Relatório Técnico – Projeto de Hospedagem de Ambiente de Desenvolvimento Seguro na AWS

Version – V1.0

24 de Novembro de 2024

Prepared by: Tales Ivalque Taveira de Freitas – Insper.

Conteúdos

Sumário

Esse projeto tem como objetivo criar um ambiente de desenvolvimento seguro e eficiente usando os serviços da AWS. A ideia é separar bem os ambientes de **desenvolvimento**, **teste** e **produção**, garantindo que cada um funcione de forma isolada para aumentar a segurança e facilitar o gerenciamento.

Além disso, focamos muito na parte de **segurança da informação**, como configurar permissões específicas com IAM, proteger os dados com criptografia e criar grupos de segurança para limitar acessos. Também usamos **ferramentas de automação** como GitHub Actions para criar pipelines que fazem o deploy, rodam testes e gerenciam o ambiente automaticamente.

A infraestrutura foi toda criada com **CloudFormation**, o que permite que ela seja gerenciada como código, facilitando mudanças e garantindo que tudo fique organizado e versionado.

A aplicação teve como ideia principal um sistema de chat adaptado a conteúdo personalizável pelo usuário.

Arquitetura

A arquitetura do projeto foi desenvolvida com o objetivo de criar um ambiente seguro, eficiente e modular, utilizando os serviços da AWS para atender aos requisitos de desenvolvimento, teste e produção. Abaixo, apresentam-se os detalhes da implementação:

---

1. Separação de Ambientes
Foram configurados três ambientes distintos para **desenvolvimento**, **teste** e **produção**, garantindo isolamento total entre eles:

- **VPCs individuais** para cada ambiente, assegurando que não haja interferência ou compartilhamento direto de recursos.
- Subnets públicas em cada VPC para a hospedagem das instâncias EC2 e acesso controlado.

---

2. Hospedagem da Aplicação
Cada ambiente possui:

- Uma instância **EC2** configurada para hospedar a aplicação Streamlit.
- Configuração da instância com o Python 3, Git, e instalação das dependências diretamente do repositório no GitHub.
- Uso de **DynamoDB** como banco de dados principal para armazenar informações de forma escalável e segura.

A arquitetura como um todo pode ser vista na Figura 1.

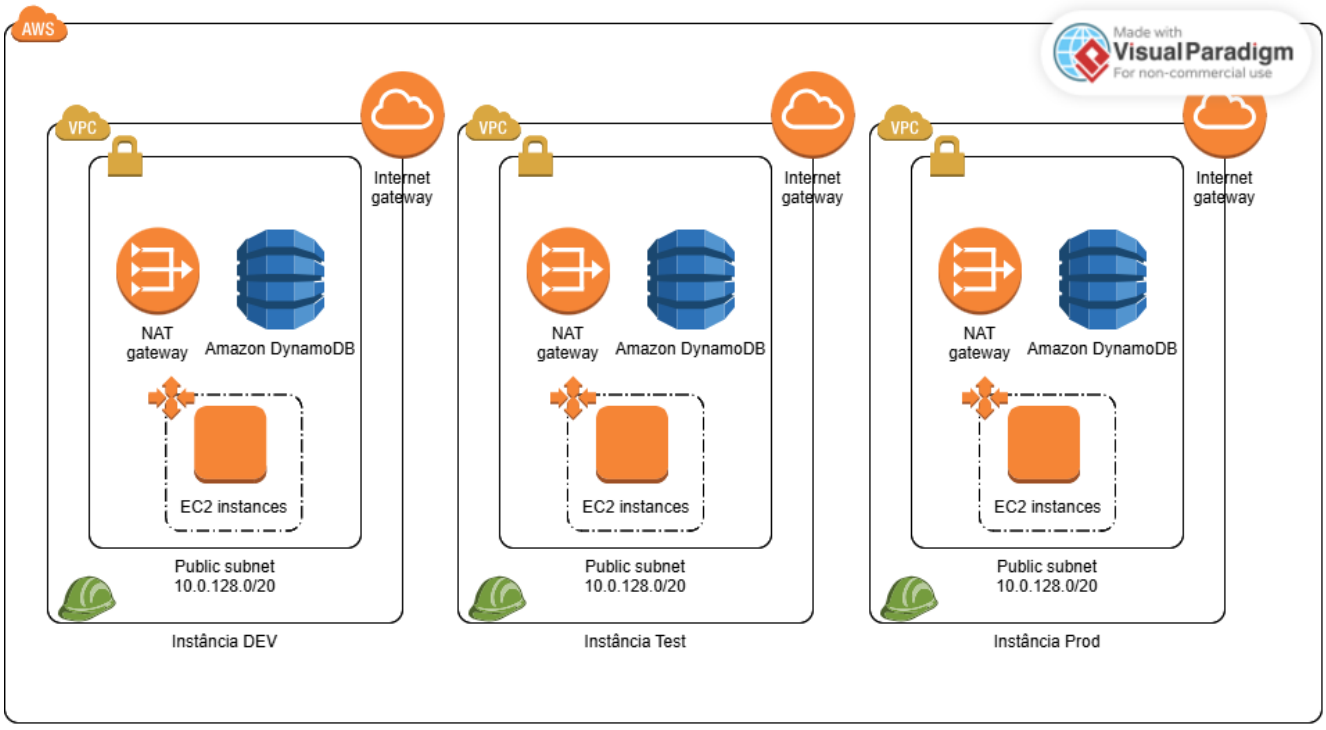

Figura 1 - Arquitetura AWS

Segurança

Os principais focos de segurança estão nos pontos:

1. Controle de Acesso (IAM - Identity and Access Management)

Cada instância EC2 foi configurada para operar com usuários IAM específicos, atribuídos exclusivamente a cada ambiente: desenvolvimento, teste e produção. Isso garante que cada instância tenha permissões limitadas ao seu ambiente, reduzindo riscos de acessos indevidos.

Além disso, os papéis IAM associados às instâncias EC2 determinam as ações permitidas. Por exemplo, uma instância do ambiente de desenvolvimento só pode acessar a tabela do DynamoDB correspondente (dev_streamlit_table). Já as tabelas de teste e produção são inacessíveis para instâncias de outros ambientes.

As permissões foram configuradas com políticas personalizadas, restringindo o acesso a recursos como tabelas do DynamoDB e buckets do S3, baseando-se no princípio do menor privilégio.

---

2. Grupos de Segurança (Security Groups)

Cada ambiente possui um grupo de segurança próprio, que regula o tráfego de entrada e saída.

- O acesso SSH (porta 22) está liberado apenas para o IP do administrador responsável, garantindo que conexões externas sejam controladas.
- O acesso à aplicação Streamlit, pela porta 8501, é limitado para conexões autorizadas.

Essas configurações isolam os ambientes e limitam o tráfego apenas ao que é estritamente necessário.

---

3. Criptografia

A criptografia foi implementada para proteger os dados tanto em repouso quanto em trânsito.

- O **DynamoDB** foi configurado com criptografia gerenciada pela AWS, protegendo automaticamente todos os dados armazenados.
- Para o tráfego em trânsito entre a aplicação e o banco de dados, o TLS (Transport Layer Security) é utilizado, garantindo comunicações seguras.

Embora o HTTPS ainda não esteja ativo na aplicação Streamlit, há planejamento para futura implementação de um certificado SSL para ampliar a proteção.

4. Configurações de Rede

Cada ambiente foi configurado com sua própria VPC, garantindo isolamento total entre desenvolvimento, teste e produção.

As instâncias EC2 estão em subnets públicas, conectadas a um Internet Gateway, para permitir acesso controlado aos recursos. As rotas foram configuradas para direcionar o tráfego apenas aos destinos necessários, protegendo os dados e reduzindo riscos de exposição indevida.

Infraestrutura como Código

No projeto, a infraestrutura foi completamente gerenciada como código utilizando o **AWS CloudFormation**.

Cada ambiente (desenvolvimento, teste e produção) foi definido em arquivos YAML separados, garantindo **modularidade** e permitindo ajustes específicos para cada caso. Por exemplo, tabelas distintas do DynamoDB foram configuradas para cada ambiente, com permissões de acesso restritas e separadas.

Além disso, os recursos, como VPCs, subnets, instâncias EC2, grupos de segurança e tabelas DynamoDB, foram provisionados de forma automatizada, eliminando erros manuais e otimizando o processo de deploy.

Essa abordagem também facilitou o **controle de mudanças**, já que todas as configurações estão documentadas e podem ser ajustadas conforme necessárias, utilizando pipelines de CI/CD para aplicar as atualizações automaticamente.

Os códigos fontes podem ser vistos no repositório de entrega.

CI/CD

No projeto, configuramos um **usuário IAM exclusivo** para gerenciar o deploy automático nos ambientes de desenvolvimento, teste e produção. Esse usuário foi projetado para ter permissões mínimas e específicas para garantir a segurança e o controle durante o processo de automação. O deploy é realizado utilizando o **GitHub Actions**, com um pipeline simplificado que realiza três etapas principais: acionamento, conexão à instância EC2 e execução da aplicação.

Etapas do Pipeline

Trigger Automático

O pipeline é acionado automaticamente quando há um push em branches específicas do repositório no GitHub. Cada branch está vinculada a um ambiente:

- **Branch dev**: Atualiza o ambiente de desenvolvimento.
- **Branch test**: Atualiza o ambiente de teste.
- **Branch main**: Atualiza o ambiente de produção.

Isso garante que os ambientes sejam atualizados apenas quando mudanças apropriadas são realizadas em suas respectivas branches.

Conexão com a EC2

O pipeline utiliza as credenciais do usuário IAM configurado para conectar-se às instâncias EC2 via SSH. A chave privada para autenticação é armazenada com segurança no **GitHub Secrets**, garantindo que o acesso seja controlado e protegido. Após a conexão, o pipeline acessa o diretório da aplicação no servidor (/home/ec2-user/streamlit_app).

Execução da Aplicação

Dentro da instância EC2, o pipeline executa os seguintes passos:

- Atualiza o repositório com o comando git pull, garantindo que a aplicação tenha o código mais recente.
- Reinicia a aplicação Streamlit utilizando o comando:

  nohup streamlit run app.py --server.port 8501 --server.address 0.0.0.0 > streamlit.log 2>&1 &

Configuração do Usuário IAM

Um usuário IAM foi criado exclusivamente para os pipelines, com as seguintes características:

- **Permissões restritas**: O usuário possui apenas as permissões necessárias para acessar as instâncias EC2 e realizar o deploy.
- **Chaves de acesso**: As credenciais do usuário são armazenadas de forma segura no GitHub Secrets, utilizadas apenas pelo GitHub Actions.