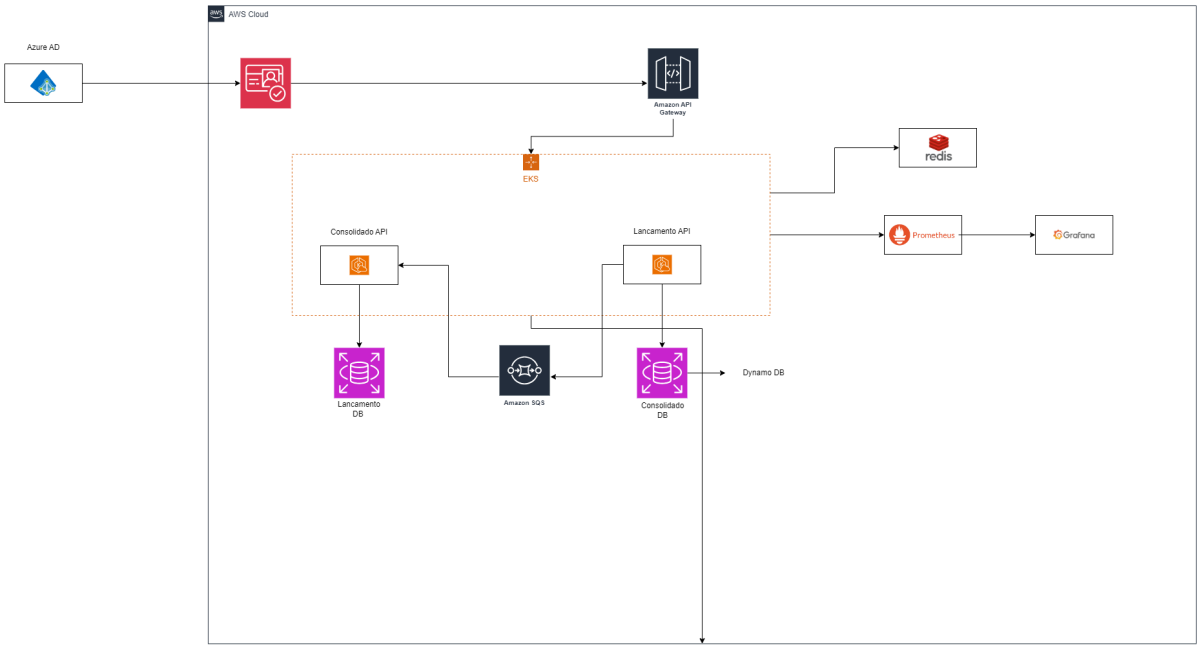
Introdução:

Para atender aos requisitos do desafio de desenvolver uma arquitetura de software escalável e resiliente para controlar lançamentos de débito e crédito e gerar o consolidado diário de um fluxo de caixa, propus uma solução baseada em microserviços. Essa abordagem garante a alta disponibilidade, escalabilidade e resiliência da aplicação, mesmo sob picos de carga e falhas de componentes. Utilizando as melhores práticas de arquitetura, como CQRS (Command Query Responsibility Segregation), Event-Driven Architecture e padrões de resiliência como Circuit Breaker e Idempotência, essa solução assegura que o sistema possa processar 50 requisições por segundo para o consolidado diário com alta performance e baixo custo operacional.

Diagrama da solução



Estrutura da Solução:

1. Arquitetura baseada em Micro serviços
    - **Dois micro serviços principais:** um para o controle de lançamentos de débitos e créditos e outro para a consolidação diária dos lançamentos.
    - Um worker para processar os lançamentos que não puderam ser processados devido há alguma falha na api de consolidação
    - **Responsabilidade única**: Cada micro serviço tem uma responsabilidade clara e um banco de dados próprio, garantindo isolamento de falhas.

2. Padrões Arquiteturais Utilizados

- **CQRS (Command Query Responsibility Segregation)**: Garantindo a separação das operações de leitura e escrita para otimizar o desempenho.
- **Dependency Injection (DI)**: Facilitando a manutenção e o desacoplamento do código.
- **Repository Pattern**: Abstração da persistência de dados para desacoplar a lógica de negócios da infraestrutura de dados.
- **Event-Driven Architecture**: Facilitando a comunicação assíncrona entre micro serviços, publicando eventos de lançamento e consolidando-os.
- **Cache-aside Pattern**: Garantindo a redução de redundância e otimização de consultas verificando se uma mensagem já foi processada.

3. Resiliência e Comunicação

- **Circuit Breaker**: Protegendo o sistema contra falhas repetidas e evita sobrecarga.
- **Idempotência e Redis**: Previnindo o reprocessamento de eventos, garantindo que a lógica de processamento seja correta e eficiente.

4. Uso de Mensageria para Comunicação Assíncrona

- **Kafka** (substituído por **SQS/SNS**): Escolha de um sistema de mensageria para garantir a entrega de mensagens entre os micro serviços com baixo custo e alto desempenho. No desafio foi feito com o kafka para garantir seu funcionamento localmente sem a necessidade de serviços externos.

5. Armazenamento e Banco de Dados

- **DynamoDB** para a API de consolidação diária: Banco NoSQL altamente escalável e gerenciado pela AWS, que é mais adequado para dados de leitura e escrita rápidos.
- **RDS (PostgreSQL)** para a API de lançamento: Banco relacional com alta disponibilidade, escalabilidade automática e fácil gerenciamento.

6. Escalabilidade e Gestão de Containers

- **Amazon EKS**: Serviço gerenciado de Kubernetes para orquestração de containers, garantindo escalabilidade e alta disponibilidade dos micro serviços.
- **Auto Scaling**: Ajuste automático da capacidade dos micro serviços conforme a carga de trabalho.

7. Monitoramento e Observabilidade

- **CloudWatch**: Monitoramento de logs e métricas na AWS para visibilidade do desempenho dos micro serviços.
- **Prometheus & Grafana**: Garantindo a coleta e visualização de métricas em tempo real para análise detalhada do desempenho e resolução de problemas.

8. API Gateway e Segurança

- **API Gateway**: Centraliza autenticação, autorização, e monitoração. Oferece cache de respostas e proteção contra DDoS, além de limitar a taxa de requisições.

- **Autenticação e Autorização**: Uso de **Azure AD** e **Amazon Cognito** para gestão de identidades, garantindo autenticação multi-fator e controle de acesso seguro aos micro serviços.

9. Armazenamento em Cache
    - **Redis**: Utilizado para cache em memória, garantindo alta performance e baixa latência, além de prevenir duplicação de eventos e controle de mensagens processadas.

Considerações Finais

A solução é escalável, resiliente, segura e de alto desempenho, adequada para atender os requisitos do desafio com alta disponibilidade e baixo custo operacional. A arquitetura é robusta, utilizando práticas modernas de micro serviços, monitoramento e integração contínua, e pode ser facilmente escalada para lidar com grandes volumes de dados e requisições.