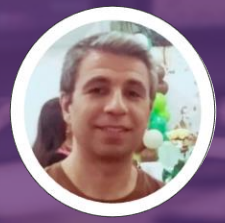
# MeetUP

ItuDevelopers®

## 10/12 • 9h

Auditório da FATEC

**SEJA BEM-VINDO**



**Entity Framework: detalhes para começar certo**

Gustavo Bellini Bigardi

**Usando Docker no desenvolvimento .NET**

Marcio Nizzola

**Liderança: comece e por você**

Thomaz Halter

MeetUP

ItuDevelopers®


CI&T

# Usando Docker no desenvolvimento .NET

Marcio Nizzola - Software Architect Sr - MVP


**Software Architect SR – CI&T**

**Professor de Tecnologia na Etec Itu**

**Membro fundador do Meetup Itu Developers**

**Formação:**

**Proc. De Dados - 1989**

**Técnico em Proc. de Dados – 91-92**

**Análise de Sistemas – 94-98**

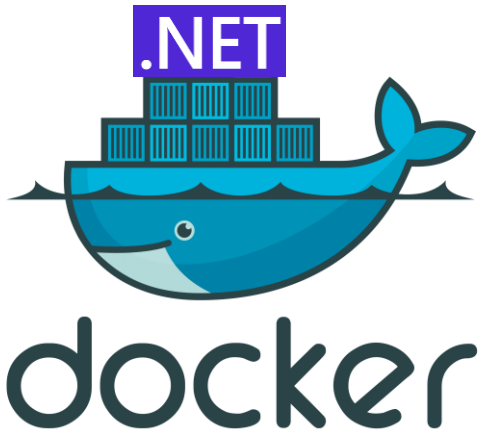
**MBA em Gestão de Projetos – 2013**

**Licenciatura para docência- 2016**

**Principais Tecnologias já utilizadas:**
DBase, Basic, Clipper, C,
Pascal, Cobol, Visual Basic, Delphi, Asp, C#, PHP,
Java, Javascript, ASP.NET, ASP.NET MVC, Visual Basic .NET., Angular, Ionic, React

MeetUP ItuDevelopers®

# Usando Docker no desenvolvimento .NET




**Por que falar sobre esta parceria (.NET x Docker) ?**

- **Não se entrega mais software como antigamente !**
- **Entregar software com qualidade é essencial**
- **Testar software em ambientes diferentes não garante que funciona**
- **Executar, provisionar e disponibilizar um novo hardware costumava levar dias, um processo muito cansativo. Com os containers Docker, a implantação leva alguns segundos.**
- **Muitas vezes os responsáveis pela infraestrutura não liberam os recursos em tempo hábil, com contêineres você pode testar a tecnologia e construir o código, enquanto espera.**
- **Com o uso de aplicações em Conteineres podemos ganhar escalabilidade, aumentando o número de instâncias da aplicação sem necessidade de provisionar máquinas físicas ou virtuais para isto.**

# O que é Docker ?

Docker é uma plataforma aberta, criada com o objetivo de facilitar o desenvolvimento, a implantação e a execução de aplicações em ambientes isolados.

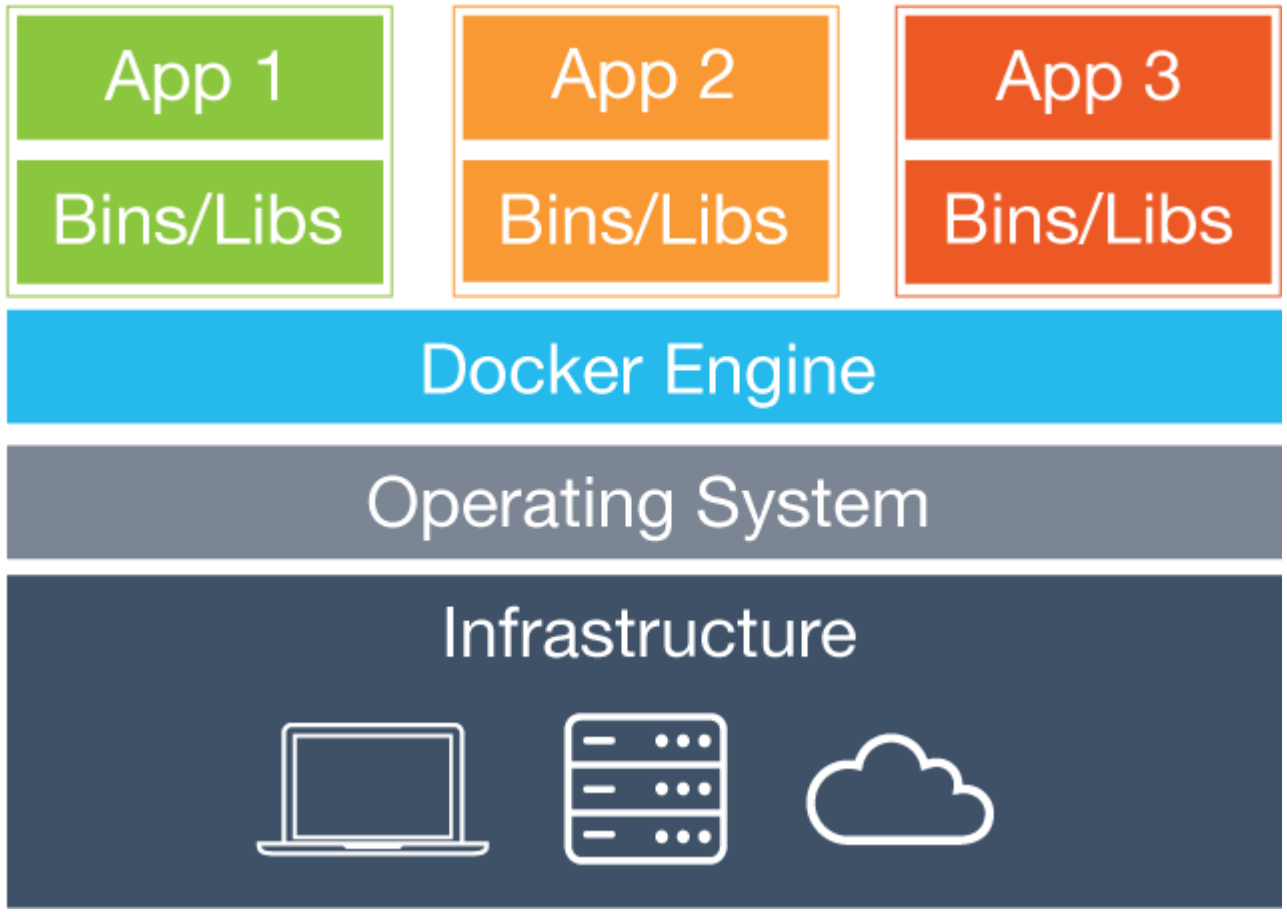
A principal diferença entre o Docker e uma máquina virtual, é que a VM tem o seu próprio sistema operacional, já o Docker compartilha o sistema operacional do host.
Com isto, um contêiner do Docker possui um tamanho muito menor que uma máquina virtual, além do que o seu consumo de recursos é muito menor.

Um container Docker é um pacote de software com todas as dependências necessárias para executar um aplicativo específico.

Sempre que um usuário executa uma imagem, um novo container é criado.

O principal atrativo do Docker é sua portabilidade. Ele permite que os usuários criem ou instalem um aplicativo complexo em uma máquina e tenham certeza de que funcionará nele !

# Como funciona o seu isolamento




O modelo de isolamento utilizado no Docker é a virtualização a nível do sistema operacional, um método de virtualização onde o kernel do sistema operacional permite que múltiplos processos sejam executados isoladamente no mesmo host.

Esses processos **isolados** em execução são denominados no Docker de container !

## Mas eu só posso utilizar para **entregar** o meu software ? NÃO !!

Esta é uma coisa muito importante que o Docker permite, podemos utilizá-lo para instalar ferramentas que podemos utilizar para integrar nossos sistemas, sem precisar instalar tudo isso no nosso PC deixando-o lento e cheio de dependências que podem até deixa-lo lento ou ter conflitos de versões.

### Contêineres de gerenciadores de Bancos de Dados:

Microsoft Sql Server
Mongo Db
MySql
Cassandra
Redis Cache

### Contêineres para utilização de filas:

RabbitMq

### Contêineres para Linguagens de programação

.NET, PHP, Go, Node, Rust, Python, até....Java !

Por que eu instalei o XAMPP, SQL Server, e mais um monte de coisas no meu PC !

# Antes de mais nada vamos entender:

**O que são imagens do Docker ?**

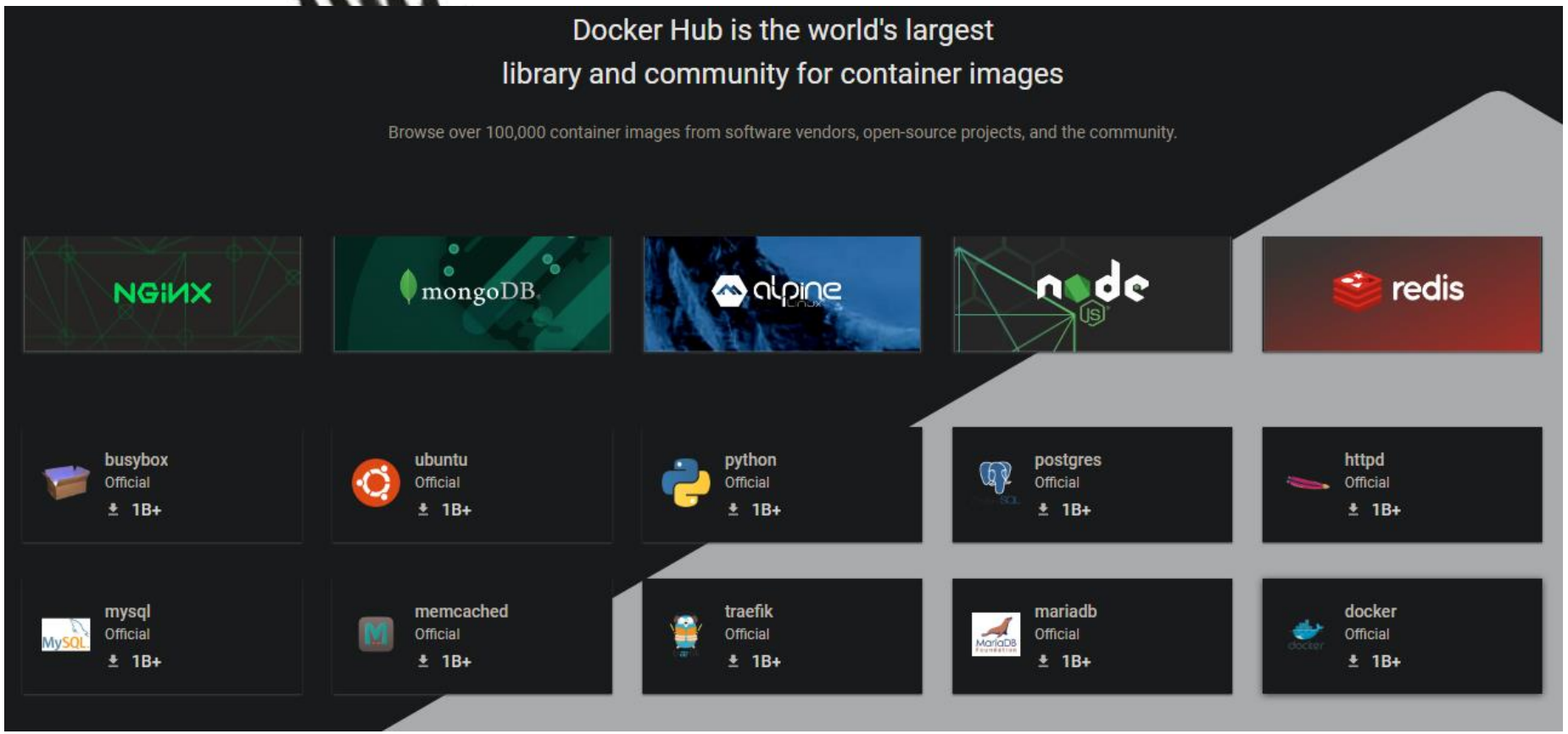
**As imagens do Docker, são compostas por sistemas de arquivos de camadas que ficam uma sobre as outras. Ela é a nossa base para construção de uma aplicação, contendo as camadas necessárias para a execução da aplicação.**

**E containers?**

**São instâncias daquela imagem que serão geradas para execução, podendo ser replicadas e instanciadas quantas vezes precisarmos.**

# Mas de onde vem as imagens?

**As imagens são disponibilizadas pelos seus criadores e em sua maioria inseridas no site DockerHub, que é onde existe um grande número de imagens base armazenadas e que também pode ser utilizado para que você suba as suas e disponibilize para uso.**



## Então vamos ao que interessa !

**Vamos criar uma aplicação onde utilizaremos contêineres rodando no Docker**

**Teremos uma WebApi que irá receber uma requisição e inserí-la numa fila do RabbitMq.**

**Teremos um Consumer que irá ouvir a fila no RabbitMq e assim que receber o pedido irá inserí-lo no banco de dados Sql Server.**

**Ok, mas qual o segredo disso?**

**WebApi + Consumer + Sql Server + RabbitMq todos rodando no Docker !**

RabbitMQ

Microsoft® SQL Server®

MeetUP ItuDevelopers®

## Para começar, vamos instalar:

1) RabbitMq


Para instalar o RabbbitMq, iremos executar o comando abaixo, quer irá baixar e instanciar um contêiner para utilizarmos:

```
docker run -d — hostname my-rabbit — name rabbit13 -p 8080:15672 -p 5672:5672 -p 25676:25676 rabbitmq:3-management
```

1) Microsoft Sql Server


```
docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=123mudar" -p 1433:1433 --name sql1 -d mcr.microsoft.com/mssql/server:2019-GA-ubuntu-16.04
```

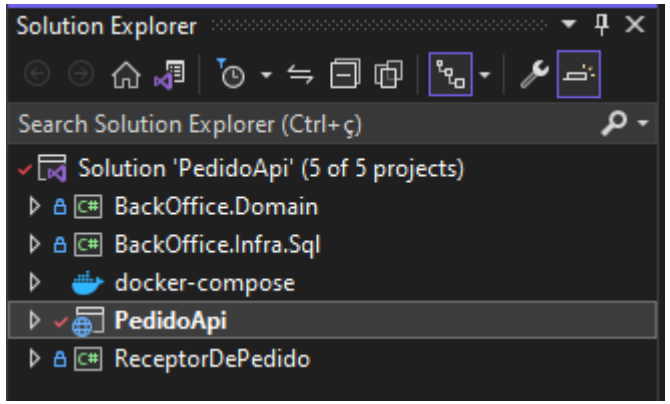
## Então vamos ao que interessa !

**Vamos então ao .NET ver como ficou estruturada a aplicação**

**Foi criada uma API, para recepção de pedidos, que insere diretamente no RabbitMq**

**Foi criado um Consumer, que lê o Rabbit Mq, e insere no banco de dados os pedidos**

**Obs: estes serão partes de um projeto maior, que terá consumers de pagamento, envio de e-mails quando alterados status, etc. (acompanhem minhas postagens futuras)**

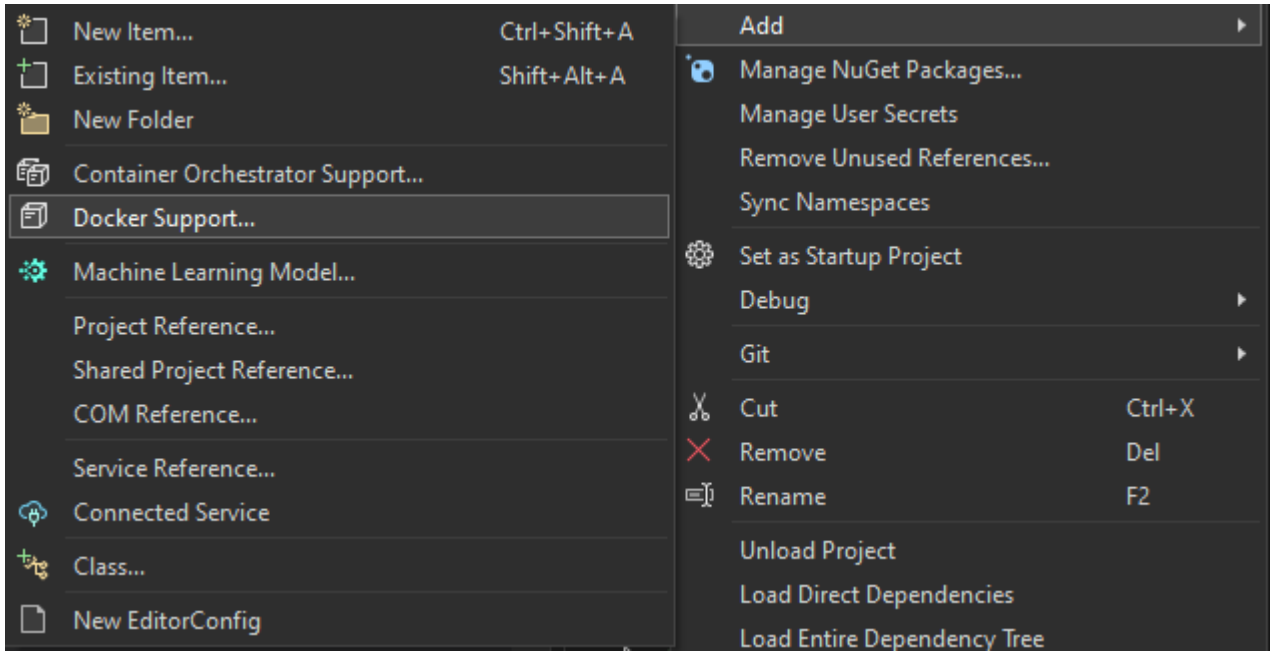



# O que é o DockerFile ?

**Para podermos gerar a nossa imagem, com a aplicação .NET iremos utilizar um imagem disponibilizada pela Microsoft para servir de base, e nela será incorporada a publicação da nossa aplicação.**

**O Dockerfile, é um arquivo com as intruções para que o Docker possa realizar o Build da nossa aplicação, inserindo os binários dentro da imagem para que possamos criar as suas instâncias.**

**Podemos construí-lo manualmente, ou utilizar o Visual Studio para cria-lo para nós.**



# O que é o DockerFile ?

## Este é o DockerFile gerado automaticamente pelo Visual Studio

Dockerfile | BaseModel.cs | appsettings.json | PedidoModel.cs | PedidoItemModel.cs | PedidosConsumer.cs | IConsumerService.cs

```
1   #See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to
2
3   FROM mcr.microsoft.com/dotnet/runtime:7.0 AS base
4   WORKDIR /app
5
6   FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
7   WORKDIR /src
8   COPY ["ReceptorDePedido/ReceptorDePedido.csproj", "ReceptorDePedido/"]
9   COPY ["BackOffice.Infra.Sql/BackOffice.Infra.Sql.csproj", "BackOffice.Infra.Sql/"]
10  COPY ["BackOffice.Domain/BackOffice.Domain.csproj", "BackOffice.Domain/"]
11  RUN dotnet restore "ReceptorDePedido/ReceptorDePedido.csproj"
12  COPY . .
13  WORKDIR "/src/ReceptorDePedido"
14  RUN dotnet build "ReceptorDePedido.csproj" -c Release -o /app/build
15
16  FROM build AS publish
17  RUN dotnet publish "ReceptorDePedido.csproj" -c Release -o /app/publish /p:UseAppHost=false
18
19  FROM base AS final
20  WORKDIR /app
21  COPY --from=publish /app/publish .
22  ENTRYPOINT ["dotnet", "ReceptorDePedido.dll"]
```

## E depois, o que faço com o DockerFile ?

Podemos criar a imagem via linha de comando e depois utilizar a imagem para criar contêineres.

Para criar a imagem, utilizamos o comando abaixo, dentro da pasta do projeto (csproj)

```
docker build -t meetup8api -f Dockerfile ..
```

E para criar um contêiner a partir dela:

```
docker run –d -p 5001:80 -e ASPNETCORE_ENVIRONMENT=Development meetup8api --name meetup8api
```

Veja que especificamos –p 5001:80, o que é isso? É o redirecionamento da porta onde é executada a aplicação, para que ela seja disponibilizada para nosso uso na porta 5001.

Ou senão, o Visual Studio cuida de gerenciar a sua execução conforme testamos a aplicação, além de também fazer o build de uma nova imagem e iniciar e fechar a sua execução no Docker !

(por isso que programador .NET começa a gostar da plataforma do Visual Studio....)

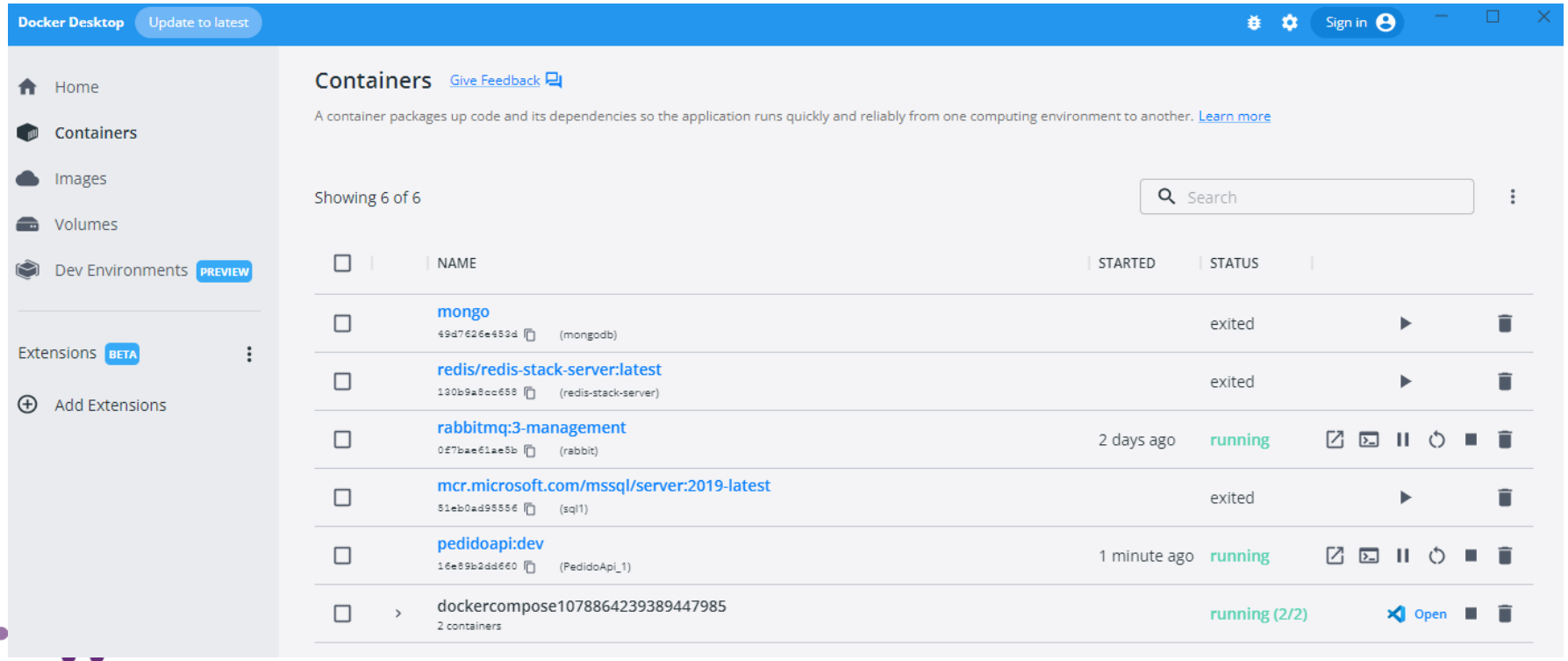
# Como ficaram os contêineres?

**Podemos ver de duas formas, via linha de comando, ou com o Docker Desktop**

**Digitando: docker ps –a**

**Podemos listar todos os contêineres da nossa instalação do docker.**

**Ou visualizar na tela :**



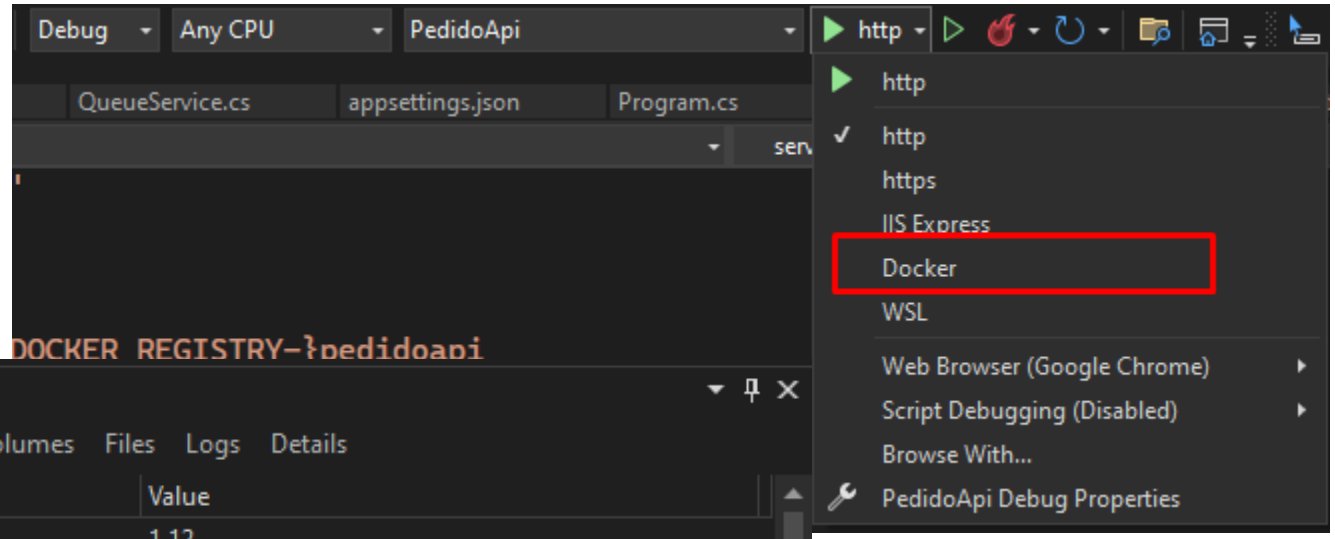

MeetUP ItuDevelopers

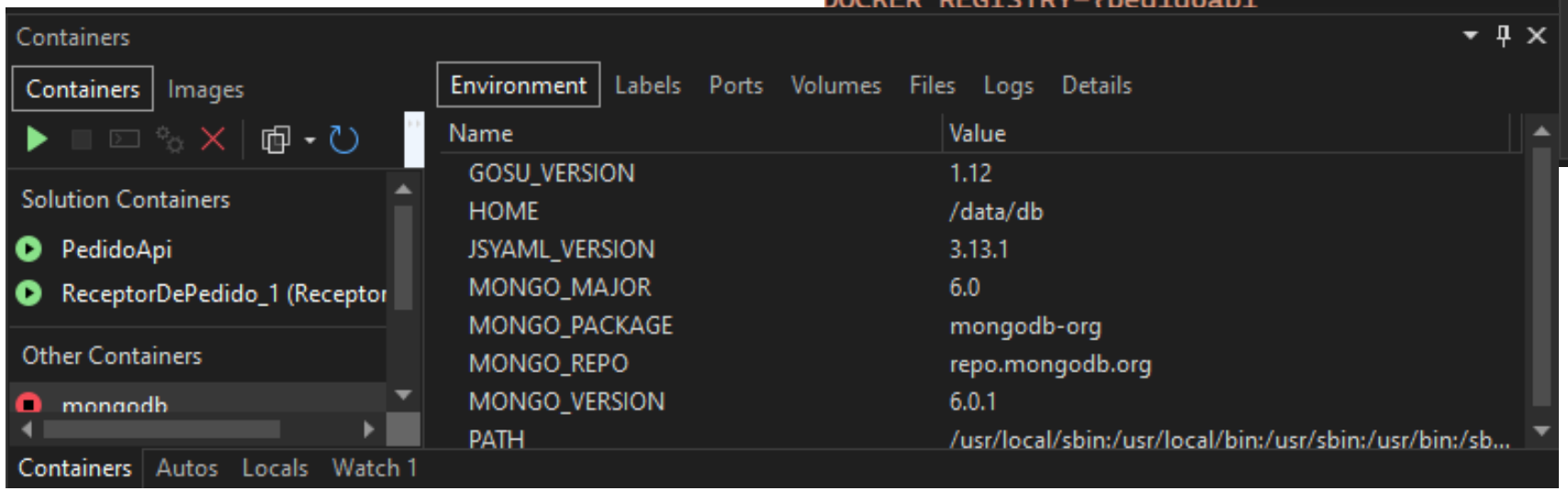
## Como isso funciona no Visual Studio ?

**Depois de criado o projeto com um Dockerfile em sua pasta é possível executar diretamente no Visual Studio, inclusive fazer Debug da aplicação com breakpoints e tudo mais.**

**Setando no toolbok para que a aplicação Inicialize utilizando Docker !**

**Abaixo poderá ver que uma tela irá aparecer Durante a execução**




MeetUP ItuDevelopers

# Docker Compose ?

É um serviço do próprio Docker que permite criarmos diversos contêineres simultaneamente, o que elimina o trabalho manual de criarmos um a um, facilitando o deploy da aplicação em múltiplos ambientes.

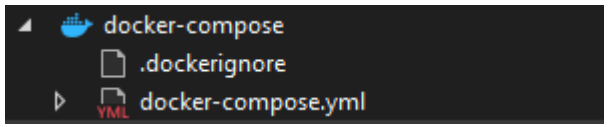
O Docker Compose é parte da aplicação instalado durante a instalação do Docker Desktop

As versões mais atuais do Visual Studio dão suporte à orquestração de Contêineres com Docker Compose

```yaml
version: '3.4'

services:
  pedidoapi:
    image: ${DOCKER_REGISTRY-}pedidoapi
    build:
      context: .
      dockerfile: PedidoApi/Dockerfile

  receptordepedido:
    image: ${DOCKER_REGISTRY-}receptordepedido
    build:
      context: .
      dockerfile: ReceptorDePedido/Dockerfile
```

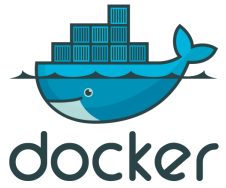



Então vamos ao que interessa !

# FINALMENTE !!!
# DEMO !!

https://youtu.be/XdCyoHAJJbo

# Comandos comuns




**Utilizar Docker consiste em sabermos utilizar seus comandos para executarmos as tarefas mais comuns.**

**Dentre elas:**

| Comandos mais comuns | |
|---|---|
| docker start <container> | Inicializa um contêiner |
| docker stop <container> | Termina a execução de um conteiner |
| docker start <container> | Inicializa um contêiner |
| docker stop <container> | Termina a execução de um conteiner |
| docker logs -f <CONTAINER> | Exibe os logs de um contêiner em execução |
| docker rm CONTAINER_NAME | Remove um contêiner |
| docker rm -f CONTAINER_NAME | Remove um contêiner |
| docker cp CONTAINER_NAME:SOURCE TARGET | Copia arquivo do contêiner para o host |
| docker cp TARGET CONTAINER_NAME:HOST | Copia arquivo do host para o conteiner |
| docker rename myweb web | Renomeia um conteiner |
| docker images | Lista as imagens presentes na instalação |

**Existem muito mais comandos, na seção de referência teremos links para páginas com vários deles.**

## Podemos utilizar contêineres na Nuvem ?

Podemos dizer que é onde eles são mais utilizados, pois justamente o conceito é voltado à distribuição do software em ambientes distintos onde será possível prover através de contêineres as configurações idênticas ao ambiente onde ele foi testado previamente.

Temos serviços de contêineres nos principais provedores de Cloud, como: Azure, Aws, Google Cloud.

Também podemos instalar o Docker em máquinas nos datacenters ou até em máquinas virtuais e ter o mesmo resultado.

Desta forma garantiremos a execução das aplicações com compatibilidade total com o local projetado para que elas funcionem !

## Mas tá tão bonito, não tem nenhuma coisa ruim pra falar desse tal de Docker?

•Velocidade– mesmo que executar um aplicativo por meio de um container do Docker seja mais rápido do que em uma máquina virtual, ainda é consideravelmente mais lento do que executar aplicativos nativamente em um servidor físico.

•Difícil de usar– O Docker não se destina a executar aplicativos que exijam uma interface gráfica do usuário (GUI). Isso significa que os usuários precisam estar familiarizados com a linha de comando e realizar todas as ações nela. A curva de aprendizado íngreme, as advertências específicas do sistema operacional e as atualizações frequentes tornam o domínio do Docker um desafio.

•Segurança– O Docker é executado no sistema operacional do host. Isso significa que qualquer software malicioso oculto em containers pode chegar à máquina host.

## Perguntas relevantes:

**Qual a diferença entre Docker e Kubernetes ?**

Docker é uma plataforma para construir e executar containers, o Kubernetes é um sistema de orquestração de containers de código aberto. O Docker é responsável pela criação de containers e o Kubernetes os gerencia em grande escala. O Docker possui seu próprio orquestrador que é o Docker Swarm.

**O Docker é pago?**

Na verdade a aplicação Docker e o Docker Desktop eram disponibilizados gratuitamente até uns anos atrás, porém desde então o Docker Desktop (interface gráfica) passou a ter sua utilização para fins empresariais cobrada, a versão por linha de comando ainda é free, inclusive conheço casos de empresas onde receberam notificação de que estão em uso e deveriam procurar licenciar !

## Mais perguntas?

## Referências:

Código fonte da Apresentação
github.com/NIZZOLA/MeetupItu8

Tutoriais Ms
learn.microsoft.com/pt-br/aspnet/core/host-and-deploy/docker/visual-studio-tools-for-docker?view=aspnetcore-7.0

Comandos Docker
tarry-fold-400.notion.site/Comandos-Docker-detalhados-b869d1ee33b843d19196b14a67c1a7ba

medium.com/xp-inc/rabbitmq-com-docker-conhecendo-o-admin-cc81f3f6ac3b

## Sigam-me nas redes sociais:

https://linktr.ee/nizzola

# Projeto Mentoria 2023

Já pensou passar por uma mentoria e aprender mais sobre desenvolvimento de Software ? Subir de nível e até ganhar mais $$ ?

Reuniões quinzenais com mentoria baseada em um roadmap de carreira de desenvolvedor .NET, e desenvolvimento de projetos reais !

Inscreva-se no formulário para participar do sorteio, que será realizado no início de janeiro e você verá o resultado publicado nas minhas redes.

## Preencha o form e concorra !

https://bit.ly/3PgVxaj

## Sigam-me nas redes sociais:

https://linktr.ee/nizzola