****FATEC Itapira "Dr. Ogari de Castro Pacheco"****

**D**ESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA****

****Aplicação Multiplataforma AWS****

## ****Professor :****

## ****Mateus Guilherme Fuini****

## ****Equipe:****

## ****Bruno Oliveira****

## ****Luis Felipe****

## ****Tiago Moraes****

## ****Rafael Oliveira****

## ****Carlos Nicioli****

****Sumário****

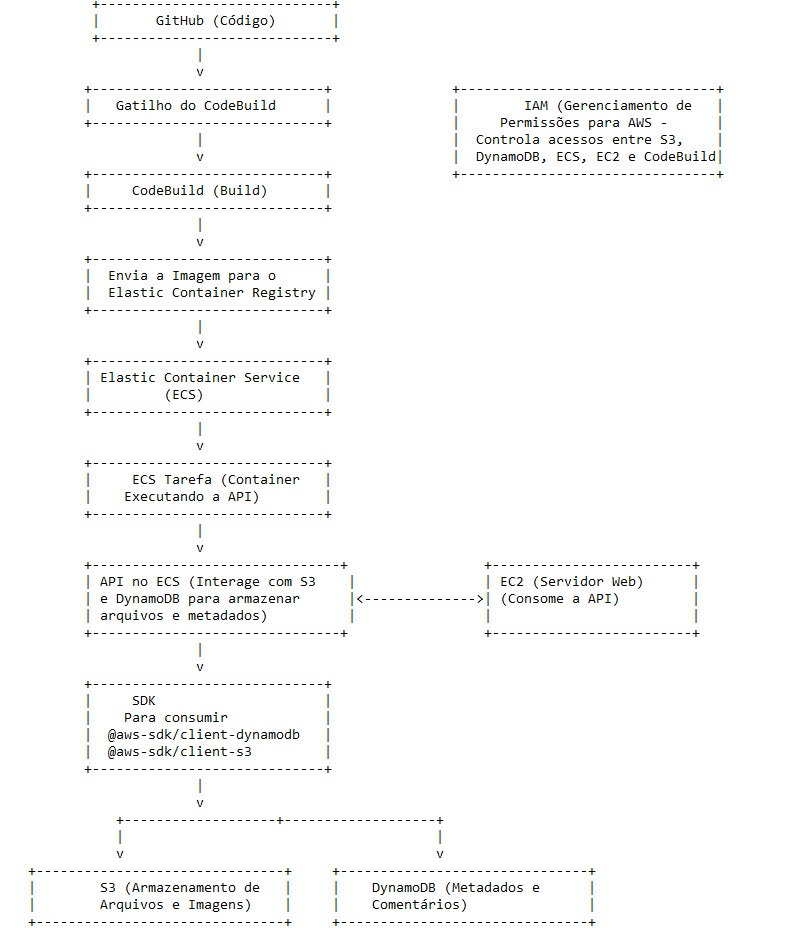
1. Introdução ...........................................................................................................
2. Diagrama da Solução ..........................................................................................
3. Justificativa dos Serviços ...................................................................................
4. Fluxos ..................................................................................................................
5. Boas Práticas Aplicadas .................................................................................….
6. Conclusão ....................................................................................................…...
7. Referências .......................................................................................................…

**1. Introdução**

Este documento descreve a arquitetura e os serviços utilizados na construção de uma solução baseada em nuvem utilizando a plataforma AWS (Amazon Web Services). O objetivo é demonstrar como os serviços integrados permitem a automação, armazenamento e orquestração de uma aplicação modular. A solução inclui um pipeline CI/CD, interação com bancos de dados e armazenamento de arquivos, além de boas práticas de segurança e escalabilidade.

### ****2. Diagrama da Solução****

Abaixo está o diagrama representando o fluxo do projeto. Ele ilustra como os componentes interagem entre si:



### ****3. Justificativa dos Serviços****

#### ****GitHub****

* **Função** : Repositório de código-fonte.
* **Justificativa** : O GitHub é usado para armazenar o código do projeto e controlar versões. Ele também ajuda a integrar o pipeline de CI/CD (Integração Contínua e Entrega Contínua), permitindo que o sistema detecte automaticamente quando há mudanças no código e inicie o processo de build.

#### ****AWS CodeBuild****

* **Função** : Realiza o processo de build do código-fonte.
* **Justificativa** : O CodeBuild é como um "trabalhador" que pega o código do GitHub, compila ele e gera uma imagem Docker pronta para ser usada. Ele é gerenciado pela AWS, então não precisamos configurar servidores ou se preocupar com manutenção.

#### ****Elastic Container Registry (ECR)****

* **Função** : Armazena imagens Docker geradas pelo CodeBuild.
* **Justificativa** : O ECR é como um "guarda-roupas" onde colocamos as imagens Docker depois de criadas. Ele é seguro e funciona bem com outros serviços da AWS, como o ECS, facilitando o uso dessas imagens em produção.

#### ****Elastic Container Service (ECS)****

* **Função** : Orquestra e executa contêineres Docker.
* **Justificativa** : O ECS é responsável por pegar as imagens Docker do ECR e rodar elas em contêineres. Ele cuida de tudo: escala os contêineres conforme necessário e garante que eles estejam sempre funcionando.

#### ****Amazon S3****

* **Função** : Armazena arquivos e imagens.
* **Justificativa** : O S3 é como um "armário" gigante na nuvem onde guardamos arquivos e imagens. Ele é seguro, rápido e pode armazenar grandes volumes de dados sem problemas.

#### ****DynamoDB****

* **Função** : Armazena metadados e comentários.
* **Justificativa** : O DynamoDB é um banco de dados NoSQL que guarda informações estruturadas, como metadados e comentários. Ele é rápido, escalável e não precisa de muita configuração manual.

#### ****IAM (Identity and Access Management)****

* **Função** : Gerencia permissões e controle de acesso.
* **Justificativa** : O IAM é como um "porteiro" que decide quem pode acessar o que. Ele garante que apenas os serviços autorizados possam acessar recursos como o S3, DynamoDB e ECS, evitando problemas de segurança.

#### ****EC2 (Elastic Compute Cloud)****

* **Função** : Atua como servidor web que consome a API.
* **Justificativa** : O EC2 é uma máquina virtual na nuvem que usamos como cliente para consumir a API. Ele simula o comportamento de um servidor web que interage com a nossa aplicação.

#### ****SDK AWS****

* **Função** : Permite a interação programática com os serviços AWS.
* **Justificativa** : O SDK AWS é como uma "biblioteca mágica" que facilita a comunicação entre a nossa API e os serviços AWS, como o S3 e o DynamoDB. Ele já vem pronto e seguro, então não precisamos nos preocupar em criar essa integração do zero.

### ****4. Fluxos****

#### ****Fluxo de Build e Deploy****

1. O código-fonte é enviado para o repositório do GitHub.
2. O CodeBuild é acionado automaticamente por um gatilho configurado no GitHub.
3. O CodeBuild realiza o build do código, gerando uma imagem Docker.
4. A imagem Docker é enviada para o Elastic Container Registry (ECR).
5. O Elastic Container Service (ECS) puxa a imagem do ECR e inicia a execução do contêiner.

#### ****Fluxo de Execução da API****

1. A API rodando no ECS recebe requisições do EC2 (servidor web).
2. A API utiliza o SDK AWS para interagir com o DynamoDB (armazenamento de metadados e comentários) e o S3 (armazenamento de arquivos e imagens).
3. As respostas são retornadas ao EC2, que as exibe para o usuário final.

### ****5. Boas Práticas Aplicadas****

* **Segurança** : Usamos o IAM para garantir que cada serviço tenha apenas as permissões necessárias, seguindo o princípio do menor privilégio.
* **Escalabilidade** : O ECS e o CodeBuild são serviços gerenciados que podem escalar automaticamente conforme a demanda.
* **Manutenibilidade** : O GitHub facilita o versionamento e colaboração, enquanto o uso de infraestrutura como código (IaC) simplifica o gerenciamento de recursos.
* **Otimização de Custos** : Usamos serviços gerenciados para reduzir custos operacionais e automatizamos o pipeline CI/CD para minimizar erros humanos.

### ****6. Comentário Final****

Este projeto foi desenvolvido para demonstrar como criar uma aplicação completa usando serviços da AWS. A ideia é que o usuário envie arquivos e comentários através de uma API, que armazena esses dados no DynamoDB e no S3. O sistema é construído de forma modular, com cada componente desempenhando uma função específica.

Uma das maiores dificuldades encontradas foi configurar as permissões no IAM. Como o IAM controla o acesso entre todos os serviços, qualquer erro nas políticas pode impedir que a aplicação funcione corretamente. Além disso, foi necessário aprender a usar o SDK AWS para integrar a API com o S3 e o DynamoDB, o que exigiu um bom entendimento de como esses serviços funcionam juntos.

Apesar dos desafios, o projeto foi uma ótima oportunidade para aprender sobre automação, segurança e boas práticas no desenvolvimento de soluções na nuvem.

# ****7. Referências:****

AMAZON WEB SERVICES. **AWS Documentation** . Disponível em: <https://aws.amazon.com/documentation/>.

GITHUB. **GitHub Docs** . Disponível em: <https://docs.github.com/>.