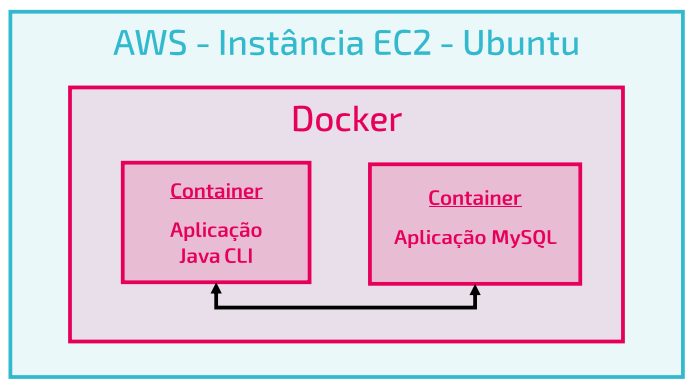
O Docker tem como objetivo criar, testar e implementar aplicações em um ambiente separado da máquina original, chamado de container. Dessa forma, o desenvolvedor consegue empacotar o software de maneira padronizada, pois o Docker é uma plataforma de containers, ou seja, dentro de um Docker pode se haver um ou mais containers com configurações variadas e independentes um do outro.

[Docker | O que é Docker, o que são Containers e como isolar ambientes? (opservices.com.br)](https://www.opservices.com.br/o-que-e-docker/)

Demonstração da estrutura que iremos realizar:

Dentro de uma instância EC2 teremos o Docker instalado e criaremos dois containers independentes um do outro (um com a aplicação java e outro com o banco MySQL) e faremos ambos se comunicarem para trocar informações.



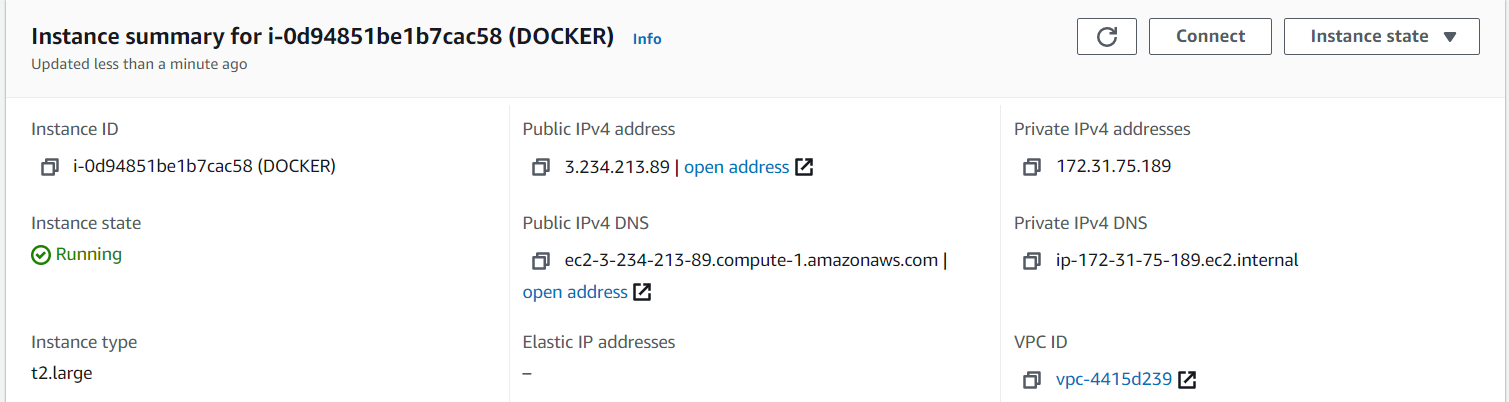
Passo 1:

Criar uma instância EC2.

Material de apoio:

[Material EC2 Marise](https://github.com/diegocrocha/10marise/blob/main/CRIA%C3%87%C3%83O%20INST%C3%82NCIA%20EC2.pdf)

Já temos a instância EC2 criada:



Utilizamos uma instância do tipo t2.large, que mesmo sendo um pouco mais cara, traz uma eficiência maior na utilização do docker com dois ou mais containers.

Dica: quando forem mudar os tipos de instancias sempre consultem os valores e fiquem ligados nisto para não gastarem todos seus créditos de forma indevida

Link para consultar valores: [Amazon EC2 tipos de instâncias - AWS](https://aws.amazon.com/pt/ec2/instance-types/)

Passo 2:

Instalar dentro da instância EC2 o Docker e criar um container com MySQL

Material de apoio:

[Criação de Banco no docker Marise](https://github.com/diegocrocha/10marise/blob/main/DOCKER%20COM%20BANCO.pdf)

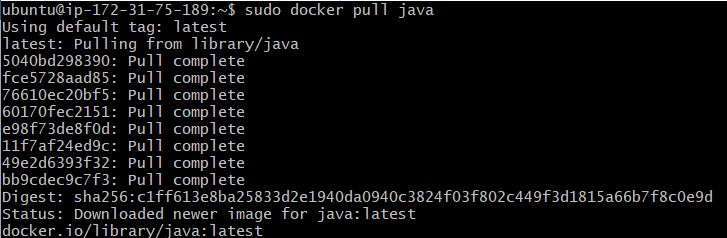
Importante: Vá somente até o ponto de “SHOW DATABASES;”

Passo 3:

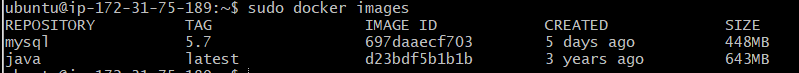
Criar um segundo docker para hospedar a nossa aplicação em java

Primeiramente iremos instalar a imagem java para utilizar no container

sudo docker pull java



sudo docker images



Vamos criar o container que conterá o java já fazendo um link com o container que há o MySQL, para isso precisamos pegar o name ou id do container MySQL

sudo docker ps



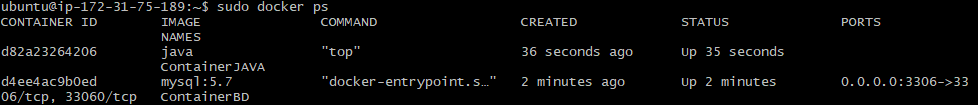
sudo docker run –name “NomeContainer” –link “NomeContainerLink”:”NomeImagemLink” –t –d “imagemContainer” top

sudo docker run --name ContainerJAVA --link ContainerBD:mysql -t -d java top



Verificar se foi criado o container

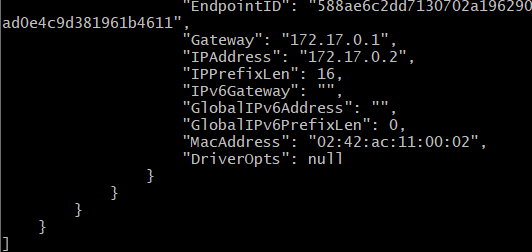
sudo docker ps



Passo 4:

Agora para verificar se a conexão foi realizada precisamos pegar o ip do container que foi linkado (neste caso o ContainerBD)

sudo docker inspect ContainerBD



Procurar pelo IPAddress. OBS: GUARDAR ESTE IP, POIS UTILIZAREMOS POSTERIORMENTE.

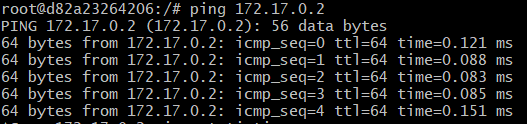
Entrar no Container que contem o JAVA com o comando:

sudo docker exec -it ContainerJAVA bash



Após entrar no container JAVA verificaremos se o link entre eles esta funcionando, utilizando o comando ping + IPAddress:

ping 172.17.0.2



Dar ctrl + c para parar o comando

Com isto nossos container estão linkados um ao outro, prontos para compartilhar informações entre si.

Importante: Dentro do contêiner JAVA será necessária a instalação de algumas dependências dos JAVA. [Utilize este material de apoio disponibilizado pela Marise.](https://github.com/diegocrocha/10marise/blob/main/dependenciaJava.pdf) (comece no 6 e vá até o 11).

Passo 5:

Agora que os containers estão prontos para compartilhar informações, vamos criar uma tabela no container do mysql.

Sendo assim saímos do container atual utilizando o comando:

exit

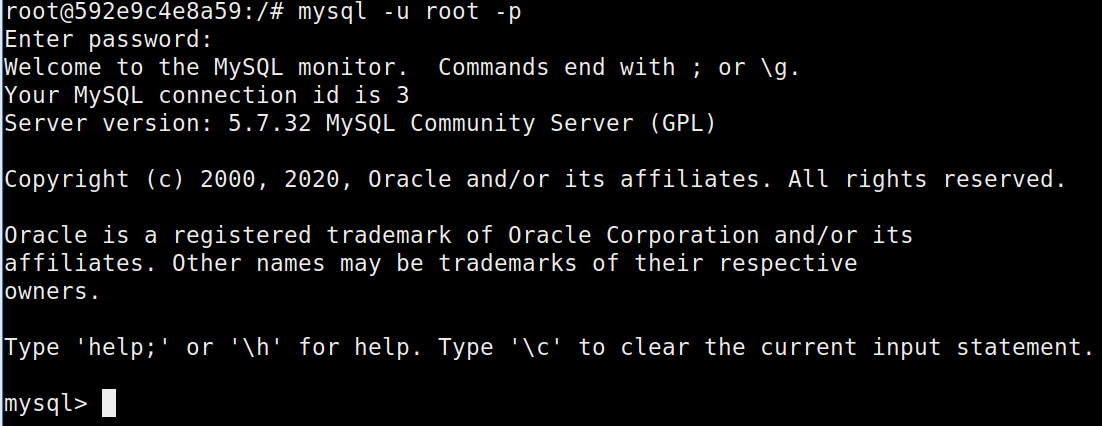


E entramos no Container mysql:

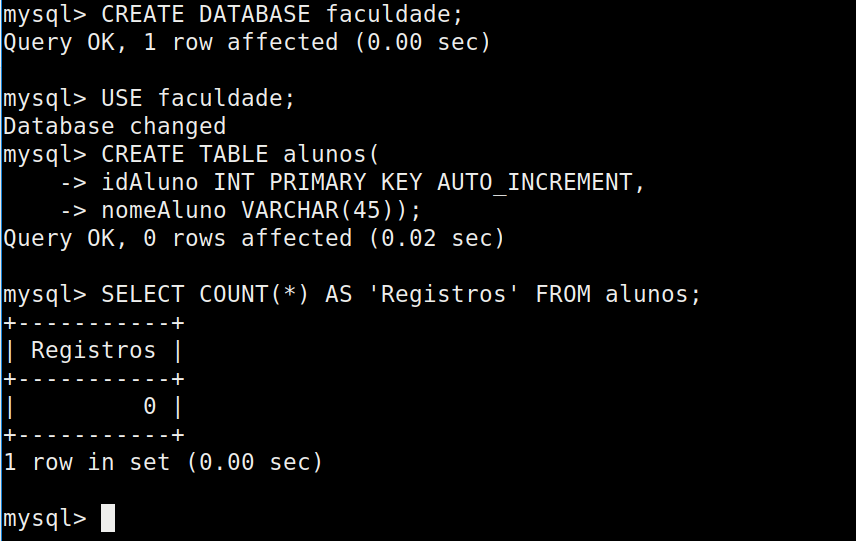
sudo docker exec -it ConteinerBD bash

Para se conectar no mysql que está dentro do container via linha de comando é usado:

mysql –u root –p



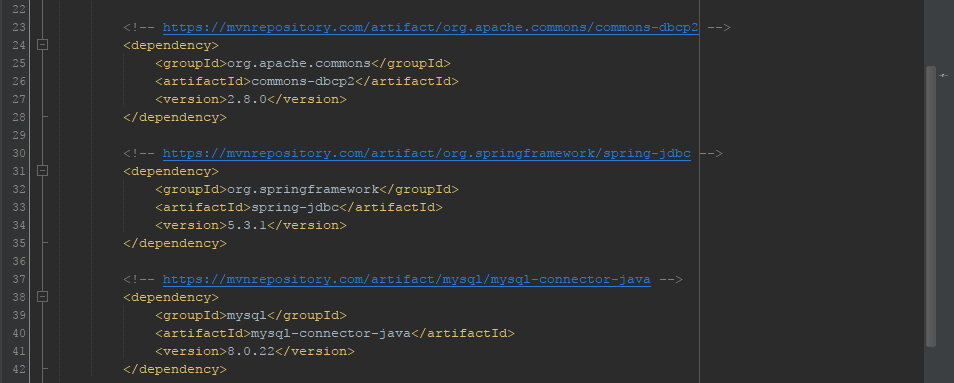
Após estar conectado ao mysql vamos criar um database e uma tabela simples para a inserção.  
Importante: Lembre do nome do database e da tabela pois serão utilizados no arquivo JAVA de insert.



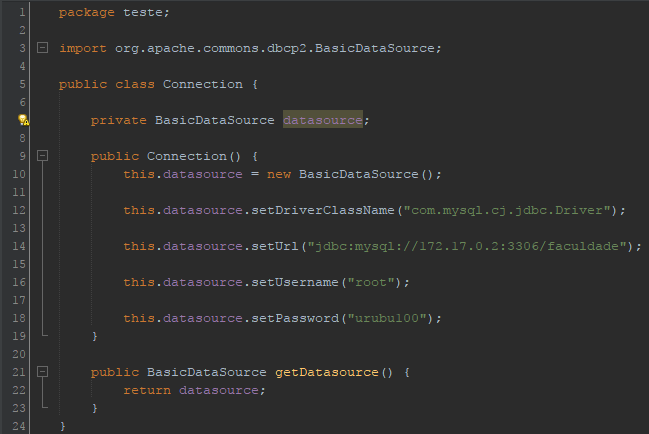
Passo 6:

Agora que vimos que os nossos containers estão criados, linkados e o mysql está configurado, vamos criar a classe de conexão e a classe de inserção com base no projeto disponibilizado pelo professor Diego.

[Material original do Professor Diego.](https://github.com/diegocrocha/10marise/tree/main/projetoDiego/banco-teste)

Abra seu NetBeans, vá até o pom.xml e insira as seguintes dependências:  


Logo após crie uma classe e insira este código:



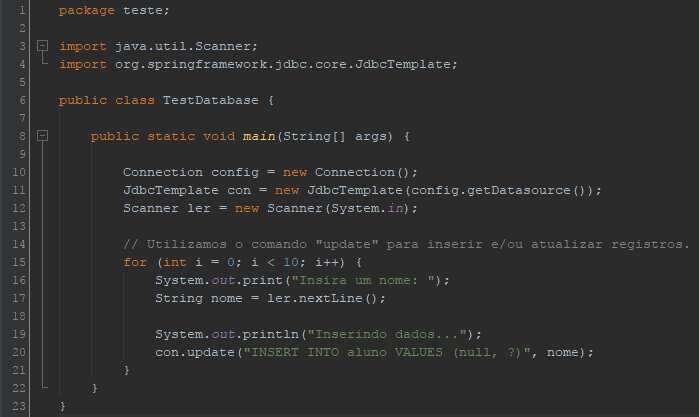
No código acima criamos um atributo do tipo BasicDataSource que irá nos auxiliar na criação de conexão com o Banco de Dados.

Logo após criamos o nosso construtor e nele criamos um objeto do tipo “BasicDataSource” para utilizarmos seus métodos.

Os métodos utilizados são:

* setDriverClassName: neste método informamos qual drive vamos utilizar para fazer a conexão.
* setUrl: aqui informamos a string de conexão o ipAdress de onde está o nosso banco de dados, a porta utilizada pelo banco de dados e qual database iremos usar.
* setUsername: Passamos o nome do usuário que tem permissão para acessar o database que você informou
* setPassword: Informamos a senha do usuário escolhido.

Depois de criar a classe de conexão vamos criar uma classe main que será executada e irá utilizar a classe de conexão para se conectar ao banco de dados.



Então, depois de criar as classes precisamos fazer um último ajuste.

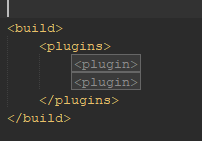
Vamos criar o .jar da nossa aplicação.

Primeiro um ponto de atenção:

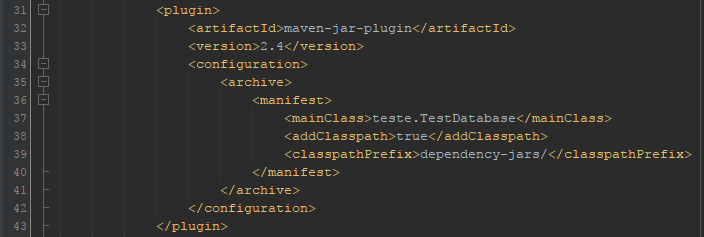
* A sua classe main **NÃO** pode ser um jFrame. Qual o motivo disso?  
  Este arquivo será utilizado dentro de um docker, ou seja, inteiramente terminal, não possui interface gráfica. Se você tentar executar algo que utilize uma interface gráfica como um JFrame, você irá receber o erro “X11”, que é um erro específico de quando você não possui um interpretador de interface gráfica. Para não tomar este erro, utilize uma classe main sem JFrame ou algo do tipo.

Agora que temos noção do que poderia nos impedir vamos adicionar o plugin.

Embaixo de “properties” adicione esses tags:

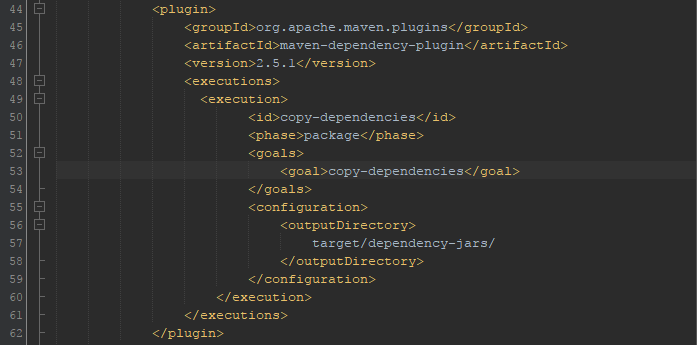


E dentro de “plugins” adicione isto:

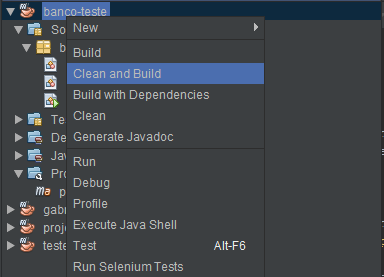


Importante: em “mainClass” você deve informar sua classe main, ou seja, a classe que terá a inserção no banco de dados.

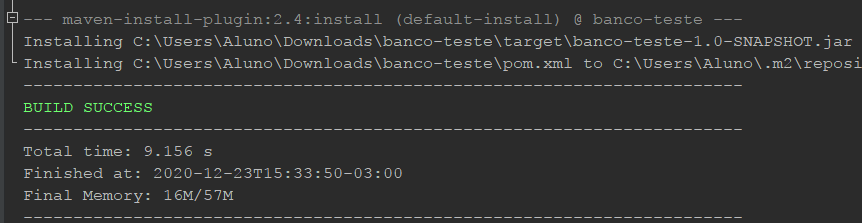
Logo abaixo do plugin anterior adicione este outro:



Agora está quase tudo pronto, só precisamos dar “clean and build”.



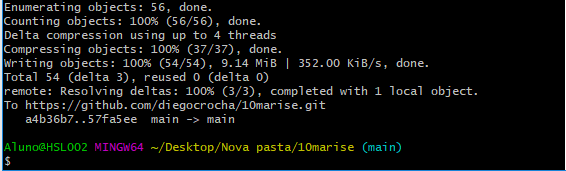
Clique com o botão direito no nome do seu projeto e clique em “clean and build”, assim ele irá gerar o nosso .jar, ou seja, o nosso java executável.

Se tudo tiver ocorrido certo, receberá “BUILD SUCCESS” no terminal e acima dele você poderá verificar seu .jar criado.  


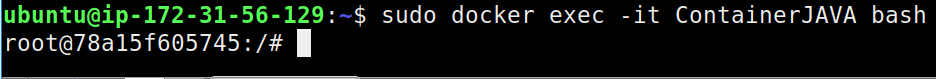
No decorrer da segunda linha você consegue verificar a criação do arquivo .jar.

Passo 7:

Suba seu repositório no GitHub.

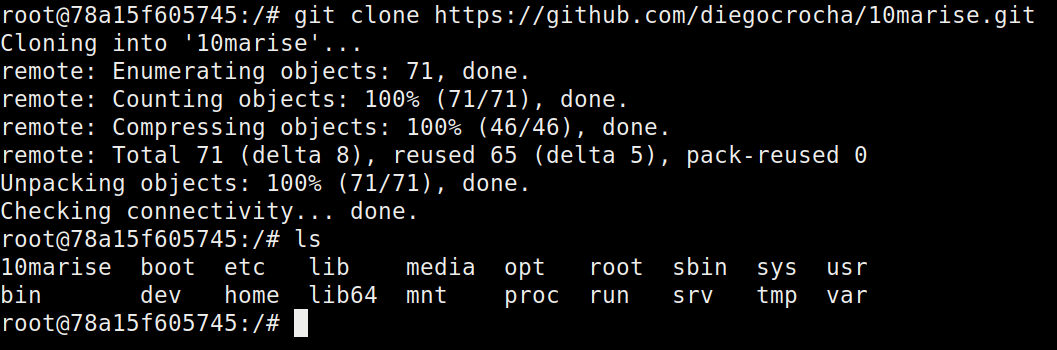


Agora entre no seu ContainerJAVA.

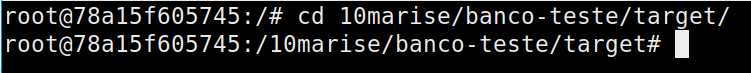


Conectado você irá fazer o clone do repositório e logo após verificar se ele está lá com sucesso.

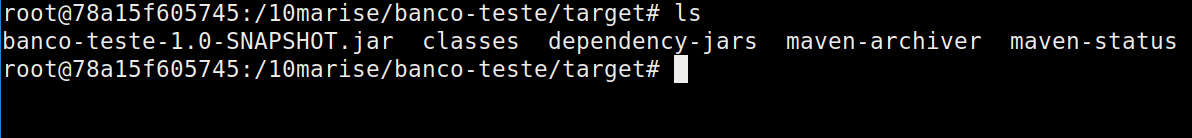
Clone: git clone “link repositório”  
Verificação: ls



Entre dentro do repositório e vá até o diretório “target”.

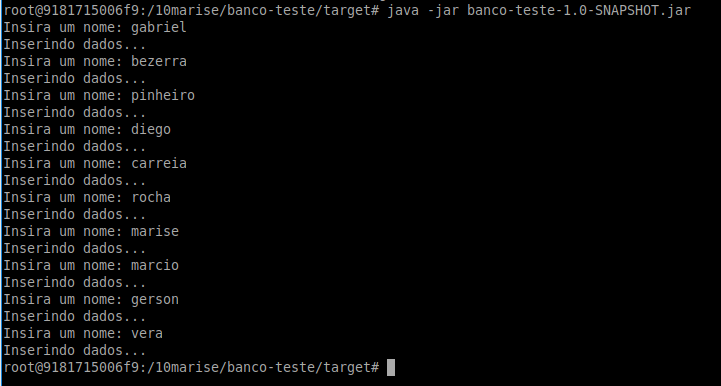


Na “target” dê um ls e você conseguira observar o seu arquivo .jar.



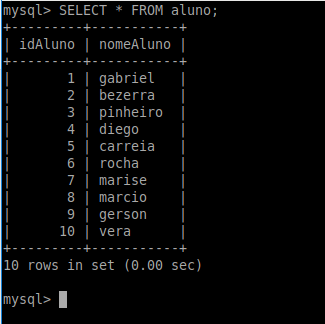
Agora vamos executar o seu java. Para isso utilize:

java –jar “nomeDoArquivo” (Que no caso é banco-teste-1.0-SNAPSHOT.jar)



Quando executamos o nosso arquivo jar com o comando acima ele rodou normalmente, não apresentou nenhum erro, ou seja, inseriu corretamente no banco de dados do outro contêiner. Bora conferir?

Abra o Conteiner mysql e dê um select para verificar se todos os dados que acabamos de inserir estão lá.



Como podemos ver os dados estão lá corretamente.

Passo Final: CONCLUSÃO

Este arquivo foi feito com o intuito de auxiliar os alunos dos semestres posteriores ao segundo semestre de 2020. Com este material vocês já conseguem criar diversos links entre contêiners, dessa forma agora é possível criar uma estrutura bem mais robusta e não se limitar somente a uma conexão entre o mysql e a sua aplicação. Espero que tenhamos ajudado, bons estudos.