Teste Técnico - Analista DevOps

Você foi contratado por uma empresa fictícia responsável por lançar um produto digital que alcançou grande sucesso e, com isso, um crescimento acelerado no número de usuários. A plataforma principal, desenvolvida em PHP, é robusta e atende bem às necessidades do negócio, mas sua arquitetura e processo de implantação não acompanharam essa evolução.

Atualmente, o processo de deploy é manual e arriscado. Novas versões são implantadas via acesso SSH direto aos servidores de produção, seguido por um `git pull` e a execução manual de comandos. Esse método já causou instabilidade em momentos de pico, e o tempo para lançar novas funcionalidades é medido em semanas, não em dias. Os ambientes de desenvolvimento e produção raramente são idênticos, o que gera o clássico problema do "funciona na minha máquina".

Para resolver esses gargalos e garantir a estabilidade e agilidade que o crescimento dessa empresa exige, foi iniciado um projeto estratégico de modernização. É aqui que você entra.

O Desafio (Sua Missão)

Sua missão, como analista em DevOps, é acompanhar a primeira e mais crucial etapa dessa transformação. Você deverá pegar nossa aplicação web legada e projetar uma fundação sólida para seu ciclo de vida moderno, aplicando as melhores práticas de containerização, automação de esteiras (CI/CD) e gerenciamento de infraestrutura como código.

Recursos Fornecidos:

- **Aplicação Exemplo:** Um link para um repositório Git com o código-fonte de uma aplicação web simples em **PHP**. O código é funcional e você não precisa modificá-lo.

Etapas da Missão

Pedimos que você realize as seguintes etapas, organizando todo o seu trabalho em um repositório no GitHub.

Etapa 1: Containerização da Aplicação

- **Tarefa:** Escreva um `Dockerfile` para a aplicação PHP fornecida.
- **O que esperamos:**
  - Um `Dockerfile` otimizado e seguro, utilizando práticas como *multi-stage builds* e a execução com um usuário não-root;
  - O `Dockerfile` deve usar uma imagem base oficial apropriada para PHP;
  - O arquivo deve ser comentado, explicando as decisões importantes.

Etapa 2: Criação do Pipeline de Integração Contínua (CI)

- **Tarefa:** Crie um pipeline de CI utilizando GitHub Actions.
- **O que esperamos:**
  - Um arquivo de workflow (`.github/workflows/main.yml`) que seja acionado a cada `push` na branch principal;
  - O pipeline deve, no mínimo, realizar os seguintes passos:

1. Fazer o checkout do código;
2. Construir a imagem Docker a partir do seu `Dockerfile`;
3. Executar uma análise de vulnerabilidades na imagem Docker construída;
4. Fazer o push da imagem para um registro público, como o Docker Hub.

Etapa 3: Infraestrutura como Código (IaC) e Implantação (CD)

- **Tarefa:** Descreva e codifique a infraestrutura necessária para implantar a aplicação containerizada na AWS.
- **O que esperamos:**
  - **Código Terraform:** Crie os arquivos Terraform necessários para provisionar um cluster Kubernetes (EKS) simples ou um serviço de container (ECS/Fargate). A escolha da tecnologia de orquestração é sua, mas deve ser justificada;
  - **Manifestos Kubernetes/Task Definition:** Inclua os arquivos de manifesto do Kubernetes (`deployment.yaml`, `service.yaml`) ou a definição de tarefa do ECS para implantar a sua aplicação;
  - **Explicação do CD:** Em seu `README.md`, explique como o seu pipeline de CI seria estendido para se tornar um pipeline de Implantação Contínua (CD), implantando automaticamente a nova versão da imagem na infraestrutura criada.

Etapa 4: Estratégia de Observabilidade

- **Tarefa:** Você não precisa implementar, apenas descrever sua estratégia.
- **O que esperamos:**
  - No `README.md`, descreva brevemente qual stack de ferramentas você escolheria para monitorar esta aplicação em produção;
  - Justifique sua escolha e explique quais as 3 principais métricas que você coletaria da aplicação ou da infraestrutura para criar um dashboard de saúde do serviço.

Formato da Entrega

- **Um único link para um repositório público no GitHub.**
- O repositório deve conter todos os artefatos criados:
  - `Dockerfile`
  - A pasta `.github/workflows/` com o arquivo de pipeline
  - Os arquivos `.tf` do Terraform
  - Os arquivos de manifesto (`.yaml`) ou a definição de tarefa (`.json`)
- **Um arquivo `README.md` bem escrito e organizado** que servirá como seu relatório final, explicando suas decisões técnicas, a justificativa da escolha de ferramentas e a estratégia de observabilidade.