Windocks: a Modern, Open, Data Delivery Platform

SQL Server Container Configuration with Extensible Key Management (SQL Server EKM)

As an enterprise data platform SQL Server supports use of a wide array of services and resources on both the local host and the domain, with a diverse ecosystem of cloud and third party solutions. Solutions like Private Key and External Key Managers, Data Integration, and High Availability and Disaster Recovery involve SQL scripts with varied credentials and proxy services.

Windocks 2.25 introduces new options to simplify integration with external resources, with control over the ordering of scripts during a container build, and secure use of network and host based credentials. Two examples help illustrate the use of these options:

SQL Server TDE: a new container requires a script to regenerate the Master database encryption certificate prior to a TDE encrypted databases being mounted.

Extensible Key Manager (EKM) support commonly requires scripts run with specific credentials to integrate containers with EKM infrastructure.

Control the order of script operations during container build

File extensions determine the order of script operations during container build. Scripts with the file extension **.sqlsys** will be run prior to databases being mounted, and **.sql** file extensions will be run after databases are mounted. The following dockerfile uses a cloned TDE encrypted database, and a SQL Server script is regenerates the Master database encryption certificate prior to the encrypted database being mounted. The **.sqlsys** file extension ensures that the script will be run prior to the databases being mounted.

```
FROM cloneimage
COPY tdescript.sqlsys .
RUN tdescript.sqlsys
```

Secure use of account credentials

SQL Server credentials are used to allow SQL Server accounts to use host and domain based resources, see: https://docs.microsoft.com/en-us/sql/relational-databases/security/authentication-access/credentials-database-engine. Windocks provides automated, secure use of account credentials for container builds, using a combination of environment variables and encrypted secrets.

As with other SQL Server scripts, this process uses file extensions to control the order of script operations. Scripts with the **.sqlsysrunas** file extension are run prior to databases being mounted, and scripts with the **.sqlrunas** extension are run following databases being mounted.

The steps involved in secure use of a credential includes the following:

1) Stop the Windocks service, and start the default SQL Server instance used for container operations.
2) Add the user credentials as a SQL Server login.
3) Add the new SQL Server user to the **SysAdmin Group** (the user needs permissions to the databases).

```
EXEC sp_addsrvrolemember 'newuser', 'sysadmin';
```

4) Return the SQL Server instance to "manual, off" state.
5) Using Explorer, navigate to **\Windocks\bin** directory, open a **cmd** prompt and enter **encrypt**
6) Enter the <**newuser**> password at the prompt and return
7) The encrypted password is recorded in the **encrypted.txt** file in the same directory. Open the file using notepad, and copy the complete password into **\windocks\config\node.conf** as illustrated. Note, the syntax for this encrypted password must be exactly as shown to support container cleanup and other operations that refer to these password.

   SQLRUNAS_PASSWORD1="paste encrypted password here"

8) Save the updated **node.conf** file
9) Restart the Windocks service

Test the use of secure user credentials

Navigate to the **\windocks\samples\testadddbwithsqlrunas** folder. Open and edit the dockerfile to reflect the environment variable implemented earlier, and the appropriate username.

```
FROM mssql-20XX
ADDDB customers customerdata.mdf
COPY cleanseData.sqlrunas .
RUN cleanseData.sqlrunas 'newuser' SQLRUNAS_PASSWORD1
```

Build a container using the Dockerfile and start the container.

```
>docker build c:\windocks\samples\testadddbwithsqlrunas
>docker start <containerid>
```

Use SQL Server Management Studio to open the container (127.0.0.1,1000X with a comma separator), and inspect the **customers** database. Select the **customers** database tables, and expand the columns, to confirm the "last name" column was deleted.

This process can be used in a wide variety of use-cases that involve host or domain level user accounts used to integrate SQL Server containers with external resources and services.

Windocks Support

For questions or assistance in working with Windocks containers, contact support@windocks.com

For further details on working with SQL Server TDE, refer to **SQL Server Containers and TDE**.