

Load testing SQL Server using HammerDB

This tutorial shows how to use HammerDB to perform load testing on a Compute Engine SQL Server instance. You can learn how to install a SQL Server instance by using the following tutorials:

- [Creating SQL Server instances](#)
(/compute/docs/instances/sql-server/creating-sql-server-instances)
- [Creating a high-performance SQL Server instance](#)
(/compute/docs/tutorials/creating-high-performance-sql-server-instance)

There are a number of load-testing tools available. Some are free and open source, while others require licenses. [HammerDB](http://www.hammerdb.com) (<http://www.hammerdb.com>) is an open source tool that generally works well to demonstrate the performance of your SQL Server database. This tutorial covers the basic steps to use HammerDB, but there are other tools available, and you should select the tools that align best to your specific workloads.

Objectives

- Configuring SQL Server for load testing.
- Installing and running HammerDB.
- Collecting runtime statistics.
- Running the TPC-C load test.

Costs

In addition to any existing SQL Server instances running on Compute Engine, this tutorial uses billable components of Google Cloud, including:

- Compute Engine
- Windows Server

The [Pricing Calculator](/products/calculator#id=7411bcbb-3399-46bf-9dd0-9642361cd988) (/products/calculator#id=7411bcbb-3399-46bf-9dd0-9642361cd988) can generate a cost estimate based on your projected usage. The provided link shows the cost estimate for the products used in this tutorial, which can average 16 dollars (US) per day. New Google Cloud users might be eligible for a [free trial](/free-trial) (/free-trial).

Before you begin

1. [Sign in](https://accounts.google.com/Login) (https://accounts.google.com/Login) to your Google Account.

If you don't already have one, [sign up for a new account](https://accounts.google.com/SignUp) (https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a Cloud project.

★ **Note:** If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

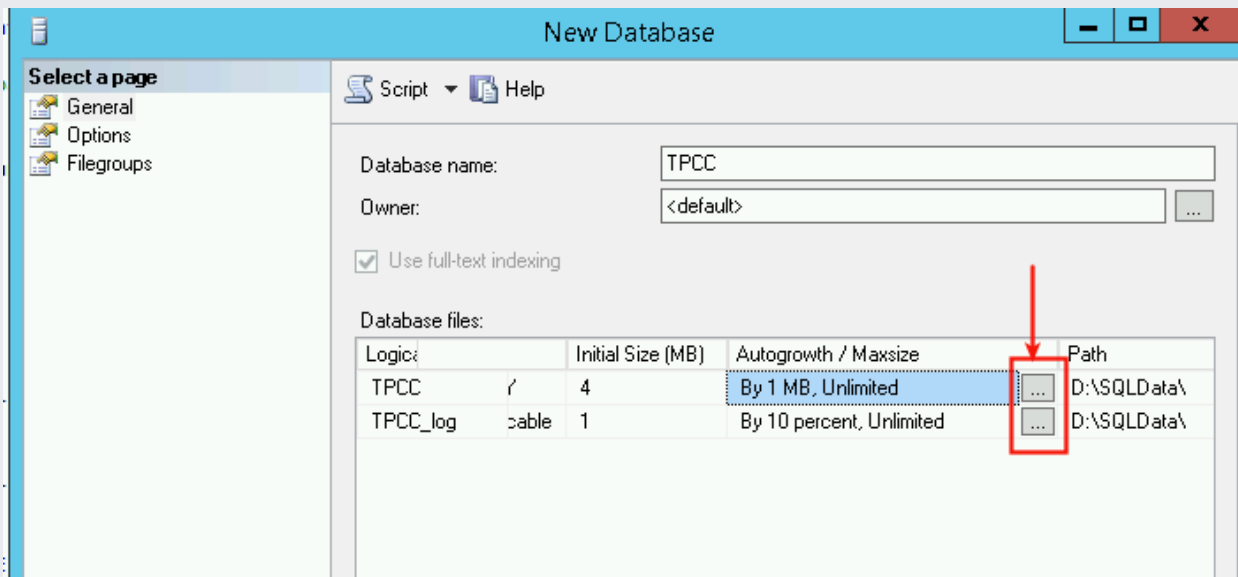
[Go to the project selector page](https://console.cloud.google.com/projectselector2/home/dashboard) (https://console.cloud.google.com/projectselector2/home/dashboard)

3. Make sure that billing is enabled for your Google Cloud project. [Learn how to confirm billing is enabled for your project](/billing/docs/how-to/modify-project) (/billing/docs/how-to/modify-project).
4. If you aren't using Windows on your local machine, install a third-party RDP client such as [Chrome RDP](https://chrome.google.com/webstore/detail/chrome-rdp/cbkkbcmldlboombapidmoeolnmdacpkch) (https://chrome.google.com/webstore/detail/chrome-rdp/cbkkbcmldlboombapidmoeolnmdacpkch) by FusionLabs.

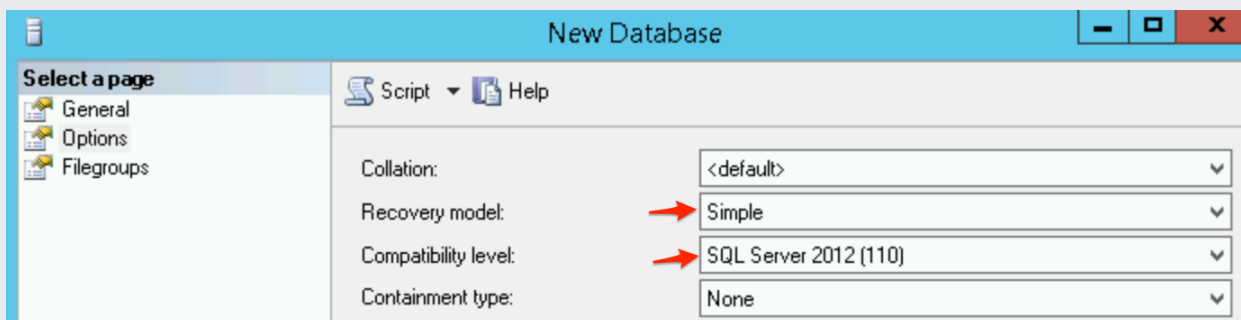
Configuring the SQL Server instance for load testing

Before you start, you should double check that your Windows [firewall rules](https://msdn.microsoft.com/en-us/library/cc646023.aspx) (https://msdn.microsoft.com/en-us/library/cc646023.aspx) are set up to allow traffic from the IP address of the new Windows instance you created. Then, create a new database for TPCC load testing and configure a user account using the following steps:

1. Right-click the **Databases** folder in SQL Server Management Studio, and then choose **New Database**.
2. Name the new database "TPCC".
3. Set the initial size of the data file to 190,000 MB and the log file to 65,000 MB.
4. Set the **Autogrowth** limits to higher values by clicking the ellipsis buttons, as shown in the following screenshot:



5. Set the data file to grow by 64 MB to unlimited size.
6. Set the log file to disable auto-growth.
7. Click **OK**.
8. In the **New Database** dialog, in the left pane, choose the **Options** page.
9. Set **Compatibility level** to **SQL Server 2012 (110)**.
10. Set the **Recovery model** to **Simple**, so that the loading doesn't fill up the transaction logs.



11. Click **OK** to create the TPCC database, which can take a few minutes to complete.
12. The preconfigured SQL Server image comes with only Windows Authentication enabled, so you will need to enable mixed mode authentication within SSMS, by following [this guide](https://msdn.microsoft.com/en-us/library/ms188670(v=sql.120).aspx) (https://msdn.microsoft.com/en-us/library/ms188670(v=sql.120).aspx).
13. [Follow these steps](https://msdn.microsoft.com/en-us/library/aa337562.aspx) (https://msdn.microsoft.com/en-us/library/aa337562.aspx) to create a new SQL Server user account on your database server that has the DBOwner permission. Name the account "loaduser" and give it a secure password.
14. Take note of your SQL Server internal IP address by using the Get-NetIPAddress commandlet, because it's important for performance and security to use the internal IP.

Installing HammerDB

You can run HammerDB directly on your SQL Server instance. However, for a more accurate test, create a new Windows instance and test the SQL Server instance remotely.

Note: You might need to disable [Internet Explorer Enhanced Security Configuration](https://technet.microsoft.com/en-us/library/dd883248(v=ws.10).aspx) (https://technet.microsoft.com/en-us/library/dd883248(v=ws.10).aspx) before downloading files to your Windows Server instance.

Creating an instance

Follow these steps to create a new Compute Engine instance:

1. In the Google Cloud Console, go to the **VM Instances** page.

GO TO THE VM INSTANCES PAGE (https://console.cloud.google.com/compute/instancesAdd)

2. Set **Name** to **hammerdb-instance**.
3. Set **Machine configuration** to at least half the number of CPUs as your database instance.
4. In the **Boot disk** section, click **Change** to begin configuring your boot disk.
5. In the **Public images** tab, choose **Windows Server 2012 R2**.

6. In the **Boot disk type** section, select **Standard persistent disk**.
7. Click **Save** to confirm your boot disk options.
8. Click **Create**.

Installing the software

When it's ready, RDP to your new Windows Server instance and install the following software:

- [SQL Server native client](http://go.microsoft.com/fwlink/?LinkID=239648&clid=0x409) (<http://go.microsoft.com/fwlink/?LinkID=239648&clid=0x409>)
- [HammerDB for Windows 64-bit](http://www.hammerdb.com/download.html) (<http://www.hammerdb.com/download.html>)

Running HammerDB

After you install HammerDB, run the `hammerdb.bat` file. HammerDB does not show up in the Start menu's applications list. Use the following command to run HammerDB:

```
C:\Program Files\HammerDB-2.20\hammerdb.bat
```

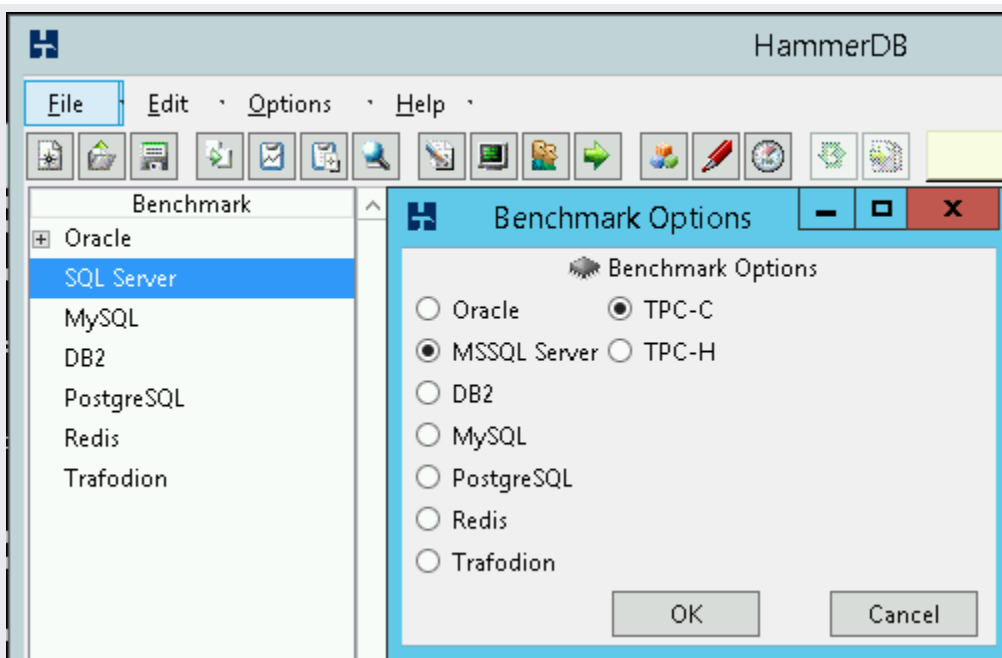
Creating the connection and schema

When the application is running, the first step is to configure the connection to build the schema.

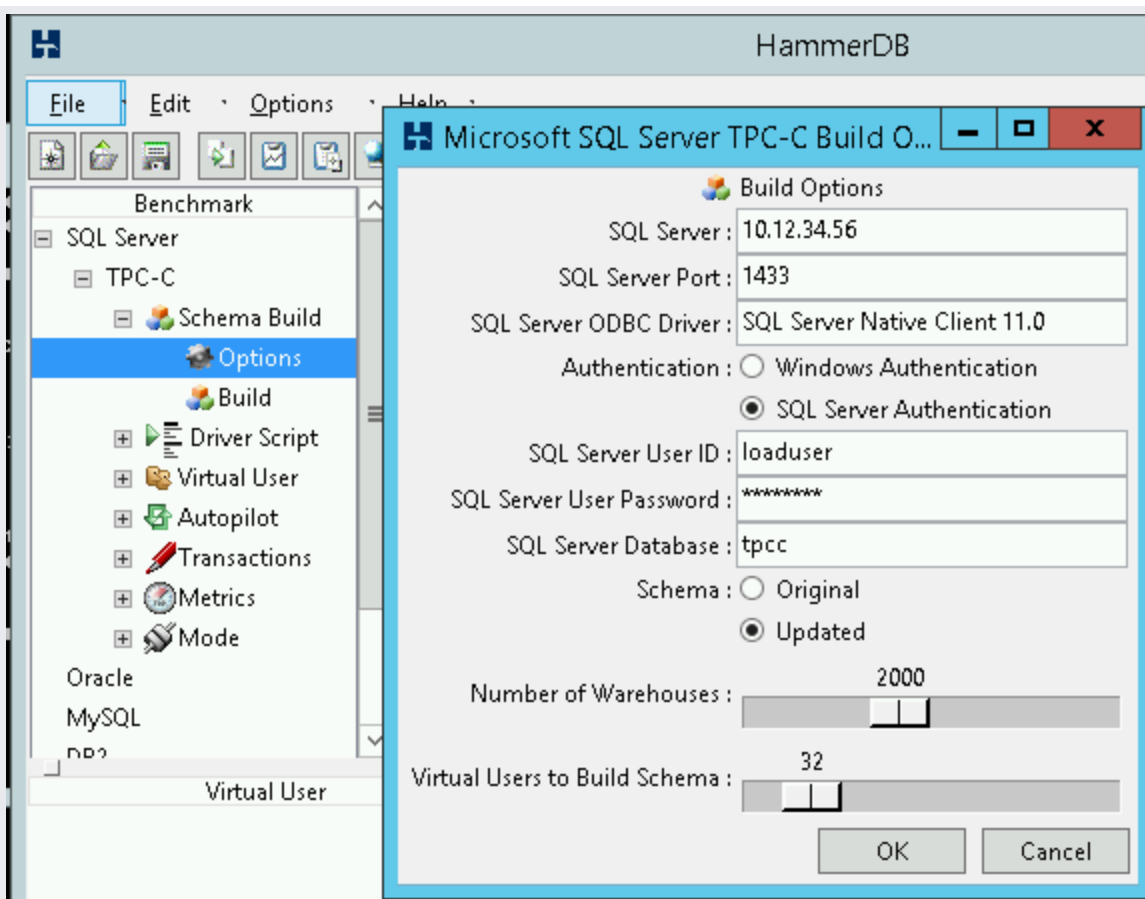
1. Double-click **SQL Server** in the **Benchmark** panel.
2. Choose **TPC-C**, an acronym that stands for: Transaction Processing Performance Council - Benchmark C. From the [TPC.org site](http://www.tpc.org/) (<http://www.tpc.org/>):

TPC-C involves a mix of five concurrent transactions of different types and complexity either executed online or queued for deferred execution. The database is comprised of nine types of tables with a wide range of record and population sizes. TPC-C is measured in transactions per minute (tpmC).

3. Click **OK**



4. In the **Benchmark** panel, next to **SQL Server**, click the plus sign (+) to expand the options.
5. Below **TPC-C**, click **Schema Build** and then double click **Options**.
6. Fill out the form to look like the figure below, using your IP address, username, and password.



7. For the **Schema** option, choose **Updated**, which creates a better TPC-C schema with more appropriate structure and better indexes.
8. In this case, the **Number of Warehouses** (the scale) is set to 2000, but you don't have to set it that high, because creating 2000 warehouses will take several hours to complete. Some guidelines suggest 10 to 100 warehouses per CPU. For this tutorial, set this value to 10 times the number of cores: 160 for a 16-core instance.
9. For **Virtual Users to Build Schema**, choose a number that is between 1- and 2-times the number of client vCPUs. You can click the grey bar next to the slider to increment the number.
10. Click **OK**
11. Double click the **Build** option below the **Schema Build** section to create the schema and load the tables. When that completes, click the red flash light icon in the top center of the screen to destroy the virtual user and move to the next step.

If you created your database with the **Simple** recovery model, you might want to change it back to **Full** at this point to get a more accurate test of a production scenario. This will not take

effect until after you take a full or differential backup to trigger the start of the new log chain.

Important: If you plan to run multiple tests, [make a full backup of your new TPC-C database](https://msdn.microsoft.com/en-us/library/ms187510(v=sql.120).aspx#SSMSProcedure) ([https://msdn.microsoft.com/en-us/library/ms187510\(v=sql.120\).aspx#SSMSProcedure](https://msdn.microsoft.com/en-us/library/ms187510(v=sql.120).aspx#SSMSProcedure)), so that you can restore it later. Backing up can save you time compared to creating the database again by using the tool. If you revert the database to a Full recovery model, you should backup the transaction logs to clear them out after each test.

Creating the driver script

HammerDB uses the driver script to orchestrate the flow of SQL statements to the database to generate the required load.

1. In the **Benchmark** panel, expand the **Driver Script** section and double-click **Options**.
2. Verify the settings match what you used in the **Schema Build** dialog.
3. Choose **Timed Test Driver Script**.
4. The **Checkpoint when complete** option forces the database to write everything to disk at the end of the test, so check this only if you plan on running multiple tests in a row.
5. To ensure a thorough test, set **Minutes of Rampup Time** to 5 and **Minutes for Test Duration** to 20.
6. Click **OK** to exit the dialog.
7. Double-click **Load** in the **Driver Script** section of the **Benchmark** panel to activate the driver script.

Microsoft SQL Server TPC-C Drive... - [] [X]

Driver Options

SQL Server : 10.12.34.56

SQL Server Port : 1433

SQL Server ODBC Driver : SQL Server Native Client 11.0

Authentication : Windows Authentication
 SQL Server Authentication

SQL Server User ID : loaduser

SQL Server User Password : *****

SQL Server Database : tpcc

TPC-C Driver Script : Standard Driver Script
 Timed Test Driver Script

Total Transactions per User : 1000000

Exit on SQL Server Error :

Keying and Thinking Time :

Checkpoint when complete :

Minutes of Rampup Time : 5

Minutes for Test Duration : 20

OK Cancel

Creating virtual users

Creating a realistic load typically requires running scripts as multiple different users. Create some virtual users for the test.

1. Expand the **Virtual Users** section and double click **Options**.
2. If you set your warehouse count (scale) to 160, then set the **Virtual Users** to 16, because the TPC-C guidelines recommend a 10x ratio to prevent row locking. Select the **Show Output** checkbox to enable error messages in the console.
3. Click **OK**

Collecting runtime statistics

HammerDB and SQL Server don't easily collect detailed runtime statistics for you. Although the statistics are available deep within SQL Server, they need to be captured and calculated on a

regular basis. If you do not already have a procedure or tool to help capture this data, you can use the procedure below to capture some useful metrics during your testing. The results will be written to a CSV file in the Windows temp directory. You can copy the data to a Google Sheet using the **Paste Special > Paste CSV** option.

To use this procedure, you first must temporarily enable **OLE Automation Procedures** to write the file to disk. Remember to disable it after testing:

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Ole Automation Procedures', 1;
GO
RECONFIGURE;
GO
```

Note: Although this procedure is very small, it can affect the total throughput reported by a fraction of a percent.

Here's the code to create the `sp_write_performance_counters` procedure in SQL Server Management Studio. Before starting the load test, you will execute this procedure in Management Studio.:

```
USE [master]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

/****
LogFile path has to be in a directory that SQL Server can Write To.
*/
CREATE PROCEDURE [dbo].[sp_write_performance_counters] @LogFile varchar (2000) = 'C:
AS
```

```

BEGIN
--File writing variables
DECLARE @OACreate INT, @OAFile INT, @FileName VARCHAR(2000), @RowText VARCHAR(500),
--Variables to save last counter values
DECLARE @LastTPS BIGINT, @LastLRS BIGINT, @LastLTS BIGINT, @LastLWS BIGINT, @LastNDS
--Variables to save current counter values
DECLARE @TPS BIGINT, @Active BIGINT, @SCM BIGINT, @LRS BIGINT, @LTS BIGINT, @LWS BIG

SELECT @Loops = case when (@SecondsToRun % @RunIntervalSeconds) > 5 then @SecondsToR
SET @LoopCounter = 0
SELECT @WaitForSeconds = CONVERT(varchar, DATEADD(s, @RunIntervalSeconds , 0), 114)
SELECT @FileName = @LogFile + FORMAT ( GETDATE(), '-MM-dd-yyyy_m', 'en-US' ) + '.txt

--Create the File Handler and Open the File
EXECUTE sp_OACreate 'Scripting.FileSystemObject', @OACreate OUT
EXECUTE sp_OAMethod @OACreate, 'OpenTextFile', @OAFile OUT, @FileName, 2, True, -2

--Write the Header
EXECUTE sp_OAMethod @OAFile, 'WriteLine', NULL, 'Transactions/sec, Active Transaction
--Collect Initial Sample Values
SET ANSI_WARNINGS OFF
SELECT
    @LastTPS= max(case when counter_name = 'Transactions/sec' then cntr_value end),
    @LastLRS = max(case when counter_name = 'Lock Requests/sec' then cntr_value end),
    @LastLTS = max(case when counter_name = 'Lock Timeouts/sec' then cntr_value end),
    @LastLWS = max(case when counter_name = 'Lock Waits/sec' then cntr_value end),
    @LastNDS = max(case when counter_name = 'Number of Deadlocks/sec' then cntr_value
    @LastAWT = max(case when counter_name = 'Average Wait Time (ms)' then cntr_value e
    @LastAWT_Base = max(case when counter_name = 'Average Wait Time base' then cntr_va
    @LastALWT = max(case when counter_name = 'Average Latch Wait Time (ms)' then cntr_
    @LastALWT_Base = max(case when counter_name = 'Average Latch Wait Time base' then
FROM sys.dm_os_performance_counters
WHERE counter_name IN (
'Transactions/sec',
'Lock Requests/sec',
'Lock Timeouts/sec',
'Lock Waits/sec',
'Number of Deadlocks/sec',
'Average Wait Time (ms)',
'Average Wait Time base',
'Average Latch Wait Time (ms)',
'Average Latch Wait Time base') AND instance_name IN( '_Total' , '' )
SET ANSI_WARNINGS ON
WHILE @LoopCounter <= @Loops

```

```

BEGIN
WAITFOR DELAY @WaitForSeconds
SET ANSI_WARNINGS OFF
SELECT
    @TPS= max(case when counter_name = 'Transactions/sec' then cntr_value end) ,
    @Active = max(case when counter_name = 'Active Transactions' then cntr_value end)
    @SCM = max(case when counter_name = 'SQL Cache Memory (KB)' then cntr_value end)
    @LRS = max(case when counter_name = 'Lock Requests/sec' then cntr_value end) ,
    @LTS = max(case when counter_name = 'Lock Timeouts/sec' then cntr_value end) ,
    @LWS = max(case when counter_name = 'Lock Waits/sec' then cntr_value end) ,
    @NDS = max(case when counter_name = 'Number of Deadlocks/sec' then cntr_value end)
    @AWT = max(case when counter_name = 'Average Wait Time (ms)' then cntr_value end)
    @AWT_Base = max(case when counter_name = 'Average Wait Time base' then cntr_value
    @ALWT = max(case when counter_name = 'Average Latch Wait Time (ms)' then cntr_valu
    @ALWT_Base = max(case when counter_name = 'Average Latch Wait Time base' then cntr
FROM sys.dm_os_performance_counters
WHERE counter_name IN (
'Transactions/sec',
'Active Transactions',
'SQL Cache Memory (KB)',
'Lock Requests/sec',
'Lock Timeouts/sec',
'Lock Waits/sec',
'Number of Deadlocks/sec',
'Average Wait Time (ms)',
'Average Wait Time base',
'Average Latch Wait Time (ms)',
'Average Latch Wait Time base') AND instance_name IN( '_Total' ,'' )
SET ANSI_WARNINGS ON

SELECT @AWT_DIV = case when (@AWT_Base - @LastAWT_Base) > 0 then (@AWT_Base - @Last
    @ALWT_DIV = case when (@ALWT_Base - @LastALWT_Base) > 0 then (@ALWT_Base - @Last

SELECT @RowText = '' + convert(varchar, (@TPS - @LastTPS)/@RunIntervalSeconds) + ',
    convert(varchar, @Active) + ', ' +
    convert(varchar, @SCM) + ', ' +
    convert(varchar, (@LRS - @LastLRS)/@RunIntervalSeconds) + ', ' +
    convert(varchar, (@LTS - @LastLTS)/@RunIntervalSeconds) + ', ' +
    convert(varchar, (@LWS - @LastLWS)/@RunIntervalSeconds) + ', ' +
    convert(varchar, (@NDS - @LastNDS)/@RunIntervalSeconds) + ', ' +
    convert(varchar, (@AWT - @LastAWT)/@AWT_DIV) + ', ' +
    convert(varchar, (@ALWT - @LastALWT)/@ALWT_DIV)

SELECT @LastTPS = @TPS,
    @LastLRS = @LRS,

```

```
@LastLTS = @LTS,  
@LastLWS = @LWS,  
@LastNDS = @NDS,  
@LastAWT = @AWT,  
@LastAWT_Base = @AWT_Base,  
@LastALWT = @ALWT,  
@LastALWT_Base = @ALWT_Base  
  
EXECUTE sp_OAMethod @OAFfile, 'WriteLine', Null, @RowText  
  
SET @LoopCounter = @LoopCounter + 1  
  
END  
  
--CLEAN UP  
EXECUTE sp_OADestroy @OAFfile  
EXECUTE sp_OADestroy @OACreate  
print 'Completed Logging Performance Metrics to file: ' + @FileName  
  
END  
  
GO
```

Running the TPC-C load test

In SQL Server Management Studio, execute the collection procedure using the following script:

```
Use master  
Go  
exec dbo.sp_write_performance_counters
```

On the Compute Engine instance where you installed HammerDB, start the test in the HammerDB application:

1. In the **Benchmark** panel, under **Virtual Users** double-click **Create** to create the virtual users, which will activate the **Virtual User Output** tab.
2. Double-click **Run** just below the **Create** option to kick off the test.

3. When the test completes you will see the Transactions Per Minute (TPM) calculation in the **Virtual User Output** tab.
4. You can find the results from your collection procedure in the `c:\Windows\temp` directory.
5. Save all of these values to a Google Sheet and use them to compare multiple test runs.

Cleaning up

After you've finished the SQL Server load-testing tutorial, you can clean up the resources that you created on Google Cloud so they won't take up quota and you won't be billed for them in the future. The following sections describe how to delete or turn off these resources.

Deleting the project

The easiest way to eliminate billing is to delete the project that you created for the tutorial.

To delete the project:


! **Caution:** Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an `appspot.com` URL, delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

[Go to the Manage resources page \(https://console.cloud.google.com/iam-admin/projects\)](https://console.cloud.google.com/iam-admin/projects)


2. In the project list, select the project that you want to delete and then click **Delete** .
3. In the dialog, type the project ID and then click **Shut down** to delete the project.

Deleting instances

To delete a Compute Engine instance:

1. In the Cloud Console, go to the **VM Instances** page.

[Go to the VM Instances page](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

2. Click the checkbox for the instance you want to delete.
3. Click **Delete**  to delete the instance.

What's next

- Review the [SQL Server best practices guide](/compute/docs/instances/sql-server/best-practices) (/compute/docs/instances/sql-server/best-practices).
- Try out other Google Cloud features for yourself. Have a look at our [tutorials](/docs/tutorials) (/docs/tutorials).

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (https://www.apache.org/licenses/LICENSE-2.0). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2020-06-23 UTC.