

Sumário

Setup inicial / Premissas	2
Configuração Proxy Reverso	2
NGINX	2
Banco de dados	4
API	8
Proxy Reverso	11
Load balancer	14

Setup inicial / Premissas

Olá! Para seguir neste tutorial, é essencial que você já tenha configurado a infraestrutura básica: VPC, Internet Gateway, Route Tables, NACL, Subredes, Gateway Nat, e Security Group. Se algum desses componentes ainda não estiver pronto, por favor, volte duas casas e complete os labs anteriores.

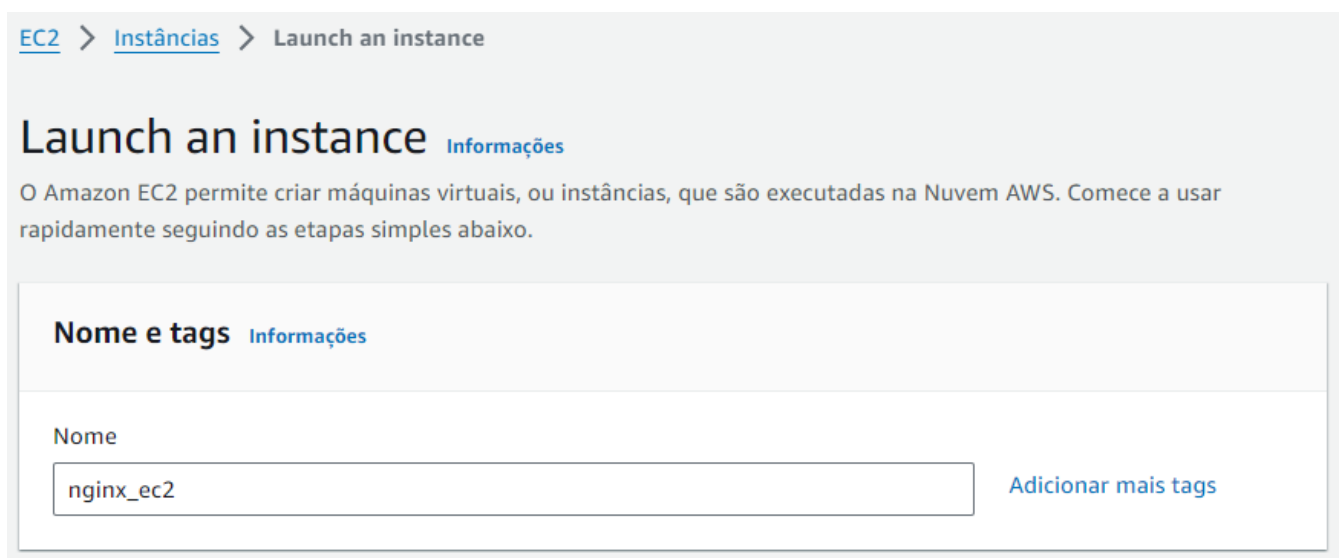
Além disso, é necessário que você tenha uma chave **“.pem”** previamente configurada no seu computador. Essa chave será usada para estabelecer uma conexão com as instâncias que vamos criar e testar. Com tudo pronto, estamos aptos a avançar!

Configuração Proxy Reverso

O nosso ponto de partida é a configuração de três componentes críticos. O primeiro será alocado em uma sub-rede pública e servirá como o servidor para o Nginx, que desempenhará as funções de um proxy reverso e balanceador de carga. O segundo componente atuará como um simulador de um serviço de banco de dados, análogo ao RDS oferecido pela AWS. Para fins de simplificação, adotaremos uma instância EC2 que rodará um contêiner MySQL para esta finalidade. A terceira peça da nossa estrutura será uma instância rodando uma API Java Spring Boot em Docker. Esta abordagem nos permitirá estabelecer uma conexão eficaz entre o proxy reverso, a API e o banco de dados dentro de nossa infraestrutura.

NGINX

No painel AWS, execute uma instância Ubuntu **vinculada a rede pública** para hospedar o Nginx, nesse exemplo chamarei a instancia de **nginx_ec2**:



EC2 > Instâncias > Launch an instance

Launch an instance [Informações](#)

O Amazon EC2 permite criar máquinas virtuais, ou instâncias, que são executadas na Nuvem AWS. Comece a usar rapidamente seguindo as etapas simples abaixo.

Nome e tags [Informações](#)

Nome

[Adicionar mais tags](#)

Configurações de rede:

▼ Configurações de rede [Informações](#)

VPC - obrigatório [Informações](#)

vpc-074cb7f2fed4a7f37 (minha-vpc)
10.0.0.0/24

↻

Sub-rede [Informações](#)

subnet-0682f5d3beebbf346
VPC: vpc-074cb7f2fed4a7f37 Proprietário: 767397774912
Zona de disponibilidade: us-east-1a Endereços IP disponíveis: 121 CIDR: 10.0.0.0/25

sub-rede-publica
↻

[Criar nova sub-rede](#)

Atribuir IP público automaticamente [Informações](#)

Habilitar

↻

[Additional charges apply](#) when outside of [free tier allowance](#)

Execute a instância e tente acessá-la via **ssh**, exemplo:

```
ubuntu@ip-10-0-0-54: ~  
brito in ~  
> ssh -i ~/.ssh/wsl2.pem ubuntu@44.208.22.157  
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
  
System information as of Wed Apr  3 17:02:35 UTC 2024  
  
System load:  0.86376953125      Processes:            105  
Usage of /:   20.4% of 7.57GB    Users logged in:     0  
Memory usage: 21%               IPv4 address for eth0: 10.0.0.54  
Swap usage:   0%  
  
Expanded Security Maintenance for Applications is not enabled.  
0 updates can be applied immediately.  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
Last login: Wed Apr  3 17:02:36 2024 from 131.72.61.70  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-10-0-0-54:~$
```

LAB PROXY REVERSO E BALANCEAMENTO DE CARGA

Próximo passo, vamos atualizar o gerenciador de pacotes e, imediatamente após, proceder com a instalação do Nginx utilizando os comandos apropriados:

- `sudo apt update`
- `sudo apt install nginx`

Ao final dessas operações, tente acessar o IP público da instância criada, você deverá ver uma página como essa:

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Não esquece de transferir sua chave ssh para lá pois as próximas configurações serão feitas em máquinas que estarão na sub-rede privada. Utilize o comando scp para transferir o arquivo, exemplo:

```
brito in ~  
> scp -i .ssh/ws12.pem .ssh/ws12.pem ubuntu@44.208.22.157:/tmp  
ws12.pem  
brito in ~ took 2.0s  
> |
```

Banco de dados

Agora execute uma instância na sub-rede privada, exemplo:

[EC2](#) > [Instâncias](#) > Launch an instance

Launch an instance [Informações](#)

O Amazon EC2 permite criar máquinas virtuais, ou instâncias, que são executadas na Nuvem AWS. Comece a usar rapidamente seguindo as etapas simples abaixo.

Nome e tags [Informações](#)

Nome

[Adicionar mais tags](#)

Configuração de rede:

▼ Configurações de rede [Informações](#)

VPC - obrigatório [Informações](#)

vpc-074cb7f2fed4a7f37 (minha-vpc)
10.0.0.0/24

Sub-rede [Informações](#)

subnet-009a6708634319d1b sub-rede-privada
VPC: vpc-074cb7f2fed4a7f37 Proprietário: 767397774912
Zona de disponibilidade: us-east-1a Endereços IP disponíveis: 123
CIDR: 10.0.0.128/25

Criar nova sub-rede

Atribuir IP público automaticamente [Informações](#)

Desabilitar

Em seguida, tente acessar a máquina via ssh usando a instância **niginx_ec2(rede pública)**, exemplo

```

ubuntu@ip-10-0-0-232: ~
ubuntu@ip-10-0-0-54:~$ ssh -i ~/.ssh/wsl2.pem ubuntu@10.0.0.232
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro

System information as of Wed Apr  3 17:22:50 UTC 2024

System load:  0.0009765625      Processes:           96
Usage of /:   20.4% of 7.57GB   Users logged in:    0
Memory usage: 21%              IPv4 address for eth0: 10.0.0.232
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Apr  3 17:22:50 2024 from 10.0.0.54
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-0-232:~$

```

LAB PROXY REVERSO E BALANCEAMENTO DE CARGA

Nessa instância utilizaremos o Docker para rodar um container de Mysql, utilize os comandos:

- `sudo apt update`
- `sudo apt install docker.io`

Adicionando o usuário para o grupo do Docker:

- `sudo usermod -a -G docker $(whoami)`

Em seguida precisaremos renovar a sessão para aplicar as modificações ao usuário, podemos sair e reconectar ou usar o comando:

- `newgrp docker`

Ao final execute o comando `docker run hello-world` sem o `sudo`, deverá aparecer algo como

```
ubuntu@ip-10-0-0-232: ~  
ubuntu@ip-10-0-0-232:~$ sudo docker run hello-world  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/  
ubuntu@ip-10-0-0-232:~$
```

LAB PROXY REVERSO E BALANCEAMENTO DE CARGA

Agora, estamos prontos para executar nosso container MySQL nesta instância. Existem diversas maneiras de realizar essa tarefa, mas aqui vai um exemplo prático:

- **`docker run --name bd_ec2 -e MYSQL_ROOT_PASSWORD=urubu100 -p 3306:3306 -d mysql:latest`**

```
ubuntu@ip-10-0-0-232: ~  
ubuntu@ip-10-0-0-232:~$ docker run --name bd_ec2 -e MYSQL_ROOT_PASSWORD=urubu100 -p 3306:3306 -d mysql:latest  
Unable to find image 'mysql:latest' locally  
latest: Pulling from library/mysql  
9a5c778f631f: Pull complete  
ccc451c3fb55: Pull complete  
db534de989c8: Pull complete  
c1a1ab6fb3ea: Pull complete  
d18a374d12e6: Pull complete  
2d9f4c3e8c03: Pull complete  
4c79cbebf62: Pull complete  
b3549fdd6799: Pull complete  
c08846a4ab7a: Pull complete  
084bd453daf0: Pull complete  
Digest: sha256:4552fcc5d3c8b8cdee76ee25cce28bf60b0eb3ce93d25ba3bfff7a66bfdcdee8  
Status: Downloaded newer image for mysql:latest  
04838e5acf7a06400c4afee92999f7cae500af197b9c61ed6b8f197f2c946d1c  
ubuntu@ip-10-0-0-232:~$
```

Logo após, execute o comando **`docker ps`**, o nosso container deve aparecer nessa listagem:

```
ubuntu@ip-10-0-0-232: ~  
ubuntu@ip-10-0-0-232:~$ docker ps  
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES  
04838e5acf7a   mysql:latest "docker-entrypoint.s..." 47 seconds ago Up 46 seconds 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp bd_ec2  
ubuntu@ip-10-0-0-232:~$
```

Precisamos criar um banco de dados chamado **banco_teste**, você pode utilizar o **Bash** dentro do próprio container para acessar o banco através do cliente MySQL e realizar a criação. Siga os passos abaixo para executar esta tarefa:

- Inicie uma sessão interativa no Bash do container onde o banco de dados está rodando, utilizando o comando Docker: **`docker exec -it bd_ec2 /bin/bash`**
- Acesse o MySQL como usuário root. Após executar o comando a seguir, insira a senha quando solicitado: **`mysql -u root -p`**
- Uma vez logado no MySQL, crie o banco de dados **banco_teste** com o comando: **`CREATE DATABASE banco_teste;`**
- Para verificar se o banco de dados foi criado com sucesso, liste todos os bancos de dados disponíveis com: **`SHOW DATABASES;`**

```
ubuntu@ip-10-0-0-232: ~  
ubuntu@ip-10-0-0-232:~$ docker exec -it bd_ec2 /bin/bash  
bash-4.4# mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 49  
Server version: 8.3.0 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> CREATE DATABASE banco_teste;
```

Após executar o comando **SHOW DATABASES**:

```
ubuntu@ip-10-0-0-232: ~  
mysql> SHOW DATABASES;  
+-----+  
| Database  
+-----+  
| banco_teste  
| information_schema  
| mysql  
| performance_schema  
| sys  
+-----+  
5 rows in set (0.00 sec)  
  
mysql> █
```

Feito isso, deixe rodando o container e termine a sessão.

API

Em seguida, é necessário iniciar uma instância EC2 para hospedar uma API de exemplo. Vamos utilizar uma API simples de CRUD para livros que inclui o Swagger para facilitar os testes. Alternativamente, pode-se utilizar um cliente HTTP avançado como o Insomnia, Postman, entre outros, para interagir com a API.

Execute uma instancia na sub-rede privada, exemplo:

EC2 > [Instâncias](#) > Launch an instance

Launch an instance [Informações](#)

O Amazon EC2 permite criar máquinas virtuais, ou instâncias, que são executadas na Nuvem AWS. Comece a usar rapidamente seguindo as etapas simples abaixo.

Nome e tags [Informações](#)

Nome

[Adicionar mais tags](#)

Configuração de rede:

▼ Configurações de rede
Informações

VPC - obrigatório
Informações

vpc-074cb7f2fed4a7f37 (minha-vpc)
10.0.0.0/24

Sub-rede
Informações

subnet-009a6708634319d1b
VPC: vpc-074cb7f2fed4a7f37 Proprietário: 767397774912
Zona de disponibilidade: us-east-1a Endereços IP disponíveis: 122
CIDR: 10.0.0.128/25

Atribuir IP público automaticamente
Informações

Desabilitar

Criar nova sub-rede

Em seguida tente acessá-la via **ssh** através da máquina **nginx_ec2(rede pública)**, exemplo:

```

ubuntu@ip-10-0-0-152: ~
ubuntu@ip-10-0-0-54:~$ ssh -i ./ssh/wsl2.pem ubuntu@10.0.0.152
The authenticity of host '10.0.0.152 (10.0.0.152)' can't be established.
ED25519 key fingerprint is SHA256:9YsB4Xkyq0adDM5PQZwIP6Qub0+g7WqoF5bzzjDD+/U.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.0.152' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed Apr  3 18:14:03 UTC 2024

System load:  0.2724609375   Processes:            102
Usage of /:   20.4% of 7.57GB Users logged in:        0
Memory usage: 21%           IPv4 address for eth0: 10.0.0.152
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-0-152:~$

```

Atualize o gerenciador de pacotes e repita o passo a passo de instalação do Docker que fizemos ao criar a EC2 que representa o banco.

- **sudo apt update**
- **sudo apt install docker.io**
- **sudo usermod -a -G docker \$(whoami)**

- **newgrp docker**

Usaremos para o exemplo essa [imagem](#). Essa imagem possui uma API REST feita com Spring Boot, que deve receber as variáveis de configuração do banco de dados, são elas:

- **DB_HOST:** endereço do banco, aqui colocaremos o endereço da EC2 que contém o container MySQL;
- **DB_USERNAME:** Aqui deveria ser um usuário para a aplicação, mas como se trata de um teste utilizaremos o próprio root;
- **DB_PASSWORD:** senha configurada para o usuário root do nosso banco;

Podemos realizar o seguinte comando para baixar a imagem Docker e já executá-la em sequência:

```
docker run -e DB_HOST=jdbc:mysql://10.0.0.232:3306/banco_teste \
-e DB_USER=root \
-e DB_PASSWORD=urubu100 \
-p 8080:8080 \
--name api_01 \
-d \
dibrito/load-balancer-api:latest
```

Notas: Altere o endereço IP destacado em vermelho para corresponder ao IP correto da sua instância.

Executando **docker ps**:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
978e44d86402	dibrito/load-balancer-api:latest	"java -jar app.jar"	41 seconds ago	Up 40 seconds	3004/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp	api_01

Para verificar se o spring rodou certinho você pode executar:

- **docker logs api_01 --follow**

```
ubuntu@ip-10-0-0-152:~$ docker logs api_01
2024-04-03T18:51:06.588Z INFO 1 --- [main] s.s.e.ExemploApiLoadBalancerApplication : Starting ExemploApiLoadBalancerApplication v0.0.1-SNAPSHOT using Java 17.0.1 with PID 1 (/app/app.jar started by root in /app)
2024-04-03T18:51:06.602Z INFO 1 --- [main] s.s.e.ExemploApiLoadBalancerApplication : No active profile set, falling back to 1 default profile: "default"
2024-04-03T18:51:08.987Z INFO 1 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-04-03T18:51:09.047Z INFO 1 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 111 ms. Found 1 JPA repository interface.
2024-04-03T18:51:10.427Z INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2024-04-03T18:51:10.452Z INFO 1 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-04-03T18:51:10.455Z INFO 1 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-04-03T18:51:10.746Z INFO 1 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-04-03T18:51:10.754Z INFO 1 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 4030 ms
2024-04-03T18:51:11.575Z INFO 1 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-04-03T18:51:11.764Z INFO 1 --- [main] org.hibernate.Version : HHH0000412: Hibernate ORM core version 6.4.4.Final
2024-04-03T18:51:11.779Z INFO 1 --- [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2024-04-03T18:51:12.394Z INFO 1 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup; ignoring JPA class transformer
2024-04-03T18:51:12.366Z INFO 1 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-04-03T18:51:13.179Z INFO 1 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@5855b0ed
2024-04-03T18:51:13.181Z INFO 1 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-04-03T18:51:13.331Z INFO 1 --- [main] org.hibernate.orm.deprecation : HHH9000025: MySQLDialect does not need to be specified explicitly using 'hibernate.dialect'
2024-04-03T18:51:16.331Z INFO 1 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
2024-04-03T18:51:16.528Z INFO 1 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-04-03T18:51:17.455Z WARN 1 --- [main] jpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2024-04-03T18:51:18.539Z INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
2024-04-03T18:51:18.578Z INFO 1 --- [main] s.s.e.ExemploApiLoadBalancerApplication : Started ExemploApiLoadBalancerApplication in 13.319 seconds (process running for 14.786)
```

LAB PROXY REVERSO E BALANCEAMENTO DE CARGA

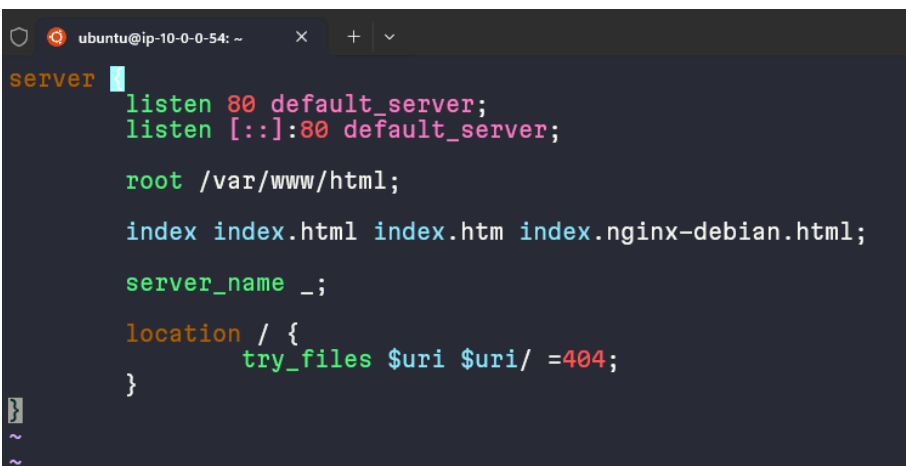
Pronto, deixe rodando a api e vamos configurar a máquina do Nginx para possibilitar acessar o Swagger realizando o proxy reverso.

Proxy Reverso

Conecte-se à instância EC2 que está rodando o Nginx, chamada `nginx_ec2`, usando SSH. Em seguida, edite o arquivo de configuração utilizando um editor de texto. Neste caso, será usado o Vim como exemplo. Execute o seguinte comando para abrir o arquivo:

- **`sudo vim /etc/nginx/sites-available/default`**

O arquivo padrão contém comentários, eu os removi para ficar mais legível, mas deve conter a seguinte parametrização:



```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

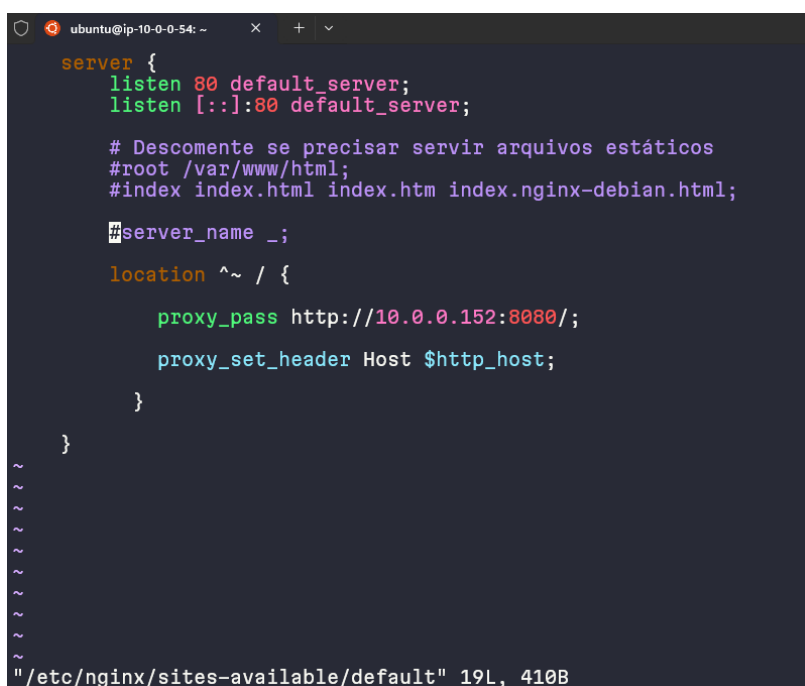
    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Dado que estamos realizando um teste, um front-end não é necessário. Podemos modificar a diretiva `location /` para redirecionar as requisições diretamente para a nossa API, como no seguinte exemplo:



```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # Descomente se precisar servir arquivos estáticos
    #root /var/www/html;
    #index index.html index.htm index.nginx-debian.html;

    #server_name _;

    location ^~ / {
        proxy_pass http://10.0.0.152:8080/;
        proxy_set_header Host $http_host;
    }
}
```

"/etc/nginx/sites-available/default" 19L, 410B

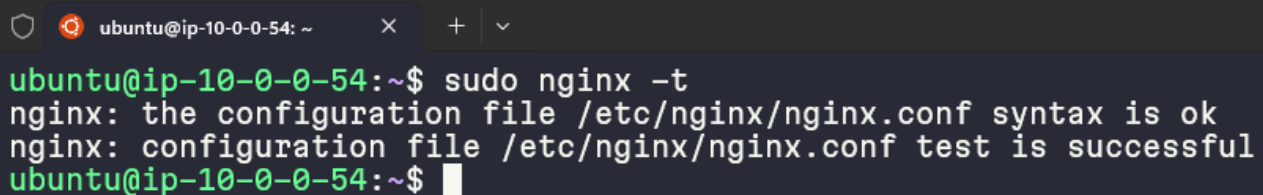
Para facilitar copie:

```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
    # Descomente se precisar servir arquivos estáticos  
    #root /var/www/html;  
    #index index.html index.htm index.nginx-debian.html;  
    #server_name _;  
    location ^~ / {  
        proxy_pass http://10.0.0.152:8080/;  
        proxy_set_header Host $http_host;  
    }  
}
```

Notas: Altere o endereço IP destacado em vermelho para corresponder ao IP correto da sua instância.

Para salvar o arquivo e sair do editor Vim, primeiro pressione a tecla ESC para sair do modo de inserção. Depois, digite : seguido de wq (que significa "Write and Quit", ou "Salvar e Sair") e pressione Enter. Isso fará com que as alterações sejam salvas e o editor seja fechado.

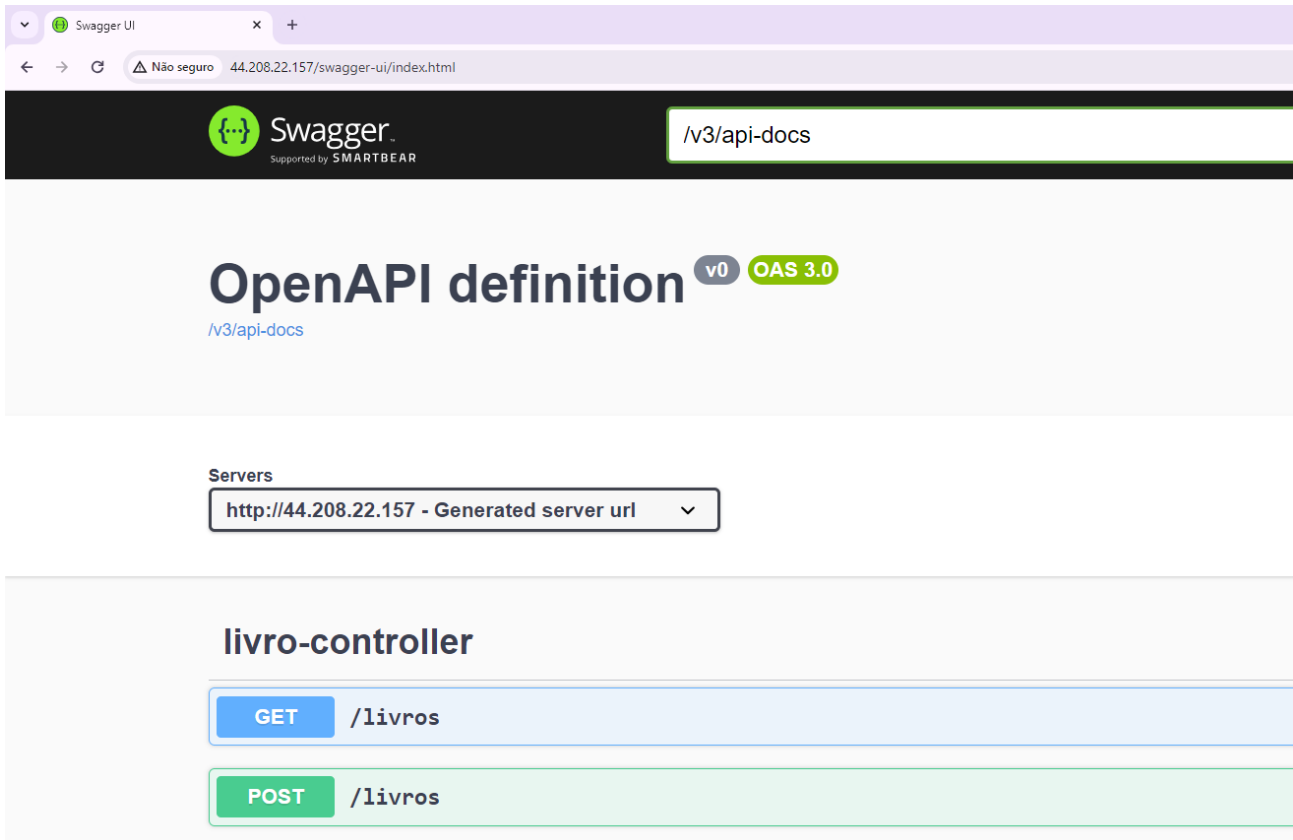
Para checar a validade da configuração do Nginx, execute o comando **sudo nginx -t**. O resultado esperado deve ser algo parecido com:



```
ubuntu@ip-10-0-0-54: ~  
ubuntu@ip-10-0-0-54:~$ sudo nginx -t  
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful  
ubuntu@ip-10-0-0-54:~$
```

LAB PROXY REVERSO E BALANCEAMENTO DE CARGA

Agora, para aplicar as alterações, é só recarregar o serviço do Nginx com o comando ***sudo systemctl reload nginx.service***. Tentando acessar o **endereço IP do servidor público**, você não encontrará a página padrão do Nginx disponível. No entanto, você deverá ser capaz de acessar o Swagger da API adicionando **/swagger-ui/index.html ao final do IP do servidor**. Também é possível fazer requisições à API REST utilizando uma ferramenta como o Insomnia.



É possível criar um livro e utilizar o endpoint de listagem para verificar se a configuração está correta. Com isso, finalizamos nosso setup incluindo um proxy reverso.

Load balancer

Agora, precisamos fazer alguns ajustes na configuração do Nginx e configurar uma instância adicional para facilitar o balanceamento de carga entre as APIs por meio do Nginx. É importante notar que esta configuração é básica e voltada para fins de teste, e ela pode ser aprimorada. Dependendo da situação e das necessidades específicas, pode ser aconselhável optar por um serviço de balanceamento de carga oferecido pelo próprio provedor de serviços em nuvem.

Inicialmente, siga o procedimento de criação da instância `api1_ec2`, mas desta vez, renomeie para `api2_ec2`, mantendo as configurações e Docker. Depois, execute a imagem da API REST de exemplo, configurando-a para se conectar ao banco `ec2` da mesma maneira. **Volte aqui quando terminar.**

Agora com duas instâncias da API conectadas ao mesmo banco de dados, é necessário configurar o Nginx para distribuir as solicitações entre elas alternadamente, utilizando para isso o algoritmo padrão do Nginx conhecido como **Round Robin**. Para realizar essa configuração, abra o arquivo de configurações do Nginx localizado em `/etc/nginx/sites-available/default` utilizando um editor de texto e configure da seguinte forma:

Exemplo de configuração:

```
ubuntu@ip-10-0-0-54: ~  
  
upstream api_backend {  
    server 10.0.0.152:8080; # Primeira instância da API  
    server 10.0.0.226:8080; # Segunda instância da API  
}  
  
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
  
    # Descomente se precisar servir arquivos estáticos  
    #root /var/www/html;  
    #index index.html index.htm index.nginx-debian.html;  
  
    #server_name _;  
  
    location ^~ / {  
        proxy_pass http://api_backend/  
        proxy_set_header Host $http_host;  
    }  
}
```

"/etc/nginx/sites-available/default" 25L, 555B

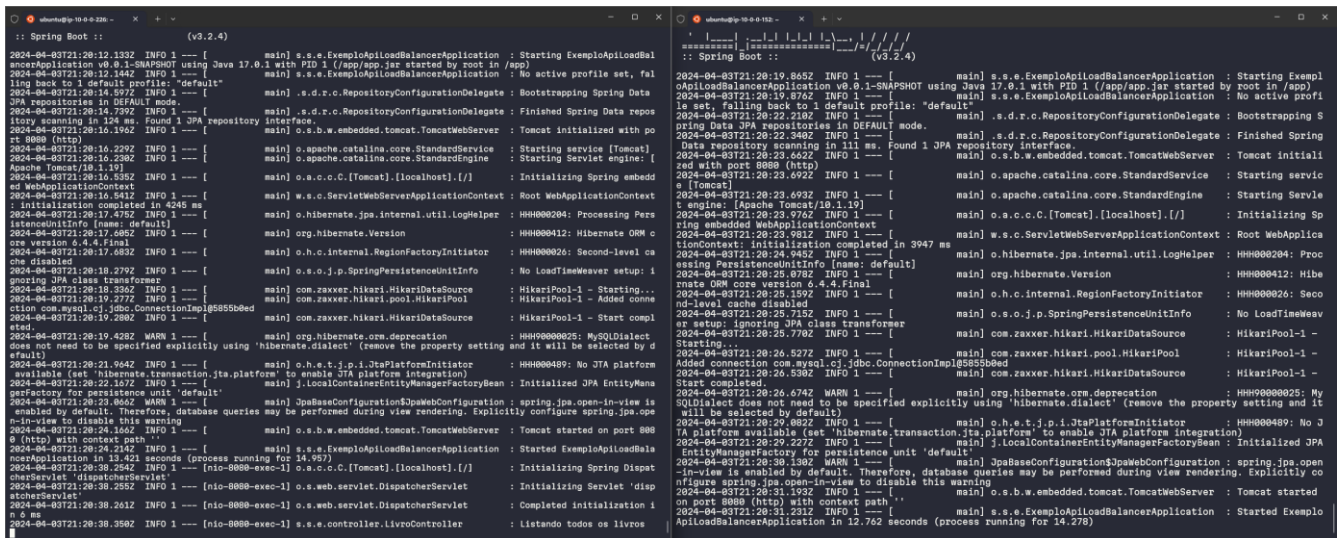
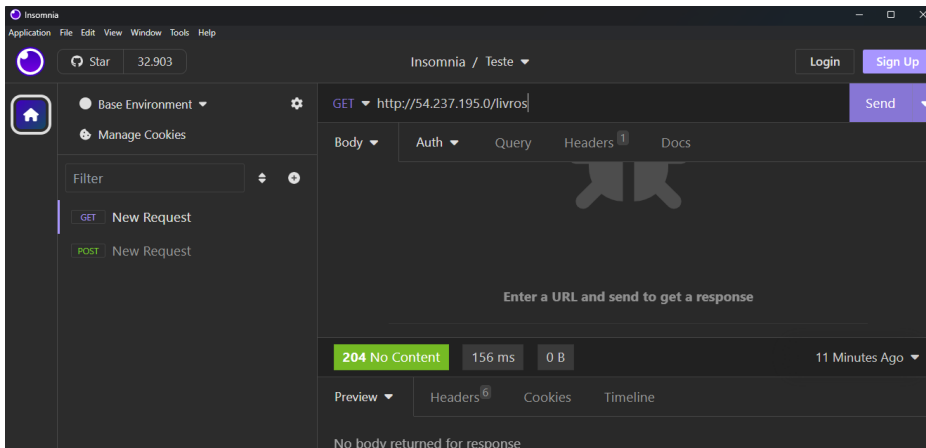
Para copiar:

```
upstream api_backend {  
    server API1-AQUI:8080; # Primeira instância da API  
    server API2-AQUI:8080; # Segunda instância da API  
}  
  
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
    # Descomente se precisar servir arquivos estáticos  
    #root /var/www/html;  
    #index index.html index.htm index.nginx-debian.html;  
    #server_name _;  
    location ^~ / {  
        proxy_pass http://api_backend/;  
        proxy_set_header Host $http_host;  
    }  
}
```

Com essa configuração aplicada, o Nginx alternará o encaminhamento das requisições entre as duas instâncias da nossa API. Para realizar um teste, abra dois terminais e conecte-se às instâncias API1 e API2, respectivamente. Em seguida, execute o comando ***docker logs nome-do-container --follow*** em cada terminal.

Para testar o encaminhamento das requisições, utilize ferramentas como Swagger ou Insomnia, direcionando as requisições para o servidor onde o Nginx está instalado. Ao fazer isso, você poderá observar nos logs como o Nginx distribui as solicitações entre as duas instâncias da API. Veja um exemplo utilizando o Insomnia:

LAB PROXY REVERSO E BALANCEAMENTO DE CARGA



Observe no terminal do lado esquerdo que a requisição GET foi registrada. Ao repetir esse processo várias vezes, você notará que o Nginx continuará alternando as requisições entre as duas instâncias, garantindo assim a distribuição do tráfego.

Parabéns por seguir até este ponto! Para explorar configurações adicionais, confira:

<https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/>