**Dockerfile on Windows**

* 05/03/2019
* 10 minutes to read
  + [](https://github.com/PatrickLang)
  + [](https://github.com/eross-msft)
  + [](https://github.com/cwilhit)
  + [](https://github.com/scooley)
  + [](https://github.com/heoelri)
  + +4

The Docker engine includes tools that automate container image creation. While you can create container images manually by running the docker commit command, adopting an automated image creation process has many benefits, including:

* Storing container images as code.
* Rapid and precise recreation of container images for maintenance and upgrade purposes.
* Continuous integration between container images and the development cycle.

The Docker components that drive this automation are the Dockerfile, and the docker build command.

The Dockerfile is a text file that contains the instructions needed to create a new container image. These instructions include identification of an existing image to be used as a base, commands to be run during the image creation process, and a command that will run when new instances of the container image are deployed.

Docker build is the Docker engine command that consumes a Dockerfile and triggers the image creation process.

This topic will show you how to use Dockerfiles with Windows containers, understand their basic syntax, and what the most common Dockerfile instructions are.

This document will discuss the concept of container images and container image layers. If you want to learn more about images and image layering, see [container base images](https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/container-base-images).

For a complete look at Dockerfiles, see the [Dockerfile reference](https://docs.docker.com/engine/reference/builder/).

**Basic Syntax**

In its most basic form, a Dockerfile can be very simple. The following example creates a new image, which includes IIS, and a ‘hello world’ site. This example includes comments (indicated with a #), that explain each step. Subsequent sections of this article will go into more detail on Dockerfile syntax rules, and Dockerfile instructions.

**Note**

A Dockerfile must be created with no extension. To do this in Windows, create the file with your editor of choice, then save it with the notation "Dockerfile" (including the quotes).

DockerfileCopy

# Sample Dockerfile

# Indicates that the windowsservercore image will be used as the base image.

FROM mcr.microsoft.com/windows/servercore:ltsc2019

# Metadata indicating an image maintainer.

LABEL maintainer="jshelton@contoso.com"

# Uses dism.exe to install the IIS role.

RUN dism.exe /online /enable-feature /all /featurename:iis-webserver /NoRestart

# Creates an HTML file and adds content to this file.

RUN echo "Hello World - Dockerfile" > c:\inetpub\wwwroot\index.html

# Sets a command or process that will run each time a container is run from the new image.

CMD [ "cmd" ]

For additional examples of Dockerfiles for Windows, see the [Dockerfile for Windows repository](https://github.com/Microsoft/Virtualization-Documentation/tree/master/windows-container-samples).

**Instructions**

Dockerfile instructions provide the Docker Engine the instructions it needs to create a container image. These instructions are performed one-by-one and in order. The following examples are the most commonly used instructions in Dockerfiles. For a complete list of Dockerfile instructions, see the [Dockerfile reference](https://docs.docker.com/engine/reference/builder/).

**FROM**

The FROM instruction sets the container image that will be used during the new image creation process. For instance, when using the instruction FROM mcr.microsoft.com/windows/servercore, the resulting image is derived from, and has a dependency on, the Windows Server Core base OS image. If the specified image is not present on the system where the Docker build process is being run, the Docker engine will attempt to download the image from a public or private image registry.

The FROM instruction's format goes like this:

DockerfileCopy

FROM <image>

Here's an example of the FROM command:

To download the ltsc2019 version windows server core from the Microsoft Container Registry (MCR):

Copy

FROM mcr.microsoft.com/windows/servercore:ltsc2019

For more detailed information, see the [FROM reference](https://docs.docker.com/engine/reference/builder/#from).

**RUN**

The RUN instruction specifies commands to be run, and captured into the new container image. These commands can include items such as installing software, creating files and directories, and creating environment configuration.

The RUN instruction goes like this:

DockerfileCopy

# exec form

RUN ["<executable>", "<param 1>", "<param 2>"]

# shell form

RUN <command>

The difference between the exec and shell form is in how the RUN instruction is executed. When using the exec form, the specified program is run explicitly.

Here's an example of the exec form:

Copy

FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN ["powershell", "New-Item", "c:/test"]

The resulting image runs the powershell New-Item c:/test command:

DockerfileCopy

docker history doc-exe-method

IMAGE CREATED CREATED BY SIZE COMMENT

b3452b13e472 2 minutes ago powershell New-Item c:/test 30.76 MB

To contrast, the following example runs the same operation in shell form:

Copy

FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN powershell New-Item c:\test

The resulting image has a run instruction of cmd /S /C powershell New-Item c:\test.

DockerfileCopy

docker history doc-shell-method

IMAGE CREATED CREATED BY SIZE COMMENT

062a543374fc 19 seconds ago cmd /S /C powershell New-Item c:\test 30.76 MB

**Considerations for using RUN with Windows**

On Windows, when using the RUN instruction with the exec format, backslashes must be escaped.

DockerfileCopy

RUN ["powershell", "New-Item", "c:\\test"]

When the target program is a Windows installer, you'll need to extract the setup through the /x:<directory> flag before you can launch the actual (silent) installation procedure. You must also wait for the command to exit before you do anything else. Otherwise, the process will end prematurely without installing anything. For details, please consult the example below.

**Examples of using RUN with Windows**

The following example Dockerfile uses DISM to install IIS in the container image:

DockerfileCopy

RUN dism.exe /online /enable-feature /all /featurename:iis-webserver /NoRestart

This example installs the Visual Studio redistributable package. Start-Process and the -Wait parameter are used to run the installer. This ensures that the installation completes before moving on to the next instruction in the Dockerfile.

DockerfileCopy

RUN powershell.exe -Command Start-Process c:\vcredist\_x86.exe -ArgumentList '/quiet' -Wait

For detailed information on the RUN instruction, see the [RUN reference](https://docs.docker.com/engine/reference/builder/#run).

**COPY**

The COPY instruction copies files and directories to the container's file system. The files and directories must be in a path relative to the Dockerfile.

The COPY instruction's format goes like this:

DockerfileCopy

COPY <source> <destination>

If either source or destination includes white space, enclose the path in square brackets and double quotes, as shown in the following example:

DockerfileCopy

COPY ["<source>", "<destination>"]

**Considerations for using COPY with Windows**

On Windows, the destination format must use forward slashes. For example, these are valid COPY instructions:

DockerfileCopy

COPY test1.txt /temp/

COPY test1.txt c:/temp/

Meanwhile, the following format with backslashes won't work:

DockerfileCopy

COPY test1.txt c:\temp\

**Examples of using COPY with Windows**

The following example adds the contents of the source directory to a directory named sqllite in the container image:

DockerfileCopy

COPY source /sqlite/

The following example will add all files that begin with config to the c:\temp directory of the container image:

DockerfileCopy

COPY config\* c:/temp/

For more detailed information about the COPY instruction, see the [COPY reference](https://docs.docker.com/engine/reference/builder/#copy).

**ADD**

The ADD instruction is like the COPY instruction, but with even more capabilities. In addition to copying files from the host into the container image, the ADD instruction can also copy files from a remote location with a URL specification.

The ADD instruction's format goes like this:

DockerfileCopy

ADD <source> <destination>

If either the source or destination include white space, enclose the path in square brackets and double quotes:

DockerfileCopy

ADD ["<source>", "<destination>"]

**Considerations for running ADD with Windows**

On Windows, the destination format must use forward slashes. For example, these are valid ADD instructions:

DockerfileCopy

ADD test1.txt /temp/

ADD test1.txt c:/temp/

Meanwhile, the following format with backslashes won't work:

DockerfileCopy

ADD test1.txt c:\temp\

Additionally, on Linux the ADD instruction will expand compressed packages on copy. This functionality is not available in Windows.

**Examples of using ADD with Windows**

The following example adds the contents of the source directory to a directory named sqllite in the container image:

DockerfileCopy

ADD source /sqlite/

The following example will add all files that begin with "config" to the c:\temp directory of the container image.

DockerfileCopy

ADD config\* c:/temp/

The following example will download Python for Windows into the c:\temp directory of the container image.

DockerfileCopy

ADD https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe /temp/python-3.5.1.exe

For more detailed information about the ADD instruction, see the [ADD reference](https://docs.docker.com/engine/reference/builder/#add).

**WORKDIR**

The WORKDIR instruction sets a working directory for other Dockerfile instructions, such as RUN, CMD, and also the working directory for running instances of the container image.

The WORKDIR instruction's format goes like this:

DockerfileCopy

WORKDIR <path to working directory>

**Considerations for using WORKDIR with Windows**

On Windows, if the working directory includes a backslash, it must be escaped.

DockerfileCopy

WORKDIR c:\\windows

**Examples**

DockerfileCopy

WORKDIR c:\\Apache24\\bin

For detailed information on the WORKDIR instruction, see the [WORKDIR reference](https://docs.docker.com/engine/reference/builder/#workdir).

**CMD**

The CMD instruction sets the default command to be run when deploying an instance of the container image. For instance, if the container will be hosting an NGINX web server, the CMD might include instructions to start the web server with a command like nginx.exe. If multiple CMD instructions are specified in a Dockerfile, only the last is evaluated.

The CMD instruction's format goes like this:

DockerfileCopy

# exec form

CMD ["<executable", "<param>"]

# shell form

CMD <command>

**Considerations for using CMD with Windows**

On Windows, file paths specified in the CMD instruction must use forward slashes or have escaped backslashes \\. The following are valid CMD instructions:

DockerfileCopy

# exec form

CMD ["c:\\Apache24\\bin\\httpd.exe", "-w"]

# shell form

CMD c:\\Apache24\\bin\\httpd.exe -w

However, the following format without the proper slashes will not work:

DockerfileCopy

CMD c:\Apache24\bin\httpd.exe -w

For more detailed information about the CMD instruction, see the [CMD reference](https://docs.docker.com/engine/reference/builder/#cmd).

**Escape character**

In many cases a Dockerfile instruction will need to span multiple lines. To do this, you can use an escape character. The default Dockerfile escape character is a backslash \. However, because the backslash is also a file path separator in Windows, using it to span multiple lines can cause problems. To get around this, you can use a parser directive to change the default escape character. For more information about parser directives, see [Parser directives](https://docs.docker.com/engine/reference/builder/#parser-directives).

The following example shows a single RUN instruction that spans multiple lines using the default escape character:

Copy

FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN powershell.exe -Command \

$ErrorActionPreference = 'Stop'; \

wget https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe -OutFile c:\python-3.5.1.exe ; \

Start-Process c:\python-3.5.1.exe -ArgumentList '/quiet InstallAllUsers=1 PrependPath=1' -Wait ; \

Remove-Item c:\python-3.5.1.exe -Force

To modify the escape character, place an escape parser directive on the very first line of the Dockerfile. This can be seen in the following example.

**Note**

Only two values can be used as escape characters: \ and `.

DockerfileCopy

# escape=`

FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN powershell.exe -Command `

$ErrorActionPreference = 'Stop'; `

wget https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe -OutFile c:\python-3.5.1.exe ; `

Start-Process c:\python-3.5.1.exe -ArgumentList '/quiet InstallAllUsers=1 PrependPath=1' -Wait ; `

Remove-Item c:\python-3.5.1.exe -Force

For more information about the escape parser directive, see [Escape parser directive](https://docs.docker.com/engine/reference/builder/#escape).

**PowerShell in Dockerfile**

**PowerShell cmdlets**

PowerShell cmdlets can be run in a Dockerfile with the RUN operation.

Copy

FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN powershell -command Expand-Archive -Path c:\apache.zip -DestinationPath c:\

**REST calls**

PowerShell's Invoke-WebRequest cmdlet can be useful when gathering information or files from a web service. For instance, if you build an image that includes Python, you can set $ProgressPreference to SilentlyContinue to achieve faster downloads, as shown in the following example.

Copy

FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN powershell.exe -Command \

$ErrorActionPreference = 'Stop'; \

$ProgressPreference = 'SilentlyContinue'; \

Invoke-WebRequest https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe -OutFile c:\python-3.5.1.exe ; \

Start-Process c:\python-3.5.1.exe -ArgumentList '/quiet InstallAllUsers=1 PrependPath=1' -Wait ; \

Remove-Item c:\python-3.5.1.exe -Force

**Note**

Invoke-WebRequest also works in Nano Server.

Another option for using PowerShell to download files during the image creation process is to use the .NET WebClient library. This can increase download performance. The following example downloads the Python software, using the WebClient library.

Copy

FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN powershell.exe -Command \

$ErrorActionPreference = 'Stop'; \

(New-Object System.Net.WebClient).DownloadFile('https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe','c:\python-3.5.1.exe') ; \

Start-Process c:\python-3.5.1.exe -ArgumentList '/quiet InstallAllUsers=1 PrependPath=1' -Wait ; \

Remove-Item c:\python-3.5.1.exe -Force

**Note**

Nano Server does not currently support WebClient.

**PowerShell scripts**

In some cases, it may be helpful to copy a script into the containers you use during the image creation process, then run the script from within the container.

**Note**

This will limit any image layer caching and decrease the Dockerfile's readability.

This example copies a script from the build machine into the container using the ADD instruction. This script is then run using the RUN instruction.

Copy

FROM mcr.microsoft.com/windows/servercore:ltsc2019

ADD script.ps1 /windows/temp/script.ps1

RUN powershell.exe -executionpolicy bypass c:\windows\temp\script.ps1

**Docker build**

Once a Dockerfile has been created and saved to disk, you can run docker build to create the new image. The docker build command takes several optional parameters and a path to the Dockerfile. For complete documentation on Docker Build, including a list of all build options, see the [build reference](https://docs.docker.com/engine/reference/commandline/build/#build).

The format of the docker build command goes like this:

DockerfileCopy

docker build [OPTIONS] PATH

For example, the following command will create an image named "iis."

DockerfileCopy

docker build -t iis .

When the build process has been initiated, the output will indicate status and return any thrown errors.

DockerfileCopy

C:\> docker build -t iis .

Sending build context to Docker daemon 2.048 kB

Step 1 : FROM mcr.microsoft.com/windows/servercore:ltsc2019

---> 6801d964fda5

Step 2 : RUN dism /online /enable-feature /all /featurename:iis-webserver /NoRestart

---> Running in ae8759fb47db

Deployment Image Servicing and Management tool

Version: 10.0.10586.0

Image Version: 10.0.10586.0

Enabling feature(s)

The operation completed successfully.

---> 4cd675d35444

Removing intermediate container ae8759fb47db

Step 3 : RUN echo "Hello World - Dockerfile" > c:\inetpub\wwwroot\index.html

---> Running in 9a26b8bcaa3a

---> e2aafdfbe392

Removing intermediate container 9a26b8bcaa3a

Successfully built e2aafdfbe392

The result is a new container image, which in this example is named "iis."

DockerfileCopy

docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

iis latest e2aafdfbe392 About a minute ago 207.8 MB

windowsservercore latest 6801d964fda5 4 months ago 0 B

**Further reading and references**

* [Optimize Dockerfiles and Docker build for Windows](https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-docker/optimize-windows-dockerfile)
* [Dockerfile reference](https://docs.docker.com/engine/reference/builder/)