Implement a geo-distributed database

In this tutorial, you configure an Azure SQL database and application for failover to a remote region, and then test your failover plan. You learn how to:

* Create database users and grant them permissions
* Set up a database-level firewall rule
* Create a [geo-replication failover group](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-geo-replication-overview)
* Create and compile a Java application to query an Azure SQL database
* Perform a disaster recovery drill

If you don't have an Azure subscription, [create a free account](https://azure.microsoft.com/free/) before you begin.

Prerequisites

To complete this tutorial, make sure the following prerequisites are completed:

* Installed the latest [Azure PowerShell](https://docs.microsoft.com/powershell/azureps-cmdlets-docs).
* Installed an Azure SQL database. This tutorial uses the AdventureWorksLT sample database with a name of **mySampleDatabase** from one of these quick starts:
  + [Create DB - Portal](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-get-started-portal)
  + [Create DB - CLI](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-get-started-cli)
  + [Create DB - PowerShell](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-get-started-powershell)
* Have identified a method to execute SQL scripts against your database, you can use one of the following query tools:
  + The query editor in the [Azure portal](https://portal.azure.com/). For more information on using the query editor in the Azure portal, see [Connect and query using Query Editor](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-get-started-portal#query-the-sql-database).
  + The newest version of [SQL Server Management Studio](https://docs.microsoft.com/sql/ssms/download-sql-server-management-studio-ssms), which is an integrated environment for managing any SQL infrastructure, from SQL Server to SQL Database for Microsoft Windows.
  + The newest version of [Visual Studio Code](https://code.visualstudio.com/docs), which is a graphical code editor for Linux, macOS, and Windows that supports extensions, including the [mssql extension](https://aka.ms/mssql-marketplace) for querying Microsoft SQL Server, Azure SQL Database, and SQL Data Warehouse. For more information on using this tool with Azure SQL Database, see [Connect and query with VS Code](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-connect-query-vscode).

Create database users and grant permissions

Connect to your database and create user accounts using one of the following query tools:

* The Query editor in the Azure portal
* SQL Server Management Studio
* Visual Studio Code

These user accounts replicate automatically to your secondary server (and be kept in sync). To use SQL Server Management Studio or Visual Studio Code, you may need to configure a firewall rule if you are connecting from a client at an IP address for which you have not yet configured a firewall. For detailed steps, see [Create a server-level firewall rule](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-get-started-portal#create-a-server-level-firewall-rule).

* In a query window, execute the following query to create two user accounts in your database. This script grants **db\_owner** permissions to the **app\_admin** account and grants **SELECT** and **UPDATE** permissions to the **app\_user** account.

SQLCopy

CREATE USER app\_admin WITH PASSWORD = 'ChangeYourPassword1';

--Add SQL user to db\_owner role

ALTER ROLE db\_owner ADD MEMBER app\_admin;

--Create additional SQL user

CREATE USER app\_user WITH PASSWORD = 'ChangeYourPassword1';

--grant permission to SalesLT schema

GRANT SELECT, INSERT, DELETE, UPDATE ON SalesLT.Product TO app\_user;

Create database-level firewall

Create a [database-level firewall rule](https://docs.microsoft.com/sql/relational-databases/system-stored-procedures/sp-set-database-firewall-rule-azure-sql-database) for your SQL database. This database-level firewall rule replicates automatically to the secondary server that you create in this tutorial. For simplicity (in this tutorial), use the public IP address of the computer on which you are performing the steps in this tutorial. To determine the IP address used for the server-level firewall rule for your current computer, see [Create a server-level firewall](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-get-started-portal#create-a-server-level-firewall-rule).

* In your open query window, replace the previous query with the following query, replacing the IP addresses with the appropriate IP addresses for your environment.

SQLCopy

-- Create database-level firewall setting for your public IP address

EXECUTE sp\_set\_database\_firewall\_rule @name = N'myGeoReplicationFirewallRule',@start\_ip\_address = '0.0.0.0', @end\_ip\_address = '0.0.0.0';

Create an active geo-replication auto failover group

Using Azure PowerShell, create an [active geo-replication auto failover group](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-geo-replication-overview) between your existing Azure SQL server and the new empty Azure SQL server in an Azure region, and then add your sample database to the failover group.

Important

These cmdlets require Azure PowerShell 4.0. This sample requires the Azure PowerShell module version 5.1.1 or later. Run Get-Module -ListAvailable AzureRM to find the version. If you need to install or upgrade, see [Install Azure PowerShell module](https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps). Run Login-AzureRmAccount to create a connection with Azure.

1. Populate variables for your PowerShell scripts using the values for your existing server and sample database, and provide a globally unique value for failover group name.

PowerShellCopy

$adminlogin = "ServerAdmin"

$password = "ChangeYourAdminPassword1"

$myresourcegroupname = "<your resource group name>"

$mylocation = "<your resource group location>"

$myservername = "<your existing server name>"

$mydatabasename = "mySampleDatabase"

$mydrlocation = "<your disaster recovery location>"

$mydrservername = "<your disaster recovery server name>"

$myfailovergroupname = "<your unique failover group name>"

1. Create an empty backup server in your failover region.

PowerShellCopy

$mydrserver = New-AzureRmSqlServer -ResourceGroupName $myresourcegroupname `

-ServerName $mydrservername `

-Location $mydrlocation `

-SqlAdministratorCredentials $(New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $adminlogin, $(ConvertTo-SecureString -String $password -AsPlainText -Force))

$mydrserver

1. Create a failover group between the two servers.

PowerShellCopy

$myfailovergroup = New-AzureRMSqlDatabaseFailoverGroup `

–ResourceGroupName $myresourcegroupname `

-ServerName $myservername `

-PartnerServerName $mydrservername `

–FailoverGroupName $myfailovergroupname `

–FailoverPolicy Automatic `

-GracePeriodWithDataLossHours 2

$myfailovergroup

1. Add your database to the failover group.

PowerShellCopy

$myfailovergroup = Get-AzureRmSqlDatabase `

-ResourceGroupName $myresourcegroupname `

-ServerName $myservername `

-DatabaseName $mydatabasename | `

Add-AzureRmSqlDatabaseToFailoverGroup `

-ResourceGroupName $myresourcegroupname ` `

-ServerName $myservername `

-FailoverGroupName $myfailovergroupname

$myfailovergroup

Install Java software

The steps in this section assume that you are familiar with developing using Java and are new to working with Azure SQL Database.

**Mac OS**

Open your terminal and navigate to a directory where you plan on creating your Java project. Install **brew** and **Maven** by entering the following commands:

bashCopy

ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

brew update

brew install maven

For detailed guidance on installing and configuring Java and Maven environment, go the [Build an app using SQL Server](https://www.microsoft.com/sql-server/developer-get-started/), select **Java**, select **MacOS**, and then follow the detailed instructions for configuring Java and Maven in step 1.2 and 1.3.

**Linux (Ubuntu)**

Open your terminal and navigate to a directory where you plan on creating your Java project. Install **Maven** by entering the following commands:

bashCopy

sudo apt-get install maven

For detailed guidance on installing and configuring Java and Maven environment, go the [Build an app using SQL Server](https://www.microsoft.com/sql-server/developer-get-started/), select **Java**, select **Ubuntu**, and then follow the detailed instructions for configuring Java and Maven in step 1.2, 1.3, and 1.4.

**Windows**

Install [Maven](https://maven.apache.org/download.cgi) using the official installer. Use Maven to help manage dependencies, build, test, and run your Java project. For detailed guidance on installing and configuring Java and Maven environment, go the [Build an app using SQL Server](https://www.microsoft.com/sql-server/developer-get-started/), select **Java**, select Windows, and then follow the detailed instructions for configuring Java and Maven in step 1.2 and 1.3.

Create SqlDbSample project

1. In the command console (such as Bash), create a Maven project.bash mvn archetype:generate "-DgroupId=com.sqldbsamples" "-DartifactId=SqlDbSample" "-DarchetypeArtifactId=maven-archetype-quickstart" "-Dversion=1.0.0"
2. Type **Y** and click **Enter**.
3. Change directories into your newly created project.

bashCopy

cd SqlDbSamples

1. Using your favorite editor, open the pom.xml file in your project folder.
2. Add the Microsoft JDBC Driver for SQL Server dependency to your Maven project by opening your favorite text editor and copying and pasting the following lines into your pom.xml file. Do not overwrite the existing values prepopulated in the file. The JDBC dependency must be pasted within the larger “dependencies” section ( ).

XMLCopy

<dependency>

<groupId>com.microsoft.sqlserver</groupId>

<artifactId>mssql-jdbc</artifactId>

<version>6.1.0.jre8</version>

</dependency>

1. Specify the version of Java to compile the project against by adding the following “properties” section into the pom.xml file after the "dependencies" section.

XMLCopy

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

1. Add the following "build" section into the pom.xml file after the "properties" section to support manifest files in jars.

XMLCopy

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.0</version>

<configuration>

<archive>

<manifest>

<mainClass>com.sqldbsamples.App</mainClass>

</manifest>

</archive>

</configuration>

</plugin>

</plugins>

</build>

1. Save and close the pom.xml file.
2. Open the App.java file (C:\apache-maven-3.5.0\SqlDbSample\src\main\java\com\sqldbsamples\App.java) and replace the contents with the following contents. Replace the failover group name with the name for your failover group. If you have changed the values for the database name, user, or password, change those values as well.

JavaCopy

package com.sqldbsamples;

import java.sql.Connection;

import java.sql.Statement;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Timestamp;

import java.sql.DriverManager;

import java.util.Date;

import java.util.concurrent.TimeUnit;

public class App {

private static final String FAILOVER\_GROUP\_NAME = "myfailovergroupname";

private static final String DB\_NAME = "mySampleDatabase";

private static final String USER = "app\_user";

private static final String PASSWORD = "ChangeYourPassword1";

private static final String READ\_WRITE\_URL = String.format("jdbc:sqlserver://%s.database.windows.net:1433;database=%s;user=%s;password=%s;encrypt=true;hostNameInCertificate=\*.database.windows.net;loginTimeout=30;", FAILOVER\_GROUP\_NAME, DB\_NAME, USER, PASSWORD);

private static final String READ\_ONLY\_URL = String.format("jdbc:sqlserver://%s.secondary.database.windows.net:1433;database=%s;user=%s;password=%s;encrypt=true;hostNameInCertificate=\*.database.windows.net;loginTimeout=30;", FAILOVER\_GROUP\_NAME, DB\_NAME, USER, PASSWORD);

public static void main(String[] args) {

System.out.println("#######################################");

System.out.println("## GEO DISTRIBUTED DATABASE TUTORIAL ##");

System.out.println("#######################################");

System.out.println("");

int highWaterMark = getHighWaterMarkId();

try {

for(int i = 1; i < 1000; i++) {

// loop will run for about 1 hour

System.out.print(i + ": insert on primary " + (insertData((highWaterMark + i))?"successful":"failed"));

TimeUnit.SECONDS.sleep(1);

System.out.print(", read from secondary " + (selectData((highWaterMark + i))?"successful":"failed") + "\n");

TimeUnit.SECONDS.sleep(3);

}

} catch(Exception e) {

e.printStackTrace();

}

}

private static boolean insertData(int id) {

// Insert data into the product table with a unique product name that we can use to find the product again later

String sql = "INSERT INTO SalesLT.Product (Name, ProductNumber, Color, StandardCost, ListPrice, SellStartDate) VALUES (?,?,?,?,?,?);";

try (Connection connection = DriverManager.getConnection(READ\_WRITE\_URL);

PreparedStatement pstmt = connection.prepareStatement(sql)) {

pstmt.setString(1, "BrandNewProduct" + id);

pstmt.setInt(2, 200989 + id + 10000);

pstmt.setString(3, "Blue");

pstmt.setDouble(4, 75.00);

pstmt.setDouble(5, 89.99);

pstmt.setTimestamp(6, new Timestamp(new Date().getTime()));

return (1 == pstmt.executeUpdate());

} catch (Exception e) {

return false;

}

}

private static boolean selectData(int id) {

// Query the data that was previously inserted into the primary database from the geo replicated database

String sql = "SELECT Name, Color, ListPrice FROM SalesLT.Product WHERE Name = ?";

try (Connection connection = DriverManager.getConnection(READ\_ONLY\_URL);

PreparedStatement pstmt = connection.prepareStatement(sql)) {

pstmt.setString(1, "BrandNewProduct" + id);

try (ResultSet resultSet = pstmt.executeQuery()) {

return resultSet.next();

}

} catch (Exception e) {

return false;

}

}

private static int getHighWaterMarkId() {

// Query the high water mark id that is stored in the table to be able to make unique inserts

String sql = "SELECT MAX(ProductId) FROM SalesLT.Product";

int result = 1;

try (Connection connection = DriverManager.getConnection(READ\_WRITE\_URL);

Statement stmt = connection.createStatement();

ResultSet resultSet = stmt.executeQuery(sql)) {

if (resultSet.next()) {

result = resultSet.getInt(1);

}

} catch (Exception e) {

e.printStackTrace();

}

return result;

}

}

1. Save and close the App.java file.

Compile and run the SqlDbSample project

1. In the command console, execute to following command.

bashCopy

mvn package

1. When finished, execute the following command to run the application (it runs for about 1 hour unless you stop it manually):

bashCopy

mvn -q -e exec:java "-Dexec.mainClass=com.sqldbsamples.App"

#######################################

## GEO DISTRIBUTED DATABASE TUTORIAL ##

#######################################

1. insert on primary successful, read from secondary successful

2. insert on primary successful, read from secondary successful

3. insert on primary successful, read from secondary successful

Perform disaster recovery drill

1. Call manual failover of failover group.

PowerShellCopy

Switch-AzureRMSqlDatabaseFailoverGroup `

-ResourceGroupName $myresourcegroupname `

-ServerName $mydrservername `

-FailoverGroupName $myfailovergroupname

1. Observe the application results during failover. Some inserts fail while the DNS cache refreshes.
2. Find out which role your disaster recovery server is performing.

PowerShellCopy

$mydrserver.ReplicationRole

1. Failback.

PowerShellCopy

Switch-AzureRMSqlDatabaseFailoverGroup `

-ResourceGroupName $myresourcegroupname `

-ServerName $myservername `

-FailoverGroupName $myfailovergroupname

1. Observe the application results during failback. Some inserts fail while the DNS cache refreshes.
2. Find out which role your disaster recovery server is performing.

PowerShellCopy

$fileovergroup = Get-AzureRMSqlDatabaseFailoverGroup `

-FailoverGroupName $myfailovergroupname `

-ResourceGroupName $myresourcegroupname `

-ServerName $mydrservername

$fileovergroup.ReplicationRole

Next steps