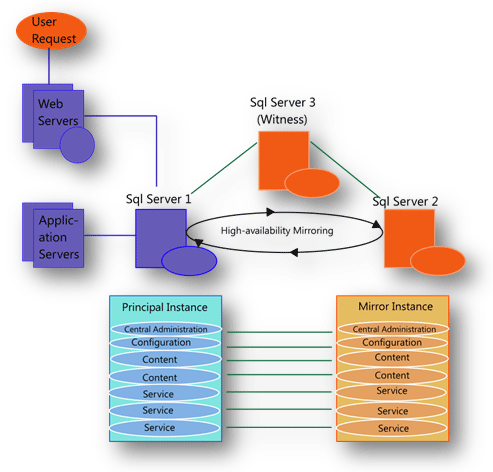
**SQL Server Database Mirroring**

To move the database transactions from one SQL Server database (Principal database) to another SQL Server database (Mirror database) on a different instance the database mirroring is used. If the principal server fails, the mirror server automatically becomes the new principal server and recovers the principal database using a witness server under high-availability mode. It is a mixture of replication and log shipping. Mirroring works only with the full recovery model.



**Difference between the Database Mirroring and Log Shipping**

|  |  |
| --- | --- |
| MIRRORING | LOG SHIPPING |
| 1. It is limited to only two Servers | 1.In this we can log ship to multiple Servers |
| 2. Mirroring with a Witness Server allows for High Availability and automatic fail over. | 2. Log shipping is only as current as how often the job runs. If we ship logs every 15 minutes, the secondary server could be as far as 15 minutes. Making it more of a Warm Standby. |
| 3. We can configure our DSN string to have both mirrored servers in it so that when they switch we notice nothing. | 3. We can leave the database in read only mode while it is being updated. Good for reporting servers. |
| 4. While mirrored, our Mirrored Database cannot be accessed. It is in Synchronizing/Restoring mode. | 4. Good for disaster recovery |
| 5. Mirroring with SQL Server 2019 standard edition is not good for load balancing |  |

**Things that are explained in this article**

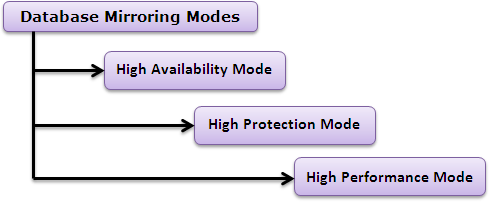
1. Roles Of the Server
2. Modes of Database Mirroring
3. Loss of Servers
4. Prerequisites
5. Restrictions
6. Endpoints
7. Creating a mirror database for mirroring
8. Database Mirroring

## Roles of the Server in Database Mirroring

* **Principal server**The principal server hosts the active copy of the database (referred to as the principal database) and services client requests. The principal server forwards all transactions to the mirror server before it applies them in the principal database.
* **Mirror server**  
    
  The mirror server hosts a copy of the principal database (referred to as the mirror database) and applies the transactions forwarded by the principal database to keep the mirror database synchronized with the principal database.
* **Witness server**  
    
  The witness server is an optional component of a database mirroring solution. When present, a witness server monitors the principal and mirror servers to ensure continued connectivity and participation in the mirror session (referred to as quorum). If either server loses quorum, the witness server assigns the principal server role, causing automatic failover from the principal server to the mirror server if necessary. A witness server is required for automatic failover; however, one witness server can support several mirror sessions because it is not an intensive job.

## Database Mirroring Modes

Database Mirroring has three different Modes



**High-Availability Mode**

In this Mode we need all the three servers, since the transaction safety level is set to FULL that results in the "Database transfer mechanism between the principal and mirror server is synchronous" which means that the principal server waits for an acknowledgement from the mirror server that the transaction log record has been recorded on the mirror server. Then, the client application gets confirmation that the transaction is committed. But if the principal server becomes unavailable then the witness server and the mirror server will form a quorum and perform automatic failover.

**High-Protection Mode**

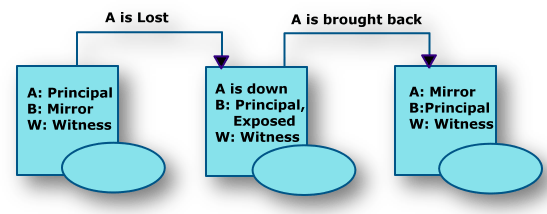
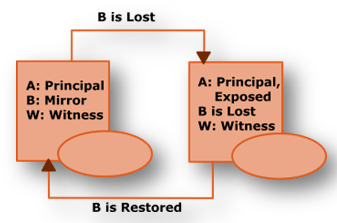
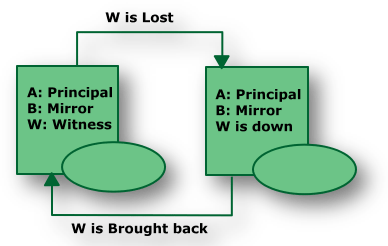
This mode is nearly the same as High-Availability mode, but the difference is that there is a need for two servers only (Principal server and Mirror server). The transaction safety level in this mode is also set to FULL that results in the same High- Availability Mode "Database transfer mechanism between the principal and mirror server is synchronous". Another difference is if the principal server becomes unavailable in this mode then we need to manually perform the failover because there is no witness server in this mode. Because the transaction safety level is set to FULL, we do not lose any committed transactions in the event of a failover.

**High-Performance Mode**

The same as High-Protection Mode, there is also a need of two servers (principal and mirror server). In this Mode the transaction safety level is set to OFF that results in "Data transfer mechanism between the principal and mirror servers is asynchronous", which means that the principal server does not wait for an acknowledgement from the mirror server that all transaction log records have been recorded on the mirror server and the client application gets confirmation that a transaction is committed as soon as the principal server has written the transaction to the log. If the Principal server in this mode becomes unavailable then we must manually perform the failover since there is no witness server in this mode. Because the transaction safety level is set to OFF, we might lose some transactions in the event of a failover.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operating Mode | Transaction safety | Transfer mechanism | Quorum required | Witness server | Failover Type |
| High Availability | FULL | Synchronous | Y | Y | Automatic or Manual |
| High Protection | FULL | Synchronous | Y | Y | Manual only |
| High Performance | OFF | Asynchronous | N | N/A | Forced only |

**Loss of Servers**

1. **PRINCIPAL Server Lost**The following scenario considers what happens when the principal server is lost in a High Availability scenario:  
     
   
2. **MIRROR Server Lost**  
     
   If the mirror server is lost first, the principal server is considered exposed because it cannot send data to the mirror.  
     
   
3. **WITNESS Server Lost**  
   When the witness server fails, mirroring continues but no automatic failover is possible. Loss of just one more server will mean there is no quorum, and the principal database will no longer be able to serve the database.  
     
   

**Prerequisites**

1. The database must be using the FULL RECOVERY or BULK LOGGED recovery models.
2. On the secondary server, we must have already restored a full database backup with the NORECOVERY or STANDBY options, so that transaction logs may be applied.

**Restrictions**

1. FILESTREAM is not supported by database mirroring. As a FILESTREAM filegroup cannot be created on the principal server. So, we can say that database mirroring cannot be configured for a database that contains FILESTREAM filegroups.
2. Database mirroring can support a maximum of about 10 databases per server instance, on a 32-bit system.
3. Database mirroring is not supported with either cross-database transactions or distributed transactions.

**EndPoints**

1. Endpoints must be correctly configured: Make sure that each server instance (the principal server, mirror server, and witness, if any) has a database mirroring endpoint.
2. Check that the port numbers are correct. To identify the port currently associated with database mirroring endpoint of a server instance, use the following Transact-SQL statement.
   1. SELECT type\_desc, port FROM sys.tcp\_endpoints;
   2. GO

**Note:** Inspect each server instance to determine whether it is listening on the correct ports.

1. Make sure that the endpoints are started (STATE=STARTED). On each server instance, use the following Transact-SQL statement.
   1. SELECT state\_desc FROM sys.database\_mirroring\_endpoints
2. To start an endpoint, use the following Transact-SQL statement.
   1. ALTER ENDPOINT Endpoint\_Mirroring
   2. STATE = STARTED
   3. AS TCP (LISTENER\_PORT = <port\_number>)
   4. FOR database\_mirroring (ROLE = ALL);
   5. GO
3. Check that the ROLE is correct. On each server instance use the following Transact-SQL statement.
   1. SELECT role FROM sys.database\_mirroring\_endpoints;
   2. GO
4. Make sure that the login from the other server has CONNECT permission. To determine who has CONNECT permission for an endpoint, on each server instance use the following Transact-SQL statement.
   1. SELECT 'Metadata Check';
   2. SELECT EP.name, SP.STATE,
   3. CONVERT(nvarchar(38), suser\_name(SP.grantor\_principal\_id))
   4. AS GRANTOR,
   5. SP.TYPE AS PERMISSION,
   6. CONVERT(nvarchar(46),suser\_name(SP.grantee\_principal\_id))
   7. AS GRANTEE
   8. FROM sys.server\_permissions SP , sys.endpoints EP
   9. WHERE SP.major\_id = EP.endpoint\_id
   10. ORDER BY Permission,grantor, grantee;
   11. GO
5. Endpoints must be correctly configured: Make sure that each server instance (the principal server, mirror server, and witness, if any) has a database mirroring endpoint.
6. Check that the port numbers are correct. To identify the port currently associated with database mirroring endpoint of a server instance, use the following Transact-SQL statement.
   1. SELECT type\_desc, port FROM sys.tcp\_endpoints;
   2. GO

Note: Inspect each server instance to determine whether it is listening on the correct ports.

1. Make sure that the endpoints are started (STATE=STARTED). On each server instance, use the following Transact-SQL statement.
   1. SELECT state\_desc FROM sys.database\_mirroring\_endpoints
2. To start an endpoint, use the following Transact-SQL statement.
   1. ALTER ENDPOINT Endpoint\_Mirroring
   2. STATE = STARTED
   3. AS TCP (LISTENER\_PORT = <port\_number>)
   4. FOR database\_mirroring (ROLE = ALL);
   5. GO
3. Check that the ROLE is correct. On each server instance use the following Transact-SQL statement.
   1. SELECT role FROM sys.database\_mirroring\_endpoints;
   2. GO
4. Make sure that the login from the other server has CONNECT permission. To determine who has CONNECT permission for an endpoint, on each server instance use the following Transact-SQL statement.
   1. SELECT 'Metadata Check';
   2. SELECT EP.name, SP.STATE,
   3. CONVERT(nvarchar(38), suser\_name(SP.grantor\_principal\_id)) |
   4. AS GRANTOR,
   5. SP.TYPE AS PERMISSION,
   6. CONVERT(nvarchar(46),suser\_name(SP.grantee\_principal\_id))
   7. AS GRANTEE
   8. FROM sys.server\_permissions SP , sys.endpoints EP
   9. WHERE SP.major\_id = EP.endpoint\_id
   10. ORDER BY Permission,grantor, grantee;
   11. GO

**Creating a Mirror Database For Mirroring**

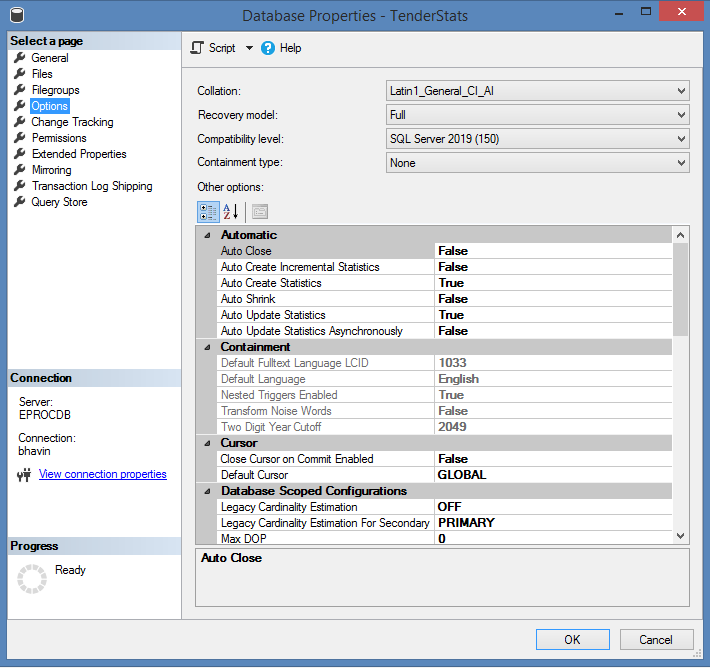
The mirror database must exist before a database mirroring session can begin.

**Steps to do this are as under**

* Make sure that Microsoft SQL Server 2019 is installed on both the principal and mirror server instances.
* Then Make it verify that the database uses the full recovery model.
* Back up the principal database to a full database backup.
* Confirm it that the system on which we are planning to create the mirror database has a disk drive with sufficient space to hold the database.
* Until the backup is on a network drive it is accessible from both systems, copy the database backup to that system.
* Restore the full database backup onto the mirror server instance to create the mirror database. RESTORE DATABASE (database name statement must specify) WITH NORECOVERY.  
    
  Note: If we restore the database filegroup by filegroup, be sure to restore the whole database.
* Typically, at least one log backup must be taken on the principal database, copied to the mirror server, and restored on the mirror database (using WITH NORECOVERY). So that, a log backup might be unnecessary, if the database has just been created and no log backup has been taken yet, or if the recovery model has just been changed from SIMPLE to FULL.

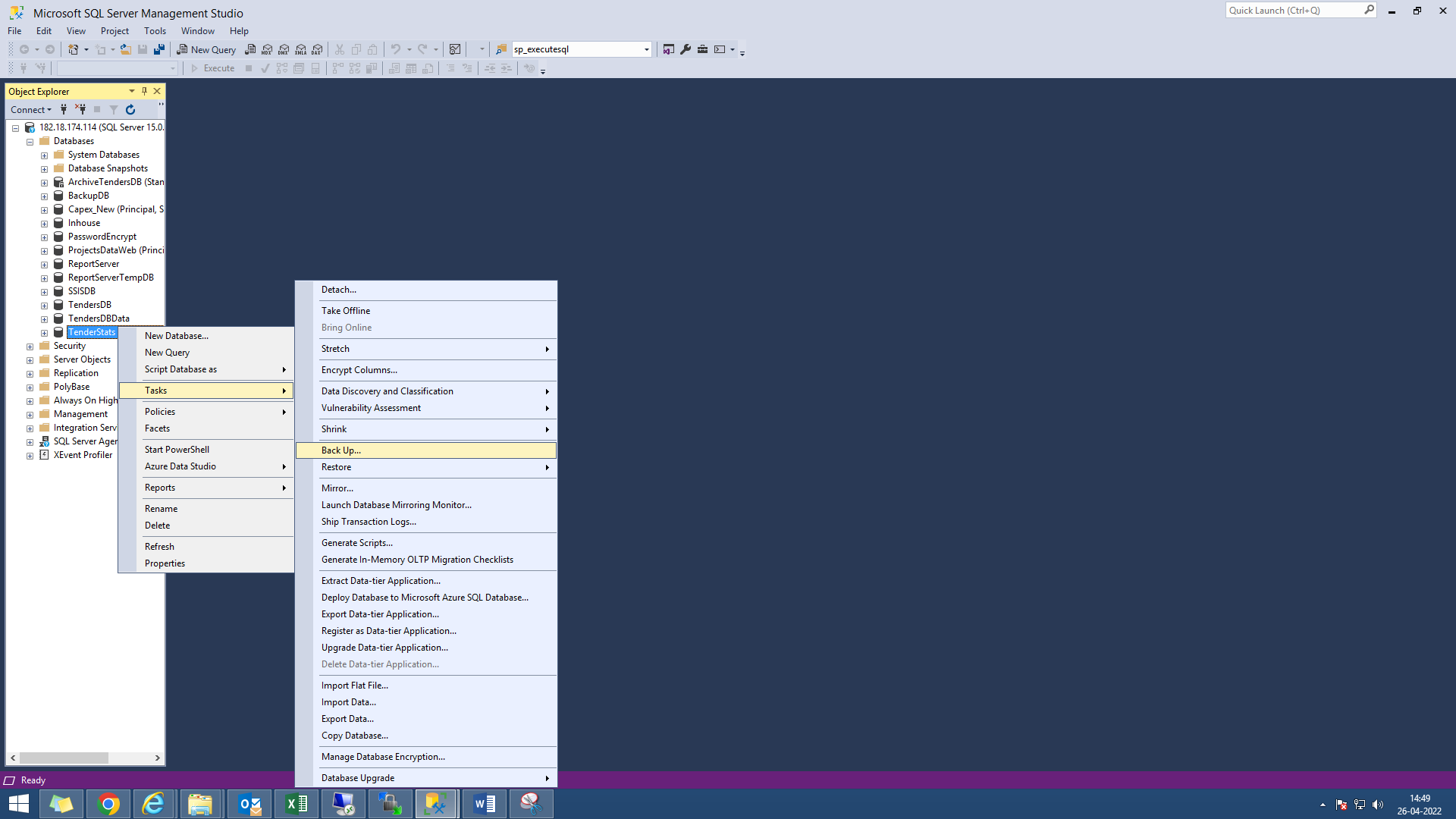
**Example**

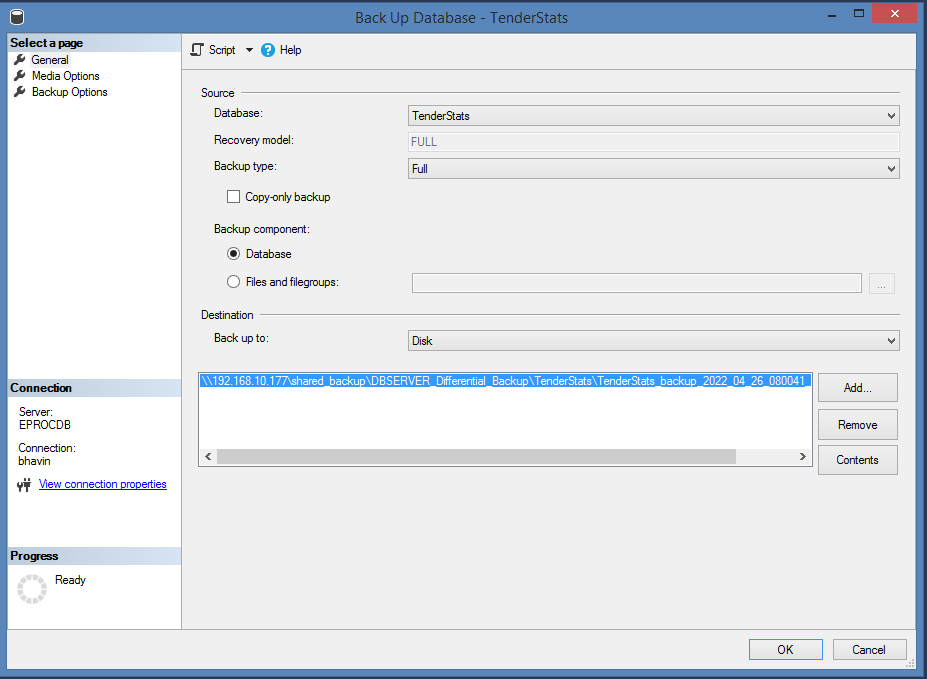
1. Let's take an example to apply database mirroring with one of the database named "tendersstats", Open SQL Server Management Studio 2019 and make sure that the database is in the Full Recovery mode. To check the recovery mode, right click the database name, select Properties and then select Options, or by using the following T-SQL:
   1. USE master;
   2. GO
   3. ALTER DATABASE tendersstats
   4. SET RECOVERY FULL;
   5. GO



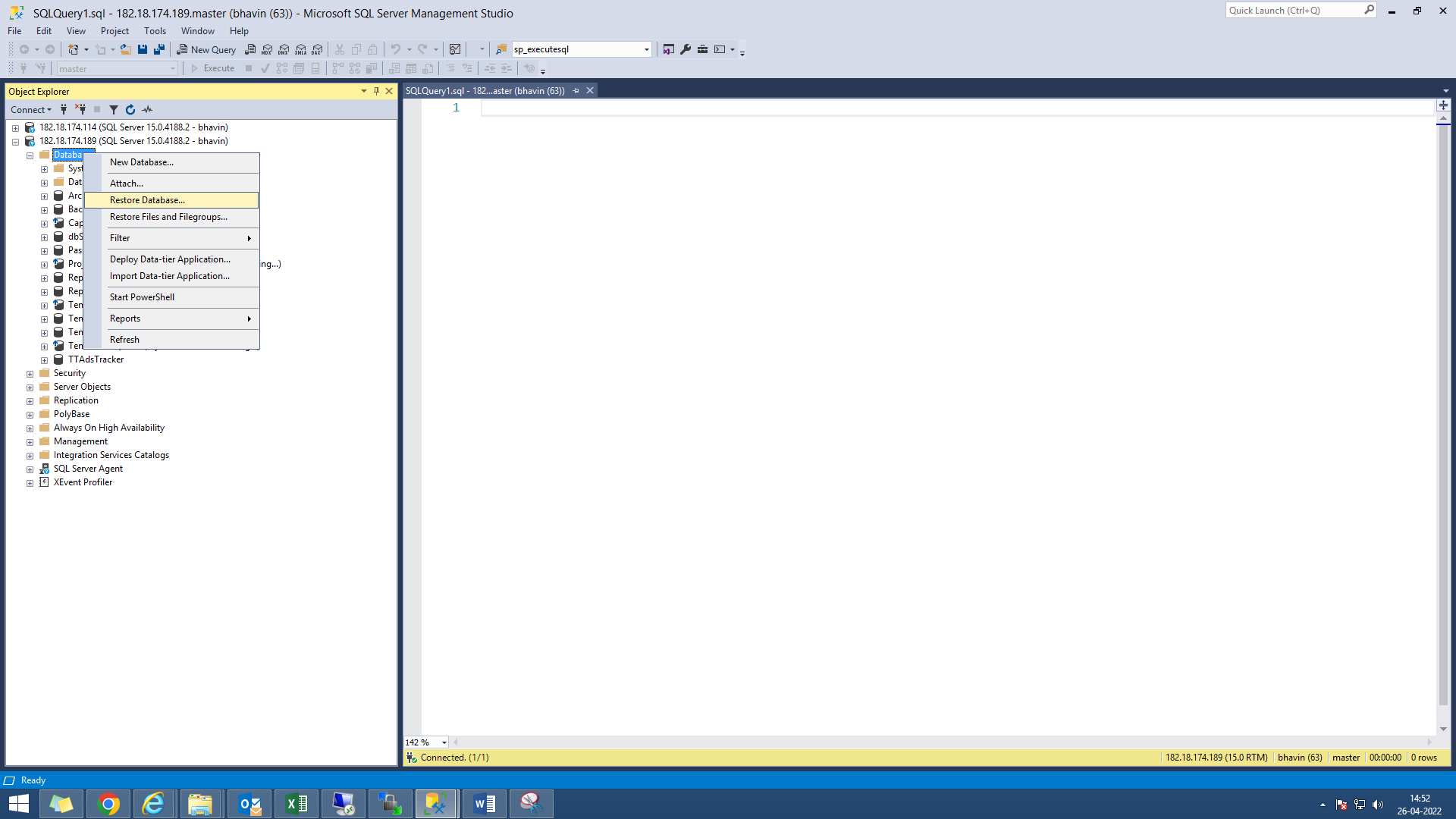
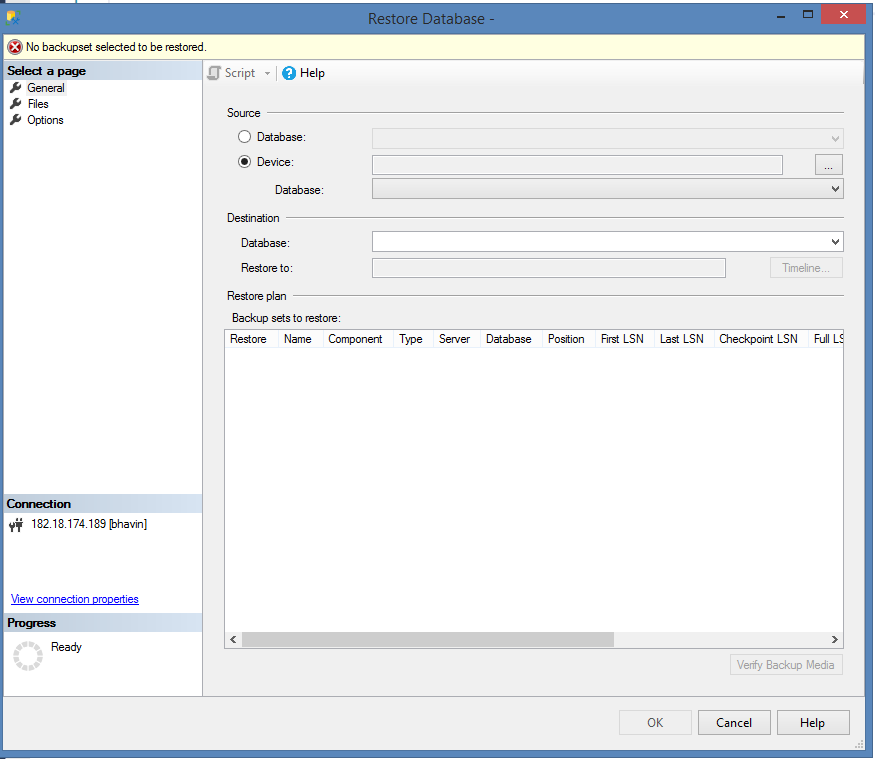
**Note:** If it is not in the FULL recovery Mode then, modify it to use the full recovery mode:

1. Once we modify the recovery model of our database to FULL, it's time to create a full Backup on the principal server instance, create a full backup of the principal database as follows:
   1. BACKUP DATABASE tendersstats
   2. TO DISK = 'D:\ tendersstats.bak'
   3. WITH FORMAT
   4. GO

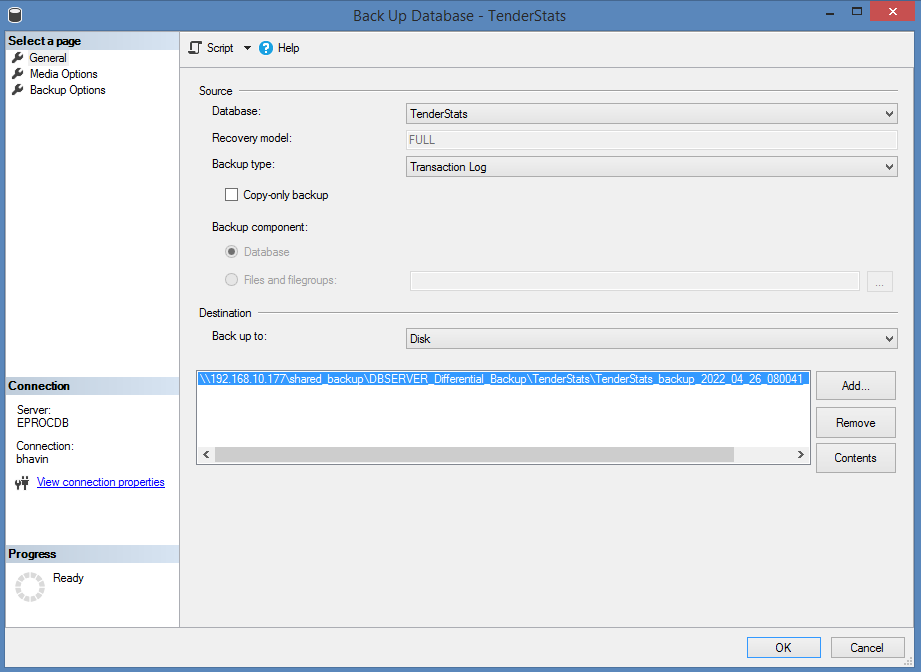




1. Now, copy the full backup and log to the mirror server.
2. Restore the full backup WITH NORECOVERY onto the mirror server instance. The restore command depends on whether the paths of principal and mirror databases are identical. Following T-SQL is used to do this:
   1. RESTORE DATABASE tendersstats
   2. FROM DISK = 'D:\ tendersstats.bak'
   3. WITH NORECOVERY
   4. GO

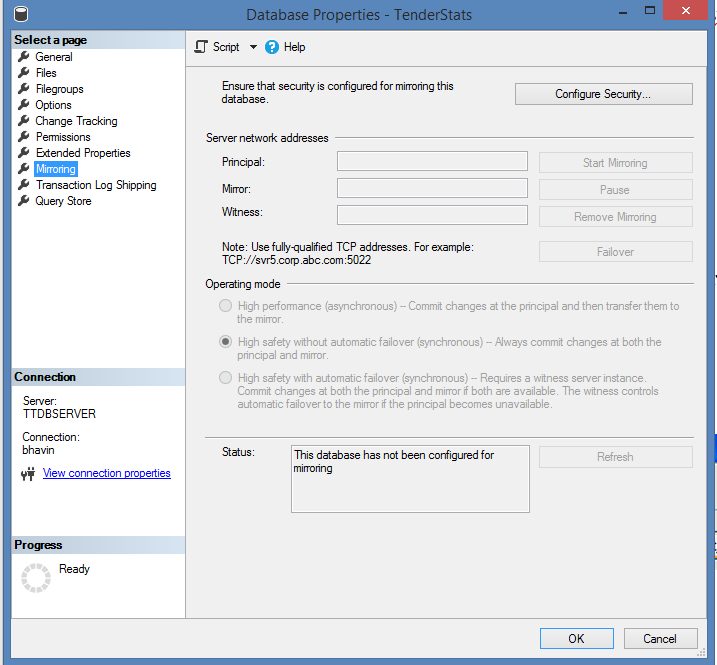
  
  
  
  
**Note:** If the path names of the principal and mirror databases differ, we cannot add a file, because on receiving the log for the add file operation, the mirror server instance attempts to place the new file in the location used by the principal database.

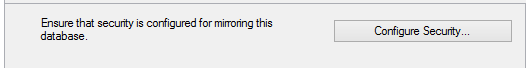
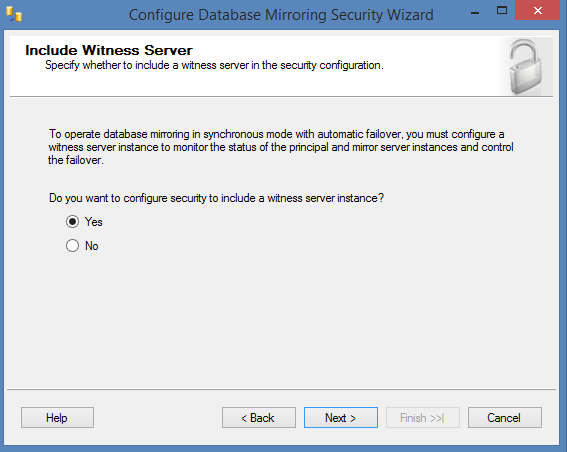
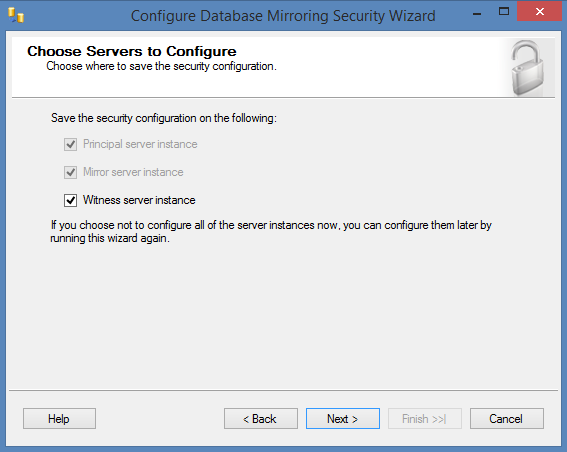
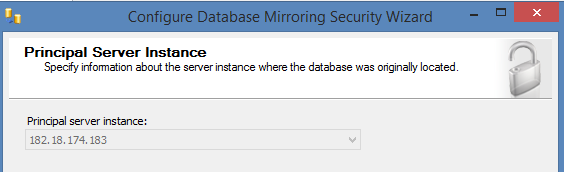
1. After creating a full backup, it is important to create a log backup on the principal database. The following Transact-SQL statement backs up the log to the same file used by the preceding full backup:
   1. BACKUP LOG tendersstats
   2. TO DISK = 'D:\ tendersstats.bak'
   3. GO

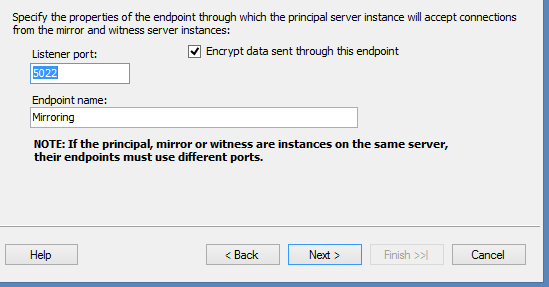
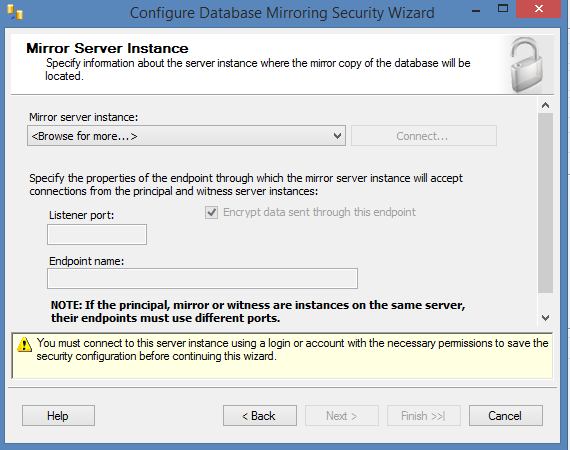
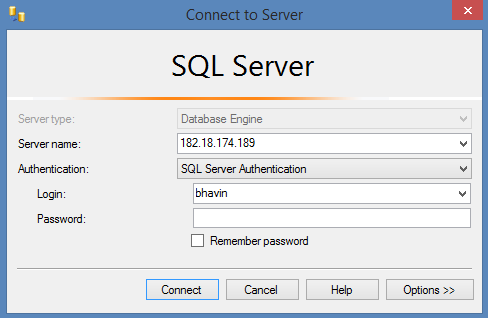
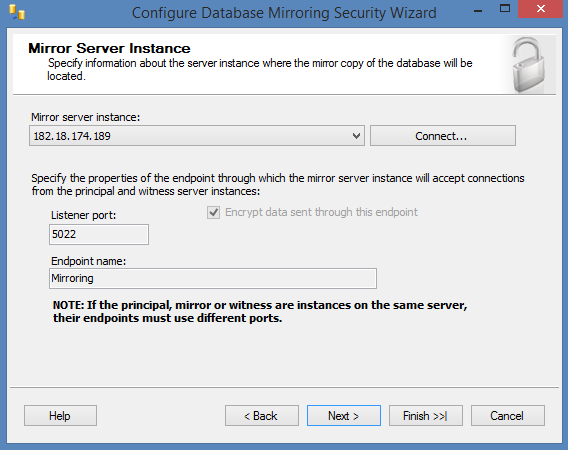
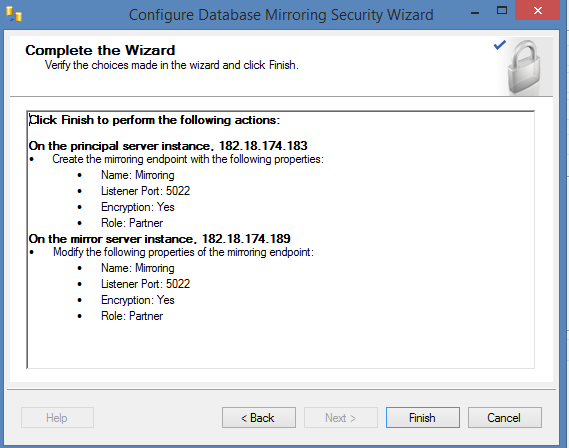
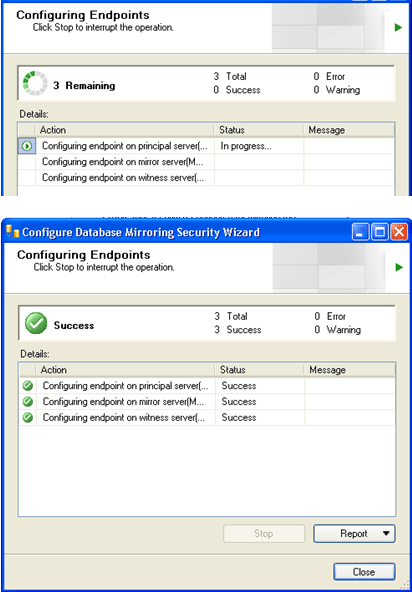
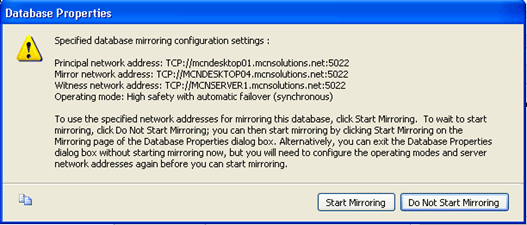
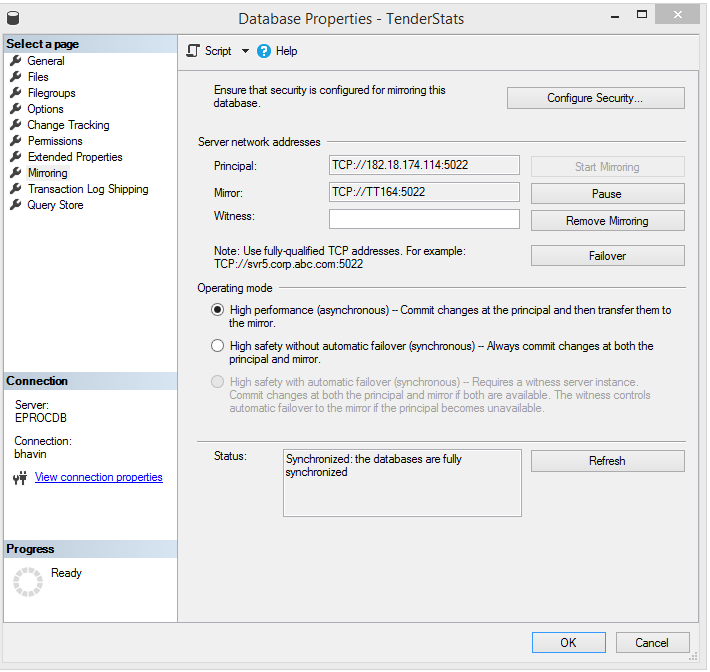
  
  
  
  
**Note:** Also the database name on the mirror must be the exact same name as the principal.

1. Before we can start mirroring, we must apply the required log backup (and any subsequent log backups). For example, the following Transact-SQL statement restores the first log from D:\ tendersstats.bak:
   1. RESTORE LOG tendersstats
   2. FROM DISK = 'D:\ tendersstats.bak'
   3. WITH FILE=1, NORECOVERY
   4. GO
2. If any additional log backups occur before starting mirroring, we also restore all of those log backups, in sequence, to the mirror server using WITH NORECOVERY. For example, the following Transact-SQL statement restores two additional logs from D:\ tendersstats.bak:
   1. RESTORE LOG tendersstats
   2. FROM DISK = 'D:\ tendersstats.bak'
   3. WITH FILE=2, NORECOVERY
   4. GO
   5. RESTORE LOG tendersstats
   6. FROM DISK = 'D:\ tendersstats2.bak'
   7. WITH FILE=3, NORECOVERY
   8. GO

Select the **principal database**. Right click on the database name and select **Properties** and then click on "**Mirroring**" the following window will appear



* Click on the "**Configure Security**. . ." button.  
    
  
* From the following window Click **Next** to get started.
* Here we have the option that if we want to use a witness server or not, select **yes** and click **Next**.  
    
  
* Again select if there is a need to use a witness server and Click **Next**.  
    
  
* Now First of all set up the **principal Server**.   
    
  

* Select the instance, Click on the checkbox if we want to encrypt the data and define the listener port, which Database Mirroring will use to communicate with the other instances in the mirror. Also select the endpoint name. Click **Next**.  
    
    
    
  **Note:** We can use the default number/Name or specify our own for both: Port, Endpoint.
* Secondly, set up the Mirror Server similarly as we have done for Principal Server: select the instance; select encrypt the data; define the listener port and select the endpoint name and connect to it. After connecting successfully Click **Next**.  
    
    
    
  **Note:** We can use the default number/Name or specify our own for both: Port, Endpoint.  
    
  
* Thirdly, it's time to set up the Witness Server as we have decided to use a witness previously. So for this also select the instance; click to encrypt the data; define the listener port and lastly select the endpoint name. Again Connect to it and Click **Next**.  
    
    
    
  **Note:** For witness server also we can use the default number/Name or specify our own for both: Port, Endpoint.
* Here we can set up special security credentials if we want to do (not Compulsory). Because all of the instances for this Demo are on the same server using the same accounts, they have been left blank. So directly Click **Next**.  
    
  
* Last but not the least Click on **"Finish".**  
    
  
* When we click on finish the following window configure the endpoints and shows its status means that it will show whether the process was successful setup or not.  
    
    
    
  **Note:** We can also see the reports if we want to see.
* Click on **close button** and we will the following window on our desktop. To begin mirroring select **"Start Mirroring"** and the mirroring process will begin. And if we want to make the changes than click on "Do Not Start Mirroring"  
    
  
* Finally our mirroring will starts as shown below. And we can see in Status box that the data has been **synchronized between the principal and the mirror.**  
    
  

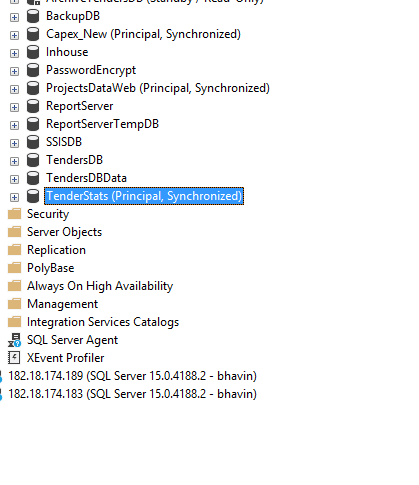
**Note:** We have some more options on this window which are as under:

**Pause:** This will stop transactions from being sent to our mirror.

**Remove Mirroring:** This will remove the mirroring configuration.

**Failover:** This will allow us to manually fail over to our mirrored copy. Once we fail over the current principal becomes the mirror and the current mirror becomes the principal.

Now we can see the status on our MSSQL window, the database is showing like this:



**SQL Server Database Mirroring Errors**

**Problem 1**

Error msg 1416, Level 16, State 31, Line 3. Database is not configured for database mirroring:

**Solution 1**

Restore the Full backup from principal server using with NORECOVERY option and also one transactional log backup from principal server using with NoRecovery option and then start configuring mirroring.

**Problem 2**

Error msg 1418, Level 16, State 1, Line 1. The server network address cannot be reached or does not exist. Check the network address name and reissue the command.

**Solution 2**

1. Restore a fairly recent full backup to the planned mirror server, with NORECOVERY
2. Then restore a transaction log, also with NORECOVERY
3. Configure endpoints correctly.
4. Check if the ports that selected are valid or not.
5. Is there a firewall (hardware or software)?

**Problem 3**

Error msg. (Microsoft SQL Server, Error 1475): "Database Mirroring cannot be enabled because the database may have bulk logged changes that have not been backed up. The last log backup on the principal must be restored on the mirror"

**Problem 4**

Error msg: (Microsoft SQL Server, Error 1478): The mirror database has insufficient transaction log data to preserve the log backup chain of the principal database. This may happen if a log backup from the principal database has not been restored on the mirror database.

**Problem 5**

Error msg: (Microsoft SQL Server, Error 1412): The remote copy of database has not been rolled forwarded to a point in time that is encompassed in the local copy of the transactional log.

**Solution 3, 4, 5**

We got the above mentioned error when we start the synchronization for database mirroring. To start the synchronization successfully follow these steps:

1. Principal Instance - Take a full backup and a log backup as well
2. Copy the full/log backups from Principal Instance to Mirror instance
3. Mirror Instance - Restore with NORECOVERY option the full backup
4. Mirror Instance - Apply the log backup
5. Principal Instance - Start synchronization