Comando de Resiliência - Operação Helius

FASE 1: DIAGNÓSTICO SISTÊMICO

Contexto imediato (Dia 0 — Segunda, 07h45):

Durante a madrugada, houve uma sequência de falhas em cascata na plataforma global da Helius.

- Vários clientes relatam **recomendações erráticas** e **dashboards desatualizados**.
- O sistema de **observabilidade** parou de enviar métricas.
- Alguns modelos de IA começaram a **retornar decisões incoerentes**.
- Há suspeita de **drift de dados** e **corrupção no grafo de dependências**.

Informações recebidas:

- **Ambiente:** cluster híbrido (AWS + edge nodes).
- **Serviços afetados:** `recommendation-engine`, `telemetry-gateway`, `mlflow-tracker`, `grafana-core`, `llm-router`.
- **Logs recentes mostram:**
    - picos de latência em `telemetry-gateway`
    - ausência de métricas de 3 regiões (us-east-1, eu-central-1, sa-east-1)
    - duplicação de mensagens MQTT em edge devices
    - spikes de custo na AWS (possível autoscaling descontrolado)
- **Neo4j grafo parcial** sugere alta centralidade de `recommendation-engine` e `grafana-core`.

Abordagem Inicial de Diagnóstico Sistêmico

O diagnóstico será dividido em três etapas interligadas:

1. Métricas e Métodos Utilizados (A Análise de Grafos em Primeiro Lugar)

Dado que o sistema de observabilidade (*Grafana/Prometheus*) está parcial ou totalmente comprometido (grafana-core e telemetry-gateway afetados), a análise de grafos se torna o método mais robusto para entender a **estrutura** da falha, enquanto os logs provêm o **timing** dos eventos.

Domínio	**Métrica/Método**	**Objetivo no Diagnóstico**	**Ação Imediata (Ferramenta)**
Topologia	**Centralidade de Intermediação** (*Betweenness Centrality*)	Identificar os nós (serviços) que atuam como pontes críticas de comunicação (os *single points of failure*). Uma falha neles garante a propagação. O recommendation-engine e grafana-core já são suspeitos de alta centralidade.	**Rodar sim/graph_analysis.ipynb** (Após reconstruir o grafo).
Topologia	**Centralidade de Grau** (*Degree Centrality*)	Quantificar as conexões diretas. Um serviço com alto grau de saída pode ser a origem do *backpressure*, e um com alto grau de entrada pode ser a vítima do excesso de requisições.	**Rodar sim/graph_analysis.ipynb** (Após reconstruir o grafo).
Topologia	**Modularidade** (*Modularity*)	Avaliar se a falha está confinada a um cluster de serviços (baixo risco de cascata global) ou se ela atravessou fronteiras de módulos (alto risco, falha sistêmica).	**Rodar sim/graph_analysis.ipynb**

Série Temporal	**Latência P95/P99**	Isolar o momento e a magnitude do pico inicial em telemetry-gateway.	Consultar **Logs** (/data/logs.jsonl) e **Prometheus/Grafana** (templates /observability/) para o *dataset* histórico.
Série Temporal	**Taxa de Incoerência de Decisão** (*Drift* de Modelo)	Quantificar o impacto funcional nos modelos de IA (llm-router).	Consultar logs de llm-router e dados do mlflow-tracker para anomalias.

A. Diagnóstico Estrutural Cruzado (Betweenness Centrality, Degree Centrality, Modularity)

Analisando os resultados da Centralidade e Modularidade:

1. Modularidade e Falha em Cascata

- **Modularidade Global (Louvain): 0.3267**
- **Interpretação:** Este valor de modularidade é considerado **baixo** para uma arquitetura de microsserviços bem compartimentada. A modularidade baixa, combinada com o alto número de comunidades (16), sugere que:
    - As comunidades (clusters de serviços) estão **fracamente coesas** internamente e o acoplamento entre elas é relativamente alto.
    - Isto **confirma a suspeita de que a arquitetura é suscetível a falhas em cascata**, pois um erro em um serviço "ponte" tem um caminho fácil para se propagar por todo o sistema (alto acoplamento estrutural).

2. Identificação da Raiz e Propagação (Pontes Críticas)

Agora, cruzamos os serviços afetados (recommendation-engine, telemetry-gateway, grafana-core, llm-router) com as pontes críticas (Top 5 Betweenness Centrality):

Posição	Serviço	Betweenness Centrality	In-Degree (Vulnerabilidade)	Out-Degree (Impacto)	Status Conhecido	Categoria na Arquitetura

0	**service_00**	**0.1059**	6	**18**	**Candidato a telemetry-gateway ou grafana-core**	Maior Ponte Crítica e Maior Impacto
1	**service_10**	0.0813	**11**	6	**Candidato a recommendation-engine ou llm-router**	Maior Vulnerabilidade (In-Degree)
2	service_07	0.0610	5	5	Não-afetado ou Falha Secundária	Alto Risco
3	service_06	0.0319	7	5	Não-afetado ou Falha Secundária	Risco Moderado
4	service_03	0.0319	6	5	Não-afetado ou Falha Secundária	Risco Moderado

Análise Focada:

1. **service_00 (Betweenness 0.1059, Out-Degree 18):**
    - Este é o **nó mais crítico (Ponte)**. Sua falha afeta o maior número de caminhos.
    - O **Out-Degree 18** é extremamente alto, indicando que ele faz chamadas ou tem dependências em 18 outros componentes.
    - **Hipótese de Identificação:** É provável que service_00 seja o **telemetry-gateway ou o grafana-core**, especialmente se um deles é responsável por monitorar (o que explica as 14 arestas monitors nos dados) e fornecer dados para muitos outros serviços (o que explica o alto Out-Degree). A falha neste serviço explica a **cegueira do sistema de observabilidade**.
2. **service_10 (In-Degree 11, Betweenness 0.0813):**
    - Este é o serviço mais **VULNERÁVEL** a ser sobrecarregado (In-Degree 11).
    - **Hipótese de Identificação:** É altamente provável que service_10 seja o **recommendation-engine** ou o **llm-router**. Se for o *engine*, a alta demanda de entrada (In-Degree) e a falha de um serviço de dependência upstream

(como o telemetry-gateway/service_00) explicam as **recomendações erráticas** e o **drift de dados** devido à falta de inputs corretos.

3. Conclusão do Diagnóstico e Rastreamento da Cascata

A análise estrutural confirma a falha em cascata e fornece a ordem provável de impacto:

1. **Ponto de Início/Vulnerabilidade:** A **Duplicação de Mensagens MQTT em *edge devices*** (contexto) sobrecarregou o serviço de ingestão/telemetria.
2. **Nó Crítico Inicial (Cegueira Sistêmica):** O **service_00** (provavelmente o telemetry-gateway ou o grafana-core conforme as métricas), sendo a ponte mais crítica, falhou devido ao pico de latência/sobrecarga, causando a **perda de métricas em 3 regiões**.
3. **Propagação Funcional (Impacto no Negócio):** A falha no service_00 e a perda de dados regionais afetou o **service_10** (provavelmente o recommendation-engine ou llm-router), um nó de alta demanda (In-Degree 11), resultando em **recomendações erráticas** e **decisões incoerentes de IA**.
4. **Loop de Feedback (Infraestrutura):** A perda de métricas regionais leva o *Autoscaler* a agir cegamente em resposta a outras métricas de *backpressure* (possivelmente as altas taxas de erro do service_10), causando o **autoscaling descontrolado** e os *spikes* de custo na AWS.

B. Diagnóstico Estrutural (Logs Grafana/Prometheus)

1. Ponto de Origem e Validação Cronológica (Logs)

A falha primária não é um evento isolado, mas uma **combinação de sobrecarga/corrupção de dados na ingestão e armazenamento regional**, confirmada pela cronologia dos logs (logs.jsonl):

Cronologia	Evento Chave	Serviço	Nível	Implicação
11:24:53Z	Outage detected in region	recommender-service (service_10)	CRITICAL	**Falha Funcional Imediata.** O serviço mais vulnerável (In-Degree 11) colapsa logo no início (recomendações erráticas).

11:27:23Z	Database connection failed	monitoring-service (service_00)	ERROR	**Cegueira Imediata.** A principal ponte crítica (Betweenness 0.1059) do sistema falha em conectar-se, explicando a perda de métricas (grafana-core afetado).
11:30:37Z	Outage detected in region	storage-service	CRITICAL	**Falha Regional do Data Layer.** Confirma que a falha começou com infraestrutura regional e dados.
11:33:08Z	Outage detected in region	ingest-service (telemetry-gateway)	CRITICAL	**Causa Raiz do Fluxo de Dados.** Confirma que a falha regional atingiu o serviço de ingestão, que processava as mensagens MQTT duplicadas, resultando em:
11:37:26Z	Data corruption detected	ingest-service (telemetry-gateway)	CRITICAL	O ponto mais crítico: a sobrecarga na ingestão não apenas causou lentidão e queda, mas também **corrompeu os dados** em trânsito.

2. Confirmação da Cascata e Cegueira

- **Estrutural (Grafo):** A **Modularidade de 0.3267** é baixa, confirmando que a arquitetura está altamente acoplada e que a falha inicial (regional) pôde se propagar facilmente do ingest-service (telemetry) para o recommender-service (negócio) e monitoring-service (observabilidade) em menos de 10 minutos.
- **Observabilidade (Alertas):** A falha no monitoring-service significa que o alerta MissingLogs deve ter sido disparado, validando a **cegueira sistêmica**. A falha no recommender-service significa que o alerta HighInvalidModelOutputs (para > 5% de respostas inválidas) estaria disparado se a telemetria estivesse funcionando.

A falha regional de conectividade/infraestrutura foi amplificada pelo ingest-service e sua consequente corrupção de dados, atingindo o coração do negócio (recommender-service) e do controle (monitoring-service) devido à estrutura crítica de pontes.

Mecanismo da Falha: Sobrecarga, Instabilidade e Corrupção

O mecanismo é um ciclo vicioso de falha de I/O (entrada/saída) amplificado pela sobrecarga, que resultou na ingestão de dados inválidos.

1. Causa Imediata: Duplicação e Sobrecarga (Backpressure)

Evidência	Serviço Chave	Efeito (Cronologia)
Duplicação de mensagens MQTT (Contexto)	ingest-service (telemetry-gateway)	A duplicação de mensagens, confirmada no contexto, gera um súbito **pico de latência** e sobrecarga. Isso é validado por High memory usage e Slow response detected em vários serviços nos logs (ex: billing-service, user-service). O ingest-service não consegue lidar com o volume.
Outage detected in region (11:33:08Z)	ingest-service	O sobrepeso da ingestão, combinado com a instabilidade regional, faz o serviço colapsar temporariamente.

2. Gatilho para Corrupção (A Amplificação)

A sobrecarga e o colapso do ingest-service levam diretamente à corrupção:

Evidência	Serviço Chave	Efeito (Cronologia)
Data corruption detected (11:37:26Z)	ingest-service	O serviço de ingestão, sob pressão extrema (latência alta/memória cheia), **não consegue mais garantir a integridade** das mensagens que está lendo ou transformando, registrando a corrupção.
Data corruption detected (11:37:51Z)	recommender-service	O ingest-service envia esses dados corrompidos para os *data stores* e, consequentemente, para o recommender-service (que é um nó de Alta Vulnerabilidade com In-Degree 11). Isso causa o drift e as **recomendações erráticas** reportadas pelos clientes.

A combinação é: **Sobrecarga de volume (duplicação) + Incapacidade de processamento do ingest-service = Escrita de Dados Corrompidos no *Storage* e *Pipeline* ML.**

Causa Raiz Mais Provável

A causa mais provável que une a falha regional, o colapso do *data store* e a duplicação de mensagens é uma **Instabilidade/Particionamento Transitório da Camada de Rede (Control Plane) nas Regiões da AWS**.

Hipótese Detalhada (A *Root Cause*)

1. **Instabilidade Regional (SPOF na Infraestrutura):** Um evento de instabilidade de rede ou uma falha de conectividade transiente afetou as regiões us-east-1, eu-central-1 e sa-east-1.
    - **Evidência:** Os logs reportam Outage detected in region em múltiplos serviços e *data stores* quase simultaneamente (11:24:53Z a 11:33:08Z), bem como a perda de métricas dessas regiões (Contexto).
2. **Duplicação de Mensagens (Mecanismo de Resiliência Inverso):**
    - O problema de conectividade fez com que o *Edge Broker* (onde os *edge devices* enviam MQTT) ou o *Gateway* de Ingestão perdessem os *acknowledgements* (ACKs) ou a sessão de conexão com os *edge devices*.
    - O protocolo MQTT, ou a lógica de resiliência nos *edge devices*, interpretou essa perda de ACK como uma falha de entrega e, para não perder dados, **agressivamente retransmitiu as mensagens**, gerando a duplicação e o pico de tráfego inicial.
3. **Cegueira e Custo Descontrolado:**
    - A instabilidade também atingiu o banco de dados do monitoring-service (Database connection failed em 11:27:23Z) e o serviço em si (service_00, a ponte crítica).
    - A perda de métricas regionais deixou o *Autoscaler* da AWS "cego" (Contexto). Ao detectar o aumento de latência e erros (causado pela duplicação e corrupção), ele tentou compensar cegamente em outras regiões não afetadas, levando ao **autoscaling descontrolado e spikes de custo**.

Ação Imediata: Contenção e Restauração da Visibilidade

Neste ponto, a mitigação de danos e a recuperação da observabilidade têm prioridade máxima sobre a investigação aprofundada da *root cause* (que já está bem delimitada).

Ação de Contenção de Dano e Custo (Prioridade 1):

Alvo	Ação Imediata	Racional

ingest-service (telemetry-gateway)	**Isolamento Regional:** Iniciar o *drain* de tráfego das regiões us-east-1, eu-central-1 e sa-east-1 (se possível, por configuração de *load balancer* ou *feature flag*). **Limitar o tráfego MQTT nos *edge devices*** para cessar a duplicação e a sobrecarga.	Parar imediatamente a ingestão de dados corrompidos (Data corruption detected em 11:37:26Z) e controlar o *backpressure* e o *autoscaling* descontrolado (spikes de custo na AWS).
recommender-service (service_10)	**Fail-Open Temporário:** Mudar o recommender-service para um modo de *fallback* que retorne recomendações estáticas ou nulas, evitando decisões incoerentes e erráticas para o cliente até que o *data stream* seja limpo.	Minimizar o impacto no negócio e a experiência do cliente, que está recebendo recomendações erráticas.

Ação de Recuperação da Observabilidade (Prioridade 2):

Alvo	Ação Imediata	Racional
monitoring-service (service_00)	**Reinício/Restauração:** Forçar o reinício do monitoring-service e grafana-core para restabelecer a conectividade com o banco de dados (Database connection failed em 11:27:23Z) e restaurar a visibilidade.	Sem observabilidade, qualquer outra ação é um tiro no escuro. Recuperar a visibilidade (Prometheus/Grafana) é crucial para auditar as ações de contenção.