Tutorial de Implementação da Infraestrutura

Este tutorial fornece instruções detalhadas para implementar a infraestrutura do projeto DevOps utilizando Terraform. Seguindo este guia, você será capaz de provisionar todos os recursos necessários na AWS de forma automatizada e consistente.

Pré-requisitos

Antes de começar, certifique-se de que você possui:

1. **Conta AWS** com permissões adequadas para criar recursos
2. **AWS CLI** instalado e configurado com suas credenciais
3. **Terraform** (versão 1.0.0 ou superior) instalado
4. **Git** instalado para clonar o repositório

Configuração Inicial

1. Clone o Repositório

Primeiro, clone o repositório do projeto:

```
git clone https://github.com/seu-usuario/devops-projeto.git
cd devops-projeto/terraform
```

2. Configure as Credenciais da AWS

Certifique-se de que suas credenciais da AWS estão configuradas. Você pode fazer isso de várias maneiras:

Opção 1: Usando o AWS CLI

```
aws configure
```

Opção 2: Variáveis de Ambiente

```
export AWS_ACCESS_KEY_ID="sua-access-key"
export AWS_SECRET_ACCESS_KEY="sua-secret-key"
export AWS_DEFAULT_REGION="us-east-1"
```

Opção 3: Arquivo de Credenciais Crie ou edite o arquivo `~/.aws/credentials`:

```
[default]
aws_access_key_id = sua-access-key
aws_secret_access_key = sua-secret-key
```

3. Personalize as Variáveis

Revise e personalize as variáveis no arquivo `variables.tf` de acordo com suas necessidades. Você pode sobrescrever os valores padrão criando um arquivo `terraform.tfvars`:

```
aws_region     = "us-east-1"
project_name   = "meu-projeto-devops"
vpc_cidr       = "10.0.0.0/16"
public_subnets = ["10.0.1.0/24", "10.0.2.0/24"]
private_subnets = ["10.0.3.0/24", "10.0.4.0/24"]
availability_zones = ["us-east-1a", "us-east-1b"]
instance_type = "t3.medium"
min_instances = 2
max_instances = 10
db_instance_class = "db.t3.medium"
db_storage = 100
db_engine = "postgres"
db_engine_version = "13"
db_name = "appdb"
db_username = "dbadmin"
```

IMPORTANTE: Nunca armazene senhas ou informações sensíveis diretamente no arquivo `terraform.tfvars`. Para senhas e outros dados sensíveis, utilize o AWS Secrets Manager ou variáveis de ambiente.

Provisionamento da Infraestrutura

1. Inicialize o Terraform

Inicialize o Terraform para baixar os plugins necessários e configurar o backend:

```
terraform init
```

Para utilizar um backend remoto (recomendado para equipes), você pode configurar o S3:

```
terraform init -backend-config="bucket=meu-bucket-terraform" -backend-
config="key=devops-projeto/terraform.tfstate" -backend-config="region=us-east-1"
```

2. Valide a Configuração

Verifique se a configuração do Terraform está correta:

```
terraform validate
```

3. Crie um Plano de Execução

Gere um plano para visualizar as alterações que serão aplicadas:

```
terraform plan -out=tfplan
```

Revise cuidadosamente o plano para garantir que todas as alterações estão de acordo com o esperado.

4. Aplique as Alterações

Aplique o plano para criar a infraestrutura:

```
terraform apply tfplan
```

Alternativamente, você pode aplicar diretamente (será solicitada confirmação):

```
terraform apply
```

5. Verifique os Recursos Criados

Após a conclusão bem-sucedida, o Terraform exibirá os outputs definidos. Você também pode visualizá-los a qualquer momento com:

```
terraform output
```

Verifique os recursos criados no console da AWS para garantir que tudo foi provisionado corretamente.

Gerenciamento da Infraestrutura

Atualização da Infraestrutura

Para atualizar a infraestrutura após alterações nos arquivos Terraform:

```
terraform plan -out=tfplan
terraform apply tfplan
```

Destruição da Infraestrutura

Para destruir todos os recursos criados (use com cautela):

```
terraform destroy
```

Visualização do Estado

Para visualizar o estado atual da infraestrutura gerenciada pelo Terraform:

```
terraform state list
```

Para ver detalhes de um recurso específico:

```
terraform state show aws_vpc.main
```

Boas Práticas

1. **Versionamento**: Mantenha os arquivos Terraform em um sistema de controle de versão.
2. **Backend Remoto**: Utilize um backend remoto (como S3) para armazenar o estado do Terraform, especialmente em equipes.
3. **Módulos**: Para infraestruturas mais complexas, considere organizar o código em módulos reutilizáveis.
4. **Variáveis de Ambiente**: Use variáveis de ambiente para credenciais e informações sensíveis.
5. **Workspaces**: Utilize workspaces do Terraform para gerenciar múltiplos ambientes (dev, staging, prod).

Solução de Problemas

Erros de Permissão

Se encontrar erros de permissão, verifique:

- Se suas credenciais AWS estão configuradas corretamente
- Se o usuário/role tem as permissões necessárias para criar os recursos

Erros de Dependência

Se recursos não puderem ser criados devido a dependências:

- Verifique se todos os recursos dependentes estão sendo criados na ordem correta
- Adicione dependências explícitas usando `depends_on` se necessário

Erros de Timeout

Se operações expirarem:

- Aumente os timeouts nos recursos relevantes
- Verifique se há problemas de conectividade com a AWS

Expansão para CD e Containerização

Após implementar a infraestrutura básica, você pode expandir o projeto para incluir:

1. **Pipeline de CD**: Configure o GitHub Actions para deploy automático na infraestrutura provisionada.
2. **Containerização**: Utilize Docker para containerizar a aplicação e implantá-la no ECS.
3. **Monitoramento**: Configure o CloudWatch para monitorar a infraestrutura e aplicação.

Para mais detalhes sobre essas expansões, consulte a documentação completa do projeto.

Conclusão

Seguindo este tutorial, você implementou com sucesso a infraestrutura necessária para o projeto DevOps utilizando Terraform. Esta abordagem de Infraestrutura como Código permite que você provisione, atualize e gerencie recursos de forma consistente e automatizada, seguindo as melhores práticas de DevOps.