**Jornada de Desafios DevOps Pleno**

**31/08/2023**

**Nome: Paulo Ricardo Alves de Almeida**

**Orientações para a jornada:** Dentro deste material você encontrará **Parte A** com 06 desafios, muito parecido com o dia a dia da atividade como DevOps Pleno. **A Parte B** é composta de 05 perguntas, sendo 03 alternativas e 02 descritivas.

**Quanto ao tempo: 1 semana** para realizar essa jornada de desafios **(até 11/09/2023)**

**Onde disponibilizar:** pode disponibilizar as respostas no GitHub ou como ou onde entender melhor.

**Lembrete:** Se conseguir entregar todos os desafios ok. Porém lembre-se, não é uma obrigação e não é eliminatório, esta etapa complementa o processo da entrevista.

**Parte A – 6 Desafios**

**1.Desafio**: Criar um Dockerfile para uma Aplicação Angular com Interação de Microaplicativos .NET

**Descrição**:

Você está trabalhando em um projeto que envolve a criação de uma aplicação web usando Angular, que se comunica com vários microaplicativos .NET. Sua tarefa é criar um Dockerfile para empacotar a aplicação Angular em um contêiner Docker, garantindo que ela possa se comunicar com os microaplicativos .NET.

**Requisitos**:

A aplicação Angular deve será hospedada em um cluster kubernetes.

A aplicação Angular deve ser capaz de interagir com os microaplicativos .NET por meio de chamadas de API.

Os microaplicativos .NET já estão em execução e acessíveis por meio de URLs específicas.

/api/parceiro

/api/orcamento

**Instruções**:

Crie um Dockerfile na raiz do projeto Angular para que a aplicação possa ser empacotada em um contêiner Docker. Certifique-se de incluir os passos necessários para configurar o ambiente de execução da aplicação e garantir a comunicação com os microaplicativos .NET. Use um servidor Node.js para hospedar a aplicação Angular.

Seu Dockerfile deve incluir:

A escolha de uma imagem base apropriada.

As etapas para instalar as dependências da aplicação Angular.

A cópia dos arquivos da aplicação Angular para dentro do contêiner.

A exposição da porta correta para o servidor Node.js.

A configuração necessária para permitir que a aplicação Angular se comunique com os microaplicativos .NET.

Você pode assumir que os arquivos da aplicação Angular estão na pasta "src/"

**Desafio 2**: Criar um Dockerfile para uma Aplicação .NET 6 com Interação de Microaplicativos .NET

**Descrição**:

Você está envolvido em um projeto que envolve a criação de uma aplicação .NET 6 que se comunica com vários microaplicativos .NET. Sua tarefa é criar um Dockerfile para empacotar a aplicação .NET em um contêiner Docker, garantindo que ela possa se comunicar com os microaplicativos .NET.

**Requisitos**:

A aplicação .NET 6 deve ser executada em um ambiente Linux.

A aplicação .NET 6 deve ser capaz de interagir com os microaplicativos .NET por meio de chamadas de API.

Os microaplicativos .NET já estão em execução e acessíveis por meio de URLs específicas.

**Instruções**:

Crie um Dockerfile na raiz do projeto .NET 6 para que a aplicação possa ser empacotada em um contêiner Docker. Certifique-se de incluir os passos necessários para configurar o ambiente de execução da aplicação e garantir a comunicação com os microaplicativos .NET.

Seu Dockerfile deve incluir:

A escolha de uma imagem base apropriada para o ambiente Linux.

Boas práticas de segurança

As etapas para compilar a aplicação .NET 6.

A cópia dos binários da aplicação compilada para dentro do contêiner.

A exposição da porta correta para a aplicação .NET.

A configuração necessária para permitir que a aplicação .NET 6 se comunique com os microaplicativos .NET.

Você pode assumir que os arquivos da aplicação .NET 6 estão na pasta "src/" e que você já possui as URLs de acesso aos microaplicativos .NET.

**Desafio 3**: Provisionamento de Infraestrutura na Azure usando Terraform para uma Aplicação Kubernetes com .NET e Angular

**Descrição:**

A empresa está iniciando um novo projeto de aplicação que envolve a criação de uma infraestrutura na Microsoft Azure. A aplicação será executada em um cluster Kubernetes e consistirá em duas partes: uma aplicação .NET e uma aplicação Angular. A aplicação .NET se conectará a um banco de dados SQL Server, armazenará dados em uma conta de armazenamento e também usará um cache Redis.

**Requisitos**:

Provisione uma infraestrutura na Azure usando o Terraform.

Crie um cluster Kubernetes com pelo menos dois nós.

Cofnigure os Deployments

Configure a aplicação .NET para se conectar a um banco de dados SQL Server chamado "dbfenix".

Configure a aplicação .NET para armazenar dados em uma Storage Account.

Configure a aplicação .NET para usar um cache Redis.

Forneça um arquivo de configuração que possa ser usado para configurar o acesso ao banco de dados, storage account e Redis na aplicação .NET.

Utilize as boas práticas do Terraform e do Kubernetes para a criação e gerenciamento da infraestrutura.

**Instruções:**

Use o Terraform para criar os recursos necessários na Azure, incluindo o cluster Kubernetes.

Crie os arquivos de manifesto do Kubernetes para os Deployments da aplicação .NET e Angular.

Configure as conexões corretas para o banco de dados, storage account e Redis na aplicação .NET.

Forneça um arquivo de configuração que contenha as informações de conexão e configuração para a aplicação .NET.

**4.Desafio**: Criar um Pipeline do Azure para Compilação e Entrega dos Projetos .NET e Angular no Kubernetes

**Descrição**:

A empresa deseja automatizar o processo de compilação e implantação das aplicações .NET e Angular no cluster Kubernetes. Para isso, você precisa criar um pipeline de integração contínua e entrega contínua (CI/CD) no Azure DevOps para compilar os projetos e implantá-los no cluster Kubernetes.

**Requisitos**:

Crie um pipeline no Azure DevOps para compilar a aplicação .NET e a aplicação Angular.

O pipeline deve ser acionado automaticamente sempre que houver mudanças no repositório de código nas branchs main e develop.

O pipeline deve criar as imagens Docker das aplicações e armazená-las no Azure Container Registry.

As imagens Docker devem ser implantadas nos Deployments correspondentes no cluster Kubernetes.

A configuração do pipeline deve ser versionada e armazenada junto com o código.

**Instruções**:

Configure um repositório de código para as aplicações .NET e Angular no Azure DevOps.

Crie um arquivo YAML para definir o pipeline de CI/CD.

Defina os estágios do pipeline para compilar as aplicações, criar imagens Docker e implantá-las no Kubernetes.

Utilize os artefatos gerados pelo pipeline para implantar as imagens Docker no Kubernetes.

**Desafio 5:**

**Descrição:**

Você é o administrador de um servidor que executa um banco de dados MongoDB em um sistema Ubuntu. A versão atual do MongoDB é um pouco desatualizada e você precisa realizar uma atualização para a versão mais recente. O banco de dados está em uso ativo e é crucial que a atualização seja realizada com o mínimo de interrupção possível.

**Tarefa:**

Descreva os passos e etapas que você seguiria para realizar com sucesso a atualização do MongoDB no sistema Ubuntu. Certifique-se de incluir as melhores práticas para minimizar a interrupção do serviço e garantir que o processo seja executado de maneira segura e eficiente.

1. Primeiro deve-se garantir o armazenamento dos dados, por isso podemos fazer o backup em outro diretório, podendo estar em outra máquina ou em um volume da Azure, devemos entrar como root e rodar:

mongodump --out /caminho/para/o/backup/diretório

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu $(lsb\_release -cs)/mongodb-org/5.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list

1. Parar o serviço do mongodb para não dar ruim:

sudo systemctl stop mongod

1. Podemos então realizar a atualização do mongodb, adicionado o repo da versão mais recente (Mongodb 6.0), e instalar o package:

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu $(lsb\_release -cs)/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

sudo apt update

sudo apt upgrade mongodb-org

1. Iniciar serviço do mongo:

sudo systemctl restart mongod

1. Verificação dos logs de erros do mongodb, para garantir que nada deu errado:

sudo tail -f /var/log/mongodb/mongod.log

1. Caso tenha dado algum erro deve-se voltar pra versão anterior e voltar o backup.
2. Documentar com data e horário a atualização feita.
3. É importante ressaltar que a atualização deve-se realizar fora de horário de pico de acessos ao banco.

Importante: em vez de realizar tudo manualmente digitando dentro da máquina ubuntu, é melhor juntar todos os comandos ou o máximo deles em arquivo “Ansible”. E utilizar o ansible para automatizar essa configuração. Claro conversando com um líder de equipe ou outro engenheiro DevOps para concordarem e verem quando rodar a atualização.

**Desafio 06:**

A empresa decidiu interromper a utilização do sistema de finanças e agora deseja realizar o desprovisionamento completo do ambiente. No entanto, é necessário preservar uma cópia do projeto, assim como dos dados do banco. Por favor, prepare uma GMUD para executar essa tarefa.

**Desprovisionamento do Sistema de Finanças**

*Versão 1.0*

**1. Objetivo:** Este documento descreve os procedimentos a serem seguidos para desprovisionar completamente o ambiente do sistema de finanças da empresa. O objetivo é interromper a utilização do sistema, preservar uma cópia do projeto e dos dados do banco e garantir uma desativação segura.

**2. Responsabilidades:**

* [Nome do Responsável] - Responsável pelo desprovisionamento.
* [Nome do DBA] - Responsável pelo backup dos dados do banco.

**3. Pré-requisitos:**

* Aprovação da equipe de gestão.
* Backup completo do banco de dados realizado e testado.

**4. Procedimentos:**

**4.1. Backup dos Dados do Banco:** O DBA deve realizar um backup completo dos dados do banco de finanças, garantindo que os dados estejam seguros e acessíveis para futuras referências. Os detalhes do backup devem ser registrados.

**4.2. Preservação do Projeto:** Antes de desativar o sistema, é necessário preservar uma cópia completa do projeto, incluindo código-fonte, configurações e documentação associada. Isso pode ser feito arquivando todos os recursos em um repositório de controle de versão ou outro local seguro.

**4.3. Desativação do Ambiente:** O responsável pelo desprovisionamento deve seguir os seguintes passos:

a. Desligar todos os serviços e servidores relacionados ao sistema de finanças.

b. Remover qualquer integração ou dependência do sistema em outros sistemas ou serviços ativos.

**4.4. Comunicação:** É fundamental informar a equipe de TI e os usuários finais sobre a desativação do sistema de finanças. Forneça orientações sobre como acessar qualquer sistema alternativo, se aplicável.

**4.5. Auditoria e Verificação:** Após a desativação, verifique se todos os recursos do sistema de finanças foram completamente desprovisionados. Isso inclui a desativação de contas de usuário, revogação de permissões e a remoção de todos os componentes relacionados ao sistema.

**5. Documentação:** Mantenha um registro detalhado de todas as ações realizadas durante o processo de desativação. Isso inclui datas, horários, responsáveis e quaisquer problemas encontrados.

**6. Conclusão:** A desativação do sistema de finanças foi concluída com sucesso, e uma cópia do projeto e dos dados do banco foi preservada para referência futura.

**Parte B - Total de 05 perguntas, sendo 03 Alternativas e 02 Descritivas**

**Pergunta 1**

1. Você deseja detectar explorações conhecidas nas imagens Docker usadas para provisionar os containers Docker.

Você precisa integrar a verificação de imagens no ciclo de vida da aplicação. A solução deve expor as explorações o mais cedo possível durante o ciclo de vida da aplicação.

**O que você deve configurar?**

A. uma tarefa executada no pipeline de integração contínua e uma tarefa agendada que analisa o registro de imagens

B. tarefas manuais realizadas durante a fase planejada e a fase de implantação

C. uma tarefa executada no pipeline de implantação contínua e uma tarefa agendada em relação a um contêiner de produção em execução

D. uma tarefa executada no pipeline de integração contínua e uma tarefa agendada que analisa o contêiner de produção

**Pergunta 2**

2. Você tem um arquivo Dockerfile que contém o seguinte código. (Os números de linha estão incluídos apenas como referência.)

1 FROM Microsoft/dotnet:2.1-sdk

2 COPY . ./

3 RUN dotnet publish -c Release -o out

4 FROM Microsoft/dotnet:2.1-sdk

5 COPY -from=0 /out /

6 WORKDIR /

7 ENTRYPOINT [“dotnet”,”app.dll”]

Você precisa garantir que a imagem seja a menor possível quando a imagem for construída.

Qual linha você deve modificar no arquivo?

A. 1

B. 3

C. 4

D. 7

**Pergunta 3**Você precisa recomendar uma estratégia de construção de contêiner Docker que atenda aos seguintes requisitos:

Minimize os tamanhos das imagens

Minimize a área de superfície de segurança da imagem final

O que você deve incluir na recomendação?

A. MultiStage build

B. Desired State Configuration (DSC) do PowerShell

C. Docker Swarm

D. SingleStage build

**Pergunta 4**

O que é RBAC e como ele funciona em um ambiente de TI? Quais são os principais benefícios de implementar o RBAC em uma organização?

O RBAC é um recurso de segurança importante em um ambiente de TI, por que ele serve como controle de acesso a determinadas informações/funções dentro desse ambiente, basicamente você terá usuários que estarão associados a roles (Regras), esses usuários só poderão acessar/executar aquilo que ele tem permissão prévia.

Este é um conceito importante dentro do Kubernetes, uma vez que o RBAC permite integrar seu cluster com diversas outras aplicações ou serviços externos como a núvem (Azure), dentre os objetos no kubernetes relacionados a isso temos:

- **Role (Privado á um namespace) e ClusterRole (irrestrito a namespace).**

**- RoleBinding e ClusterRoleBinding.**

**- ServiceAccount.**

**- Secrets.**

**Pergunta 5**

Descreva os três principais componentes de um sistema RBAC. Como esses componentes colaboram para garantir o acesso apropriado aos recursos dentro de uma organização?

Desafio: Atualização do MongoDB em um Sistema Ubuntu

* Funções (Roles):
  + definem as responsabilidades ou cargos dos usuários ou entidades em uma organização. Cada função tem um conjunto específico de permissões associadas a ela.
* Permissões (Permissions):
  + especificam o que é permitido ou proibido em um sistema. Cada permissão é atribuída a uma função.
* **Relações entre Funções e Permissões:**
  + definem quem pode fazer o quê. Essas relações estabelecem as políticas de controle de acesso