Plano de Aula: Deploy Automatizado de Aplicação Java na AWS com Terraform e GitHub Actions

Objetivo do Curso

Este plano de aula guia os alunos na construção de um pipeline de CI/CD (Integração e Entrega Contínua) completo e **sem custos**. O objetivo é implantar uma aplicação Java (Spring Boot) em uma instância EC2 na AWS, automatizando todo o processo de provisionamento de infraestrutura com Terraform e orquestrando o build e deploy com GitHub Actions. Ao final, os alunos terão experiência prática com as ferramentas mais modernas de DevOps do mercado.

Tecnologias Utilizadas

- **Cloud:** Amazon Web Services (AWS)
- **CI/CD:** GitHub Actions
- **Infraestrutura como Código (IaC):** Terraform
- **Aplicação:** Java 17 com Spring Boot e SQLite
- **Controle de Versão:** Git e GitHub

Roteiro Prático: Passo a Passo do Projeto

Módulo 0: Preparação do Ambiente (O Alicerce de Tudo)

Antes de escrevermos qualquer código, precisamos preparar nossas ferramentas e contas.

- **Passo 1: Configurando sua conta AWS e o Usuário IAM**
    - **O que fazer:** Criar uma conta na AWS e, crucialmente, criar um usuário no serviço **IAM** com permissões de acesso programático (Chave de Acesso e Chave Secreta). Este usuário será utilizado pelas ferramentas de automação.
- **Passo 2: Configurando seu Repositório no GitHub**
    - **O que fazer:** Criar um novo repositório público no GitHub. Ele abrigará o código da aplicação Java, os scripts de infraestrutura do Terraform e o workflow do GitHub Actions.

Módulo 1: Desenvolvendo a Aplicação Java

Agora vamos criar a aplicação que será a estrela do nosso deploy.

- **Passo 3: Criando a Aplicação Spring Boot com Banco de Dados SQLite**
  - **O que fazer:** Usar o Spring Initializr para gerar um projeto Java com as dependências `Spring Web`, `Spring Data JPA` e `JDBC para SQLite`. O objetivo é desenvolver uma API REST simples (ex: CRUD de "Alunos") com Entidade, Repositório e Controlador.
- **Passo 4: Criando o Script de População do Banco de Dados**
  - **O que fazer:** Criar um arquivo `data.sql` na pasta `src/main/resources`. O Spring Boot executará este script automaticamente na inicialização para popular o banco de dados com dados iniciais.

Módulo 2: Descrevendo a Infraestrutura com Terraform

Com a aplicação pronta, vamos dizer à AWS, via código, qual ambiente queremos para ela.

- **Passo 5: Escrevendo seus arquivos Terraform**
  - **O que fazer:** Na pasta `infra/`, criar os arquivos de configuração do Terraform:
    - **`provider.tf`**: Declara a AWS como provedor.
    - **`main.tf`**: Define os recursos principais: uma instância EC2 `t2.micro` (Free Tier) com Amazon Linux 2 e um Security Group para liberar as portas 8080 (HTTP) e 22 (SSH).
    - **`user_data`**: Dentro do recurso EC2, usar este bloco para passar um script que instala o Java na instância no momento da sua criação.
    - **`outputs.tf`**: Configura um "output" para que o Terraform informe o IP público da instância EC2 após a sua criação.

Módulo 3: Construindo a Automação com GitHub Actions

Esta é a fase onde conectamos tudo, criando o pipeline de CI/CD.

- **Passo 6: Configurando os Segredos no GitHub**
  - **O que fazer:** No repositório GitHub (`Settings > Secrets and variables > Actions`), configurar os seguintes segredos para evitar expor credenciais no código:
    - `AWS_ACCESS_KEY_ID` : A chave de acesso do usuário IAM.
    - `AWS_SECRET_ACCESS_KEY` : A chave secreta do usuário IAM.
    - `SSH_PRIVATE_KEY` : A chave SSH privada para acesso à instância EC2.
    - `SSH_PUBLIC_KEY` : A chave SSH pública correspondente.

- **Passo 7: Escrevendo o Workflow do GitHub Actions**
  - **O que fazer:** Criar o arquivo `deploy.yml` dentro de `.github/workflows/` para descrever o processo de automação, contendo os passos de: Checkout, Setup Java, Build com Maven, Configuração de Credenciais AWS, Setup Terraform, `terraform init`, `terraform destroy` (para limpeza), `terraform apply` (para criação), cópia do artefato `.jar` via SCP e execução do script de inicialização na instância via SSH.

Módulo 4: Vendo a Mágica Acontecer e Limpando o Ambiente

- **Passo 8: Testando o Pipeline**
  - **O que fazer:** Fazer uma alteração no código (ex: adicionar um novo aluno no `data.sql`), e realizar o `git commit` e `git push`. Acompanhar a execução do workflow em tempo real na aba "Actions" do GitHub.
- **Passo 9: Validando o Deploy**
  - **O que fazer:** Após a conclusão do workflow, obter o IP público da instância nos logs e acessar `http://<IP_DA_EC2>:8080/alunos` no navegador para verificar se a aplicação está no ar com os dados atualizados.

- **Passo 10: Destruindo a Infraestrutura**
  - **O que fazer:** Ensinar sobre a importância do comando `terraform destroy` para ser executado ao final dos estudos, garantindo que nenhum recurso fique ativo e gere custos. *Nota: Nosso pipeline já faz isso automaticamente a cada execução, mas é fundamental entender o comando isoladamente*.

Fundamentos Teóricos e Material de Apoio

1. Infrastructure as Code (IaC): O que é e por que é importante

A Infraestrutura como Código (IaC) é a prática de gerenciar e provisionar a infraestrutura de TI por meio de código, em vez de processos manuais. Isso traz benefícios como automação, consistência e a capacidade de versionar a infraestrutura.

Referências:

- Infrastructure as Code (IaC) — What is it? (The Startup - Medium)
- What is Infrastructure as Code (IaC) and Why It's Transforming DevOps (Medium)
- 5 reasons to use Infrastructure as Code (IaC) (Digital Power - Medium)

2. Terraform: O básico da sintaxe HCL, provedores, recursos, variáveis e outputs

O Terraform é uma ferramenta de IaC de código aberto criada pela HashiCorp. Ele utiliza uma linguagem declarativa chamada HCL (HashiCorp Configuration Language) para definir a infraestrutura desejada.

Referências:

- Terraform Configuration Files: A Comprehensive Guide (Medium)
- Let's Terraform Part 3: Basic Syntax and How To Create Resources (YouTube)
- Terraform Configuration: Overview Of HCL Syntax (Build5Nines)

3. CI/CD: O que significam integração contínua e entrega/implantação contínua

CI/CD é uma prática essencial no desenvolvimento de software moderno que automatiza as fases de integração, teste e entrega.

Referências:

- Continuous Integration and Continuous Delivery (CI/CD) (Medium)
- A Complete Guide to CI/CD: Continuous Integration and Continuous Delivery Explained
- CI/CD: What is Continuous Integration and Delivery? (YouTube)

4. GitHub Actions: O que são workflows, jobs, steps e actions

GitHub Actions é a plataforma de CI/CD nativa do GitHub, que permite automatizar tarefas diretamente do seu repositório.

Referências:

- GitHub Actions: Your First Workflow (Basic — step by step) (Medium)
- Understanding the basics of GitHub Actions (Medium)
- Automate Your Workflow: Learn GitHub Actions in 10 Minutes! (YouTube)

5. AWS Free Tier: Quais serviços estão incluídos e seus limites

O AWS Free Tier permite que novos usuários experimentem uma variedade de serviços da AWS gratuitamente, dentro de certos limites.

Referências:

- Understanding the AWS Free Tier: What Can You Do for $0? (Jeevi Academy)
- Understanding AWS Free Tier Guide (Medium)
- AWS Free Tier Limits - Everything You Need To Know (SourceFuse)