**Nome:** Henrique Rocha Bomfim **TecHacker 7º Sem Eng Comp**

# Documentação do Projeto: Infraestrutura AWS com Terraform

## 1. Introdução e Objetivo do Projeto

Este documento descreve o processo de criação e configuração de uma infraestrutura na Amazon Web Services (AWS) utilizando Terraform. [cite: 2] O objetivo é hospedar uma aplicação web composta por um frontend (Next.js) e um backend (API FastAPI com PostgreSQL), seguindo as melhores práticas de segurança, modularização e infraestrutura como código. [cite: 2] Adicionalmente, um sistema de detecção de intrusão e monitoramento de segurança (Wazuh) é integrado ao ambiente. [cite: 3] Esta documentação serve como um registro do desenvolvimento até o ponto de implantação via terraform apply. [cite: 3]

## 2. Configuração Inicial do Ambiente AWS

### 2.1. Criação de Usuário IAM Admin

O primeiro passo foi a criação de um usuário IAM (Identity and Access Management) na conta AWS com permissões administrativas. [cite: 4] Esta abordagem evita o uso da conta root para operações do dia a dia, seguindo as melhores práticas de segurança da AWS. [cite: 5] **Ações Realizadas:** \* Criação de um novo usuário IAM. [cite: 6] \* Atribuição da política AdministratorAccess a este usuário. [cite: 6] \* Geração de chaves de acesso (Access Key ID e Secret Access Key) para este usuário, para permitir o acesso programático. [cite: 7]

### 2.2. Configuração de Credenciais AWS (AWS CLI)

As chaves de acesso geradas para o usuário IAM Admin foram configuradas localmente no ambiente de desenvolvimento utilizando o AWS Command Line Interface (AWS CLI). [cite: 8] **Ações Realizadas:** \* Instalação do AWS CLI. [cite: 9] \* Execução do comando aws configure. [cite: 9] \* Inserção do Access Key ID, Secret Access Key, região padrão (ex: us-east-2) e formato de saída padrão (ex: json). [cite: 10] \* Isso criou um perfil de credenciais (geralmente o perfil default) que o Terraform utilizará para autenticar com a AWS. [cite: 11]

## 3. Configuração do Ambiente de Desenvolvimento Local

### 3.1. Instalação do Terraform

O Terraform foi instalado no ambiente de desenvolvimento local para permitir a criação e gerenciamento da infraestrutura como código. [cite: 12] **Ações Realizadas:** \* Download do binário do Terraform do site oficial. [cite: 13] \* Configuração do PATH do sistema para incluir o executável do Terraform, permitindo que ele seja chamado de qualquer diretório no terminal. [cite: 13]

## 4. Estrutura do Projeto Terraform (INFRA-PROJETO-AWS)

O projeto Terraform foi organizado de forma modular para promover a reutilização e a clareza. [cite: 14]

### 4.1. Visão Geral

A estrutura do projeto consiste em um módulo raiz que orquestra módulos locais para diferentes componentes da infraestrutura. [cite: 15]

INFRA-PROJETO-AWS/ ├── modules/ │ ├── ec2/ # Módulo para instâncias EC2 │ ├── security/ # Módulo para Security Groups │ └── vpc/ # Módulo para VPC e rede ├── main.tf # Configuração principal do módulo raiz ├── variables.tf # Declaração de variáveis de entrada do projeto ├── terraform.tfvars # Valores para as variáveis de entrada ├── outputs.tf # Saídas do projeto ├── provider.tf # Configuração do provedor AWS [cite: 16, 17] ├── database\_postgres.tf # Configuração do RDS e Secrets Manager [cite: 17] ├── user\_data\_frontend.sh # Script de inicialização do frontend [cite: 17] ├── user\_data\_backend.sh # Script de inicialização do backend [cite: 17] └── user\_data\_wazuh\_server.sh # Script de inicialização do servidor Wazuh

### 4.2. Módulo Raiz

Localizado na pasta INFRA-PROJETO-AWS/, o módulo raiz contém: \* main.tf: Define os recursos globais (como a configuração IAM para EC2, o perfil da instância, os data sources para os scripts de user data, e os recursos para os servidores Wazuh e Zabbix) e chama os módulos locais (VPC, Security, EC2). [cite: 18] \* variables.tf: Declara todas as variáveis de entrada do projeto (ex: região, CIDRs, URLs de repositórios, configurações de banco de dados, tipos de instância para os servidores de monitoramento). [cite: 19] \* terraform.tfvars: Fornece os valores específicos para as variáveis de entrada para este ambiente de implantação. [cite: 20] \* outputs.tf: Define quais informações serão exibidas após a aplicação do Terraform (ex: IP público do frontend, endpoint do RDS, IP do servidor Wazuh). [cite: 21] \* provider.tf: Configura o provedor AWS, especificando a região e o perfil de credenciais. [cite: 22] \* database\_postgres.tf: Define os recursos para o banco de dados RDS PostgreSQL da aplicação e o armazenamento seguro de suas credenciais no AWS Secrets Manager. [cite: 23]

### 4.3. Módulos Locais

* **Módulo VPC (modules/vpc/)**: Responsável pela criação da Virtual Private Cloud (VPC), subnets públicas e privadas em múltiplas Zonas de Disponibilidade, Internet Gateway (IGW), NAT Gateway (com Elastic IP) e as tabelas de rotas associadas para garantir a conectividade correta. [cite: 24]
* **Módulo Security (modules/security/)**: Gerencia os Security Groups. Cria grupos para o frontend (permitindo tráfego HTTP/S público), para o backend (permitindo tráfego na porta da API apenas a partir do security group do frontend), para o banco de dados RDS (permitindo acesso apenas do backend SG) e para o servidor Wazuh. [cite: 25]
* **Módulo EC2 (modules/ec2/)**: Define um módulo genérico para criar instâncias EC2. É chamado múltiplas vezes pelo módulo raiz para as instâncias do frontend, backend e servidor Wazuh, cada uma com suas configurações específicas. [cite: 27]

## 5. Componentes da Infraestrutura e Decisões de Design

### 5.1. Rede (VPC)

Uma VPC customizada (10.0.0.0/16) foi projetada com: [cite: 28] \* **Subnets Públicas**: Para recursos que precisam de acesso direto à internet, como a instância EC2 do frontend e o servidor Wazuh (para acesso ao dashboard, restringido por IP). [cite: 28] Estas subnets têm rotas para o Internet Gateway. [cite: 29] \* **Subnets Privadas**: Para recursos que não devem ser diretamente acessíveis pela internet, como a instância EC2 do backend e o banco de dados RDS. [cite: 29] Estas subnets têm rotas para um NAT Gateway para permitir acesso de saída à internet. [cite: 30] \* **Zonas de Disponibilidade (AZs)**: As subnets foram distribuídas em múltiplas AZs (ex: us-east-2a, us-east-2b) para aumentar a resiliência. [cite: 31]

### 5.2. Segurança

* **Security Groups**: [cite: 32]
  + **Frontend SG**: Permite tráfego de entrada nas portas 80 (HTTP) e 443 (HTTPS) de qualquer origem (0.0.0.0/0). [cite: 32]
  + **Backend SG**: Permite tráfego de entrada na porta da aplicação backend (ex: 8000) somente a partir do Security Group do frontend. [cite: 33] Isso isola o backend da internet direta. [cite: 34]
  + **RDS SG**: Permite tráfego de entrada na porta do PostgreSQL (5432) somente a partir do Security Group do backend. [cite: 34]
  + **Wazuh Server SG**: Permite tráfego de entrada na porta 443 (HTTPS para o dashboard) de um IP específico (ex: my\_home\_ip\_cidr), e nas portas 1514/TCP e 1515/TCP (comunicação dos agentes) a partir dos Security Groups do frontend e backend.
  + **Acesso SSH**: As regras de SSH (porta 22) foram removidas dos Security Groups das instâncias EC2 em favor do uso do AWS Systems Manager Session Manager para acesso seguro. [cite: 35]
* **IAM (Identity and Access Management)**: [cite: 36]
  + Uma IAM Role (ec2\_role) foi criada para as instâncias EC2. [cite: 36]
  + Políticas anexadas a esta role incluem: AmazonSSMManagedInstanceCore (para Session Manager)[cite: 37], permissão para ler segredos do AWS Secrets Manager (credenciais do banco de dados da aplicação e chave SSH do GitHub para deploy do backend)[cite: 38], e opcionalmente AmazonS3ReadOnlyAccess. [cite: 38]
  + Um IAM Instance Profile (ec2\_profile) foi criado para associar a role às instâncias. [cite: 39]

### 5.3. Instâncias de Cômputo (EC2)

* **Instância Frontend**: Localizada em uma subnet pública[cite: 40], associada ao frontend\_sg[cite: 41], executa a aplicação Next.js via user\_data\_frontend.sh. [cite: 41]
* **Instância Backend**: Localizada em uma subnet privada[cite: 42], associada ao backend\_sg[cite: 42], executa a API FastAPI com PostgreSQL via user\_data\_backend.sh. [cite: 43]
* **Instância Wazuh Server**: Localizada em uma subnet pública (para acesso ao dashboard, com SG restrito), executa o Wazuh “all-in-one” via user\_data\_wazuh\_server.sh. Utiliza um tipo de instância adequado (ex: t3.medium).

### 5.4. Banco de Dados (RDS PostgreSQL)

* Uma instância RDS PostgreSQL (ex: db.t3.micro, versão 16.x) é provisionada para a aplicação. [cite: 44]
* Localizada nas subnets privadas utilizando um aws\_db\_subnet\_group. [cite: 44]
* Protegida por um Security Group (rds\_sg) que permite acesso apenas da instância backend. [cite: 45]
* Configurada para não ser publicamente acessível e com armazenamento criptografado. [cite: 46]

### 5.5. Gerenciamento de Segredos (AWS Secrets Manager)

* As credenciais do banco de dados RDS PostgreSQL da aplicação são geradas aleatoriamente e armazenadas de forma segura no AWS Secrets Manager. [cite: 47]
* A chave SSH privada para deploy do repositório backend (Deploy Key do GitHub) também é armazenada no Secrets Manager.
* As instâncias EC2 (especificamente o backend) utilizam sua IAM Role para buscar essas credenciais em tempo de execução. [cite: 48]

### 5.6. Acesso às Instâncias (AWS Systems Manager Session Manager)

Para acesso seguro às instâncias EC2 (frontend, backend e Wazuh server) sem expor a porta SSH (22), o AWS Systems Manager Session Manager foi configurado. [cite: 49] Isso é habilitado pela IAM Role com a política AmazonSSMManagedInstanceCore. [cite: 50]

### 5.7. Monitoramento de Segurança e Detecção de Intrusão (Wazuh)

* **Wazuh Server**: Uma instância EC2 dedicada hospeda a instalação “all-in-one” do Wazuh (Manager, Indexer, Dashboard).
* **Agentes Wazuh**: Instalados nas instâncias EC2 do frontend e backend através de seus respectivos scripts user\_data. Os agentes são configurados para reportar ao IP privado do servidor Wazuh.
* **Comunicação**: Os Security Groups foram configurados para permitir que os agentes enviem dados para o servidor Wazuh nas portas 1514/TCP e 1515/TCP. O acesso ao dashboard do Wazuh (porta 443) é restrito ao IP do administrador.

## 6. Scripts de User Data

Scripts de inicialização (user\_data) são usados para configurar as instâncias EC2 no primeiro boot: [cite: 51] \* **user\_data\_frontend.sh**: Instala Node.js, Git, Nginx[cite: 51], clona o repositório frontend[cite: 52], instala dependências, realiza o build do Next.js[cite: 52], configura Nginx como proxy reverso[cite: 53], inicia a aplicação com pm2[cite: 54], e instala/configura o agente Wazuh. \* **user\_data\_backend.sh**: Instala Python, pip, venv, Git, jq, openssh-client[cite: 54], configura a chave SSH para clonar o repositório backend (branch aws-deploy)[cite: 55], cria ambiente virtual, instala dependências (incluindo FastAPI, Uvicorn, Gunicorn, psycopg2-binary, Alembic)[cite: 56], busca credenciais do banco de dados do Secrets Manager[cite: 57], executa migrações do Alembic[cite: 58], cria e inicia serviço systemd para a API[cite: 58], e instala/configura o agente Wazuh. \* **user\_data\_wazuh\_server.sh**: Baixa e executa o script de instalação “all-in-one” do Wazuh.

## 7. Versionamento e Modularização

* **Modularização**: O código Terraform foi dividido em módulos locais (VPC, Security, EC2) para melhor organização, reutilização e manutenibilidade. [cite: 59]
* **Versionamento**: O projeto de infraestrutura é gerenciado usando Git. [cite: 60] Foi criada uma branch aws-deploy no repositório da API para conter as configurações específicas da AWS. [cite: 61] O Terraform também especifica as versões requeridas dos provedores. [cite: 62]

## 8. Próximos Passos: Deploy com Terraform

Com toda a configuração revisada e considerada correta, os próximos passos são executar os comandos Terraform para provisionar a infraestrutura na AWS: [cite: 63] \* terraform init -upgrade: Para inicializar o diretório de trabalho, baixar os provedores e módulos. [cite: 63] \* terraform validate: Para verificar a sintaxe e consistência da configuração. [cite: 64] \* terraform plan: Para revisar as ações que o Terraform executará. [cite: 65] \* terraform apply: Para aplicar as configurações e criar os recursos na AWS. [cite: 66]

Após o apply, serão realizados testes para verificar a funcionalidade do frontend, do backend, a comunicação entre eles, o acesso ao banco de dados, o funcionamento do Wazuh (dashboard e registro de agentes) e o acesso às instâncias via Session Manager. [cite: 67]