CHAPTER 1: EXECUTIVE SUMMARY

Microsoft is committed to enabling customers to choose the best platform for their data and applications. With this commitment in mind, Microsoft SQL Server 2017 is now available on Red Hat Enterprise Linux, the world’s leading enterprise Linux platform and preferred choice for enterprise Linux virtual machine (VM) workloads on Microsoft Azure.

Enterprise agility depends on open platforms such as SQL Server 2017 and Azure, and it also depends on the ability to react to information quickly. Businesses today need to be responsive to minute-by-minute changes, and, as a result, they are looking for a scalable and secure database-management system that makes use of in-memory transactions and real-time insights into their businesses. They are also looking to transform their businesses in a way that embraces all their data, the cloud, and artificial intelligence (AI). SQL Server 2017 is the first release of SQL Server that is also available on Linux, and it has everything companies need to run modern workloads—with in-memory, security, and advanced analytics built-in.

SQL Server 2019

The SQL Server 2019 public preview on Red Hat Enterprise Linux enables new scenarios that are not covered in this operations guide, including container support, integration with machine learning with R and Python, and support for server-side Java with the Java language extension.

SQL Server 2017 on Red Hat Enterprise Linux delivers a simple and compelling solution to this requirement. It can run both transactional and analytical workloads with great efficiency on the same system, in-memory or on disk, in a hybrid transactional/analytical processing (HTAP) scenario.

The purpose of this guide is to provide step-by-step instructions to deploy and configure SQL Server 2017 on Red Hat Enterprise Linux to optimize for both transactional and analytical workloads in an Azure VM, in accordance with best practices before, during, and after deployment.

You should read this guide if you are considering a deployment of SQL Server 2017 on Red Hat Enterprise Linux in Azure, either within or independent of an Active Directory environment.
Chapter 1: Executive summary ........ 1

Chapter 2: Deploying SQL Server 2017 on Red Hat Enterprise Linux for HTAP workloads .......... 4
   Introducing SQL Server 2017 on Red Hat Enterprise Linux .......... 4
   Combining the best of both worlds with HTAP ........... 4

Chapter 3: Planning an Azure deployment of SQL Server 2017 on Red Hat Enterprise Linux .......... 6
   Azure image selection for SQL Server 2017 on Red Hat Enterprise Linux .......... 6
   Sizing considerations for hybrid workloads ................. 7
   Managing availability .......... 12
   Licensing and pricing .......... 12
   Planning checklist .......... 13
   Provisioning the VM in Azure .......... 16

Chapter 4: Post-provisioning tasks in Red Hat Enterprise Linux .......... 17
   Before you begin .......... 17
   Objectives .......... 17
   Connecting to the new SQL Server 2017 on Red Hat Enterprise Linux VM .......... 18
   Configuring Red Hat Enterprise Linux for SQL Server 2017 .......... 18

Chapter 5: Joining the SQL Server on Red Hat Enterprise Linux machine to an Active Directory domain .......... 29
   Before you begin .......... 29
   Objectives .......... 29
   Change the default port for SQL Server (recommended) .......... 30
   Prepare the server to join the domain .......... 32
   Join the server to the domain .......... 34
   Configure a service account for SQL Server 2017 on the domain controller .......... 38
   Set the service principal name (SPN) for the SQL Server 2017 service account .......... 38
   Configure a keytab file to allow the service account to automatically sign in to the Active Directory domain at boot .......... 39
   Further reading .......... 45

Chapter 6: Configuring SQL Server 2017 on Red Hat Enterprise Linux .......... 46
   Before you begin .......... 46
   Objectives .......... 46
   Change the default port for SQL Server 2017 (if it is not already changed) .......... 46
   Prepare the environment for SQL Server 2017 administration .......... 47
   Configure default locations for database and log files .......... 49
   Change the location of TempDB .......... 50
   Test the configuration: Allow an Active Directory user permission to connect to SQL Server 2017 .......... 53
   Disable the sa account (recommended) .......... 56
   Connect to SQL Server 2017 from Azure Data Studio .......... 56

Chapter 7: Configuring HTAP and operational analytics .......... 59
   Objectives .......... 59
   In-Memory OLTP .......... 59
   Configuring columnstore indexes for operational analytics .......... 59

Chapter 8: Connecting Power BI to your database .......... 62
   Before you begin .......... 62
   Objectives .......... 62
   Connect Power BI Desktop to a SQL Server on Red Hat Enterprise Linux VM .......... 62
   Create a dashboard in the Power BI service .......... 64

Chapter 9: Configuring high availability and disaster recovery for SQL Server 2017 on Red Hat Enterprise Linux VMs in Azure .......... 65
   Configure availability groups in SQL Server 2017 for manual failover .......... 65
   Back up and restore SQL Server 2017 on Red Hat Enterprise Linux in Azure .......... 69

Chapter 10: Monitoring tools in SQL Server 2017 on Red Hat Enterprise Linux in Azure .......... 70
   T-SQL .......... 70
   Insight widgets in Azure Data Studio .......... 70
   SQL Server Management Studio (SSMS) .......... 71
   Command-line tools in Red Hat Enterprise Linux .......... 72

Next Steps .......... 72
Here is a preview of what you will cover in this operations guide:

- **Chapter 2** provides an overview of the HTAP configuration.
- **Chapter 3** introduces detailed planning steps to be taken in advance of provisioning the Azure VM to be used for SQL Server 2017 on Red Hat Enterprise Linux.
- **Chapter 4** specifies recommended steps to configure a newly provisioned Red Hat Enterprise Linux VM for SQL Server 2017.
- **Chapter 5** provides step-by-step instructions for joining the new Red Hat Enterprise Linux VM to an Active Directory domain.
- **Chapter 6** details recommended configuration steps to take in SQL Server 2017 after you complete server-level configuration. It then provides instructions to download and populate a sample database.
- **Chapter 7** demonstrates how to configure the imported database for HTAP and operational analytics through the use of columnstore indexes.
- **Chapter 8** shows how to connect this database to Power BI and how to import existing reports.
- **Chapter 9** provides guidance for how to manage high availability for the Linux VM, followed by best practices for disaster recovery.
- **Chapter 10** describes various tools that can be used to monitor the performance of SQL Server 2017 on Red Hat Enterprise Linux.

After you read this guide, you should be able to deploy and configure SQL Server 2017 on Red Hat Enterprise Linux in an Azure VM according to best practices, either within or independent of an Active Directory environment. You should also be able to configure a database with columnstore indexes to enable operational analytics alongside a transactional workload (HTAP), connect the database to analytics tools, and perform routine administrative tasks such as backups and performance monitoring.

**SQL Server 2017 on Linux Quick Start Guide**

CHAPTER 2: DEPLOYING SQL SERVER 2017 ON RED HAT ENTERPRISE LINUX FOR HTAP WORKLOADS

SQL Server 2017 runs on Red Hat Enterprise Linux and other Linux distributions. With SQL Server 2017, you have flexibility in your choice of platform and can enjoy the advantages of SQL Server 2017 along with the excellent stability, security, and total cost of ownership (TCO) of the world’s leading enterprise Linux distribution. Because of its unique combination of excellent features, performance, and stability, SQL Server 2017 on Red Hat Enterprise Linux in Azure serves as an ideal platform to support both in-memory transactions and real-time intelligence over operational data with in-memory analytics in HTAP workloads.

INTRODUCING SQL SERVER 2017 ON RED HAT ENTERPRISE LINUX

SQL Server 2017 on Linux, in fact, is not a rewrite or a port. Though the application runs on multiple operating systems, the underlying database engine remains the same. The SQL Server 2017 versions on Windows and on Linux share a common code base that accesses operating system functions through a platform-abstraction layer. (Note, however, that native calls are still used for CPU-, memory-, and disk-intensive operations.)

SQL Server 2017 on Linux features

Because all editions of SQL Server (regardless of the underlying platform) share the same codebase, all features in the T-SQL surface area, such as in-memory tables, columnstore, and security-related features, work the same. Customers are now able to modernize their applications with SQL Server 2017 for enhanced performance, and they are also able to choose the platform they want. (Note that customers have the freedom to later lift-and-shift their SQL Server databases to an Azure SQL Database managed instance, if at some point they are looking for a managed platform as a service [PaaS].)

For information about SQL Server 2017 features not currently supported on Linux, see Release notes for SQL Server 2017 on Linux. For general information about SQL Server 2017 on Linux, see SQL Server on Linux Frequently Asked Questions (FAQ).

COMBINING THE BEST OF BOTH WORLDS WITH HTAP

One of the great features of SQL Server is the ability to make use of in-memory technology to dramatically improve the transactional speed of business-critical applications. Traditionally, however, customers have not been able to have real-time visibility into their businesses because performing analytical queries against transactional tables would slow down their systems. For analytical queries, transactional data has typically been copied and moved through an extract, transform, and load (ETL) process to a separate reporting database (as shown in Figure 2.1), which is optimized for standard reporting by using indexes. Because of the complex nature of this process, there is a necessary delay between the time when the transactional data is written and when analytics can be performed.
Real-time operational analytics can run alongside an OLTP workload without impacting its performance.

Now, however, it is possible to add a columnstore index on top of the transactional tables. While transactions use rowstore, the compressed columnstore technology in clustered indexes is optimized for analytical queries without affecting the transactional performance. The columnstore index is kept up to date automatically and in real time. This new HTAP scenario is shown in Figure 2.2.

Because the indexed data mirrors rowstore data, real-time operational analytics can run alongside an OLTP workload without impacting its performance. These real-time analytics capabilities, coupled with the memory-optimized table capabilities of SQL Server 2017, enable unprecedented speed with in-memory OLTP—creating a powerful HTAP platform.

In the following chapters, you will deploy SQL Server 2017 on Red Hat Enterprise Linux and then configure a database to support HTAP functionality.
CHAPTER 3: PLANNING AN AZURE DEPLOYMENT OF SQL SERVER 2017 ON RED HAT ENTERPRISE LINUX

When you provision your SQL Server 2017 on Red Hat Enterprise Linux virtual machine (VM) in Azure, you will be prompted to choose from among various options related to the VM image type, VM size, disk storage, networking, security, and other features. These are essentially design decisions that should come as a result of careful planning. You should know, therefore, which options you will be presented with and how you will configure them in advance of provisioning any server destined for a production environment. Some of these configuration choices might be predetermined by existing policies set by your organization's IT team.

This chapter discusses important design decisions for your SQL Server 2017 on Red Hat Enterprise Linux VM that you need to make in advance of deployment. The end of the chapter presents a planning checklist of the options you will see during provisioning as a way to help you to prepare for this deployment.

AZURE IMAGE SELECTION FOR SQL SERVER 2017 ON RED HAT ENTERPRISE LINUX

The Azure Marketplace contains VM images of Red Hat Enterprise Linux that come preinstalled with SQL Server 2017. Provisioning one of these preconfigured images is, in fact, the preferred way to set up SQL Server 2017 in a new Linux VM on Azure for most use cases.

You can view the available preconfigured images for SQL Server 2017 on Red Hat Enterprise Linux by visiting the Microsoft Azure portal at http://portal.azure.com. Then, in the Marketplace Search box, search for SQL RHEL. You will see search results similar to those shown in Figure 3.1. Note that deploying the images for SQL Server 2017 Developer and SQL Server 2017 Express requires no licensing fees, because they are used only for testing, development, or unusually small workloads.

<table>
<thead>
<tr>
<th>NAME</th>
<th>PUBLISHER</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Server 2017 Enterprise on Red Hat Enterprise Linux 7.4 (RHEL)</td>
<td>Microsoft</td>
</tr>
<tr>
<td>Free SQL Server License: SQL Server 2017 Developer on Red Hat Enterprise Linux 7.4 (RHEL)</td>
<td>Microsoft</td>
</tr>
<tr>
<td>Free SQL Server License: SQL Server 2017 Express on Red Hat Enterprise Linux 7.4 (RHEL)</td>
<td>Microsoft</td>
</tr>
<tr>
<td>SQL Server 2017 Web on Red Hat Enterprise Linux 7.4 (RHEL)</td>
<td>Microsoft</td>
</tr>
<tr>
<td>SQL Server 2017 Standard on Red Hat Enterprise Linux 7.4 (RHEL)</td>
<td>Microsoft</td>
</tr>
</tbody>
</table>

Figure 3.1. Preconfigured images available for SQL Server 2017 on Red Hat Enterprise Linux in the Azure Marketplace

To know which of these five images to choose, you first need to determine the scaling requirements for your deployment of SQL Server 2017. Table 3.1 describes the scale limits for each of the associated SQL Server 2017 editions.
SIZING CONSIDERATIONS FOR HYBRID WORKLOADS

The workload you plan to run on SQL Server 2017 will influence the amount of resources you want to assign to the Red Hat Enterprise Linux VM that it will run on. Generally speaking, HTAP workloads are both read- and write-heavy and require more memory and storage capacity than is typically required for plain OLTP systems. If, as opposed to running your analytic workloads on disk, you will run your analytic workloads in RAM in clustered columnstore indexes, you will then need to predict the size of those indexes and then factor in the additional memory capacity that will be needed to support them on top of the original RAM requirements. Finally, if you plan also to run transactional workloads in RAM (a best-case scenario in terms of performance), then you need to account for those additional memory requirements too.

For an overview of sizing and design considerations for hosting SQL Server 2017 on Azure, see Performance guidelines for SQL Server in Azure Virtual Machines.
**VM size**

VM sizes in Azure refer to the available VM-instance configurations. VM sizes vary by the type of CPU, the number of virtual CPUs, the amount of memory assigned, and other factors that affect performance. Each VM size has a particular name. For the SQL Server 2017 Enterprise on Red Hat Enterprise Linux image, the default VM size is "Standard DS13 v2," which comes with 8 virtual CPUs and 56 GB of RAM.

You can begin to explore the available VM sizes in Azure at the following source: Sizes for Linux virtual machines in Azure.

**Memory-optimized VMs for in-memory optimized hybrid transactional and analytical workloads**

For HTAP use cases targeting the best available performance with memory-optimized tables and clustered columnstore indexes, choosing a VM size in the memory-optimized family is recommended. For example, VMs within the memory-optimized M-series offer the highest memory capacity. The GS-series and DSv2-series 11–15 (including the default Standard DS13 v2) are also optimized for large memory requirements.

The following memory-optimized VMs are considered to meet the minimum size requirements for SQL Server 2017:

- DS3_v2 or higher for SQL Server 2017 Enterprise edition
- DS2_v2 or higher for SQL Server 2017 Standard and Web editions

You can read more about memory-optimized VM sizes at Memory optimized virtual machine sizes.

**VMs that support premium storage and read-only caching**

Another recommendation is to choose a memory-optimized VM that supports premium storage and read-only caching for that premium storage. (Read-only caching is recommended for the disk reserved for database files, including TempDB.) Note that the default VM size assigned to the SQL Server 2017 Enterprise on Red Hat Enterprise Linux image does support premium storage and read-only caching on that premium storage.

**Sizing the VM for input/output (I/O) capacity**

When choosing a particular VM size, be aware that each size has scale limits and performance specifications for I/O operations per second (IOPS), bandwidth, and the number and type of disks that can be attached. Therefore, you need to make sure you choose a VM size that can scale up to your workloads’ requirements for cached and uncached IOPS, throughput, and size.

For example, for the DS14_v2 VM selected in Figure 3.2, the maximum IOPS for cached data is 64,000, and 54,000 for uncached data, the maximum throughput is 512 MB/s for cached data, and 768 MB/s for uncached data, and the maximum number of attached disks of 64.

This and similar information about other VM sizes is available at Memory optimized virtual machine sizes.
Create a virtual machine

Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization.
Looking for classic VMs? Create VM from Azure Marketplace

PROJECT DETAILS
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription  
  * Resource group  

Create new

INSTANCE DETAILS
* Virtual machine name  
  * Region  
  * Availability options  
  * Image  
  * Size  

SQLRhel2  
East US  
No infrastructure redundancy required  
SQL Server 2017 Enterprise on Red Hat Enterprise Linux 7.4 (RHEL)  
Browse all images and disks  

Standard DS14 v2  
16 vcpus, 112 GB memory  
Change size

Figure 3.2. The VM size for the SQL Server 2017 Enterprise on Red Hat Enterprise Linux image is specified during the provisioning process—you should research which size is best suited to your needs ahead of time

Planning disk storage

For storage planning, it is recommended that you attach at least two additional disks to the VM beyond the operating system disk: one for log files and one for data files, including TempDB. The following sections describe the types of disks that are available, along with recommendations and guidelines about the types and number of disks to choose.

Disk types

When selecting disk types during VM provisioning in Azure, you will see the options shown in Figure 3.3

It is recommended that you attach at least two additional disks to the VM beyond the operating system disk: one for log files and one for data files...
Of the three disk types shown in Figure 3.3, premium solid-state drives (SSDs) are recommended for SQL Server 2017 workloads because of their high I/O capabilities. As mentioned above, not all VM sizes support premium storage.

Note that a fourth disk option, Ultra SSD, is also available for some VM sizes and locations. This final disk type delivers by far the best read and write performance. It is therefore the best option for the most transaction-heavy database workloads (ones requiring ~50,000 IOPS per disk).

Finally, a new "Lsv2-series" VM size has been announced for general availability in 2019 for certain regions, including East US, East US 2, West Europe, SE Asia, West US 2, and North Europe. These VMs in the storage-optimized family include NVM Express (NVMe) disks that support throughput rates of up to 3,400,000 read IOPS and 22,000 MB per second (MBps).

For more information about these four disk types, visit What disk types are available in Azure? For more information about the new Lsv2-series disks, see Announcing the general availability of Lsv2-series Azure Virtual Machines.
Disk sizes

As with VM sizes, different disk sizes are available in Azure, each with its own name. Premium SSD size names begin the letter P, such as P30 and P50. It is recommended that your disks for log and data files each use a size of P30 or higher. See Table 3.2 for the throughput and capacity specifications for the recommended premium SSD sizes. (Disk sizes marked with an asterisk are currently available in preview.)

<table>
<thead>
<tr>
<th>PREMIUM SSD SIZES</th>
<th>P30</th>
<th>P40</th>
<th>P50</th>
<th>P60</th>
<th>P70</th>
<th>P80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk size in GiB</td>
<td>1,024</td>
<td>2,048</td>
<td>4,096</td>
<td>8,192</td>
<td>16,384</td>
<td>32,767</td>
</tr>
<tr>
<td>IOPS per disk</td>
<td>Up to 5,000</td>
<td>Up to 7,500</td>
<td>Up to 7,500</td>
<td>Up to 12,500</td>
<td>Up to 15,000</td>
<td>Up to 20,000</td>
</tr>
<tr>
<td>Throughput per disk</td>
<td>Up to 200 MiB/sec</td>
<td>Up to 250 MiB/sec</td>
<td>Up to 250 MiB/sec</td>
<td>Up to 480 MiB/sec</td>
<td>Up to 750 MiB/sec</td>
<td>Up to 900 MiB/sec</td>
</tr>
</tbody>
</table>

Table 3.2. Premium SSD sizes P30 and above in Azure

Number of disks required

You can use Table 3.2 to help you determine how many disks of what type you will need to meet your throughput and size requirements. For example, if the IOPS requirements for the data file are 30,000 IOPS, then you would need to stripe over six P30 disks (6 x 5,000 IOPS apiece = 30,000) or four P40 or P50 disks (4 x 7,500 IOPS apiece = 30,000).

Additional guidelines

Note also the following guidelines about configuring storage for your SQL Server 2017 on Red Hat Enterprise Linux VM:

- Enable read-only caching on the disks hosting the data files and TempDB data files. Doing so significantly improves throughput for read-heavy workloads.

- Do not enable caching on disks hosting the log file. (Important: Stop the SQL Server 2017 service when changing the cache settings for an Azure VM disk.)

For more comprehensive guidance on determining your storage requirements, visit Azure premium storage: design for high performance.
MANAGING AVAILABILITY

At the time of this writing, operating system–level clustering via Pacemaker with automatic failover is not supported on Azure VMs. This means that automatic failover using SQL Server AlwaysOn Failover Cluster Instances or SQL Server AlwaysOn Availability Groups (AGs) is not supported. However, you can still use Pacemaker and AGs for scale-out scenarios with readable secondary replicas and perform manual failover. In addition to this, Azure supports many other high-availability features. To configure high availability, the following strategies are recommended:

• **Configure Pacemaker or a read-scale SQL Server AlwaysOn AG for manual failover.** Although automatic failover is not yet supported for Linux VMs, manual failover is possible with Pacemaker or a cluster-less AG, also known as a read-scale AG. A read-scale AG is supported by one or more read-only secondaries, which allow read-only queries to execute after failover is complete. After manual failover, you can also promote a secondary node to the primary node.

• **Set up multiple VMs in an availability set:** Grouping two or more VMs in an availability set ensures that at least one VM is available at all times during a maintenance event. You can choose to configure an availability set on the Basic (first) configuration page when you create a new VM.

• **Use managed disks instead of unmanaged disks in an availability set:** Managed disks align storage availability with VM availability, ensuring that storage replicas are isolated in separate fault domains of an availability set. They also greatly simplify management by eliminating the need for storage accounts. Managed disks are the default option for disks in the SQL Server 2017 on Red Hat Enterprise Linux images.

• **Use scheduled events to take action when impacting events have occurred:** Enabling scheduled events notifies the VM of maintenance events that can affect your VM.

• **Add a load balancer with availability sets:** A load balancer can enhance application resiliency and distribute workloads between VMs.

• **Use availability zones to cater for datacenter-level failures:** Availability zones are separate locations within the Azure region. Data is protected through replication in the different zones in the region. If failure occurs in one of the zones, the replicated data will be available in other zones.

*For more information about configuring high availability for Linux VMs in Azure, see Manage the availability of Linux virtual machines, along with Chapter 9 of this guide.*

LICENSING AND PRICING

In general, the costs of hosting a SQL Server 2017 on Linux VM in Azure can be separated into SQL Server 2017 licensing costs, Azure VM–usage costs, and disk-usage costs.

**VM pricing**

The costs associated with SQL Server 2017 on Red Hat Enterprise Linux images are simplified because they all include SQL Server 2017 licensing costs. By default, the pricing for the images reflects a pay-as-you-go plan that is associated with your choice of VM size. You can preview these costs when you select the VM size during initial configuration. Note, however, that you can use Azure reservations to help you save money by pre-paying for one year or three years of VMs or Azure SQL Database compute capacity. You can buy reservations in the Azure portal.

*For more information on Linux VM pricing, visit Linux Virtual Machines Pricing. For more information about Azure reservations (also called reserved instances), see What are Azure Reservations?*
Azure Hybrid Benefit and SQL Server 2017

Azure Hybrid Benefit is a cost-effective licensing option available for VMs on which SQL Server 2017 is installed manually on a Linux VM (as opposed to a preconfigured SQL Server 2017 on Linux image), and when your organization already has on-premises licenses.

*For more information on Azure Hybrid Benefit, visit Azure Hybrid Benefit for SQL Server on Azure Virtual Machines.*

Disk pricing and costs

Each of the four disk types is associated with a different cost structure. The following factors affect disk pricing and costs:

- Disk type
- Disk size
- Snapshots
- Outbound data transfers
- Number of transactions (standard hard-disk drive [HDD] managed disks and standard SSD managed disks only)

*For more information about the price structure associated with various disk types, see Managed Disks pricing.*

PLANNING CHECKLIST

Before you provision your SQL Server 2017 on Red Hat Enterprise Linux VM, you can use the following checklist to review, research, and pre-select the configuration settings that you’ll encounter during the provisioning process. The items in the list all represent separate design and planning decisions you need to make before you provision the VM.

Choose an image type

- SQL Server 2017 Enterprise on Red Hat Enterprise Linux 7.4 (RHEL)
- SQL Server 2017 Developer on Red Hat Enterprise Linux 7.4 (RHEL)
- SQL Server 2017 Standard on Red Hat Enterprise Linux 7.4 (RHEL)
- SQL Server 2017 Web on Red Hat Enterprise Linux 7.4 (RHEL)
- SQL Server 2017 Express on Red Hat Enterprise Linux 7.4 (RHEL)

*For more information about this selection, see Sizes for Linux virtual machines in Azure.*

Choose an Azure deployment model

- Resource Manager (default)
- Classic (deprecated)

*For more information on this selection, see Azure Resource Manager vs. classic deployment: Understand deployment models and the state of your resources.*
The following checklist is organized according to the configuration pages in the Create A Virtual Machine wizard in the Azure portal. However, you will need to make decisions about these same items regardless of whether you use the Azure portal, PowerShell, Azure CLI, or Azure Resource Manager templates for provisioning. Note also that the organization of these settings within the Azure portal might change.

Create a VM—Basics page
The bulleted design decisions to make below correspond to configuration settings that, at the time of this writing, appear on the first (Basics) page of the Create A Virtual Machine wizard.

Project details
1. Choose an Azure subscription.
2. Choose a resource group within that Azure subscription.

For more information about this decision, see Organizing subscriptions and resource groups within the Enterprise.

Instance details
3. Choose a VM name.

Availability options
5. Choose from among these availability options:
   • No infrastructure redundancy required
   • Availability set
   • Availability zone

For more information about this decision, see Azure VMs: Availability Sets and Availability Zones.

Administrator account
7. Choose the authentication type for the administrator account.
   • Password: Choose an administrator password.
   • Secure Shell (SSH): Generate the key pair and make the public key available during provisioning.
8. Choose the administrator username.

Create a VM—Disks page
The bulleted design decisions below correspond to configuration settings that appear on the second (Disks) page of the Create A Virtual Machine wizard.
Disk options
1. Choose the operating system disk type.

For more information about this decision, see What disk types are available in Azure?

Data disks
2. Determine what your I/O requirements are.
3. Determine how many disks, and of what disk type and disk size, you need to meet those I/O requirements.

For more information about this decision, see Azure premium storage: design for high performance.

4. Decide whether you will use managed disks.
   • Yes (default)
   • No

For more information about this decision, see Introduction to Azure managed disks.

Create a VM—Networking page
The bulleted design decisions below correspond to configuration settings that appear on the third (Networking) page of the Create A Virtual Machine wizard.

Configure virtual networks
1. Choose a virtual network. See What is Azure Virtual Network?
2. Choose a subnet within that virtual network. See Subnets.
3. Choose which available public IP address you want to assign this VM to (optional).
4. Choose which network interface controller (NIC) network security group (NSG) the VM will belong to. See Security groups.

Public inbound ports
5. Determine which of the following inbound ports (default) should be opened when the VM is provisioned:
   • HTTP
   • HTTPS
   • SSH
   • RDP

6. Decide which inbound ports (non-default) should be opened after provisioning but before production, and for which services.

7. Decide whether you will enable accelerated networking for the VM. See Create a Linux virtual machine with Accelerated Networking.
Load balancing
8. Decide whether you will place the VM behind an existing load balancing solution.
   See Load balancers.

Create a VM—Management page
The bulleted design decisions below correspond to configuration settings that appear on the fourth (Management) page of the Create A Virtual Machine wizard.

Monitoring
1. Decide whether you want to keep boot diagnostics enabled on the VM (default = on).
   • See Microsoft Azure Support diagnostic information and memory dump collection.
2. Decide whether you want to enable operating system guest diagnostics on the VM (default = off).
   • See Add extended metrics for Azure virtual machines.
3. Decide whether you want to store diagnostics data in a storage account.
   • See Store and view diagnostic data in Azure Storage.

Identity
4. Decide whether you want to enable a system-assigned managed identity for the VM.
   • See What is managed identities for Azure resources?

Auto-shutdown
5. Decide whether you want your VM to automatically shut down daily.

PROVISIONING THE VM IN AZURE
After you have completed the list of tasks on the checklist, you are ready to provision your SQL Server 2017 on Red Hat Enterprise Linux VM in the Azure portal.

For complete, step-by-step instructions on provisioning process, see Provision a Linux SQL Server virtual machine in the Azure portal.
CHAPTER 4: POST-PROVISIONING TASKS IN RED HAT ENTERPRISE LINUX

After you have provisioned the SQL Server 2017 on Red Hat Enterprise Linux image, you should perform a number of configuration steps in the operating system before you move on to configure SQL Server 2017 itself. To begin, you need to connect to the VM hosted in Azure that you have just provisioned.

BEFORE YOU BEGIN
To begin performing the steps in this chapter, you should have:

• Fully provisioned a SQL Server 2017 on Red Hat Enterprise Linux VM from the Azure portal.
• Attached two extra disks to the VM.
• Configured the VM to allow external connections through the default SSH port (TCP 22).
• Either configured an administrator password or supplied an SSH public key to allow the SSH connection.
• Started the VM.

OBJECTIVES
This chapter describes procedures that allow you to complete the following tasks:

• Use a local client computer to connect to the VM in Azure via an SSH connection
• Change the default port for SSH
• Prepare the two additional disks attached to the VM for the SQL Server data and log files, respectively
CONNECTING TO THE NEW SQL SERVER 2017 ON RED HAT ENTERPRISE LINUX VM

You will connect to the SQL Server 2017 on Red Hat Enterprise Linux VM by means of an SSH connection from your client machine. If you are connecting from a Windows 10 client, you should know that Windows 10 version 1809 and later supports OpenSSH, the open-source version of SSH, as an installable component. Once OpenSSH is installed, you can initiate SSH connections directly in PowerShell without having to install an SSH client. For more information about installing, configuring, and using OpenSSH in Windows 10, see OpenSSH in Windows.

For Windows users who prefer the ease and convenience of a GUI tool for SSH, you can download and install an SSH client. Two free examples are PuTTY (available at www.putty.org) and MobaXterm (available at https://mobaxterm.mobatek.net/).

For information about using the SSH client in Linux, see Using the Linux client.

Regardless of the SSH client you choose to connect to your new SQL Server 2017 on Red Hat Enterprise Linux VM, when you do connect, you will need to specify the public IP address of that VM. This IP address is available in the Azure portal.

Securing your SSH connection with cryptographic keys

It is recommended that you use an SSH key pair to secure your SSH connection to the remote VM, as opposed to a password. If you did not generate a key pair and supply the public key during the provisioning of the VM, you can do that at any point while connected to the VM by using the instructions in the following article: Quick steps: Create and use an SSH public-private key pair for Linux VMs in Azure. After generating the key pair, you will need to supply the private key in the SSH client on your local system.

Performing administrative tasks in Red Hat Enterprise Linux with sudo privileges

When you connect to the new SQL Server 2017 on Red Hat Enterprise Linux VM through SSH, you will sign in with the administrator account that you created during the provisioning process for the VM. You will use this administrator account with sudo privileges to perform operating system–level administrative tasks. Note that the root account in Red Hat Enterprise Linux is disabled in the VM and should not be used when signing into the environment.

CONFIGURING RED HAT ENTERPRISE LINUX FOR SQL SERVER 2017

At this point, you should have established an SSH connection to the SQL Server 2017 on Red Hat Enterprise Linux VM and have signed in with your administrator account. Next, this chapter will cover the operating system configuration steps that you need to take to prepare for SQL Server 2017 configuration.

These steps include:

• Installing Security-Enhanced Linux (SELinux) policy-management tools.

• Changing the default port for SSH.

• Configuring the drives and directories for SQL Server 2017.
Installing SELinux policy-management tools

SELinux is a Linux kernel security module that is enforced in Red Hat Enterprise Linux by default. To configure SELinux to allow the default port settings to be changed for services such as SSH, you first need to install SELinux policy-management tools, including the semanage utility. Semanage is used to configure certain elements of SELinux policy. To install the SELinux policy-management tools, enter the following command at the SSH prompt:

```
sudo yum install policycoreutils-python
```

Changing the default port for SSH

The default TCP port for SSH is 22. It is highly recommended that you change this port immediately to a value above 20,000. Perform the following procedure to do so:

1. Authorize SSH server to bind to the port of your choice using the semanage utility.

Use the following command to change the default SSH port to a value above 20,000. This examples uses port 23,456, but you should choose your own value and record it. In this step, you will use the semanage (SELinux policy-management tool) utility to edit the security policy and allow the SSH server to bind to the port specified, instead of the default.

```
sudo semanage port -a -t ssh_port_t -p tcp 23456
```

2. Verify the policy was updated.

```
sudo semanage port --list | grep ssh_port
```

3. Open the newly chosen port in the operating system firewall.

Type the following command to open the chosen port in the Red Hat Enterprise Linux firewall. Replace 23456 in the example below with the port above 20,000 that you have selected for SSH.

```
sudo firewall-cmd --zone=public --add-port=23456/tcp --permanent
```

4. Verify the port was opened in the firewall.

```
sudo firewall-cmd --list-all
```

5. Edit the configuration of the SSH server to change the port that it listens over.

Although you have configured SELinux to allow SSH to bind to the new port, you haven’t yet configured the SSH service to listen over that new port. To do so, you need to edit the SSH configuration file. You can use the following command to open the file in the text editor nano:

```
sudo nano /etc/ssh/sshd_config
```

In the configuration file, look for the following line:

```
#Port 22
```

Un-comment the line and change 22 to the number of your new port over 20,000. Continuing with the example of port 23,456, the line should appear as the following after you have edited it:

```
Port 23456
```

Press Ctrl+X to exit, press Y to save the changes, and then press Enter to keep the default file name.

(For more information about the nano text editor, type man nano.)
6. Restart the SSH server so it binds and listens on the new port.

To apply the changes to the SSH server, you now need to restart it by typing the following command at the SSH prompt:

```
sudo systemctl restart sshd
```

7. Remove the old firewall rule for port 22, which is no longer in use.

You now need to ensure that Red Hat Enterprise Linux no longer leaves the firewall open for the default SSH port. You can do this by removing the default SSH firewall rule:

```
sudo firewall-cmd --zone=public --remove-port=22/tcp --permanent
```

**Note:** You can safely ignore the red "Warning: NOT_ENABLED: 22:tcp" message that appears after you remove this default firewall rule.

8. Reload the operating system firewall for the changes to take effect.

To apply the changes to the firewall, you need to reload it by typing the following command:

```
sudo firewall-cmd --reload
```

9. Update the inbound port rule in the Azure NSG to the new SSH port.

The new SQL Server 2017 on Red Hat Enterprise Linux VM was placed in an NSG when it was provisioned. When you selected the option to open the SSH port during the provisioning process, an inbound port rule allowing inbound traffic through TCP port 22 was then also assigned to that same NSG. Now, you have to edit that default inbound port rule for SSH so that it allows traffic through your newly chosen SSH port instead of port 22.

A. In the Azure portal, navigate to the SQL Server 2017 on Red Hat Enterprise Linux VM.

B. In the navigation menu for the VM, select **Networking**, as shown in Figure 4.1.

![Figure 4.1. Accessing networking features for the SQL Server 2017 on Red Hat Enterprise Linux VM](image)

C. In the **Networking** page that opens, double-click the **SSH** rule, as shown in Figure 4.2.
D. In the SSH page that opens, replace the value 22 in the **Destination port ranges** field with the new TCP port number that you assigned to the SSH server in Red Hat Enterprise Linux. Figure 4.3 shows this step by continuing with the example SSH port, **23456**.

E. On the SSH page, click **Save**.
Configuring the drives and directories for SQL Server 2017

When you provisioned your SQL Server 2017 on Red Hat Enterprise Linux VM, it automatically came attached with an operating system disk. You then attached two additional disks to be used for the data and log files, respectively. Now, you have to partition and format those two additional disks, and finally mount the new partitions to directories on the root file system.

Striping disks for improved I/O

If you have more disks available, you can use the pvcreate and lvcreate commands to create a RAID0 logical striped volume to get increased I/O throughput. For more information, see Creating a RAID0 (striped) logical volume.

1. Create three directories, with "mssql" as the parent and "log" and "data" for the mount points of the two additional drives.

You first need to create new data and log directories on the root file system that will act as the mount points for the two additional disks. You will put these two directories in a parent mssql directory.

To perform this step, enter the following three commands, one at a time:

```bash
sudo mkdir /mnt/mssql
sudo mkdir /mnt/mssql/data
sudo mkdir /mnt/mssql/log
```

2. Change the permissions of those directories to the mssql user.

When SQL Server 2017 is installed, a user mssql and group mssql are created. Enter the following two commands, one at a time, to change the ownership of your new directories to this user and group. This step will allow SQL Server 2017 to interact with these directories.

```bash
sudo chown mssql:mssql /mnt/mssql/log
sudo chown mssql:mssql /mnt/mssql/data
```

3. Verify that the new directories have been created.

```bash
sudo ls -lash /mnt/mssql/
```
You should see the following output. The last two rows are named "data" and "log," and the ownership is set to the mssql user and group.

| 0 drwxr-xr-x. 4 root  root 29 Mar 13 02:45 . |
| 0 drwxr-xr-x. 5 root  root 48 Mar 13 02:44 .. |
| 0 drwxr-xr-x. 4 mssql mssql 248 Mar 26 21:46 data |
| 0 drwxr-xr-x. 2 mssql mssql  86 Mar 26 21:46 log |

4. Partition the first additional hard drive with the parted interactive utility.

The SQL Server 2017 on Red Hat Enterprise Linux VM currently has device pointers corresponding to four drives mounted to the root file system. Type the following at the Bash prompt:

```bash
ls /dev | grep ‘sd’
```

You should see the following output:

```
sda
sdb
sdc
sdd
```

The main operating system disk is mounted as /dev/sda. Azure automatically mounts a temp drive as /dev/sdb. The two additional disks that you attached during provisioning are mounted as /dev/sdc and /dev/sdd, respectively.

You now need to partition the attached devices mounted as /dev/sdc and /dev/sdd so that you can then mount their partitions to the root file system. To perform this task, use the parted utility.

Perform the following steps:

A. To open the parted utility to partition the first attached disk, type the following command:

```
sudo parted /dev/sdc
```
B. At the **(parted)** command prompt that appears, type **help** to see the list of commands. You will then see the following output:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>help [COMMAND]</td>
<td>Print general help, or help on COMMAND</td>
</tr>
<tr>
<td>mklabel,mktbl LABEL-TYPE</td>
<td>Create a new disklabel (partition table)</td>
</tr>
<tr>
<td>mkpart PART-TYPE [FS-TYPE] START END</td>
<td>Make a partition</td>
</tr>
<tr>
<td>name NUMBER NAME</td>
<td>Name partition NUMBER as NAME</td>
</tr>
<tr>
<td>print [devices</td>
<td>free</td>
</tr>
<tr>
<td>quit</td>
<td>Exit program</td>
</tr>
<tr>
<td>rescue START END</td>
<td>Rescue a lost partition near START and END</td>
</tr>
<tr>
<td>resizepart NUMBER END</td>
<td>Resize partition NUMBER</td>
</tr>
<tr>
<td>rm NUMBER</td>
<td>Delete partition NUMBER</td>
</tr>
<tr>
<td>select DEVICE</td>
<td>Choose the device to edit</td>
</tr>
<tr>
<td>disk_set FLAG STATE</td>
<td>Change the FLAG on selected device</td>
</tr>
<tr>
<td>disk_toggle [FLAG]</td>
<td>Toggle the state of FLAG on selected device</td>
</tr>
<tr>
<td>set NUMBER FLAG STATE</td>
<td>Change the FLAG on partition NUMBER</td>
</tr>
<tr>
<td>toggle [NUMBER [FLAG]]</td>
<td>Toggle the state of FLAG on partition NUMBER</td>
</tr>
<tr>
<td>unit UNIT</td>
<td>Set the default unit to UNIT</td>
</tr>
<tr>
<td>version</td>
<td>Display the version number and copyright</td>
</tr>
</tbody>
</table>

C. At the parted command prompt, type **p** to view the partition table.
You will see the following output. Notice that no partitions are listed at the bottom, below the row of column headers beginning with "Number" and "Start."

(parted) p
Error: /dev/sdc: unrecognized disk label
Model: Msft Virtual Disk (scsi)
Disk /dev/sdc: 1098GB
Sector size (logical/physical): 512B/4096B
Partition Table: unknown
Disk Flags:

<table>
<thead>
<tr>
<th>Number</th>
<th>Start</th>
<th>End</th>
<th>Size</th>
<th>Type</th>
<th>File system</th>
<th>Flags</th>
</tr>
</thead>
</table>

D. At the parted command prompt, type `mklabel gpt` to create a new MBR partition table.
E. At the parted command prompt, type `mkpart primary 2048MB 1098GB` to create a new primary partition.
F. To verify that a new partition has been created, type `p` at the parted command prompt to view the partition table again.

You will see the following output. Note that a new partition 1 now appears at the bottom.

(parted) p
Model: Msft Virtual Disk (scsi)
Disk /dev/sdc: 1098GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

<table>
<thead>
<tr>
<th>Number</th>
<th>Start</th>
<th>End</th>
<th>Size</th>
<th>Type</th>
<th>File system</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2048MB</td>
<td>1098GB</td>
<td>1096GB</td>
<td>primary</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

G. At the parted command prompt, type `quit` to quit the interactive shell.

5. Use the `mkfs.xfs` utility to format the new partition with the XFS file system.

You have now created a partition on the first attached disk and have specified the Linux partition system. Next, you need to format the disk with the XFS file system. To perform this step, use the `mkfs` command as follows:

```bash
sudo mkfs.xfs /dev/sdc1
```

*Note:* The formatting process will take a few moments to complete.
6. Use `blkid` to list the block device UUID of the new partition.

Now that the first attached disk is partitioned and formatted, you can mount it to the root file system. But to do so, you will need to copy the UUID of the associated device. To obtain the UUID of the newly partitioned device, use the `blkid` command.

   A. Enter the following command, specifying the device containing the partition you want to mount:

   ```
sudo blkid /dev/sdc1
   ```

   You will see an output similar to the following:

   ```
   /dev/sdc1: UUID="8adaf0df-84f2-45ef-ad9d-33b0382e4236" TYPE="xfs"
   ```

   B. Copy the UUID value to the clipboard.

   **Tip:** In the PuTTY utility, any text that you highlight in the SSH terminal window is automatically saved to the clipboard.

7. Enter the partition UUID and mount point in the `fstab` configuration file to automatically mount the partition on boot.

The configuration file `/etc/fstab` allows you to automatically mount partitions on boot. You can edit the `fstab` file to enter the UUID of the device and supply a desired mount point (directory) for its associated partition.

   A. To open the `fstab` for editing, type the following command:

   ```
sudo nano /etc/fstab
   ```

   The `fstab` configuration file appears:

   ```
   #
   # /etc/fstab
   # Created by anaconda on Wed Oct 31 08:36:45 2018
   #
   # Accessible filesystems, by reference, are maintained under ‘/dev/disk’
   # See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
   #
   UUID=437f5efe-afcb-4d8f-99e0-a44d73dda2f9 / xfs defaults 0 0
   UUID=f7f27132-9ca2-4de3-9a55-4a90d35fc73d /boot xfs defaults 0 0
   ```

   B. Paste the UUID of the `/dev/sdc1` device you have copied to the clipboard in a new third line below the other two UUIDs. Be sure that the format is identical to the other two entries beginning with "UUID=", so that no quotation marks appear anywhere on the line.
C. To the right of the UUID you have pasted, enter `/mnt/mssql/data`, `xfs`, `defaults`, `0`, and `0`, so that each of these five values is aligned with the associated values in the preceding line. *(For information about any of these values, see Fstab.)* After you have finished editing the file, it should appear as follows, only with different UUIDs specified:

```plaintext
#
# /etc/fstab
# Created by anaconda on Wed Oct 31 08:36:45 2018
#
# Accessible filesystems, by reference, are maintained under ‘/dev/disk’
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=437f5efe-afcb-4d8f-99e0-a44d73dda2f9 / xfs defaults 0 0
UUID=f7f27132-9ca2-4de3-9a55-4a90d35fc73d /boot xfs defaults 0 0
UUID=8ada0dfc-84f2-45ef-ad9d-33b0382e4236 /mnt/mssql/data xfs defaults 0 0
```

D. Press **Ctrl+X** to exit the nano text editor, type **y** to save the changes, and then accept the default file name.

8. Mount the partition.

Although you have configured the new partition to auto-mount to the file system on boot, it is still not currently mounted because you haven’t yet rebooted the system. Instead of rebooting, however, you can use the mount command to read the fstab file and mount the device.

A. Enter the following command to mount the first attached disk:

```bash
sudo mount /dev/sdc1
```

B. Next, verify that the new device is mounted by entering the following command:

```bash
df -h
```
You will see output similar to the following. You should see that the entry for /dev/sdc1 indicates that it is mounted on /mnt/mssql/data.

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/sda2</td>
<td>32G</td>
<td>3.8G</td>
<td>28G</td>
<td>12%</td>
<td>/</td>
</tr>
<tr>
<td>devtmpfs</td>
<td>56G</td>
<td>0</td>
<td>56G</td>
<td>0%</td>
<td>/dev</td>
</tr>
<tr>
<td>tmpfs</td>
<td>56G</td>
<td>0</td>
<td>56G</td>
<td>0%</td>
<td>/dev/shm</td>
</tr>
<tr>
<td>tmpfs</td>
<td>56G</td>
<td>9.0M</td>
<td>56G</td>
<td>1%</td>
<td>/run</td>
</tr>
<tr>
<td>tmpfs</td>
<td>56G</td>
<td>0</td>
<td>56G</td>
<td>0%</td>
<td>/sys/fs/cgroup</td>
</tr>
<tr>
<td>/dev/sdal</td>
<td>497M</td>
<td>101M</td>
<td>397M</td>
<td>21%</td>
<td>/boot</td>
</tr>
<tr>
<td>/dev/sdb1</td>
<td>221G</td>
<td>2.1G</td>
<td>208G</td>
<td>1%</td>
<td>/mnt/resource</td>
</tr>
<tr>
<td>tmpfs</td>
<td>12G</td>
<td>0</td>
<td>12G</td>
<td>0%</td>
<td>/run/user/1000</td>
</tr>
<tr>
<td>/dev/sdc1</td>
<td>1023G</td>
<td>33M</td>
<td>1023G</td>
<td>1%</td>
<td>/mnt/mssql/data</td>
</tr>
<tr>
<td>/dev/sdd1</td>
<td>1023G</td>
<td>33M</td>
<td>1023G</td>
<td>1%</td>
<td>/mnt/mssql/log</td>
</tr>
</tbody>
</table>

9. Repeat steps 4 through 8 for the log disk (/dev/sdd).

Now that you have created and formatted a partition for the data disk to be used by SQL Server 2017, you need to do the same for the log disk. To do so, you can repeat steps 4 through 8, substituting /dev/sdd for /dev/sdc, /dev/sdd1 for /dev/sdc1, and /mnt/mssql/log for /mnt/mssql/data. At the end of the process, run the following command again:

df -h

If the output matches what you see below, with a new entry for /dev/sdd1 mapped to /mnt/mssql/log, then you have completed the operating system configuration steps and can continue to the next chapter.

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/sda2</td>
<td>32G</td>
<td>3.8G</td>
<td>28G</td>
<td>13%</td>
<td>/</td>
</tr>
<tr>
<td>devtmpfs</td>
<td>56G</td>
<td>0</td>
<td>56G</td>
<td>0%</td>
<td>/dev</td>
</tr>
<tr>
<td>tmpfs</td>
<td>56G</td>
<td>0</td>
<td>56G</td>
<td>0%</td>
<td>/dev/shm</td>
</tr>
<tr>
<td>tmpfs</td>
<td>56G</td>
<td>9.0M</td>
<td>56G</td>
<td>1%</td>
<td>/run</td>
</tr>
<tr>
<td>tmpfs</td>
<td>56G</td>
<td>0</td>
<td>56G</td>
<td>0%</td>
<td>/sys/fs/cgroup</td>
</tr>
<tr>
<td>/dev/sdal</td>
<td>497M</td>
<td>101M</td>
<td>397M</td>
<td>21%</td>
<td>/boot</td>
</tr>
<tr>
<td>/dev/sdb1</td>
<td>221G</td>
<td>2.1G</td>
<td>208G</td>
<td>1%</td>
<td>/mnt/resource</td>
</tr>
<tr>
<td>tmpfs</td>
<td>12G</td>
<td>0</td>
<td>12G</td>
<td>0%</td>
<td>/run/user/1000</td>
</tr>
<tr>
<td>/dev/sdc1</td>
<td>1023G</td>
<td>33M</td>
<td>1023G</td>
<td>1%</td>
<td>/mnt/mssql/data</td>
</tr>
<tr>
<td>/dev/sdd1</td>
<td>1023G</td>
<td>33M</td>
<td>1023G</td>
<td>1%</td>
<td>/mnt/mssql/log</td>
</tr>
</tbody>
</table>
CHAPTER 5: JOINING THE SQL SERVER ON RED HAT ENTERPRISE LINUX MACHINE TO AN ACTIVE DIRECTORY DOMAIN

SQL Server 2017 includes its own built-in authentication, and it does not need to join an Active Directory domain. However, SQL Server 2017 can be added to an Active Directory domain for security benefits such as single sign-on (SSO).

Joining a SQL Server 2017 on Red Hat Enterprise Linux VM to an Active Directory domain requires a number of configuration steps. Here is a summary of those steps and a preview of the procedures that follow in this chapter:

• Change the default port for SQL Server (recommended).
• Prepare the server to join the Active Directory domain.
• Join the server to the Active Directory domain.
• Configure a service account for SQL Server 2017 on the domain controller.
• Set the service principal name for the SQL Server 2017 service account in Active Directory.
• Configure a keytab file allowing the service account to automatically sign in to the Active Directory domain at startup.

BEFORE YOU BEGIN

To begin performing the steps in this chapter, you should have:

• A provisioned SQL Server 2017 on Red Hat Enterprise Linux VM from the Azure portal that is in the same virtual network as an Active Directory domain controller.
• An SSH connection to the SQL Server on Red Hat Enterprise Linux VM.
• Domain admin credentials for the domain controller's domain.
• A connection (physical or via Remote Desktop) to a Windows 10 or Windows Server machine in the same domain.

OBJECTIVES

This chapter describes procedures that allow you to complete the following tasks:

• Join a SQL Server 2017 on Red Hat Enterprise Linux VM to an Active Directory domain.
Joining a Red Hat Enterprise Linux host to an Active Directory domain: direct and indirect integration

This chapter describes the method of directly joining a SQL Server on Red Hat Enterprise Linux host to an Active Directory domain by using an existing Active Directory domain controller, a method also known as direct integration.

However, there is an alternative, indirect method of integrating a SQL Server on Red Hat Enterprise Linux host with an Active Directory domain. With indirect integration, Linux hosts are first connected to Identity Management (IdM) in Red Hat Enterprise Linux, which is then connected to Active Directory.

More specifically, an IdM server is used to create a realm. The recommended way to perform indirect integration from the Linux realm to the Active Directory domain is through a forest-level trust between the two. This trust allows for authentication and communication between the Linux realm resources and the Active Directory domain resources. This method also allows for Linux administrators to have full control of the Linux environment without needing Windows domain admin permissions.

To read more about indirect integration, see Red Hat Enterprise Linux 7 Windows Integration Guide.

For a tutorial on setting up indirect integration, see Tutorial: Configuring Red Hat Enterprise Linux to allow Active Directory users to login to Microsoft SQL Server 2019 leveraging a trust-level setup between Red Hat Enterprise Linux Identity Management and Microsoft Active Directory.

CHANGE THE DEFAULT PORT FOR SQL SERVER (RECOMMENDED)

This procedure to improve security is not a prerequisite to join the SQL Server on Red Hat Enterprise Linux VM to the Active Directory domain, but if you plan to change the SQL Server port at some point, it’s recommended that you do so before the Active Directory join operation. If, instead, you change the SQL Server port after you join the server to the domain, you will then have to repeat some steps covered in this chapter after you do change the port.

1. Change the default port SQL Server listens on.

To change the default listening port for SQL Server, type the following command at the Bash shell on the SQL Server on Red Hat Enterprise Linux machine, substituting a chosen port number over 20000 for 65432. Be sure to record the new port number.

```
sudo /opt/mssql/bin/mssql-conf set network.tcpport 65432
```

2. To verify the mssql configuration, type the following:

```
sudo cat /var/opt/mssql/mssql.conf
```
At the bottom of the output, you should see lines similar to the following, indicating that the TCP port for SQL Server has been set to the port number you selected:

```
[network]
tcpport = 65432
```

3. Restart SQL Server.

For the new port number to be applied, you now need to restart SQL Server by typing the following command:

```
sudo systemctl restart mssql-server
```

4. Open the selected port in the operating system firewall.

After you set the SQL Server port, you have to make sure you leave the same port open in the operating system firewall. Run the following command to do so, substituting your newly chosen SQL Server port for 65432.

```
sudo firewall-cmd --zone=public --add-port=65432/tcp --permanent
```

5. To verify the port configuration, type the following command:

```
sudo firewall-cmd --list-all
```

In the output, you should see a line similar to the following, with the number listed of the open firewall port you have just configured:

```
ports: 23456/tcp 65432/tcp
```

6. Reload the operating system firewall for the changes to take effect.

Type the following command to apply the changes:

```
sudo firewall-cmd --reload
```

7. Add an inbound port rule to the Azure NSG to allow the selected SQL Server port.

Although you have just opened the required port on the operating system firewall, you still need to allow traffic through this port in the associated NSG...

```
1. In the Azure portal, navigate to the SQL Server 2017 on Red Hat Enterprise Linux VM.
2. In the navigation menu for the VM, select Networking.
3. In the Networking page that opens, click the Add inbound button.
4. In the Add Inbound Security Rule page that opens, configure the following fields as described below:
   A. In the Destination port ranges field, enter your new chosen port number for SQL Server.
   B. For the Protocol value, click TCP so it is highlighted.
   C. In the Name field, type SQL Server.
```
SQL port number

In the exercises that follow in this chapter, the port number 65432 is used merely as a placeholder for your newly chosen SQL Server port.

PREPARE THE SERVER TO JOIN THE DOMAIN

You need to perform some preparatory steps before you can join the VM to the domain. First, you need to install several packages, including realmd, which allows you to configure domain membership. Second, you need to configure DNS settings for the VM that will allow it to locate and join the domain.

Install packages used for joining the server to the domain

A number of packages need to be installed on the SQL Server 2017 on Red Hat Enterprise Linux VM before you can join it to the domain. These four packages are described below:

- **realm**: Installing this package enables you to use the realm command, which is used to perform the actual task of joining the domain.

- **krb5-workstation**: This package contains the Kerberos client for Linux.

- **samba-common-tools**: This package contains extensions to the native Samba client, which is used to support the file sharing protocol Server Message Block (SMB).

- **sssd**: This package contains the System Security Services Daemon (SSSD) that facilitates the authentication between the local client and the domain controller. Note that realmd configures SSSD, and if something goes wrong, SSSD logs should be the first place to look.

To install these packages, sign in to the SQL Server 2017 on Red Hat Enterprise Linux VM via an SSH connection, and then enter the following command at the prompt:

```
sudo yum install realmd krb5-workstation samba-common-tools sssd -y
```

Installed packages

To view the installed packages on your Red Hat Enterprise Linux VM, type the following:

```
sudo yum list installed
```

Edit the network configuration to set the domain controller as the primary DNS server

Perform the following steps to set the domain controller as the primary DNS server in the SQL Server 2017 on Red Hat Enterprise Linux VM. To complete this procedure, you will need to obtain the IP address of the Active Directory domain controller.
Using the nmcli utility

As an alternative to the procedure shown below, you can also use the nmcli utility to configure the DNS server, hostname, DNS suffix, and other network settings. For more information on nmcli, see Using the NetworkManager command line tool, nmcli.

1. Enter the following command to open the server’s network configuration file in a text editor.

   ```bash
   sudo nano /etc/sysconfig/network-scripts/ifcfg-eth0
   ```

2. Edit the network configuration file in the text editor.

   A. Look for a line that begins "PEERDNS=" in the configuration file.

      i. If a configuration setting appears as `PEERDNS=yes` in the configuration file, then change it to `PEERDNS=no`.

      ii. If a "PEERDNS=" setting does not yet exist, add the following as a new line of text to the bottom of the configuration file:

         ```
         PEERDNS=no
         ```

   B. Next, add the following line to the end of the configuration file, substituting the IP address of a DNS server in the Active Directory domain (which is typically any domain controller) for **<DNS server IP address>**.

      ```
      DNS1=**<DNS server IP address>**
      ```

      For example, if the IP address of the nearest accessible domain controller in the target domain is 10.0.0.8, you should edit the configuration file so that the values assigned to PEERDNS and DNS1 appear as follows:

      ```
      PEERDNS=no
      DNS1=10.0.0.8
      ```

   After you have finished editing the network-configuration file, exit and save the changes to the file.

Set the appropriate DNS suffix for the machine

To prepare the VM to join the Active Directory domain, you should set its DNS suffix so that it matches the name of the Active Directory domain. At the Bash prompt on the SQL Server 2017 on Red Hat Enterprise Linux VM, type the following, substituting the host name of the machine for `[hostname]` and the name of the Active Directory domain for `[domain name]`.

```bash
sudo hostnamectl set-hostname [hostname].[domain name]
```

For example, if the host name is SQLRhel1 and the Active Directory domain name is contoso.com, you would type the following:

```bash
sudo hostnamectl set-hostname SQLRhel1.contoso.com
```

You can then use the hostname command to verify that a fully-qualified domain name (FQDN) is set:

```bash
hostname
```
Restart the network for the changes to take effect

The new settings that you have just configured will apply after you restart the network. Enter the following command:

```
sudo systemctl restart network
```

Then, to verify the new DNS server setting, type the following:

```
sudo cat /etc/resolv.conf
```

You should see the following output, with the IP address you have specified in place of 10.0.0.8:

```
# Generated by NetworkManager
nameserver 10.0.0.8
```

**JOIN THE SERVER TO THE DOMAIN**

In the following procedure, you will use the realm command to join the local machine to the domain and then verify the configuration.

Run the realm command

Run the following command to join the SQL Server on Red Hat Enterprise Linux VM to your Windows domain, substituting the name of your domain for *contoso.com* and the name of a domain user account for *user@CONTOSO.COM*. (Include the single quotation marks.)

```
sudo realm join contoso.com -U 'user@CONTOSO.COM' --v
```

Provide the password for the user account when prompted.

Verify the new configuration in the Kerberos client configuration file

After you receive a message indicating that the machine is successfully enrolled in the specified realm (domain), you can verify the new configuration of the VM by typing the following:

```
cat /etc/krb5.conf
```

Look for an uncommented line that looks like the following, only with the name of your domain specified in place of *CONTOSO.COM*:

```
default_realm = CONTOSO.COM
```

Verify the DNS configuration through nslookup

At the Bash prompt, type the following, substituting the host name of your SQL Server 2017 on Red Hat Enterprise Linux VM for *SqlRhel1*.

```
nslookup SqlRhel1
```
You should receive output similar to the following. Look to the bottom of the output to find the new FQDN, along with the IP address of the SQL Server 2017 on Red Hat Enterprise Linux VM, verifying that the VM has been registered in the appropriate forward lookup zone hosted on the DNS server. In the case of the output shown below, the DNS server address is 10.0.0.2 and the address of the SQL Server 2017 on Red Hat Enterprise Linux VM is 10.0.0.5.

<table>
<thead>
<tr>
<th>Server:</th>
<th></th>
<th>Address:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10.0.0.2</td>
<td>10.0.0.2#53</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name:</th>
<th>SQLRhel1.contoso.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address:</td>
<td>10.0.0.5</td>
</tr>
</tbody>
</table>

Verify that the new computer account exists in the domain

Perform the following steps on a Windows 10 or Windows Server computer joined to the domain. (If you do not perform this step on a domain controller, ensure that the computer is connected to the domain network.)

1. Open a PowerShell prompt.
   A. In the search box next to the start menu, type `powershell`.
   B. In the Best match list that appears, click `Windows PowerShell`. A PowerShell window appears.

2. At the PowerShell prompt, type the `get-adcomputer` command.

Use the `get-adcomputer` command to verify the presence of a computer account for the server in the Active Directory domain. Substitute the name of the SQL Server 2017 on Red Hat Enterprise Linux VM for `[computer name]`. The name is case-sensitive. Do not specify a domain name with the command.

```
get-adcomputer [computer name]
```

For example, if the name of the VM is SQLRhel1, then you want to type the following:

```
get-adcomputer SQLRhel1
```

You should see output similar to the following, which indicates that a computer account for the machine has been added to the Active Directory domain.

<table>
<thead>
<tr>
<th>DistinguishedName:</th>
<th>CN=SQLRHEL1,CN=Computers,DC=fabrikam,DC=com</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNSHostName:</td>
<td>sqlrhel1.fabrikam.com</td>
</tr>
<tr>
<td>Enabled:</td>
<td>True</td>
</tr>
<tr>
<td>Name:</td>
<td>SQLRHEL1</td>
</tr>
<tr>
<td>ObjectClass:</td>
<td>computer</td>
</tr>
<tr>
<td>ObjectGUID:</td>
<td>c03221e8-1367-4320-b8a6-9fc611355436</td>
</tr>
<tr>
<td>SamAccountName:</td>
<td>SQLRHEL1$</td>
</tr>
<tr>
<td>SID:</td>
<td>S-1-5-21-2093534575-1107673975-3226666838-1103</td>
</tr>
<tr>
<td>UserPrincipalName:</td>
<td></td>
</tr>
</tbody>
</table>
Tip: If you don’t remember the computer name of the machine, you can type `hostname` at the Bash shell prompt on the Linux machine.

Verify that a user can obtain a Kerberos ticket from the domain controller

Now that you have joined the machine to the domain, you need to verify that Kerberos is properly functioning. To do so, you will attempt to obtain a Kerberos ticket for a user. You will then display the contents of the Kerberos ticket cache to verify that the ticket was obtained.

1. Run the `kinit` command to obtain a Kerberos ticket.

To obtain and cache a Kerberos ticket for an Active Directory user, type the following command, substituting the name of an Active Directory user account for `user@CONTOSO.COM`.

```
Important
Linux is case-sensitive. Make sure that you put the domain in all caps, as shown in the example.
```

```
kinit user@CONTOSO.COM
```

Supply the user password when prompted.

2. Verify that the Kerberos ticket was acquired and saved in the local cache.

Now you can type the following command to display the cache of saved Kerberos tickets.

```
klist
```

You should see an output similar to the following, indicating that a ticket was obtained and saved:

```
Ticket cache: KEYRING:persistent:1000:1000
Default principal: user@CONTOSO.COM

Valid starting       Expires               Service principal
03/14/2019 04:58:09  03/14/2019 14:58:09  krbtgt/CONTOSO.COM@CONTOSO.COM
                renew until 03/21/2019 04:58:01
```

3. You should also verify that the identity information is properly pulled from Active Directory and that all the user groups are fetched as expected. To do so, type the following command:

```
id user@CONTOSO.COM
```
You should see output similar to the following, indicating that identity and group information has been retrieved from Active Directory:

```
uid=1737200500(usr@Contoso.com) gid=1737200513(domain users@Contoso.com)
groups=1737200513(domain users@Contoso.com),1737200512(domain admins@Contoso.com),1737200572(denied rodc password replication group@Contoso.com),1737200519(enterprise admins@Contoso.com),1737200518(schema admins@Contoso.com),1737200520(group policy creator owners@Contoso.com)
```

Test the configuration: Allow an Active Directory user to sign in to the SQL Server on Red Hat Enterprise Linux VM

You can demonstrate that the SQL Server on Red Hat Enterprise Linux host is joined to an Active Directory domain by having an Active Directory user from that domain sign in to the machine. To do so, you first need to grant an Active Directory user access.

1. Type the following command, substituting the name of a suitable Active Directory user account for user@contoso.com.

```
sudo realm permit user@contoso.com
```

2. Now sign in to the local machine with that same domain user account by typing the following command, once again substituting the name of your Active Directory user account for user@contoso.com.

```
su user@contoso.com
```

3. When prompted, provide the password for the user to finish the process of signing in.

You should see a new prompt that resembles the following, indicating that the Active Directory user has signed in:

```
[user@contoso.com@sqlrhel1 user]$
```

This command demonstrates that the domain join operation for the local machine has succeeded.

4. Type `exit` to return to the prompt for the local user account.
CONFIGURE A SERVICE ACCOUNT FOR SQL SERVER 2017 ON THE
DOMAIN CONTROLLER

In this procedure, you will create a service account (a user account intended for a service) in Active
Directory that the SQL Server 2017 service will use to sign in to the domain. This service account
provides a security context for services in Active Directory and determines the service's ability to
access local and network resources.

Create a service account for SQL Server 2017

To create a service account for SQL Server 2017, complete the following procedure on a Windows 10
or Windows Server machine that is joined to the domain. You will require domain admin privileges to
perform this step.

1. Open a PowerShell prompt with administrator privileges.
   A. In the search box next to the start menu, type powershell.
   B. In the Best match list that appears, right-click Windows PowerShell, and then click Run as
      administrator. A PowerShell window appears.

2. At the PowerShell prompt, type the following command to create a new account named "mssql":

   ```powershell
   New-ADUser mssql -AccountPassword (Read-Host -AsSecureString "Enter Password")
   -PasswordNeverExpires $true -Enabled $true
   ```

   When prompted, enter a password of your choice that meets the password requirements for the
domain. Note that the "-PasswordNeverExpires" flag sets the password to never expire.

   For more information about the New-ADUser cmdlet, see New-ADUser.

SET THE SERVICE PRINCIPAL NAME (SPN) FOR THE SQL SERVER 2017
SERVICE ACCOUNT

To use Kerberos authentication with SQL Server 2017 requires that a service principal name (SPN)
for the service be registered with Active Directory. SPNs are used by Kerberos authentication to
associate an instance of a service, such as a particular server running SQL Server 2017, with a
service account, such as the one you have just created.

For more information about SPNs, see Service Principal Names and Register a Service
Principal Name for Kerberos Connections.

To create an SPN for the SQL Server 2017 service, complete the following procedure on a Windows
10 or Windows Server machine that is joined to the domain. You will require domain admin privileges
to perform this step.

1. Open a command prompt with administrator privileges.
   A. In the search box next to the start menu, type cmd.
   B. In the Best match list that appears, right-click Command Prompt, and then click Run
      as administrator.
2. Use the `setspn` command to create an SPN for the SQL Server 2017 service account.

Enter the following command on your domain controller to create the SPN for the SQL Server 2017 service account (mssql), substituting the FQDN of the SQL Server 2017 on Red Hat Enterprise Linux VM for `[FQDN]` and your newly chosen SQL Server port for `[SQLServerPort]`.

```
setspn -a MSSQLSvc/[FQDN]:[SQLServerPort] mssql
```

For example, if the FQDN of the SQL Server 2017 on Red Hat Enterprise Linux VM is SQLRhel1.contoso.com and the configured SQL Server port is 65432, you would type the following:

```
setspn -a MSSQLSvc/SQLRhel1.contoso.com:65432 mssql
```

You should see output similar to the following, indicating that the operation was successful:

```
Checking domain DC=contoso,DC=com
Registering ServicePrincipalNames for CN=mssql,CN=Users,DC=contoso,DC=com
MSSQLSvc/SQLRhel1.contoso.com:65432
Updated object
```

3. Verify that the SPN is registered.

To verify that the SPN is registered for the VM, run the following command from the command prompt, substituting the appropriate host name for `SQLRhel1`.

```
setspn -L SQLRhel1
```

You should see output similar to the following:

```
Registered ServicePrincipalNames for CN=SQLRHEL1,CN=Computers,DC=contoso,DC=com:
    HOST/sqlrhel1.contoso.com
    HOST/SQLRHEL1
```

Now verify that the SPN for SQL Server is associated with the mssql service account by typing the following command:

```
setspn -L mssql
```

You should see output similar to the following, indicating that the SQL Server service is properly registered to its service account:

```
Registered ServicePrincipalNames for CN=mssql,CN=Users,DC=fabrikam,DC=com:
    MSSQLSvc/SQLRhel1.contoso.com:65432
```

**CONFIGURE A KEYTAB FILE TO ALLOW THE SERVICE ACCOUNT TO AUTOMATICALLY SIGN IN TO THE ACTIVE DIRECTORY DOMAIN AT BOOT**

You have now created a service account and an associated SPN for the SQL Server 2017 service. However, you still need to configure the service account to sign in to the Active Directory domain automatically whenever the SQL Server 2017 on Red Hat Enterprise Linux VM boots. To do so, complete the following steps.
Obtain a current Kerberos ticket for a newly created service account

On the SQL Server 2017 on Red Hat Enterprise Linux VM, type the following command at the Bash prompt to receive a current ticket for the newly created service account, mssql. Replace CONTOSO.COM with the name of your Active Directory domain, and be sure to use all caps when specifying the domain, as in this example:

```
kinit mssql@CONTOSO.COM
```

You can then verify that a ticket was obtained and cached by typing the following command at the Bash prompt:

```
klist -A
```

Within the output, you should see a section similar to the following, indicating that a Kerberos ticket for mssql was obtained:

```
Ticket cache: KEYRING:persistent:1000:krb_ccache_5MbknxK
Default principal: mssql@CONTOSO.COM

Valid starting       Expires              Service principal
03/17/2019 21:03:46  03/18/2019 07:03:46  krbtgt/CONTOSO.COM@CONTOSO.COM
renew until 03/24/2019 21:03:41
```

Use the kvno command to check the version number of the Kerberos ticket received

Run the following command to check the version of the ticket received, substituting the FQDN of your SQL Server 2017 on Red Hat Enterprise Linux VM for SQLRhel1.CONTOSO.COM and the newly chosen SQL Server port for 65432. Record the key version number ("kvno") displayed in the output because you will need to specify it later, when you create the keytab file.

```
kvno MSSQLSvc/SQLRhel1.contoso.com:65432
```

You should see output similar to the following. The version number appears at the end. In the particular case whose output is displayed below, the key version number is 3.

```
MSSQLSvc/SQLRhel1.contoso.com:65432@contoso.COM: kvno = 3
```

Create a keytab file to enable the service account mssql to automatically authenticate to the domain at boot

A keytab is a text file that maps a user account or service account to an encrypted hash of that account’s password. Keytab files allow a service running in Linux to automatically sign in to Active Directory on startup. In the keytab file, you must specify the associated SPN for the service and the key version number obtained in the last step.

For more information about keytabs, see Active Directory: Using Kerberos Keytabs to integrate non-Windows systems.

1. Use the ktutil utility to create the keytab file.

   A. To begin the process of creating the keytab file, type the following command at the Bash prompt on your SQL Server 2017 on Red Hat Enterprise Linux VM:

   ```
sudo ktutil
```
B. Now, create and add key entries to a new mssql.keytab file. At the ktutil prompt, type the following command, substituting the FQDN of your SQL Server 2017 on Red Hat Enterprise Linux VM for SQLRhel1.contoso.com, your Active Directory domain for CONTOSO.COM, your newly chosen port for 65432, and the key version number obtained above for [key version number].

```
addent -password -p MSSQLSvc/SQLRhel1.contoso.com:65432@CONTOSO.COM -k [key version number] -e aes256-cts-hmac-sha1-96
```

When prompted, provide the password you assigned the service account. After this step is complete, you will have added a cryptographic hash of the password using the algorithm specified.

C. Then, again at the ktutil prompt, type the following command, substituting the FQDN of your SQL Server 2017 on Red Hat Enterprise Linux VM for SQLRhel1.contoso.com, your Active Directory domain for CONTOSO.COM, the newly chosen SQL Server port for 65432, and the key version number obtained above for [key version number].

```
addent -password -p MSSQLSvc/SQLRhel1.contoso.com:65432@CONTOSO.COM -k [key version number] -e rc4-hmac
```

Once again, provide the password when prompted. After this step is complete, the command will have added a second cryptographic hash of the password. Both of these cryptographic hashes are used by Active Directory.

D. Now, you need to write (append) the changes you’ve made to a new keytab file for the mssql account. Type the following at the ktutil prompt:

```
wkt /var/opt/mssql/secrets/mssql.keytab
```

2. Append the local computer account keys from the krb5.keytab file to mssql.keytab.

By now, you have created a new keytab file named mssql.keytab, which includes encrypted keys needed for the mssql service account to authenticate to the domain automatically. However, the same mssql.keytab file also needs to include keys allowing the Active Directory computer account of the local machine to authenticate to the domain. You will obtain this information from an existing keytab file called krb5.keytab, which is generated by the system.

To get the information you need in krb5.keytab, you will read the contents of this file into the ktutil keylist and edit its contents so that only the entries corresponding to the Active Directory computer account for the local machine are visible. Then you will append those entries for the Active Directory computer account to your mssql.keytab file without affecting the original contents of the krb5.keytab file.

A. To load the contents of the krb5.keytab into the ktutil keylist, type the following command at the ktutil prompt:

```
rkt /etc/krb5.keytab
```

B. Next, type the following command at the ktutil prompt to display the contents of the keylist:

```
list
```
You will see output similar to the following:

<table>
<thead>
<tr>
<th>slot</th>
<th>KVNO</th>
<th>Principal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>MSSQL/SQLRhel1.contoso.com:<a href="mailto:65432@contoso.com">65432@contoso.com</a></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>MSSQL/SQLRhel1.contoso.com:<a href="mailto:65432@contoso.com">65432@contoso.com</a></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>host/sqlrhel1.contoso.com@CONTOSO.COM</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>host/SQLRHEL1@CONTOSO.COM</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>host/sqlrhel1.contoso.com@CONTOSO.COM</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>host/SQLRHEL1@CONTOSO.COM</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>host/sqlrhel1.contoso.com@CONTOSO.COM</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>host/SQLRHEL1@CONTOSO.COM</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>host/sqlrhel1.contoso.com@CONTOSO.COM</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>host/SQLRHEL1@CONTOSO.COM</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>host/sqlrhel1.contoso.com@CONTOSO.COM</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>host/SQLRHEL1@CONTOSO.COM</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
</tbody>
</table>

C. Use the `delent` command to delete each key entry in the keylist, one by one, that does **not** correspond to the User Principal Name (UPN) of the local machine. (A UPN for a computer account appears in the format "computername$@domain.com.") In the output above, for example, the computer account entries that you should **not** delete from the keylist are found in slots 13 through 17. All others should be deleted. To delete an entry, type the following command at the ktutil prompt, substituting a particular slot number to delete for `[slot number]`.

```
delent [slot number]
```

Be sure to type the list command at the ktutil prompt as needed to view how the slot numbers have changed:

```
list
```
At the end of the process, the output generated by the list command should be similar to the following:

```
ktutil:  list
slot  KVNO  Principal
--  ----  -----------------------------------------------
1    2     SQLRHEL1$@CONTOSO.COM
2    2     SQLRHEL1$@CONTOSO.COM
3    2     SQLRHEL1$@CONTOSO.COM
4    2     SQLRHEL1$@CONTOSO.COM
5    2     SQLRHEL1$@CONTOSO.COM
```

D. When the keylist includes keys only for the local machine’s UPN, append the keylist to the mssql.keytab file by typing the following command at the ktutil prompt:

```
wkt /var/opt/mssql/secrets/mssql.keytab
```

E. To verify that the keys have been written to the mssql.keytab file, first type the following to read its contents into the ktutil utility:

```
rkt /var/opt/mssql/secrets/mssql.keytab
```

F. Then display the contents of the keylist:

```
list
```
You should see output similar to the following:

<table>
<thead>
<tr>
<th>slot</th>
<th>KVNO</th>
<th>Principal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>MSSQLSvc/sqlrhel1.contoso.com:<a href="mailto:65432@CONTOSO.COM">65432@CONTOSO.COM</a></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>MSSQLSvc/sqlrhel1.contoso.com:<a href="mailto:65432@CONTOSO.COM">65432@CONTOSO.COM</a></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>SQLRHEL1$@CONTOSO.COM</td>
</tr>
</tbody>
</table>

G. Now that you have finished configuring the mssql.keytab file, you can exit the ktutil tool by typing **quit** at the ktutil prompt:

```
quit
```

Configure the owner and permission settings for the mssql.keytab file

Perform the following steps to change the ownership and permission settings on the file.
1. Change the owner of the mssql.keytab file to the mssql account.

Type the following command to set the mssql service account as the owner of the mssql.keytab file:

```
sudo chown mssql:mssql /var/opt/mssql/secrets/mssql.keytab
```

2. Change the SQL Server 2017 keytab file to read access only, to keep it from being modified.

Type the following command to ensure that the contents of the mssql.keytab file will not be modified:

```
sudo chmod 400 /var/opt/mssql/secrets/mssql.keytab
```
Configure SQL Server 2017 to use the keytab file for Kerberos authentication

To complete the procedure for allowing the service account to automatically sign in to the domain at boot, perform the following procedure.

1. Use mssql-conf to assign mssql.keytab as the keytab file for SQL Server 2017.

Type the following command to configure SQL Server 2017 to use the keytab file you have just configured:

```
sudo /opt/mssql/bin/mssql-conf set network.kerberoskeytabfile /var/opt/mssql/secrets/mssql.keytab
```

2. To verify that the SQL Server configuration is set to use the keytab file, type the following at the Bash prompt.

```
sudo cat /var/opt/mssql/mssql.conf
```

At the bottom of the configuration file, you should see the following line in the `[network]` section, indicating that SQL Server is set to use the keytab file:

```
kerberoskeytabfile = /var/opt/mssql/secrets/mssql.keytab
```

3. Restart SQL Server 2017 for those changes to take effect.

Type the following command to restart the SQL Server 2017 service on the local machine and adopt the new changes.

```
sudo systemctl restart mssql-server
```

4. To verify mssql-server restarts successfully, type the following command:

```
sudo systemctl status mssql-server
```

By now, you’ve taken significant steps toward enabling the SQL Server service and Active Directory users to authenticate each other through the Kerberos protocol. However, you need to perform some additional steps in SQL Server (such as installing SQL Server tools and setting a password for the SQL Server sa user account) before you can attempt to have an Active Directory user sign in to it. Proceed to Chapter 6 to complete these and other configuration steps for SQL Server.

**FURTHER READING**

*To find out more about integrating Red Hat Enterprise Linux into an Active Directory environment, see* [Integrating Linux Systems with Active Directory Environments](#)
CHAPTER 6: CONFIGURING SQL SERVER 2017 ON
RED HAT ENTERPRISE LINUX

In this chapter, you will set up the SQL Server 2017 in a Red Hat Enterprise Linux environment. This
setup includes the following steps:

• Changing the default port for SQL Server 2017 (if it is not already changed)
• Preparing the environment for SQL Server 2017 administration
• Configuring locations for database and log files
• Importing a database

BEFORE YOU BEGIN
You need to complete certain configuration steps before you begin performing the steps described in
this chapter. These configuration steps include the following:

• If your VM running SQL Server 2017 on Red Hat Enterprise Linux is planned for Active Directory
domain membership:
  • Complete all steps through Chapter 5 before beginning this chapter.
• If your VM running SQL Server 2017 on Red Hat Enterprise Linux is not planned for Active Directory
domain membership:
  • Complete all steps through Chapter 4 before beginning this chapter.

OBJECTIVES
This chapter describes procedures that allow you to complete the following tasks:

• Install SQL Server 2017 command-line utilities.
• Enable an administrator to use SQL Server 2017 utilities without explicitly specifying their paths.
• Set the SQL Server 2017 administrator password.
• Configure default locations on separate disks for data and log files.
• Change the location of the TempDB database.
• Enable an Active Directory user to connect to SQL Server 2017.
• Import a database.

CHANGE THE DEFAULT PORT FOR SQL SERVER 2017 (IF IT IS NOT
ALREADY CHANGED)

If you skipped Chapter 5 because your SQL Server 2017 on Red Hat Enterprise Linux VM does
not need to join an Active Directory domain, then you missed the important step of changing the
default port used by SQL Server 2017 to improve security. If you have not yet changed the default
port (TCP port 1433) for SQL Server 2017, return to Chapter 5 now and perform the first procedure
listed, “Change the default port for SQL Server 2017 (Recommended),” and then return to perform the
remaining steps in this chapter.
If you joined the SQL Server 2017 on Red Hat Enterprise Linux VM to an Active Directory domain without changing the default SQL Server port, but you now want to change the default SQL Server 2017 port, then you need to complete the following steps in order:

1. Change the default port according to the steps listed in Chapter 5 in the section “Change the default port for SQL Server 2017 (recommended).”

2. Recreate the service principal name for the mssql service account you created in Chapter 5 by repeating the steps in the section “Set the service principal name (SPN) for the SQL Server 2017 service account.”

3. Next, while you are still signed in to a Windows computer as a domain administrator, you should delete the former SPN registered to mssql that specified the old port. Type the following, substituting the FQDN of the SQL Server 2017 on Red Hat Enterprise Linux VM for SQLRhel1.contoso.com:

   ```
   setspn -D MSSQLSvc/SQLRhel1.contoso.com:1433 mssql
   ```

4. Delete the mssql.keytab file you created in Chapter 5 by entering the following command at the Bash prompt:

   ```
   sudo rm /var/opt/mssql/secrets/mssql.keytab
   ```

5. Recreate the mssql.keytab file by repeating all of the steps in the Chapter 5 section named “Configure a keytab file to allow the service account to automatically sign in to the Active Directory domain at boot.”

**PREPARE THE ENVIRONMENT FOR SQL SERVER 2017 ADMINISTRATION**

The following sections describe the process of installing command line tools, configuring the PATH environment variable to run these tools without having to specify a path, and finally setting the password for the system admin (sa) account.

**Install SQL Server 2017 command-line tools and the Microsoft ODBC driver for SQL Server 2017 on Linux**

To begin preparing SQL Server 2017, you need to install packages that will later let you use important SQL Server 2017 administration tools on the machine, such as sqlcmd, bcp, and mssql-cli. You also need to install the Microsoft Open Database Connectivity (ODBC) driver that will allow you to connect the server to most data sources.

To install sqlcmd, bcp, and the ODBC driver, type the following command at the Bash prompt:

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

Whenever prompted, type **yes** to accept the license terms and continue the installation.
Install mssql-cli

mssql-cli is an interactive command-line tool for querying SQL Server. Use the following procedure to install and start the tool:

```
# Import Microsoft repository key
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc

# Register the Microsoft RedHat repository
curl https://packages.microsoft.com/config/rhel/7/prod.repo | sudo tee /etc/yum.repos.d/microsoft.repo

# Install mssql-cli
sudo yum install mssql-cli

# Start mssql-cli
mssql-cli
```

Azure Data Studio for SQL Server administration

SQL Server 2017 on Red Hat Enterprise Linux can be managed from GUI tools or from command-line tools. Azure Data Studio is a lightweight, cross-platform GUI tool that can be used to manage SQL Server. For more information, see the section “Connect to SQL Server 2017 from Azure Data Studio,” later in this chapter.

Allow SQL Server 2017 command-line tools to be run without specifying a path

Type the following command at the Bash shell to edit the bashrc configuration file and make SQL Server 2017 command-line tools accessible from the Bash shell without having to explicitly specify a path:

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

To verify this was successfully added you can read out the bashrc file by:

```
cat ~/.bashrc
```

Use the source command to load the current shell settings

Now type the following to load the new settings for the PATH environment variable into the current session:

```
source ~/.bashrc
```
Set the SQL Server 2017 system admin (SA) password

You now need to assign a password to the SQL Server 2017 system administrator account, SA. To do so, begin by typing the following command to stop the SQL Server 2017 service:

```bash
sudo systemctl stop mssql-server
```

When the prompt reappears, type the following command to set the SA password. Enter and confirm a password of your choice, and be sure to record it.

```bash
sudo /opt/mssql/bin/mssql-conf set-sa-password
```

Now run the following command to start SQL Server 2017 again:

```bash
sudo systemctl start mssql-server
```

**Forgot the sa password?**

If the sa password is entered incorrectly more than three times, then the account will be locked.

To reset the sa password, see [SQL Server on Linux: How to Change SA password](#).

---

**CONFIGURE DEFAULT LOCATIONS FOR DATABASE AND LOG FILES**

Following the recommended design for SQL Server 2017 installations, you have attached two additional disks to be used for log and data files (including TempDB). You have also mounted the first of these attached disks to the /mnt/mssql/log directory in the root file system, and the second of these disks to the /mnt/mssql/data directory.

Next, you will configure SQL Server 2017 so that new log and data files are created and stored on the appropriate disks by default.

1. **Set the log disk as the default location for storing log files.**

Type the following command to assign log files to the disk that you have dedicated to them:

```bash
sudo /opt/mssql/bin/mssql-conf set filelocation.defaultlogdir /mnt/mssql/log
```

2. **Set the data disk as the default location for newly created databases.**

Now, assign the default location for new databases to the disk that you have dedicated to data.

```bash
sudo /opt/mssql/bin/mssql-conf set filelocation.defaultdatadir /mnt/mssql/data
```

3. **Verify the configuration changes.**

```bash
sudo cat /var/opt/mssql/mssql.conf
```

You should see the following section in the output, indicating the new locations for the database and log files:

```bash
[filelocation]
defaultlogdir = /mnt/mssql/log
defaultdatadir = /mnt/mssql/data
```
4. **Restart the SQL Server 2017 service for the changes to take effect.**

Run the following command to restart the SQL Server 2017 service and commit these new settings:

```bash
sudo systemctl restart mssql-server
```

**CHANGE THE LOCATION OF TEMPDB**

Following recommended SQL Server 2017 design, you now want to move the TempDB database to the data disk. To perform this transfer, you need to use T-SQL statements. For this, you can use the `sqlcmd` utility.

1. **Connect to the database server by using the sqlcmd utility.**

Run the following command to connect to the local installation of SQL Server 2017, substituting your configured SQL Server port number for `65432`:

```bash
sqlcmd -S localhost,65432 --U sa
```

The `-S` option here is used to specify a server, and the `-U` option specifies the SQL Server 2017 sign in to use. Provide the SA password when prompted. After you are authenticated, you should see a new prompt that looks like the following:

```
>1
```

The "1" signifies that this is the first line of a T-SQL statement, and the `sqlcmd` prompt is the point at which the T-SQL statement will start when you type it in.

2. **Verify that you can query data from this SQL Server 2017 instance.**

Type the following T-SQL statements at the `sqlcmd` prompt, and then press Enter:

```
1> SELECT @@VERSION
2> GO
```

You should see output similar to the following:

```
Microsoft SQL Server 2017 (RTM-CU13) (KB4466404) - 14.0.3048.4 (X64)
 Nov 30 2018 12:57:58
 Copyright (C) 2017 Microsoft Corporation
 Enterprise Edition (64-bit) on Linux (Red Hat Enterprise Linux Server 7.6 (Maipo))

(1 rows affected)
1>
```

*It is recommended that your log files and data files (including TempDB) be stored on separate disks.*
3. **Use T-SQL to change the location of the TempDB database.**

Run the following T-SQL statement at the sqlcmd prompt to create a new tempdb.mdf file in /mnt/mssql/data:

```
ALTER DATABASE tempdb MODIFY FILE
(NAME = tempdev, FILENAME = '/mnt/mssql/data/tempdb.mdf', SIZE=256)
GO
```

Output should appear indicating that the file "tempdev" has been modified in the system catalog, and that the new path will be used the next time the database is started.

4. **Use T-SQL to change the location of the TempDB log file.**

Now, run the following T-SQL statement at the sqlcmd prompt to create a new TempDB log file in /mnt/mssql/log:

```
ALTER DATABASE tempdb MODIFY FILE
(NAME = templog, FILENAME = '/mnt/mssql/log/templog.ldf', SIZE=256)
GO
```

You should receive output indicating that the file "tempdev" has been modified in the system catalog, and that the new path will be used the next time the database is started. Before you restart, however, you can verify that the new settings are configured properly.

5. **Use T-SQL to verify the new location of the TempDB data and log files.**

Enter and run the following T-SQL statement at the sqlcmd prompt to verify the new file locations for TempDB:

```
SELECT name, physical_name AS CurrentLocation, state_desc
FROM sys.master_files
WHERE database_id = DB_ID(N'tempdb');
```

GO
You should see output similar to the following. In the lower half of the output, you'll see that the correctly configured paths to tempdb.mdf and templog.ldf are specified.

<table>
<thead>
<tr>
<th>name</th>
<th>CurrentLocation</th>
<th>state_desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>tempdev</td>
<td>/mnt/mssql/data/tempdb.mdf</td>
<td>ONLINE</td>
</tr>
<tr>
<td>templog</td>
<td>/mnt/mssql/log/templog.ldf</td>
<td>ONLINE</td>
</tr>
</tbody>
</table>

(2 rows affected)

Apply the changes and delete the original TempDB data and log files

You have created files for the TempDB database and log file in new locations, but you still need to apply those changes, in addition to deleting the TempDB database and log files in the original locations. To do so, you first need to return to the Bash shell. Type `quit` to exit the sqlcmd utility:

```
quit
```

Type the following to restart SQL Server 2017 and apply the new changes. Restarting SQL Server 2017 at this point also deletes the original log file.

```
sudo systemctl restart mssql-server
```

To verify that the TempDB files were recreated in the new location you can list the contents of those directories.

```
sudo ls /mnt/mssql/data
```

```
sudo ls /mnt/mssql/log
```

Once you have verified that the TempDB was recreated in the new location, you can remove the original TempDB files, which are located in the following directory: `/var/opt/mssql/data`. Type the following command to list the contents of that directory:

```
sudo ls /var/opt/mssql/data/
```
To remove any remaining TempDB files, type the following commands:

```bash
sudo rm /var/opt/mssql/data/templog.ldf
sudo rm /var/opt/mssql/data/tempdb.mdf
```

**TEST THE CONFIGURATION: ALLOW AN ACTIVE DIRECTORY USER PERMISSION TO CONNECT TO SQL SERVER 2017**

With the SA password set and the sqlcmd utility installed, you're almost ready to verify the final configuration steps performed in Chapter 5 by having an Active Directory user authenticate and connect to SQL Server 2017. Note, however, that still at this point, the only way to connect to the SQL Server 2017 instance is by using the SQL Server 2017 SA account.

To allow selected Active Directory users to connect to SQL Server 2017, you first need to create at least one SQL Server 2017 account based on a selected Active Directory user or group account. According to best practices, it's best to assign permissions to groups instead of to individual users, so you'll first create an Active Directory group named SQLUsers for which you will then create a SQL Server account.

Create an Active Directory group for users who need to sign in to the SQL Server 2017 on Red Hat Enterprise Linux host machine

To create a new Active Directory group named, for example, "SQLUsers," open an elevated PowerShell prompt on a Windows 10 or Windows Server computer joined to the domain, and then type the following command:

```powershell
New-ADGroup -Name "SQLUsers" -GroupScope Universal
```

You can then verify that the group is created by ensuring that no error is generated when you type the following command:

```powershell
Get-ADGroup SQLUsers
```

After verifying that the group has been created, you now need to add all appropriate users to the SQLUsers group. For example, to add the users SQLUser1@contoso.com and SQLUser2@contoso.com to this group, type the following command at the PowerShell prompt from the same domain-joined computer:

```powershell
Add-ADGroupMember -Identity SQLUsers -Members SQLUser1,SQLUser2
```

The following command will display the membership of the SQLUsers group. You can now use it to verify that the new accounts have been added:

```powershell
Get-ADGroupMember -Identity SQLUsers
```
You should see output similar to the following, indicating that the new members have been added to the group.

<table>
<thead>
<tr>
<th>distinguishedName</th>
<th>name</th>
<th>objectClass</th>
<th>objectGUID</th>
<th>SamAccountName</th>
<th>SID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN=SQLUser2,CN=Users,DC=contoso,DC=com</td>
<td>SQLUser2</td>
<td>user</td>
<td>f42d65ee-6175-4449-89e9-284c75da23df</td>
<td>SQLUser2</td>
<td>S-1-5-21-2093534575-1107673975-3226666838-1606</td>
</tr>
<tr>
<td>CN=SQLUser1,CN=Users,DC=contoso,DC=com</td>
<td>SQLUser1</td>
<td>user</td>
<td>6d12ccc8-fd33-4f7e-b671-63d0253be154</td>
<td>SQLUser1</td>
<td>S-1-5-21-2093534575-1107673975-3226666838-1605</td>
</tr>
</tbody>
</table>

Create a SQL Server account for an Active Directory group

Now that the Active Directory group has been created and contains the selected Active Directory users, you need to create a SQL Server account for the group. To do so, return to the command prompt on the SQL Server on Red Hat Enterprise Linux host. Use the sqlcmd utility again to connect to the local instance of SQL Server 2017 as the user SA. Type the following command, replacing your newly chosen SQL Server 2017 port for 65432:

```
sqlcmd -S localhost,65432 -U sa
```

When the sqlcmd prompt reappears, run the following T-SQL statement, substituting the name of your domain for Contoso and the name of the Active Directory group for SQLUsers:

```
CREATE LOGIN [CONTOSO\SQLUsers] FROM WINDOWS;
GO
```

To verify that the account was created, type the following:

```
SELECT name FROM sys.server_principals;
GO
```

You should see the new group name at the end of the list.

More info: For more information about SQL Server signins and database users, see Walkthrough for the security features of SQL Server on Linux.
Restart the SQL Server 2017 service for those changes to take effect
To restart the SQL Server 2017 service and apply the changes, you first need to return to the Bash shell. Type `quit` to exit the sqlcmd utility:

```
quit
```

At the Bash prompt, type the following to restart SQL Service 2017:

```
sudo systemctl restart mssql-server
```

Grant the Active Directory group the permissions to sign in to the SQL Server 2017 on Red Hat Enterprise Linux host machine
Next, you need to use the `sudo realm permit` command to grant permission to the Active Directory group for which you just created the account for the local host.

Type the following command, substituting the name of your Active Directory domain name for `CONTOSO` and the name of the group account for `SQLUsers`:

```
sudo realm permit –g SQLUsers@CONTOSO.COM
```

Sign in to the host machine with the Active Directory user
Now, sign in to the local machine with an Active Directory user account that is a member of the group to which you have just allowed access. Type the following command, substituting the name of the Active Directory user account for `SQLUser1@contoso.com`.

```
su SQLUser1@contoso.com
```

When prompted, provide the password for the user to finish the sign in process.

A Bash shell prompt for the Active Directory user appears:

```
[SQLUser1@contoso.com@sqlrhel1 user]$
```

Connect to SQL Server 2017 by using an Active Directory user account
At the same Bash prompt for the Active Directory user, enter the following command to connect to SQL Server 2017, substituting the name of the local SQL Server 2017 host for `SQLRhel1`, and the new SQL Server 2017 port number for `65432`.

```
sqlcmd –S SQLRhel1,65432
```

A sqlcmd prompt should appear, indicating that the instance of SQL Server 2017 and the Active Directory user were able to authenticate each other.

You can now type `quit` to leave the sqlcmd prompt and exit to the Bash shell:

```
quit
```

Now, type `exit` to return to the Bash prompt for the local user account:

```
exit
```
DISABLE THE SA ACCOUNT (RECOMMENDED)

Best practices for SQL Server administration security is to create accounts for specific database roles and disable and/or rename the sa account that has complete access to the SQL Server instance.

To create a UserAdmin account to handle the creation of new accounts and granting memberships to server roles, run the following at the sqlcmd prompt:

```sql
CREATE LOGIN UserAdmin WITH PASSWORD = '***Complex Password***
GO
```

To grant that user permissions to create SQL account and grant role memberships, run the following T-SQL statement:

```sql
ALTER SERVER ROLE securityadmin ADD MEMBER UserAdmin
GO
```

To verify those changes, list the role memberships, run the following statement:

```sql
SELECT SRM.role_principal_id, SP.name AS Role_Name,
       SRM.member_principal_id, SP2.name AS Member_Name
FROM sys.server_role_members AS SRM
JOIN sys.server_principals AS SP ON SRM.Role_principal_id = SP.principal_id
JOIN sys.server_principals AS SP2 ON SRM.member_principal_id = SP2.principal_id
ORDER BY SP.name, SP2.name
GO
```

To add a domain user or group to a server role, run the following command, substituting **role name**, **domain**, and **username or group** accordingly:

```sql
ALTER SERVER ROLE **role-name** ADD MEMBER [**domain**\**username or group**]
```

Finally, if you want to disable the account, run the following when you are signed in to the SQL Server as sa:

```sql
ALTER LOGIN [SA] DISABLE
```

CONNECT TO SQL SERVER 2017 FROM AZURE DATA STUDIO

Azure Data Studio is a lightweight, cross-platform database graphical tool that you can use to manage SQL Server 2017. Versions exist for Windows, macOS, and Linux.

Use the following page to download and install the appropriate version of Azure Data Studio on your local computer: Download and install Azure Data Studio.
Connect to the SQL Server 2017 instance from Azure Data Studio

Open Azure Data Studio, and a Connection Details pane appears, as shown in Figure 6.1. In the Server field, specify the server in the form [IP address],[port number], as in the example 10.0.0.8,65432. (As a reminder, you can obtain the public IP address of this VM from Azure Portal.) For Authentication type, specify SQL Login. (For Windows authentication, your local computer would need to be a member of the same Active Directory domain as the SQL Server host.) For User name, enter sa or the alternative administrator account you have created. Enter the password, and then click Connect.

![Figure 6.1. Configuring a connection to the SQL Server 2017 host in Azure Data Studio](image)

A connection to the database server opens, revealing a dashboard similar to the one shown in Figure 6.2.
Import a database from a backup file

If you are migrating a database from another location, you can copy a full database backup (.bak) file to the Linux VM and then use Azure Data Studio to restore the database on the new server. For more information, see Restore a database from a backup file.
CHAPTER 7: CONFIGURING HTAP AND OPERATIONAL ANALYTICS

HTAP refers to a dual-database role that supports both transactional and analytical workloads. A database configured for HTAP, in other words, supports analytics on an organization’s live operational data.

For the transactional portion of HTAP, it’s recommended that tables be created or reconfigured to run in memory.

The analytical portion of HTAP uses an updatable columnstore index on a rowstore table. The columnstore index maintains a copy of the data, so the OLTP and analytics workloads run against separate copies of the data. This arrangement minimizes the performance impact of both workloads running at the same time. SQL Server automatically maintains index changes so that OLTP changes are always up to date for analytics. With this design, it is possible and practical to run analytics in real time on up-to-date data. This works for both memory-optimized and disk-based tables.

What are columnstore indexes?

A columnstore index is a technology for storing, retrieving, and managing data by using a columnar data format. SQL Server 2017 supports both clustered and non-clustered columnstore indexes. Both function the same, but they do have differences in purpose and in the features that they support. For more information about columnstore indexes, see Choose the best columnstore index for your needs.

OBJECTIVES

This chapter describes procedures that allow you to complete the following tasks:

- Configure a new or existing table as memory-optimized.
- Create a clustered columnstore index for operational analytics in memory.
- Create a nonclustered columnstore index for operational analytics on disk.

IN-MEMORY OLTP

In-memory tables offer optimal performance for OLTP workloads. Tables can be configured as memory-optimized in T-SQL simply by using the WITH (MEMORY_OPTIMIZED = ON) clause with a CREATE TABLE or ALTER TABLE statement. (See below for an example.) Of course, the available RAM must be able to support the size of the table.

For more information about in-memory, see In-Memory OLTP (In-Memory Optimization).

CONFIGURING COLUMNSTORE INDEXES FOR OPERATIONAL ANALYTICS

To get started with real-time analytics, identify the tables in your operational schema that contain data required for analytics. For each table identified, drop all unnecessary indexes and replace them with a single columnstore index. Doing so should improve the overall performance of your OLTP workload because there will be fewer indexes to maintain.

For information about dropping tables in T-SQL, see DROP TABLE (Transact-SQL).
Create clustered columnstore indexes for operational analytics on in-memory tables

The preferred method to configure HTAP is on in-memory tables because it optimizes both OLTP and analytical performance. For a table that is memory-optimized, you should allow operational analytics by creating a clustered columnstore index for that table. (You do not need to specify columns because a clustered columnstore index automatically includes all the columns.)

The following example creates a memory-optimized table with a clustered columnstore index. You would typically see the benefits of such an index if the table included at least a few million rows of data.

```sql
CREATE TABLE t_account (
    accountkey int NOT NULL PRIMARY KEY NONCLUSTERED,
    Accountdescription nvarchar (50),
    accounttype nvarchar(50),
    unitsold int,
    INDEX t_account_cci CLUSTERED COLUMNSTORE
)
WITH (MEMORY_OPTIMIZED = ON );
GO
```

Create nonclustered columnstore indexes for disk-based operational analytics

If your database server lacks enough available RAM to store a desired table in memory, you can still configure operational analytics on-disk with excellent performance by creating a non-clustered columnstore index for the table. Note that when you create a nonclustered columnstore index, you choose specific columns for the index. For example, the following T-SQL statement creates a nonclustered columnstore index for six columns in the Sales.OrderLines table:

```sql
CREATE NONCLUSTERED COLUMNSTORE INDEX [NCCX_Sales_OrderLines] ON [Sales].[OrderLines]
( [OrderID], [StockItemID], [Description], [Quantity], [UnitPrice], [PickedQuantity]
)WITH (DROP_EXISTING = OFF, COMPRESSION_DELAY = 0) ON [USERDATA]
GO
```
The simple procedures above are all you need to run operational analytics without making any changes to your applications. Analytics queries will run against the columnstore index, and OLTP operations will keep running against your remaining indexes required for OLTP. The OLTP workloads will continue to perform, but they will incur some additional overhead to maintain the columnstore index.

For more information on using columnstore indexes for operational analytics, see Get started with Columnstore for real time operational analytics.
CHAPTER 8: CONNECTING POWER BI TO YOUR DATABASE

Power BI is a powerful collection of mutually integrated software services, apps, and connectors that help you quickly uncover patterns and other insights from large and diverse datasets. With Power BI, you can combine, cleanse, shape, and model big data from different sources and use this data to generate striking visualizations. You can then publish these visualizations in reports, share your insights with others, and help your team make informed decisions quickly.

Power BI integrates with many external data sources and tools, including Microsoft Excel and SQL Server 2017 databases.

BEFORE YOU BEGIN

You need to complete certain configuration steps before you begin performing the steps described in this chapter. These steps include the following:

• Deployed a SQL Server on Red Hat Enterprise Linux VM
• Configured a SQL Server account to allow authentication to the server

OBJECTIVES

This chapter describes procedures that allow you to complete the following tasks:

• Connect Power BI Desktop to an instance of SQL Server

CONNECT POWER BI DESKTOP TO A SQL SERVER ON RED HAT ENTERPRISE LINUX VM

Power BI Desktop is a client application for Power BI that you can download from PowerBI.Microsoft.com. Once the application is installed, you can begin the process of connecting to your SQL Server 2017 on Red Hat Enterprise Linux VM by pointing to Get Data on the File menu, and then clicking SQL Server, as shown in Figure 7.1.
When you are prompted to specify a SQL Server database server, specify your Azure VM's public IP address and the port that you have assigned to SQL Server 2017 in the format [public IP address]:[port]. Next, when you are asked to supply credentials, choose Windows on the left to specify an Active Directory User name and Password if you are connecting from a domain-joined machine. Otherwise, click Database on the left, and then specify a SQL Server account User name and Password, as shown in Figure 7.2.
After you supply credentials, you can select a database and choose tables to load into Power BI Desktop. Note that if you connect Power BI to a production database containing operational data, it would be important to create columnstore indexes beforehand on the tables you want to import. Otherwise, the Power BI queries can degrade the performance of your transactional database. When the selected tables are loaded, you can begin to select fields and then create visualizations and reports from the selected data.

To learn more about Power BI, see What is Power BI?

CREATE A DASHBOARD IN THE POWER BI SERVICE

Power BI service (also known as Power BI online) is the cloud-based version of Power BI, which is accessed through https://app.powerbi.com. The Power BI service offers a set of features that complement those available through Power BI Desktop.

For an introduction to the Power BI service, see Tutorial: Get started with the Power BI service (app.powerbi.com). For a comparison of the features available in Power BI Desktop and those available in the Power BI service, see Comparing Power BI Desktop and the Power BI service.

Dashboards in the Power BI service

Dashboards tell a story based on reports or saved datasets, and they also provide a way to monitor important metrics at a glance. Available only through the Power BI service, dashboards are essentially single-page arrangements of selected tiles that display statistics and data visualizations created in a Power BI report. The visualizations from dashboards can be based on data from a wide variety of sources, such as databases, storage files, analytics data sources, metrics data sources, real-time streaming data sources, or any combination of these sources.

Real-time dashboards include tiles based on streaming or frequently updated data sources, allowing you to understand trends and gain insights from data as quickly as possible.

You can use a SQL Server database as a data source for a dashboard. Though the Power BI service cannot directly connect to a SQL Server database, you can create a report in Power BI Desktop, publish the report to the Power BI service, and configure the refresh settings so that the data is kept up-to-date. You can then use this report as a tile in a dashboard.

Important

You need a Power BI Pro license to create dashboards.

For more information on publishing a report based on a SQL Server dataset to the Power BI service, see Tutorial: Connect to on-premises data in SQL Server. (Note that a data gateway is not needed in our scenario because the SQL Server 2017 instance is hosted in Azure.) To find out how to create a dashboard from a report, see Create a Power BI dashboard from a report. For information on configuring refresh settings for dashboards, see Data refresh in Power BI.
CHAPTER 9: CONFIGURING HIGH AVAILABILITY AND DISASTER RECOVERY FOR SQL SERVER 2017 ON RED HAT ENTERPRISE LINUX VMS IN AZURE

Availability groups (AGs) configured for SQL Server 2017 on Linux in bare-metal installations typically rely on a clustering solution such as Pacemaker. However, at the time of this writing, no clustering solution for Linux virtual machines is supported or recommended in production environments running in Azure. To achieve high availability, a SQL Server 2017 AG can still be used for manual failover. This solution can then be supplemented by using availability sets or availability zones in Azure.

CONFIGURE AVAILABILITY GROUPS IN SQL SERVER 2017 FOR MANUAL FAILOVER

The supported Linux distributions for SQL Server 2017 do not support a clustering solution in the cloud. However, SQL Server 2017 AGs without any underlying cluster can still be configured to support manual failover from a primary replica to a secondary replica. While this feature does not fully provide high availability (HA) on its own, it is an important building block of HA for deployments of SQL Server 2017 on Linux in Azure.

A cluster-less AG is also known as a read-scale AG. It is configured by specifying the option CLUSTER_TYPE = NONE when creating (or altering) an AG. For example, the following T-SQL script creates an AG named “ag1.” The script configures the AG replicas with SEEDING_MODE = AUTOMATIC. This setting causes SQL Server 2017 to automatically create the database on each secondary server after it is added to the AG. If you want to update the script for your environment, you should replace the `<node1>` and `<node2>` values with the names of the SQL Server 2017 instances that host the replicas. Then replace the `<65432>` value with the SQL Server port that you have set for each instance. Be sure to run the script on the primary SQL Server 2017 replica.

```
CREATE AVAILABILITY GROUP [ag1]
WITH (CLUSTER_TYPE = NONE)
FOR REPLICA ON
  N'<node1>' WITH (
    ENDPOINT_URL = N'tcp://<node1>:<65432>',
    AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
    FAILOVER_MODE = MANUAL,
    SEEDING_MODE = AUTOMATIC,
    SECONDARY_ROLE (ALLOW_CONNECTIONS = ALL)
  ),
 N'<node2>' WITH (
    ENDPOINT_URL = N'tcp://<node2>:<65432>',
    AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
    FAILOVER_MODE = MANUAL,
    SEEDING_MODE = AUTOMATIC,
    SECONDARY_ROLE (ALLOW_CONNECTIONS = ALL)
  );

ALTER AVAILABILITY GROUP [ag1] GRANT CREATE ANY DATABASE;
```
After manual failover, you can promote a secondary replica to the primary replica. Figure 9.1 provides an illustration of the status of the read-scale AG before and after failover.

High availability with SQL Server Availability Groups for manual failover of SQL Server instance failure

Cluster-less Availability Group

Administrator manually fails over and promotes a secondary to a primary:

Cluster-less Availability Group

The manual failover to the secondary replica can occur before or after the primary has failed. If you perform the manual failover before failure, no data is lost. If you perform the manual failover after failure, some data loss is possible.

The steps for failing over are different in each of these two cases. *For information on these steps and on how to promote the secondary replica to a primary replica, see Fail over the primary replica on a read-scale Availability Group. For more general information about read-scale AGs on Linux, see Configure a SQL Server Availability Group for read-scale on Linux.*
Using Azure CLI to query VM status

Before attempting manual failover, it's crucial to verify that your SQL Server 2017 VM in Azure is unreachable because the server itself is truly in a failed state. Any type of network connectivity problems between you and the host VM can also lead to the database server appearing unreachable. If failover were attempted in such a case, it could lead to a split-brain scenario, with two nodes acting as if they’re primaries. To verify the status of a VM and rule out network connectivity issues, type the following command:

```
az vm get-instance-view --name "VM-Name" --resource-group "RG-Name" --query instanceView.statuses[1] --output table
```

Read-only routing: an alternative design for HTAP

Read-scale AGs improve availability of SQL Server 2017 on Linux in Azure by making manual failover possible. But they can also provide an alternative way to support HTAP workloads.

In this deployment design, the readable secondary replica can be used to scale out HTAP. Analytics queries are redirected to the readable secondary replica instead of to a columnstore index.

Redirecting read-only queries to a secondary replica is known as read-only routing. To configure read-only routing on the secondary replica, in the ADD REPLICA or MODIFY REPLICA WITH clause of the CREATE AVAILABILITY GROUP or ALTER AVAILABILITY GROUP statement, specify the SECONDARY_ROLE option as follows, substituting the address of your SQL Server 2017 VM for system-address and your SQL Server port for port:

```
SECONDARY_ROLE (READ_ONLY_ROUTING_URL = 'TCP://system-address:port')
```

For example, for a secondary server whose FQDN is SQLRhel2.contoso.com and whose SQL Server 2017 port is configured as 65432, you would use the following clause:

```
SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://SQLRhel2.contoso.com:65432')
```

Read-only routing has the following benefits:

- For HTAP scenarios, this design offloads the read-only workload from the primary replica. It speeds up transactional queries on the primary, in addition to analytical queries on the readable secondary replica for real-time Power BI dashboards and applications.
- As you have seen, read-only routing enables you to keep an extra copy of your mission-critical data, allowing you to manually fail over to it if necessary.

*For information on configuring read-only routing for an AG, see Configure read-only routing for an Always On availability group.*

Configure an Azure Availability Set or Azure Availability Zone for your VM

To enhance the availability of a read-scale AG, you should also assign your VM to an Azure Availability Set or an Azure Availability Zone.
Azure Availability Sets

An Azure Availability Set is a group of two or more VMs in the same Azure datacenter location. Availability Sets offer 99.95 percent availability for a VM, and they are available for all Azure regions.

For more information about Azure Availability Sets, see Tutorial: Create and deploy highly available virtual machines with the Azure CLI.

Azure Availability Zones

Availability Zones are an alternative to Availability Sets. Azure Availability Zones refer to different datacenter locations within an Azure region. By assigning a VM to an Azure Availability Zone, you ensure that copies are kept in separate zones. Availability Zones offer 99.99 percent availability for a VM, but, at the time of this writing, they are available only for the following Azure regions:

- West US 2
- Central US
- East US
- East US 2
- UK South
- France Central
- West Europe
- North Europe
- Southeast Asia

To learn more about Azure Availability Zones, see What are Availability Zones in Azure?

To read about the similarities and differences between Availability Sets and Availability Zones, visit Azure VMs: Availability Sets and Availability Zones.

Configure an internal load balancer for the AG

An availability group requires a load balancer when the SQL Server 2017 instances are on Azure VMs. The load balancer—typically an internal load balancer—stores the IP address for the availability group listener. If an availability group spans multiple regions, each region needs a load balancer.

For more information about configuring a load balancer for an AG in Azure, see Configure a load balancer for an Always On availability group in Azure. See also Internal Load Balancer for a general introduction to this topic.

For an overview of strategies you can use to manage the availability of Linux VMs in Azure, see Manage the availability of Linux virtual machines.
BACK UP AND RESTORE SQL SERVER 2017 ON RED HAT ENTERPRISE LINUX IN AZURE

While it’s important to use technologies in Azure to keep SQL Server 2017 on Linux up and running as much as possible, it’s just as important to plan and implement a sound disaster-recovery strategy. A strategy begins by defining your specific goals for recovery-point objectives (RPOs) and recovery-time objectives (RTOs). Then, you need to create a backup and recovery plan that will allow you to meet those goals.

In general, database administrators should plan a backup strategy that includes a combination of full backups, incremental backups, and frequent transaction log backups. *(For more information, see Best Practices for Maintaining SQL Server SAP Systems.)* After a hardware loss or disk failure affecting the database files, you can then use these various backups to restore the database to the point of failure. To do so, first restore the last full database backup and the last differential database backup, followed by all the subsequent transaction log backups up to the point of failure.

*For an introduction to backup and restore strategies for Azure VMs, see Backup and Restore for SQL Server in Azure Virtual Machines.*

The following section introduces additional strategies, tools, and considerations for your SQL Server 2017 backup strategy.

**Managed backups**

Managed backups automate the creation of backups to Azure storage. Using SQL Server Managed Backup to Microsoft Azure, you can create a backup plan by specifying only the retention period and storage location. Although advanced settings are available, they are not required. SQL Server Managed Backup to Microsoft Azure schedules, performs, and maintains the backups.

*For more information about managed backups, see SQL Server Managed Backup to Windows Azure and Configure Managed Backup (SQL Server Management Studio).*

**Backup to Azure Blob storage**

You can back up and restore a SQL Server database directly to Azure Blob storage, a feature which is also known as backup to URL. The main benefit of this strategy is reliable, offsite, limitless, and inexpensive storage.

*For detailed information about this option, see SQL Server Backup to URL and SQL Server Backup and Restore with Microsoft Azure Blob Storage Service.*

**Backup and restore tools in Red Hat Enterprise Linux**

You can take backups of databases from SQL Server 2017 on Red Hat Enterprise Linux with the same tools as other platforms. On a Linux server, you can use the `sqlcmd` utility to connect to SQL Server 2017 and take backups. From Windows, you can connect to SQL Server 2017 on Red Hat Enterprise Linux and take backups with the user interface. The backup functionality is the same across platforms. For example, you can back up databases locally, to remote drives, or to Azure Blob storage.

*See Backup and restore SQL Server databases on Linux for more information about this topic.*
CHAPTER 10: MONITORING TOOLS IN SQL SERVER 2017 ON RED HAT ENTERPRISE LINUX IN AZURE

The SQL Server 2017 environment is dynamic, and administrators need to monitor this changing environment carefully to ensure that performance requirements can continue to be met. Tools for monitoring SQL Server 2017 performance include T-SQL, Azure Data Studio, SQL Server Management Studio (SSMS), and the Red Hat Enterprise Linux command line.

T-SQL
T-SQL works the same in SQL Server 2017, regardless of whether the underlying platform is Windows or Linux. Examples of T-SQL elements useful for monitoring include Database Consistency Checker (DBCC) commands such as DBCC CHECKDB, dynamic management views such as sys.dm_os_wait_stats, stored procedures such as sp_trace_create, and built-in functions such as @@CPU_BUSY. For more information about performance and monitoring elements in T-SQL, see Performance Monitoring and Tuning Tools.

INSIGHT WIDGETS IN AZURE DATA STUDIO
Insight widgets are customizable charts and graphs based on T-SQL statements that you add to server and database monitoring dashboards. Using insight widgets, you can build customized server- and database-management dashboards, such as the one shown in Figure 10.1.

Azure Data Studio is a lightweight, cross-platform database GUI tool that you can use to manage SQL Server 2017 from Windows, macOS, and Linux.

Figure 10.1. An example dashboard made of insight widgets in Azure Data Studio
To create an insight widget for performance monitoring that lists the five slowest queries in a database, see Tutorial: Add the five slowest queries sample widget to the database dashboard. For more general information about creating your own insight widgets, see Tutorial: Build a custom insight widget.

SQL SERVER MANAGEMENT STUDIO (SSMS)

You can run SSMS on a Windows computer to connect to an instance of SQL Server 2017 on Red Hat Enterprise Linux. SSMS offers a broad set of monitoring features, including the ones described below.

Standard reports and performance dashboard

Standard reports provide important monitoring information about the activity and performance of your SQL Server 2017 instance. The Performance Dashboard (at the top of the list in Figure 10.2) is especially useful for help identifying whether SQL Server 2017 is experiencing a performance bottleneck, such as the CPU or input/output (I/O). See Performance Dashboard for more information about this tool.

![Figure 10.2. Standard reports in SSMS](image)

SQL Server Profiler

SQL Server Profiler is an interface to create and manage traces and analyze and replay trace results. Events are saved in a trace file that can later be analyzed or used to replay a specific series of steps when trying to diagnose a problem. See SQL Server Profiler for more information about this tool.

Activity Monitor

Activity Monitor in SSMS shows live performance metrics, such as % Processor Time, along with information like recent expensive queries and active processes. For more information about this tool, see Open Activity Monitor (SQL Server Management Studio).
Query Store
Query Store automatically captures a history of queries, plans, and runtime statistics, and it retains these for your review. It separates data by time windows so that you can see database usage patterns and understand when query plan changes happened on the server. For more information, see Monitoring performance by using the Query Store.

To download and install the latest version of SSMS, see Download SQL Server Management Studio.

COMMAND-LINE TOOLS IN RED HAT ENTERPRISE LINUX
To monitor operating system–level performance data on the SQL Server 2017 on Red Hat Enterprise Linux VM, you can use a wide variety of command-line tools. For information about these tools, see Built-in command-line monitoring tools and Troubleshoot SQL Server on Linux.

NEXT STEPS
Congratulations! You’ve made it to the end of the Microsoft SQL Server 2017 on Red Hat Enterprise Linux in Azure VMs Operations Guide.

For more information about SQL Server on Red Hat Enterprise Linux, visit www.redhat.com/en/explore/microsoft-sql-server-rhel.

To watch a video about SQL Server on Linux, visit: www.youtube.com/watch?v=nFt4T7rKkhE.


To learn more about SQL Server on Azure VMs, visit https://azure.microsoft.com/en-us/services/virtual-machines/sql-server/.


For a general overview of Azure VMs, visit https://azure.microsoft.com/en-us/services/virtual-machines/.

For sales questions about Microsoft SQL Server 2017, contact a Microsoft representative at (800) 426-9400 in the United States or (877) 568-2495 in Canada. To locate a solution provider, see Search solution providers.