Documentaçao Atividade WordPress Docker

Aluno: Thiago Sandre Trevisan

Documento criado a partir da atividade prática,

Sobre Docker na AWS





Arquitetura da Atividade proposta



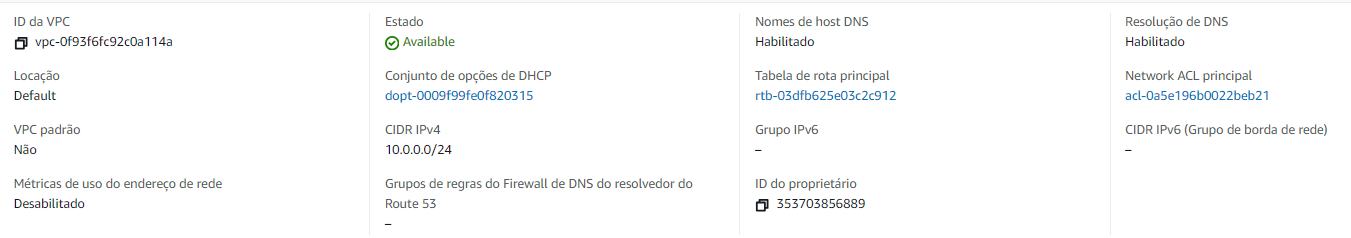
Etapas a serem feitas dentro da atividade



VPC

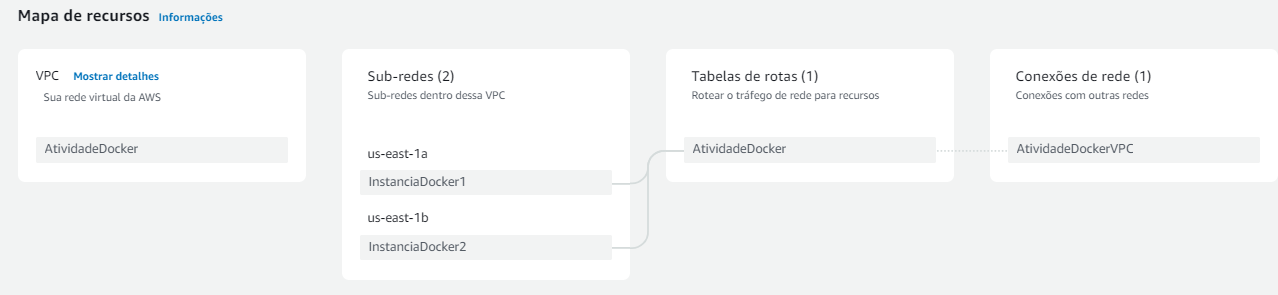


VPC



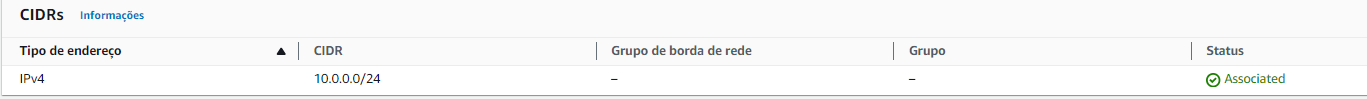
Vpc criada para a divisão de sub-nets e disponibilidade de internet para ambas, vinculada a uma Tabela de Rota e uma Network ACL.

Mapa Recursos



Mapa da VPC sobre como está configurada, onde mostra as sub-nets dentro da VPC (InstaciaDocker1, InstanciaDocker2) que estão vinculados a Tabela de Rotas para o trafego de rede para recursos que está vinculada a uma Network ACL.

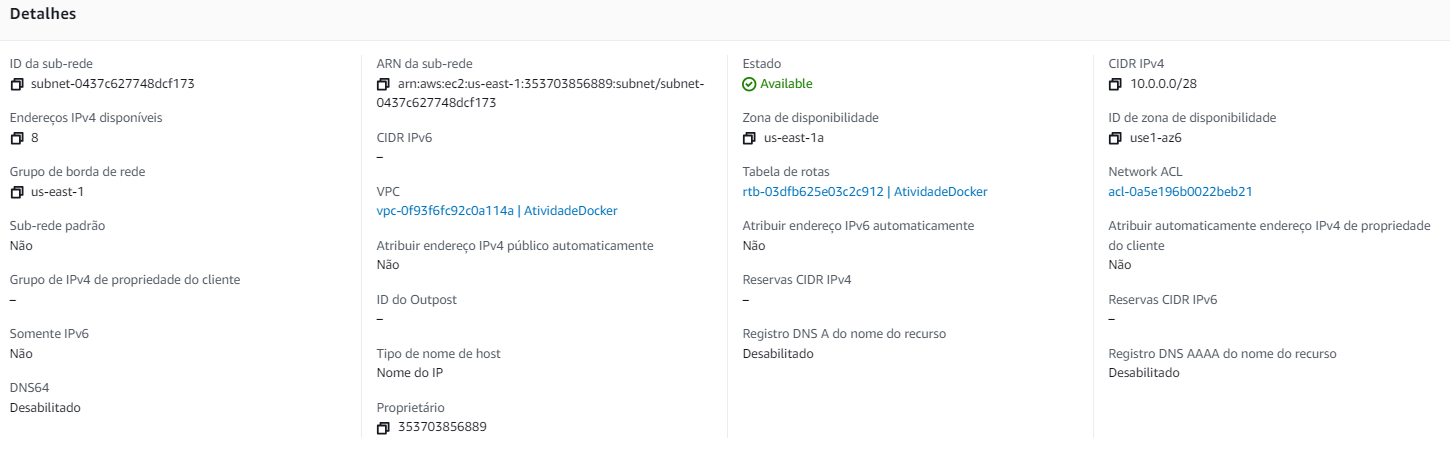
CIDRs



IPV4 associado a VPC (10.0.0.0/24

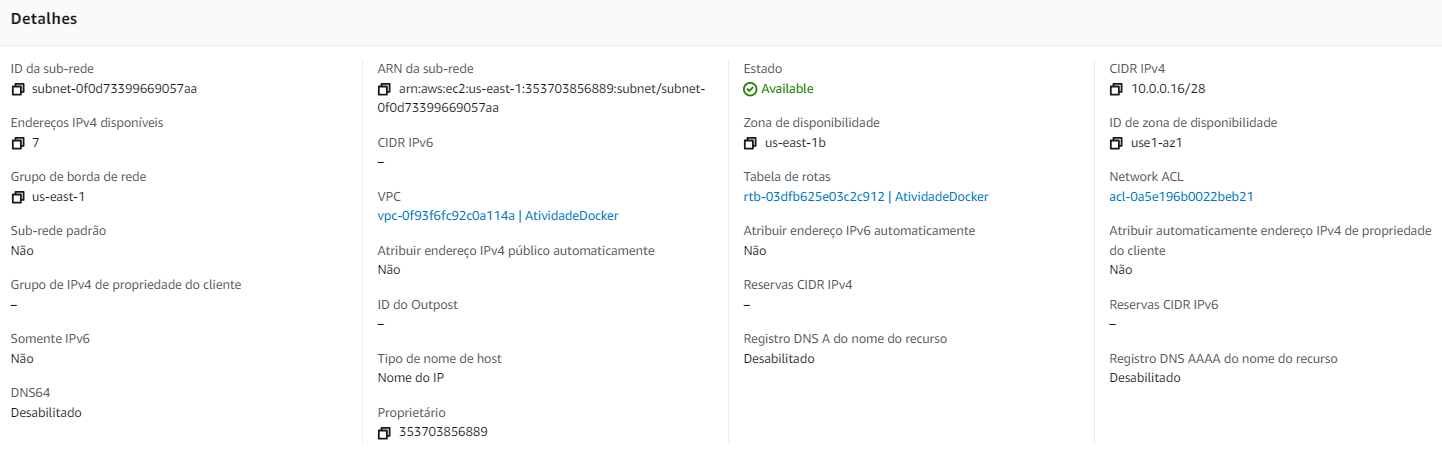
Subnets

InstanciaDocker1

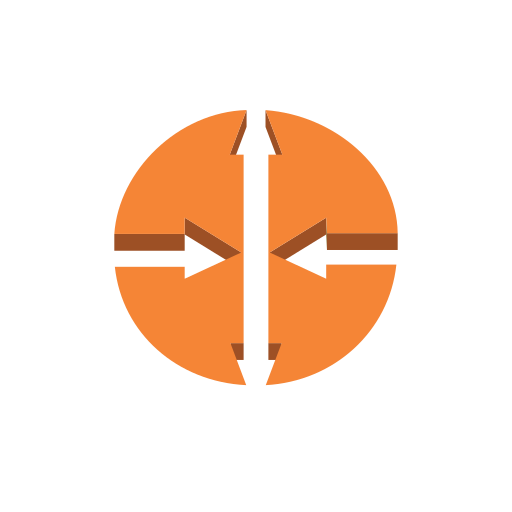


Subnet1 criada e vinculada a primeira instância criada na zona de disponibilidade us-east-1a com um CIDR (10.0.0.0/28), vinculada a Tabela de rotas para o trafego de rede para recursos e a VPC (AtividadeDocker).

InstânciaDocker2

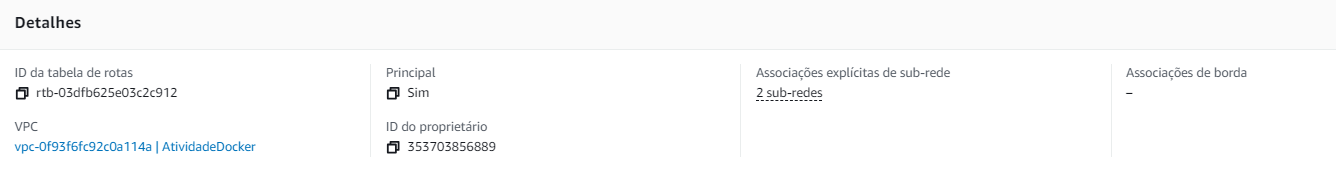


Subnet2 criada e vincula a segunda instância criada na zona de disponibilidade us-east-1B com um CIDR (10.0.0.16/28), vinculada a Tabela de rotas juntamente a primeira subnet para o trafego de rede para recursos e a VPC (AtividadeDocker).



Route Tables

AtividadeDocker

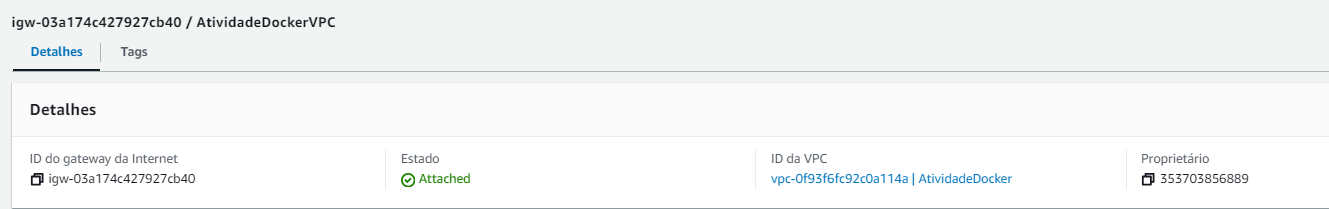


Vinculada a VPC (AtividadeDocker) mais associação das duas subnets existentes e a um Internet Gatway para acesso a internet.



Rotas vinculadas a Route Table, tendo um Gatway de Internet vinculado e um ip local.

Gatway de Internet

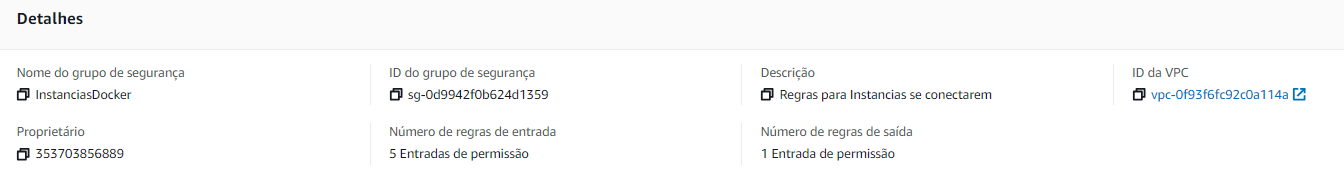


Criado com a função de fornecer internet para VPC e outras funções que viriam ser vinculadas com a VPC.

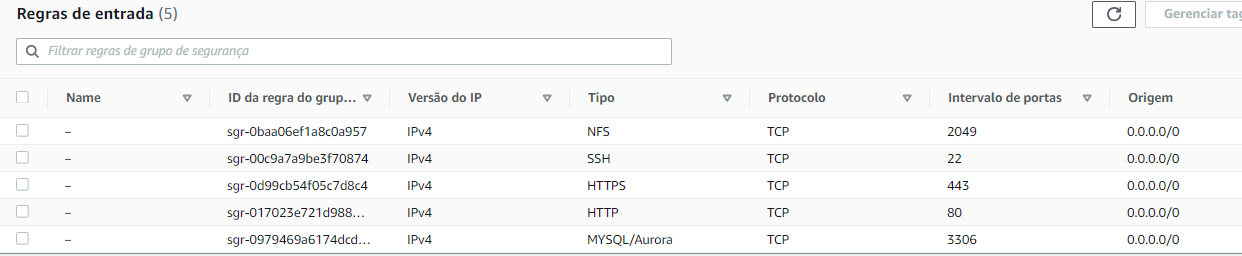


Security Groups

InstanciasDocker



Criada com o objetivo de fornecer portas e acesso para as seguintes funções:



SSH: Libera o acesso a instancia a partir do chave gerada (AtividadeDocker.pem).

NFS: Libera o acesso a função EFS, para que assim ambas as instâncias pudessem acessar um arquivo compartilhado pela função da própria AWS (EFS).

HTTPS: Libera o acesso para o navegador para que a conexão seja criptografada .

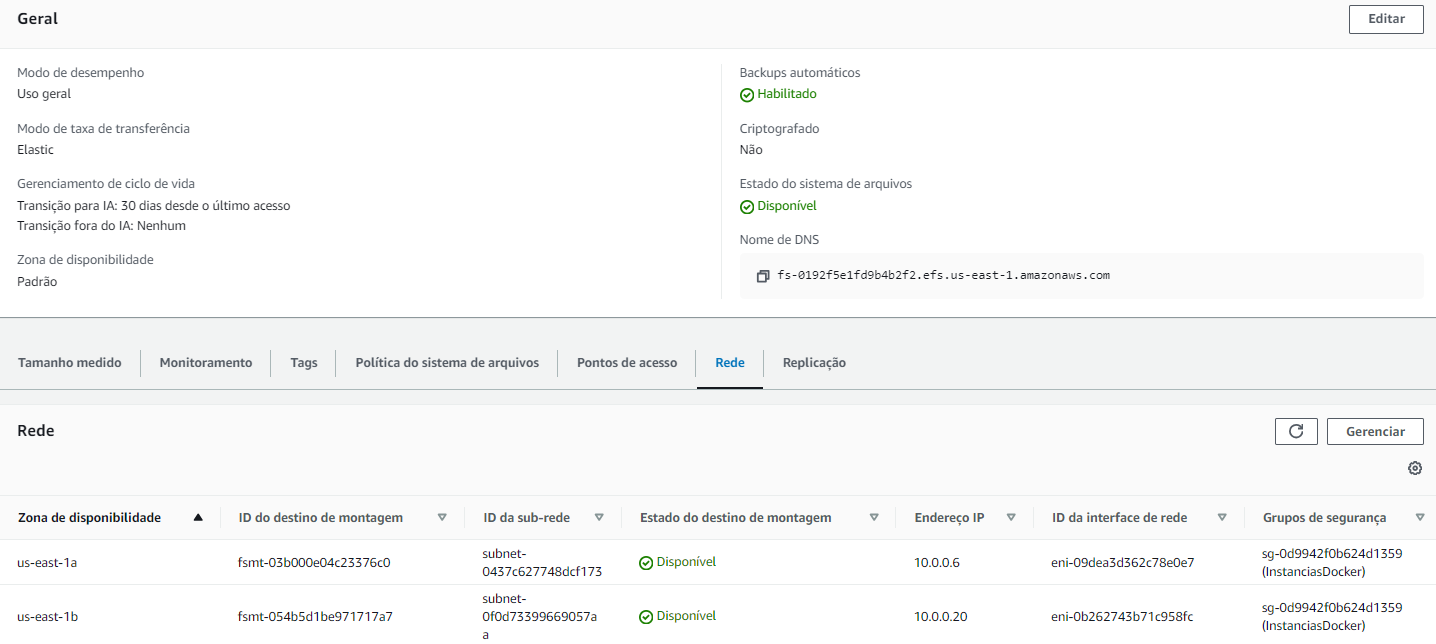
HTTP: Libera o acesso ao navegador para a comunicação da aplicação com o navegador.

MYSQL/Aurora: Libera o acesso ao banco de dados criado pelo RDS.



EFS

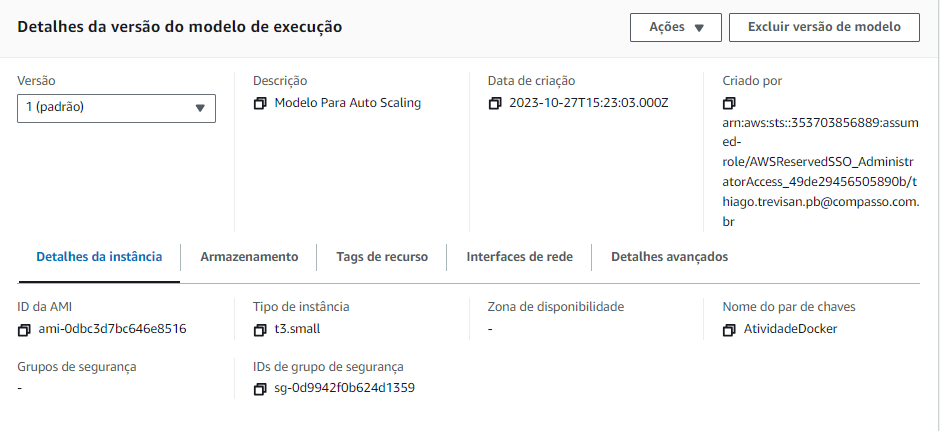
DockerEFS



Criado para a compartilhar arquivos dentro de uma pasta especificada pelo usuário, vinculado as duas subnets existentes para ter acesso a endereço de IPV4 de cada subnet fazendo com que assim possa ser gerado uma conexão para ambas as instâncias.

Modelo de Execução das instâncias

(Auto Scaling)



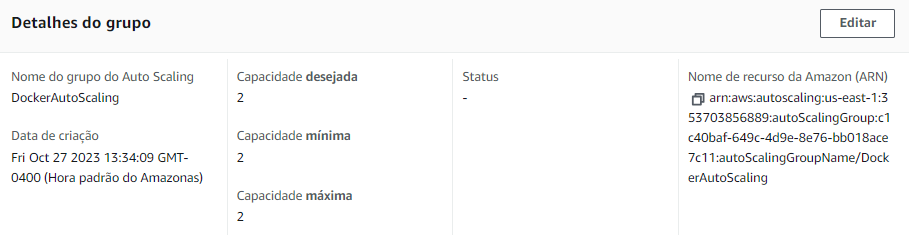
Modelo de execução criado com a função de iniciar as instâncias dentro do auto scaling, com o tipo de instância t3.small, 16G de armazenamento, Chave SSH,

Security Group \*Na imagem não possui um security group, mas ao criar a instância a partir do auto scaling ele atribui o security group configurado para o modelo de execução\*.



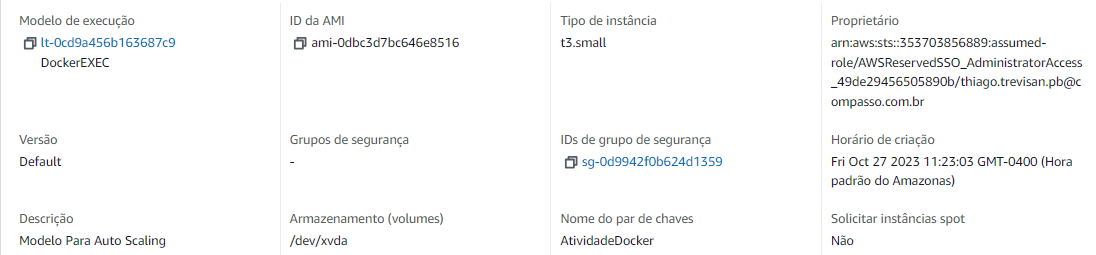
Auto Scaling

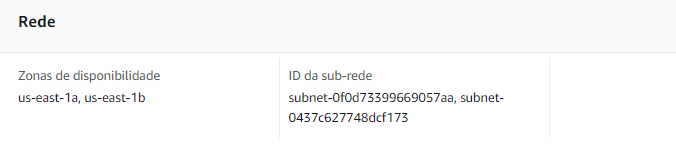
Nome: DockerAutoScaling



Detalhe sobre a capacidade desejada sobre o auto Scaling de 2 para 2.

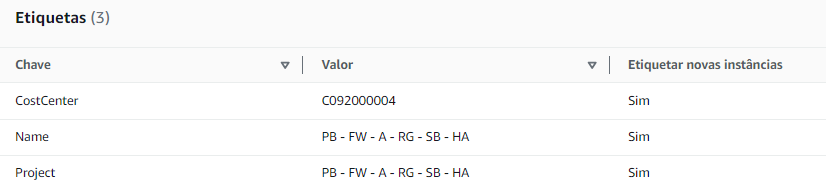
Modelo de Execução Auto Scaling



Rede Auto Scaling

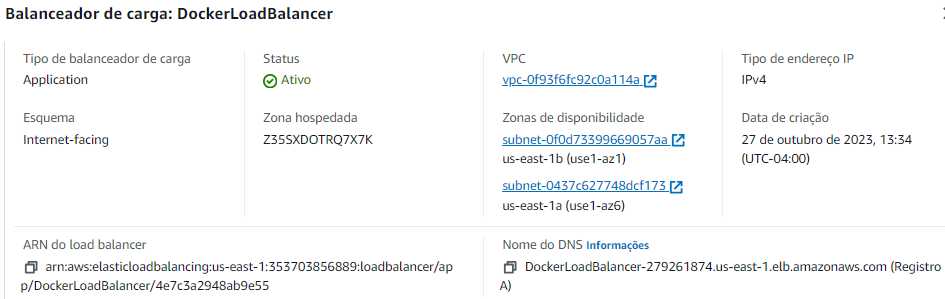
Duas zonas de disponibilidade 1a e 1b e duas subnets para cada zona, fazendo com que assim cada instância esteja em uma zona diferente assim como a atividade pediu.

Tags Auto Scaling





Load Balancer

Detalhes 

Tipo de Load Balancer: Application.

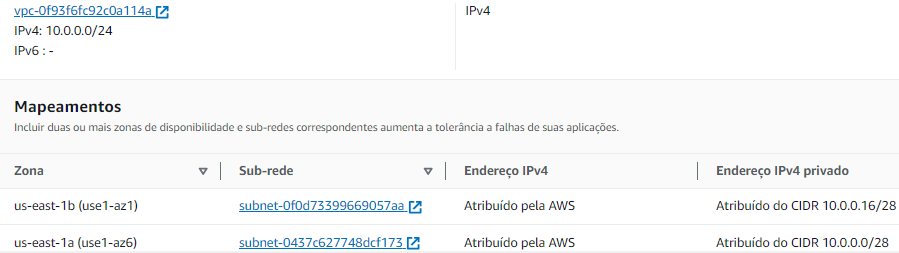
Tipo de Endereço: IPV4.

Zonas de Disponibilidade: 1A e 1B.

Esquema: Internet-facing.

VPC: AtividadeDocker.

Mapeamento de Rede

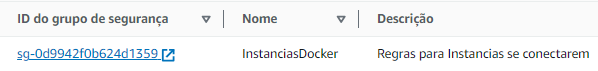


IPV4:10.0.0.0/24 (mesmo da VPC).

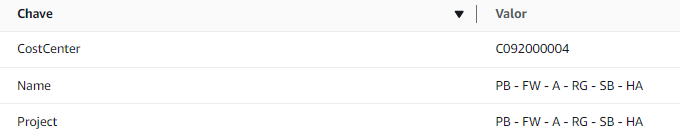
Zonas 1A (10.0.0.0/28) e 1B (10.0.0.16/28).

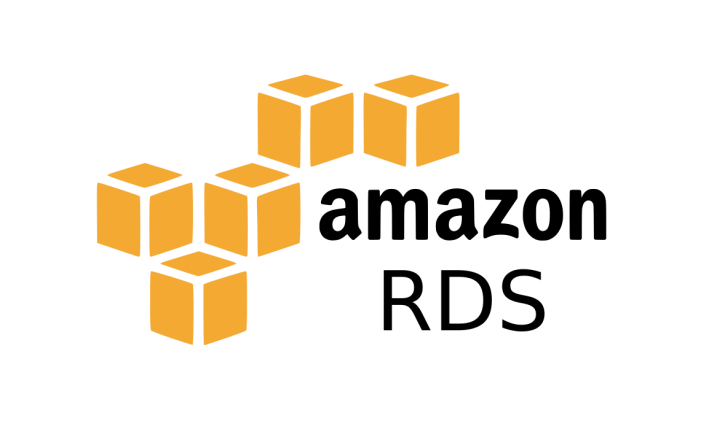
Onde ele redireciona para ambas as zonas o acesso do cliente na aplicação.

Security Groups Load Balancer



Mesmo Security Group que é usado para as instâncias

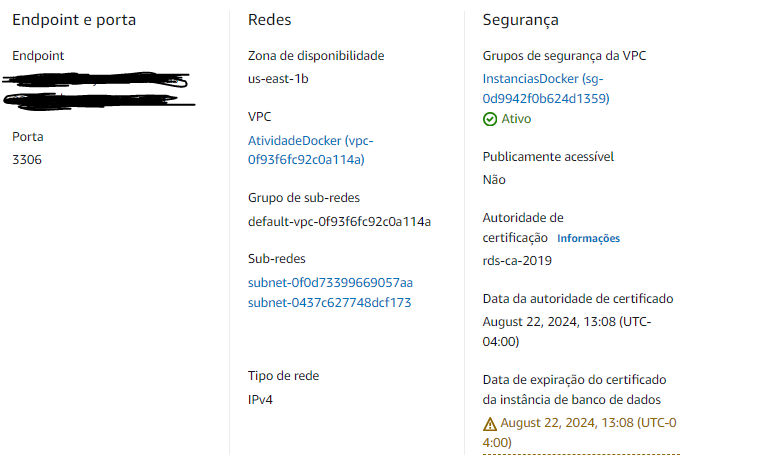
Tags Load Balancer



RDS

(Banco Dados Mysql)

Segurança e conexão



Tipo de Rede: IPV4

Security Group: InstanciasDocker.

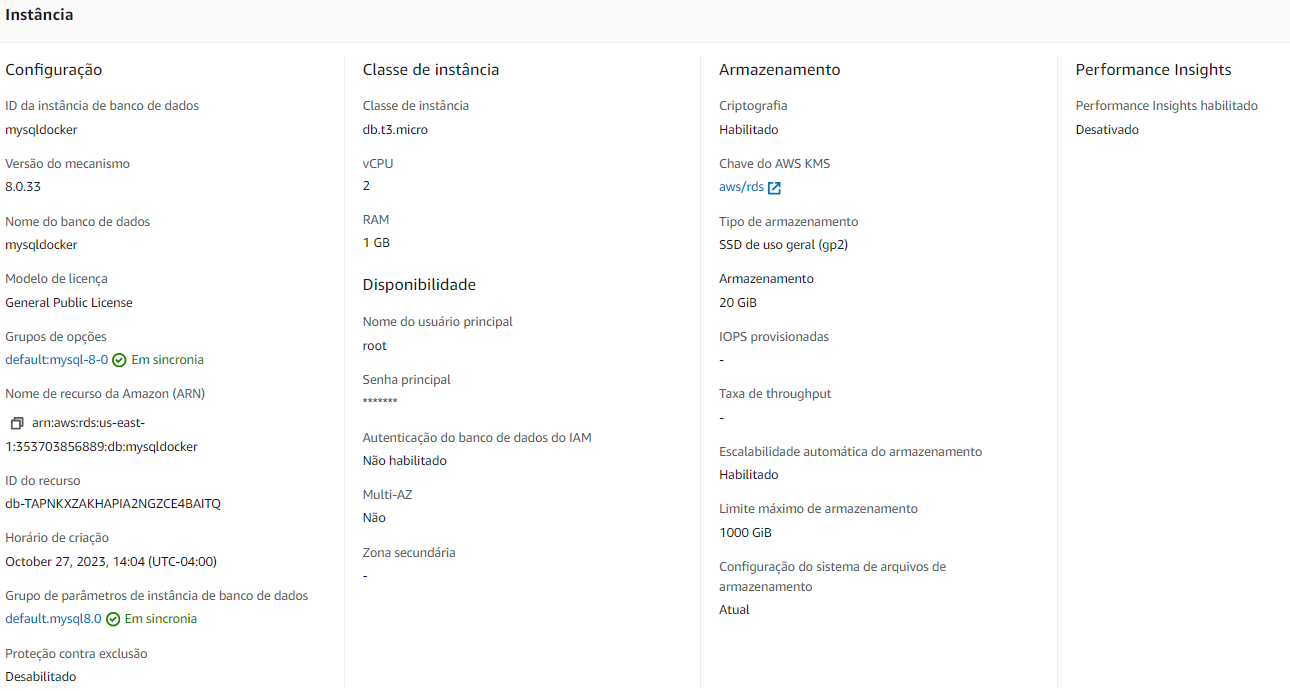
Porta: 3306.

Sub-nets: 1A e 1B .

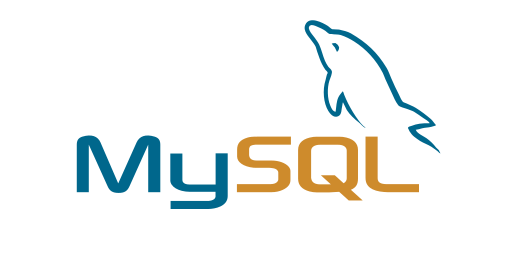
Zona de Disponibilidade: 1B .

VPC: AtividadeDocker

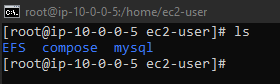
Configuração



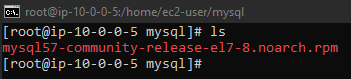
Criado no nível gratuito vinculado a VPC mais as duas subnets existentes em zonas de disponibilidade diferentes (1A e 1B), com um armazenamento de 20GB e com o intuito de servir como base para a conexão mais a instalação do WordPress.



Instalação Mysql Instancia



Primeiro eu criei uma pasta pro mysql usando o comando “mkdir mysql”



Dentro dele eu executei o comando

“sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022

wget <http://dev.mysql.com/get/mysql57-community-release-el7-8.noarch.rpm>”.

Instalando assim um executavel do Mysql-community

Em seguida executo o comando

“sudo yum localinstall -y mysql57-community-release-el7-8.noarch.rpm”.

Para baixar o pacote

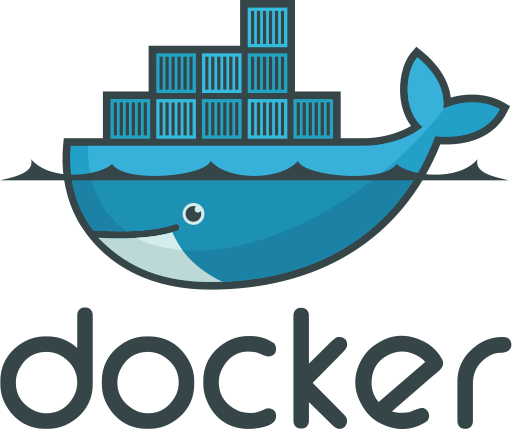
Logo depois executo a biblioteca instalada na maquina

“sudo yum install -y mysql-community-server”.

Depois eu só preciso iniciar o mysql e habilitar ele dentro da instância

“sudo systemctl start mysqld”.

“sudo systemctl enable mysqld”.



Instalação Docker Instancia

Executo o comando para a instalação do docker

“yum install docker –y”.

Depois eu starto a bibloteca instalada

“systemctl start docker”.

Então deixo ele ativo para que toda vez em que a instância for iniciada ele execute o progama.

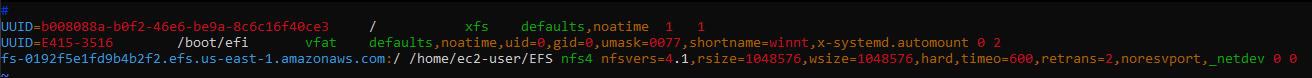
systemctl enable docker



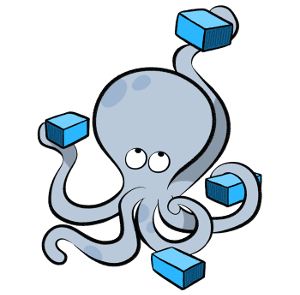
Montagem Do EFS

Executo o comando vim /etc/fstab para acessar as montagens automáticas e adiciono a seguinte linha para deixar a mount automática.

“fs-0192f5e1fd9b4b2f2.efs.us-east-1.amazonaws.com:/ /home/ec2-user/EFS nfs4 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport,\_netdev 0 0”.



Onde eu determino id do EFS criado especifico o local da montagem



Docker-Compose

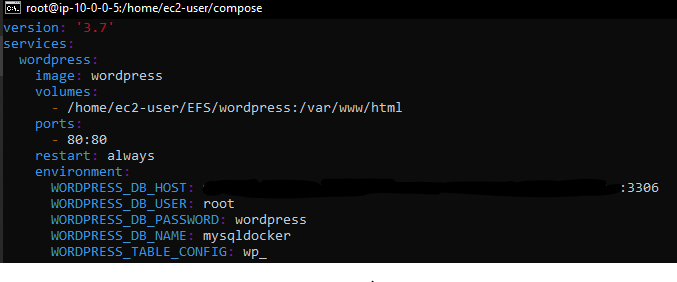
Executo o Comando

“sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose”.

Baixo o compose via git-hub e dou permissão para ele se tornar executável.

“sudo chmod +x /usr/local/bin/docker-compose”.

Arquivo docker-compose.yml



Especifico a versão do serviço

Image: wordpress (especifica a imagem da aplicação).

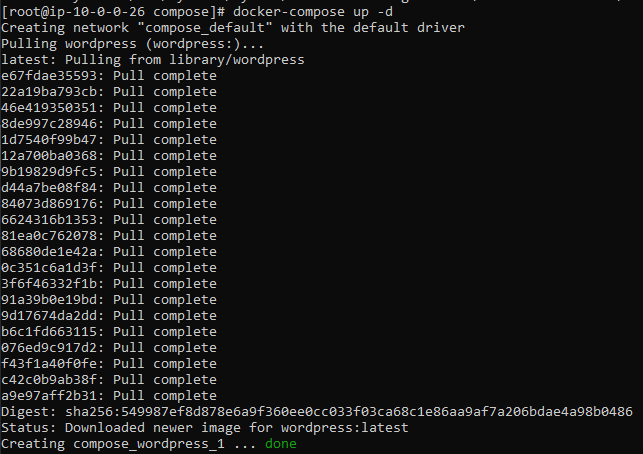
Volumes: /home/ec2-user/EFS/wordpress:/var/www/HTML (Falo pra ele jogar todos os arquivos do wordpress dentro do diretório especificado na montagem).

Ports: Especifico a porta onde aplicação será exposta.

Enviroment: Onde é especificado as variáveis para o acesso ao banco de dados (RDS)

Subindo aplicação

“docker-compose up –d”



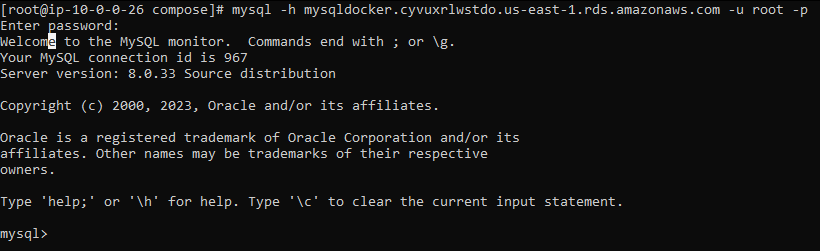
Ele irá buscar a versão mais recente do wordpress baixar ele e subir e deixar a aplicação exposta na porta 80

Depois execute o comando docker ps, para verificar o estado da aplicação

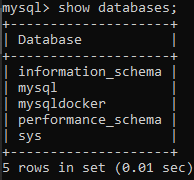
Usando o link do Load Balancer

Executo o comando para se conectar com o banco de dados

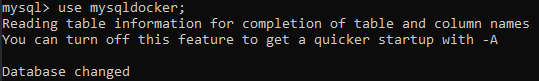
“mysql -h mysqldocker.cyvuxrlwstdo.us-east-1.rds.amazonaws.com -u root -p”



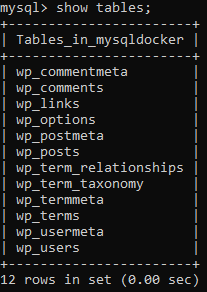
Dentro do banco de dados eu vou atrás da database do wordpress



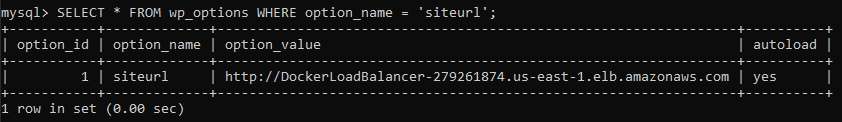
Achando a database eu uso o comando “use mysqldocker;” e entro na database



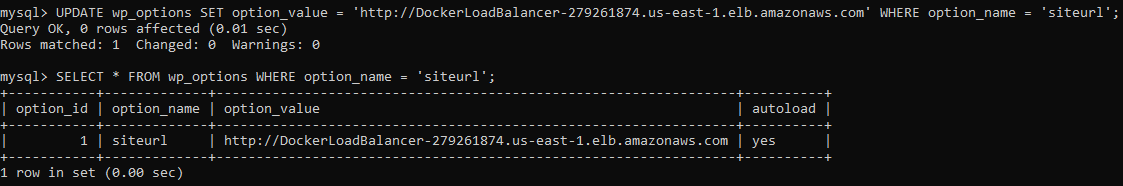
Dentro da database eu uso o comando “show tables;” para procurar uma tabela chamada wp\_options



Achando a tabela eu executo o comando “SELECT \* FROM wp\_options WHERE option\_name = 'siteurl';” para procurar a url do wordpress



Encontrando a url do site eu executo novamente outro comando para alterar para a URL do Load Balancer “UPDATE wp\_options SET option\_value = 'http://DockerLoadBalancer-279261874.us-east-1.elb.amazonaws.com' WHERE option\_name = 'siteurl';”



Após fazer tudo isso é apenas jogar o link do Load Balancer no Google e verificar se tudo está funcionando corretamente

