Steps to Enable TDE (Transparent Data Encryption) on Availability Group – SQL Server

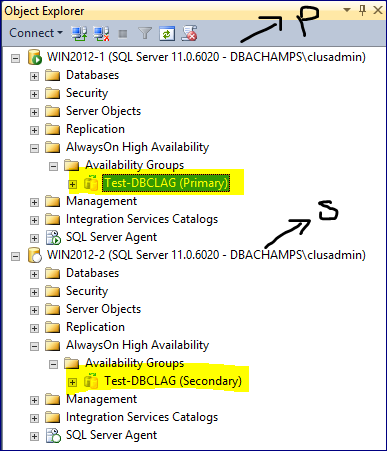
Transparent Data Encryption (TDE) encrypts SQL Database files by encrypting data at rest. In a situation where our physical media such as data, log and backup files get stolen, the malicious person can restore/attach the database and retrieve data.

TDE protects this by not letting the database restored/attached without the associated certificate and master key.

**Note: When enabling TDE, we need to make sure to backup the certificate and the key associated with the certificate. Without this certificate, we will never be able to restore/attach the database to a different server. The certificate should be available even if the TDE is disabled, part of the transaction log may remain still protected and the certificate may be required until a full database backup is taken.**

Enabling TDE isn’t as straightforward as it is for a database outside of an availability group. Databases that are in the availability group requires certain considerations and precautions to enable TDE which is explained step by step as follows:

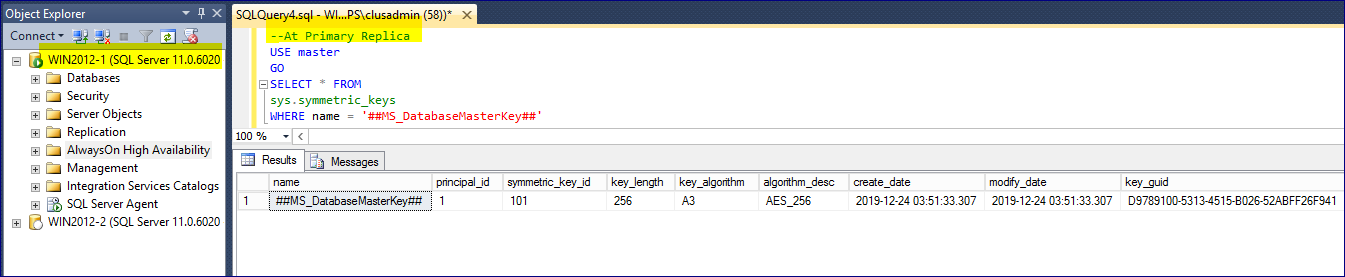
1**.Here we have a 2-node cluster and can see only 1 test database (DB Name: Test) on the PRIMAY REPLICA which is not part of the availability group yet. When enabling TDE on AlwaysOn, we need to make sure that our database has been removed from the AG or else the database on the secondary nodes will change to SUSPECT MODE.**



ON THE PRIMARY NODE – Verify that the primary node has a Database Master Key (DMK) in the master database.

USE master  
GO

SELECT \* FROM  
sys.symmetric\_keys  
WHERE name = ‘##MS\_DatabaseMasterKey##’



2. Use Create Master Key if it doesn’t exist (or) use Alter Master Key if it already exists and change the password if it is not known.

**Note: The Service Master Key is the root of SQL Server's**[**Encryption Hierarchy**](http://sqlity.net/en/2357/encryption-hierarchy/)**. There can only be 1 service master key per SQL Server instance. The service master key is used to protect (encrypt) other keys, mainly the database master keys. It cannot be used directly to encrypt data. And we can't create one our self.**

USE master;  
GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD = ‘P@$$word1234’;  
GO

**(0r) – Below is used to Alter the existing master key and provide the new password.**

use master

go

alter master key add encryption by password = 'P@$$word1234'

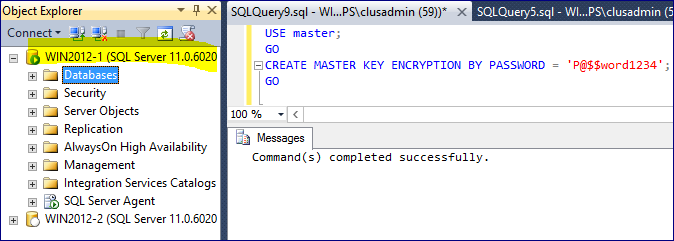
go

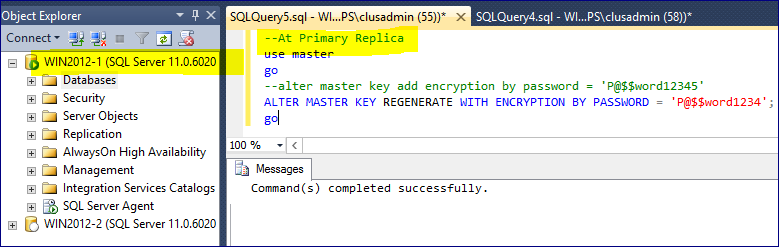
**(0r) – Below is used to Overwrite (Re-Generate) the existing master key and provide the new password.**

Use master

go

ALTER MASTER KEY REGENERATE WITH ENCRYPTION BY PASSWORD = 'P@$$word1234';



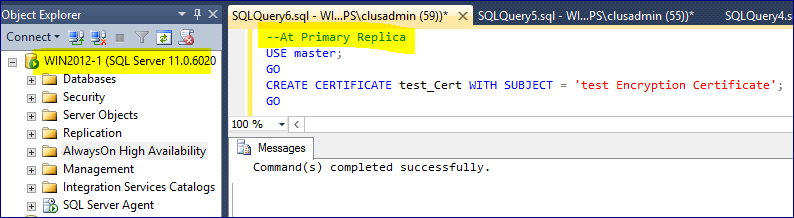


**Note: Make sure to use a complex password and store it in a password vault to avoid any risk of compromisation.**

3. Create the Certificate for the test database

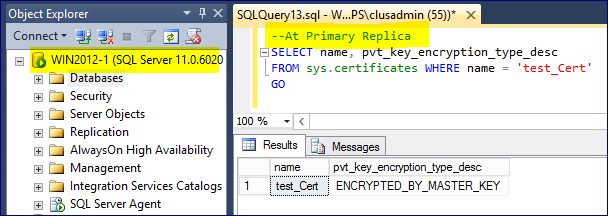
USE master;  
GO

CREATE CERTIFICATE test\_Cert WITH SUBJECT = ‘test Encryption Certificate’;  
GO



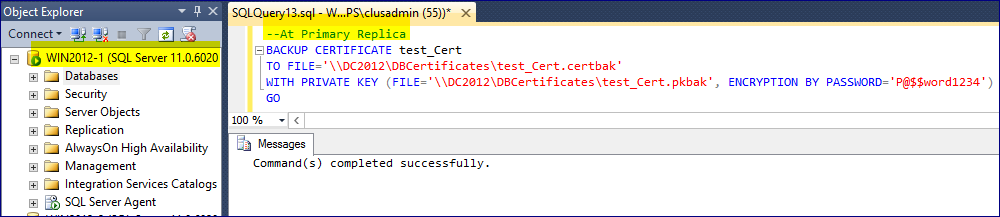
4. Run the following script to check if the certificate was created

SELECT name, pvt\_key\_encryption\_type\_desc FROM sys.certificates WHERE name = ‘test\_Cert’  
GO



5. Backup the certificate on a shared location where all other nodes has access and keep it in a secure place

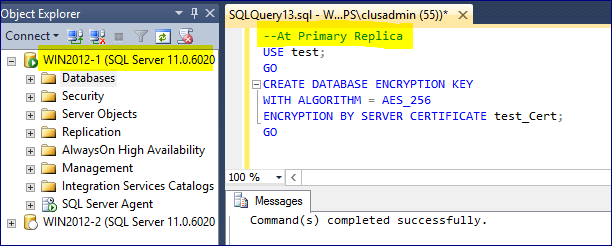
BACKUP CERTIFICATE test\_Cert  
TO FILE=’\\DC2012\DBCertificates\test\_Cert.certbak’  
WITH PRIVATE KEY (FILE=’\\DC2012\DBCertificates\test\_Cert.pkbak’, ENCRYPTION BY PASSWORD=’P@$$word1234′)  
GO



6. Create AES\_256 encryption using the certificate on the required database to be added in AG.

USE test;    
GO

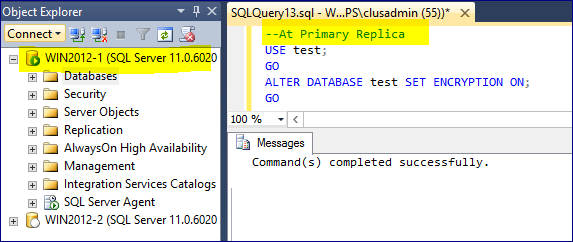
CREATE DATABASE ENCRYPTION KEY  
WITH ALGORITHM = AES\_256  
ENCRYPTION BY SERVER CERTIFICATE test\_Cert;  
GO



7. Enable the encryption on the required database to be added in AG.

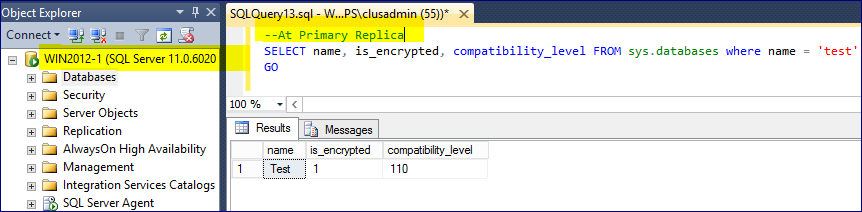
USE test;  
GO

ALTER DATABASE test SET ENCRYPTION ON;  
GO

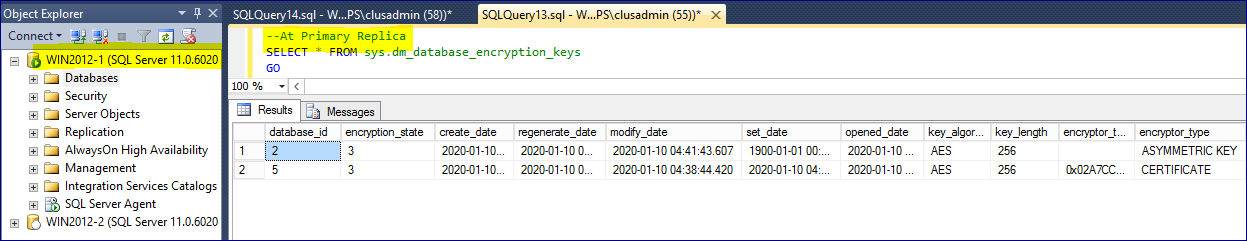


8. Check to verify the database is encrypted ?

SELECT name, is\_encrypted, compatibility\_level FROM sys.databases where name = ‘test’  
GO

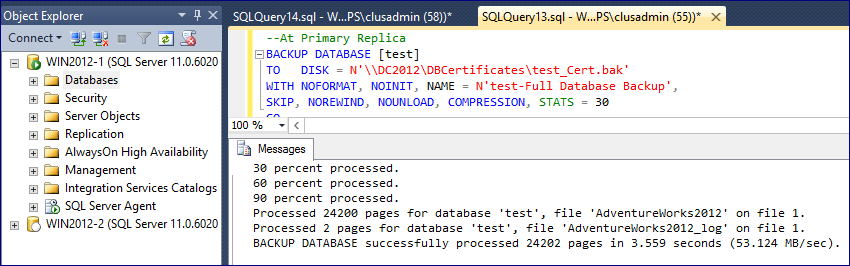


**How to find details about each database that is encrypted ?**  
SELECT \* FROM sys.dm\_database\_encryption\_keys  
GO



9. Take a full backup of the database. Note: Perform copy-only backup if it’s part of a backup maintenance plan so that we don’t break the existing backup chain if you need to revert at any time incase.

BACKUP DATABASE [test]  
TO   DISK = N’\\DC2012\DBCertificates\test\_Cert.bak’ WITH NOFORMAT, NOINIT, NAME = N’test-Full Database Backup’,  
SKIP, NOREWIND, NOUNLOAD, COMPRESSION, STATS = 30  
GO



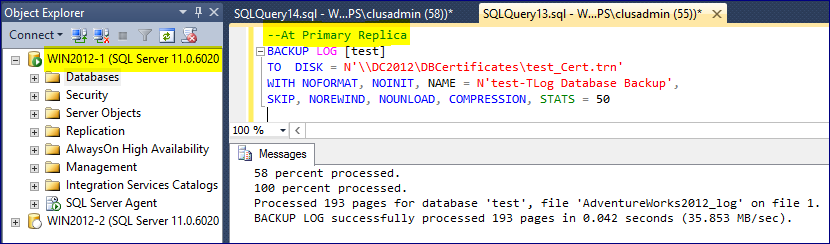
10. Take a log backup (To make it a part of an AG with TDE enabled a log backup is required)

--At Primary Replica

BACKUP LOG [test]

TO DISK = N'\\DC2012\DBCertificates\test\_Cert.trn'

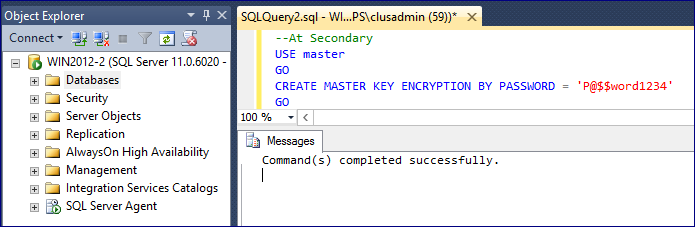
WITH NOFORMAT, NOINIT, NAME = N'test-TLog Database Backup', SKIP, NOREWIND, NOUNLOAD, COMPRESSION, STATS = 50



11. ON THE SECONDARY NODES – Create the same Database Master Key (DMK) in the master database that was created on Node 1 for Node 2.

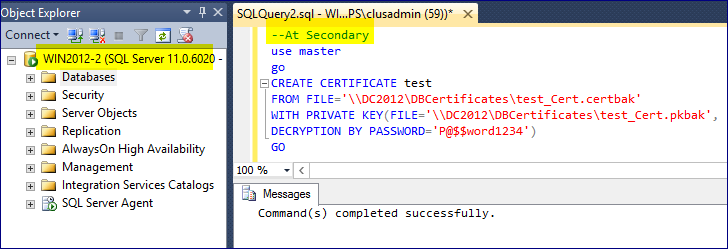
Run this script on all secondary nodes (at Node 2)  
USE master  
GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD = ‘P@$$word1234’  
GO



12. Transfer the certificate from the certificate backup on secondary node: Node 2

CREATE CERTIFICATE test  
FROM FILE=’\\DC2012\DBCertificates\test\_Cert.certbak’  
WITH PRIVATE KEY(FILE=’\\DC2012\DBCertificates\test\_Cert.pkbak’,  
DECRYPTION BY PASSWORD=’P@$$word1234′)  
GO



13. Restore the full backup followed by the log backup of the encrypted test database with No-Recovery mode. We want the database to be in restoring mode so we can join it later to the Availability Group via script.

**Note**: On secondary node, we are using the backups that we took earlier on the primary node.

--At Secondary

USE [master]

GO

RESTORE DATABASE [test]

FROM DISK = '\\DC2012\DBCertificates\test\_Cert.bak' WITH FILE = 1, NORECOVERY, NOUNLOAD, STATS = 1

GO

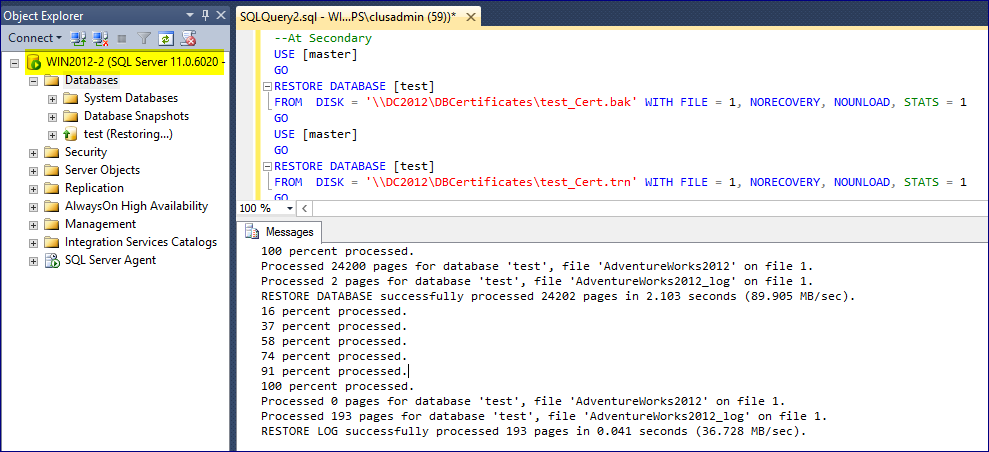
USE [master]

GO

RESTORE DATABASE [test]

FROM DISK = '\\DC2012\DBCertificates\test\_Cert.trn' WITH FILE = 1, NORECOVERY, NOUNLOAD, STATS = 1

GO

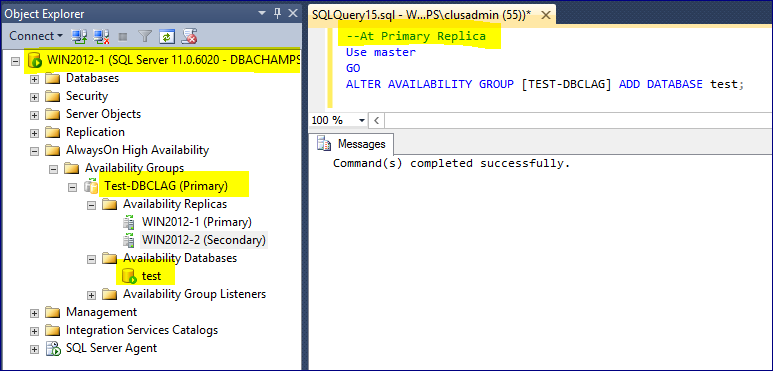


**BACK ON THE PRIMARY NODE**

14. Go back to the Primary node and add the database to the availability group. Refresh all of the nodes and we will see that the test database has been successfully added to the AG.

Use master  
GO

ALTER AVAILABILITY GROUP [TEST-DBCLAG] ADD DATABASE test;



Reference: <https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption?view=sql-server-2017>

<https://www.virtual-dba.com/how-to-enable-tde-availability-group/>

References:

<https://www.mssqltips.com/sqlservertip/3146/configuring-transparent-data-encryption-with-sql-server-2012-alwayson-availability-groups/>

<https://jbswiki.com/2019/06/27/adding-transparent-data-encryption-tde-enabled-user-database-to-always-on-availability-group/>

<https://www.c-sharpcorner.com/article/implementing-transparent-data-encryption-in-always-on/>

<https://www.risual.com/2019/01/24/setting-up-sql-server-tde-in-an-alwayson-environment/>

<https://sqlsailor.com/2012/01/03/drop-master-key-understanding-encryption-hierarchy/>

<http://kushagrarakesh.blogspot.com/2015/09/there-is-already-master-key-in-database.html>

<https://mattsql.wordpress.com/2012/11/13/migrating-sql-server-databases-that-use-database-master-keys/>

<https://joeschoice.com/encryption-in-ms-sql-part-2/>

<https://dba.stackexchange.com/questions/233040/how-to-change-sql-server-master-key-password>

<https://learnsqlserverhadr.com/how-to-enable-transparent-data-encryption-in-an-existing-sql-server-availability-group/>

<https://www.sqlservercentral.com/articles/enabling-tde-on-databases-in-an-alwayson-scenario>

<https://www.sqlshack.com/how-to-add-a-tde-encrypted-user-database-to-an-always-on-availability-group/>

<https://sqlstudies.com/2015/08/11/database-master-keys-can-have-more-than-one-password/>