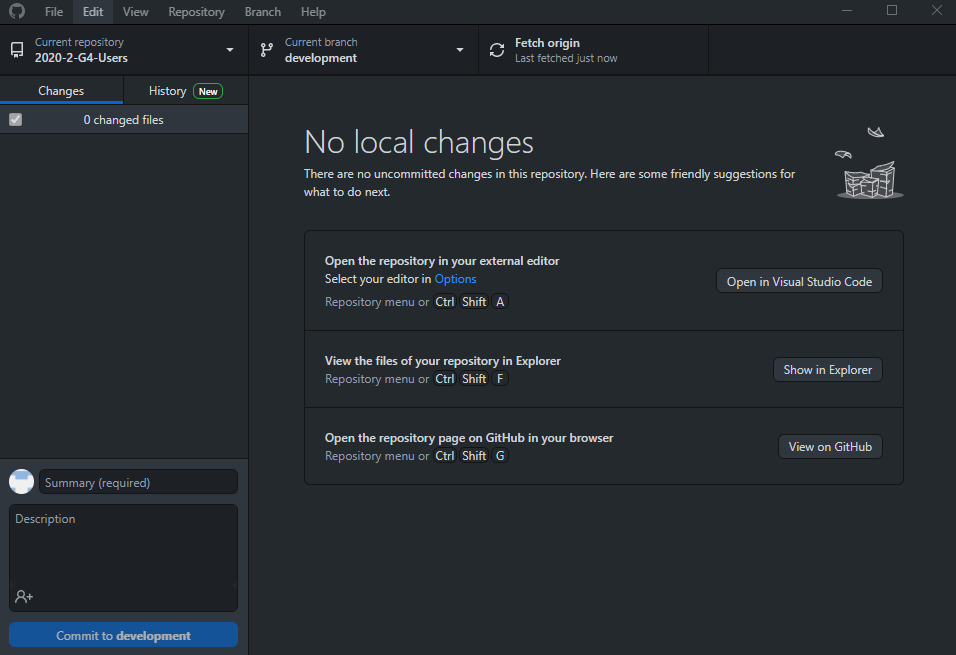
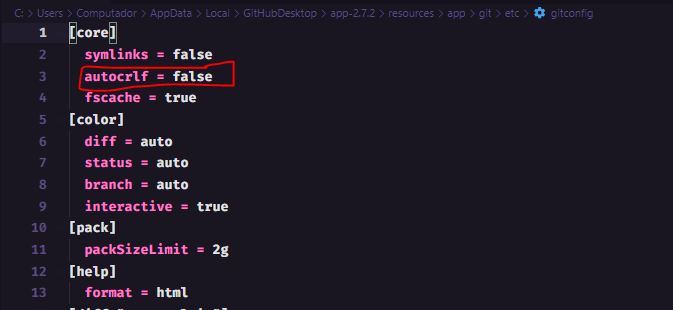
Instanciar a aplicação SIGeD – Windows 10

Sistema de Gerenciamento de Demandas

1 – Inicialmente deve-se instalar o GITHUB DESKTOP



3 – Localizar o arquivo **gitconfig** no diretório de instalação do GITHUB DESKTOP **\GitHubDesktop\app-2.7.2\resources\app\git\etc** no sistema de arquivos do Windows (observar que seu editor de texto deve salvar o arquivo com quebra de linha no formato LF (UNIX) )



Colocar nas propriedades do [core] o item **autocrlf = false**, este procedimento impede o GITHUB de modificar a quebra de linha nos arquivos do projeto para o padrão do Windows (CRLF)

4 – Agora, deve-se fazer o CLONE do projeto na máquina servidora

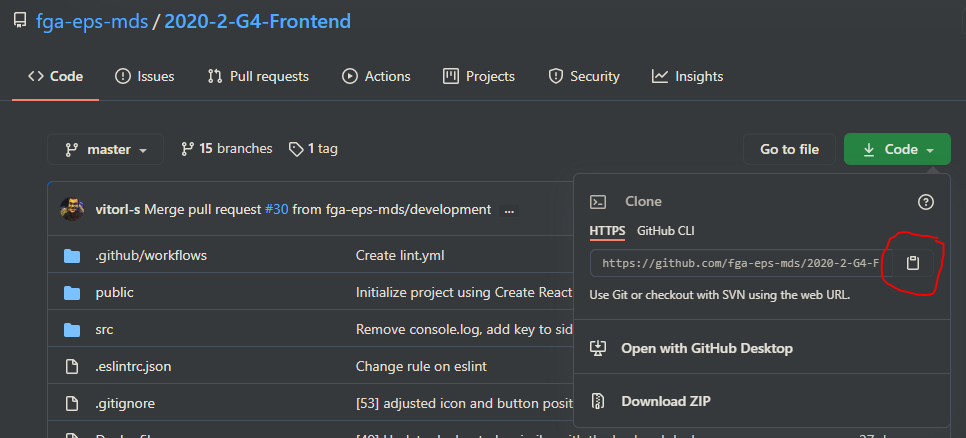
<https://github.com/fga-eps-mds/2020-2-G4-Frontend>

<https://github.com/fga-eps-mds/2020-2-G4-Users>

<https://github.com/fga-eps-mds/2020-2-G4-Clients>

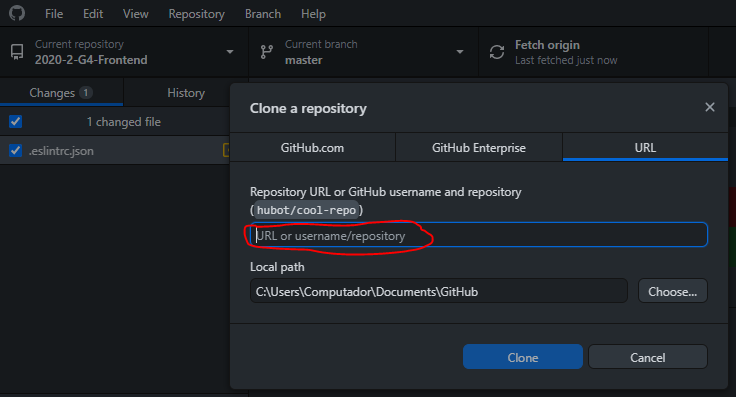
<https://github.com/fga-eps-mds/2020-2-G4-Demands>

<https://github.com/fga-eps-mds/2020-2-G4-Sectors>



Copie a URL do repositório

5 – Utilizando o GITHUB DESKTOP (Utilize a branch Development, pois está mais atualizada)



Informe o Local onde será colocado o projeto na máquina servidora

e cole o endereço do repositório (repetir o procedimento para todos os projetos)

6 – Prepara o Windows 10 para trabalhar com uma máquina virtual LINUX

Antes de instalar qualquer distribuição do Linux no Windows, você precisará primeiro habilitar o recurso opcional "Subsistema do Windows para Linux".

Abra o **PowerShell como administrador** e execute:

**C:\> dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart**

Para atualizar para o WSL 2, você precisa estar executando o Windows 10.

* Para sistemas x64: Versão 1903 ou superiores, com o Build 18362 ou superiores.
* Para sistemas ARM64: Versão 2004 ou superiores, com o Build 19041 ou superiores.
* Os builds inferiores a 18362 não dão suporte a WSL 2. Use o Assistente do Windows Update para atualizar a sua versão do Windows.

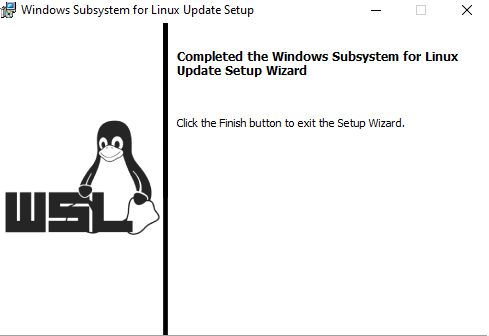
Para verificar a sua versão e o número de build, selecione a tecla do logotipo do Windows + R, digite winver e selecione OK. (Ou digite o comando ver no prompt de comando do Windows). Atualize para a versão mais recente do Windows no menu Configurações.

Reinicie o computador para concluir a instalação do WSL e a atualização para o WSL 2.

**C:\> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart**

Agora, Baixe o pacote de atualização do kernel do Linux

**https://wslstorestorage.blob.core.windows.net/wslblob/wsl\_update\_x64.msi**

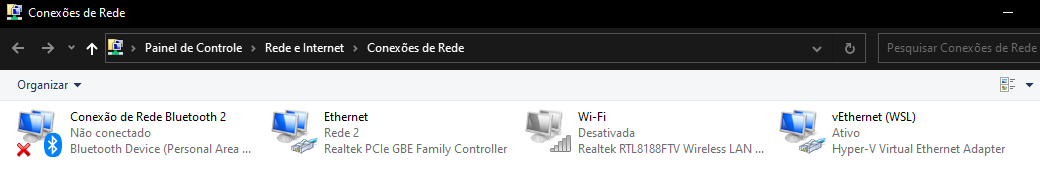


Execute o pacote de atualização baixado na etapa anterior. (Clique duas vezes para executar. Você receberá uma solicitação para fornecer permissões elevadas; selecione 'sim' para aprovar essa instalação.

Abra o PowerShell e execute este comando para definir o WSL 2 como a versão padrão ao instalar uma nova distribuição do Linux:

**C:\> wsl --set-default-version 2**

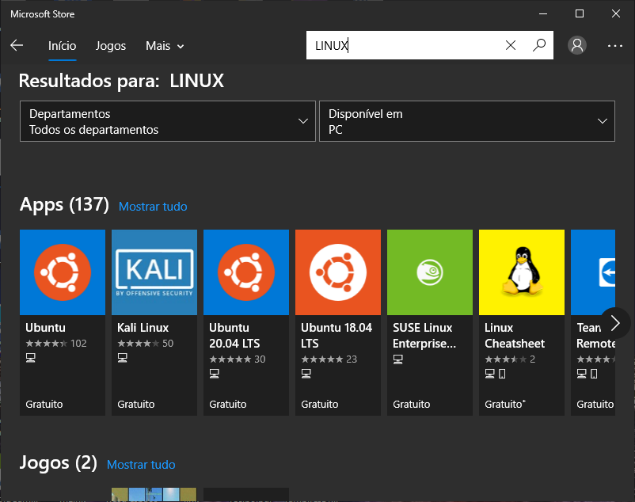
Definir DNS primario 8.8.8.8 e DNS secundário para o adaptador de rede (WSL)



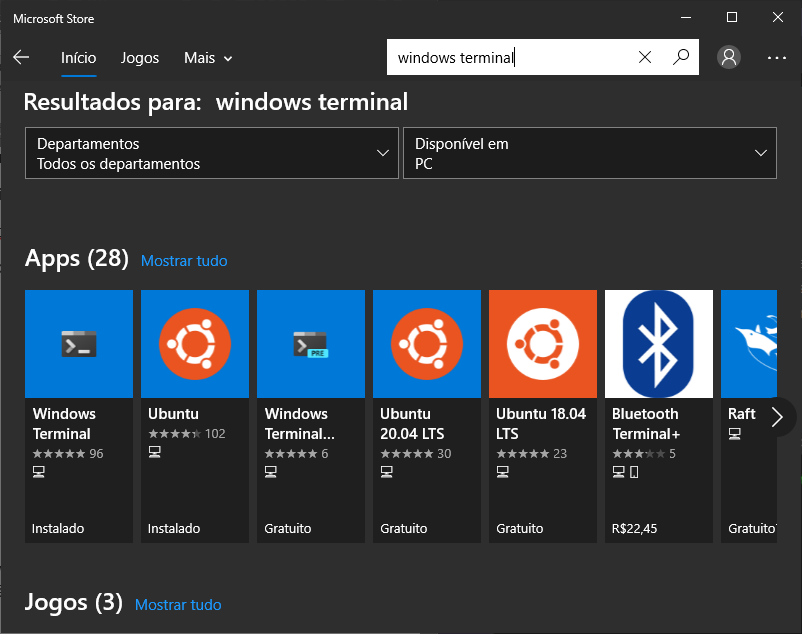
Preferencialmente incluir DNS em todos os adaptadores de rede que possua em seu computador

7 – Instalar uma distribuição LINUX

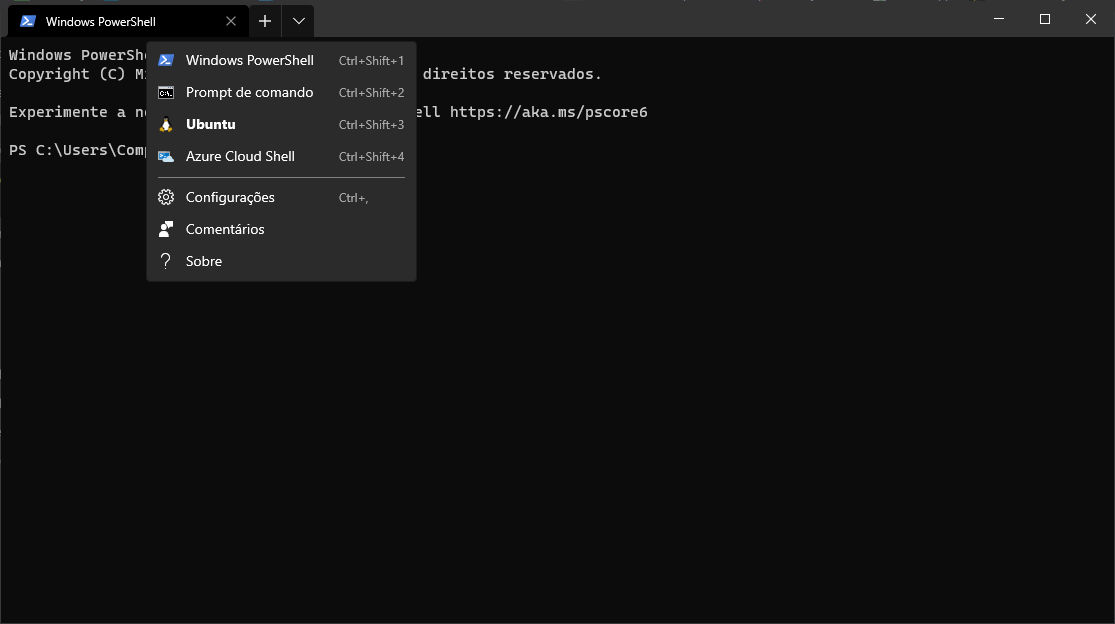
Abra a Microsoft Store e escolha sua distribuição do Linux favorita, e instale:



Instale também o WINDOWS TERMINAL, permite que se trabalhe com mais de um terminal separado por guias e também para sistemas operacionais diferentes



Execute o WINDOWS TERMINAL e escolha no menu a distribuição Linux instalada



Na primeira vez que você iniciar uma distribuição do Linux recém-instalada, uma janela de console será aberta e será solicitado que você aguarde um ou dois minutos para que os arquivos sejam descompactados e armazenados em seu PC. Todas as futuras inicializações deverão levar menos de um segundo.

Em seguida, você precisará criar uma **conta de usuário e uma senha** para sua nova distribuição do Linux.

8 – Instale o DOCKER DESKTOP e DOCKER-COMPOSE no Linux Ubuntu 20

**DOCKER DESKTOP**

Primeiro, atualize sua lista existente de pacotes:

**$ sudo su**

Informe a senha de root

**$ apt update**

Em seguida, instale alguns pacotes pré-requisito que deixam o apt usar pacotes pelo HTTPS:

**$ apt install apt-transport-https ca-certificates curl software-properties-common**

Então, adicione a chave GPG para o repositório oficial do Docker no seu sistema:

**$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**

Adicione o repositório do Docker às fontes do APT:

**$ add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"**

Em seguida, atualize o banco de dados do pacote com os pacotes do Docker do recém adicionado repositório:

**$ apt update**

Certifique-se de que você está prestes a instalar do repositório do Docker ao invés do repositório padrão do Ubuntu:

**$ apt-cache policy docker-ce**

Você verá um resultado assim, embora o número da versão para o Docker possa ser diferente:

Output of apt-cache policy docker-ce

docker-ce:

Installed: (none)

Candidate: 5:19.03.9~3-0~ubuntu-focal

Version table:

5:19.03.9~3-0~ubuntu-focal 500

500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages

Observe que o docker-ce não está instalado, mas o candidato para a instalação é do repositório do Docker para o Ubuntu 20.04 (focal).

Finalmente, instale o Docker:

**$ apt install docker-ce**

O Docker deve agora ser instalado, o daemon iniciado e o processo habilitado a iniciar no boot. Verifique se ele está funcionando:

**$ systemctl status docker**

O resultado deve ser similar ao mostrado a seguir, mostrando que o serviço está ativo e funcionando:

Output

● docker.service - Docker Application Container Engine

Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)

Active: active (running) since Tue 2020-05-19 17:00:41 UTC; 17s ago

TriggeredBy: ● docker.socket

Docs: https://docs.docker.com

Main PID: 24321 (dockerd)

Tasks: 8

Memory: 46.4M

CGroup: /system.slice/docker.service

└─24321 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Instalando o Docker agora não dá apenas o serviço do Docker (daemon), mas também o utilitário de linha de comando docker, ou o cliente do Docker.

Caso não tenha iniciado o daemon, faça:

**$ dockerd**

Se tiver problema de permissão, faça:

**$ usermod -aG docker your-user**

Você será solicitado a digitar a senha do seu usuário para continuar.

Confirme que seu usuário agora está adicionado ao grupo docker digitando:

**$ id -nG**

Output

sammy sudo docker

Se você precisar adicionar um usuário ao grupo docker com o qual você não está logado, declare esse nome de usuário explicitamente usando:

**$ usermod -aG docker username**

**DOCKER-COMPOSE**

**$ curl -L "https://github.com/docker/compose/releases/download/1.29.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose**

Em seguida, defina as permissões corretas para que o comando docker-compose seja executável:

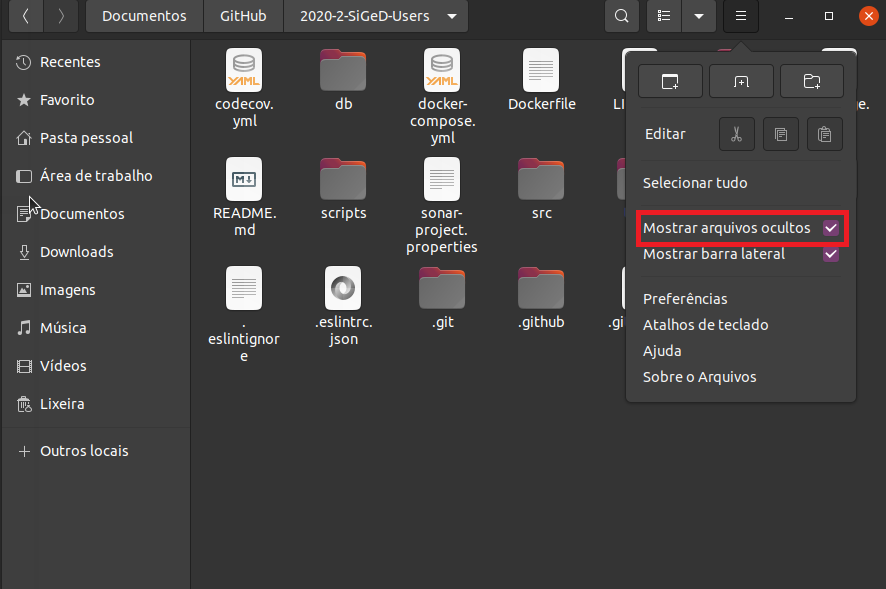
**$ chmod +x /usr/local/bin/docker-compose**

Para verificar se a instalação foi bem-sucedida, execute:

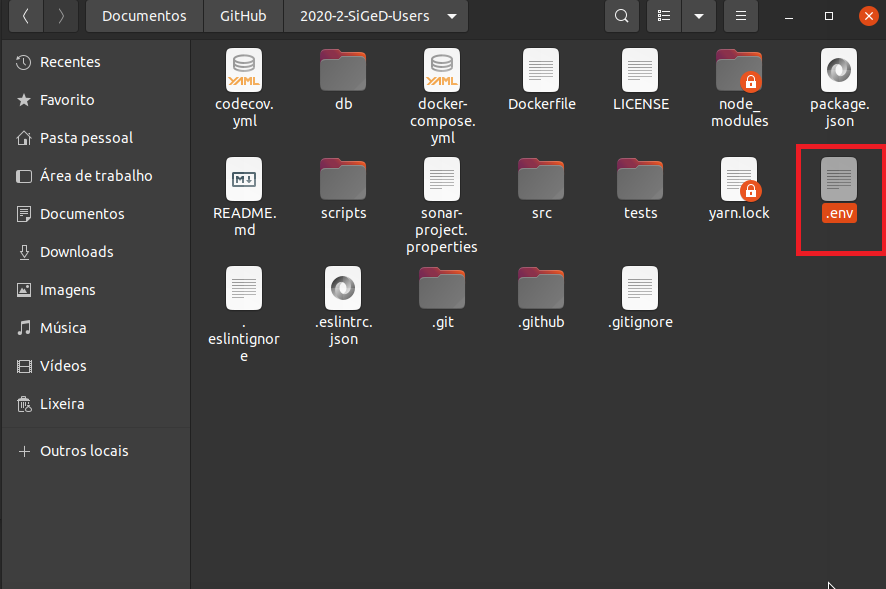
**$ docker-compose --version**

**Preparação para gerar as imagens**

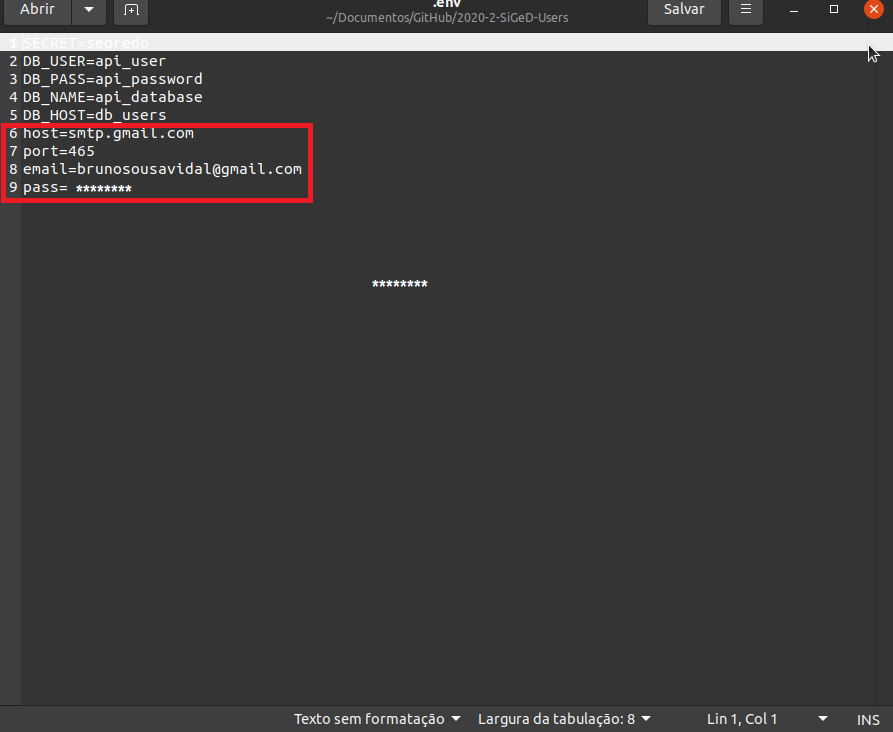
Antes de gerar o container docker do microserviço Users, deve-se configurar o email que será utilizado para envio das novas senhas para os usuários do sistema. Inicialmente pede-se para mostrar os arquivos ocultos do diretório.



Edite o arquivo .env para realizar as alterações



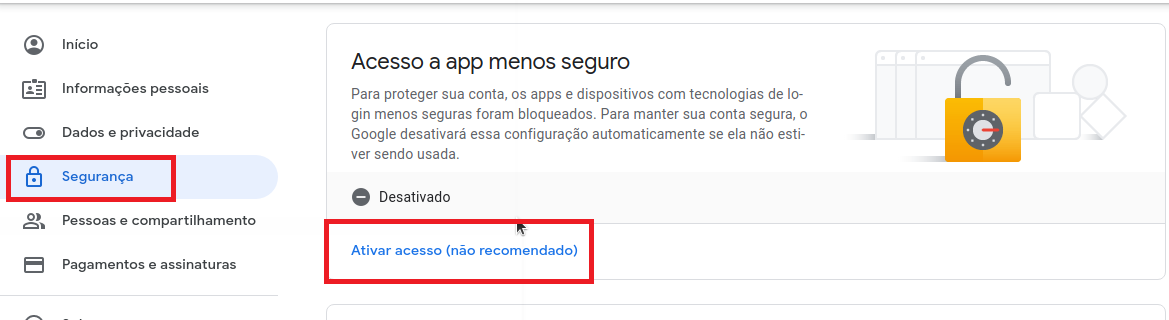
Caso seja uma conta de gmail, basta manter o protocolo e porta



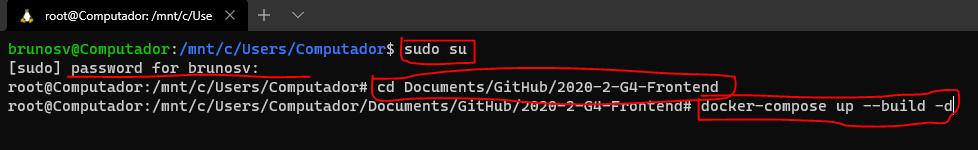
Em seguida, no navegador, peça para gerenciar sua conta email, para permitir acesso pelo aplicativo externo na sua conta de email, desabilitando a segurança.



Permitir acesso ao app menos seguro, isso fará com que o SiGeD consiga enviar email utilizando essa conta.



9 – Criando as images do projeto no DOCKER



**$ sudo su**  - permite que você trabalhe sempre como superusuário root

**$ cd pasta-onde-esta-o-projeto** – entre no diretório onde se encontra o projeto que será utilizado para a criação da image do docker (lembrar que no linux a barra de separação das pastas não é a barra invertida do windows)

**docker network create siged\_backend –** Para criar a rede que irá possibilitar a comunicação entre as maquinas dos micro serviços

**docker-compose up --build -d**  - este comando irá construir uma “image” do projeto no repositório do DOCKER

Repita o processo para todos os projetos

|  |  |
| --- | --- |
| # cd /GitHub/2020-2-G4-Users  # docker-compose up --build -d | # cd /GitHub/2020-2-G4-Demands  # docker-compose up --build -d |
| # cd /GitHub/2020-2-G4-Clients  # docker-compose up --build -d |  |

No DOCKER DESKTOP, você poderá utilizar os seguintes comandos para visualizar as images e instancias das images

Para acompanhar o log de um serviço no docker

**$ docker logs -f {microserviço}**

Para incluir no repositorio do docker hub

**$ docker image tag 2020-2-g4-frontend\_frontend:latest brunosv/2020-2-g4-frontend\_frontend:v1**

**Para baixar do repositorio do docker hub**

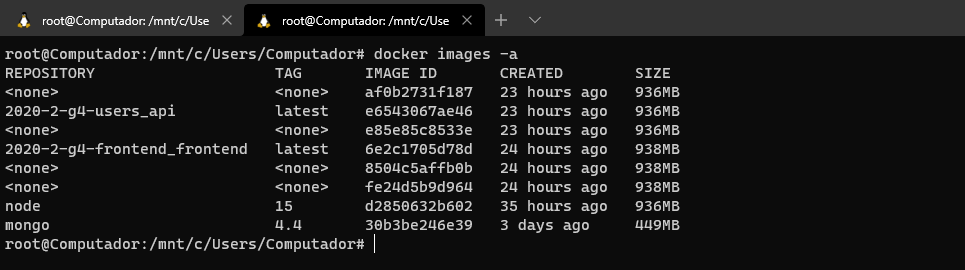
**$ docker login  usuario e senha**

**$ docker images ls**

**$ docker container run -it brunosv/2020-2-g4-frontend\_frontend:v1 /bin/bash**

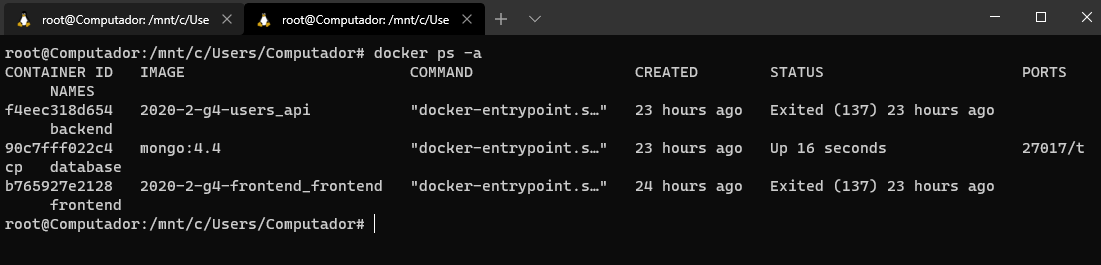
Para listar as images contidas no DOCKER

**$ docker images –a**



Para listar as instancias do container

**$ docker ps –a**



Caso queira criar uma nova instancia do banco de dados mongo:4.4 sem autenticação

**$ docker run -d -e AUTH=no mongo:4.4**

Para criar uma nova instancia de uma image incluindo usuário root na criação da instancia

**$ docker run -d --name database -e MONGO\_INITDB\_ROOT\_USERNAME=root -e MONGO\_INITDB\_ROOT\_PASSWORD=password -e MONGO\_INITDB\_ROOT\_DATABASE=root-db mongo:4.4**

Para apagar as images intermediarias no processo de construção

**$ docker image prune**

Caso queira Iniciar um instancia já criada

**$ docker start database**

Caso queira Parar um instancia já criada

**$ docker stop database**

Instale o mongodb-clients

**$ apt install mongodb-clients**

Para executar um container com o superusuário

**$ docker exec -u 0 -it nameContainer whoami**

Para ter acesso ao terminal de uma instacia de image no docker com superusuário

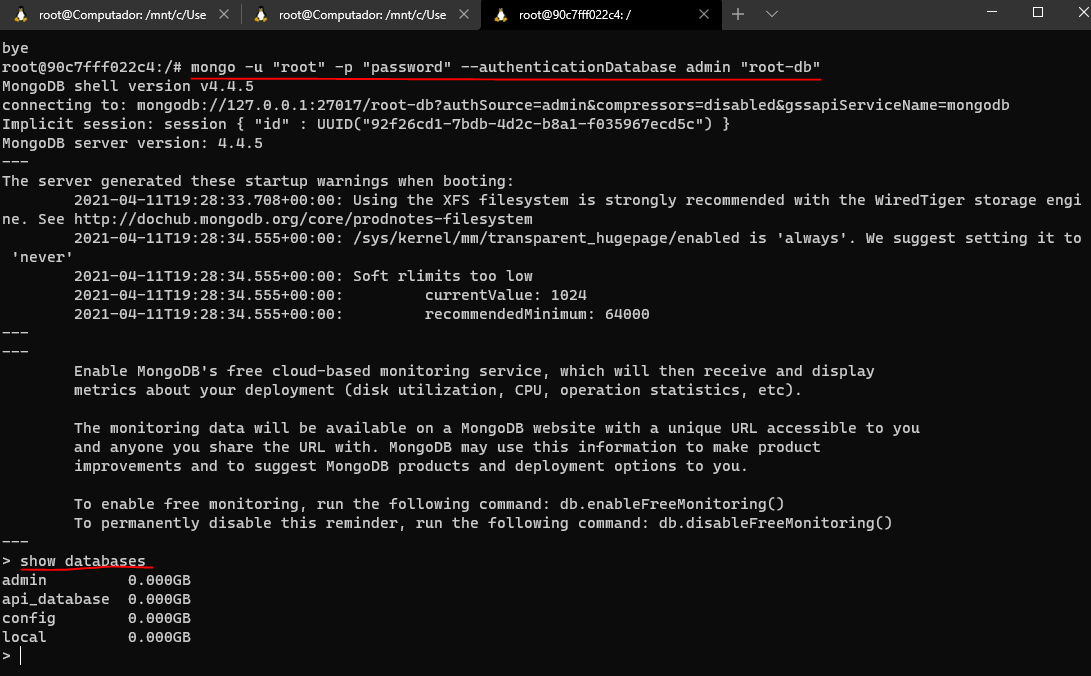
**$ docker exec -u 0 -it database /bin/bash**

Para acessar o banco de dados root-db contido no host database com o usuário root e senha password

**# mongo -u "root" -p "password" --authenticationDatabase admin "root-db"**

**> show database**

**> use admin**



Verifica se o usuário root tem todas as permissões

**> db.getUser("root", {showCredentials: true, showPrivileges: true, showAuthenticationRestrictions: true})**

Caso não exista

**> db.createUser({user:"root", pwd:"password", roles:[{role: "userAdminAnyDatabase", db:"root-db"},{role: "userAdmin", db:"root-db"},{role: "readWrite", db:"root-db"},{role: "dbAdmin", db:"root-db"},{role: "clusterAdmin", db:"root-db"},{role: "readWriteAnyDatabase", db:"root-db"},{role: "dbAdminAnyDatabase", db:"root-db"}]})**

Atribuir mais privilégios na conta root

**> db.updateUser( "root", { roles : [{role: "userAdminAnyDatabase", db:"root-db"},{role: "userAdmin", db:"root-db"},{role: "readWrite", db:"root-db"},{role: "dbAdmin", db:"root-db"},{role: "clusterAdmin", db:"root-db"},{role: "readWriteAnyDatabase", db:"root-db"},{role: "dbAdminAnyDatabase", db:"root-db"}] } )**

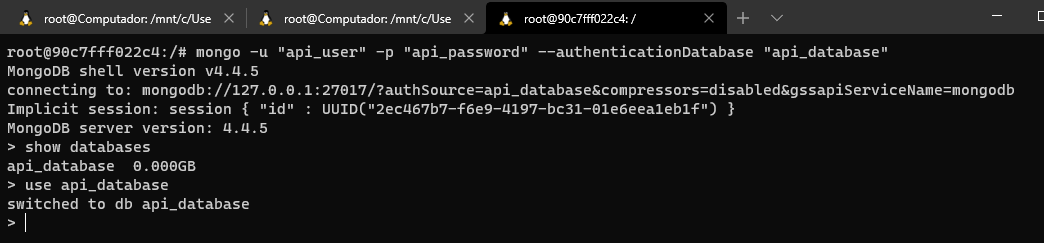
**> exit**

Para acessar o banco de dados api\_database contido no host database com o usuário api\_user e senha api\_password

**# mongo -u "api\_user" -p "api\_password" --authenticationDatabase "api\_database"**

**> show database**

**> use api\_database**



Verifica se o usuario api\_user tem todas as permissões

**> db.getUser("api\_user", {showCredentials: true, showPrivileges: true, showAuthenticationRestrictions: true})**

Caso não exista

**> db.createUser({user:"api\_user", pwd:"api\_password", roles:[{role:"readWrite", db: "api\_database"}]})**

Listar usuários criados

**> db.User.find()**

Caso queira excluir um usuário

**> db.dropUser('api\_user')**

**> exit > exit**

Para matar a instancia de uma image

**$ docker kill database**

Para excluir uma instancia de uma image

**$ docker rm database**

Para executar o arquivo SH de scripts no mongo para criação de usuários na construção

**# cd docker-entrypoint-initdb.d**

**# mongo < mongo-init.sh**

Para executar comandos pelo bash do container docker

**$ docker container exec database sh -c 'cd /etc && ls'**

Para limpar todas as images e instancias do banco de dados mongodb

**$ docker stop $(docker ps -a)**

**$ docker rm $(docker ps -a -f status=exited -q)**

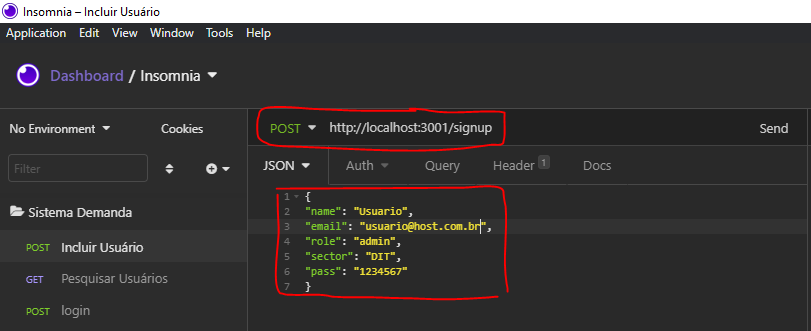
**$ docker rmi $(docker images -a -q)**

Criando um cluster (opcional)

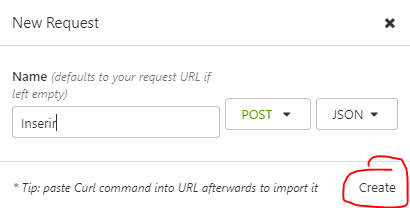
<https://imasters.com.br/data/como-criar-um-cluster-de-mongodb-utilizando-docker>

10 - Instale o Insomnia REST

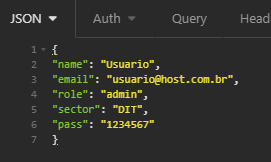
Crie uma requisição com o nome INCLUIR, do tipo POST



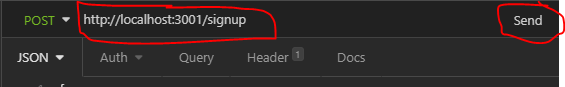
Crie uma requisição com o nome INCLUIR, do tipo POST, utilizando JSON



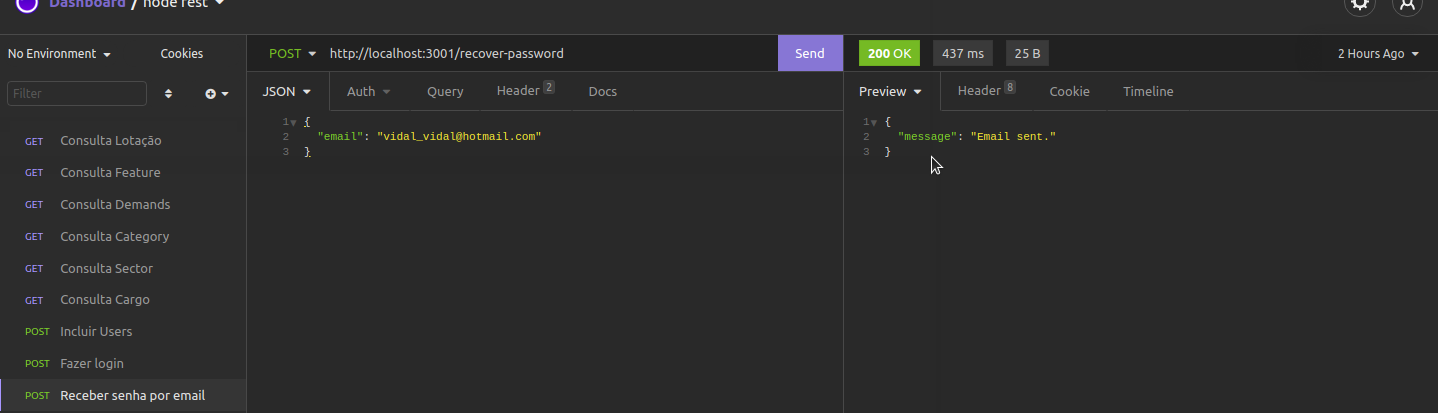
Crie o script JSON para incluir um usuário no banco de dados



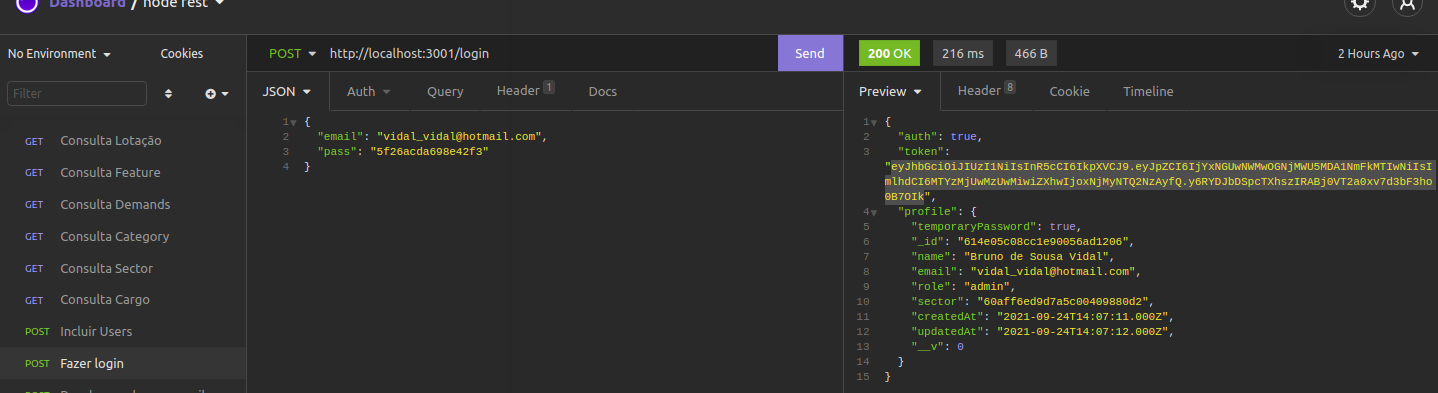
Informe o endereço de acesso ao banco de dados de usuário e envie



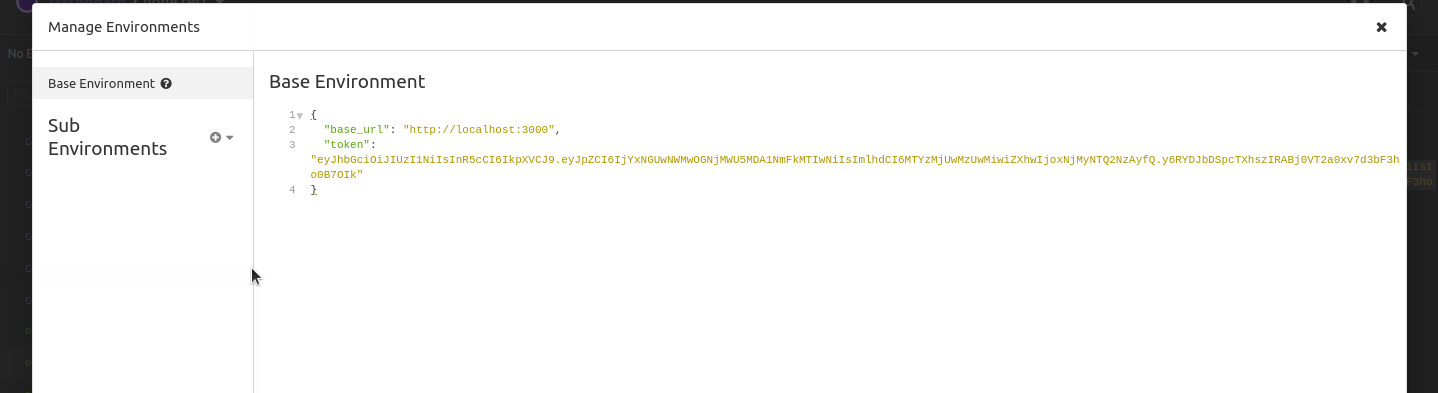
Ainda no insominia, fazer a chamada para o envio de email para a conta de email do usuário cadastrado.



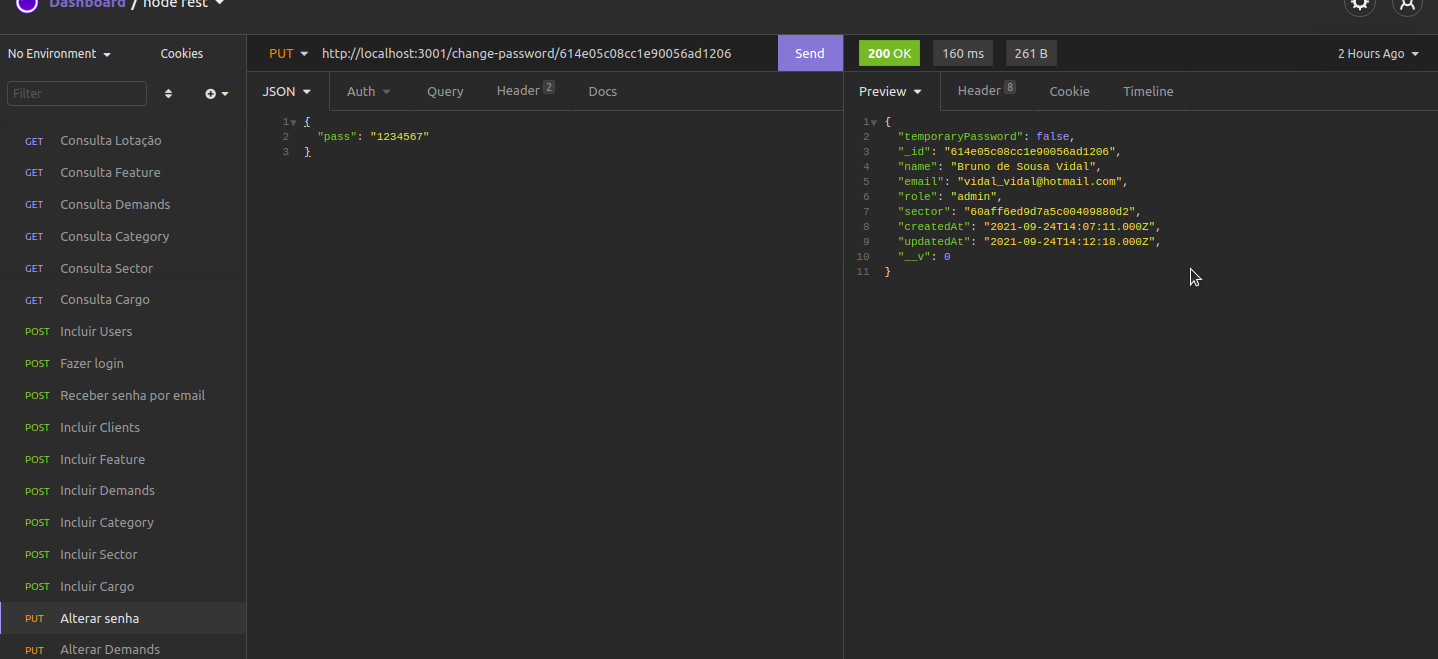
Pegue a senha inicial gerada e encaminhada por email e inclua na chamada para alterar senha.



Copie o token da instancia da aplicação e coloque nas variáveis de ambiente



Por fim, faça a alteração de senha para acesso definitivo ao sistema



Pronto, agora o sistema poderá ser acessado pelo usuário e senha cadastrados.

11 – No navegador [http://localhost:3000](http://localhost:3000/)



Informe o email e senha criada (usuario@host.com.br, 1234567)

12 – Popular o banco de dados MongoDB

$ docker exec -u 0 -it db\_clients /bin/bash

# mongo -u "root" -p "password" --authenticationDatabase admin "root-db"

Crie um arquivo texto e salve com a extensão ( arquivo.js )

show databases

use clients\_database

show collections

db.clients.drop()

db.clients.insert(

[

{

"userID": "614e05c08cc1e90056ad1206",

"name": "Mariana da silva",

"cpf": "12352172548",

"email": "marianasilva@mp.gov.br",

"phone": "995282542",

"secondaryPhone": "12542541",

"address": "Rua B nº 35-AB, Centro",

"office": "614e18607acd2100490329d6",

"active": true,

"location": "614e0cfaad1b71004a89a754",

"history": [{

"userID": "614e05c08cc1e90056ad1206",

"date": "27/05/2021",

"label": "created"

}],

"features": "614e0e77ad1b71004a89a767",

"image": ""

},

{

"userID": "614e05c08cc1e90056ad1206",

"name": "Paula Regina",

"cpf": "12682542548",

"email": "paularegina@mpf.gov.br",

"phone": "995282542",

"secondaryPhone": "12542541",

"address": "Rua B nº 35-AB, Centro",

"office": "614e18607acd2100490329d6",

"active": true,

"location": "614e0cfaad1b71004a89a754",

"history": [{

"userID": "614e05c08cc1e90056ad1206",

"date": "27/05/2021",

"label": "created"

}],

"features": "614e0e77ad1b71004a89a767",

"image": ""

}

]

)

No prompt do SO execute:

# mongo < diretorio\_do\_arquivo\arquivo.js

$ docker system prune -a (limpa todas as imagens, conteiners, volumes e redes não utilizadas ou pendentes)

$ docker volume rm volume\_nome (remove volumes)