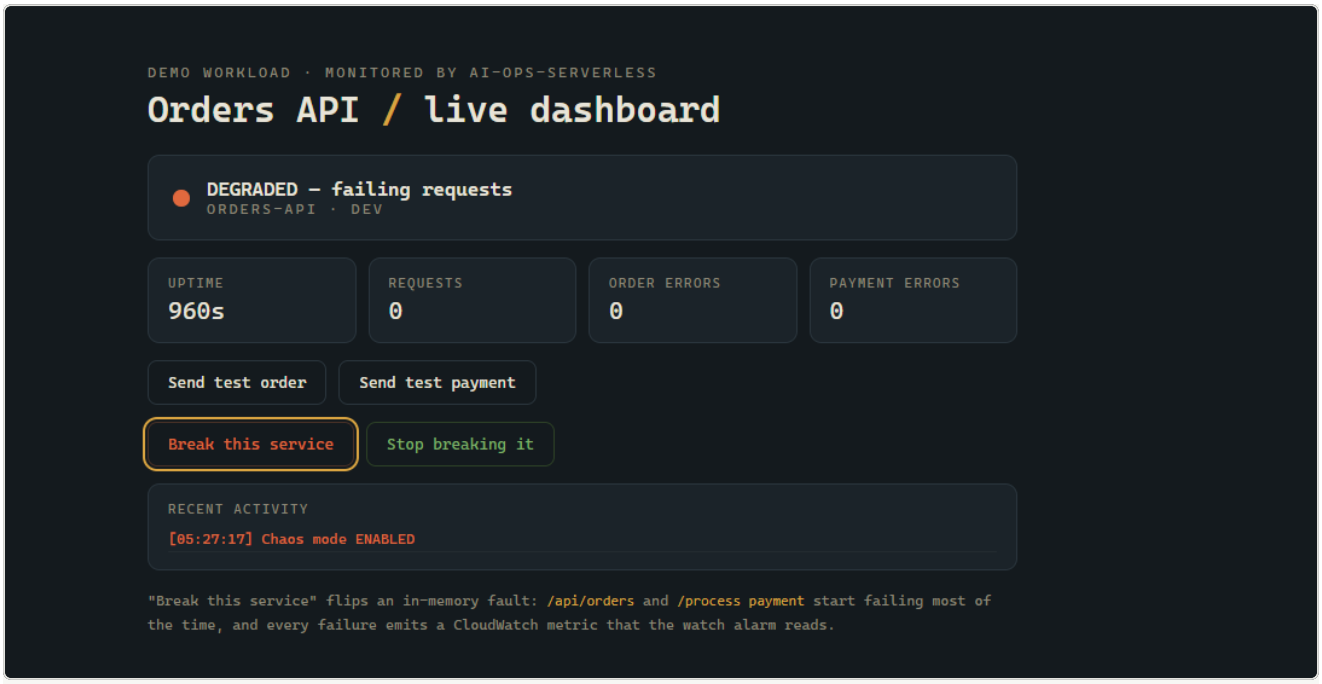

FIG. 2 O serviço começa a falhar e os erros acumulam-se acima do limite aceitável. Este é o gatilho que dispara o alarme no CloudWatch.

Em menos de um minuto, o alarme chega ao Step Functions, que aciona a análise automática. O Claude lê os dados do alarme e escreve um resumo claro do que está a acontecer, sem exigir que ninguém abra logs manualmente.

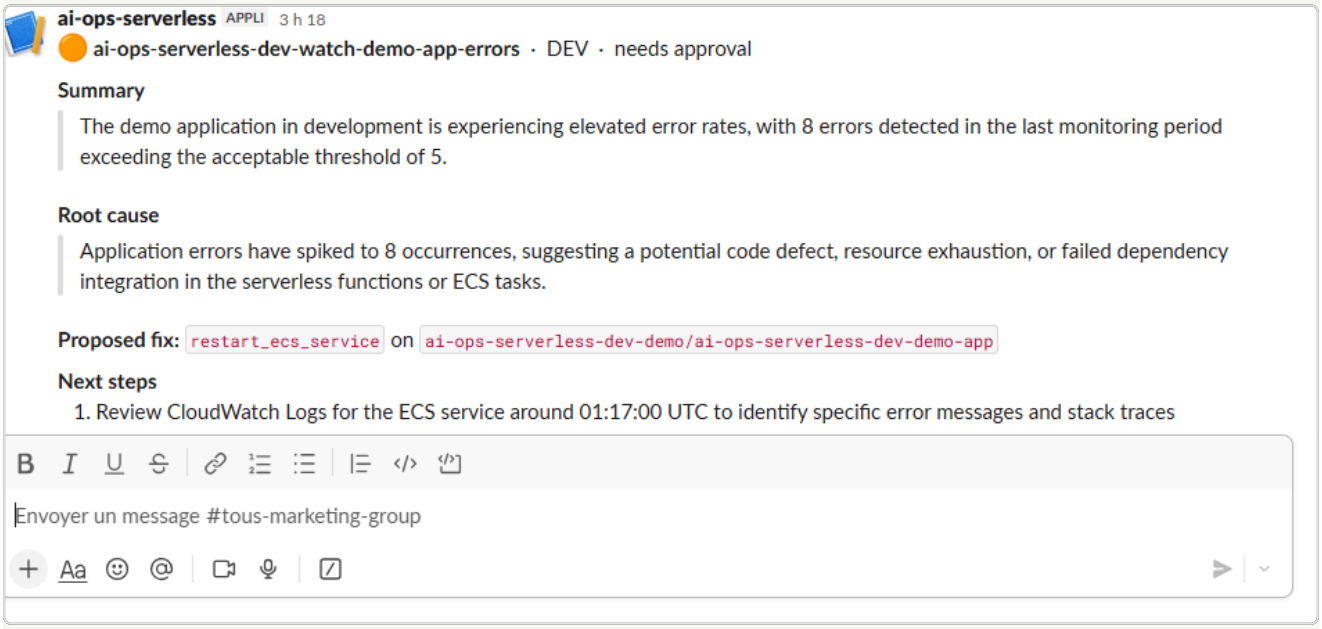

FIG. 3 A explicação chega diretamente ao Slack: o resumo do problema, a causa provável e a ação proposta, prontos para revisão pela equipa.

A ação sugerida não é executada de imediato. Fica à espera no Mission Control até uma pessoa decidir. Note que a plataforma indica explicitamente que o alvo da ação foi resolvido pelo sistema, e não escolhido pela IA.

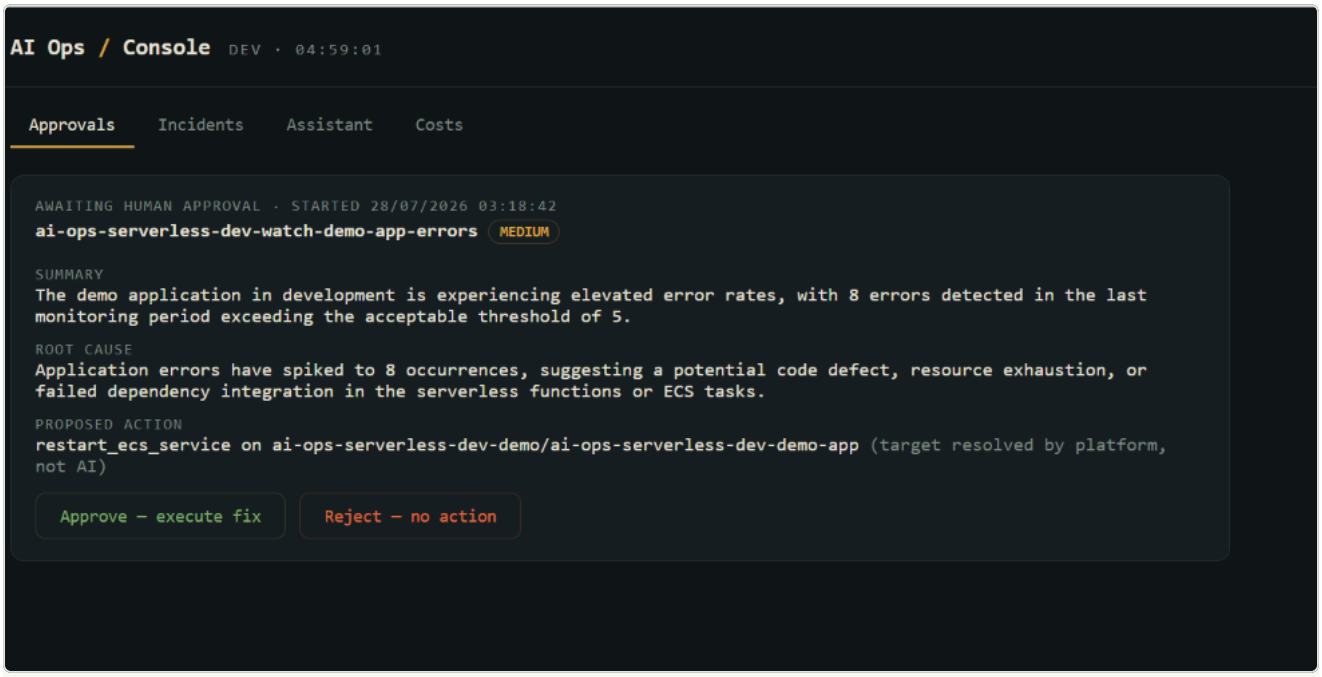

FIG. 4 O dashboard Mission Control mostra o incidente à espera de aprovação, com dois botões claros: aprovar e corrigir, ou rejeitar e não fazer nada.

Assim que a correção é aprovada, tudo fica registado automaticamente, com identificadores de execução, hora de início e hora de conclusão.

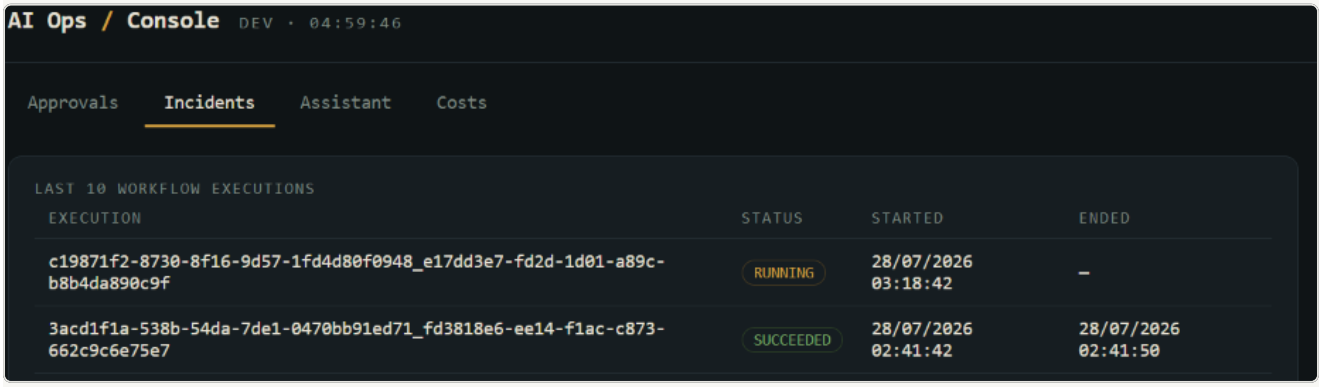

FIG. 5 Histórico das últimas execuções do fluxo, com estado, hora de início e hora de fim de cada incidente processado.

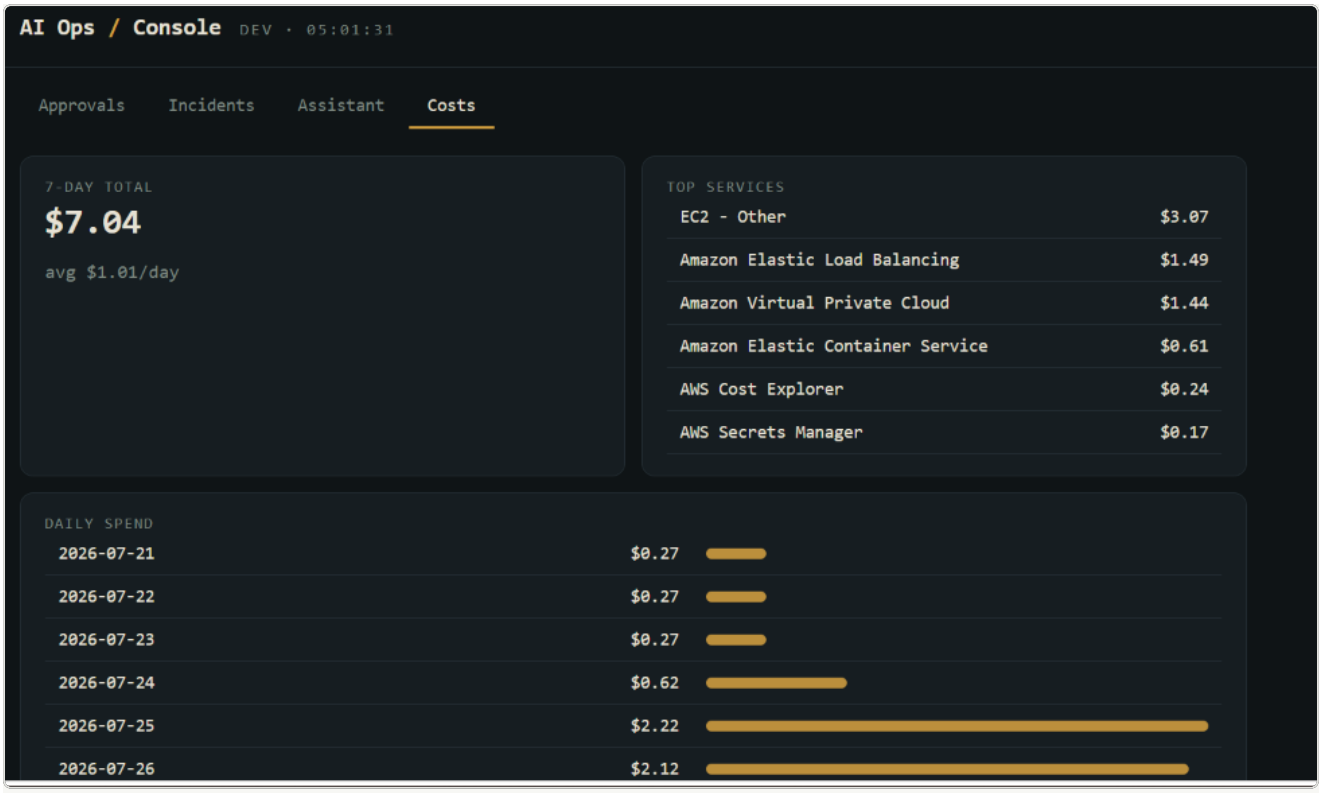

FIG. 6 O separador de custos mostra o gasto real dos últimos 7 dias e os serviços AWS que mais pesam na fatura, com total transparência.

Para além da resposta a incidentes, a plataforma também responde a perguntas operacionais do dia a dia, lendo a documentação da própria equipa.

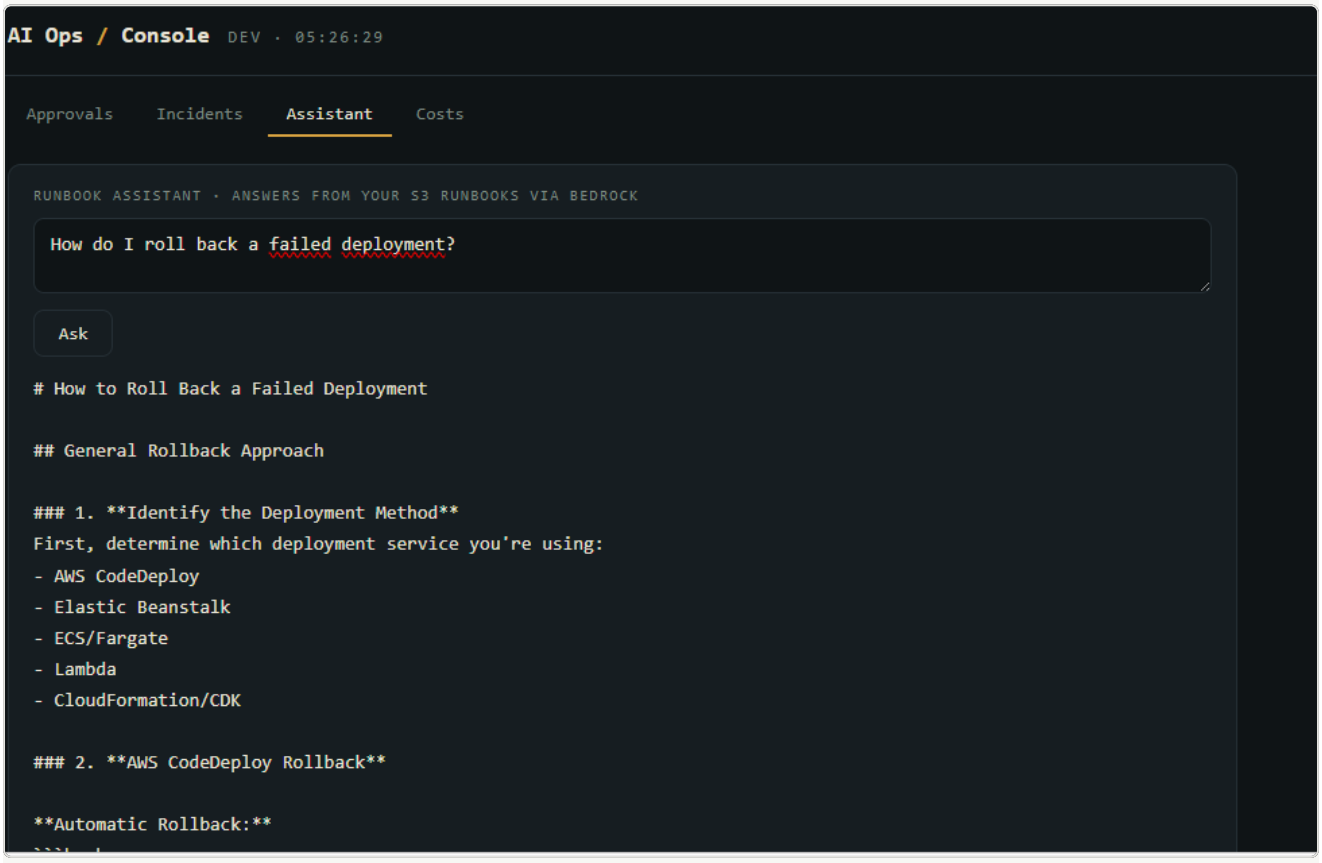

FIG. 7 O assistente de runbooks responde a uma pergunta real sobre como reverter uma implementação com falhas, com base nos documentos guardados pela equipa no S3.

— 11 · PARA RECRUTADORES E EQUIPAS TÉCNICAS

Competências demonstradas

Este projeto não é um tutorial seguido passo a passo. É uma decisão de arquitetura atrás da outra, documentada e justificada.

ARQUITETURA E DESENHO

- Arquitetura orientada a eventos
- Sistemas de IA com supervisão humana
- Padrões serverless
- Otimização de custos
- Infraestrutura como código (8 módulos)

AWS E CLOUD

- Lambda
- Step Functions
- EventBridge
- API Gateway
- ECS Fargate
- ECR
- CloudWatch
- IAM com privilégio mínimo
- Secrets Manager
- S3
- VPC e Networking

IA E MACHINE LEARNING

- Integração de API LLM (Claude)
- Prompt engineering
- RAG (Retrieval Augmented Generation)
- Guardrails de segurança para IA

ENGENHARIA E OPERAÇÕES

- Automação de resposta a incidentes
- Trilhas de auditoria e conformidade
- Monitorização em tempo real
- CI/CD com GitHub Actions
- Testes automatizados (pytest)

— 12 · O QUE VEM A SEGUIR

Roadmap

FASE	FOCO
Agora	Completar cobertura Terraform, expandir catálogo de runbooks

Q3 2026	Suporte a várias regiões, análise de custos avançada
Q4 2026	Remediação mais ampla (SSM Automation), suporte a EKS
2027	Framework de ações personalizadas, aprovações baseadas em políticas

Estado atual

A funcionar de ponta a ponta

- App de demonstração (ECS Fargate) com alarmes reais
- Fluxo CloudWatch, EventBridge e Step Functions
- Análise via Amazon Bedrock (Sonnet 4.5)
- Aprovação humana no Mission Control
- Notificações no Slack
- Pipeline CI/CD com GitHub Actions

Próximos passos

- Logging estruturado adicional
- Permissões do Mission Control em Terraform
- Suporte a várias regiões
- Ações de remediação mais amplas

13 · SOBRE O AUTOR

Patricio Lumbe

Cloud Engineer, especializado em AWS e Infraestrutura como Código. Este projeto demonstra padrões reais de resposta a incidentes em produção: arquitetura orientada a eventos, integração segura de IA, Infraestrutura como Código e governança humana sobre sistemas automatizados.

contact@patriciolumbe.com

Todo o código é versionado, testado e implementado via CI/CD.