Nelson Souza Follow
Apr 30, 2021 · 6 min read

RavenDB — Criando Cluster com Docker

Como podemos criar um cluster do RavenDB com Docker?

Olá tudo bem? Mais um post rápido sobre RavenDB. E hoje iremos criar um cluster usando o Docker. Não irei abordar aqui o que é Docker e tudo mais, então se quiser ver alguma coisa, já fiz um post antigo aqui: Docker — VirtualBox + Linux.

Neste post estarei usando o WSL 2 com Docker instalado em minha máquina Windows.

. . .

Como funciona um cluster no RavenDB?

A arquitetura de clusters de banco de dados é formada pela redundância do seu banco de dados em 2 ou mais instâncias. Estas instâncias são balanceadas de forma a dividir com eficiência a quantidade de requisições ao banco, fazendo com que tenha uma maior disponibilidade dos dados e tolerância a falhas. Essas instâncias são chamadas de nós *(ou nodes em inglês)*. Cada nó tem um estado e tipo específicos.

> *Neste post também não estarei abordando tudo sobre um comportamento e conhecimento geral de cluster no RavenDB. Então, sugiro conhecer um pouco mais no link aqui:* ***Cluster Topology****.*

No cluster, o banco de dados será replicado para vários nós. Um grupo de nós no cluster que contém o mesmo banco de dados é chamado de ***Grupo de Banco de Dados****.*

- Este número de nós no grupo de banco de dados é definido pelo fator de replicação ***(Replication Factor)*** no momento em que se cria o banco de dados.
- Os documentos são mantidos em sincronia nos nós do ***Grupo de Banco de Dados*** com uma replicação ***master-master (ou também conhecido como multi-master)****.*

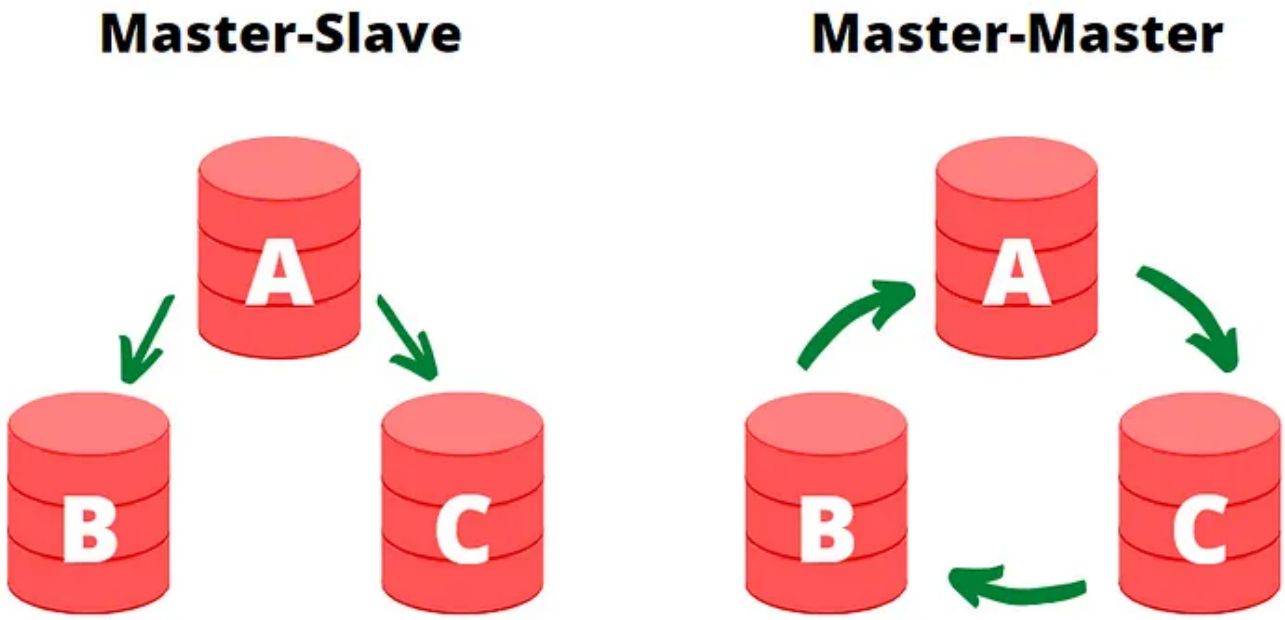
Normalmente um cluster irá trabalhar com *master-slave* e/ou *master-master*. E o que seriam esses 2 caras?

Master-Slave: garante que toda a escrita será feita em uma instância do nó *master*, e toda consulta será obtida nos *slaves* evitando assim concorrência na escrita com o nó master. No entanto, se esse *master* por acaso tiver um problema ou alguma falha, você não conseguirá ter acesso aos dados mais atualizados e provavelmente terá acesso somente aos *slaves* com os antigos.

Master-Master: todos as escritas e leituras são feitos em todos os nós, pois existe a replicação ***sincronizada*** entre todos os nós ao mesmo tempo. Se por exemplo um dos nós cair e quando este voltar, será atualizado automaticamente com os dados gravados nos outros 2 nós que estavam online. Essa ***sincronização*** se baseia em ***consistência eventual****.* Em algum momento os dados estarão atualizados, ou seja, não espere que seja imediato, mas eles estarão lá!

> *O RavenDB trabalha com Master-Master e a recomendação, ou boa prática assim dizendo, é ter pelo menos 3 nós em um cluster, para garantir uma melhor alta disponiblidade.*

Abaixo temos um desenho mais ou menos dos modelos falados acima:



. . .

Escrevendo o docker-compose.yaml

Bem, primeiro de tudo, devemos ter uma licença do RavenDB, seja ela Community ou Development. Você pode obter ela aqui: ***https://ravendb.net/buy*** bastando preencher as informações e em seguida irá receber email com um JSON. Este conteúdo é a nossa licença.

Vamos subir uma única instância para sabermos do que se tratam algumas das configurações. Não estarei abordando todas. Para saber mais um pouco olhe aqui: **Installation: Running in a Docker Container**.

Abaixo temos o *docker-compose.yaml*:

```
version: '3.7'
services:
  raven1:
    image: ravendb/ravendb:5.1.7-ubuntu.20.04-x64
    container_name: raven1
    hostname: raven1
    ports:
      - 8080:8080
      - 38888:38888
    environment:
      - RAVEN_Security_UnsecuredAccessAllowed=PublicNetwork
      - RAVEN_Setup_Mode=None
      - RAVEN_License_Eula_Accepted=true
```

```
    volumes:
      - raven-data:/opt/RavenDB/Server/RavenData

volumes:
  raven-data:
```

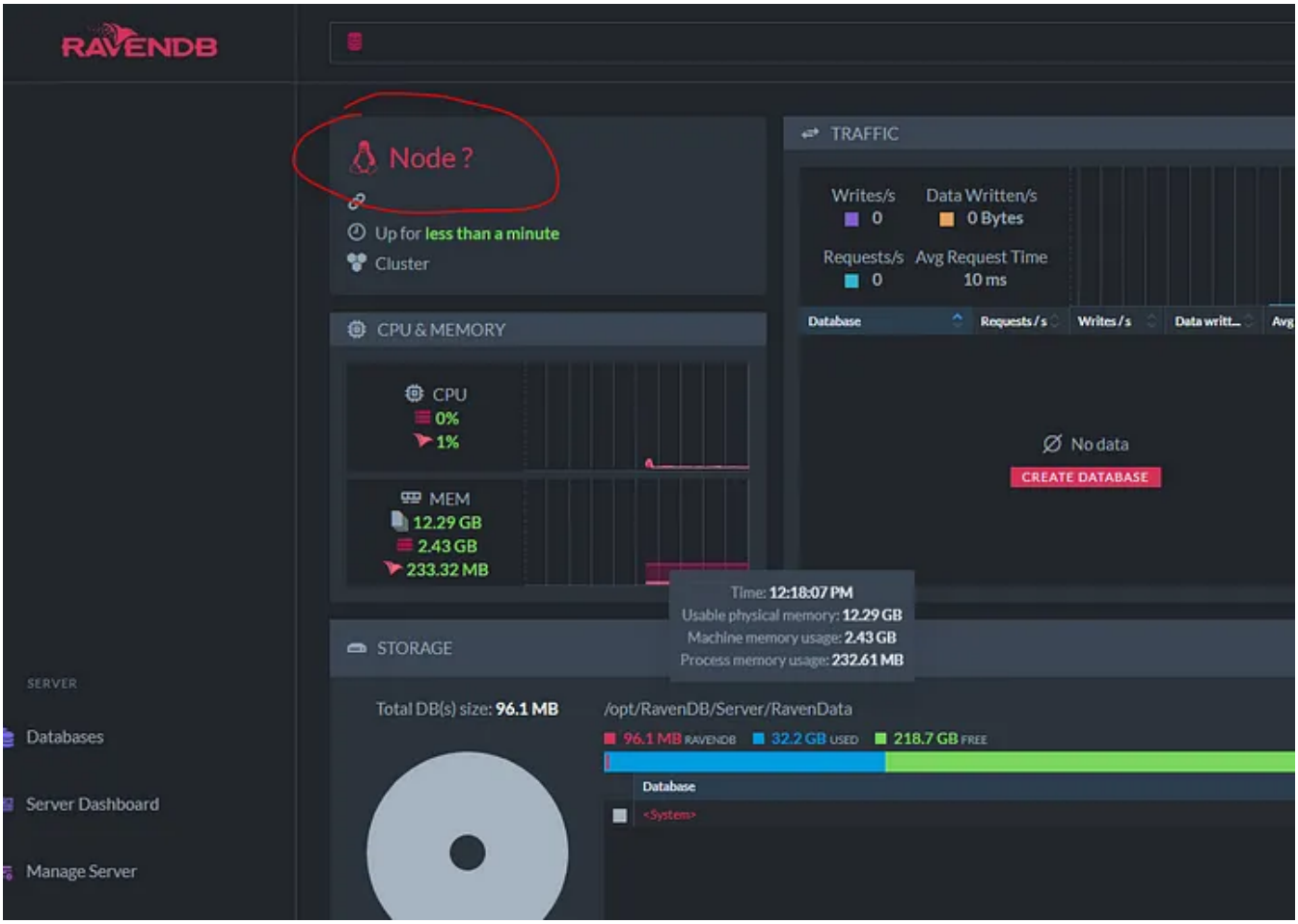
> *Neste código acima, estou usando a imagem mais atual do RavenDB, mas pode ser que te interessar, a partir da versão 4.*

- **Container name** é o nome do container em si, é opcional, é somente para uma melhor visualização quando listamos os containers ativos no Docker.
- **Hostname** estou alterando o container id para um nome mais amigável.
- Por padrão do RavenDB utiliza as portas *8080* e *38888*, neste caso deixei mapeada para poder acessar de fora do container as mesmas portas. Quando formos usar outras instâncias, podemos mudar as portas. Mais a frente veremos isso.
- **Volumes** é onde será gravado os bancos de dados criados no RavenDB.
- Em environment coloquei algumas coisas aí:
  - ***RAVEN_Security_UnsecuredAccessAllowed*** como não estou usando nenhum certificado digital e nenhuma configuração em especial, é somente para desenvolvimento, então nosso cluster será “inseguro”.
  - ***RAVEN_Setup_Mode*** para que não mostre a tela de
  - ***RAVEN_License_Eula_Accepted*** aqui o nome já diz, são aqueles termos que praticamente ninguém lê! 😂

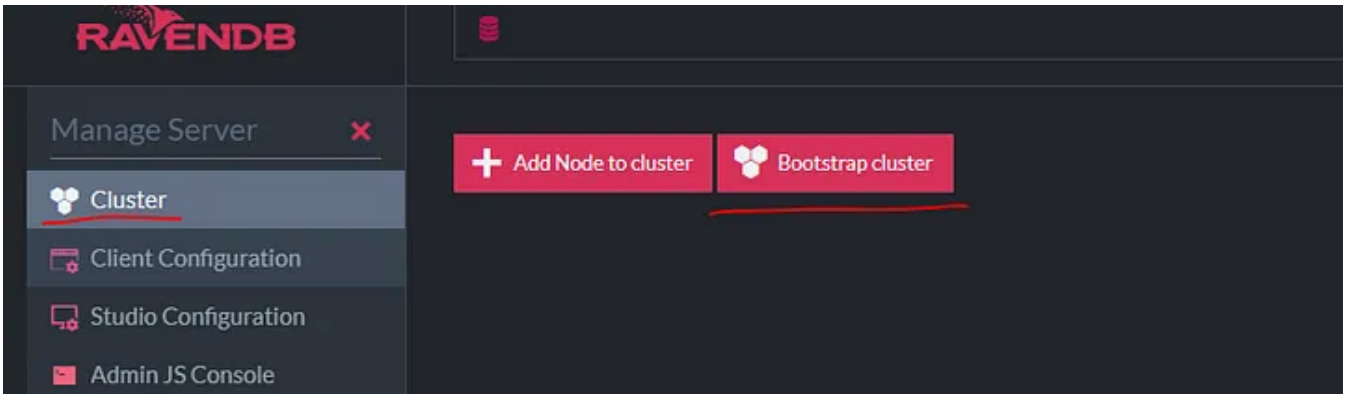
Executando o comando:

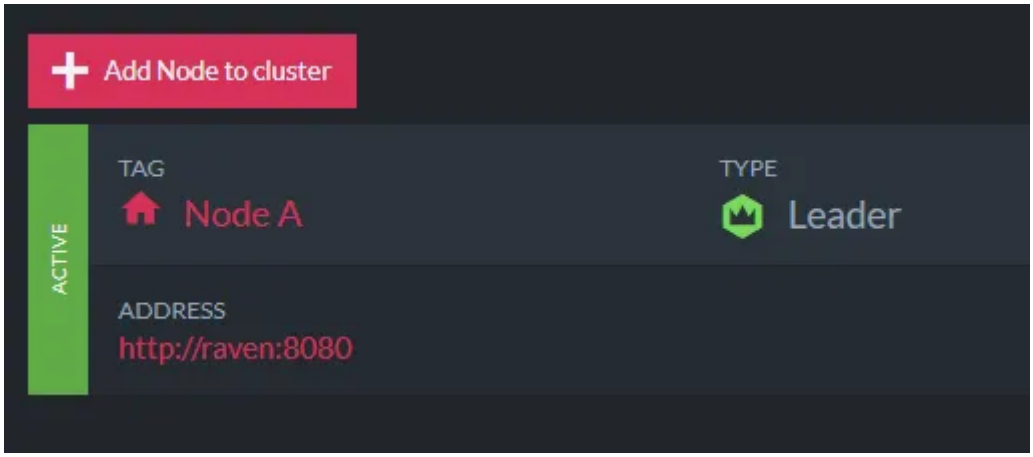
```
docker-compose up -d
```

Temos o resultado abrindo ***http://localhost:8080*** e observe que não foi atribuído um nó. Geralmente são nomeados por letras A,B,C.

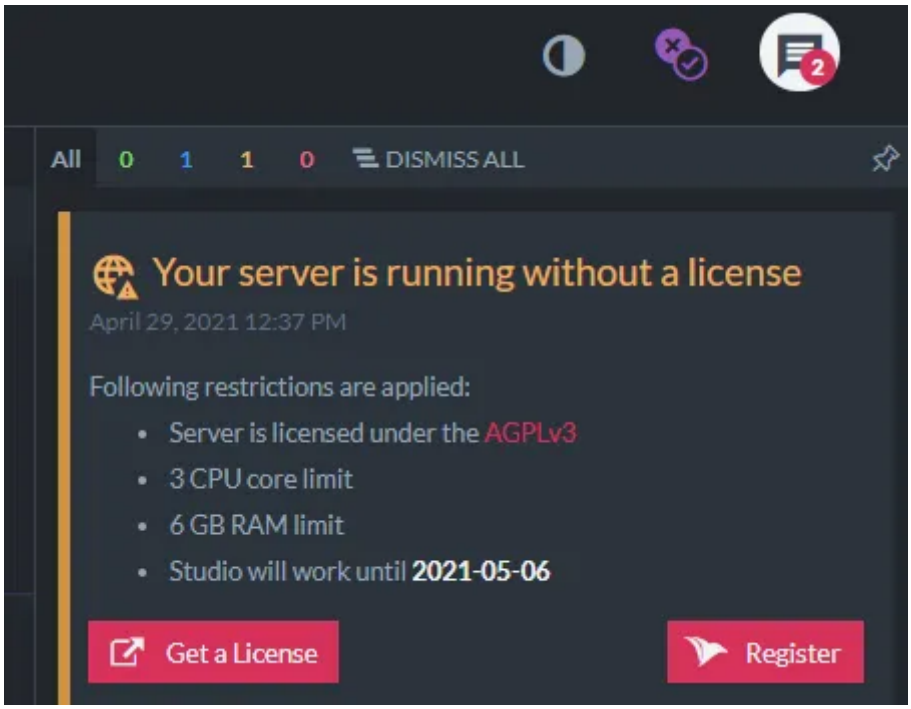


Você pode atribuir o primeiro nó indo ao menu *Cluster / Bootstrap* cluster que automaticamente o RavenDB configura.





Além disso, lembra da licença? Hora de aplicar bastando colar o JSON recebido:



Uma outra forma de setar essas configurações de forma manual utilizando POST usando curl:

```
curl 'http://localhost:8080/admin/license/activate' -H 'Content-
Type: application/json; charset=UTF-8' --data-binary '{
        "Id": "c4559449-ed9e-4fda-969d-d9fe67f921e8",
        "Name": "NELSON SOUZA",
        "Keys": [
            "xxxxx"
        ]
 }'
```

E aqui para registrar o nó A:

```
curl 'http://localhost:8080/admin/license/set-limit?
nodeTag=A&newAssignedCores=1' -X POST -H 'Content-Type:
application/json; charset=utf-8'
```

. . .

Criando um cluster com 3 instâncias

Bom, até aí tudo bem. Vamos então alterar nosso docker-compose.yaml para subir um cluster com 3 nós e vamos executar mais uns comandos novos com o curl para adicionar os nós B e C ao cluster.

> *Deixei destacado em negrito os nomes dos hostnames e portas que agora estamos utilizando não somente a 8080*

```
version: '3.7'
services:
  raven1:
    image: ravendb/ravendb:5.1.7-ubuntu.20.04-x64
    container_name: raven1
    hostname: raven1
    ports:
      - 8080:8080
      - 38888:38888
    environment:
      - RAVEN_Security_UnsecuredAccessAllowed=PublicNetwork
      - RAVEN_Setup_Mode=None
      - RAVEN_License_Eula_Accepted=true
    volumes:
      - raven-data:/opt/RavenDB/Server/RavenData

  raven2:
    image: ravendb/ravendb:5.1.7-ubuntu.20.04-x64
    container_name: raven2
    hostname: raven2
    ports:
      - 8081:8080
      - 38889:38888
    environment:
      - RAVEN_Security_UnsecuredAccessAllowed=PublicNetwork
      - RAVEN_Setup_Mode=None
      - RAVEN_License_Eula_Accepted=true

  raven3:
    image: ravendb/ravendb:5.1.7-ubuntu.20.04-x64
    container_name: raven3
    hostname: raven3
    ports:
```

```
      - 8082:8080
      - 38890:38888
    environment:
      - RAVEN_Security_UnsecuredAccessAllowed=PublicNetwork
      - RAVEN_Setup_Mode=None
      - RAVEN_License_Eula_Accepted=true

volumes:
  raven-data:
```

E os comandos curl:

- Para registrar a licença o RavenDB *(não esqueça de inserir sua licença como parâmetro de data-binary)*

```
curl 'http://localhost:8080/admin/license/activate' -H 'Content-
Type: application/json; charset=UTF-8' --data-binary '{
        "Id": "c4559449-ed9e-4fda-969d-d9fe67f921e8",
        "Name": "NELSON SOUZA",
        "Keys": [
            "xxxxx"
        ]
 }' --compressed
```

- Para adicionar o nó principal e os outros nós ao cluster:

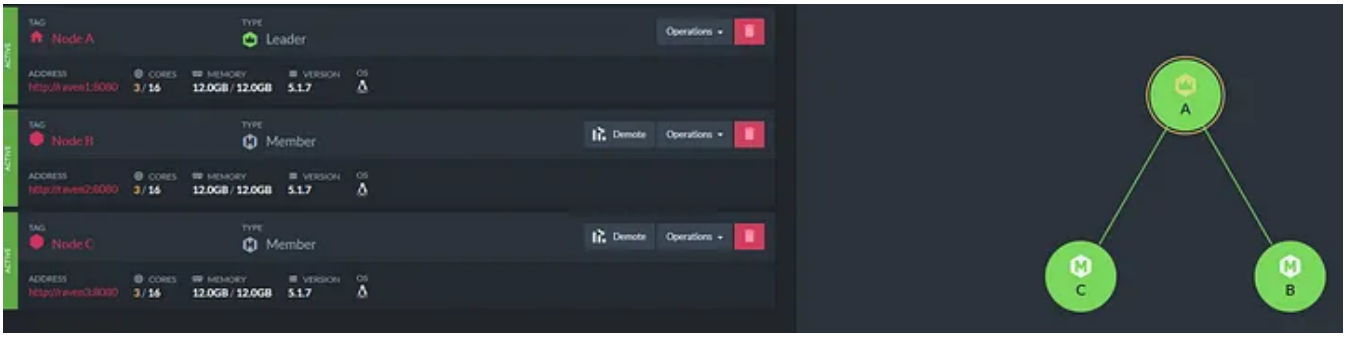
```
curl 'http://localhost:8080/admin/license/set-limit?
nodeTag=A&newAssignedCores=3' -X POST -H 'Content-Type:
application/json; charset=utf-8' -H 'Content-Length: 0' --compressed

curl 'http://localhost:8080/admin/cluster/node?
url=http%3A%2F%2Fraven2%3A8080&assignedCores=3' -X PUT -H 'Content-
Type: application/json; charset=utf-8' -H 'Content-Length: 0' --
compressed

curl 'http://localhost:8080/admin/cluster/node?
url=http%3A%2F%2Fraven3%3A8080&assignedCores=3' -X PUT -H 'Content-
Type: application/json; charset=utf-8' -H 'Content-Length: 0' --
compressed
```

Uma boa dica é criar um arquivo bash contendo essas coisas todas mais o comando *docker-compose up -d*, assim executa tudo de uma única vez! 👍

E o resultado será este:

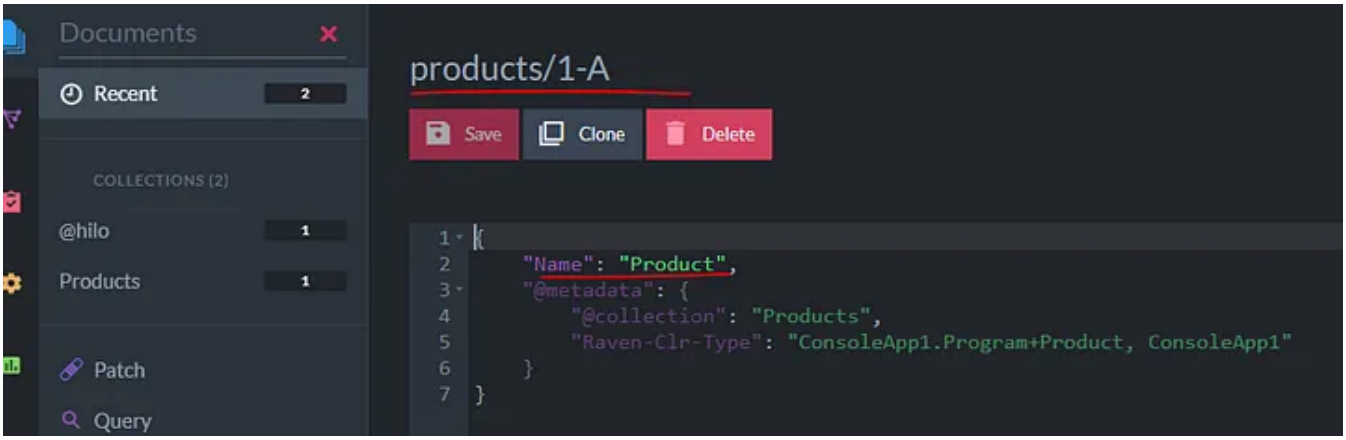


Testando a conexão com no cluster usando C#:

```
public class Product
{
  public string Id { get; set; }
  public string Name { get; set; }
}

static async Task Main(string[] args)
{
 var store = new DocumentStore
 {
   Urls = new[]
   {
    "http://localhost:8080",
    "http://localhost:8081",
    "http://localhost:8082"
   },
   Database = "Demo"
 }.Initialize();

 using (var session = store.OpenAsyncSession())
 {
   var prod = new Product {Name = "Product"};
   await session.StoreAsync(prod);
   await session.SaveChangesAsync();
 }
}
```



. . .

Conclusão

É super simples subir um cluster de RavenDB com Docker. Sinceramente, é bem mais fácil que os concorrentes que demandam mais uma série outros passos. 😂

Até mais!

Referências

NoSQL Database Documentation

We hereby welcome you into the RavenDB Documentation. Feel free to explore the sizeable world of our premier NoSQL...

ravendb.net

Cluster: Overview

RavenDB's clustering provides redundancy and an increased availability of data that is consistent across a...

ravendb.net

Ravendb Clustering Docker Docker Compose Net Core