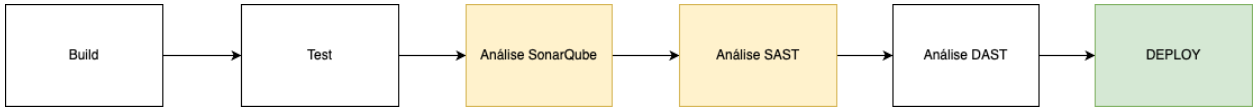
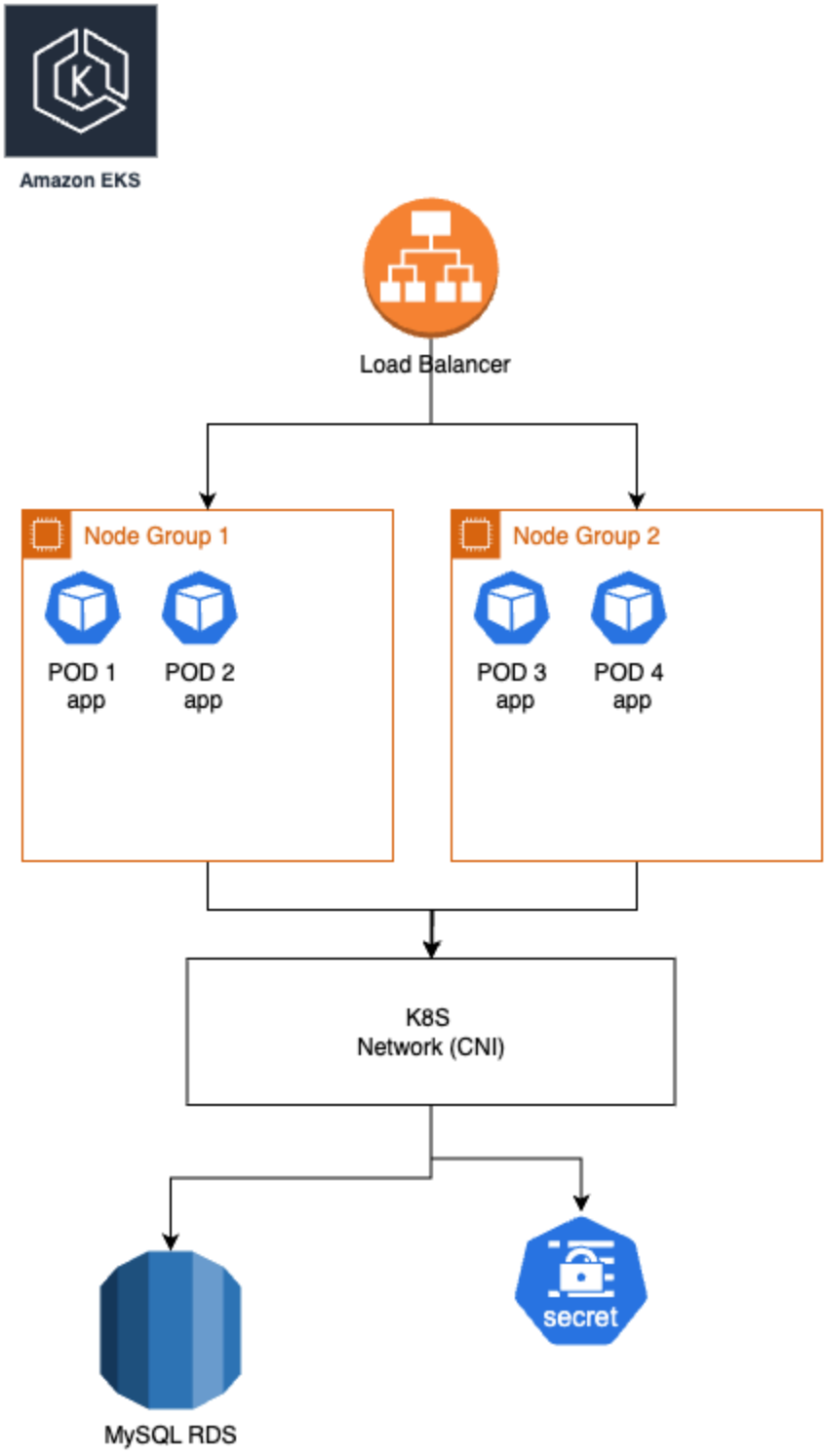
1. Implementação de um pipeline de CI/CD usando Github Actions

Estrutura do Pipeline



Build: transforma o código fonte em um artefato executável
Test: garante que o código funcione conforme o esperado
Análise SonarQube: analisa estaticamente a qualidade do código gerando um relatório
SAST: detecta vulnerabilidades de segurança no código estático
DAST: detecta vulnerabilidades de segurança em tempo de execução no ambiente de teste
Deploy: movimenta o código testado e aprovado para um ambiente não produtivo e produtivo.

2. Diagrama de K8S


Amazon EKS
Load Balancer
Node Group 1
POD 1
app
POD 2
app
Node Group 2
POD 3
app
POD 4
app
K8S
Network (CNI)
secret
MySQL RDS

Load Balancer

Função: Distribui o tráfego de entrada entre os nós do cluster Kubernetes.
Decisão de Design: Utilizar um serviço gerenciado, como o AWS ELB/NLB, para simplificar a configuração e manutenção.
Considerações de Segurança: Configurar regras de segurança para permitir apenas o tráfego necessário e proteger contra ataques.

Kubernetes Nodes

Função: Hospedam os pods que contêm as aplicações.
Decisão de Design: Ter pelo menos dois nós para garantir alta disponibilidade e resiliência.
Considerações de Segurança: Aplicar políticas de segurança como RBAC (Role-Based Access Control) para controlar o acesso ao cluster.

Pods

Função: Executam as aplicações dentro do cluster Kubernetes.
Decisão de Design: Dividir as aplicações em múltiplos pods para facilitar o escalonamento e a gestão.
Considerações de Segurança: Utilizar boas práticas de segurança na construção das imagens dos contêineres, como imagens minimais e sem vulnerabilidades conhecidas.

Kubernetes Network

Função: Facilita a comunicação entre os pods dentro do cluster.
Decisão de Design: Utilizar um plugin CNI (Container Network Interface) adequado, como Calico ou Flannel, para gerenciar a rede.
Considerações de Segurança: Implementar políticas de rede para controlar o tráfego entre os pods e limitar a exposição.

Database

Função: Armazenamento de dados persistentes para as aplicações.
Decisão de Design: Utilizar um banco de dados em contêiner, como MySQL, ou um serviço gerenciado como AWS RDS para facilitar a administração e backup.
Considerações de Segurança: Configurar acesso seguro, utilizar criptografia em trânsito e em repouso, e implementar controles de acesso rigorosos.

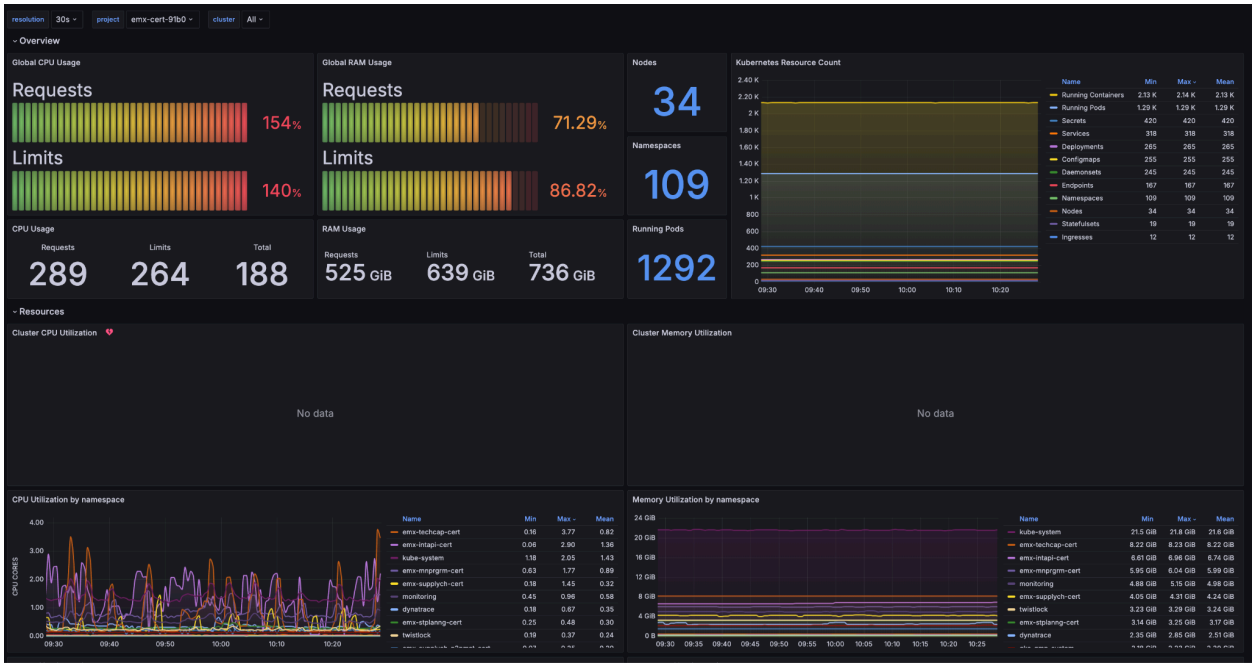
Secret Manager

Função: Armazenar informações sensíveis, como credenciais de banco de dados e chaves de API.
Decisão de Design: Utilizar o Kubernetes Secrets para gerenciar segredos de forma segura.
Considerações de Segurança: Proteger o acesso aos segredos utilizando RBAC e garantir que os segredos sejam encriptados.

3. Observabilidade

   Grafana: Para criação de dashboards personalizados e interativos.

   Prometheus: Para coleta de métricas e configuração de alertas.



Listagem de Recursos e Utilização:

- **CPU Usage:** Apresenta o uso de CPU das diferentes namespaces dentro do cluster Kubernetes. Mostra a utilização atual, limites definidos e a capacidade total disponível.
- **RAM Usage:** Exibe o uso de memória RAM por namespace, apresentando solicitações, limites e a capacidade total.
- **Cluster Utilization:** Inclui métricas globais de utilização de CPU e memória do cluster.
- **Kubernetes Resource Count:** Mostra a contagem de recursos no cluster, como pods, containers, serviços, namespaces, etc.

Painel de Alerta:

- **Critical Alert & Warning Alert:** Indicadores que mostram o estado atual dos alertas críticos e de aviso baseados nas métricas monitoradas. Isso ajuda na identificação rápida de problemas que precisam de atenção imediata.

- **Burn Rate Alerts:** Este painel mostra a taxa de consumo do orçamento de erros, que é importante para entender se o sistema está perto de exceder seus limites de SLO.
- **Error Budget Trend:** Exibe a tendência do orçamento de erros, ajudando a monitorar a saúde do serviço ao longo do tempo.
- **Namespace Selector:** Permite filtrar as métricas e os gráficos por namespaces específicos, facilitando a análise segmentada.
- **Time Range Selector:** Permite ajustar o intervalo de tempo das métricas exibidas, ajudando na análise de tendências e na identificação de anomalias em períodos específicos.

Filtros:

- **Namespace Selector:** Permite filtrar as métricas e os gráficos por namespaces específicos, facilitando a análise segmentada.
- **Time Range Selector:** Permite ajustar o intervalo de tempo das métricas exibidas, ajudando na análise de tendências e na identificação de anomalias em períodos específicos.

Latência (Latency):

- **Backend Latency:** Tempo médio de latência no backend, monitorado por Load Balancer (LB).

Tráfego (Traffic):

- **Bytes Sent as Responses:** Quantidade de dados enviados como respostas por segundo.

Erros (Errors):

- **Code Response Error per Second:** Taxa de respostas com erro (códigos HTTP 4xx e 5xx).

Saturação (Saturation):

- **CPU and Memory Usage:** Uso de CPU e Memória por namespace no cluster Kubernetes.

- **SLI (Service Level Indicator):**
    - **Disponibilidade:** Percentual de solicitações HTTP bem-sucedidas (200 OK) vs. total de solicitações.

    - **Latência:** Tempo médio de resposta das solicitações HTTP.
- **SLO (Service Level Objective):**
    - **Disponibilidade:** Meta de 99% de solicitações bem-sucedidas em um período de 30 dias.
    - **Latência:** Meta de tempo de resposta médio inferior a 500ms.
- **Indicadores de SLO:**
    - **Critical Alert:** Estado crítico do serviço baseado nos SLOs.
    - **Warning Alert:** Estado de aviso do serviço baseado nos SLOs.
    - **Error Budget:** Quantidade de erros permitidos antes de violar o SLO.

Conclusão

A criação de um Dashboard de Observabilidade de Infraestrutura Cloud que integra SLI/SLO, os Four Golden Signals e o monitoramento de Kubernetes é fundamental para garantir a saúde, desempenho e confiabilidade dos sistemas. Este dashboard oferece uma visão unificada e detalhada, permitindo que as equipes de DevOps e SRE monitorem eficientemente o estado dos serviços e tomem ações proativas para resolver problemas antes que eles impactem os usuários finais.

Benefícios:

Visibilidade Integrada: Combina métricas essenciais de disponibilidade, latência, tráfego, erros e saturação com a contagem e utilização de recursos do Kubernetes, proporcionando uma visão holística do sistema.

Monitoramento Contínuo: Ferramentas como Prometheus e Grafana garantem a coleta contínua e visualização em tempo real das métricas, permitindo detecção e resposta rápida a anomalias.

Alertas Proativos: Painéis de alerta configuráveis ajudam a detectar e priorizar problemas críticos e de aviso, garantindo que os SLOs sejam atendidos e os orçamentos de erro sejam gerenciados eficazmente.

Personalização: Filtros por namespace e intervalos de tempo permitem personalizar a visualização conforme necessário, facilitando a análise segmentada e específica de diferentes partes do sistema.