

# ENTREGA FINAL

## 1. Escolha da Ferramenta de IaC

**Terraform** foi escolhido, em vez de **CloudFormation**, por várias razões:

- **Multi-Provider Support:** O Terraform é perfeito em relação ao provedor, permitindo que você gerencie recursos na AWS, Azure, Google Cloud, entre outros, usando a mesma linguagem.
- **Estado Declarativo:** O Terraform mantém um estado do ambiente, permitindo visualizações claras e atualizações incrementais.
- **Módulos Reutilizáveis:** Os módulos do Terraform permitem uma organização e reutilização facilitadas do código.

## 2. Recursos Criados na Configuração

### a. Cluster ECS

Um cluster ECS ( `aws_ecs_cluster` ), que é o agrupamento lógico de serviços e tarefas que você executa usando o Amazon ECS (Elastic Container Service). Este cluster é o ponto de partida para executar contêineres.

### b. Task Definition

A definição da tarefa ( `aws_ecs_task_definition` ) especifica o contêiner que será executado, incluindo a imagem Docker, as variáveis de ambiente e os recursos (CPU e memória) necessários. Aqui eu indiquei a imagem do meu aplicativo.

```
resource "aws_ecs_task_definition" "task_definition" {
  family                = "meu-app"
  requires_compatibilities = ["FARGATE"] # Indica que a tarefa usa Fargate
  network_mode          = "awsvpc"      # Necessário para
```

## Fargate

```
container_definitions = jsonencode([{\n  name      = "meu-container"\n  image      = "minha-imagem" # URL da imagem Docker\n  memory     = "512" # Memória alocada\n  cpu        = "256" # CPU alocada\n  essential = true\n\n  portMappings = [{\n    containerPort = 80 # Porta do contêiner\n    hostPort      = 80 # Para Fargate, o hostPort é sempre\n    # igual ao containerPort\n    protocol      = "tcp"\n  }]\n}])\n}
```

## c. Load Balancer

Eu configurei um Load Balancer (`aws_lb`), que serve para distribuir o tráfego entre instâncias ou serviços (neste caso, contêineres do ECS), garantindo alta disponibilidade e distribuição de carga.

- O Load Balancer também pode ser usado para monitorar a saúde dos contêineres, retirando do tráfego aqueles que não respondem conforme esperado.

```
resource "aws_lb" "application_load_balancer" {\n  name            = "meu-load-balancer"\n  internal        = false\n  load_balancer_type = "application"\n  security_groups = [aws_security_group.lb_security_group.\n  id]\n  subnets        = [\n    "subnet-XXXXXXX",\n  ]\n}
```

```
    "subnet-YYYYYYYYY"
  ]
}
```

## d. Serviço ECS

O serviço ECS ( `aws_ecs_service` ) é responsável por garantir que uma quantidade especificada de tarefas esteja sempre em execução. Isso significa que se algum contêiner falhar, o serviço ECS iniciará automaticamente novas instâncias.

- Foi configurado o serviço para utilizar o Load Balancer recém-criado:

```
resource "aws_ecs_service" "service" {
  name           = "meu-servico-ecs"
  cluster        = aws_ecs_cluster.cluster.id
  task_definition = aws_ecs_task_definition.task_definition.arn
  desired_count  = 1
  launch_type    = "FARGATE" # Usa Fargate para gerenciamento serverless

  network_configuration {
    subnets          = ["subnet-VALID01", "subnet-VALID02"]
    security_groups   = [aws_security_group.lb_security_group.id]
    assign_public_ip  = "ENABLED" # Atribui IPs públicos para acessar o serviço
  }

  load_balancer {
    target_group_arn = aws_lb_target_group.target_group.arn
    container_name   = "meu-container"
    container_port    = 80
  }
}
```

### 3. Por que usar Amazon ECS em vez de EC2?

Aqui estão algumas razões pelas quais eu optei pelo Amazon ECS (especificamente Fargate) em vez de instâncias EC2 tradicionais:

- **Gerenciamento de Contêineres:** O ECS é otimizado para executar e gerenciar contêineres. Ele permite que você empacote seu aplicativo em contêineres e escale facilmente com base na demanda.
- **Sem Gerenciamento de Infraestrutura:** Ao optar pelo Amazon ECS, especialmente ao utilizar o Fargate, você não precisa se preocupar com o gerenciamento das instâncias subjacentes. O Fargate cuida disso para você, permitindo que você se concentre no código e na lógica de negócios, em vez de na infraestrutura. Isso significa não ter que provisionar, monitorar ou gerenciar servidores EC2 manualmente.
- **Escalabilidade Automática:** O ECS permite escalabilidade automatizada com base na demanda da aplicação. Você pode definir políticas para que o ECS inicie automaticamente mais contêineres ou interrompa contêineres com menos tráfego, permitindo que você otimize custos e recursos.
- **Erros e Recuperação:** O ECS proporciona um gerenciamento mais robusto de contêineres. Se um contêiner falhar, o ECS pode reiniciá-lo automaticamente, garantindo que a quantidade desejada de instâncias esteja sempre ativa e funcional.
- **Integração com Outros Serviços da AWS:** O ECS se integra facilmente a outros serviços da AWS, como o CloudWatch para monitoramento, o IAM para controle de acesso e o Route 53 para gerenciamento de DNS. Essa integração facilita a criação de uma arquitetura de microserviços altamente confiável e escalável.
- **Implantação Rápida e consistente:** Com a definição do serviço e da tarefa no ECS, você pode implantar atualizações de contêiner rapidamente e de maneira consistente. Isso reduz a janela de tempo para colocar novas versões de software em produção.
- **Custo Eficiente:** Quando você utiliza o Fargate com ECS, você paga apenas pelos recursos que usa durante a execução dos contêineres, sem necessidade de pagar por instâncias EC2 o tempo todo. Isso pode reduzir

significativamente os custos, pois você evita a cobrança de capacidade subutilizada.

## 4. Resumo do Código Criado

- **Estrutura Modularizada:** O código foi estruturado para separar diferentes componentes da infraestrutura, como a definição da rede, o cluster ECS, a task definition e o equilíbrio de carga. Isso melhora a legibilidade e a manutenção do código.
- **Configurações de Rede:** Configurações de VPC, sub-redes e grupos de segurança foram criadas para isolar e proteger os recursos do ECS, garantindo que suas instâncias apenas aceitem tráfego de fontes autorizadas.
- **Deploy Visualizável e Reversível:** O Terraform fornece o comando `terraform plan`, que mostra quais mudanças serão aplicadas antes de fazer qualquer modificação no ambiente. Isso reduz o risco de erros durante o provisionamento e facilita a reversibilidade, caso você precise voltar a uma configuração anterior.
- **Configurabilidade:** O código é altamente configurável, permitindo que você ajuste rapidamente parâmetros como o número de instâncias, a imagem do contêiner, a configuração de CPU e memória e as portas expostas.

## 5. Considerações Finais

Ao optar pela implementação de microserviços com o Amazon ECS, você está garantindo uma estrutura mais moderna, escalável e menos dependente de hardware específico. O ECS fornece um ambiente capaz de lidar com aplicações que exigem flexibilidade e resiliência, ajudando a manter a eficiência operacional.

Uma vez que seu código esteja em execução, monitore continuamente utilizando o AWS CloudWatch e adapte sua infraestrutura conforme necessário para otimizar a performance e os custos.

Essas práticas garantirão que você utilize ao máximo o potencial do Amazon ECS e da AWS, permitindo que seus aplicativos sejam altamente disponíveis, escaláveis e de fácil gestão.

Retorno do terraform apply e exibindo que o código está versionado no cloudshell;

```
aws
Services
[Alt+S]
EC2 S3 IAM Billing and Cost Management CodeCommit CodePipeline CodeBuild Cloud9 CloudShell
CloudShell
us-east-1 +
Actions

# aws_ecs_service.service will be created
resource "aws_ecs_service" "service" {
  cluster = "arn:aws:ecs:us-east-1:211125683276:cluster/meu-cluster-ecs"
  deployment_maximum_percent = 200
  deployment_minimum_healthy_percent = 100
  desired_count = 1
  enable_ec2_managed_tags = false
  enable_execute_command = false
  iam_role = (known after apply)
  id = (known after apply)
  launch_type = (known after apply)
  name = "meu-service-ecs"
  platform_version = (known after apply)
  scheduling_strategy = "REPLICA"
  tags_all = (known after apply)
  task_definition = "arn:aws:ecs:us-east-1:211125683276:task-definition/minha-task-definition:4"
  triggers = (known after apply)
  wait_for_steady_state = false
}

load_balancer {
  container_name = "meu-container"
  container_port = 80
  target_group_arn = "arn:aws:elasticloadbalancing:us-east-1:211125683276:targetgroup/meu-target-group/b73e455e90bde6"
}

network_configuration {
  assign_public_ip = false
  security_groups = [
    "sg-8c2358872e4d7ec2c",
  ]
  subnets = [
    "subnet-0b0d0c0e62673d8f",
    "subnet-0f7307281e5d6f317",
  ]
}
}

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
Enter a value: yes

aws_ecs_service.service: Creating...
aws_ecs_service.service: Creation complete after 1s [id=arn:aws:ecs:us-east-1:211125683276:service/meu-cluster-ecs/meu-service-ecs]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[cloudshell-user@ip-10-138-190-241 terraform-codebuild]$ terraform state list
aws_codebuild_project.project
aws_ecs_cluster.cluster
aws_ecs_service.service
aws_ecs_task_definition.task_definition
aws_iam_role.codebuild_role
aws_iam_role.codepipeline_role
aws_iam_role_policy_attachment.codebuild_policy_attachment
aws_iam_role_policy_attachment.codepipeline_policy_attachment
aws_lb.load_balancer
aws_lb_listener.listener
aws_lb_target_group.target_group
aws_s3_bucket.artifacts
aws_security_group.lb_security_group
```

```
[cloudshell-user@ip-10-138-190-241 terraform-codebuild]$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .terraform.lock.hcl

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .terraform/
        terraform.tfstate
        terraform.tfstate.backup

no changes added to commit (use "git add" and/or "git commit -a")
[cloudshell-user@ip-10-138-190-241 terraform-codebuild]$
```

## 1. Criação do Bucket S3 para Armazenar Artefatos

O bucket S3 ( `aws_s3_bucket` ) é um repositório de armazenamento na nuvem que você está criando para armazenar os artefatos gerados pelo processo de build no AWS CodeBuild. Aqui está a parte do código relevante:

hcl

```
# Criação do bucket do S3 para armazenar os artefatos
resource "aws_s3_bucket" "artifacts" {
  bucket = "meu-bucket-de-artefatos"
}
```

- **Nome do Bucket:** O nome do bucket precisa ser único globalmente; portanto, certifique-se de que "meu-bucket-de-artefatos" não seja utilizado por outra conta AWS.
- **Objetivo:** Este bucket será utilizado para armazenar artefatos gerados durante a execução do build, como binários, pacotes ou qualquer outro arquivo que você deseja que esteja disponível após a execução do CodeBuild.

## 2. Criação do Projeto do CodeBuild

O projeto do CodeBuild ( `aws_codebuild_project` ) é definido para compilar seu código e executar testes. Aqui está a parte do código que gerencia isso:

hcl

```
# Criação do projeto do CodeBuild
resource "aws_codebuild_project" "project" {
  name           = "meu-projeto-codebuild"
  description    = "Projeto do CodeBuild para o meu aplicativo"
  service_role   = aws_iam_role.codebuild_role.arn

  artifacts {
    type = "NO_ARTIFACTS"
  }

  environment {
    compute_type = "BUILD_GENERAL1_SMALL"
    image        = "aws/codebuild/standard:4.0"
    type         = "LINUX_CONTAINER"
  }

  source {
    type          = "GITHUB"
    location      = "https://github.com/23Ant/terraform-codebuild.git"
    git_clone_depth = 1
  }
}
```

## Explicação dos Componentes:

- **name:** Nome do projeto do CodeBuild, neste caso, `"meu-projeto-codebuild"`.
- **description:** Uma breve descrição sobre o que este projeto faz.
- **service\_role:** O ARN da role do IAM que permissões o CodeBuild pode usar para acessar outros serviços da AWS, como o S3, por exemplo.
- **artifacts:** Especifica o tipo de artefatos gerados pelo build. Aqui, está definido como `"NO_ARTIFACTS"` porque, neste caso, não há artefatos sendo gerados, mas



you can alter this in the future to include artifacts that you want to store in the S3 bucket. If you want to store build output, you would change it to something like:

```
artifacts {
  type = "S3"
  location = aws_s3_bucket.artifacts.bucket
  packaging = "ZIP"
  path = "artefatos/"
}
```

- **environment:** Define the environment where the build will occur.
  - **compute\_type:** The type of computation that you are using. `"BUILD_GENERAL1_SMALL"` is a smaller specification.
  - **image:** The Docker image to be used for the build. `"aws/codebuild/standard:4.0"` corresponds to a standardized version of the build tool.
  - **type:** The type of environment; in this case, we are using a Linux container.
- **source:** Specifies the source of the code to be built.
  - **type:** The type of repository of origin; here, you are using GitHub.
  - **location:** The URL of the GitHub repository that contains the source code.
  - **git\_clone\_depth:** Specifies the depth of the clone. A value of `1` indicates that you want only the latest version of the repository.

### 3. Criação da Role IAM para o CodeBuild

In addition to the CodeBuild project and the S3 bucket, you are also creating an IAM role that CodeBuild will use to obtain the necessary permissions to operate. Here is the part of the code:

hcl

```

# Criação da role do IAM para o CodeBuild
resource "aws_iam_role" "codebuild_role" {
  name = "codebuild-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Principal = {
          Service = "codebuild.amazonaws.com"
        }
      }
    ]
  })
}

```

## Etapas criadas dentro da AWS

Feito a estrutura, estarei mostrando agora o ECS criado.

Amazon Elastic Container Service > Clusters > meu-cluster-ecs > Services

## meu-cluster-ecs

[Refresh](#) [Update cluster](#) [Delete cluster](#)

### Cluster overview

ARN am:aws:ecs:us-east-1:211125603276:cluster/meu-cluster-ecs	Status <span>Active</span>	CloudWatch monitoring <span>Default</span>	Registered container instances -
Services Draining -	Active 1	Tasks Pending -	Running -
Encryption Managed storage -	Fargate ephemeral storage -		

**Services (1)** [Info](#)

[Refresh](#) [Manage tags](#) [Update](#) [Delete service](#) [Create](#)

Filter services by value

Filter launch type: Any launch type

Filter service type: Any service type

Service name	ARN	Status	Service type	Deployments and tasks	Last deployment	Task definition	L...
meu-service-ecs	am:aws:ec...	<span>Active</span>	REPLICA	0/1 Tasks running	In progress	minha-task-defin...	EC2

No LOADBALANCE a vpc, subnet definidas e criadas.

EC2 > Load balancers > meu-load-balancer

## meu-load-balancer

[Refresh](#) [Actions](#)

### Details

Load balancer type Application	Status <span>Active</span>	VPC <a href="#">vpc-093b125eb5f5fa24b</a>	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z3SSXD0TRQ7X7K	Availability Zones <a href="#">subnet-00d9bd50e8e3673d8</a> us-east-1e (use1-az3) <a href="#">subnet-0f7307281e55df317</a> us-east-1c (use1-az1)	Date created August 9, 2024, 16:44 (UTC-03:00)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:211125603276:loadbalancer/app/meu-load-balancer/bed797cbd530234		DNS name <a href="#">Info</a> <a href="#">meu-load-balancer-609237053.us-east-1.elb.amazonaws.com</a> (A Record)	

[Listeners and rules](#) [Network mapping](#) [Resource map - new](#) [Security](#) [Monitoring](#) [Integrations](#) [Attributes](#) [Tags](#)

### Listeners and rules (1)

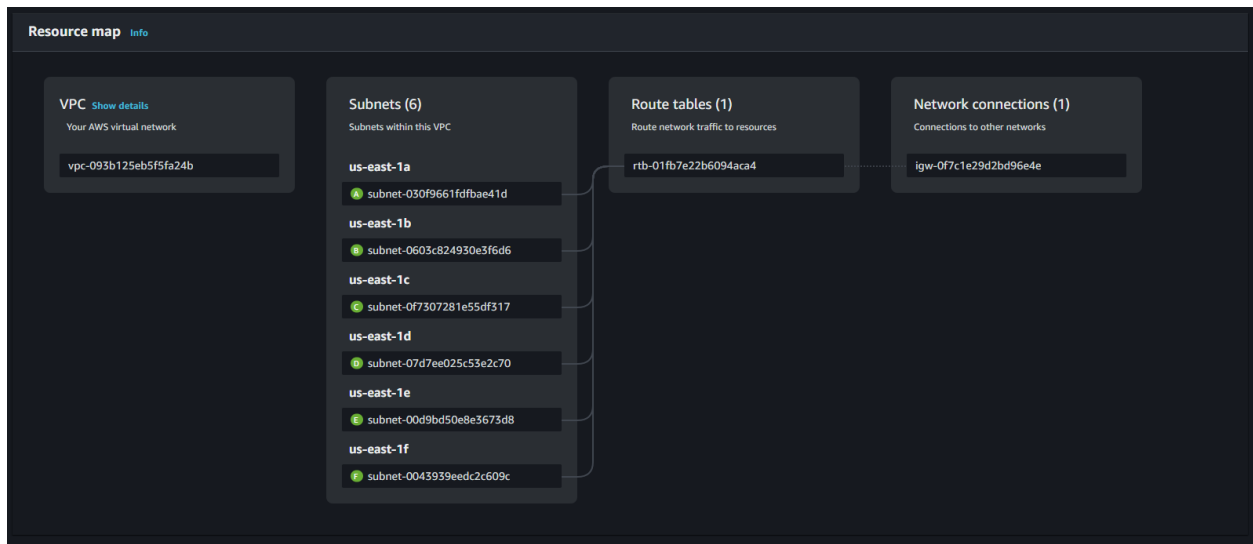
[Refresh](#) [Manage rules](#) [Manage listener](#) [Add listener](#)

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS
<a href="#">HTTP:80</a>	Forward to target group <ul style="list-style-type: none"> <li><a href="#">meu-target-group</a> 1 (100%)</li> <li>Target group stickiness: Off</li> </ul>	<a href="#">1 rule</a>	<a href="#">ARN</a>	Not applicable	Not applicable	Not applicable

Resource Map da nossa rede;



Dentro do IAM, Roles geradas.

The screenshot shows the AWS IAM Roles page. It displays a list of roles with the following columns: Role name, Trusted entities, and Last activity.

<input type="checkbox"/>	Role name	Trusted entities	Last activity
<input type="checkbox"/>	<a href="#">AWSCodePipelineServiceRole-us-east-1-my-terraform-build</a>	AWS Service: codepipeline	Yesterday
<input type="checkbox"/>	<a href="#">AWSCodePipelineServiceRole-us-east-1-my-terraform-pipeline</a>	AWS Service: codepipeline	-

Policies de permissões codebuild, geradas.

[IAM](#) > [Policies](#) > [AWSCodePipelineServiceRole-us-east-1-my-terraform-build](#)

# AWSCodePipelineServiceRole-us-east-1-my-terraform-build

Info

EditDelete

Policy used in trust relationship with CodePipeline

Policy details

Type

Customer managed

Creation time

August 08, 2024, 14:59 (UTC-03:00)

Edited time

August 08, 2024, 14:59 (UTC-03:00)

ARN

arn:aws:iam::211125603276:policy/service-role/AWSCodePipelineServiceRole-us-east-1-my-terraform-build

Permissions

Entities attached

Tags

Policy versions (1)

Access Advisor

Permissions defined in this policy

Info

EditSummaryJSON

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Q Search

Allow (25 of 420 services) 

Show remaining 395 services < 1 2 >

Service	Access level	Resource	Request condition
<a href="#">AppConfig</a>	Limited: Read, Write	All resources	None
<a href="#">Cloud Control API</a>	Full access	All resources	None
<a href="#">CloudFormation</a>	Full access	All resources	None
<a href="#">CloudWatch</a>	Full access	All resources	None
<a href="#">CodeBuild</a>	Limited: Read, Write	All resources	None

Service	Access level	Resource	Request condition
<a href="#">AppConfig</a>	Limited: Read, Write	All resources	None
<a href="#">Cloud Control API</a>	Full access	All resources	None
<a href="#">CloudFormation</a>	Full access	All resources	None
<a href="#">CloudWatch</a>	Full access	All resources	None
<a href="#">CodeBuild</a>	Limited: Read, Write	All resources	None
<a href="#">CodeCommit</a>	Limited: Read, Write	All resources	None
<a href="#">CodeDeploy</a>	Limited: List, Write	All resources	None
<a href="#">CodeStar Connections</a>	Limited: Read	All resources	None
<a href="#">Device Farm</a>	Limited: List, Read, Write	All resources	None
<a href="#">EC2</a>	Full access	All resources	None
<a href="#">EC2 Auto Scaling</a>	Full access	All resources	None
<a href="#">Elastic Beanstalk</a>	Full access	All resources	None
<a href="#">Elastic Container Registry</a>	Limited: List	All resources	None
<a href="#">Elastic Container Service</a>	Full access	All resources	None
<a href="#">ELB</a>	Full access	All resources	None
<a href="#">ELB v2</a>	Full access	All resources	None
<a href="#">IAM</a>	Limited: Write	All resources	iam:PassedToService = cloudformation.amazonaws.com,elasticbeanstalk.amazonaws.com,ec2.amazonaws.com,ecs-tasks.amazonaws.com(if Exists)
<a href="#">Lambda</a>	Limited: List, Write	All resources	None
<a href="#">OpsWorks</a>	Limited: List, Write	All resources	None

Policies de permissoes geradas codepipeline.

ENTREGA FINAL

13

[IAM](#) > [Policies](#) > [AWSCodePipelineServiceRole-us-east-1-my-terraform-pipeline](#)

### AWSCodePipelineServiceRole-us-east-1-my-terraform-pipeline Info

[Edit](#)[Delete](#)

Policy used in trust relationship with CodePipeline

Policy details

Type Customer managed	Creation time August 08, 2024, 14:52 (UTC-03:00)	Edited time August 08, 2024, 14:52 (UTC-03:00)	ARN arn:aws:iam::211125603276:policy/service-role/AWSCodePipelineServiceRole-us-east-1-my-terraform-pipeline
--------------------------	---	---	---

Permissions

Entities attached

Tags

Policy versions (1)

Access Advisor

Permissions defined in this policy Info

[Edit](#)[Summary](#)[JSON](#)

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Allow (25 of 420 services)

☒ Show remaining 395 services < 1 2 >

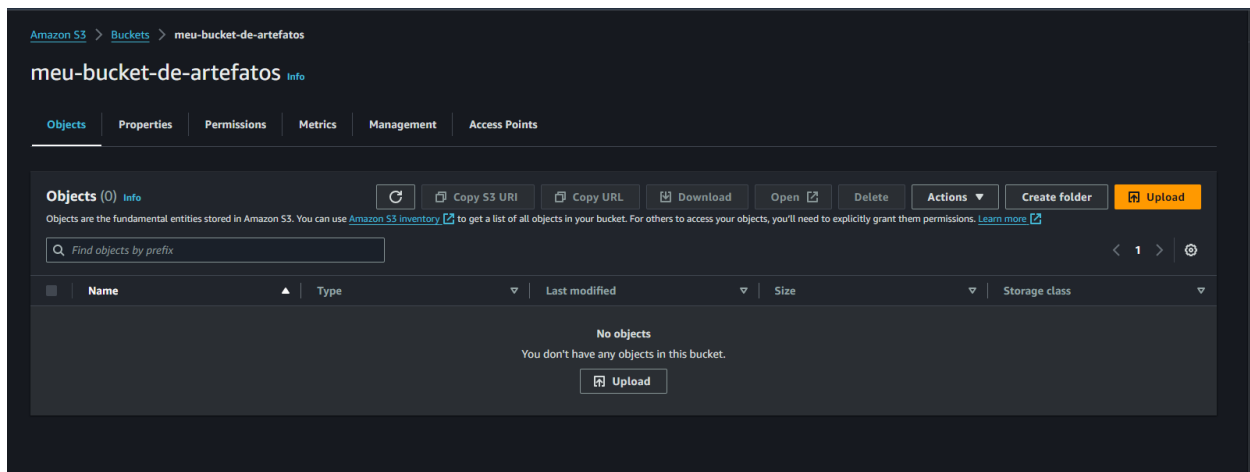
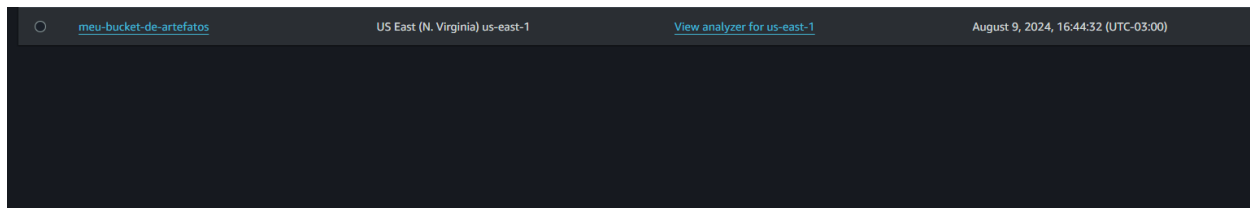
Service	Access level	Resource	Request condition
<a href="#">AppConfig</a>	Limited: Read, Write	All resources	None
<a href="#">Cloud Control API</a>	Full access	All resources	None
<a href="#">CloudFormation</a>	Full access	All resources	None
<a href="#">CloudWatch</a>	Full access	All resources	None
<a href="#">CodeBuild</a>	Limited: Read, Write	All resources	None

© 2024 Amazon Web Services, Inc. or its affiliates

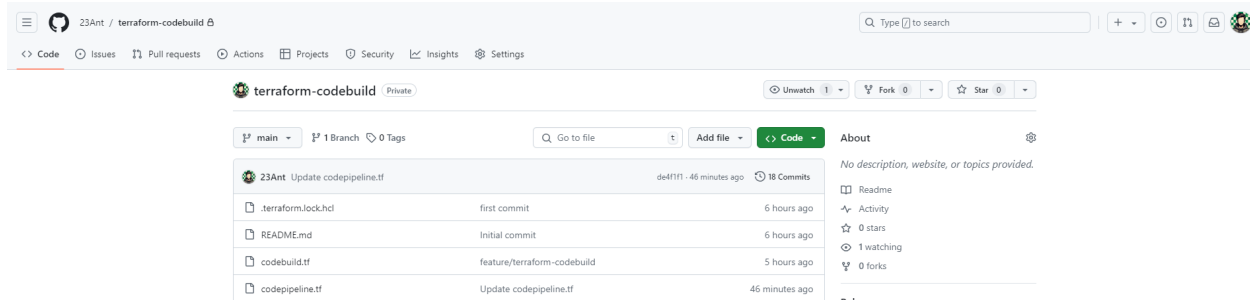
[Privacy](#)[Terms](#)[Cookie](#)

<a href="#">AppConfig</a>	Limited: Read, Write	All resources	None
<a href="#">Cloud Control API</a>	Full access	All resources	None
<a href="#">CloudFormation</a>	Full access	All resources	None
<a href="#">CloudWatch</a>	Full access	All resources	None
<a href="#">CodeBuild</a>	Limited: Read, Write	All resources	None
<a href="#">CodeCommit</a>	Limited: Read, Write	All resources	None
<a href="#">CodeDeploy</a>	Limited: List, Write	All resources	None
<a href="#">CodeStar Connections</a>	Limited: Read	All resources	None
<a href="#">Device Farm</a>	Limited: List, Read, Write	All resources	None
<a href="#">EC2</a>	Full access	All resources	None
<a href="#">EC2 Auto Scaling</a>	Full access	All resources	None
<a href="#">Elastic Beanstalk</a>	Full access	All resources	None
<a href="#">Elastic Container Registry</a>	Limited: List	All resources	None
<a href="#">Elastic Container Service</a>	Full access	All resources	None
<a href="#">ELB</a>	Full access	All resources	None
<a href="#">ELB v2</a>	Full access	All resources	None
<a href="#">IAM</a>	Limited: Write	All resources	iam:PassedToService = cloudformation.amazonaws.com,elasticbeanstalk.amazonaws.com,ec2.amazonaws.com,ecs-tasks.amazonaws.com(if Exists)
<a href="#">Lambda</a>	Limited: List, Write	All resources	None
<a href="#">OpsWorks</a>	Limited: List, Write	All resources	None
<a href="#">RDS</a>	Full access	All resources	None

Bucket criado no s3 pelo codedeploy.



Plataforma versionada dentro do github, dando git clone com url de desenvolvedor setada no development settings.



## Personal access tokens (classic)

[Generate new token](#)

[Revoke all](#)

Tokens you have generated that can be used to access the [GitHub API](#).

**terraform-codebuild** — `admin:enterprise`, `admin:gpg_key`, `admin:org`, `admin:org_hook`,  
`admin:public_key`, `admin:repo_hook`, `admin:ssh_signing_key`, `audit_log`, `codespace`, `copilot`, `delete:packages`, `delete_repo`, `gist`,  
`notifications`, `project`, `repo`, `user`, `workflow`, `write:discussion`, `write:packages`

Last used within the last week

[Delete](#)

**⚠ This token has no expiration date.**

CODIGO CRIADOS

## codebuild.tf

### Criação do projeto do CodeBuild

```
resource "aws_codebuild_project" "project" {
  name      = "meu-projeto-codebuild"
  description = "Projeto do CodeBuild para o meu aplicativo"
  service_role = aws_iam_role.codebuild_role.arn

  artifacts {
    type = "NO_ARTIFACTS"
  }

  environment {
    compute_type = "BUILD_GENERAL1_SMALL"
    image       = "aws/codebuild/standard:4.0"
    type        = "LINUX_CONTAINER"
  }

  source {
    type      = "GITHUB"
    location  = ""
    https://github.com/23Ant/terraform-codebuild.git
    git_clone_depth = 1
  }
}
```

### Criação da role do IAM para o CodeBuild

```
resource "aws_iam_role" "codebuild_role" {
  name = "codebuild-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
```



```
{  
  Action = "sts:AssumeRole"  
  Effect = "Allow"  
  Principal = {  
    Service = "  
    codebuild.amazonaws.com"  
  }  
}  
})  
}
```

## Anexação da política do IAM à role do CodeBuild

```
resource "aws_iam_role_policy_attachment" "codebuild_policy_attachment" {  
  policy_arn = "arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess"  
  role       = aws_iam_role.codebuild_role.name  
}
```

## Criação do bucket do S3 para armazenar os artefatos

```
resource "aws_s3_bucket" "artifacts" {  
  bucket = "meu-bucket-de-artefatos"  
}
```

---

## codepipeline.tf

```
provider "aws" {  
  region = "us-east-1"  
}
```

## Criação da role do IAM para o CodePipeline

```
resource "aws_iam_role" "codepipeline_role" {
  name = "codepipeline-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Principal = {
          Service = "
codepipeline.amazonaws.com"
        }
      }
    ]
  })
}
```

## Anexação da política do IAM à role do CodePipeline

```
resource "aws_iam_role_policy_attachment" "codepipeline_policy_attachment" {
  role      = aws_iam_role.codepipeline_role.name
  policy_arn = "arn:aws:iam::aws:policy/AWSCodePipeline_FullAccess"
}
```

## Criação do cluster do ECS

```
resource "aws_ecs_cluster" "cluster" {
  name = "meu-cluster-ecs"
}
```

## Criação da definição da tarefa do ECS

```

resource "aws_ecs_task_definition" "task_definition" {
  family      = "minha-task-definition"
  network_mode = "awsvpc"
  container_definitions = jsonencode([
  {
    name      = "meu-container"
    image     = "minha-imagem"
    cpu       = 256
    memory    = 512
    essential = true
    portMappings = [
    {
      containerPort = 80
      hostPort      = 80
      protocol      = "tcp"
    }
  ]
  }
  ])
}

```

## Criação do target group do load balancer

```

resource "aws_lb_target_group" "target_group" {
  name      = "meu-target-group"
  port      = 80
  protocol  = "HTTP"
  target_type = "ip"
  vpc_id    = "vpc-093b125eb5f5fa24b" # ID DA MINHA VPC

  health_check {
    path          = "/"
    interval      = 30
    timeout       = 5
    healthy_threshold = 2
    unhealthy_threshold = 2
  }
}

```

```
protocol      = "HTTP"
}
}
```

## Criação do load balancer

```
resource "aws_lb" "load_balancer" {
  name          = "meu-load-balancer"
  internal      = false
  load_balancer_type = "application"
  security_groups = [aws_security_group.lb_security_group.id]

  subnets      = [
    "subnet-00d9bd50e8e3673d8",
    "subnet-0f7307281e55df317"
  ] # IDs das minhas subnets
}
```

## Criação do listener do load balancer

```
resource "aws_lb_listener" "listener" {
  load_balancer_arn = aws_lb.load_balancer.arn
  port             = 80
  protocol         = "HTTP"

  default_action {
    type = "forward"
  }
  target_group_arn = aws_lb_target_group.target_group.arn
}
```

## Criação do serviço do ECS

```
resource "aws_ecs_service" "service" {
  name      = "meu-servico-ecs"
  cluster  = aws_ecs_cluster.cluster.id
}
```

```
task_definition = aws_ecs_task_definition.task_definition.arn
desired_count   = 1
```

## Configuração do load balancer

```
load_balancer {
  target_group_arn = aws_lb_target_group.target_group.arn
  container_name   = "meu-container"
  container_port   = 80
}
```

## Configuração da rede

```
network_configuration {
  subnets      = [
    "subnet-00d9bd50e8e3673d8",
    "subnet-0f7307281e55df317"
  ] # IDs das minhas subnets
  security_groups = [aws_security_group.lb_security_group.id]
  assign_public_ip = false # "ENABLED" para habilitar IPs públicos
}
```

## Outras configurações do serviço do ECS...

```
} #iria fazer a instalação do sql mas nao deu tempo
```

## Criação do grupo de segurança para o load balancer

```
resource "aws_security_group" "lb_security_group" {
  name_prefix = "lb-sg"
  vpc_id      = "vpc-093b125eb5f5fa24b" # ID da minha VPC
  ingress {
    from_port = 80
  }
}
```

```

to_port    = 80
protocol   = "tcp"
cidr_blocks = ["0.0.0.0/0"] # Permitindo acessos públicos na porta 80
}

egress {
  from_port = 0
  to_port   = 0
  protocol  = "-1" # Permitindo todo o tráfego de saída
  cidr_blocks = ["0.0.0.0/0"] # Permitindo a saída para qualquer IP
}
}

```

**resultado:**

```

[cloudshell-user@ip-10-138-190-241 terraform-codebuild]$ terraform state list
aws_codebuild_project.project
aws_ecs_cluster.cluster
aws_ecs_task_definition.task_definition
aws_iam_role.codebuild_role
aws_iam_role.codepipeline_role
aws_iam_role_policy_attachment.codebuild_policy_attachment
aws_iam_role_policy_attachment.codepipeline_policy_attachment
aws_lb.load_balancer
aws_lb_listener.listener
aws_lb_target_group.target_group
aws_s3_bucket.artifacts
aws_security_group.lb_security_group

```