Windocks: a Modern, Open, Data Delivery Platform

SQL Server Containers and Transparent Data Encryption (TDE) Databases

Using SQL Server TDE with Windocks containers is straightforward and uses the same SQL Server scripts normally used with TDE. The process begins by implementing the encryption key in the Master database of the Windocks default instance used for container support.

One article that is helpful is: https://www.sqlshack.com/how-to-configure-transparent-data-encryption-tde-in-sql-server/

Implement an encryption key, and then generate an encryption certificate as shown:

```
USE Master;
GO
CREATE MASTER KEY ENCRYPTION
BY PASSWORD='InsertStrongPasswordHere'
GO
```

```
CREATE CERTIFICATE TDE_Cert
WITH
SUBJECT='Database_Encryption';
GO
```

Backup and Restoring the TDE encryption key:

Remember to back up the certificate to a known location, and a separate server is recommended rather than the local server as shown here.

```
BACKUP CERTIFICATE TDE_Cert
TO FILE = 'C:\temp\TDE_Cert'
WITH PRIVATE KEY (file='C:\temp\TDE_CertKey.pvk',
ENCRYPTION BY PASSWORD='InsertStrongPasswordHere';
GO
```

The encryption certificate is restored to a server using the following scripts.

```
USE Master;
GO
CREATE MASTER KEY ENCRYPTION
BY PASSWORD='InsertStrongPasswordHere'
GO
```

```
USE Master;
GO
CREATE CERTIFICATE TDE_Cert
FROM FILE = 'C:\Temp\TDE_Cert'
WITH PRIVATE KEY (FILE = 'C:\TDE_CertKey.pvk'
DECRYPTION BY PASSWORD = 'InsertStrongPasswordHere';
GO
```

Applying Encryption to a User-defined Database

Encryption is applied to a database, through the following two steps, referencing the existing encryption certificate "TDE_Cert". The encryption is enabled in the second script.

```
USE <DB>;
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE TDE_Cert;
GO
```

```
USE Master;
ALTER DATABASE <DB>
SET ENCRYPTION ON;
GO
```

Working with SQL Server Containers

Windocks containers are created by cloning an assigned SQL Server instance that is identified in the **\windocks\config\node.config** file. Each container inherits the attributes of the default instance Master database, and that includes the encryption key and certificate, along with user permissions, and other configuration settings.   When working with the assigned default instance, be sure the Windocks service is turned off, and return the default instance to an "off" state following updates to the instance.

Windocks containers inherit the Master database encryption certificate, which can be confirmed by viewing the container in SSMS.   The Windows Server recognizes the container as a new session, and using the inherited certificate generates a session related error (SQL Server error 15581: "please open the key in the session prior to this operation."   The certificate needs to be refreshed, which is accomplished with the following script. Following the refresh of the certificate, the container can be used to restore encrypted backups and other operations that use of the encryption certificate.

```
USE MASTER;
ALTER SERVICE MASTER KEY FORCE REGENERATE;
OPEN MASTER KEY DECRYPTION BY PASSWORD='StrongPassword';
ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY;
CLOSE MASTER KEY;
```

The workflow to use a TDE enabled container is illustrated below in two examples.   The first builds a database clone image using encrypted backup files.   Once the image is built a second dockerfile delivers a TDE enabled container with the encrypted cloned databases.   Both examples reference the above script as **tdesetup.sqlsys**.   These samples also assume the SQL Server host instance has the appropriate encryption certificates enabled in the Master database.

Please note that Windocks supports running scripts on the container prior to mounting databases, or after.   In the case of TDE the scripts need to be applied prior to attempting to work with encrypted databases, to refresh the encryption certificate.   This is accomplished by using scripts with the file extension **script.sqlsys.**   In other cases scripts need to be run after the databases are mounted and available, such as when configuring support for database mail.   In these cases use the normal **script.sql** extension.

This example illustrates the building of clone based image, using encrypted backups.

```
>docker build –t <imagename> \path\to\dockerfile

FROM mssql-2012
COPY tdesetup.sqlsys .
RUN tdesetup.sqlsys
SETUPCLONING FULL \\path\to\encryptedbackup
```

This illustrates the building of a new container, using the image built above. The **>docker build –t** is used to build clonable images, and **>docker build** is used to build a container. On delivery of the container, the container is started with **>docker start <containerid>.**

```
>docker build <imagename>

FROM <imagename>
COPY tdesetup.sqlsys .
RUN tdesetup.sqlsys
```

In summary, when working with SQL Server TDE each container requires a script run to refresh the Master database encryption key. Scripts included in Windocks dockerfiles are processed either before or after databases are mounted or added, using script.sqlsys or script.sql file extensions, respectively

Windocks Support

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