# Lesson2 - Deploy MSSQL in Kuberneties cluster

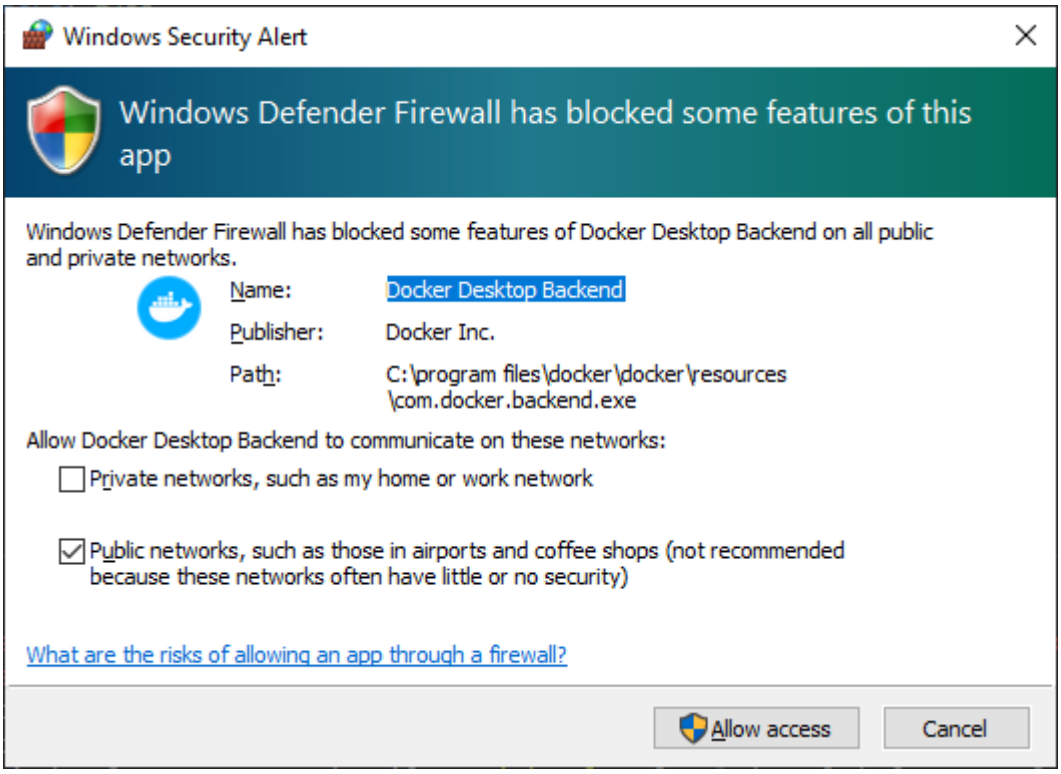
In Lesson1 we cloned the source code, build and run the application but it failed because we didn't install MSSQL. In order to deploy MSSQL server in Kuberneties cluster we will use the commands below:

**cd ..**

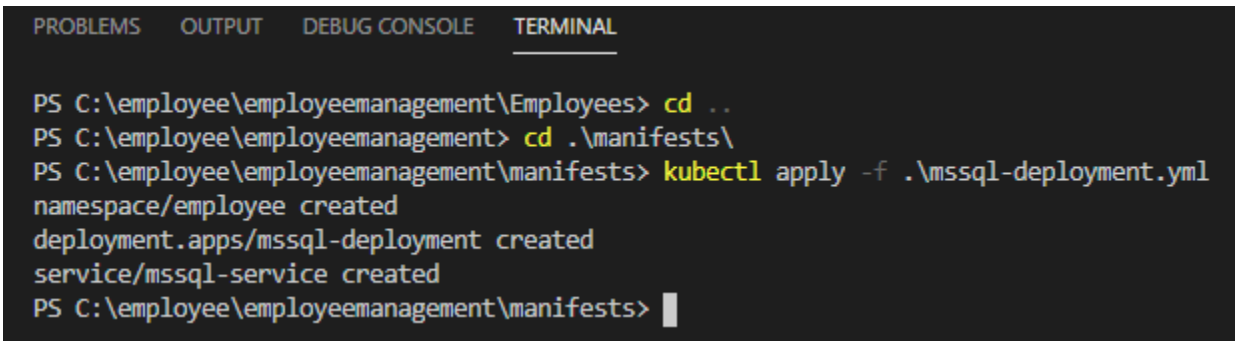
**cd manifests**

**kubectl apply -f .\mssql-deployment.yml**

Click **Allow Access** for the Windows Security Alert:




You will see that the deployment was created.



**kubectl get all -n employee** - this command shows the deployment to Kuberneties.

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\employee\employeemanagement\manifests> kubectl get all -n employee
NAME                                     READY   STATUS    RESTARTS   AGE
pod/mssql-deployment-cc564c444-6dv79     1/1     Running   0          76s

NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/mssql-service   LoadBalancer   10.109.174.249   localhost     1433:32751/TCP   77s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment apps/mssql-deployment    1/1     1            1           77s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset apps/mssql-deployment-cc564c444     1         1         1       77s
PS C:\employee\employeemanagement\manifests>
```

**We can see in the screen above some Kuberneties components:**

**Namespace –** logic area named **employee** that contains all the components that we deployed below.

**Deployment -** deploy pods with several settings named **mssql-deployment**.

**Pod** – the MSSQL server is running in a docker container within a pod named **mssql**.

**Replicaset –** shows how many replicas of the application, MSSQL, are running. In or case **1** replica.

**Service –** the pod of the MSSQL server has service named **mssql-service** that enables to connect the pod internally within the Kuberneties cluster with **CLUSTER-IP (10.100.124.249)** and externally in our PC with **EXTERNL_IP (localhost)**

The **mssql-deployment.yml** contains the Microsoft SQL Server docker image which exists in **DockerHub**:

https://hub.docker.com/_/microsoft-mssql-server

| | | | | | |
|---|---|---|---|---|---|
| 2017-CU10-ubuntu | amd64 | No Dockerfile | Ubuntu 16.04 | 09/13/2018 03:18:19 | 09/13/2018 03:18:19 |
| 2017-CU9-ubuntu | amd64 | No Dockerfile | Ubuntu 16.04 | 09/10/2018 23:26:19 | 09/10/2018 23:26:19 |
| 2017-CU8-ubuntu | amd64 | No Dockerfile | Ubuntu 16.04 | 09/10/2018 23:28:14 | 09/10/2018 23:28:14 |
| 2017-CU7-ubuntu | amd64 | No Dockerfile | Ubuntu 16.04 | 09/10/2018 23:30:43 | 09/10/2018 23:30:43 |
| 2017-CU6-ubuntu | amd64 | No Dockerfile | Ubuntu 16.04 | 09/10/2018 23:33:09 | 09/10/2018 23:33:09 |

**mssql-deployment.yml content:**

namespace: “employee”

deployment: “mssql-deployment”

replicas: “1”,

pod name: “mssql”,

container name: “mssql”,

image: “mcr.microsoft.com/mssql/server:2017-CU8-ubuntu”

service: “mssql-service”

```yaml
apiVersion: v1
kind: Namespace
metadata:
    name: employee
    labels:
      name: employee
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mssql-deployment
  namespace: employee
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mssql
  template:
    metadata:
      labels:
        app: mssql
    spec:
      securityContext:
        fsGroup: 10001
      containers:
      - name: mssql
        image: 'mcr.microsoft.com/mssql/server:2017-CU8-ubuntu'
        resources:
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mssql-service
  namespace: employee
spec:
  selector:
    app: mssql
  ports:
    - protocol: TCP
      port: 1433
      targetPort: 1433
  type: LoadBalancer
```

---

Once the MSSQL pod is running we need to create the database for the application. For that we will use the **Entity Framework** which is part of .NET core 3.1 and enables to create the database with default data.

**cd ..**

**cd Employees**

**dotnet ef database update** (creates **EmployeeDB** database in our MSSQL server with initial data.)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\employee\employeemanagement\manifests> cd ..
PS C:\employee\employeemanagement> cd .\Employees\
PS C:\employee\employeemanagement\Employees> dotnet ef database update
Build started...
Build succeeded.
Done.
PS C:\employee\employeemanagement\Employees>
```

The next step is building and running the application again:

**dotnet build** & **dotnet run**

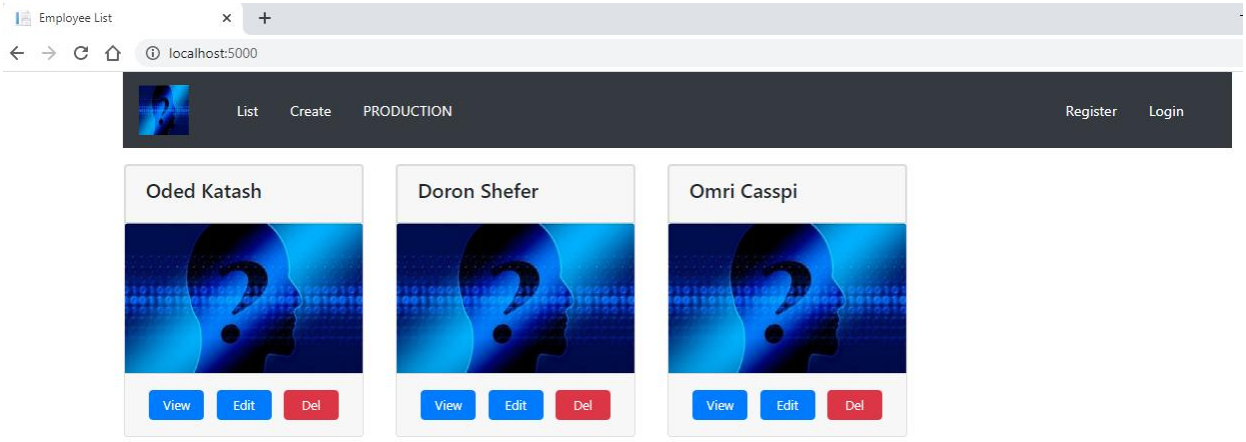
```
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\employee> cd .\employeemanagement\
PS C:\employee\employeemanagement> cd .\Employees\
PS C:\employee\employeemanagement\Employees> dotnet build
Microsoft (R) Build Engine version 16.7.2+b60ddb6f4 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  Restored C:\employee\employeemanagement\Employees\Employees.csproj (in 1.56 sec).
  Employees -> C:\employee\employeemanagement\Employees\bin\Debug\netcoreapp3.1\Employees.dll
  Employees -> C:\employee\employeemanagement\Employees\bin\Debug\netcoreapp3.1\Employees.Views.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:30.76
PS C:\employee\employeemanagement\Employees> dotnet run
```

Open Google Chrome or any Other browser and enter the URL: http://localhost:5000




The application is working now against MSSQL in Kuberneties cluster.

Use **CTRL+C** in the terminal to exit the application

Done!

## Advanced Commands for MSSQL pod

Connect to MSSQL pod and check:

**kubectl -n employee exec -it mssql-deployment-6bcb97764c-m675k -- /bin/sh**

MSSQL edition (version):

**/opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT @@SERVERNAME,@@VERSION" -P "MyDemoPwd2021!" -W**

```
# /opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT @@SERVERNAME,@@VERSION" -P "MyDemoPwd2021!" -W

- -
mssql-deploymen Microsoft SQL Server 2017 (RTM-CU8) (KB4338363) - 14.0.3029.16 (X64)
        Jun 13 2018 13:35:56
        Copyright (C) 2017 Microsoft Corporation
        Developer Edition (64-bit) on Linux (Ubuntu 16.04.4 LTS)

(1 rows affected)
# exit
```

Databases:

**/opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT Physical_Name FROM sys.master_files" -P "MyDemoPwd2021!" -W**

```
# /opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT Physical_Name FROM sys.master_files" -P "MyDemoPwd2021!" -W
Physical_Name
-------------
/var/opt/mssql/data/master.mdf
/var/opt/mssql/data/mastlog.ldf
/var/opt/mssql/data/tempdb.mdf
/var/opt/mssql/data/templog.ldf
/var/opt/mssql/data/model.mdf
/var/opt/mssql/data/modellog.ldf
/var/opt/mssql/data/MSDBData.mdf
/var/opt/mssql/data/MSDBLog.ldf
/var/opt/mssql/data/EmployeeDB.mdf
/var/opt/mssql/data/EmployeeDB_log.ldf

(10 rows affected)
#
```



List of tables in EmployeeDB database:

**/opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT * FROM EmployeeDB.INFORMATION_SCHEMA.TABLES" -P "MyDemoPwd2021!" -W**

```
# /opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT  *  FROM EmployeeDB.INFORMATION_SCHEMA.TABLES" -P "MyDemoPwd2021!" -W
TABLE_CATALOG TABLE_SCHEMA TABLE_NAME TABLE_TYPE
------------- ------------ ---------- ----------
EmployeeDB dbo __EFMigrationsHistory BASE TABLE
EmployeeDB dbo Employees BASE TABLE
EmployeeDB dbo AspNetRoles BASE TABLE
EmployeeDB dbo AspNetUsers BASE TABLE
EmployeeDB dbo AspNetRoleClaims BASE TABLE
EmployeeDB dbo AspNetUserClaims BASE TABLE
EmployeeDB dbo AspNetUserLogins BASE TABLE
EmployeeDB dbo AspNetUserRoles BASE TABLE
EmployeeDB dbo AspNetUserTokens BASE TABLE

(9 rows affected)
#
```

Show list of employees in Employees table:

**/opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT * FROM EmployeeDB.dbo.Employees" -P "MyDemoPwd2021!" -W**

```
# /opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT  *  FROM EmployeeDB.dbo.Employees" -P "MyDemoPwd2021!" -W
Id Name Email PhotoPath Role
-- ---- ----- --------- ----
1 Oded Katash oded@gmail.com NULL 0
2 Doron Shefer doron@gmail.com NULL 1
3 Omri Casspi omri@gmail.com NULL 2

(3 rows affected)
#
```

Show the columns in Employees table:

**/opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT * FROM EmployeeDB.dbo.Employees WHERE 1=0" -P "MyDemoPwd2021!" -W**

```
# /opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "SELECT * FROM EmployeeDB.dbo.Employees WHERE 1=0" -P "MyDemoPwd2021!" -W
Id Name Email PhotoPath Role
-- ---- ----- --------- ----

(0 rows affected)
#
```

Update employee email:

**/opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "UPDATE EmployeeDB.dbo.Employees SET Email = 'odedkat@nba.com' WHERE Id=1" -P "MyDemoPwd2021!" -W**

```
# /opt/mssql-tools/bin/sqlcmd -S localhost,1433 -U sa -Q "UPDATE EmployeeDB.dbo.Employees SET Email = 'odedkat@nba.com' WHERE Id=1" -P "MyDemoPwd202
1!" -W

(1 rows affected)
#
```