



Neste artigo vou apresentar os conceitos básicos relativos ao Docker sob o ponto de vista de um desenvolvedor .NET.



Hoje vamos criar uma [imagem](#) para uma aplicação ASP .NET Core MVC; a seguir vamos criar um [contêiner](#) e executar o contêiner e a aplicação. ([artigo anterior](#))



As tarefas que vamos cumprir para realizar esse objetivo serão:

1. Criar uma aplicação ASP .NET Core MVC no Linux usando a ferramenta de linha de comando e o VS Code;
2. Após criar a aplicação vamos fazer o deploy da aplicação;
3. A seguir vamos criar uma imagem para esta aplicação contendo a [ASP .NET Core e os runtimes do .NET Core](#) e as bibliotecas otimizadas para rodar a ASP .NET Core em produção.

Para acompanhar a primeira tarefa veja este artigo : [Criando uma aplicação Web no Linux](#)

Depois, retorne e continue com a criação da imagem neste artigo.

Criando uma imagem para uma aplicação ASP .NET Core MVC

Agora que já criamos a aplicação ASP .NET Core MVC vamos criar a imagem.

Para fazer isso primeiro temos que publicar a aplicação web que criamos na tarefa anterior usando o comando **dotnet publish**.

Este comando empacota o aplicativo e suas dependências em uma pasta para implantação em um sistema de hospedagem.

O comando **dotnet publish** compila o aplicativo, lê suas dependências especificadas no arquivo de projeto e publica o conjunto de arquivos resultantes em um diretório. A saída inclui os seguintes ativos:

- Código [IL \(Linguagem Intermediária\)](#) em um assembly com uma extensão **dll**.
- Arquivo **.deps.json** que inclui todas as dependências do projeto.
- Arquivo **.runtime.config.json** que especifica o tempo de execução compartilhado esperado pelo aplicativo, bem como outras opções de configuração para o tempo de execução;
- As dependências do aplicativo, que são copiadas do [cache NuGet](#) para a pasta de saída.

Ao término da execução do comando, podemos fazer a implantação da aplicação. No nosso caso iremos usar a saída do comando para gerar uma imagem no Docker.

No terminal, estando na pasta do projeto, digite o comando:

dotnet publish --configuration Release --output dist

```

macoratti@linux: ~/projetos/App1
Arquivo Editar Ver Pesquisar Terminal Ajuda
macoratti@linux:~/projetos/App1$ dotnet restore
  Restauração concluída em 76,36 ms para /home/macoratti/projetos/App1/App1.csproj.
macoratti@linux:~/projetos/App1$ sudo dotnet publish --configuration Release --output dist
Microsoft(R) Build Engine versão 15.8.169+g1ccb72aefa para .NET Core
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

  Restauração concluída em 61,3 ms para /home/macoratti/projetos/App1/App1.csproj.
  App1 -> /home/macoratti/projetos/App1/bin/Release/netcoreapp2.1/App1.dll
  App1 -> /home/macoratti/projetos/App1/bin/Release/netcoreapp2.1/App1.Views.dll
  App1 -> /home/macoratti/projetos/App1/dist/
macoratti@linux:~/projetos/App1$

```

Neste comando estamos criando um conjunto de arquivos que contém tudo que é preciso para rodar a aplicação. Usamos dois argumentos:

- 1- O argumento `--configuration Release` indica que estamos usando o modo **Release** que é o modo usado na produção.
- 2- O argumento `--output dist` especifica que o projeto compilado será copiado para uma pasta **dist**.

Nota: A partir do **.NET Core 2.0**, não é necessário executar **dotnet restore**, pois ele é executado de forma implícita por todos os comandos, como **dotnet build** e **dotnet run**, que exigem a ocorrência de uma restauração.

Ao final veremos a aplicação publicada na pasta **dist** e pronta para ser usada:

```

macoratti@linux: ~/projetos/App1/dist
Arquivo Editar Ver Pesquisar Terminal Ajuda
macoratti@linux:~/projetos/App1/dist$ ls -l
total 336
-rw-r--r-- 1 root 223232 nov  6 12:48 App1.deps.json
-rw-r--r-- 1 root  11264 nov  6 12:48 App1.dll
-rw-r--r-- 1 root   2172 nov  6 12:48 App1.pdb
-rw-r--r-- 1 root    213 nov  6 12:48 App1.runtimeconfig.json
-rw-r--r-- 1 root  71168 nov  6 12:48 App1.Views.dll
-rw-r--r-- 1 root   5632 nov  6 12:48 App1.Views.pdb
-rw-r--r-- 1 root   146 nov  4 17:05 appsettings.Development.json
-rw-r--r-- 1 root    105 nov  4 17:05 appsettings.json
-rw-r--r-- 1 root   453 nov  6 12:48 web.config
drwxr-xr-x 6 root  4096 nov  6 12:48 wwwroot
macoratti@linux:~/projetos/App1/dist$

```

Olhando o conteúdo da pasta **dist** vemos o arquivo **App1.dll**.

Este arquivo contém o código do projeto que criamos e fornece o ponto de entrada para executar a aplicação correspondente ao comando **ENTRYPOINT** no arquivo **Dockerfile**.

Vamos agora criar o arquivo **Dockerfile**.

Criando o arquivo Dockerfile

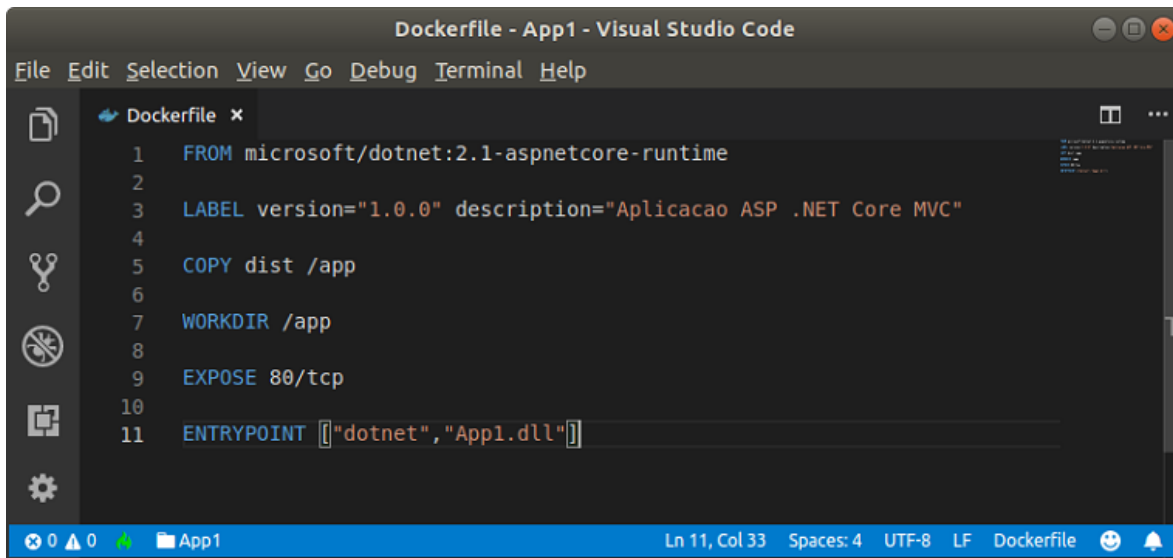
A primeira tarefa que temos que fazer é definir os passos que vamos usar para criar a imagem.

1. Definir uma imagem base
2. Definir informações para a imagem
3. Definir a pasta de trabalho
4. Copiar os arquivos da pasta **dist** para uma pasta no contêiner
5. Expor a porta do contêiner e definir em qual porta o servidor vai atender
6. Definir o ponto de entrada a aplicação

Vamos criar o arquivo **Dockerfile** na pasta da aplicação que é a pasta **App1**. Vamos entrar na pasta e abrir o Visual Studio Code digitando o comando `code`.

```
cd App1  
code .
```

Com o Visual Studio Code aberto crie o arquivo **Dockerfile** e inclua o código abaixo:



```
Dockerfile - App1 - Visual Studio Code  
File Edit Selection View Go Debug Terminal Help  
1 FROM microsoft/dotnet:2.1-aspnetcore-runtime  
2  
3 LABEL version="1.0.0" description="Aplicacao ASP .NET Core MVC"  
4  
5 COPY dist /app  
6  
7 WORKDIR /app  
8  
9 EXPOSE 80/tcp  
10  
11 ENTRYPOINT ["dotnet", "App1.dll"]  
Ln 11, Col 33 Spaces: 4 UTF-8 LF Dockerfile
```

Nota: O nome do arquivo tem que ser exatamente **Dockerfile**

Os comandos definidos no arquivo **Dockerfile** foram:

```
FROM microsoft/dotnet:2.1-aspnetcore-runtime  
LABEL version="1.0.1" description="Aplicacao ASP .NET Core MVC"  
COPY dist /app  
WORKDIR /app  
EXPOSE 80/tcp  
ENTRYPOINT ["dotnet", "App1.dll"]
```

Vamos entender cada uma das instruções usadas:

FROM microsoft/dotnet:2.1-aspnetcore-runtime

Definimos a imagem base como : [microsoft/dotnet:2.1-aspnetcore-runtime](#). Esta imagem já contém o SDK e os runtimes;

LABEL version="1.0.1" description="Aplicacao ASP .NET Core MVC"

Definimos a versão e a descrição;

COPY dist /app

Copia os arquivos da pasta **dist**, onde esta o deploy da nossa aplicação, para a pasta **/app** no contêiner;

WORKDIR /app

O comando **WORKDIR** define o diretório de trabalho para o contêiner, o que é útil se você precisar executar comandos ou usar arquivos sem precisar especificar o caminho completo a cada vez. No exemplo o comando define o caminho para a pasta **/app** que o comando **COPY** criou e que contém os arquivos do aplicativo;

EXPOSE 80/tcp

Esse comando informa ao Docker que ele pode disponibilizar a porta 80 para o tráfego TCP de fora do contêiner. Para o aplicativo de exemplo, isso é necessário para que o servidor ASP.NET Core Kestrel possa receber requisições HTTP;

ENTRYPOINT ["dotnet", "App1.dll"]

Este comando informa ao Docker para executar a ferramenta de linha de comando **dotnet** para executar o arquivo **App1.dll**, que criamos no deploy. O caminho para o arquivo **App1.dll** não precisa ser especificado porque é considerado dentro do diretório especificado pelo comando **WORKDIR**, que conterá todos os arquivos do aplicativo.

Dessa forma já temos o arquivo **Dockerfile** criado para gerar a imagem. Para fazer isso usamos o comando **build** e informamos o **nome da imagem**, a **tag** e um **ponto(.)**. O comando fica assim:

`docker build -t aspnetcoremvc/app1:1.0 .`

onde:

- docker build** -> O comando
- t** -> Parâmetro usado para informar que a imagem pertence ao meu usuário
- aspnetcoremvc/app1:1.0** -> O nome da imagem e a tag atribuída à imagem
- .** -> significa o diretório atual (*pois dei o build dentro da pasta do DockerFile*)

```

macoratti@linux: ~/projetos/App1
Arquivo Editar Ver Pesquisar Terminal Ajuda
macoratti@linux:~/projetos/App1$ sudo docker build -t aspnetcoremvc/app1:1.0 .
Sending build context to Docker daemon 7.873MB
Step 1/6 : FROM microsoft/dotnet:2.1-aspnetcore-runtime
--> 1fe6774e5e9e
Step 2/6 : LABEL version="1.0.0" description="Aplicacao ASP .NET Core MVC"
--> Running in c76ced865846
Removing intermediate container c76ced865846
--> 4d6a7042f935
Step 3/6 : COPY dist /appDemo
--> fa70ccc9e1b6
Step 4/6 : WORKDIR /appDemo
--> Running in 08639a6a3351
Removing intermediate container 08639a6a3351
--> b95dc5b980
Step 5/6 : EXPOSE 80/tcp
--> Running in 24f6d1f2dfc3
Removing intermediate container 24f6d1f2dfc3
--> 518495b5208f
Step 6/6 : ENTRYPOINT ["dotnet", "App1.dll"]
--> Running in a4c730c63c8e
Removing intermediate container a4c730c63c8e
--> a90c99ff854c
Successfully built a90c99ff854c
Successfully tagged aspnetcoremvc/app1:1.0
macoratti@linux:~/projetos/App1$

```

Exibindo as imagens: `docker images`

```

macoratti@linux: ~/projetos/App1
Arquivo Editar Ver Pesquisar Terminal Ajuda
macoratti@linux:~/projetos/App1$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
aspnetcoremvc/app1  1.0                a90c99ff854c       2 hours ago        257MB
macoratti/nginx     1.0                1be6668aea57       21 hours ago       202MB
microsoft/dotnet    2.1-aspnetcore-runtime 1fe6774e5e9e       2 weeks ago        255MB
nginx               latest             dbfc48660aeb       3 weeks ago        109MB
debian              latest             be2868bebababab    3 weeks ago        101MB
debian              8                  efdec82af25a       3 weeks ago        127MB
hello-world         latest             4ab4c602aa5e       8 weeks ago        1.84kB
macoratti@linux:~/projetos/App1$

```

Vemos a imagem **aspnetcoremvc/app1:1.0** criada com a **TAG** igual a **1.0**, o ID, data de criação e tamanho.

Agora para concluir, podemos criar um contêiner com a nossa imagem usando o comando:

`docker container create -p 3000:80 --name MvcContainer aspnetcoremvc/app1:1.0`

Aqui usamos o parâmetro **--name** para dar um novo ao contêiner criado, e estamos mapeando a porta 3000 para a porta 80.

Para exibir o contêiner criado em execução digite : **`docker container ps`**

```
macoratti@linux: ~/projetos/App1
Arquivo Editar Ver Pesquisar Terminal Ajuda
macoratti@linux:~/projetos/App1$ sudo docker container ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
1778a50c2747       aspnetcoremvc/app1:1.0  "dotnet App1.dll"   2 hours ago        Up 7 minutes       0.0.0.0:3000->80/tcp  MvcContainer
```

Vemos assim o nosso contêiner criado a partir de uma imagem que geramos localmente contendo a ASP.NET Core, os runtimes e nossa aplicação.

Agora basta iniciar o container digitando: **docker container start MvcContainer**

Para conferir basta abrir o navegador em <http://localhost:3000>.



ID	Nome	Categoria	Preco
10	Caneta	Escolar	\$100,00
20	Borracha	Escolar	\$200,00
30	Caderno	Escolar	\$300,00
40	Tablet	Eletrônicos	\$400,00
50	IPhone	Eletrônicos	\$500,00

Vemos nossa aplicação sendo executada a partir do contêiner que acabamos de criar.

Com isso acabamos de concluir o ciclo de criação da aplicação, criação de uma imagem e a criação e execução do contêiner para a nossa aplicação ASP .NET Core MVC.

Na [próxima aula](#) vamos veremos como trabalhar com [volumes](#).