SQL Server Installation on Linux

In addition to Windows platforms, SQL Server is supported on Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), and Ubuntu. It is also supported as a Docker image, which can run on Docker Engine on Linux or Docker for Windows/Mac. Specifically, this article will focus on the necessary steps required to install SQL Server on RHEL, SLES, and Ubuntu platforms.

Prerequisites

RHEL:

To install SQL Server on Red Hat Enterprise Linux, you must be running operating system versions RHEL versions 7.7 - 7.9 or RHEL versions 8.0 - 8.5, with a minimum of 2GB of memory. Additionally, underlying file systems must be either XFS or EXT4, and at least 6 GB of free disk space is required.

RHEL version 8 does not come preinstalled with python2, which is a requirement of SQL Server. In this case, it needs to be installed before the SQL Server installation. As such, the following Bash Linux shell commands need to be executed, and verification that python2 is selected as the interpreter needs to be made:

```
sudo alternatives --config python
# If not configured, install python2 and openssl10 using the following commands:
sudo yum install python2
sudo yum install compat-openssl10
# Configure python2 as the default interpreter using this command:
sudo alternatives --config python
```

SLES:

To install SQL Server on SUSE Linux Enterprise Server, you must be running operating system version v15 with a minimum of 2GB of memory. Additionally, underlying file systems must be either XFS or EXT4, and at least 6 GB of free disk space is required.

Ubuntu:

To install SQL Server on an Ubuntu platform, you must be running operating system versions 16.04, 18.04, or 20.04 with a minimum of 2GB of memory. Additionally, underlying file systems must be either XFS or EXT4, and at least 6 GB of free disk space is required.

Installation of SQL Server

RHEL:

To install SQL Server on RHEL, the following commands need to be executed from the Bash Linux shell to install the **mssql-server** package:

First, download the SQL Server 2019 Red Hat repository configuration file:

For RHEL version 7:

sudo curl -o /etc/yum.repos.d/mssql-server.repo
https://packages.microsoft.com/config/rhel/7/mssql-server-2019.repo

For RHEL version 8:

sudo curl -o /etc/yum.repos.d/mssql-server.repo
https://packages.microsoft.com/config/rhel/8/mssql-server-2019.repo

Next, run the following command to install SQL Server:

sudo yum install -y mssql-server

Next, after the SQL Server package install completes, run **mssql-conf setup** and follow the prompts to set the SA account password and choose the appropriate edition that you are running:

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

.
Once configuration is complete, verify that the SQL Server service is running by issuing the following command:

systemctl status mssql-server

Finally, if your database server installation requires remote server access, open the SQL Server port on the firewall on RHEL. The default SQL Server port is TCP 1433 (for security reasons, you may have changed this default port number in an earlier step). If you are running **FirewallD** for your firewall, you can use the following commands:

sudo firewall-cmd --zone=public --add-port=1433/tcp --permanent
sudo firewall-cmd --reload

SLES:

To install SQL Server on SLES, the following commands need to be executed from the Bash Linux shell to install the **mssql-server** package:

First, download the SQL Server 2019 SLES repository configuration file:

sudo zypper addrepo -fc
https://packages.microsoft.com/config/sles/15/mssql-server-2019.repo

Next, refresh the repositories:

sudo zypper --gpg-auto-import-keys refresh

Next, make sure that the Microsoft package signing key is installed on your system, and use the following command to import the key:

sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc

Next, execute the following command to install SQL Server:

sudo zypper install -y mssql-server

Next, after the package installation finishes, execute **mssql-conf setup** and follow the prompts to set the SA account password and choose the appropriate edition that you are running:

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

Next, once the configuration has completed, verify that the database service is running via:

systemctl status mssql-server

Finally, if your database server installation requires remote server access, you might also need to open the SQL Server TCP port (default 1433 or port that you may have changed it to in an earlier step) on your firewall. If you are using the SuSE firewall, you need to edit the **/etc/sysconfig/SuSEfirewall2** configuration file. Modify the **FW_SERVICES_EXT_TCP** entry to include the SQL Server port number as follows:

FW_SERVICES_EXT_TCP="1433"

Ubuntu:

To install SQL Server on Ubuntu, the following commands need to be executed from the Bash Linux shell to install the **mssql-server** package:

First, import the public repository GPG keys:

wget -qO- https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -

Next, register the SQL Server Ubuntu repository for SQL Server:

For Ubuntu version 16.04:

sudo add-apt-repository "$(wget -qO-

https://packages.microsoft.com/config/ubuntu/16.04/mssql-server-2019.list)"

For Ubuntu version 18.04:

sudo add-apt-repository "$(wget -qO-

https://packages.microsoft.com/config/ubuntu/18.04/mssql-server-2019.list)"

For Ubuntu version 20.04:

sudo add-apt-repository "$(wget -qO-

https://packages.microsoft.com/config/ubuntu/20.04/mssql-server-2019.list)"

Next, execute the following commands to install SQL Server:

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

Next, after the SQL Server package installation completes, run **mssql-conf setup** and follow the prompts to set the SA account password and choose your appropriate edition:

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

Next, once the configuration completes, verify that the database service is running via:

systemctl status mssql-server --no-pager

Finally, if remote database connectivity is required, you may also need to open the SQL Server TCP port (default 1433) on your Ubuntu firewall.

Installation of SQL Server Command-Line Tools

In order to interact with the SQL Server database, tools are required that can run Transact-SQL statements on the database server. The following Bash Linux shell steps are required to install the SQL Server command-line tools: **sqlcmd** and **bcp**:

RHEL:

First, download the Microsoft Red Hat repository configuration file.

For RHEL version 7:

```
sudo curl -o /etc/yum.repos.d/msprod.repo
    https://packages.microsoft.com/config/rhel/7/prod.repo
```

For RHEL version 8:

```
sudo curl -o /etc/yum.repos.d/msprod.repo
    https://packages.microsoft.com/config/rhel/8/prod.repo
```

Next, only if you had a previous version of **mssql-tools** installed, remove any older unixODBC packages as follows:

```
sudo yum remove unixODBC-utf16 unixODBC-utf16-devel
```

Next, execute the following command to install **mssql-tools** with the unixODBC developer package:

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

Next, add **/opt/mssql-tools/bin/** to your **PATH** environment variable. This enables you to run the tools without ever having to prefix the command with a full path specification. The following commands should be executed to modify the **PATH** for both login sessions and interactive/non-login sessions:

```
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
source ~/.bashrc
```

Finally, test to make sure that your database can be interacted with using the **sqlcmd** utility:

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

SLES:

First, add the SQL Server repository to Zypper:

```
sudo zypper addrepo -fc https://packages.microsoft.com/config/sles/15/prod.repo
sudo zypper --gpg-auto-import-keys refresh
```

Next, install the **mssql-tools** with the unixODBC developer package:

sudo zypper install -y mssql-tools unixODBC-devel

Next, add **/opt/mssql-tools/bin/** to your **PATH** environment variable. This enables you to run the tools without ever having to prefix the command with a full path specification. The following commands should be executed to modify the **PATH** for both login sessions and interactive/non-login sessions:

```
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
source ~/.bashrc
```

Finally, test to make sure that your database can be interacted with using the **sqlcmd** utility:

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

Ubuntu:

First, by default, **curl** is not installed on Ubuntu. To install **curl**, execute the following:

```
sudo apt-get update
sudo apt install curl
```

Next, import the public repository GPG keys

curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -

Next, register the Microsoft Ubuntu repository:

For Ubuntu version 16.04:

curl https://packages.microsoft.com/config/ubuntu/16.04/prod.list | sudo tee /etc/apt/sources.list.d/msprod.list

For Ubuntu version 18.04:

curl https://packages.microsoft.com/config/ubuntu/18.04/prod.list | sudo tee /etc/apt/sources.list.d/msprod.list

For Ubuntu version 20.04:

curl https://packages.microsoft.com/config/ubuntu/20.04/prod.list | sudo tee /etc/apt/sources.list.d/msprod.list

Next, update the sources list and execute the installation command with the unixODBC developer package:

```
sudo apt-get update
sudo apt-get install mssql-tools unixodbc-dev
```

Next, to update to the latest version of **mssql-tools** execute the following commands:

```
sudo apt-get update
sudo apt-get install mssql-tools
```

Next, add **/opt/mssql-tools/bin/** to your **PATH** environment variable. This enables you to run the tools without ever having to prefix the command with a full path specification. The following commands should be executed to modify the **PATH** for both login sessions and interactive/non-login sessions:

```
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
source ~/.bashrc
```

Finally, test to make sure that your database can be interacted with using the **sqlcmd** utility:

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

Conclusion

Microsoft SQL Server is no longer limited to Microsoft Windows platforms. It has been ported to a variety of Linux platforms including Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), and Ubuntu. The expansion of platform support over the years has helped solidify SQL Server's relevance in the database server industry.