😄😄 Aplicação Flask + MySQL na Azure 😄😄

Obs.: Testando em VM Debian com Docker

Versão 3 final: 22/mai/2023

Azure: MySQL / ACR / ACI

Objetivos:

- Clonar e executar app Flask/Python.
- Criar banco de dados do Azure para MySQL.
- Instalar Azure CLI.
- Criar ACR.
- Criar imagem Docker local.
- Enviar imagem Docker para ACR.
- Criar instância de container ACI via console Azure.
- Criar instância de container ACI via Azure CLI.

❄❄

0. Limpar VM, excluir contêineres e instalar pacotes Debian GNU/Linux.

```
rm -rf *

docker rm $(docker ps -qa) --force > /dev/null 2>&1

sudo apt update

sudo apt install libmariadb-dev-compat \
libmariadb-dev mariadb-client -y
```

1. Clonar a aplicação Python/Flask do Github.

```
git clone \
https://github.com/roger437unix/azure_flask_mysql.git
```

2. Entrar no diretório criado pela extração.

```
cd azure_flask_mysql
```

3. Criar um ambiente virtual Python e instalar os bibliotecas.

```
python3 -m venv .venv

source .venv/bin/activate

pip install -r requirements.txt
```

4. Executar a aplicação localmente.

```
flask run --host 0.0.0.0
```

4.1 Teste no navegador do host hospedeiro:

http://192.168.56.101:5000

http://192.168.56.101:5000/consultas

▶ Observe erros ao tentar inserir um novo registro e nas consultas devido ausência de banco de dados.

❄❄

5. Crie um banco na Azure para usar com a aplicação.

[Servidores do Banco de Dados do Azure para MySQL]

- User: tux
- Password: Mud@r123

▶ Na guia Rede

▶ Permitir o acesso público de qualquer serviço do Azure de dentro do Azure para esse servidor ✓

▶ Regras de Firewall

▶ + Adicionar o endereço IP do cliente atual (200.153.69.210) ✓

5.1 Após a implantação do Serviço MySQL na Azure.

▶ Menu Parâmetros do servidor

▶ require_secure_transport OFF

▶ Salvar

✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶✶

5.2. Conexão ao servidor MySQL utilizando MySQL Workbenck para criar um novo banco e tabela.

```
CREATE DATABASE banco;

USE banco;

CREATE TABLE tbl_dados(
id int auto_increment primary key,
nome varchar(50) not null,
email varchar(50) not null,
senha varchar(50) not null);
```

❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄

6. Configure a aplicação Python/Flask com o endpoint [Nome do servidor] do banco criado.

```
nano app.py
```

Obs.: Necessário configurar o arquivo da aplicação em dois locais:

- Consultar ==> Linha 28
- Inserir ==> Linha 66

6.1 Executar a aplicação novamente.

```
flask run --host 0.0.0.0
```

6.2 Teste no navegador do host hospedeiro:

http://192.168.56.101:5000

http://192.168.56.101:5000/consultas

****

7. AZURE CLI em VM Debian local.

7.1 Instalar Azure CLI.

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

7.2 Fazer Login Azure.

```
az login --use-device-code
```

7.3 Criar Grupo de recursos.

```
az group create --name rg-container --location eastus
```

7.4 Listar todos Grupos de Recursos.

```
az group list | grep -w "name"
```

```
    "name": "rg-banco",
    "name": "rg-container",
```

7.5 Criar variável para nome do serviço ACR.

```
SEU_PRIMEIRO_NOME_EM_LETRA_MINUSCULA=""

DIA_HORA=$(date +%d%m%y%H%M%S)

MEU_ACR=$SEU_PRIMEIRO_NOME_EM_LETRA_MINUSCULA$DIA_HORA

echo $MEU_ACR
```

7.6 Criar Azure Container Registry.

```
az acr create --resource-group rg-container --name $MEU_ACR \
--sku Basic --admin-enabled true
```

❌ Obs.: Não é permitido criar ACR com nomes repetidos.

7.7 Fazer login no Registro de Container do Azure.

```
az acr login --name $MEU_ACR
```

▶ Saída: Login Succeeded

7.8 Descobrir o nome "AcrLoginServer" do Azure Container Registry.

```
LOGINSERVER=$(az acr list --resource-group rg-container \
--query "[].{acrLoginServer:loginServer}" \
--output table | grep .io)


echo $LOGINSERVER
```

❄❄

7.9 Criar arquivo Dockerfile para nova imagem Docker.

```
nano Dockerfile
```

```
FROM python:3.10-rc-slim-bullseye
WORKDIR /app
COPY . .
RUN apt update
RUN apt install gcc libmariadb-dev-compat libmariadb-dev -y
RUN pip3 install -r requirements.txt
CMD ["flask", "run", "--host=0.0.0.0", "--port=8080"]
```

7.10 Construindo a nova imagem Docker.

```
docker build -t $LOGINSERVER/acr-flask-mysql-8080 .
```

7.11 Verificando todas as imagens.

```
docker images
```

```
REPOSITORY                                          TAG
IMAGE ID       CREATED         SIZE
tux170523150811.azurecr.io/acr-flask-mysql-8080   latest
5cebbc85f38a   3 minutes ago   549MB
```

7.12 Enviar imagem para ACR [Azure Container Registry].

```
docker push $LOGINSERVER/acr-flask-mysql-8080
```

❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄❄

8. Criar container AZURE ACI usando imagem armazenada em ACR via Console Azure.

[Azure - Instâncias de contêiner]

- Origem da imagem: Registro de Container do Azure

- Registro: SEU_NOME + DATA_HORA

- Imagem: acr-flask-mysql-8080

Avançar: Rede >

- Portas: 8080

- Protocolos de portas: TCP

****

9. Criar container AZURE ACI usando imagem armazenada em ACR via CLI Azure.

9.1 Idenficar senha do Registry [ACR] via CLI.

```
ACR_PASSWD=$(az acr credential show --name $MEU_ACR \
--resource-group rg-container | grep -B 3 password2 \
| grep -w value | cut -d'"' -f4)

echo $ACR_PASSWD
```

9.2 Criar contêiner ACI via CLI.

```
az container create --resource-group rg-container \
--name flask-mysql-cli \
--image $LOGINSERVER/acr-flask-mysql-8080 \
--dns-name-label $MEU_ACR$(date +%h) --ports 8080 \
--location eastus --registry-username $MEU_ACR \
--registry-password $ACR_PASSWD
```

****