Microsoft

SQL Server 2017 on Linux
Quick Start Guide

Contents

Who should read this guide?

This technical guide is for database architects, administrators, and developers who are looking to understand and explore the latest Microsoft SQL Server capabilities. The organization of topics and sections in this guide is based on questions and feedback from the SQL Server on Linux Engineering Town Hall webinar series. Each section provides a brief on selected SQL Server topics, including answers to frequently asked questions and links to additional documentation.

With this guide, you should attain a solid foundational skillset for installing, administering, and managing SQL Server on Linux, along with practical knowledge of several SQL Server features and capabilities.

Getting started with SQL Server on Linux

Why SQL Server with Linux?

Today's customers are demanding more flexibility in their choice of platform, ensuring they can get the maximum impact from their data estate. Microsoft offers a high-quality, enterprise-level database platform that aligns with customer needs. SQL Server on Linux enables customers to choose the best operating system (OS), or combination of operating systems, for their environment—whether it's open source, proprietary, or a mixture of both.

Along with increased flexibility, SQL Server 2017 extends more support for Linux distributions, which impacts your bottom line. With SQL Server 2017 on Linux, it's possible to realize a lower total cost of ownership (TCO) and quicker return on investment (ROI). All required features are built in, including in-memory capabilities, security, migration tools, and high availability/disaster recovery (HADR) at no extra cost. This means you can avoid extra licenses, training time, and hardware costs while ensuring that your initial investment in SQL Server pays off.

> If your company is primarily running on Linux, then you've previously been limited in your choices of database servers. Microsoft has removed that limitation with the flexibility to run your database workloads on Linux.

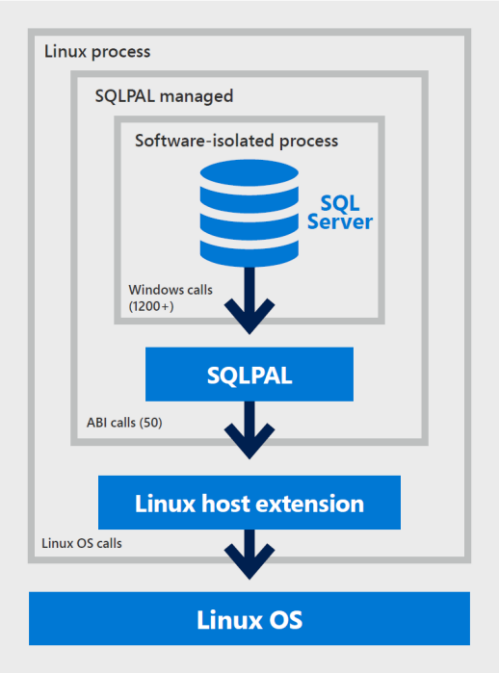
Supported platforms

SQL Server 2017 is supported on Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), and Ubuntu. It's also supported as a Docker image, which can run on Docker Engine on Linux or Docker for Windows and Mac. Additionally, Microsoft supports deploying and managing SQL Server containers by using OpenShift and Kubernetes. Plus, you can provision a Linux SQL Server virtual machine (VM) on Microsoft Azure.

The currently supported versions of these platforms can be found in the SQL Server installation guide for Linux.

Architectural changes

SQL Server on Windows and Linux both use a common code base. That is, the SQL Server core engine hasn't been altered to allow it to run on Linux. However, SQL Server introduced a Platform Abstraction Layer (SQLPAL) that's responsible for abstraction of calls and communication between SQL Server and the underlying OS. The host extension is simply a native Linux application. Low-level OS functions are native calls to optimize the input/output (I/O), memory, and CPU usage. When the host extension starts, it loads and initializes SQLPAL, which then brings up SQL Server. The SQLPAL software isolates processes that are merely a collection of threads and allocations, providing the required translation for the rest of the code. Adding this new layer to the SQL Server architecture means that users have the same enterprise-level core features and benefits that have made SQL Server so powerful on Windows, regardless of the operating environment.



Comparing SQL on Windows vs. Linux

The core database engine for SQL Server on Linux includes essential services for storing, processing, and securing data. It also supports replication, full-text search, tools for managing relational and XML data, and integration for database analytics.

The table below highlights the available features on Windows and Linux, respectively.

		Windows	Linux
Editions	Developer, Express, Web, Standard, Enterprise	●	●
Services	Database Engine, Integration Services	●	●
	Master Data Services, Data Quality Services	●	
Mission-critical performance	Maximum number of cores	Operating system maximum	Operating system maximum
	Maximum memory utilized per instance	Operating system maximum	Operating system maximum
	Maximum database size	524 PB	524 PB
	Basic OLTP (basic In-Memory OLTP, basic operational analytics)	●	●
	Advanced OLTP (advanced In-Memory OLTP, advanced operational analytics, adaptive query processing)	●	●
	Basic high availability (2-node single database failover, non-readable secondary)	●	●
	Advanced high availability (Always On, multi-node, multi-database failover, readable secondaries)	●	●

		Windows	Linux
Security	Basic security (basic auditing, Row-Level Security, Dynamic Data Masking, Always Encrypted)	●	●
	Advanced security (Transparent Data Encryption)	●	●
Data warehousing	PolyBase	●	
	Basic data warehousing/data marts (basic In-Memory Columnstore, partitioning, compression)	●	●
	Advanced data warehousing (advanced In-Memory Columnstore)	●	●
	Advanced data integration (Fuzzy Grouping and Lookups)	●	
Tools	Windows ecosystem: full-fidelity management and development tools (SSMS and SSDT), command-line tools	●	●
	Linux/OSX/Windows ecosystem: development tools (Visual Studio Code), GUI-based admin tools (SSMS, VS Code, SQL Operation Studio), command-line tools	●	●
Developer	Programmability (T-SQL, CLR, Data Types, JSON, Graph)	●	●
	Windows file system integration (FileTable)	●	
BI & Advanced Analytics	Corporate business intelligence (Analysis Services, Reporting Services, multi-dimensional models, basic tabular model)	●	
	Machine learning services (R, StreamInsight, and Python integration)*	●	●
Hybrid cloud	Stretch Database	●	

*This applies for SQL Server 2019.

For a list of features that aren't supported on Linux and other key issues, view the Unsupported features & services and Known issues pages.

Note: SQL Server Management Studio (SSMS) isn't supported on Linux. However, you can install SSMS on a Windows computer and connect to SQL Server on Linux remotely. Microsoft also provides an mssql-tools package for the Linux command line. Azure Data Studio (formerly SQL Operations Studio) is another cross-platform database tool for managing SQL Server on Linux. See Managing SQL Server for further discussion.

Does SQL Server on Linux support replication?

All types of replication will be supported in SQL Server 2019 on Linux instances. The Linux instances of SQL Server 2019 can participate in transactional, merge, and snapshot replication topologies in the publisher, distributor, or subscriber roles.

Can I enable distributed transactions on SQL Server on Linux?

Yes. SQL Server 2019 on Linux instances will initiate and participate in distributed transactions. This is enabled by Microsoft Distributed Transaction Coordinator (MSDTC) and RPC Endpoint Mapper functionality within SQL Server.

SQL Server installation on Linux

Distinct system requirements and prerequisites must be met to install SQL Server on Linux.

The minimum requirements for installation are as follows:

- **Memory**, 2 GB; disk space, 6 GB; **processor speed**, 2 GHz; and **processor core**, 2 cores
- **File system support** is only **XFS** or **EXT4** (other file systems, such as BTRFS, are unsupported)
- **Processor type** can be x64-compatible only
- **Network File System (NFS)** considerations:
    - For NFS remote shares in production, use NFS version 4.2 or higher
    - Locate only the /var/opt/mssql directories on the NFS mount

Are there any major differences regarding installation of SQL Server for Red Hat vs. SUSE Linux Enterprise Server vs. Ubuntu?

There's very little difference. You just need to use respective distro such as yum for Red Hat Enterprise Linux, zypper for SUSE Linux Enterprise Server, and apt for Ubuntu. Other installation steps are common.

Installing SQL Server packages

Microsoft maintains package repositories for installing SQL Server and supports installation via native package managers with their distros.

Follow basic installation process

At a basic level, the setup process breaks down as follows:

1. Add the appropriate package repository information for your Linux distribution.
2. Run the native package installer command for your required distro.

- For Red Hat Enterprise Linux (RHEL) 7.3+:

```
sudo yum install -y mssql-server
```

- For SUSE Linux Enterprise Server (SLES) v12 SP2:

```
sudo zypper install -y mssql-server
```

- For Ubuntu 16.04:

```
sudo apt-get install -y mssql-server
```

3. Run the configuration script using mssql-conf to complete setup:

```
sudo /opt/mssql/bin/mssql-conf setup
```

4. Optionally, enable communication for the appropriate (default 1433) port on your firewall.

5. Optionally, install the SQL Server command-line tools, Microsoft ODBC drivers, and their dependencies. Use Full-Text Search, and SQL Server Integration Services, as required.

Install from offline sources

You can also install SQL Server offline, if your Linux machine doesn't have access to the online repositories. The package files are available for download directly from the Microsoft repository. Once you download the files, move the package to your Linux machine and install the database engine package with the missing dependencies.

Run a SQL Server container image with Docker

Microsoft has a public repository of SQL Server Docker images in the Docker Store. You can use Docker to pull and run the SQL Server container image mssql-server-linux:

```
sudo docker pull mcr.microsoft.com/mssql/server:vNext-CTP2.0-ubuntu
```

To run the container image with Docker, you can use the bash shell (Linux/macOS) or elevated PowerShell command prompt:

```
sudo docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=<YourStrong!Passw0rd>' \
   -p 1433:1433 --name sql1 \
   -d mcr.microsoft.com/mssql/server:vNext-CTP2.0-ubuntu
```

For further details, go to Quickstart: Run the SQL Server container image with Docker.

From where can I pull official images for SQL Server on Linux for Docker Engine?

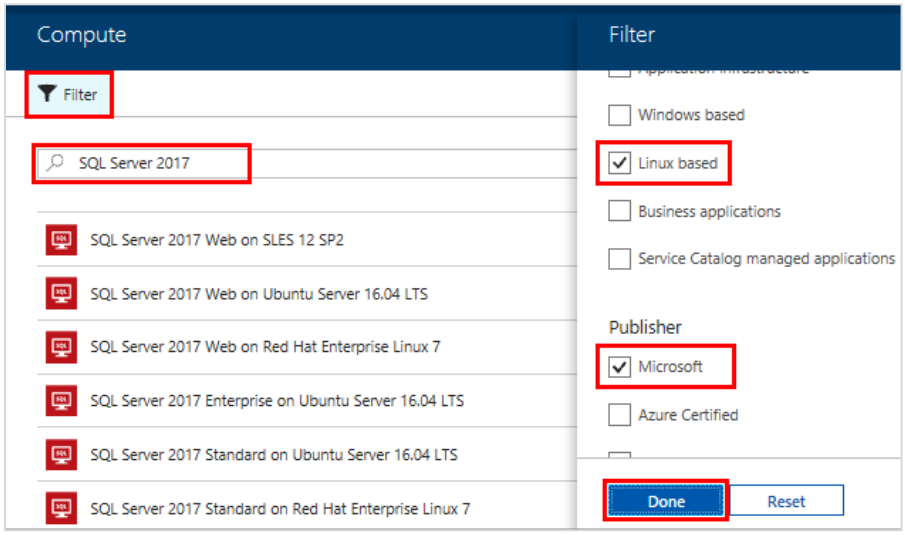
The official images will be available on mcr.microsoft.com/mssql/server. These images consist of SQL Server running on Linux based on Ubuntu 16.04.

What is the folder location of all SQL Server files installed on Linux? What are the permission criteria?

All files of SQL Server are located in the /var/opt/mssql file folder from the default installation. This folder needs to be owned by the mssql user and belong to the mssql group, which should have read-write permissions of all files and directories under it. Microsoft doesn't support changing the ownership of SQL Server directory and files from the default installation. The mssql account and group are specifically used for SQL Server and have no interactive login access.

Provision a Linux SQL Server virtual machine in the Azure portal

SQL Server 2017 images are available running on Linux, while earlier SQL Server versions are available running on Windows. Gallery images are regularly updated with security and functionality patches. The gallery images allow you to spin up a new SQL Server VM in less than 10 minutes.



You can also run SQL Server 2017 on other platforms like Amazon Web Services, DigitalOcean, and Google Cloud Platform. For more information, go to Quickstart: Run the SQL Server in the cloud.

Does Microsoft provide any tool, app, or utility for making required configuration changes—such as modifying the TCP port number or the SQL Server directories?

Yes. There's a configuration tool for SQL Server on Linux—mssql-conf—to change these kinds of configuration parameters.

Configuration capabilities

Configuration management for SQL Server on Linux can be done using the mssql-conf script.

You can use the mssql-conf utility to set the following parameters:

- Change the TCP port used by SQL Server to listen for remote connections with mssql-confg:

```
sudo /opt/mssql/bin/mssql-conf set network.tcpport <new_tcp_port>
```

- Use the following commands to change the properties filelocation.defaultdatadir and filelocation.defaultlogdir settings to new database and log-file locations. (The default file location of data and log file is /var/opt/mssql/data.)

```
sudo chown mssql[new path]
sudo chgrp mssql[new path]
sudo /opt/mssql/bin/mssql-conf set filelocation.defaultdata \
dir[new path]
sudo /opt/mssql/bin/mssql-conf set filelocation.defaultlog \
dir[new path]
sudo systemctl restart mssql-server
```

- Change the default dump directory location by using the filelocation.defaultdumpdir property with the set command.
- For changing the SQL Server instance name running on a Linux server, first drop the existing SQL Server instance name using the system-stored procedure sp_dropserver. Then add the new SQL Server instance name using the system-stored procedure sp_addserver with mssql-confg.
- Update the property filelocation.defaultbackupdir setting to point to the new backup directory location with the set command.
- Use the memory setting memory.memorylimitmb, which represents the total memory available for SQL Server. (Note that isn't the SQL Server Max or Min memory setting.) By default, SQL Server uses 80 percent of the total RAM.

```
sudo /opt/mssql/bin/mssql-conf set memory.memorylimitmb 3328
```

- Set the property hadr.hadrenabled to 1. This will enable availability groups on SQL Server.
- Use the set-collation option to change the values for any of the supported collations.
- To enable or disable multiple trace flags, use a space in between the traceflags values with the on clause at the end of the command, as in this example:

```
#/opt/mssql/bin/mssql-conf traceflag 2345 3226 3456 on
```

For more information on various other parameter settings with mssql-conf, go to Configure SQL Server on Linux with the mssql-conf tool.

Once you're done with installation, you can follow further best practices and configuration guidelines for performance optimization and more.

What is the change in licensing for running SQL Server on Linux as compared to Windows?

SQL Server is licensed the same way for both Windows and Linux. In fact, you can license SQL Server, and then choose to use that license on the platform of your choice. SQL Server editions are fundamentally the same for Windows and Linux.

Licensing

You can employ licensing on physical machines as well as on virtual machines or containers, or use an Azure VM for SQL Server on Linux for a per-usage cost.

Physical machine licensing

Two main licensing models apply to SQL Server, including:

- **Server + CAL**. Providing low-cost access to incremental SQL Server deployments, this is an option to license users and/or devices.
    - Each server running SQL Server software must have a valid license. Additionally, each user and/or device accessing a licensed SQL Server should have a SQL Server Client Access License (CAL) that's the same version or newer. For example, a user would need a SQL Server 2012 or 2017 CAL to access a SQL Server 2012 Standard Edition server.
- **Per core**. Core-based licensing is applicable when you're unable to count users/devices, have internet/extranet-based workloads, or have systems that integrate with external-facing workloads.

- When running SQL Server in a physical operating system environment (OSE), you should license all physical cores on the server. A minimum of four core licenses is required for each physical processor on the server.

Licensing for virtualization and containers

Two primary licensing options exist for VMs and containers in SQL Server, including:

- **License individual virtual machines and containers**.
  - When you want to license a VM or container with core licenses, you need to purchase a core license for each virtual core (virtual thread) allocated to the VM—or the number of cores configured for access by the container. The minimum is four core licenses per VM or container.
  - To license a single VM or container with a server license (for Standard Edition only), purchase a server license for each VM or container and a CAL for each user or device.
- **License for maximum densities in highly virtualized or high-density container environments**.
  - By fully licensing the server (or server farm) with Enterprise Edition core licenses and Software Assurance (SA) coverage based on the total number of physical cores on the servers, you can deploy an unlimited number of VMs or containers on the server. This lets you use the entire capacity of the licensed hardware.

Licensing for SQL Server on Azure virtual machines

You can license SQL Server on Azure VMs through the per-minute cost of running the gallery image. This means you pay only for what you use. Alternatively, through Software Assurance, you can transfer existing licenses into Azure with the bring-your-own-license (BYOL) gallery images.

Licensing for high availability

If the secondary replica you're using for failover support is set to "not readable," it's not necessary to separately license it for SQL Server. If the replica is readable or serving data—such as reports to clients running active SQL Server workloads, or performing work such as additional backups from secondary servers—then you must separately license it for SQL Server. Plus, the server running the active replica must have a valid license and be covered with Software Assurance. For each licensed primary replica covered with Software Assurance, you're allotted one secondary replica with equal or lesser compute capacity.

For more information on how to buy SQL Server, visit the Microsoft data platform page.

Administering and securing SQL Server

When connecting to SQL Server on Linux, you need to keep your organization database secure. Security is at the core of SQL Server on Red Hat Enterprise Linux. It provides integrated authentication with Active Directory (AD) and includes a robust set of features—such as auditing, Row-Level Security, Dynamic Data Masking, Always Encrypted, and Transparent Data Encryption—that separates and protects organizational data at rest and in motion.

Authentication and AD integration

SQL Server authentication adds a security layer via a username and password. You can maintain control by creating a login in the master database. Then, to grant others access to SQL Server, you can assign categorical permissions and privileges by using the built-in fixed database roles.

You can configure SQL Server on Linux to support AD authentication, also known as integrated authentication. This enables domain-joined clients on either Windows or Linux to authenticate to SQL Server using their domain credentials and the Kerberos protocol. As a result, you can have integrated security logins with SQL Server on Linux systems—meaning applications can use AD authentication when connecting to databases, without the need to store credentials in application configuration files. Currently, only AD logins can be authenticated. Local Linux account logins are not supported.

Before you configure AD authentication, you need to set up an AD Domain Controller (Windows) on your network.

Here's how you can set up integrated authentication with Active Directory:

1. Join the SQL Server host to AD domain using realmd.
2. Create an AD user for SQL Server and set ServicePrincipalName (SPN).
3. On your domain controller, run the New-ADUser PowerShell command to create a new AD user with a password that never expires. Set the SPN using the setspn.exe tool.
4. Configure SQL Server service keytab with the ktutil command.
5. Create AD-based logins in Transact-SQL (T-SQL).
6. Connect to SQL Server using AD authentication.

For further details on each step, go to the Tutorial: Use Active Directory authentication with SQL Server on Linux.

Note: At this time, certificate-based authentication is the only method supported for the database mirroring endpoint. The Windows authentication method will be enabled in a future release. For more information, see Security limitations for SQL Server on Linux.

Is there any support for third-party Active Directory providers with SQL Server on Linux?

SQL Server 2019 preview on Linux supports OpenLDAP, which allows third-party providers to join Active Directory. You can configure a SQL Server on Linux host machine with AD authentication with third-party AD providers, such as PowerBroker Identity Services (PBIS), Vintela Authentication Services (VAS), and Centrify.

SQL Server security features and configuration

Transparent Data Encryption

Transparent Data Encryption (TDE) encrypts data files as they're stored on the hard drive. The database files can't be read without access to the encryption key. The database can be moved only by authorized users, as high-level administrators can manage, back up, and recreate the key. When TDE is configured, the tempdb database is also automatically encrypted.

Backup Encryption

Backup Encryption shields backup data files. By specifying the encryption algorithm and the encryptor (a certificate or asymmetric key) when creating a backup, you can create an encrypted backup file.

```
USE master;
GO
CREATE CERTIFICATE BackupEncryptCert WITH SUBJECT = 'Database backups';
GO
BACKUP DATABASE [AdventureWorks2014] TO
    DISK = N'/var/opt/mssql/backups/AdventureWorks2014.bak'
    WITH  COMPRESSION,  ENCRYPTION (
       ALGORITHM = AES_256,
       SERVER CERTIFICATE = BackupEncryptCert
    ),
STATS = 10;
GO
```

Always Encrypted

Always Encrypted secures sensitive data on the client side. It can protect personal data, such as credit card numbers or national identification numbers (Social Security numbers), stored in SQL

Server databases. It encrypts sensitive data inside their applications and never reveals the encryption keys to the database engine.

Row-Level Security

Row-Level Security (RLS) enables you to manage access to rows in a table based on a customizable policy. With RLS, you can control access to individual rows in a database table based on the characteristics of the user executing a query (for example, group membership or execution context).

Dynamic Data Masking

Dynamic Data Masking (DDM) limits sensitive data exposure by obscuring it to non-privileged users. Use an ALTER TABLE statement to add a masking function to the specific column in the respective table:

```
USE AdventureWorks2014;
GO
ALTER TABLE Person.EmailAddress
ALTER COLUMN EmailAddress
ADD MASKED WITH (FUNCTION = 'email()');
GO
```

Fine-grained audit features help you enforce a data audit policy and track user activity. Track and log both server-level and individual database events. To learn more about these security features, see Walkthrough for the security features of SQL Server on Linux.

Performance tuning

SQL Server offers leading performance optimization features—such as clustered and non-clustered indexes and memory-optimized tables—to improve performance for database workloads. You can easily enable and configure these features with the native T-SQL commands.

Columnstore index

A columnstore index is a technology for storing and querying large stores of data in a columnar data format. Compared with traditional indexes, this index compresses data to reduce memory and disk footprint, filtering scans automatically through rowgroup elimination, and processing queries in batches. Though this can be helpful for data warehousing, not all data warehousing scenarios benefit from columnstore indexes. Visit Which Columnstore Index is right for my workload for guidelines on where columnstore indexes are appropriate.

There are two types of columnstore indexes:

- A **non-clustered columnstore index (NCCI)** is a read-only index created on a table that already possesses a standard clustered index. It also may be created on a heap table. An NCCI can include one or more columns from the table and have an optional condition that filters the rows. It contains a copy of part or all rows and columns in the underlying table. An NCCI enables real-time operational analytics. The online transaction processing (OLTP) workload uses the underlying clustered index while analytics run concurrently on the columnstore index.

  You can create the non-clustered index as shown in the following T-SQL commands:

  ```
  CREATE NONCLUSTERED COLUMNSTORE INDEX
  [IX_SalesOrderDetail_ColumnStore]
     ON Sales.SalesOrderDetail  (UnitPrice, OrderQty, ProductID);
  GO
  ```

- A **clustered columnstore index** sorts and stores the actual data rows in the table or view based on their key value. This means that all columns are included in the index. What's the difference between these types? A non-clustered index is a secondary index created on a rowstore table, while a clustered columnstore index serves as the primary storage for the entire table.

In-Memory OLTP

In-Memory OLTP is a memory-optimized database engine integrated into SQL Server, optimized for transaction processing. It vastly improves the speed of OLTP applications because SQL Server only needs to interact with the data in memory. In-Memory OLTP solves for both contention on hotspots in database tables or for log-write. However, it's not suitable for all scenarios, such as when most queries are performing aggregation over large ranges of data.

You can find a list of use cases at Usage Scenarios for In-Memory OLTP. To use In-Memory OLTP, you should set the database to a compatibility level of at least 130.

In-Memory OLTP technology is based on two main components: memory-optimized tables and natively compiled stored procedures.

Is there a difference in performance for SQL Server on Linux as compared to Windows?

The performance of SQL Server on Linux compared to Windows is the same—sometimes even better.

SQL Server 2017 on Red Hat Enterprise Linux delivers superior results over the previous top non-clustered TPC-H@1000GB results for SQL Server 2016 Enterprise Edition on Windows.

These results included:

- 6 percent higher performance
- 5 percent lower price/performance

Source: Microsoft, Red Hat, and HPE Collaboration Delivers Choice & Value to Enterprise Customers

- **Memory-optimized table**. Memory-optimized tables are stored in active memory. These tables typically need to be frequently accessed for transaction-processing applications. The primary store for memory-optimized tables is main memory. So, unlike disk-based tables, data doesn't need to be read in from disk into memory buffers. To create a memory-optimized table, use the MEMORY_OPTIMIZED = ON clause. Note that memory-optimized tables can't be the target of a MERGE operation. Use INSERT, UPDATE, and DELETE statements instead.
- **Natively compiled stored procedure**. Natively compiled stored procedures are sets of T-SQL statements that can be called by OLTP to speed query execution and the processing of business logic. SQL Server supports natively compiled stored procedures that access memory-optimized tables, marking them with NATIVE_COMPILATION. The related T-SQL statements are compiled to machine code and stored as native DLLs, enabling faster data access and more efficient query execution than traditional T-SQL statements.

You can also create natively compiled stored procedures to access data in memory-optimized tables. Here's an example syntax:

```
CREATE PROCEDURE dbo.usp_add_kitchen
   @dept_id int,
   @kitchen_ count int NOT NULL

 WITH
   EXECUTE AS OWNER,SCHEMABINDING,
   NATIVE_COMPILATION
 AS
   BEGIN ATOMIC WITH (
     TRANSACTION ISOLATION LEVEL = SNAPSHOT,
     LANGUAGE = N'us_ english'
   )
   UPDATE dbo.Departments
   SET kitchen_count = ISNULL(kitchen_count, 0) +  @kitchen_count
   WHERE id = @dept_id
   END;
GO
```

Query Store

The SQL Server Query Store collects detailed performance information about queries, execution plans, and runtime statistics. Query Store isn't active by default, but you can enable it with ALTER DATABASE:

```
ALTER DATABASE AdventureWorks SET QUERY STORE = ON;
```

Typical use cases for the Query Store feature include:

- Discover and identify the most expensive queries related to CPU, I/O, memory, and so on.
- Track information about query regressions, comparing the execution plan generated by the current query engine to the older one. The Query Store helps you identify performance and fix regressions fast.
- Visualize the full history of query executions.

Automatic tuning and adaptive query processing

SQL Server uses adaptive query processing and automatic tuning to further increase database performance and address performance degradations, including:

- **Adaptive query processing**. With adaptive query processing, SQL Server adapts to customer workloads by optimizing its query plan based on the performance of the previous query. You can make workloads automatically eligible for adaptive query

processing by enabling compatibility level 140 for the database. To set this, use this T-SQL command:

```
ALTER DATABASE [WideWorldImporters] SET COMPATIBILITY LEVEL = 140;
```

- **Automatic tuning**. Using the rich telemetry of Query Store, automatic tuning maintains data query performance by detecting and automatically correcting performance issues. Automatic tuning features include automatic plan correction, which identifies problematic query execution plans and fixes SQL plan performance issues. You can enable automatic tuning using the following command:

```
ALTER DATABASE current
SET AUTOMATIC_TUNING ( FORCE_LAST_GOOD_PLAN = ON );
```

Troubleshooting performance issues

SQL Server includes a few tools to help troubleshoot performance issues. The Query Store provides insight on query plan choice and performance. Dynamic management views give you server state information for monitoring server instance health, diagnosing problems, and tuning performance. The Performance Dashboard helps you quickly identify performance bottlenecks. For more information on improving performance, go to Troubleshoot SQL Server on Linux.

Are high-availability technologies such as Always On Availability Groups available for SQL Server on Linux? If so, how does this work in the absence of important components like Windows Server Failover Clustering?

High-availability technologies are available for SQL Server on Linux using Pacemaker for the clustering solution. Pacemaker for Availability Groups and Failover Cluster Instances use a cluster manager to provide required functionality. However, these technologies are not currently supported in VMs on the cloud.

Implementing high availability

SQL Server enables various HADR scenarios—Always On Failover Cluster Instances (FCIs), Always On Availability Groups (AGs), and log shipping—that can help your organization achieve a wide range of availability service-level agreements (SLAs).

Common tasks occur across all features for availability configurations of SQL Server on Linux, including:

1. **Copy files**. Copying files from one server to another is an important task for HA on Linux. A common method is to use the command-line utility scp, which stands for "secure copy."
2. **Configure the firewall**. Linux distributions have a built-in firewall. All required ports must be enabled and opened. To find common ports needed for highly available SQL Server deployments on Linux, go to Configure the firewall.
3. **Provide business continuity**. The clustering mechanism currently supported by Microsoft for AGs and FCIs is Pacemaker with Corosync only. Pacemaker coordinates communications and resource management, while Corosync provides dedicated two-way communication between nodes. Integration with Pacemaker enables you to orchestrate the monitoring, failure detection, and automatic failover required for a comprehensive HA solution.
4. **Install SQL Server packages for high availability**. Under Linux, two packages should be installed with SQL Server: SQL Server Agent (mssql-server-agent) and the HA package (mssql-server-ha).

Always On Failover Cluster Instances

Always On FCIs are HADR tools that provide business continuity on an instance or server level. FCIs protect against server or instance failure due to networking, hardware, operating system, or software issues.

Below are several features to be aware of while setting up FCIs on Linux:

- **Clustering**. As mentioned earlier, the clustering layer is managed by Pacemaker. On Red Hat Enterprise Linux, it's provided by the RHEL High Availability Add-On. On SUSE Linux Enterprise Server, the necessary packages are provided by the SUSE Linux Enterprise High Availability Extension (HAE).
- **Number of instances and nodes**. Unless you're using containers, a Linux-based FCI can have only have one instance of SQL Server per Linux server.

- **IP address and hostname**. Each SQL Server instance needs its own IP address and hostname, which are used internally by Pacemaker to communicate with specific instances. Pacemaker setup involves a new virtual resource with its own IP address. You can connect applications to data without identifying cluster-internal resources.
- **Shared storage**. Some form of shared storage is required for FCIs on Linux or Windows Server. On Linux, you can use Internet Small Computer Systems Interface (iSCSI) and Network File System (NFS). On Windows Server, you can use iSCSI and Server Message Block (SMB). For configurations spanning multiple locations, you should sync data stored in one datacenter across locations. For default user data and log file locations, the system databases must always exist at /var/opt/mssql/data for all instances.
- **FCI resource group**. Create FCIs on Linux nodes in a resource group. To do this, use the pcs resource command. Once the FCI is online, you can issue normal SQL statements using SSMS or sqlcmd.

```
sudo pcs resource create <FCIResourceName> ocf:mssql:fci op defaults
timeout=60s --group <RGName>
```

- **Pacemaker failover**. When Pacemaker detects a failure, it switches the active node to a different, healthy cluster member and removes the failed node from the pool of available cluster members, a process known as fencing.

For configuring HA with Always On FCIs, you need to install and configure SQL Server on each node of the distribution and provide storage that all SQL nodes can access. Configure storage, present the storage to the cluster nodes, and then move the database files to the new storage. Also, make sure that Pacemaker packages are installed on each node with the FCI resource agent. With these nodes, you can create a cluster. Configure the cluster resources for SQL Server, File System, and virtual IP resources, and then push the configuration to the cluster. This way you can set up FCI for HA on Linux environments. For more details, go to Failover Cluster Instances – SQL Server on Linux.

High availability with Always On Availability Groups

Always On Availability Groups are an enterprise-level HA solution. This feature enables you to maximize availability for one or more databases. An availability group is one or more groups of databases that fail over to a backup together.

For Always On Availability Groups under Linux-based SQL Server installations, consider characteristics and configuration settings, including:

- **Configure settings for SQL Server Availability Groups on Linux**. When creating an availability group on Linux servers, make sure to enable availability groups on each Linux node with endpoints and certificates. Set the cluster type to **External** or **None**. External type means using Pacemaker with the availability group, while None means that there is no requirement to use Pacemaker. An External cluster type with Pacemaker helps you query the instances of SQL Server in the availability group and orchestrate failover to maintain high availability. A cluster type of None only supports manual failover from a primary to a secondary replica and is mostly targeted to the read-scale scenario. Use T-SQL (or the New Availability Group Wizard in SSMS on Windows) to create an availability group with the desired cluster type.

```
CREATE AVAILABILITY GROUP [ag1]
    WITH (DB_FAILOVER = ON, CLUSTER_TYPE = EXTERNAL)
    FOR REPLICA ON
sudo pcs resource create ag_cluster ocf:mssql:ag ag_name=ag --master
meta notify=true
            AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
            FAILOVER_MODE = EXTERNAL,
            SEEDING_MODE = AUTOMATIC
            ),
        N'<node2>'
         WITH (
            ENDPOINT_URL = N'tcp://<node2>:<5022>',
            AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
            FAILOVER_MODE = EXTERNAL,
            SEEDING_MODE = AUTOMATIC
                );
ALTER AVAILABILITY GROUP [ag1] GRANT CREATE ANY DATABASE;
```

  After creating an availability group on SQL Server, make sure to also create the corresponding resources in Pacemaker:

  To get started with Availability Groups in SQL Server on Linux, go to Create and configure an availability group for SQL Server on Linux.

- **Number of replicas and cluster nodes**. SQL Server 2017 Standard Edition supports two nodes in an availability group and one database per availability group, while the

Enterprise Edition can have up to nine nodes in an availability group. Configuration-only nodes do not count against these limits. Use a configuration-only replica if you want to configure a two-replica set with the ability to automatically fail over to another replica. Read more about replicas and cluster nodes.

- **Configuration-only replica and quorum**. To ensure continuous operation, make sure to check that Pacemaker is configured properly while verifying quorum and STONITH are implemented properly from a Pacemaker perspective. Plus, review any SQL Server requirements such as a configuration-only replica. For more information, see Configuration-only replica and quorum.
- **SQL Server resource agent for Pacemaker**. In SQL Server, there's now a sequence number option for sys.availability_groups. Through this option, Pacemaker can identify to what degrees secondary replicas are up to date with the primary replica. With each availability group configuration change, Pacemaker updates the sequence_number. Configuration changes may include failover, replica addition, or removal.

Is it possible to configure a read-scale replica in a Linux environment?

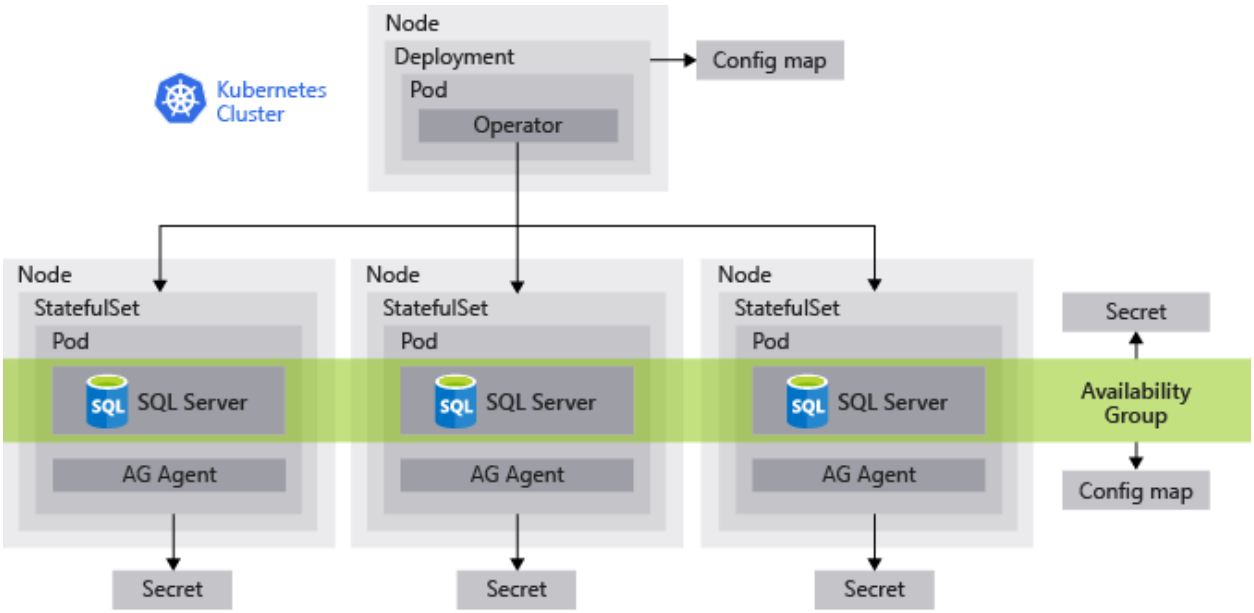
Yes, you can configure a SQL Server Always On Availability Group for read-scale workloads on Linux. An availability group with CLUSTER_TYPE = NONE can include replicas for or read-scale on Linux. Read-scale replicas can be used between Windows and Linux to replicate data one way.

Log shipping on Linux

Log shipping is available in both Windows and Linux. On Linux, SQL Server Agent jobs aren't part of the base installation of SQL Server. They are added with the mssql-server-agent package, which must be installed to use log shipping. SQL Server on Linux uses the Common Internet File System (CIFS) and the Samba networking protocol service to store the transaction log backups on a network file share. The SQL Server Agent on Linux periodically runs a stored procedure that forwards log backups to the secondary servers. As on Windows, recovering a database that's protected with log shipping is a manual operation.

Kubernetes support for SQL Server

SQL Server 2019 supports availability groups on containers in Kubernetes. Kubernetes can orchestrate containers running SQL Server instances to provide a highly available set of databases with SQL Server Always On Availability Groups. For instructions on deploying a container in Kubernetes, go to Deploy a SQL Server container in Kubernetes.



The image above shows an example of a four-node Kubernetes cluster. For more information, go to Always On availability groups for SQL Server containers.

Configure a SQL Server container in Kubernetes clusters for high availability

With persistent storage for high availability, Azure Kubernetes Service (AKS) provides a solution for using containers in a high-availability environment. If the SQL Server instance fails within a Kubernetes container (or pod), Kubernetes automatically recreates it in a new container that attaches to the same persistent storage. In the event of a Kubernetes node failure, AKS helps you be resilient. To learn more about deploying Kubernetes in a failover configuration, go to Deploy a SQL Server container in Kubernetes with Azure Kubernetes Services.

What's the built-in monitoring tool for SQL Server on Linux?

You can monitor the performance of your SQL Server on Linux environment through an open-source community solution like collectd, InfluxDB, or Grafana. Through system dynamic management views, you can collect various types of information about SQL Server, including Linux process information.

Is it possible to use Activity Monitor for a SQL Server instance running on Linux?

You can use SQL Server Management Studio (SSMS) on Windows to connect SQL Server instances running on Linux remotely for Activity Monitoring. SSMS displays information about SQL Server processes, gathering data continuously over time to track performance trends of a SQL Server instance on Linux.

Monitoring SQL Server

Various tools and services are available with SQL Server on Linux to monitor activity and performance.

InfluxDB, collectd, and Grafana

InfluxDB provides performance and flexibility, collectd gives you a lightweight tool for gathering system performance information, and Grafana furnishes a rich and interactive tool for visualizing the data. collectd continuously runs in a container on your SQL Server on Linux environment, pushing metrics to InfluxDB. You can then visualize data via the Grafana dashboard, which reads data from InfluxDB. To learn more, go to Monitor your SQL Server on Linux performance using collectd, InfluxDB, and Grafana.

Dynamic Management Views

SQL Server Dynamic Management Views (DMVs) are built-in views and functions that return information about the server. You can use DMVs to monitor the health and performance of your SQL Server instance. SQL Server offers a set of DMVs specifically designed to access catalog metadata for SQL Server on Linux, including CPU information, memory, and SQL Server threads. For more information, go to System Dynamic Management Views.

Live Query statistics in SQL Server Management Studio

Live Query statistics in SSMS on Windows that connect to SQL Server instances running on Linux display real-time statistics for query execution. This is a useful tool for performance monitoring and debugging query performance issues. The live query plan shows overall query progress and

operator-level run-time execution statistics such as the number of rows produced, elapsed time, and operator progress. For more information, go to SQL Server monitoring partners.

Managing SQL Server

To manage and automate SQL Server administrative tasks, a number of graphical user interface (GUI) tools and command-line tools are available with Linux.

Is there a graphical tool to manage the SQL Server engine on Linux?

You can use SSMS, which runs on Windows only but can manage SQL Server running on Linux. You can also take advantage of Azure Data Studio, a new tool that runs on macOS and Linux in addition to Windows.

Graphical tools

Azure Data Studio (formerly SQL Operations Studio)

A new cross-platform tool for SQL Server built on VS Code, Azure Data Studio runs on Windows, macOS, and Linux. Azure Data Studio offers a modern, keyboard-focused T-SQL coding experience with built-in features such as multiple tab windows, a rich T-SQL editor, IntelliSense (code completion), keyword completion, code snippets, code navigation, and source control integration (Git). With on-demand T-SQL queries, you can view and save results in common formats such as text, JSON, or Microsoft Excel. Azure Data Studio helps you organize your database connections and browse database objects. You can download the latest GA build for Linux by using installers or the tar.gz archive on the Azure Data Studio download page.

Here's an example of. rpm installation:

```
cd ~
yum install ./Downloads/azuredatastudio-linux-<version string>.rpm

azuredatastudio
```

SQL Server Management Studio

While SSMS isn't available on Linux, you can still use it from Windows to manage SQL Server instances on Linux OS remotely. SSMS can help you back up, restore, view, and edit databases, plus create and edit T-SQL queries, scripts, and database objects. SSMS is continually updated and optimized. To download and install the latest version, go to Download SQL Server Management Studio.

VS Code and SQL Server extensions

The mssql extension for VS Code enables you to connect to SQL Server, query with T-SQL statements, and view the results. It offers T-SQL language features like T-SQL IntelliSense, syntax highlighting, linting, code navigation, and code snippets. You can establish a connection to the SQL Server Integration Services (SSIS) Catalog using an SSIS extension, and then use T-SQL statements to deploy an SSIS project to the SSIS Catalog. Once you've installed VS Code, you can also install mssql extensions and connect to any SQL Server instance. VS Code is free and available on Linux, macOS, and Windows, with a rich ecosystem of extensions found in the VS Code Extension Marketplace. To download VS Code, see Download Visual Studio Code.

How can I see my execution plans on Linux?

You can view execution plans (still a preview feature) using Azure Data Studio. This tool is similar to SSMS, but is available on Windows, Linux, and macOS.

System Center Management Pack

With the latest Microsoft System Center Management Pack for SQL Server 2017, you can discover and monitor SQL Server 2017 database engines, databases, and other related components on Windows and Linux deployments. Also, with support for agentless monitoring on Linux, you can shift monitoring workloads to management servers included in the SQL Server Monitoring Pool. This enables you to remove System Center Operations Manager (SCOM) and data-processing overhead from the SQL Server host. All monitoring workflows have predefined thresholds and complementary knowledge-base articles. To download the Management Pack, go to Microsoft System Center Management Pack for SQL Server 2017.

Command-line tools

The mssql-tools package contains the sqlcmd and bcp utility for connecting to SQL Server. Find the mssql-tools package for your platform: Red Hat Enterprise Linux, Ubuntu, SUSE Linux Enterprise Server, or Docker. You can also install mssql tools with the unixODBC developer package, as shown in this example:

```
sudo yum install mssql-tools unixODBC-devel
```

sqlcmd

The sqlcmd command-line query utility helps you execute ad hoc T-SQL statements, system procedures, and script files at the shell prompt. For connecting the SQL Server with sqlcmd, use this SQL Server authentication:

```
sqlcmd -S localhost -U SA -P '<YourPassword>'
```

bcp

Available on Linux, the bcp command-line utility lets you efficiently bulk-copy rows into SQL Server tables, or export data out of tables into data files in a user-specified format. Follow this example query to connect to SQL Server locally and import the data from the data file into the table in the database:

```
bcp TestEmployees in ~/test_data.txt -S localhost -U sa -P <your_password> -d
BcpSampleDB -c -t  ','
```

mssql-cli command-line query tool

For querying SQL Server, the mssql-cli command-line query tool is an interactive command-line open-source tool that works across platforms—Windows, macOS, or Linux. It includes modern features such as auto-completion and syntax highlighting. To download the tool directly, go to the GitHub page. Based on Python, mssql-cli uses a preferred installer program (pip) to install it as shown below. Once installed, launch the tool from the shell prompt using mssql-cli --help.

```
pip install mssql-cli
```

What is the difference between sqlcmd and mssql-cli, if both are command-line tools?

mssql-cli is a Python pip package and contains new and enhanced features such as T-SQL IntelliSense, multi-lines queries, and syntax highlighting.

The key differentiator is that mssql-cli has a better interactive experience for editing and running T-SQL queries in the command line. Imagine working in a jump-box scenario where a user can't access a GUI tool. mssql-cli returns readable output compared to sqlcmd, along with many configuration options. However, mssql-cli doesn't provide scripting support like -i, so sqlcmd should be used for non-interactive scenarios. Currently a preview product, mssql-cli will also include this support eventually.

mssql-scripter command-line utility

Use the mssql-scripter command-line utility on Linux to generate data definition language (DDL) and data manipulation language (DML) T-SQL scripts for database objects in SQL Server. You can save the generated T-SQL script to a .sql file, or pipe it to Linux utilities such as sed, awk, and grep for further transformations. An open-source tool, mssql-scripter is built using Python. You can get it from the Python website.

```
pip install mssql-scripter
```

SQL Server PowerShell

SQL Server PowerShell lets you automate server administration and application deployment. PowerShell support for SQL Server is currently available on Windows, so you can use it when you have a Windows machine that can connect to a remote SQL Server instance on Linux. The PowerShell language supports more complex logic than T-SQL scripts, giving SQL Server administrators the ability to build robust administration scripts. For more information, go to Use PowerShell on Windows to Manage SQL Server on Linux.

Database File System

To explore information about your database using native Bash commands, you can use the Database File System (DBFS). DBFS works through Filesystem in Userspace (FUSE) to mount SQL Server dynamic management views (DMVs) as a virtual file system. This gives you the ability to view the contents of the virtual files in the mounted virtual directory, seeing the same data you would see if you had run a SQL query. DBFS exposes live data from SQL Server DMVs as virtual files in a virtual directory on Linux. For more information on DBFS, go to the GitHub page.

Migration and upgrade

Microsoft offers migration tools to manage the transition to SQL Server for many popular relational database management solutions. So, for migrating your existing data and databases to SQL Server on Linux, you have options.

Migrate from other database servers

You can migrate databases from other database systems to SQL Server on Linux. This includes Microsoft Access, DB2, MySQL, Oracle, and SAP ASE (formerly SAP Sybase ASE).

SQL Server Migration Assistant

With SQL Server Migration Assistant (SSMA), you can assess your source database, convert the source database schemas to SQL Server schemas and migrate them, and then migrate data to the target server. Guiding you through the process with a graphical user interface, SSMA helps you manage migration and uploads of your data, highlighting any issues encountered during the conversion.

Data Migration Assistant

Compatibility issues can affect database functionality before you migrate to SQL Server. Data Migration Assistant (DMA) helps you detect these issues and offer suggestions on how to address them before proceeding. It recommends performance and reliability improvements for your target environment, including newer features that are available through a version upgrade.

With the migration workflow, you get support for migrating database schemas, data and users, server roles, and SQL Server logins.

Database Experimentation Assistant

Database Experimentation Assistant (DEA) helps you evaluate which version of SQL Server you should target for upgrade—from previous SQL Server versions starting with 2005 and above. Plus, the tool guides you through a successful upgrade with analysis reports such as query metrics for compatibility errors, degraded queries and query plans, and other workload comparison reports. You can set up automated workload captures and replay of production databases, perform statistical analysis on traces collected using both old and new instances, and visualize collected data. To install the DEA tool, go to the Microsoft Download Center and run 'DatabaseExperimentationAssistant.exe.'

Migrate from SQL Server on Windows

You can migrate SQL Server databases on Windows to SQL Server on Linux using various techniques.

Import and export using SQL Server Management Studio or SqlPackage.exe

Using SSMS and SqlPackage.exe, you can export a database from Windows and import it to SQL Server on Linux. SSMS and SqlPackage.exe are Windows applications, so you need a Windows machine to connect to a remote SQL Server instance on Linux. SSMS enables you to use export and import database features, while the SqlPackage.exe command-line utility automates the export and import of BACPAC files. The SQLPackage utility ships with the latest versions of SQL Server Management Studio and SQL Server Data Tools for Visual Studio, or you can download the latest version directly from the Microsoft Download Center.

Migrate a SQL Server database from Windows to Linux using backup and restore

Using SQL Server backup and restore functionality, you can migrate a SQL Server database on Windows to SQL Server on Linux. First, you need to create a backup file from a Windows machine, and transfer and copy it to the appropriate directory on the target Linux machine. Then use the RESTORE DATABASE T-SQL command:

```
RESTORE DATABASE YourDB
FROM DISK = '/var/opt/mssql/backup/YourDB.bak'
WITH MOVE 'YourDB' TO '/var/opt/mssql/data/YourDB.mdf',
MOVE 'YourDB_Log' TO '/var/opt/mssql/data/YourDB_Log.ldf';
GO
```

Migrate structured data

Additional techniques enable you to import raw data exported from other databases or data sources.

Bulk copy data with bcp

The bulk copy program (bcp) command-line utility helps you copy data in bulk between an instance of SQL Server 2017 on Linux and a data file in a user-specified format. For more information, go to bcp Utility.

Extract, transform, and load data for SQL Server on Linux with SSIS

SQL Server Integration Services (SSIS) assists you with extracting data from multiple sources and formats, transforming and cleaning data, and loading data into multiple destinations. To run SSIS packages on a Linux machine, first install SQL Server Integration Services, as SSIS is not included in the SQL Server 2017 on Linux installation. Run the SSIS package on a Linux machine and use ODBC connections to do the extract, transform, and load (ETL). You can use Linux system scheduling tools such as cron to schedule the package.

Migrate to Linux Docker container

Restore a SQL Server database in a Linux Docker container

You can move and restore a SQL Server backup file into a SQL Server 2017 Linux container. To do this, pull and run the container image from Docker Hub, and then copy a backup file into the required folder of the container. Call the RESTORE DATABASE command to restore the database inside the container. See the below sample code to restore the database:

```
sudo docker exec -it sql1 /opt/mssql-tools/bin/sqlcmd \
-S localhost -U SA -P '<YourNewStrong!Passw0rd>' \
-Q 'RESTORE DATABASE WideWorldImporters FROM DISK =
"/var/opt/mssql/backup/wwi.bak" WITH MOVE "WWI_Primary" TO
 "/var/opt/mssql/data/WideWorldImporters.mdf", MOVE "WWI_UserData" TO
 "/var/opt/mssql/data/WideWorldImporters_userdata.ndf", MOVE "WWI_Log" TO
 "/var/opt/mssql/data/WideWorldImporters.ldf", MOVE "WWI_InMemory_Data_1" TO
 "/var/opt/mssql/data/WideWorldImporters_InMemory_Data_1"'
```

Conclusion

SQL Server 2017 on Linux is an illustration of the Microsoft commitment to supporting flexible, reliable data management solutions. Install SQL Server on the platform of your choice, including Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Ubuntu, Docker, or SQL Server containers by using OpenShift and Kubernetes. You can also provision a Linux SQL Server virtual machine on Azure. Get the enterprise-grade technologies already offered with SQL Server, such as Active Directory authentication, Row-Level Security, Transparent Data Encryption, In-Memory OLTP, In-Memory Clustered Columnstore, Availability Groups, and Automatic Tuning. All of these capabilities are now available on the Linux platform. Plus, with support for open-source and cross-platform tools, you can easily and effectively migrate, monitor, and manage your SQL Server databases using your existing methods and processes.

Resources

Get started with SQL Server on Linux

Explore SQL Server on Linux documentation

Read the SQL Server 2017 licensing datasheet

Receive guidance for installing SQL Server 2017 on Linux

Find answers to frequently asked questions about SQL Server on Linux

Watch a webinar on frequently asked questions about SQL Server on Linux