Windows Docker SQL Server mit Datenbank

Kathegorie
SQL Server

Benedikt
Schackenberg

Was ist ein Docker Container?

Unter einem Docker Container versteht man einen isolierten, kompakten Ausführungskontext für eine App oder einen Dienst, die, gemeinsam mit anderen Containern, einen Kernel nutzen. Durch die Verwendung von Container kann eine App bereitgestellt und von jedem Docker-Host konsistent ausgeführt und Risiken von Environment-Problemen wie beispielsweise Inkompatible Versionen von Bibliotheken auf dem Host-Computer oder die Beeinträchtigung externer Prozesse vermieden werden.

Docker Container wurden ursprünglich unter Linux entwickelt, können jetzt aber auch unter Windows 10 und Windows Server 2016 /2019 erstellt und ausgeführt werden. In einem großen Pool von Container-Hosts, auch Docker-Schwarm (engl. Docker Swarm) genannt, kann eine Mischung aus Windows- und Linux-Hosts verwendet und ein Stapel (engl. Stack) mit beiden Container-Typen erstellt werden. Diese Art von Stack ist dann ein Hybrid Application Stack.
Um nun einen Container in einen solchen Hybrid Application Stack zu integrieren wird in dieser Anleitung ein Windows-Container mit einem MSSQL-Server-Developer Image mit bereits eingefügter Datenbank erstellt. Als Datenbank wird hier die von Microsoft bereitgestellte Beispiel-Datenbank Wide World Importers verwendet.

Voraussetzungen

Tool	Link
Docker für Windows	https://www.docker.com/products/docker-desktop

Nachdem Docker für Windows erfolgreich installiert wurde, muss es so konfiguriert werden, dass es kompatibel mit Windows-Containern ist. Hiefür lediglich das Docker-Icon rechtsklicken und switch to windows containers ausgewählen.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Loslegen

Als ersten Schritt die PowerShell Konsole als Administrator öffnen und das Kommando

```
# Create a project directory to build the image from.
New-Item -ItemType Directory -Path "$env:UserProfile\Projects\
DockerWideWorldImporters"
Set-Location -Path "$env:UserProfile\Projects\
DockerWideWorldImporters"

# Download WideWorldImporters-Full.bak from GitHub.
Invoke-WebRequest -UseBasicParsing -Uri https://github.com/
Microsoft/sql-server-samples/releases/download/wide-world-
importers-v1.0/WideWorldImporters-Full.bak -OutFile .\
WideWorldImporters-Full.bak

# Create Dockerfile for image.
New-Item -Name Dockerfile
```

ausführen. Hier wird nun ein Repository erstellt, um alle für den Bau des Containers, relevanten Daten zu speichern. Weiterhin wird die Backup-Datei der Wide World Imports Datenbank von Git heruntergeladen und ein Dockerfile mit dem Namen Dockerfile erstellt und in dem Repository gespeichert.

Der Bau des Containers

Die Docker Images werden in einem sogenannten Docker-File definiert. Das Docker-File, welches im vorherigen Schritt für den Windows-Container erstellt, würde so aussehen:

```
# escape=`

FROM microsoft/mssql-server-windows-developer

ENV user_name _
ENV user_password _

WORKDIR C:\

ADD .\WideWorldImporters-Full.bak C:\
ADD .\Invoke-BuildActions.ps1 C:\
ADD .\Invoke-RunActions.ps1 C:\

SHELL ["powershell", "-File"]

RUN .\Invoke-BuildActions.ps1

EXPOSE 1433

CMD .\Invoke-RunActions.ps1 -Username $env:user_name -Password $env:user_password
-Verbose
```

Zeile 1. Hier wird der Escape-Character von \ auf ‘ geändert, da wir hier einen Windows Container definieren und das Symbol \ für Dateipfade verwendet werden muss und Docker so keine Verwechslungen unterlaufen.

Zeile 3. Jedes Docker-Image hat ein übergeordnetes Image, ein sogenanntes parent image. In diesem Fall ist das parent image das MSSQL-Server-Windows-Developer Image. Dieses installiert SQL Server 2016 Developer SP1 und setzt das Kennwort des SA-Kontos auf einen angegebenen Wert zurück.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Zeilen 5-6. Mit der Anweisung ENV werden Umgebungsvariablen in der Laufzeitumgebung gesetzt. Üblicherweise werden beim Erstellen eines Containers über dieses Image mit dem Befehl docker run Werte für diese Parameter übergeben. In diesem Fall erstellt das Image ein Benutzerkonto mit Lese – und Schreibrechten für die Wide World Importers Datenbank. Die hier verwendeten Umgebungsvariablen stehen für den Benutzernamen und das Kennwort die bei der Erstellung dieses Kontos verwendet werden sollen.

Zeile 8. Hier wird das Verzeichnis festgelegt, das zum Ausführen andere ADD-, RUN- oder CMD-Anweisungen in dieser Docker-Datei verwendet werden soll.

Zeilen 10-12. Mit der ADD-Anweisung werden Dateien und Verzeichnisse vom Host-Dateisystem zum Image-Dateisystem hinzugefügt. Hier werden die Backup-Datei der Datenbank und zwei PowerShell-Skripts hinzugefügt. Die Bedeutung der PowerShell-Skripts werden später noch erläutert.

Zeile 14. Mit dieser Anweisung wird die Standard-Shell festgelegt, welche beim Ausführen von RUN- oder CMD- Anweisungen verwendet werden soll. Für Windows-Container lautet der Standard-Shell-Befehl [”cmd”, ”/S“, ”/C“]. Da in disem Beispiel PowerShell-Skripte verwendet werden, wird der Befehl mit ”Powershell”, ”-File“ überschrieben.

Zeile 16. RUN-Anweisungen werden als Teil des Erstellungsprozesses des Docker-Images ausgeführt. Hier wird das Skript Invoke-BuildActions.ps1 ausgeführt.

Zeile 18. Die Anweisung EXPOSE teilt docker mit, dass der Container zur Laufzeit auf den angegebenen Port hören darf. Der Standardport für SQL Server ist 1433.

Zeile 20. Die CMD-Anweisung definiert den Befehl, der ausgeführt wird sobald ein Container aus diesem Image erstellt wird. Es kann pro Image nur eine von diesen Anweisungen geben.

Es sollte nicht vergessen werden dass RUN- Anweisungen zur Erstellungszeit und CMD-Anweisungen zu Laufzeit, also bein Starten eines Containers, ausgeführt werden.

Die PowerShell Skripts

Nun werden die im Dockerfile erwähnten und verwendeten PowerShell Skripts definiert, erstellt und erläutert. Um die beiden Skripts zu erstellen und direkt mit PowerShell ISE zu öfnnen, einfach folgende Commandos in der PowerShell Konsole (Als Administrator, ganz wichtig!) ausführen:

```
New-Item -Name Invoke-BuildActions.ps1
New-Item -Name Invoke-RunActions.ps1

powershell_ise.exe "Invoke-BuildActions.ps1, Invoke-RunActions.ps1"
```

Invoke-BuildActions

```
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201
-Force
Set-PSRepository -Name PSGallery -InstallationPolicy Trusted
Install-Module SqlServer -Force -AllowClobber

Import-Module SqlServer


New-Item -Type Directory -Path C:\SqlData

Invoke-Sqlcmd -ServerInstance localhost -Query `
"USE [master] RESTORE DATABASE [WideWorldImporters] `
FROM DISK = N'C:\WideWorldImporters-Full.bak' `
WITH MOVE N'WWI_Primary' TO N'C:\SqlData\WideWorldImporters.mdf', `
MOVE N'WWI_UserData' TO N'C:\SqlData\WideWorldImporters_UserData.
ndf', `
MOVE N'WWI_Log' TO N'C:\SqlData\WideWorldImporters.ldf', `
MOVE N'WWI_InMemory_Data_1' TO N'C:\SqlData\WorldWideImporters_
InMemory_Data_1'"
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

In diesem Skript wird:

- Ein neues SqlServer Modul aus der PowerShell Galerie installiert. Das parent image bringt zwar bereits das SQLPS PowerShell Modul mit sich, aber das SqlServer Modul ist eine wesentlich bessere Version.
- Ein Directory erstellt, in welches einige der WideWorldImporters-Datenbankdateien beim Wiederherstellten verschoben werden sollen.
- Das SQL-Kommando zum wiederherstellen der Datenbank mithilfe der .bak Datei ausgeführt. Der Speicherort der Datenbankdateien muss mit MOVE-Deklarationen geändert werden, da der Befehl RESTORE andernfalls versucht, dieselben Dateispeicherorte zu verwenden, von denen die Sicherung stammt.

Invoke-RunActions

```
Param (
    [Parameter(Mandatory=$true)][string] $Username,
    [Parameter(Mandatory=$true)][string] $Password
)

if($Username -like "$*") {
    $Username = Invoke-Expression -Command $Username
}
if($Password -like "$*") {
    $Password = Invoke-Expression -Command $Password
}

if($Username -eq "_") {
    Write-Host "ERROR: A name for the new user account is required.
Please supply a '--env user_name' variable with the 'docker run'
command."
    Exit(1)
}
if($Password -eq "_") {
    Write-Host "ERROR: A password for the new user account is required. Please
supply a '--env user_password' variable with the 'docker run' command."
    Exit(1)
}

$SecurePassword = ConvertTo-SecureString -String $Password -AsPlainText -Force
$Credential = New-Object -TypeName System.Management.Automation.PSCredential
-ArgumentList $Username, $SecurePassword
Add-SqlLogin -ServerInstance localhost -LoginName $Username -LoginType SqlLogin
-DefaultDatabase WideWorldImporters -Enable -GrantConnectSql -LoginPSCredential
$Credential | Out-Null
Write-Verbose "Created SQL login for user $Username."

$Server = New-Object -TypeName Microsoft.SqlServer.Management.Smo.Server
-ArgumentList localhost
$Database = $Server.Databases['WideWorldImporters']

$User = New-Object -TypeName Microsoft.SqlServer.Management.Smo.User
-ArgumentList $Database, $Username
$User.Login = $Username
$User.Create()
Write-Verbose "Created user $Username."

Invoke-Sqlcmd -ServerInstance localhost -Database WideWorldImporters -Query
"ALTER ROLE db_datareader ADD MEMBER $Username"
Invoke-Sqlcmd -ServerInstance localhost -Database WideWorldImporters -Query
"ALTER ROLE db_datawriter ADD MEMBER $Username"
Write-Verbose "Granted user $Username read and write access to Database:
WideWorldImporters."

.\start.ps1 -sa_password $env:sa_password -ACCEPT_EULA $env:ACCEPT_EULA -attach_
dbs "$env:attach_dbs" -Verbose
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

In diesem Skript werden:

- Neue Umgebungsvariablen für den Benutzernamen und das Kennwort für ein neues Benutzerkonto angelegt.
- Ein Benutzerkonto mit dem angegebenen Benutzernamen und Kennwort erstellt, und diesem Lese- und Schreibrechte für die WideWorldImporters-Datenbank vergeben.
- Das PowerShell Skript start.ps1 ausgeführt. Dieses Skript wird eigentlich von der CMD-Anweisung des übergeordneten Docker-Image MSSQL-Server-Windows-Developer ausgeführt. Dieses würde aber das momentane Skript überschreiben weshalb es hier ausgeführt werden muss. Somit geht sein Laufzeitverhalten nicht verloren. Für dieses Skript muss die SQL-Server-Endbenutzer-Lizenzvereinbarung akzeptiert werden.

Das Image bauen und den Container starten

Nun kann das Image "gebuildet", also gebaut und ein Container dieses Images erstellt werden. Hierfür müssen in der PowerShell Konsole folgende Befehle ausgeführt werden:

```
docker build --tag wideworldimporters .

docker run --detach --env sa_password=AdminPass123! --env user_
name=WWIUser --env user_password=UserPass123! --env ACCEPT_EULA=Y
wideworldimporters

docker ps

docker exec --interactive <ContainerId> powershell
```

Zeile 1. Hier wird das Image gebaut, die Wiederherstellung durchgeführt usw.

Zeile 3. Hier wird ein neuer Container mit dem vorher gebauten Image erstellt. Es werden Benutzername und Passwort für einen Benutzer erstellt, ein Passwort für den Administrator festgelegt und die EULA akzeptiert.

Zeile 5. Mit diesem Kommando kann zum einen überprüft werden ob der Container gestartet ist, und zum anderen die ID des Containers abgelesen werden.

Zeile 7. Startet eine PowerShell Session auf dem Container.

Um nun zu prüfen ob die Datenbank betriebsbereit ist, kann nun auf dem Container folgendes Kommando ausgeführt werden:

```
Invoke-Sqlcmd -ServerInstance localhost -Database WideWorldImporters `
-Username WWIUser -Password UserPass123! `
-Query "SELECT Count(CityId) AS NumCities FROM Application.Cities"
```

Als Output sollte nun

```
NumCities
---------
    37940
```

zu sehen sein.

Somit wurde erfolgreich ein Container mit der Datenbank von WideWolrdImporters erstellt.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612