**1. Desafio Fácil: Automatizar uma Aplicação Web Simples com Docker e Nginx**

**Objetivo:** Criar uma aplicação web simples usando Docker, configurar o Nginx como um proxy reverso e implantá-la dentro de um contêiner Docker. Usar Bash para automatizar o processo de configuração.

**Etapas:**

* **Criar uma aplicação web básica**: Desenvolver uma página simples em HTML e implantá-la em um contêiner Docker.
* **Usar Docker**: Containerizar a aplicação web com um Dockerfile e criar uma configuração Docker Compose para gerenciar o contêiner facilmente.
* **Configurar o Nginx**: Configurar o Nginx como um proxy reverso para direcionar o tráfego para a aplicação dentro do contêiner.
* **Automatizar com Bash**: Escrever um script Bash para automatizar o processo de puxar as imagens, iniciar os contêineres e configurar o Nginx.

**Habilidades praticadas:**

* Fundamentos de Docker
* Script Bash
* Configuração do Nginx
* Implantação básica de uma aplicação web

**2. Desafio Médio: Dashboard de Visualização de Dados com um Backend Dockerizado**

**Objetivo:** Criar um dashboard de visualização de dados que exibe informações de uma API backend. O backend será containerizado com Docker, e você documentará a API usando Swagger. O front-end consumirá a API e visualizará os dados.

**Etapas:**

* **Criar uma API REST**: Construir uma API REST simples que fornece dados (por exemplo, de um arquivo JSON ou banco de dados) usando um framework como Flask ou Express.
* **Documentar com Swagger**: Adicionar documentação Swagger à API para facilitar o entendimento e os testes dos endpoints.
* **Dockerizar o backend**: Containerizar a API backend com Docker, criando um Dockerfile e um arquivo Docker Compose para rodar o serviço em um contêiner.
* **Criar um front-end de dashboard**: Construir um front-end (por exemplo, usando JavaScript ou um framework leve como React) que busque dados da API e os exiba de forma gráfica (por exemplo, com gráficos ou tabelas).
* **Provisionar o ambiente**: Usar uma ferramenta de provisionamento (como Ansible) para automatizar a configuração e implantação do ambiente, incluindo o backend e o frontend.

**Habilidades praticadas:**

* Docker e containerização
* Desenvolvimento de API e documentação com Swagger
* Visualização de dados (frontend)
* Provisionamento de ambiente com Ansible
* Trabalhando com APIs REST

**3. Desafio Difícil: Configurar um Cluster Kubernetes com Pipeline CI/CD para uma Aplicação de Microserviços**

**Objetivo:** Configurar um cluster Kubernetes para hospedar uma aplicação de microserviços com múltiplos contêineres. Automatizar a construção e o deploy da aplicação usando um pipeline CI/CD (Jenkins, GitLab CI, etc.), e configurar o monitoramento e escalabilidade com Kubernetes.

**Etapas:**

* **Configurar Kubernetes**: Criar um cluster Kubernetes (pode usar Minikube ou um provedor de nuvem como GKE ou EKS).
* **Criar uma aplicação de múltiplos contêineres**: Construir uma aplicação de múltiplos contêineres (por exemplo, um serviço de frontend, um serviço de backend e um banco de dados) e implantá-la no cluster Kubernetes usando manifestos Kubernetes.
* **Configurar o Nginx como Ingress Controller**: Configurar o Nginx como o ingress controller para direcionar o tráfego para diferentes serviços dentro do cluster Kubernetes.
* **Implementar um pipeline CI/CD**: Usar Jenkins, GitLab CI ou GitHub Actions para automatizar o pipeline CI/CD para construir e implantar os microserviços no Kubernetes quando mudanças são feitas no repositório.
* **Configurar escalabilidade e monitoramento**: Implementar escalabilidade horizontal de pods e configurar ferramentas de monitoramento (por exemplo, Prometheus, Grafana) para monitorar a saúde e o desempenho dos microserviços.

**Habilidades praticadas:**

* Kubernetes (configuração, escalabilidade, monitoramento)
* Aplicações de múltiplos contêineres
* Configuração de pipeline CI/CD e automação
* Configuração do Nginx para ingress
* Monitoramento e escalabilidade de microserviços