

发布.NET Core3.0至Docker

胡鸿辉

https://blog.csdn.net/H_King_H

2019-11-12

创建.NET Core 3.0项目

创建新的 ASP.NET Core Web 应用程序

.NET Core

ASP.NET Core 3.0

 空

用于创建 ASP.NET Core 应用程序的空项目模板，此模板中没有任何内容。

 API

用于创建包含 RESTful HTTP 服务示例控制器的 ASP.NET Core 应用程序的项目模板，此模板还可以用于 ASP.NET Core MVC 视图和控制器。

 Web 应用程序

用于创建包含示例 ASP.NET Core Razor 页面内容的 ASP.NET Core 应用程序的项目模板。

 Web 应用程序(模型视图控制器)

用于创建包含示例 ASP.NET Core MVC 视图和控制器的 ASP.NET Core 应用程序的项目模板，此模板还可以用于 RESTful HTTP 服务。

 Angular

用于创建 ASP.NET Core 应用程序的项目模板，其中包含 Angular。

 React.js

用于创建 ASP.NET Core 应用程序的项目模板，其中包含 React.js。

[获取其他项目模板](#)

身份验证

不进行身份验证

[更改](#)

高级

☐ 为 HTTPS 配置(C)

☐ 启用 Docker 支持(E)
(需要 [Docker Desktop](#))

Linux

作者: Microsoft

源: .NET Core 3.0.0

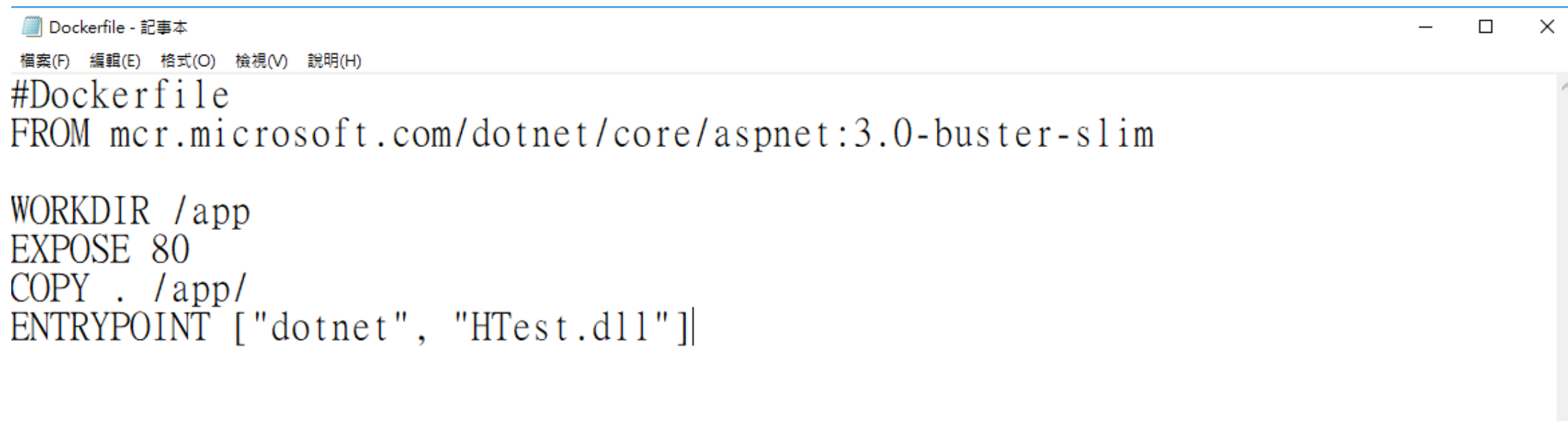
上一步(B)

创建

编写Dockerfile文件

- Dockerfile是一个文件格式的配置文件，用户可以使用dockerfile来快速构建自定义的镜像。由一行行命令语句组成，并且支持#开头的注释行。**Dockerfile文件是没有后缀名的**
- Dockerfile主题内容一般分为4部分
 - 基础镜像信息
 - 维护者信息使用label指令
 - 镜像操作指令
 - 容器启动时执行指令

Dockerfile文件说明



```
#Dockerfile
FROM mcr.microsoft.com/dotnet/core/aspnet:3.0-buster-slim

WORKDIR /app
EXPOSE 80
COPY . /app/
ENTRYPOINT [ "dotnet", "HTest.dll" ]
```

FROM:指定所创建镜像的基础镜像

WORKDIR-配置工作目录

EXPOSE-声明镜像内服务监听的端口

COPY-复制内容到镜像

ENTRYPOINT-启动镜像的默认入口命令

利用PSCP上传文件至Linux

- 命令： pscp -r D:\LinuxByWindows\Htest [root@0.0.0.0:/usr/local/docker/](#)
- Pscp {0} -r {1}@{2}:{3}
- {0}：Windows文件地址，文件地址中尽量不要出现中文
- {1}：Linux服务器用户名，如：root
- {2}：Linux服务器地址，也就是Linux的Server IP
- {3}：文件上传到Linux的最终地址，也就是目录

可参考我之前写的PPT：Putty pscp Windows与Linux文件互传.pptx

文件已上传成功，如下图：

```
[root@i57-25g314-245-1-1-1 ~]# cd /usr/local/docker/
[root@i57-25g314-245-1-1-1 docker]# ls
HTest
[root@i57-25g314-245-1-1-1 docker]#
```

根据Dockerfile构建镜像

- 我的Dockerfile是在发布文件夹下的

本機磁碟 (D:) > LinuxByWindows > <u>HTest</u>				
名稱	修改日期	類型	大小	
appsettings.Development.json	2019/11/11 下午 ...	JSON File	1 KB	
appsettings.json	2019/11/11 下午 ...	JSON File	1 KB	
<u>Dockerfile</u>	2019/11/11 下午 ...	檔案	1 KB	
HTest	2019/11/11 下午 ...	檔案	85 KB	
HTest.deps.json	2019/11/11 下午 ...	JSON File	112 KB	
HTest.dll	2019/11/11 下午 ...	應用程式擴充	9 KB	
HTest.pdb	2019/11/11 下午 ...	Program Debug ...	2 KB	
HTest.runtimeconfig.json	2019/11/11 下午 ...	JSON File	1 KB	
web.config	2019/11/11 下午 ...	XML Configurati...	1 KB	

根据Dockerfile构建镜像

- Linux进入HTest文件夹

```
[root@localhost ~]# cd HTest  
[root@localhost ~]#
```

- 构建镜像命令：**docker build -t hest-core-mvc .**

-t ---指定镜像名称

命令结尾处**.**---表示build上下文为当前目录，默认情况下docker会使用在上下文的**根目录**下找到**Dockerfile**文件

根据Dockerfile构建镜像

docker build -t hest-core-mvc .

```
[root@12w239g2:~/HTest]# cd HTest
[root@12w239g2:~/HTest]# docker build -t hest-core-mvc .
Sending build context to Docker daemon 220.7 kB
Step 1/5 : FROM mcr.microsoft.com/dotnet/core/aspnet:3.0-buster-slim
--> 930743cb4e19
Step 2/5 : WORKDIR /app
--> 21536c701200
Removing intermediate container 99f634b2285c
Step 3/5 : EXPOSE 80
--> Running in 82a1155d6055
--> 87f16d820604
Removing intermediate container 82a1155d6055
Step 4/5 : COPY . /app/
--> 6c9d2129f9b6
Removing intermediate container 92c59323b450
Step 5/5 : ENTRYPOINT dotnet HTest.dll
--> Running in 79218c2c5e1c
--> 821fe5e24ddd
Removing intermediate container 79218c2c5e1c
Successfully built 821fe5e24ddd
[root@12w239g2:~/HTest]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<u>hest-core-mvc</u>	latest	821fe5e24ddd	10 seconds ago	207 MB
mcr.microsoft.com/dotnet/core/aspnet	3.0-buster-slim	930743cb4e19	3 weeks ago	207 MB

构建容器

执行命令：`docker run --name hest-core-mvc -d -p 50879:80 hest-core-mvc`

参数说明

-d ,表示在后台以守护态（**daemonized**）形式运行容器

-p 外部端口与内部容器端口映射。

--name 指定容器的名称。当然可以不指定，默认会为我们创建

最后一个参数 **hest-core-mvc** 就是我们刚创建的镜像名称

通过**docker ps** 查看启动（**status:Up**）的容器 **docker ps -a** 可以看到未启动(**status:Exited**)容器 如果启动失败，如果程序发布没问题，那基本上是**Dockerfile**的文件，请仔细检查，尤其是**copy**命令和**ENTRYPOINT**命令

```
[root@12w239g5n1a2151 HTest]# docker run --name hest-core-mvc -d -p 50879:80 hest-core-mvc
47b5dd5522becd1b8f7b2086d11aea3701543aebb9894233cd591140095832a
[root@12w239g5n1a2151 HTest]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
47b5dd5522be	hest-core-mvc	"dotnet HTest.dll"	11 seconds ago	Up 11 seconds	0.0.0.0:50879->80/tcp	hest-core-mvc ✓

```
[root@12w239g5n1a2151 HTest]#
```

验证发布结果（请注意打开相应端口）

GET http://192.168.1.105:50879/weatherforecast

Params Authorization Headers (1) Body ● Pre-request Script Tests

KEY	VALUE
Key	Value

Body Cookies Headers (4) Test Results

Pretty Raw Preview JSON ↺

```
1 {
2   {
3     "date": "2019-11-10T23:11:15",
4     "temperatureC": 54,
5     "temperatureF": 129,
6     "summary": "Balmy"
7   },
8   {
9     "date": "2019-11-10T23:11:15",
10    "temperatureC": 11,
11    "temperatureF": 51,
12    "summary": "Scorching"
13  },
14  {
15    "date": "2019-11-10T23:11:15",
16    "temperatureC": -5,
17    "temperatureF": 24,
18    "summary": "Warm"
19  },
20  {
21    "date": "2019-11-10T23:11:15",
22    "temperatureC": -19,
23    "temperatureF": -2,
24    "summary": "Scorching"
25  },
26  {
27    "date": "2019-11-10T23:11:15",
28    "temperatureC": 32,
29    "temperatureF": 89,
30    "summary": "Freezing"
31  }
32 }
```

停止指定容器

- 第一步：docker ps 查看所有启动的容器
- 第二步：docker stop {0} 停止指定容器，容器ID
- 第三（四）步：查看容器docker ps -a (可查询到未启动容器)

```
[root@izwz9gdn4z7f2lcn1vds3 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
94430346b5da      htest-core-mvc    "dotnet HTest.dll"  21 hours ago       Up 21 hours        0.0.0.0:50879->80/tcp netcore-mvc

[root@izwz9gdn4z7f2lcn1vds3 ~]# docker stop 94430346b5da
94430346b5da

[root@izwz9gdn4z7f2lcn1vds3 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
94430346b5da      htest-core-mvc    "dotnet HTest.dll"  21 hours ago       Exited (0) 5 seconds ago
[redacted]

[root@izwz9gdn4z7f2lcn1vds3 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
94430346b5da      htest-core-mvc    "dotnet HTest.dll"  21 hours ago       Exited (0) 5 seconds ago
[redacted]
```

删除指定容器

- 删除指令，删除名字中帶htest的容器：

`docker rm -f $(docker ps -a | grep "htest*" | awk '{print $1}')`

```
root@iZ7z5q3t1z4t0k0l0d05:~# docker rm -f $(docker ps -a | grep "htest*" | awk '{print $1}')
```

```
94430346b5da
```

```
root@iZ7z5q3t1z4t0k0l0d05:~# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

```
root@iZ7z5q3t1z4t0k0l0d05:~# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

删除镜像

- 查看镜像：docker images
- 删除镜像，删除名字中有htest的镜像：

docker rmi --force `docker images | grep htest | awk '{print \$3}'`

```
[root@172.30.0.1 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hctest-core-mvc      latest             ab4044c64184       22 hours ago       207 MB
mcr.microsoft.com/dotnet/core/aspnet  3.0-buster-slim    930743cb4e19       3 weeks ago        207 MB
[root@172.30.0.1 ~]# docker rmi --force `docker images | grep htest | awk '{print $3}'`
Untagged: hctest-core-mvc:latest
Deleted: sha256:ab4044c64184ee95d8fcf7479f9c7f82b3869be0a6a728a1945ca314ec62d9e6
Deleted: sha256:13cdfd657bf1bd721efb0686ff9f7330b512ff819dcf294bcdfb578446ac4eee
Deleted: sha256:52520e8588db53288db967f6bf53b3f17bea4777b9aa798fc13ea33dd65f1219
Deleted: sha256:7af5a58effbfaab427ada945e15942ba8d82ea2172583b3570a8700d548dcdb0
Deleted: sha256:21b16a079c2e7883708988ccd20775693e4c7e59f5570ad3cde3da8e3792322b
Deleted: sha256:3d68c749093bd5078473966d8ebab5910ed53f9178502141d71ac007a8e0d75e
[root@172.30.0.1 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mcr.microsoft.com/dotnet/core/aspnet  3.0-buster-slim    930743cb4e19       3 weeks ago        207 MB
[root@172.30.0.1 ~]#
```

Docker常用基本命令

docker start {0} 一启动
docker stop {0} ---停止
docker ps --查看容器列表
docker images --查看镜像列表

可查询：

<https://www.runoob.com/docker/docker-command-manual.html>

```
Management Commands:
  container  Manage containers
  image      Manage images
  network    Manage networks
  node       Manage Swarm nodes
  plugin     Manage plugins
  secret     Manage Docker secrets
  service    Manage services
  stack      Manage Docker stacks
  swarm      Manage Swarm
  system     Manage Docker
  volume     Manage volumes

Commands:
  attach     Attach to a running container
  build      Build an image from a Dockerfile
  commit     Create a new image from a container's changes
  cp         Copy files/folders between a container and the local filesystem
  create     Create a new container
  diff       Inspect changes on a container's filesystem
  events     Get real time events from the server
  exec       Run a command in a running container
  export     Export a container's filesystem as a tar archive
  history    Show the history of an image
  images     List images
  import     Import the contents from a tarball to create a filesystem image
  info       Display system-wide information
  inspect    Return low-level information on Docker objects
  kill       Kill one or more running containers
  load       Load an image from a tar archive or STDIN
  login      Log in to a Docker registry
  logout     Log out from a Docker registry
  logs       Fetch the logs of a container
  pause     Pause all processes within one or more containers
  port       List port mappings or a specific mapping for the container
  ps         List containers
  pull       Pull an image or a repository from a registry
  push       Push an image or a repository to a registry
  rename     Rename a container
  restart    Restart one or more containers
  rm         Remove one or more containers
  rmi        Remove one or more images
  run        Run a command in a new container
  save       Save one or more images to a tar archive (streamed to STDOUT by default)
  search     Search the Docker Hub for images
  start      Start one or more stopped containers
  stats      Display a live stream of container(s) resource usage statistics
  stop       Stop one or more running containers
  tag        Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top        Display the running processes of a container
  unpause   Unpause all processes within one or more containers
  update     Update configuration of one or more containers
  version    Show the Docker version information
  wait       Block until one or more containers stop, then print their exit codes
```