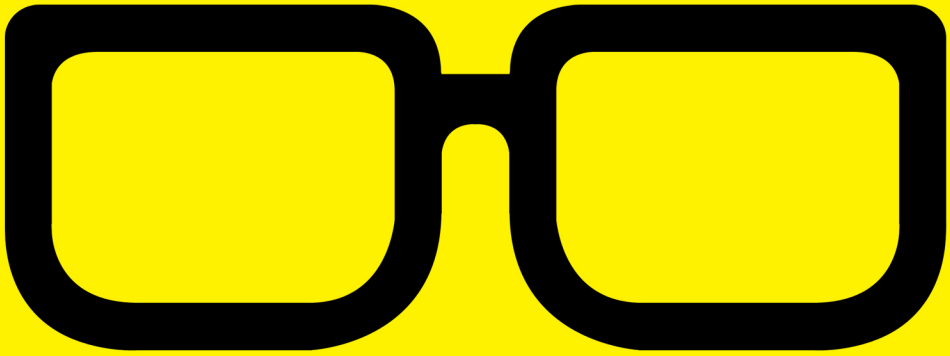
SPONSORED BY

GEEK GUIDE



SQL Server on Linux

LINUX JOURNAL

Table of Contents

REUVEN M. LERNER offers training in Python data science to companies around the world. He has written two programming ebooks (*Practice Makes Python* and *Practice Makes Regexp*), publishes the free weekly "Better developers" newsletter (at http://lerner.co.il/newsletter), and runs the "Weekly Python Exercise" subscription service at http://WeeklyPythonExercise.com. Reuven tweets at @reuvenmlerner and lives in Modi'in, Israel, with his wife and three children.

GEEK GUIDES:
Mission-critical information for the most technical people on the planet.

About the Sponsor

SUSE, a Micro Focus company, provides and supports enterprise-grade Linux and open-source solutions with exceptional service, value and flexibility. With partners and communities, we innovate, adapt and deliver secure Linux, cloud infrastructure and storage software to create solutions for mixed enterprise IT environments. We help customers harness the benefits and power of an open enterprise that can empower their possibilities.

SQL Server on Linux

REUVEN M. LERNER

Introduction

Let's get this out of the way: I'm still surprised to be writing this Geek Guide. It's no secret that Microsoft was long perceived as the enemy of open-source software. Indeed, it's not unusual for people in open-source forums, when faced with a question from someone using a Microsoft product, to rudely refuse to help.

But, even the most skeptical open-source fan must admit that Microsoft has changed during the last few years. It has released many developer tools and libraries as open source, and it has started to include Linux as a supported platform for some of its software. Microsoft is embracing open source in different ways—sometimes as a way to license its software and sometimes as an additional platform on which to run its commercial products.

Does this sound like a big change? Yes, it does. And, it's one that takes some getting used to, especially if you have been using open source for a long time, as I have. I probably opened more browser tabs on Microsoft.com while preparing this Geek Guide than in the entire preceding year.

And yet, once you get over the surprise, it makes sense. Microsoft, like other commercial companies, can't ignore the success of Linux. With the large and growing number of companies using Linux for their server infrastructure, Microsoft would be foolish to ignore the business potential of selling to such companies. Indeed, Microsoft's Azure hosting service offers Linux as an option, and it's a very popular one—one third of the VMs running on Azure are using Linux.

A recent addition to Microsoft's offerings for Linux is SQL Server. Microsoft SQL Server is an offering that includes not just a relational database server, but a suite of applications and tools for analysis, integration and machine learning. I certainly have many consulting clients who have used it for many years. But until now, running SQL Server meant using Windows. The large number of organizations that have standardized on Linux-based servers thus were non-starters for Microsoft's marketing and sales departments.

The bottom line then is that if you're running a Linux-based server, you now have another option to consider for your database needs: Microsoft SQL Server.

This raises a number of questions though. What is SQL Server? Why would you want to use it, as opposed to any other database? How do you use it, and what can you do with it?

In this Geek Guide, I try to answer those questions, while also exploring some of the things you can do with it. Even if you decide not to use SQL Server, I do think it's worthwhile for open-source advocates to understand what options are available to them, and what features are available in some of the most advanced commercial databases.

What Is SQL Server?

SQL Server has been around since 1989, with frequent releases that are generally identified by year (for example, SQL Server 2012, SQL Server 2014 and SQL Server 2016). SQL Server works on a client-server model and (as its name indicates) uses SQL as a query language.

SQL Server actually uses a version of SQL known as T-SQL (Transact-SQL), developed years ago by Sybase. T-SQL includes not only the standard SQL queries, but it also includes provisions for procedures, conditional execution, exception handling and local variable definitions. T-SQL thus allows you to perform certain tasks within a database query that in another database might require the use of a separate programming language or procedural add-on. As someone who has used PostgreSQL for many years, T-SQL reminds me somewhat of Pl/PgSQL.

SQL Server and its associated tools provide you with the ability to analyze data, integrate with external systems and produce reports based on queries. Open-source developers are used to writing such programs themselves or piecing together solutions using various open-source projects; Microsoft sees its software solutions as various parts of an integrated whole, and as part of a larger suite of solutions.

In addition to the standard set of SQL data types, SQL Server provides several of its own, including hierarchyid (for tracking hierarchical information), xml (for XML data) and geographical information. All told, the data types provided by SQL Server are similar to those in other relational databases I've used—including such useful ones as "datetime" and "datetimeoffset", for tracking not only points in time but also the distance between them.

Where things get more interesting is in the actual storage of data, in tables and indexes. SQL Server, like other relational databases, is primarily row-based. However, there has been increasing interest in columnar databases in the past few years, which can save time and space for particular types of data and queries.

In SQL Server 2016, columnar storage is achieved via the new "columnstore" clustered index. Like other clustered index solutions, this one reorders the data into a format that is more efficient to search and retrieve. But, this one keeps the data in columns rather than rows, which is perfect for certain types of analysis. Moreover, once you cluster the data using columnstore, new and updated rows are kept in a "deltastore", buffered until there is enough new data to justify the (potentially heavy load of) an actual insert into, and then reordering, of the columnstore.

SQL Server also includes R, a programming language designed for statistical analysis with a very large number of third-party libraries. This means that if you're planning to crunch numbers stored in your database, perform statistical analysis or even create a machine-learning model, you might well be able to do so within the

Microsoft's rollout of SQL Server for Linux is a ground-breaking event, in that it's making one of Microsoft's most famous and popular commercial offerings available on the most popular open-source operating system.

database, rather than export the data to a client program and analyze it there. Indeed, Microsoft is making a push to make machine learning readily available, with Python (arguably the most popular language for data science) coming in SQL Server 2017.

Even before SQL Server was made available on Linux, many developers wrote applications that connected to SQL Server from Linux. That is, your client would run on Linux, while the server ran on Windows. Now, both the client and the server can sit on Linux-based systems.

For example, if you want to write a Python program that talks to SQL Server, you can use the "pymssql" package from PyPI. You first will need to install the FreeTDS libraries, which implement the underlying TDS network protocol used by SQL Server. Once you have done so, however, there should be no difference between your program connecting to SQL Server on Windows and SQL Server on Linux.

And that, of course, is the point. Microsoft's rollout of SQL Server for Linux is a ground-breaking event, in that

it's making one of Microsoft's most famous and popular commercial offerings available on the most popular open-source operating system. At the same time, it's something of a non-event, in that Microsoft is trying hard to make the Linux version of SQL Server as similar as possible to the Windows version. In the Microsoft dream world, SQL Server will be identical on both platforms, allowing you to choose whichever platform is more appropriate for your organization.

SQL Server vs. Other Databases

If you're a typical Linux user, you probably have used open-source databases, such as MySQL and PostgreSQL. What advantages does SQL Server bring to the table?

A main one, I would argue, is also the most boring: it's what people are used to. I have worked with many companies through the years whose DBAs have many years of experience working with SQL Server. The company wants to switch over to Linux for a variety of reasons, but the DBA is nervous about switching to an unfamiliar database. Besides, software people know very well that switching database vendors can be a time-consuming and difficult event, even under the best of circumstances. If you can stick with the same product on a different OS, you can reduce the risk to your business.

Let's assume though that you're working on a totally new project and can choose a database. What would lead you to consider SQL Server? A few features caught my eye that might be particularly appropriate for your data and, thus, make it a contender.

Another reason to consider SQL Server is its relative ease of management, especially when compared with other commercial offerings, such as Oracle.

First and foremost, SQL Server consistently scores very high on performance tests. Benchmarking in the database world is a sensitive subject, in no small part because the stakes are so high—but also because establishing the data set to test, and the queries to run on it, aren't always obvious. Microsoft has performed extremely well in a number of benchmarks; even if it's not always the top contender, it's always very near the top.

Another reason to consider SQL Server is its relative ease of management, especially when compared with other commercial offerings, such as Oracle. To someone used to working with MySQL and PostgreSQL, the notion that a GUI might make the database easier to administer might sound laughable. But there are very many DBAs out there who prefer a GUI, and SQL Server gives it to them.

Well, sort of—SQL Server itself, for Linux, doesn't include administrative tools. This means you'll need to use the CLI (unlikely), the GUI from a remote machine, a plugin for Visual Studio (which now works on Linux) or a third-party solution. Microsoft says that it's working on porting these administrative tools to Linux as well.

Although data, table and index types certainly are of interest when you're looking into a database, one of the key considerations is performance. SQL Server has a reputation for being fast, and third-party reports (such as Gartner) indicate that it's consistently one of the fastest databases out there. This doesn't mean it's always going to be the fastest with all data types and all loads, but it does mean that it's almost certainly going to be fast enough for whatever data you're working with, even if that's a large amount.

If you do need to squeeze more efficiency out of your database, SQL Server allows you to partition data across multiple tables within the same database, or to replicate data across multiple servers for high availability and reduced risk. It also provides, in the Enterprise version, something known as Peer-to-Peer Transactional Replication. This is similar to multi-master replication, already available in MySQL and on the horizon in PostgreSQL.

SQL Server also now offers something called "in-memory OLTP", which seems similar to MySQL's in-memory tables (that is, the MEMORY storage engine). Although this potentially can use a huge amount of memory, there's clearly no contest between the performance that you can get from an in-memory table, as opposed to that of one that resides on disk, no matter how fast the disk is.

Microsoft also has added features in the area of "high availability", ensuring that your server remains accessible even when one or more servers are unavailable or offline. Microsoft claims that the HA solution included with SQL Server handles all of the standard use cases you would

Finally, SQL Server offers the ability to encrypt the database-related data on your filesystem, as well as to encrypt backups.

expect, including failover to local or remote servers.

Finally, Microsoft is touting the security features associated with SQL Server. It's important to point out that SQL Server, like many other databases in the Linux world, doesn't run as the "root" user. Rather, it is designed to run under another user. This is an important security consideration; you don't want your database server to have complete access to your entire system.

SQL Server's security resembles that of PostgreSQL in that you can assign particular permissions to any number of users. Thus, you can say that users A, B and C have read/write permission on inventory, read permission on budgeting and no permissions on salary. You also can assign row-level permissions; based on the results of entering a query, you then can indicate which users may take which actions. You similarly can indicate that the results from certain queries, or certain rows, should be "masked" from users of certain roles.

Finally, SQL Server offers the ability to encrypt the database-related data on your filesystem, as well as to encrypt backups. Given the importance and sensitivity of data to most organizations, these security features can

decrease the risk that an unauthorized outsider can read your data. The notion of encrypting backups is perhaps not new to others, but I hadn't heard about it before, and it strikes me as a particularly good idea.

SQL Server on Linux vs. on Windows

Is SQL Server on Linux the same product as SQL Server on Windows? The answer, of course, is both "yes" and "no". On the one hand, Microsoft has worked hard to port SQL Server to Linux, partly by creating a software compatibility layer. This allowed Microsoft to keep much of the codebase the same for both platforms. Even if this fact doesn't directly affect end users, I have to believe it was done both to ease the rollout and also to ensure that as much as possible will be common on both platforms.

And indeed, this would seem to be the case. The SQL Server product itself works the same as it does on Windows. Whether it's data types, indexing, T-SQL procedures or even command-line programs like `bcp` ("bulk copy"), SQL Server on Linux appears fairly complete. According to the Microsoft representatives with whom I spoke, some integration services currently are lacking right now, and are coming in 2017. But, the core database works the same, which means you should (in theory) be able to unplug a Windows-based SQL Server instance and replace it with a new Linux-based SQL Server instance. Indeed, Microsoft says that so long as you're using features that exist in both versions, such a switcheroo is supported and should work.

Another difference between the Windows and Linux

versions of SQL Server has to do with installation. Microsoft provides versions of SQL Server that work on the latest versions of Red Hat, SUSE and Ubuntu distributions. Moreover, installation is done using the same tools you would use to install anything else—yum, zypper or apt-get. Of course, you'll need to add Microsoft's repository before you can download and install it. And although you can freely download SQL Server for now, this is a commercial product, which means at some point, you'll also need to purchase a license.

For this Geek Guide, Microsoft and SUSE provided me with an Azure cloud server running SUSE Linux Enterprise Server. At first, I was surprised that they hadn't also installed SQL Server—after all, I imagined that the installation would be long and difficult. But Microsoft has provided extremely detailed installation instructions, starting with adding the repository to your package manager's list and then indicating which packages you'll need to install.

There is also support for Docker-based installations. I didn't try it, but if your organization is using Docker, it's possible that this will be a better option.

After adding the Microsoft repository to your system, you'll need to install the "mssql-server" package, as well as the "mssql-tools" and "unixodbc-dev" packages for command-line administration and connection libraries. You'll also likely want to install "mssql-server-agent" (for scheduled jobs) and "mssql-server-fts" (for full-text search).

On SUSE Linux, I installed SQL Server by first adding the Microsoft repository information (and GPG keys) to my

zypper configuration:

```
sudo zypper addrepo -fc https://packages.microsoft.com/config/
➥sles/12/mssql-server.repo
sudo zypper --gpg-auto-import-keys refresh

sudo zypper addrepo -fc https://packages.microsoft.com/config/
➥sles/12/prod.repo
sudo zypper --gpg-auto-import-keys refresh
```

Note that there are two separate repositories and two separate sets of keys. The first is for the server itself, and the second is for the tools associated with it.

Once the Microsoft repositories were added, I could install the server:

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

Following that, I installed the tools for working with the software:

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

I then also installed the full-text search system, as well as the scheduling system, which SQL Server requires:

```
sudo zypper install mssql-server-agent
sudo zypper install mssql-server-fts
```

These commands will install software into several

different directories:

- /opt/mssql is where the main SQL Server software resides. You probably won't run things directly from here.

- /var/opt/mssql is where the user files, such as database files, log files and configuration files are stowed.

- /opt/mssql-tools is where two command-line tools, `sqlcmd` and `bcp` reside. You almost certainly will want to add /opt/mssql-tools/bin to your `PATH`, so that you can run `sqlcmd` on your own.

- /opt/microsoft contains libraries for ODBC. I largely ignored this directory in my installation.

Once you have installed these packages—or at the very least, the main server package—you will need to configure the server. To be honest, configuring the server involves little more than running `mssql-conf setup` and choosing an administrator password. The password must fulfill a lot of security requirements, such as length and diversity of characters. I found these requirements (which were specified in the online documentation, but not in the program's prompt) to be a bit annoying, but understandable.

Once I had set up the system, I was able to use the `sqlcmd` program to create a new database and

then perform some queries. I first connected to the server using:

```
sqlcmd  -S localhost -U SA -P mypw
```

Notice that I'm using the `-S` argument to indicate the hostname (that is, localhost), the `-U` argument to indicate the user name (`SA`, the user name for the "system administrator"), and the `-P` argument with a password. Once I do this, I'm put into `sqlcmd`, whose prompt looks like this:

```
1>
```

Yes, that's right—`sqlcmd` presents you with a prompt, and then it's up to you to decide what to do with it. Perhaps the first thing to do is create a new database:

```
1> create database reuven
2> go
```

Unlike many other command-line database clients I've used, `sqlcmd` doesn't use a semicolon (;) to indicate when commands end. Rather, you say "go" on a line by itself. If things succeed, you typically don't get any output. If they fail, you'll get an error message—for example:

```
1> create database lj
2> go
1> create database lj
2> go
```

```
Msg 1801, Level 16, State 3, Server sles12mssql, Line 1
Database 'lj' already exists. Choose a different database name.
```

Once I've created a database, I can connect to it from the command line by adding the `-d` parameter (for "database") to the `sqlcmd` command:

```
sqlcmd  -S localhost -U SA -P mypw -d reuven
```

Now I can create tables and indexes, insert data and perform queries. Basic readline functionality is there, such as moving to the start and end of the line with C-a and C-e, but I found that more advanced readline facilities, such as moving forward and backward by word (M-b and M-f), as well as searching backward in the command history (C-r) didn't exist. Whether this is because it has yet to be implemented, or because the assumption is that most people will not want to work via sqlcmd, isn't clear.

Cloud Servers

Given how easily you can install SQL Server on a system, this raises the question of whether you would want to do so. Cloud servers are extremely popular, for a variety of reasons. You don't need to spec and purchase hardware; you pay only for the resources that you need and consume, and you can scale up easily and quickly. Moreover, the operating system installation and maintenance are often not your concern, either because it's taken care of by the cloud provider or because your machine's configuration cannot easily be changed.

So, should you install SQL Server on a cloud server or on your own dedicated (or leased) hardware? Or run Azure SQL Database as a cloud-native service? That's not an easy question to answer. I would say it depends on your use case and on your company's IT infrastructure and policies. Microsoft certainly has gone out of its way to ensure that SQL Server can be installed and run easily on cloud servers—first and foremost on its Azure hosting system. While Microsoft has provided instructions for installation on a number of distributions, it specifically has partnered with SUSE for this purpose, which may well mean better support for enterprise use cases from the two companies.

A major advantage of using cloud servers is the ability to scale up over time. You can add another server, install another copy of SQL Server and start running right away. You also can set up high-availability servers to ensure that your system is always responsive and available. Indeed, if you're using a cloud provider with multiple geographic locations, you can set up your HA systems to be in different zones, so that even if one of those zones go down, you will be in good shape.

Learning More

Although SQL Server's availability on Linux doesn't indicate anything about Linux's technical maturity, which has been evident for years, it does show something about Linux's commercial and business maturity. We are now, rather surprisingly, in the position of having Microsoft porting one of its best known, most powerful and most popular products to Linux.

A database is a sophisticated piece of software, and using it requires more than just installation and playing around—particularly if you care about performance as well as taking advantage of everything it has to offer. For this reason, you likely will want to take advantage of the many learning resources that Microsoft has made available.

The main website for SQL Server has an entire section for using it on Linux: https://docs.microsoft.com/en-us/sql/linux. This documentation includes not only detailed instructions for installing SQL Server on a variety of Linux distributions, but also for installing and working with the various tools. It includes full documentation for SQL Server, including all of the features that I mention here. I experimented with several of those features and found the documentation to be quite complete, with many examples and explanations.

Microsoft's documentation pages, like those for many of our favorite open-source projects, include space for comments and questions at the end of each page. Thus, it's not unusual to find useful nuggets of information in the comment section, both from people making suggestions and from Microsoft employees who offer support. The responses that I saw from Microsoft came very soon after they were asked, demonstrating Microsoft's interest in promoting Linux as a fully-fledged and supported platform.

It's important to note that even—or especially—if you're an experienced user of Linux and an open-source database, you will need to spend some time to learn to think in a new way. Microsoft has its own culture, and the way things are done under SQL Server—although different from

those in the Linux and open-source worlds—is established and used by many people.

Moreover, there are many websites, forums, books and blogs about SQL Server. So while the documentation and support for Linux-specific issues are relatively new, there is no shortage of information. Documentation and forums are available at https://docs.microsoft.com/en-us/sql.

Conclusion

Let's be honest. Many Linux users will use only open-source databases. And for many of them, that's an excellent fit. But many organizations, especially those that have used SQL Server on Windows but are moving to Linux, undoubtedly are going to be happy to have their favorite database available there. Moreover, if you're comfortable with a commercial database, SQL Server offers no small number of features that should pique your interest—whether it's performance, security or high availability.

In short, open-source advocates should welcome SQL Server onto the Linux platform and see it as additional proof that open source is alive and well, and that commercial software companies and open-source users can work for mutual benefit. If you're not wedded to a particular database product and are interested in high scalability and flexibility—not to mention compatibility with the Microsoft ecosystem of tools—you should take a look at SQL Server.■