Microsoft

Administering a SQL Database Infrastructure

Exam Ref 70-764

Victor Isakov

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Microsoft

Exam Ref 70-764 Administering a SQL Database Infrastructure

Victor Isakov

Exam Ref 70-764 Administering a SQL Database Infrastructure

Published with the authorization of Microsoft Corporation by:
Pearson Education, Inc.

ISBN-13: 978-1-5093-0383-0
ISBN-10: 1-5093-0383-9

Library of Congress Control Number: 2017953072
First Printing September 1 17

Editor-in-Chief	Greg Wiegand
Acquisitions Editor	Trina MacDonald
Development Editor	Troy Mott
Managing Editor	Sandra Schroeder
Senior Project Editor	Tracey Croom
Editorial Production	Backstop Media
Copy Editor	Christina Rudloff
Indexer	Julie Grady
Proofreader	Christina Rudloff
Technical Editor	Martin 'MC' Brown
Cover Designer	Twist Creative, Seattle

Contents at a glance

	Introduction	*ix*
	Preparing for the exam	*xiii*
CHAPTER 1	**Configure data access and auditing**	**1**
CHAPTER 2	**Manage backup and restore of databases**	**65**
CHAPTER 3	**Manage and monitor SQL Server instances**	**179**
CHAPTER 4	**Manage high availability and disaster recovery**	**265**
	Index	*381*

This page intentionally left blank

Contents

Introduction **ix**

Organization of this book ix

Microsoft certifications x

Acknowledgments x

Microsoft Virtual Academy x

Quick access to online references x

Errata, updates, & book support xi

We want to hear from you xii

Stay in touch xii

Preparing for the exam *xiii*

Chapter 1 Configure data access and auditing **1**

Skill 1.1: Configure encryption 2

Implement column-level encryption 2

Implement Always Encrypted 9

Configure transparent data encryption 21

Implement backup encryption 25

Configure encryption for connections 26

Troubleshoot encryption errors 27

Skill 1.2 Configure data access and permissions 28

Create and maintain users 29

Create and maintain custom roles 36

Manage database object permissions 38

Configure row-level security 41

Configure dynamic data masking 47
Configure user options for Azure SQL Database 49
Skill 1.3: Configure auditing 50
Configure an audit on SQL Server 51
Query the SQL Server audit log 55
Manage a SQL Server audit 56
Configure an Azure SQL Database audit 57
Analyze audit logs and reports from Azure SQL Database 60
Thought experiment. 61
Thought experiment answers 62
Chapter summary 63

Chapter 2 Manage backup and restore of databases 65

Skill 2.1: Develop a backup strategy 66
Design a backup strategy 66
Back up databases 70
Back up VLDBs 87
Manage transaction log backups 90
Configure backup automation 106
Skill 2.2 Restore databases 141
Design a restore strategy 141
Restore a database 145
Perform piecemeal restores 148
Perform page recovery 154
Perform point-in-time recovery 157
Restore a filegroup 161
Develop a plan to automate and test restores 162
Skill 2.3 Manage database integrity 163
Implement database consistency checks 163
Identify database corruption 167
Recover from database corruption 169
Thought experiment. 173
Thought experiment answers 175
Chapter summary 176

Chapter 3 Manage and monitor SQL Server instances 179

Skill 3.1: Monitor database activity 180
Monitor current sessions 180
Identify sessions that cause blocking activity 183
Identify sessions that consume tempdb resources 186
Configure the data collector 188

Skill 3.2 Monitor queries 197
Manage the Query Store 197
Configure Extended Events and trace events 205
Identify problematic execution plans 214
Troubleshoot server health using Extended Events 217

Skill 3.3 Manage indexes 218
Identify and repair index fragmentation 218
Identify and create missing indexes 221
Identify and drop underutilized indexes 223
Manage existing columnstore indexes 225

Skill 3.4 Manage statistics 227
Identify and correct outdated statistics 227
Implement Auto Update Statistics 231
Implement statistics for large tables 233

Skill 3.5 Monitor SQL Server instances 235
Configure database mail 235
Create and manage operators 242
Create and manage SQL Agent alerts 243
Define custom alert actions 245
Define failure actions 246
Configure policy based management 247
Identify available space on data volumes 253
Identify the cause of performance degradation 254

Thought experiment. 260

Thought experiment answers 261

Chapter summary 262

Chapter 4 Manage high availability and disaster recovery 265

Skill 4.1: Design a high availability solution 266

Skill 4.2: Design a disaster recovery solution 270

Skill 4.3: Implement log shipping 271

- Architect log shipping 271
- Configure log shipping 275
- Monitor log shipping 284

Skill 4.4: Implement Availability Groups 287

- Architect Availability Groups 287
- Configure Windows clustering 298
- Create an Availability Group 304
- Configure read-only routing 322
- Monitor Availability Groups 325
- Manage failover 327
- Create Distributed Availability Group 331

Skill 4.5: Implement failover clustering 332

- Architect failover clustering 333
- Configure failover clustering 338
- Manage Shared Disks 371
- Configure Cluster Shared Volumes 371

Thought experiment 375

Thought experiment answers 378

Chapter summary 379

Index *381*

Introduction

First and foremost, thank you for your purchase and all the best of luck in your endeavor to become certified and an expert in the SQL Server data platform. The 70-764 exam is intended for database professionals who perform installation, maintenance, and configuration tasks on the SQL Server platform. Other responsibilities include setting up database systems, making sure those systems operate efficiently, and regularly storing, backing up, and securing data from unauthorized access.

This book is geared toward database administrators who are looking to train in the administration of SQL Server 2016 infrastructure. To help you prepare for the exam you can use Microsoft Hyper-V to create SQL Server virtual machines (VMs) and follow the examples in this book. You can download an evaluation copy of Windows Server 2016 from *https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-2016/*. SQL Server 2016 can be downloaded for free from *https://www.microsoft.com/en-us/sql-server/sql-server-downloads*. You can download the AdventureWorks databases from *https://msftdbprodsamples.codeplex.com/*. The Wide World Importers database can be downloaded from *https://github.com/Microsoft/sql-server-samples/releases/tag/wide-world-importers-v1.0*.

This book covers every major topic area found on the exam, but it does not cover every exam question. Only the Microsoft exam team has access to the exam questions, and Microsoft regularly adds new questions to the exam, making it impossible to cover specific questions. You should consider this book a supplement to your relevant real-world experience and other study materials. If you encounter a topic in this book that you do not feel completely comfortable with, use the "Need more review?" links you'll find in the text to find more information and take the time to research and study the topic. Great information is available on MSDN, TechNet, and in blogs and forums.

Organization of this book

This book is organized by the "Skills measured" list published for the exam. The "Skills measured" list is available for each exam on the Microsoft Learning website: *https://aka.ms/examlist*. Each chapter in this book corresponds to a major topic area in the list, and the technical tasks in each topic area determine a chapter's organization. If an exam covers six major topic areas, for example, the book will contain six chapters.

Microsoft certifications

Microsoft certifications distinguish you by proving your command of a broad set of skills and experience with current Microsoft products and technologies. The exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop, or implement and support, solutions with Microsoft products and technologies both on-premises and in the cloud. Certification brings a variety of benefits to the individual and to employers and organizations.

MORE INFO **ALL MICROSOFT CERTIFICATIONS**

For information about Microsoft certifications, including a full list of available certifications, go to *https://www.microsoft.com/learning*.

Acknowledgments

Victor Isakov I would like to dedicate this book to Christopher, Isabelle, Marcus and Sofia. With your love and "infinite patience" I am the luckiest guy on this planet! It would be remiss of me not to also thank Trina MacDonald and Troy Mott for their "infinite patience" in helping me complete this "impossible task."

Microsoft Virtual Academy

Build your knowledge of Microsoft technologies with free expert-led online training from Microsoft Virtual Academy (MVA). MVA offers a comprehensive library of videos, live events, and more to help you learn the latest technologies and prepare for certification exams. You'll find what you need here:

https://www.microsoftvirtualacademy.com

Quick access to online references

Throughout this book are addresses to webpages that the author has recommended you visit for more information. Some of these addresses (also known as URLs) can be painstaking to type into a web browser, so we've compiled all of them into a single list that readers of the print edition can refer to while they read.

Download the list at *https://aka.ms/exam764administersql/downloads*.

The URLs are organized by chapter and heading. Every time you come across a URL in the book, find the hyperlink in the list to go directly to the webpage.

Errata, updates, & book support

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

https://aka.ms/exam764administersql/errata

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at mspinput@microsoft.com.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to *https://support.microsoft.com*.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

https://aka.ms/tellpress

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*.

This page intentionally left blank

Important: How to use this book to study for the exam

Certification exams validate your on-the-job experience and product knowledge. To gauge your readiness to take an exam, use this Exam Ref to help you check your understanding of the skills tested by the exam. Determine the topics you know well and the areas in which you need more experience. To help you refresh your skills in specific areas, we have also provided "Need more review?" pointers, which direct you to more in-depth information outside the book.

The Exam Ref is not a substitute for hands-on experience. This book is not designed to teach you new skills.

We recommend that you round out your exam preparation by using a combination of available study materials and courses. Learn more about available classroom training at *https://www.microsoft.com/learning*. Microsoft Official Practice Tests are available for many exams at *https://aka.ms/practicetests*. You can also find free online courses and live events from Microsoft Virtual Academy at *https://www.microsoftvirtualacademy.com*.

This book is organized by the "Skills measured" list published for the exam. The "Skills measured" list for each exam is available on the Microsoft Learning website: *https://aka.ms/examlist*.

Note that this Exam Ref is based on publicly available information and the author's experience. To safeguard the integrity of the exam, authors do not have access to the exam questions.

This page intentionally left blank

CHAPTER 4

Manage high availability and disaster recovery

It is important to understand the difference between high availability and disaster recovery. It is not uncommon for management in organizations to misunderstand these concepts and use the wrong technology for their SQL Server infrastructure. In this last chapter, we examine the high availability technologies available in SQL Server, as promised in Chapter 2, "Manage backup and restore of databases."

With high availability, you are using technology in SQL Server to minimize the downtime of a given database solution to maximize its availability. With disaster recovery, however, you are using technology to recover from a disaster incident, potentially minimizing the amount of data lost. In some cases, data loss is acceptable, because the imperative is to get your database solution online as soon as possible. That is why it is critical to engage with all stakeholders to determine the business requirements. With both high availability and disaster recovery people and processes play a key part, so make sure you don't focus solely on the technology.

The exam will test your ability to design the appropriate high availability solution for a given scenario, which is why Skill 4.1 starts with a discussion about high availability and the primary considerations for designing a particular solution. Skill 4.2 then covers the designing of a disaster recovery solution, which commonly goes hand-in-hand with a high availability solution. Given how we covered disaster recovery in Chapter 2, a detailed discussion will not be required here. Skill 4.3 examines the log shipping technology in SQL Server and how it is primarily used to provide disaster recovery. Skill 4.4 then details Availability Groups and examines how they can be used to provide both high availability and scale-out capability to your databases. Finally, in Skill 4.5 we implement failover clustering solutions. Although this high availability technology has been available since SQL Server 2000, it's commonly used in the industry and should not be discounted as an old, unused technology. Microsoft keeps investing in failover clustering, and we will learn about how SQL Server can take advantage of cluster shared volumes.

High availability technologies are complex and involve a lot of set up and configuration, so this chapter has many figures that show you their installation, configuration, and administration processes. Make sure you examine the various options in the figures and listings in this chapter to best prepared for the exam.

NOTE **PREPARING FOR THE EXAM**

To help you prepare for the exam and help you familiarize yourself with the high availability capabilities in SQL Server, you can use Hyper-V and setup a number of Virtual Machine (VMs) on your computer. It is recommended that you first install and configure a domain controller VM. All SQL Server VMs should be joined to the domain. It is also recommended that you use Windows Server 2016, because the examples are based on it.

SQL Server 2016/2017 Developer Edition is equivalent to the Enterprise Edition and now available for free at *https://www.microsoft.com/en-us/sql-server/sql-server-downloads*.

There is no free version of Windows Server 2016, however, you can download the Evaluation Edition, which will work for 180 days without re-arming it, at *https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-2016/*.

Skills covered in this chapter:

- Design a high availability solution
- Design a disaster recovery solution
- Implement log shipping
- Implement Availability Groups
- Implement failover clustering

Skill 4.1: Design a high availability solution

High availability, as the name suggests, is concerned with making sure that your database is highly available. The cost of an unavailable database solution, in today's modern, globalized 24x7, Internet connected world can be catastrophic to your organization.

One of the first questions you should be asking of your organization is what availability is required for your database solution. This will form part of your Service Level Agreement (SLA) for your database solution. Availability is usually expressed as a percentage of uptime in a given year, and can be expressed as follows:

$$\text{Availability} = \frac{\text{Actual uptime}}{\text{Expected uptime}} \times 100\%$$

This is commonly referred to as the number of nines required. Table 4-1 shows the availability, the number of nines, and how much down time that corresponds to annually.

TABLE 4-1: Number of nines for high availability.

Availability	Nines	Annual downtime	Monthly downtime	Weekly downtime	Daily downtime
90%	1	36.5 days	72 hours	16.8 hours	2.4 hours
95%	1.5	18.25 days	36 hours	8.4 hours	1.2 hours
99%	2	3.65 days	7.20 hours	1.68 hours	14.4 minutes
99.5%	2.5	1.83 days	3.60 hours	50.4 minutes	7.2 minutes
99.9%	3	8.76 hours	43.8 minutes	10.1 minutes	1.44 minutes
99.95%	3.5	4.38 hours	21.56 minutes	5.04 minutes	43.2 seconds
99.99%	4	52.56 minutes	4.38 minutes	1.01 minutes	8.66 seconds
99.995%	4.5	26.28 minutes	2.16 minutes	30.24 seconds	4.32 seconds
99.999%	5	5.26 minutes	25.9 seconds	6.05 seconds	864.3 milliseconds
99.9999%	6	31.5 seconds	2.59 seconds	604.8 milliseconds	86.4 milliseconds
99.99999%	7	3.15 seconds	262.97 milliseconds	60.48 milliseconds	8.64 milliseconds
99.99999999%	8	315.569 milliseconds	26.297 milliseconds	6.048 milliseconds	0.864 milliseconds
99.999999999%	9	31.5569 milliseconds	2.6297 milliseconds	0.6048 milliseconds	0.0864 milliseconds

As you can see, achieving even four nines might be difficult. Four nines represents only 4.38 minutes of downtime per month. Now consider how long it takes for your servers to be rebooted. On modern servers, that have a large amount of memory, it might take you 15-30 minutes for them to boot up, as they run through their BIOS memory checks. In most cases these BIOS memory checks cannot be turned off. Consider further how often you patch your Windows environment, which typically requires a reboot, and how long that takes. Do not underestimate the potential complexity of achieving anything beyond three nines.

When determining your SLA you should also define what constitutes downtime in the context of your SLA. There are two types of downtime:

- **Planned downtime** Planned downtime refers to the downtime incurred by your maintenance tasks. These maintenance tasks might include patching hardware or software, hardware maintenance, patching the Windows operating system, or patching SQL Server. Planned downtime is typically scheduled and controlled through business processes. Consequently, there is typically no data loss.
- **Unplanned downtime** Unplanned downtime refers to downtime that is incurred due to an unexpected incident that causes outage. Examples include:
  - Hardware failures, such as with a disk drive or power supply unit failing or bad firmware in a hardware vendor's HBA.
  - Data center failure, such as with a power failing, or flooding occurring.

- Software failure, such as with Windows crashing, SQL Server hanging, or a corrupt database.
- User error, such as dropping a database, or accidentally deleting data.

In some SLAs there are only penalties enforced for unplanned downtime.

Once you have determined your organizations availability requirements you can assess which SQL: Server high availability technology fits your business requirements. You might need to take in multiple factors, including:

- **Whether automatic failover is required** Certain high availability technologies and configurations do not offer automatic failover. In certain use cases an organization might not require high availability.
- **Failover speed** Different high availability technologies offer different failover speeds, so understanding how quickly a failover needs to take will help you choose the appropriate solution.
- **Scalability** Whether or not you need to scale out your database solution impacts your high availability technology selection. Scaling out your database can provide both performance and uptime benefits. Availability Groups offer the best scale-out capability.
- **Infrastructure between data centers** The latency and throughput between sites/data centers will directly impact what high availability technologies can be implemented. Latency is more important than distance. You do not want automatic fail overs to be performed due to slow response times between data centers. In this case automatic failover might not be required.
- **Connecting applications** What applications are accessing the database and what network libraries they use to connect to the databases will also play an important factor in any design. Certain high availability technologies might not work as well with older applications.
- **Recovery model** This is a very important consideration. What recovery model is being used by a database and the volume of transactions experienced by the database will dictate what high availability technology can be used. Remember that Availability Groups require the databases to be using the full recovery model. We covered recovery models in Chapter 2.
- **Number of databases** Whether the database solution involves multiple databases that need to fail over as a single unit is another important factor and design consideration.
- **Database size** This covers the size of the databases and how much it will cost to potentially replicate those databases on multiple instances of SQL Server. Very Large Databases (VLDBs) might be too expensive to host on multiple SQL Server instances. They might also be larger than what is possible to fit locally on a server, for example.

- **Database administrator skill set** Determine whether your organization has a team of database administrators and how experienced they are. Certain high availability technologies are more complex to administer.
- **SQL Server Edition** The SQL Server licensing implications and associated costs with a different edition is a very important factor in organizations. Certain high availability technologies are only available in the Enterprise Edition.

IMPORTANT **DESIGNING A HIGH AVAILABILITY SOLUTION**

When designing a high availability solution, you need to take into account all the things that can possibly fail. You also need to provide redundancy at every level that is cost effective for your organization. There a plenty of tales/urban myths about multi-million dollar highly available solutions failing due to a non-redundant component that cost an insignificant amount. Some of those tales are true!

SQL Server supports the following high availability technologies:

- **Failover clustering** With failover clustering you rely on the features of the Windows operating system to build a cluster of separate nodes that together provide redundancy at the hardware and software levels.
- **Transactional replication** With transactional replication various replication agents are reading a database's transaction log, storing those captured transactions in a separate database, and then replicating those transactions to other databases located on different servers.
- **Database mirroring** With database mirroring the database engine automatically transmits the transaction log to one other server when the same database exists.
- **Availability groups** Availability groups are an evolution of Database Mirroring where the transaction log can be transmitted in real time to multiple servers that maintain a copy of the database.
- **Log shipping** With log shipping multiple copies of the database are kept on multiple servers through scheduled log backups and restores.

Each high availability technology will have its own set of associated costs, and pros, and cons. As a database administrator, it is up to you to assess your business requirements and architect the appropriate high availability solution. In this book, we will focus on Log Shipping, Availability Groups and Failover Clustering.

Don't forget that you can combine high availability technologies. For example, you can take advantage of failover clustering locally in one data center to provide high availability and use log shipping to another data center for disaster recovery purposes.

IMPORTANT **IMPLEMENTING A PROOF-OF-CONCEPT FOR YOUR HIGH AVAILABILITY SOLUTION**

There is no substitute for implementing a Proof-of-Concept (POC) for your high availability solution to ensure that it will work exactly as you expect it to. It will also allow you to test your processes and determine whether the high availability technology will impact your databases solution. Implementing a POC is so easy these days with virtualization and the cloud. Just do it!

Skill 4.2: Design a disaster recovery solution

Whereas high availability is concerned about mitigating against different types of failures, disaster recovery is concerned about what to do in the case of a failure occurring. A disaster recovery solution involves technology and processes that will enable you to restore your availability with an appropriate data loss in an appropriate timeframe.

To design an appropriate disaster recovery plan, you need to engage the appropriate stakeholders in your organization to articulate the following requirements:

- Recovery Time Objective (RTO)
- Recovery Point Objective (RPO)
- Recovery Level Objective (RLO)

When designing your disaster recovery plan you need to take in multiple additional factors, including:

- The size of the databases
- How long it will take to restore hardware
- Whether you can take advantage of the cloud
- How long it will take to restore the Windows operating system
- How long it will take to restore the SQL Server environment
- The order in which you will need to perform the various tasks in your disaster recovery plan

These considerations and others were covered in depth in Chapter 2. Make sure you understand the concepts and considerations covered there, because they will impact your high availability design. The exam might ask you to design a high availability and disaster recovery solution for the same given scenario.

In a lot of organization's cases their database solutions are "too big to fail." In such cases you need to rely more on high availability, redundancy, and processes to ensure that you never have to recovery from a disaster. Good luck!

IMPORTANT **YOU CANNOT CHANGE OR BEAT THE LAWS OF PHYSICS**

Organizations and management typically do not appreciate the complexity and time that it will take to recover from a disaster. If you have a multi-terabyte database solution that needs to be recovered in the event of a disaster occurring, it will take days! You cannot change or beat the laws of physics. Restoring a 4-10TB database on to the fastest flash storage today will still potentially take a number of days. This means you can have days when your organization might lose a phenomenal amount of money and customers. That is one of the reasons why it is important to periodically test your disaster recovery plan.

Skill 4.3: Implement log shipping

Log shipping is typically used for disaster recovery scenarios. It can, however, also be used to synchronize data warehouses and scale out a reporting solution. Although log shipping has always been possible with SQL Server, it was introduced as a supported feature with the release of SQL Server 2000, which includes an interface and a number of system tables in the Microsoft system database.

This objective covers how to:

- Architect log shipping
- Configure log shipping
- Monitor log shipping

Architect log shipping

Log shipping has a very simple architecture. It uses a number of SQL Server Agent jobs. When architecting a log shipping solution, make sure you get the schedules for the various log shipping jobs scheduled at the right time. Depending on your RTO and RPO you typically need to let preceding jobs complete before running the next job. Don't forget to revisit your schedules periodically as your database might have grown in size and consequently the jobs take longer to run. As always, it is important to get the security for all the scheduled jobs correct, especially because you are running the jobs on different servers.

The log shipping architecture, shown in Figure 4-1, contains the following elements:

- **Primary server** The primary server contains the database that is going to be log shipped. There can only be one primary server.
- **Primary database** The primary database is the source database on the primary server that is log shipped. The database can't be using the SIMPLE recovery model.
- **Secondary server** The secondary server contains the copy of the primary database that is periodically updated through log shipping. There can be multiple secondary servers. The secondary server can be on a higher version of SQL Server from the primary server.

- **Secondary database** The secondary database is a copy of the primary database that is hosted on the primary server. The secondary database can be potentially used for a reporting purpose (SELECT queries).
- **Monitor server** The monitor server is a separate SQL Server instance that monitors the overall state of the log shipping solution. A monitor server is optional.
- **Backup job** The backup job is a SQL Server Agent job that performs the backup job periodically and logs history to the local and monitor server. The backup job runs on the primary server.
- **Copy job** The copy job is a SQL Server Agent job that performs the backup job periodically and logs history to the local and monitor server. The copy job runs on the secondary server.
- **Restore job** The restore job is a SQL Server Agent job that performs the restore job periodically and logs history to the local and monitor server. It also deletes old file and history. The restore job runs on the secondary server.
- **Alert job** The altert job is a SQL Server Agent job that generates an alert whenever a log shipping error occurs. The alert job runs on the monitor server.
- **Log ship agent** The log ship agent is a process (sqllogship.exe) that is invoked by the SQL Server Agent jobs to perform the log shipping jobs.

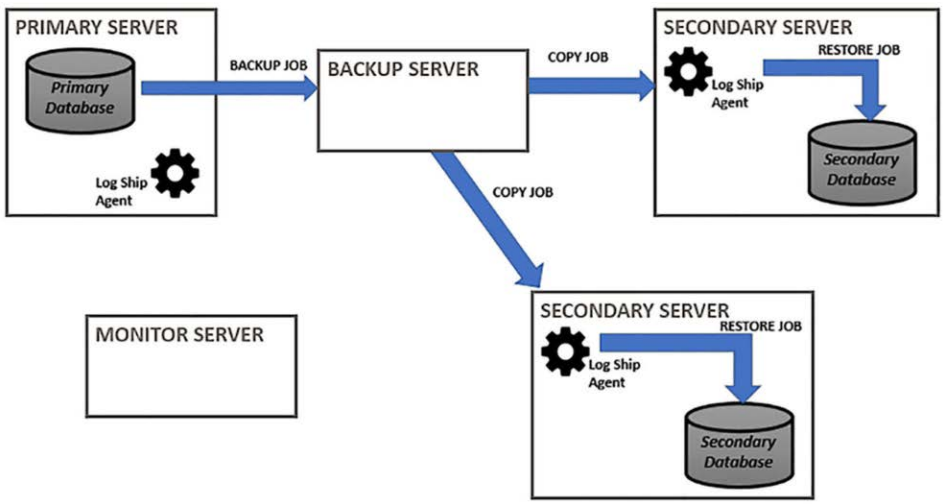

FIGURE 4-1 Log shipping architecture

Log shipping works by scheduling a number of jobs via the SQL Server Agent. The core jobs include:

1. Performing a transaction log backup locally on the primary server. The backup destination can be local or a UNC path.
2. Copying the log backup to a secondary server. Multiple secondary servers are possible.

3. Restoring the log backup on the secondary server. The database on the secondary server can be left in the NORECOVERY or STANDBY state. Under the STANDBY state users will be able to perform SELECT queries against the log shipped database.

Figure 4-2 shows the high-level architecture of a log shipping solution with the previous three steps being performed.

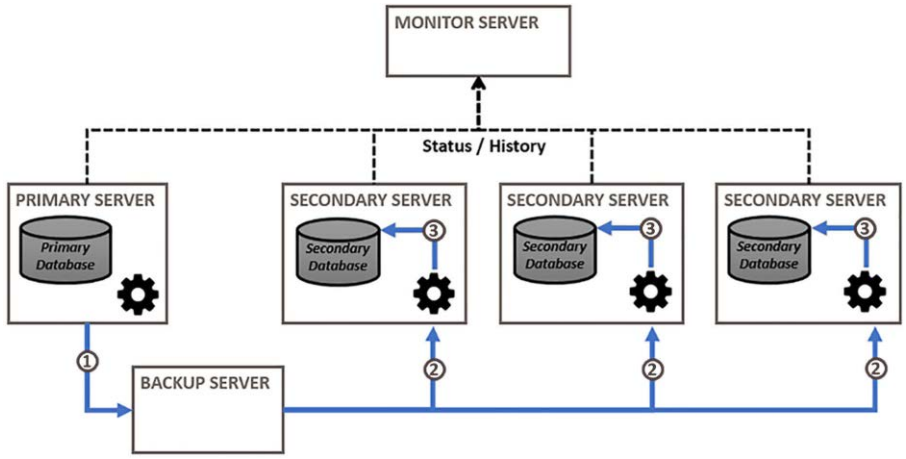

FIGURE 4-2 Log shipping steps

The pros of using log shipping include:

- Support for any edition of SQL Server.
- Scope of protection is at the database level.
- Users can potentially query the secondary database, thereby offloading reporting queries from the primary database.
- Support for multiple secondary servers.
- Support for a delay between changes made to the primary database and the secondary database. This can be important for disaster recovery where an organization might want to protect against accidental deletion of dataset.
- Data changes to the secondary database can be scheduled at a frequency appropriate to the business.

The cons of using log shipping include:

- There is no automatic fail over.
- Manual failover is more complicated than other high-availability technologies.
- Users can't query the database while a transaction log is being restored.
- Data loss is possible. For example, if the primary server or primary database fails, and you cannot access the orphaned log transactions, data will be lost.
- Log shipped databases have to use the full recovery model.

- Log shipping will impact your backup strategy. You will need to re-design your backup strategy so as to use log shipping's log backups instead of your own. If you perform log backup outside of log shipping it will break the log-chain and log shipping will start failing.
- A break in the log backup-chain will break log shipping. The log backup-chain can be broken by changing the database to a SIMPLE recovery model, or by performing a log backup outside of log shipping.
- Log shipping relies on the SQL Server Agent running. If the SQL Server Agent is stopped for any reason on the primary or secondary servers the secondary database will fall further behind the primary database, which can potentially lead to data loss or inability to meet you RPO and RTO objectives.

Use log shipping for the following use cases:

- Disaster recovery within a data center between servers. You can introduce a delay between when log backups are restored on the secondary server in case of user error.
- Disaster recovery between data centers in the case of a data center being unavailable or a disaster happening where the database is lost in the primary data center.
- Disaster recovery that has a delay been transactions being made on the primary database and being replayed on the secondary databases. This is not possible with Availability Groups.
- Disaster recovery between sites that have a long distance between them, are unreliable, are expensive, or have a long latency.
- Offload reporting from the OLTP primary databases. Reports running against the secondary database will no longer cause contention and consume resources on the primary server. The secondary servers can be located closers to the business units.

NOTE **CUSTOM LOG SHIPPING SOLUTION**

Sometimes Microsoft's log shipping implementation is inappropriate for your business needs. In this case, it is possible to create your own custom log shipping solution through SQL Server Agent jobs. The main benefit of using Microsoft's log shipping solution is the ease of deployment and automatic retry logic, which can otherwise be difficult to implement. It is also possible to enhance Microsoft's log shipping implementation by injecting steps into the SQL Server Agent jobs that are created.

Configure log shipping

Use SQL Server Management Studio to configure log shipping, because it is much easier than creating the log shipping script yourself. If you want, you can use SQL Server Management Studio to only generate the log shipping configuration script and not configure log shipping itself. You can then review and save the script before executing it.

To practice setting up log shipping set up the following VMs in Hyper-V:

1. A domain controller (ADDS) for the SQL.LOCAL domain
2. A SQL Server instance (PRIMARY) joined to SQL.LOCAL
3. A SQL Server instance (SECONDARY) joined to SQL.LOCAL
4. A SQL Server instances (database administrator) joined to the domain
5. This server is optional
6. It is used to demonstrate the monitor server
7. You do not have to set up a monitor server
8. A Windows file server (STORAGE) joined to the domain
9. This server is optional
10. It is used for the backup files
11. You could use a share created on the domain controller instead, or either of the SQL Server instances

The following steps show how to configure log shipping from a primary server to a single secondary server. Users will not have access to the secondary server for reporting purposes.

1. Open SQL Server Management Studio and connect to the primary SQL Server instance.
2. Expand the Databases folder.
3. Right-click on the primary database and click on the Options page.
4. Make sure the primary database is using the full recovery model.
5. Click on the Transaction Log Shipping page.
6. Click on the Enable This As A Primary Database In A Log Shipping configuration.
7. Click on the Backup Settings button to configure the log shipping backup on the primary server.
8. Configure the following transaction log backup settings, as shown in Figure 4-3.
   - UNC network path to where the log backups will be performed
   - Optionally, if the log backups will be performed locally, the local path
   - Duration after which the backup files will be automatically deleted

- Duration after which an alert will be generated if backups fail
- The backup job name
- The database backup compression option

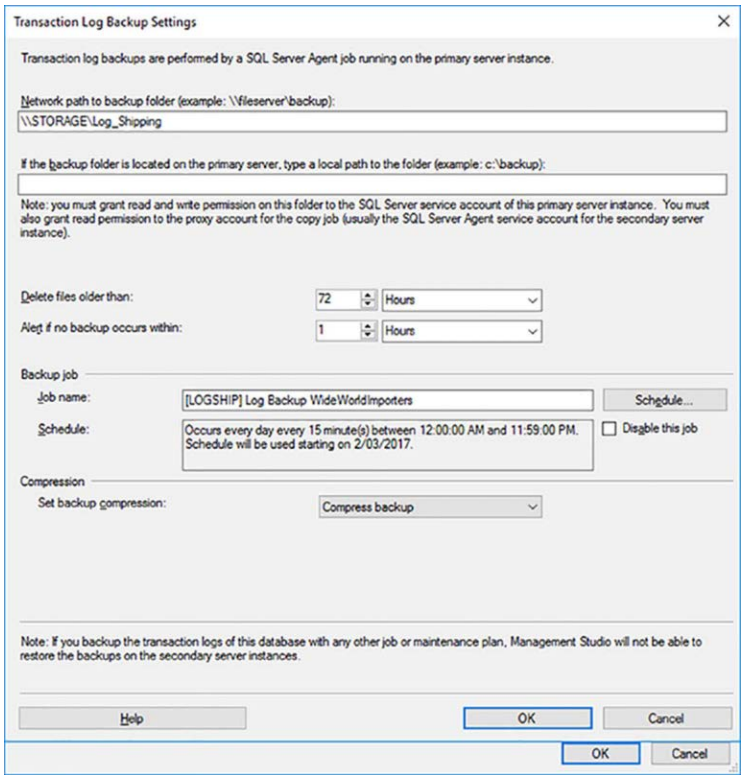

FIGURE 4-3 Enable log shipping for primary database

9. Click on the Schedule button.
10. Configure the log backup schedule to occur daily every 15 minutes.
11. Click on the OK button to close the Transaction Log Backup Settings dialog box.
12. Click on the Add button to add a secondary server.
13. Click on the Connect button to authenticate against the secondary server.
14. Enter the secondary server's name and click on the Connect button.
15. The primary database needs to be initialized on the secondary server before logs can be shipped to it. Configure the following secondary database properties, as shown in Figure 4-4.
    - Secondary database name.
    - Generate a full backup of the primary database and restore it on the secondary server.

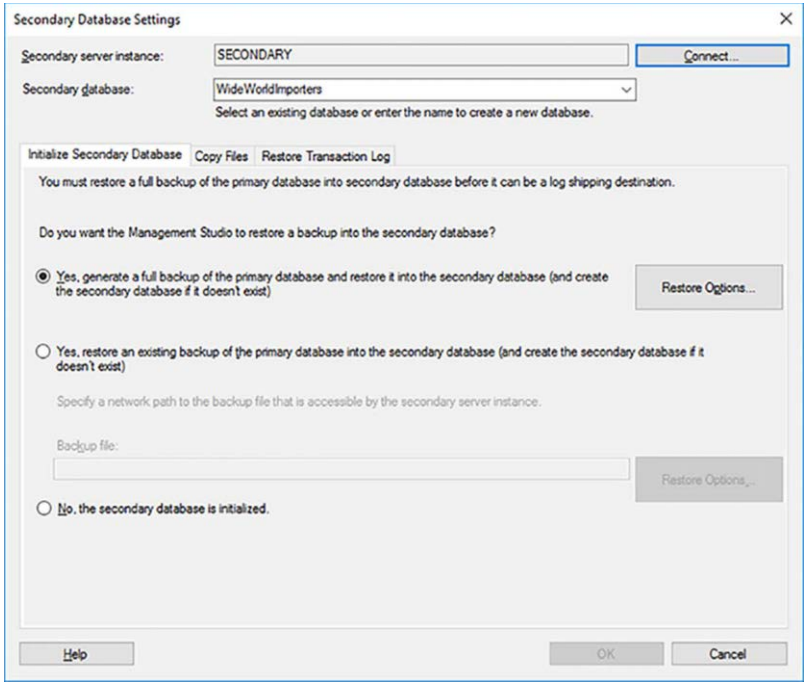

FIGURE 4-4 Initialize Secondary Database

16. Click on the Restore Options button.
17. Configure the secondary database's data and log paths *on the secondary server*.
18. Click on the Copy Files tab.
19. Configure the following properties of the copy job, as shown in Figure 4-5.
    - Destination folder for the copied log backup files. You can use a local path *on the secondary server* or a UNC path.
    - Duration before the log backup files will be deleted.
    - Name for the copy job.
    - A schedule for the copy job, similar to how the schedule for the backup job was configured via the Schedule button.

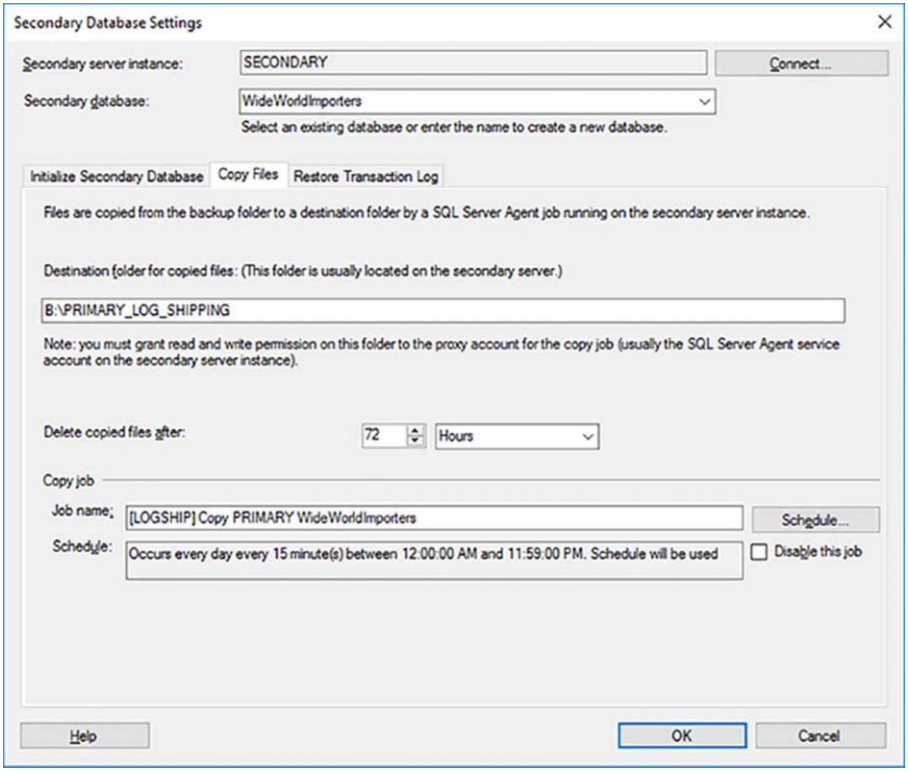

FIGURE 4-5 Log Shipping Copy Job Properties

20. Click on the Restore Transaction Log tab
21. Configure the following properties of the restore transaction log job, as shown in Figure 4-6, and click on the OK button.
    - What recovery model the secondary database will remain in after the restore transaction log completes.
    - With the NORECOVERY recovery model users will not be able to access the secondary database. Subsequent log backups will not be blocked because there are no locks acquired by users within the database.
    - With the STANDBY recovery model users will be able to use the secondary database in a read-only fashion. However, there is potential for these users to block subsequent restore operations. A restore cannot be performed if users have locks acquired in the secondary database. Check the Disconnect Users In The Database When Restoring Backups check box if you want log shipping to immediately disconnect any users before restoring the log. Users might not be happy!

- Whether you want a delay before log backups are restored. *This can be very important for protecting against user errors*, such as accidental modifications to a table or an accidental table truncation.
- Duration before you will be alerted if no restore operation occurs.
- Name for the restore transaction log job.
- A schedule for the restore transaction log job, similar to how the schedule for the backup job was configured via the Schedule button.
- For a data warehouse scenario, you might want to only restore the database once at the end of the day or after midnight. In this case the scheduled restore transaction log job will restore all the log backups required in the correct sequence. Because log shipping keeps track of the history of what has been performed in the [msdb] system database, it is very resilient.
- Your backup, copy and restore jobs can run at different frequencies. It is not uncommon to backup and copy the log files at a faster frequency, such as every 15 minutes, than the restore job, that can run hourly, or even once a day.

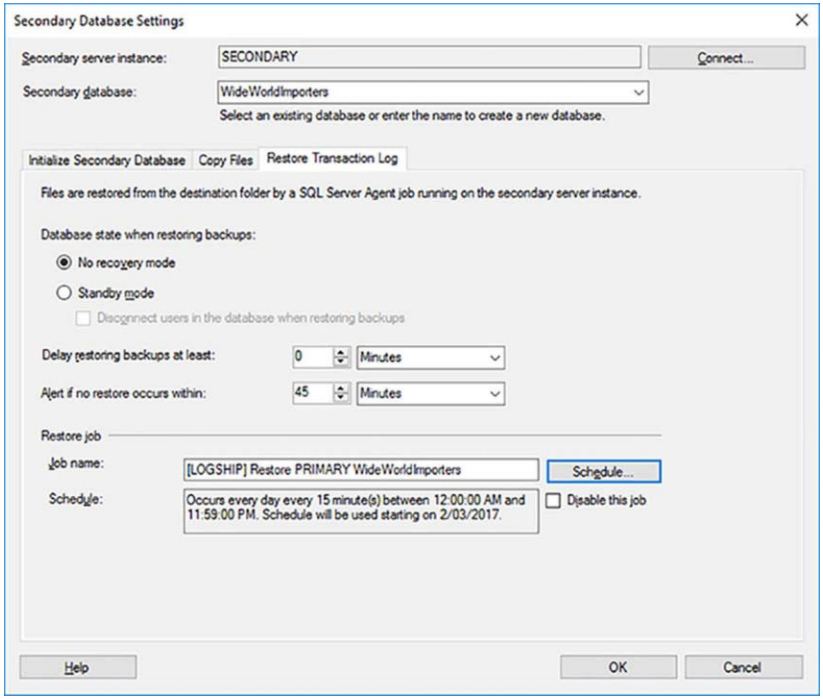

FIGURE 4-6 Log shipping restore transaction log properties

22. Click on the Script drop-down list and select the Script Action To A New Query Window option to review and/or save the log shipping configuration to a Transact-SQL script.

23. Click on the OK button to deploy the log shipping configuration.

Listing 4-1 shows the Transact-SQL script that was generated to configure the log shipping solution.

LISTING 4-1 Log shipping configuration

```
-- Execute the following statements at the Primary to configure Log Shipping
-- for the database [PRIMARY].[WideWorldImporters],
-- The script needs to be run at the Primary in the context of the [msdb] database.
------------------------------------------------------------------------------------
-- Adding the Log Shipping configuration

-- ****** Begin: Script to be run at Primary: [PRIMARY] ******

DECLARE @LS_BackupJobId     AS uniqueidentifier
DECLARE @LS_PrimaryId     AS uniqueidentifier
DECLARE @SP_Add_RetCode     As int

EXEC @SP_Add_RetCode = master.dbo.sp_add_log_shipping_primary_database
        @database = N'WideWorldImporters'
        ,@backup_directory = N'\\STORAGE\Log_Shipping'
        ,@backup_share = N'\\STORAGE\Log_Shipping'
        ,@backup_job_name = N'[LOGSHIP] Log Backup WideWorldImporters'
        ,@backup_retention_period = 4320
        ,@backup_compression = 2
        ,@backup_threshold = 60
        ,@threshold_alert_enabled = 1
        ,@history_retention_period = 5760
        ,@backup_job_id = @LS_BackupJobId OUTPUT
        ,@primary_id = @LS_PrimaryId OUTPUT
        ,@overwrite = 1

IF (@@ERROR = 0 AND @SP_Add_RetCode = 0)
BEGIN

DECLARE @LS_BackUpScheduleUID   As uniqueidentifier
DECLARE @LS_BackUpScheduleID   AS int

EXEC msdb.dbo.sp_add_schedule
        @schedule_name =N'Every 15 minutes'
        ,@enabled = 1
        ,@freq_type = 4
        ,@freq_interval = 1
        ,@freq_subday_type = 4
        ,@freq_subday_interval = 15
        ,@freq_recurrence_factor = 0
        ,@active_start_date = 20170302     -- Change as appropriate
        ,@active_end_date = 99991231
        ,@active_start_time = 0
        ,@active_end_time = 235900
        ,@schedule_uid = @LS_BackUpScheduleUID OUTPUT
        ,@schedule_id = @LS_BackUpScheduleID OUTPUT
```

```
EXEC msdb.dbo.sp_attach_schedule
        @job_id = @LS_BackupJobId
        ,@schedule_id = @LS_BackUpScheduleID

EXEC msdb.dbo.sp_update_job
        @job_id = @LS_BackupJobId
        ,@enabled = 1

END

EXEC master.dbo.sp_add_log_shipping_alert_job

EXEC master.dbo.sp_add_log_shipping_primary_secondary
        @primary_database = N'WideWorldImporters'
        ,@secondary_server = N'SECONDARY'
        ,@secondary_database = N'WideWorldImporters'
        ,@overwrite = 1

-- ****** End: Script to be run at Primary: [PRIMARY]  ******

-- Execute the following statements at the Secondary to configure Log Shipping
-- for the database [SECONDARY].[WideWorldImporters],
-- the script needs to be run at the Secondary in the context of the [msdb] database.
-----------------------------------------------------------------------------------------
-- Adding the Log Shipping configuration

-- ****** Begin: Script to be run at Secondary: [SECONDARY] ******

DECLARE @LS_Secondary__CopyJobId  AS uniqueidentifier
DECLARE @LS_Secondary__RestoreJobId     AS uniqueidentifier
DECLARE @LS_Secondary__SecondaryId     AS uniqueidentifier
DECLARE @LS_Add_RetCode     As int

EXEC @LS_Add_RetCode = master.dbo.sp_add_log_shipping_secondary_primary
        @primary_server = N'PRIMARY'
        ,@primary_database = N'WideWorldImporters'
        ,@backup_source_directory = N'\\STORAGE\Log_Shipping'
        ,@backup_destination_directory = N'B:\PRIMARY_LOG_SHIPPING'
        ,@copy_job_name = N'[LOGSHIP] Copy PRIMARY WideWorldImporters'
        ,@restore_job_name = N'[LOGSHIP] Restore PRIMARY WideWorldImporters'
        ,@file_retention_period = 4320
        ,@overwrite = 1
        ,@copy_job_id = @LS_Secondary__CopyJobId OUTPUT
        ,@restore_job_id = @LS_Secondary__RestoreJobId OUTPUT
        ,@secondary_id = @LS_Secondary__SecondaryId OUTPUT

IF (@@ERROR = 0 AND @LS_Add_RetCode = 0)
BEGIN

DECLARE @LS_SecondaryCopyJobScheduleUID     As uniqueidentifier
DECLARE @LS_SecondaryCopyJobScheduleID     AS int

EXEC msdb.dbo.sp_add_schedule
        @schedule_name =N'Every 15 minutes'
        ,@enabled = 1
        ,@freq_type = 4
```

```
        ,@freq_interval = 1
        ,@freq_subday_type = 4
        ,@freq_subday_interval = 15
        ,@freq_recurrence_factor = 0
        ,@active_start_date = 20170302     -- Change as appropriate
        ,@active_end_date = 99991231
        ,@active_start_time = 0
        ,@active_end_time = 235900
        ,@schedule_uid = @LS_SecondaryCopyJobScheduleUID OUTPUT
        ,@schedule_id = @LS_SecondaryCopyJobScheduleID OUTPUT

EXEC msdb.dbo.sp_attach_schedule
        @job_id = @LS_Secondary__CopyJobId
        ,@schedule_id = @LS_SecondaryCopyJobScheduleID

DECLARE @LS_SecondaryRestoreJobScheduleUID     As uniqueidentifier
DECLARE @LS_SecondaryRestoreJobScheduleID     AS int

EXEC msdb.dbo.sp_add_schedule
        @schedule_name =N'Every 15 minutes'
        ,@enabled = 1
        ,@freq_type = 4
        ,@freq_interval = 1
        ,@freq_subday_type = 4
        ,@freq_subday_interval = 15
        ,@freq_recurrence_factor = 0
        ,@active_start_date = 20170302     -- Change as appropriate
        ,@active_end_date = 99991231
        ,@active_start_time = 0
        ,@active_end_time = 235900
        ,@schedule_uid = @LS_SecondaryRestoreJobScheduleUID OUTPUT
        ,@schedule_id = @LS_SecondaryRestoreJobScheduleID OUTPUT

EXEC msdb.dbo.sp_attach_schedule
        @job_id = @LS_Secondary__RestoreJobId
        ,@schedule_id = @LS_SecondaryRestoreJobScheduleID

END

DECLARE @LS_Add_RetCode2  As int

IF (@@ERROR = 0 AND @LS_Add_RetCode = 0)
BEGIN

EXEC @LS_Add_RetCode2 = master.dbo.sp_add_log_shipping_secondary_database
        @secondary_database = N'WideWorldImporters'
        ,@primary_server = N'PRIMARY'
        ,@primary_database = N'WideWorldImporters'
        ,@restore_delay = 0
        ,@restore_mode = 0
        ,@disconnect_users  = 0
        ,@restore_threshold = 45
        ,@threshold_alert_enabled = 1
        ,@history_retention_period   = 5760
        ,@overwrite = 1
```

```
END

IF (@@error = 0 AND @LS_Add_RetCode = 0)
BEGIN

EXEC msdb.dbo.sp_update_job
        @job_id = @LS_Secondary__CopyJobId
        ,@enabled = 1

EXEC msdb.dbo.sp_update_job
        @job_id = @LS_Secondary__RestoreJobId
        ,@enabled = 1

END
-- ****** End: Script to be run at Secondary: [SECONDARY] ******
GO
```

EXAM TIP

Make sure you familiarize yourself with the key statements and parameters in the log shipping creation script for the exam.

Figure 4-7 shows the log shipping backup job and step created on the primary server. Note how the log shipping back up job does not run any Transact-SQL commands. Instead it invokes the **sqllogship.exe** agent with a number of parameters. The copy and the backup jobs are also run on the secondary server. If you connect to the secondary server in SQL Server Management Studio, the secondary database is permanently in a restoring state.

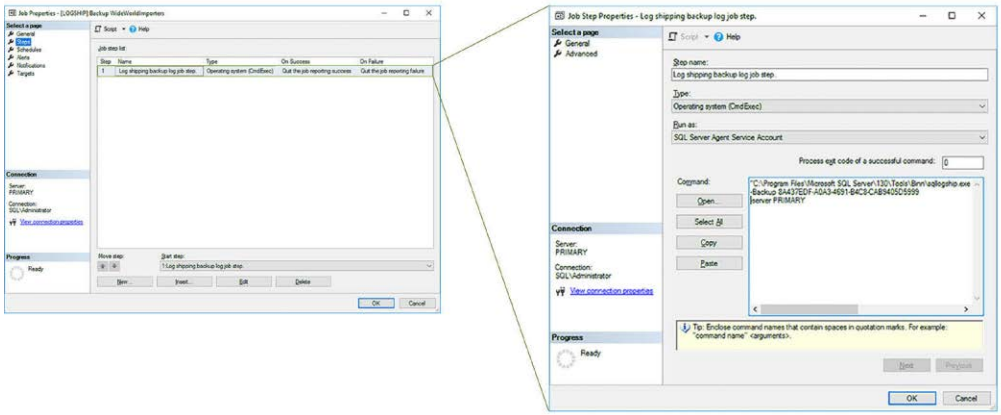

FIGURE 4-7 Log shipping backup job on primary server

NOTE **CUSTOMIZING LOG SHIPPING JOBS**

There is nothing preventing you from customizing the log shipping jobs created by SQL Server Management Server. For example, you could robocopy the log backups immediately after the log copy job step completes to your disaster recovery server.

Because log shipping uses an agent, it is difficult to customize log shipping. That is why it is not uncommon for database administrators to develop and implement their own custom log shipping through Transact-SQL scripts.

The sqllogship.exe agent supports the following parameters:

```
sqllogship
    -server instance_name
{
    -backup primary_id |
    -copy secondary_id |
    -restore secondary_id
}
[ -verboselevel level ]
[ -logintimeout timeout_value ]
[ -querytimeout timeout_value ]
```

To help troubleshoot log shipping you can change the -verboselevel parameter as required. Table 4-2 shows the different levels supported. The default value used is 3.

TABLE 4-2 SQLLOGSHIP.EXE -VERBOSELEVEL PARAMETER OPTIONS

Level	Description
0	Output no tracing and debugging messages
1	Output error-handling messages
2	Output warnings and error-handling messages
3	Output informational messages, warnings, and error-handling messages
4	Output all debugging and tracing messages

Monitor log shipping

It is important to monitor your log shipping to ensure that log shipping is working as expected, because it could potentially impact your RPO/RTO SLAs. Log shipping allows you to create a separate monitor server that will monitor log shipping jobs on the primary and secondary servers. If a customized threshold expires, an alert will be generated to indicate that a job has failed.

The following steps show how to configure a monitor server for your log shipping solution.

1. Open SQL Server Management Studio and connect to the primary SQL Server instance.
2. Expand the Databases folder.
3. Right-click on the primary database and click on the Transaction Log Shipping page.

4. Check the Use A Monitor Server Instance check box.
5. Click on the Settings button to configure the monitor server.
6. Click on the Connect button to authenticate against the monitoring server.
7. Provider the server name and authentication details for the monitor server in the Connect to Server dialog box and click the Connect button.
8. Configure the following details, as shown in Figure 4-8, to configure the monitor server:
   - Credentials to be used by the monitor server. The best and easiest set up is to impersonate the proxy account of the log shipping jobs.
   - The history retention after which history will be deleted.
     - In a production environment, it is not uncommon to configure such information for a number of years.
   - The name of the alert job.

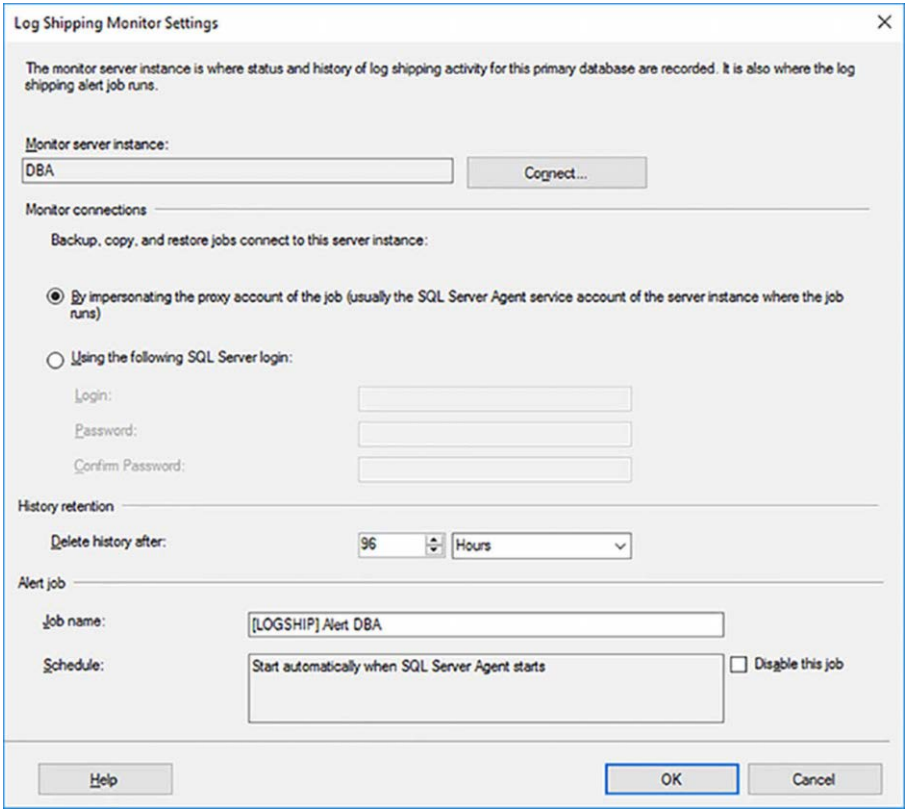

FIGURE 4-8 Configuring the monitor server settings

9. Click on the OK button to close the Log Shipping Monitor Settings dialog box.
10. Click on the OK button for SQL Server Management Studio to configure the log shipping monitor.

Listing 4-2 shows the Transact-SQL script that was generated to configure the log shipping monitor server.

LISTING 4-2 Log shipping configuration.

```
-- ****** Begin: Script to be run at Monitor: [DBA] ******

EXEC msdb.dbo.sp_processlogshippingmonitorsecondary
        @mode = 1
        ,@secondary_server = N'SECONDARY'
        ,@secondary_database = N'WideWorldImporters'
        ,@secondary_id = N''
        ,@primary_server = N'PRIMARY'
        ,@primary_database = N'WideWorldImporters'
        ,@restore_threshold = 45
        ,@threshold_alert = 14420
        ,@threshold_alert_enabled = 1
        ,@history_retention_period    = 5760
        ,@monitor_server = N'DBA'
        ,@monitor_server_security_mode = 1
-- ****** End: Script to be run at Monitor: [DBA] ******
```

The log shipping monitor server runs the log shipping alert job. Instead of running an executable, it executes the sys.sp_check_log_shipping_monitor_alert system stored procedure.

With the log shipping monitor configured you can now execute a number of reports to see the current state of log shipping. The log shipping reports will be different depending on whether you execute them from the monitor, the primary, or the secondary log shipping server.

To generate a report, use the following steps:

1. Open SQL Server Management Studio and connect to the log shipping SQL Server instance.
2. Right-click on the SQL Server instance that you want to execute the report against.
3. Select the Reports option.
4. Select the Standard Reports option.
5. Click the Transaction Log Shipping Status report.

Figure 4-9 shows the Transaction Log Shipping Status report generated on the monitoring server. It shows all the servers in the log shipping configuration.

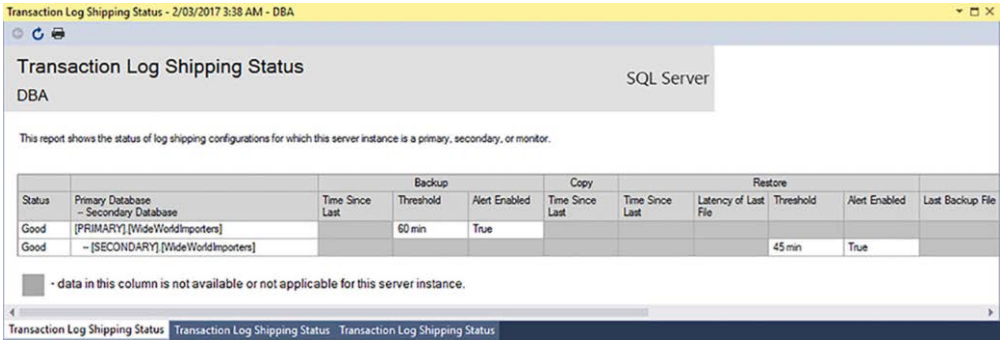

FIGURE 4-9 Transaction Log Shipping Status report on monitoring server

Skill 4.4: Implement Availability Groups

Introduced in SQL Server 2012, Availability Groups revolutionized high availability by dropping the reliance on specific hardware and giving you the option of scaling out your database solution. With Availability Groups, high availability is achieved by combining the capabilities of Windows with the Database Engine. Consequently, Availability Groups are hardware, shared storage, and cloud provider agnostic. Don't automatically use Availability Groups because they are "better than clustering", or because "clustering is going away." Both of those assertions are false.

This objective covers how to:

- Architect Availability Groups
- Configure Windows clustering
- Create an Availability Group
- Configure read-only routing
- Manage failover
- Create distributed Availability Group

Architect Availability Groups

Availability Groups have the most complex architecture out of all the high availability technologies, so make sure you understand how they work, their limitations and how best to implement them. Do not be seduced by automatically using Availability Groups because they represent new technology. It is perfectly valid to continue using Failover Clustering as a high availability solution, because it is not going away and Microsoft continues to improve it in every release of Windows Server. An organization should have some operational maturity to successfully manage Availability Groups. Furthermore, certain edge cases may degrade performance. Ideally you should perform a Proof-of-Concept (POC) before deploying Availability Groups into production.

The Availability Group architecture, shown in Figure 4-10, contains the following elements:

- **Availability group** An Availability Group is a container that represents a unit of fail over. An Availability Group can have one or more user databases. When an Availability Group fails over from one replica (a SQL Server instance) to another replica, all of the databases that are part of the Availability Group fail over. This might be particularly important for multi database solutions like Microsoft Biztalk, Microsoft SharePoint, and Microsoft Team Foundation Server (TFS).
- **Primary replica** A primary replica is a SQL Server instance that is currently hosting the Availability Group that contains a user database that can be modified. You can only have one primary instance at any given point in time.
- **Secondary replica** A secondary replica is a SQL Server instance that is hosting a copy of the Availability Group. The user databases within the Availability Group hosted on a secondary replica can't be modified. Different versions of SQL Server support a different maximum number of secondary replicas:
  - SQL Server 2012 supports four secondary replicas
  - SQL Server 2014-2016 supports eight secondary replicas
- **Failover partner** A failover partner is a secondary replica that has been configured as an automatic failover destination. If something goes wrong with a primary replica the Availability Group will be automatically failed over to the secondary replica acting as a failover partner. Different versions of SQL Server support a different maximum number of failover partners:
  - SQL Server 2012-2016 supports one failover partner
  - SQL Server 2016 supports two failover partners
- **Readable secondary replica** A readable secondary replica is a secondary replica that has been configured to allow select queries to run against it. When a SQL Server instance acts as a readable secondary the database engine will automatically generate temporary statistics in the [tempdb] system database to help ensure optimal query performance. Furthermore, row-versioning, which also uses the [tempdb] system database, is used by the database engine to remove blocking contention.
- **Availability group listener** An Availability Group listener is a combination of a virtual network name (VNN) and virtual IP (VIP) address that can be used by client applications to connect to the databases hosted within the Availability Group. The VNN and its VIP is stored as a DNS entry in Active Directory (AD). An Availability Group can have multiple Availability Group listeners. The primary use of an Availability Group listener is to provide a level of abstraction from the primary replica. Applications connect to the Availability Group listener and not the current primary replica's physical server name.
- **Primary database** A primary database is a user database hosted on the primary replica of an Availability Group replica.
- **Secondary database** A secondary database is a user database hosted on any of the secondary replicas of an Availability Group.

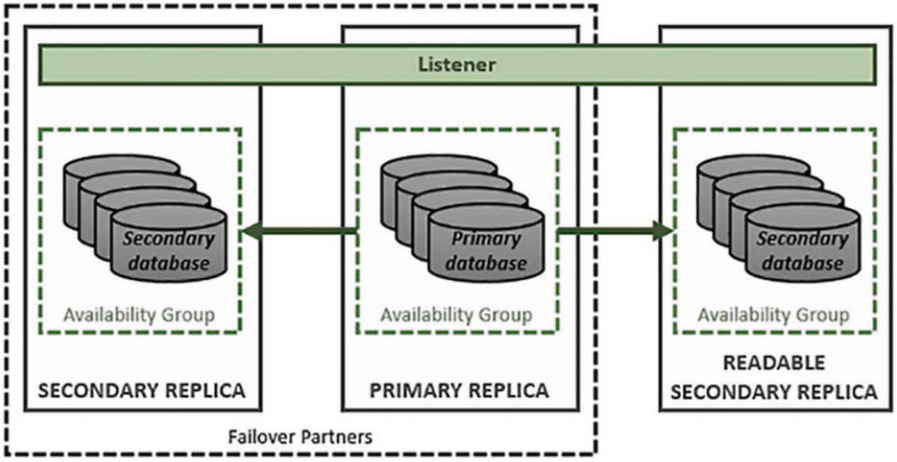

FIGURE 4-10 Availability group architecture

The primary replica can have multiple roles. Any given SQL Server instance could be both a readable secondary and a failover partner.

Availability Groups work by automatically transmitting up to 60KB transaction log buffers (in memory structures, also referred to as log blocks, that are written to first, before they are flushed to disk using the Write-Ahead-Logging (WAL) protocol) as they fill up or when a commit transaction event occurs. Consequently, the primary database and secondary database can be synchronized in real-time.

Availability groups support two different synchronization modes:

- **Synchronous commit** With synchronous commit mode the secondary replica is kept synchronized synchronously, in real-time. The secondary database is an exact binary match of the primary database. Because the databases are kept in sync, this implies a performance overhead on primary, which can effect performance; both databases wait to be in sync before the commit. Synchronous mode facilitates the failover capability of Availability Groups. Ideally, with synchronous mode you have very low network latency between the primary replica and secondary replica. If there is a network issue between the primary replica and the secondary replica the Availability Group will automatically switch over to asynchronous mode. This allows transactions to still be completed on the synchronous replica if the secondary replica is offline or there is a network issue.
- **Asynchronous commit** With asynchronous mode the secondary replica is kept synchronized asynchronously with no guarantee that the primary database and secondary database are an exact match at any given point in time and space. The primary replica transmits the log buffers as quickly as it can. There should be minimal or no impact to the transactions running in the primary database. Asynchronous mode tolerates a higher network latency, and is typically used between data centers.

Figure 4-11 shows how synchronous commit works between replicas in an Availability Group. The key to the synchronous commit is to harden the log on the secondary replica as quickly as possible and send that acknowledgement back to the primary replica.

1. A client application starts a transaction in the primary database.
2. The transaction starts consuming log blocks with operations that need to be performed to complete the transaction.
   - In the background, the secondary replica is requesting the Log Blocks to be transmitted. The primary and secondary replica need to coordinate what needs to be transmitted using the Log Sequence Number (LSN) and other information.
3. The log block becomes full or a commit transaction operation is performed. The database engine's Log Manager persists (flushes) the Log Block to the log file on the disk and to the *Log Pool* used by Availability Groups.
4. The *Log Capture* thread reads the Log Block from the Log Pool and sends it to all secondary replicas.
   - There is a separate log capture thread for each secondary replica. This allows for parallel updating of secondary replicas.
   - The log content is compressed and encrypted before being sent out on the network.
   - The log content is compressed and encrypted before it gets sent to the secondary replica.
5. The *Log Receive* thread on the secondary replica receives the Log Block.
6. The Log Receive thread writes the Log Block to the Log Cache on the secondary replica.
7. The Redo thread applies the changes from the Log Block to the database as the Log Cache is being written to:
   - There is a separate redo thread per secondary database.
   - When the Log Block fills, or a commit log operation is received, the Log Cache is hardened onto the disk where the transaction log of the secondary database is located.
8. An acknowledgement is sent by the synchronous secondary replica to the primary replica to acknowledge that the log has been hardened. This is the key step because it guarantees that no data loss is possible.

IMPORTANT **HARDENING THE LOG ON THE SECONDARY REPLICAS**

Hardening the log buffers on the secondary represents the key operation in Availability Groups as it means no data loss is possible, even if the secondary replica crashes.

9. If the Redo thread falls behind the rate at which the Log Blocks are being written to and flushed out of the log cache, it starts reading the log blocks from the transaction log on disk and apply them to the secondary database.

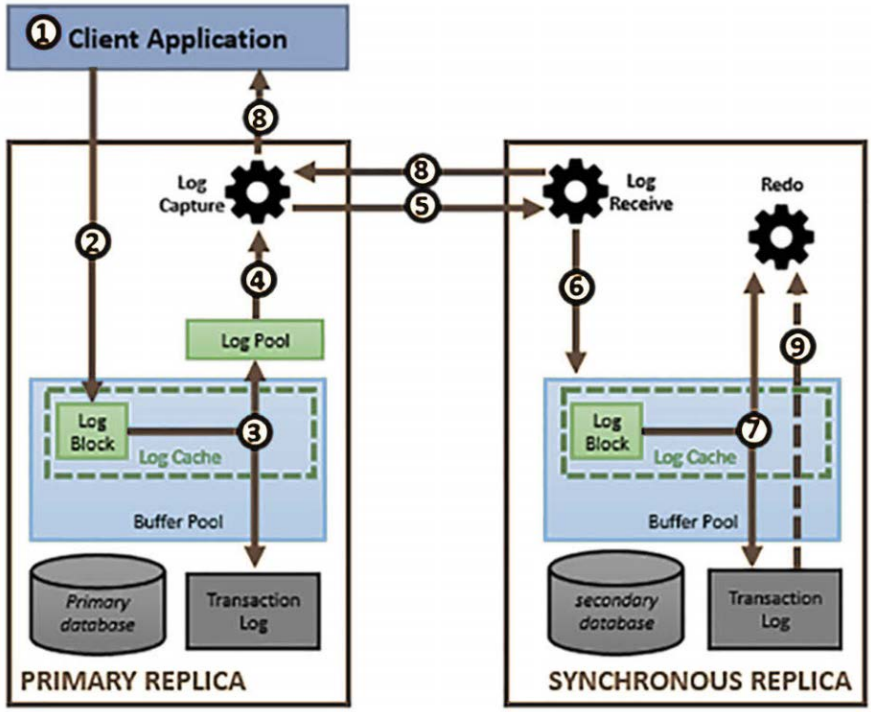

FIGURE 4-11 Availability Group Synchronous Commit

Figure 4-12 shows how asynchronous commit works between replicas in an Availability Group. The process is similar to the synchronous commit process, except that the acknowledgement message of a successful commit is sent after the log blocks are persisted locally on the Primary Replica. The key to the asynchronous commit is to minimize the impact on the Primary Replica.

1. A client application starts a transaction in the primary database.
2. The transaction starts consuming Log Blocks with operations that need to be performed to complete the transaction.
   - In the background, the secondary replica requests the Log Blocks to be transmitted. The primary and secondary replica needs to coordinate what needs to be transmitted using the Log Sequence Number (LSN) and other information.
3. The Log Block becomes full or a commit transaction operation is performed. The database engine's Log Manager persists (flushes) the Log Blocks to the log file on the disk and to the *Log Pool* used by Availability Groups.
4. If all of the secondary replicas are using asynchronous commit mode, the acknowledgement of a successful commit is effectively sent to the client application.

- Concurrently, the Log Capture thread reads the Log Blocks from the log pool and transmits them to the secondary replica.
- There is one Log Capture thread per replica, so all replicas are synchronized in parallel.
- The log content is compressed and encrypted before being sent on the network.

5. On the secondary replica, the Log Receive thread receives the Log Blocks from the network.
6. The Log Receive thread writes the received Log Blocks to the Log Cache on the secondary replica.
7. As the Log Blocks are written to the Log Cache, the Redo thread reads the changes and applies them to the pages of the database so that it will be in sync with the primary database.
   - When the Log Cache on the secondary becomes full, or a commit transaction log record is received, the contents of the log cache is hardened onto the disk of the secondary replica.
8. If the Redo thread falls behind the rate at which the Log Blocks are being written to and flushed out of the Log Cache, it starts reading the Log Blocks from the transaction log on disk, applying them to the secondary database.

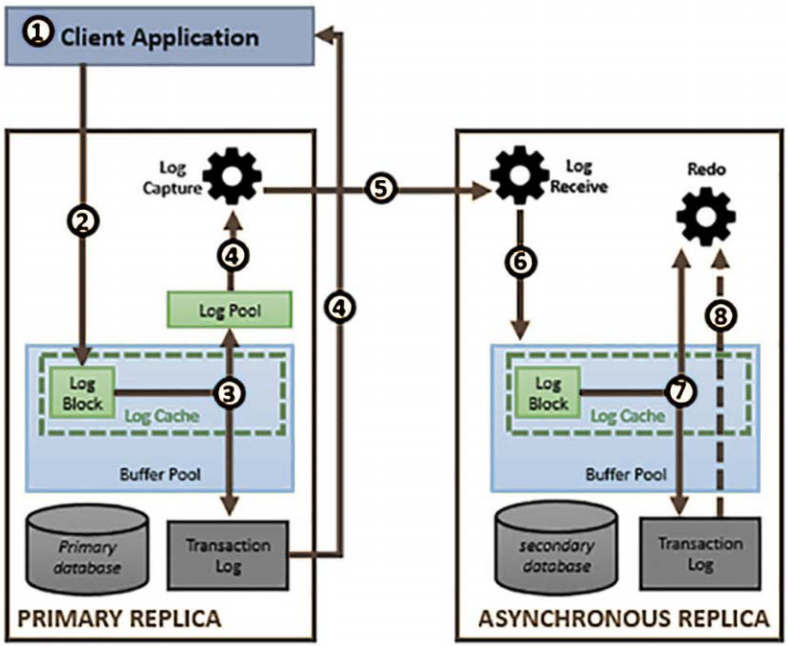

FIGURE 4-12 Availability Group Asynchronous commit

Log stream compression was introduced in SQL Server 2014 as a means of improving the performance of Availability Groups. However, the default behavior of log stream compression has changed in SQL Server 2016:

- Log stream compression is disabled by default for synchronous replicas. This helps ensures OLTP performance is not slowed down on the primary replica. Log stream compression consumes more processor resources and adds a latency.
  - You can change this default behavior by enabling trace flag 9592.
- Log stream compression is enabled by default for asynchronous replicas.
  - You can change this default behavior by enabling trace flag 1462.
- Log stream compression is disabled by default for Automatic Seeding to reduce processor usage on the primary replica.
- You can change this default behavior by enabling trace flag 9567.

The release of SQL Server 2016 brought support for Basic Availability Groups, which are designed to replace Database Mirroring in Standard Edition. You should no longer be implementing Database Mirroring on SQL Server 2016, because it has been deprecated and is scheduled to be dropped from the product in a future release. Basic Availability Groups are considered a replacement for Database Mirroring, so their limitations "mimic" the limitations of Database Mirroring.

Basic Availability Groups have a number of limitations, including:

- Limited to two replicas (primary and secondary).
- Only support one database per Availability Group.
- Can't add a replica to an existing Availability Group.
- Can't remove a replica to an existing Availability Group.
- Can't upgrade a basic Availability Group to an advanced Availability Group.
- Only supported in Standard Edition.
- No read access on secondary replica (no support for readable secondaries).
- Backups cannot be performed on the secondary replica.

The pros of using Availability Groups include:

- Support for any edition of SQL Server
  - SQL Server 2016 Standard Edition only supports Basic Availability Groups
- Provides automatic failover

IMPORTANT **AUTOMATIC FAILOVER IN AVAILABILITY GROUPS**

Availability groups will not failover with certain issues at the database level, such as a database becoming suspect due to the loss of a data file, deletion of a database, or corruption of a transaction log.

- Typically, provides faster failover when compared to failover clustering. This is because when there is a failover event in a failover cluster the SQL Server instance has to be started on the node to which you are failing over to. This can potentially take longer as the binaries have to load into memory, memory has to be allocated to the Database Engine and the databases have to be automatically recovered. Typically though, this is not an important factor in reality, as there are other considerations that are more important.
- Automatic page repair. Each replica tries to automatically recover from a page corruption incident on its local database. This is limited to certain types of errors that prevent reading a database page.
  - If the primary replica cannot read a page it broadcasts a request for a correct copy to all the secondary replicas and replaces the corrupt page from the first secondary replica that provides the correct page.
  - If a secondary replica can't read a page, it requests a correct copy of the page from the primary replica.
- Supports 2 failover partners (with the release of SQL Server 2016).
- No data loss is possible between two synchronous replicas, since data is modified on both in real-time.
- Does not rely on shared storage, which represents a single point in failure.
  - Each replica has its own separate copy of the database.
- Does not require a SAN, which can be expensive, slow, or not available in various cloud providers.
- Typically, can provide much faster performance due to the ability to use local flash storage attached to very fast buses such as PCIe. SANs cannot provide this level of storage performance.
- Scope of protection is at the database or database group level.
  - A group of databases can be protected together, which can be important for software solutions such as Microsoft SharePoint, Microsoft BizTalk, and Microsoft Team Foundation Services (TFS).
- Support for up to secondary eight replicas.
- Support for both synchronous and asynchronous modes. This flexibility is important for implementing Availability Groups within and between data centers, depending on business requirements and technical constraints.
- Read operations can be offloaded from the primary replica to readable secondaries. This allows you to scale out your database solution in certain use cases. This represents one of the major benefits of Availability Groups over other high availability solutions.
- Backup and database consistency check operations can be offloaded from the primary replica to the secondary replica.
  - Secondary replicas support performing log backups and copy-only backups of a full database, file, or filegroup.

IMPORTANT **LICENSING SECONDARY REPLICAS**

Offloading read operations to the secondary replica will require the secondary replica to be licensed. Likewise, offloading backup and database consistency check operations to the secondary replica will require the secondary replica to be licensed.

The cons of using Availability Groups include:

- Replica databases have to use the full recovery model.
  - Some production database solutions should not use the full recovery model due to the amount of transaction log activity that they incur. An example of this includes the search and logging databases in Microsoft SharePoint.
- They are much more difficult to manage.
  - Logins are not automatically synchronized between replicas. You can take advantage of contained databases to help mitigate this.
  - SQL Server Agent jobs, alerts, and operators are not automatically synchronized between replicas.
  - Patching SQL Server instances are more complicated than failover clustering, especially where there is a lot of database modification during any potential outage window. You don't want the Availability Groups to send a queue to grow to a size such that it can never synchronize, and you will be forced to re-initialize the replica database.
- No support for providing a delay between when changes are made on the primary database and the secondary database.
- Impacting database performance in certain highly transactional workloads in OLTP database scenarios.
- Might not support synchronous mode where your network is unreliable or has a high latency, as in the case between data centers.
- Might not be supported by certain applications. Engage your application developers or vendor to determine whether there are potentially any issues.
- You are limited with what operations can be performed on a database replica. In such cases, you have to remove the database from the Availability Group first. For example, the following operations can't be performed on a database that is part of an Availability Group:
  - Detaching the database
  - Taking a database offline
- Does not fully support Microsoft Distributed Transaction Coordinator (DTC or MSDTC). This depends on the version of SQL Server that you are using and how your applications used the DTC.
  - SQL Server 2016 has *limited support for DTC*.

MORE INFO **SUPPORT FOR DTC IN AVAILABILITY GROUPS**

For more information about how DTC is supported in SQL Server 2016 and Window Server 2016 visit *https://blogs.technet.microsoft.com/dataplatform/2016/01/25/sql-server-2016-dtc-support-in-availability-groups/ and https://msdn.microsoft.com/en-us/library/ms366279.aspx*.

Use Availability Groups for the following use cases:

- Providing a high availability solution where there is no shared storage.
- Providing a high availability solution where the business does not want to use shared storage. Different reasons include:
  - Poor performance of your shared storage.
  - Expense of your shared storage.
  - Shared storage represents a single point of failure.
- Providing high availability or disaster recovery between data centers without having to rely upon geo-clustering/stretch clusters that rely on more complicated and expensive storage synchronization technology.
- Offloading reporting from the primary OLTP database. This is where Availability Groups really shine, as they represent the only scale-out technology within SQL Server. This can also be implemented between different sites if users don't require the data to be 100% up to date, as in the case of a data warehouse where they are reporting on historical data.
- Providing high availability between more than three data centers. With SQL Server 2016's ability to have two failover partners, you can effectively build a triple redundant solution.

SQL Server 2014 introduced the following enhancements to Availability Groups:

- Number of secondary replicas was increased to eight.
- Increased availability of readable secondaries, such as if the primary replica became unavailable.
- Enhanced diagnostics through new functions like `is_primary_replica`.
- New DMVs, such as `sys.dm_io_cluster_valid_path_names`.

SQL Server 2016 added the following:

- Basic Availability Groups with Standard Edition
- Distributed Availability Groups
- Domain-independent Availability Groups (Active Directory is no longer required)
- Improvements in the log transport's performance
- Load balancing between readable secondary replicas
- Support for Group Managed Service Accounts (GMSA)
- Support for two automatic failover targets

- Automatic seeding of databases through the log transport
- Limited Microsoft Distributed Transaction Coordinator (MSDTC) support
- Support for updatable columnstore indexes on secondary replicas
- Support for encrypted databases
- Support for SSIS Catalog
- Improvements in database level failover triggers

EXAM TIP

Make sure you familiarize yourself, at least at a high level, with the new Availability Groups features in SQL Server 2016. Undoubtedly, the exam writers will be seduced by writing exam questions that will test on the new SQL Server 2016 features.

Architect readable secondaries

One of the major benefits of Availability Groups is to offload your reports and read-only operations from the primary replica. By offloading read operations to these secondary replicas you remove the contention created readers blocking writers and vice versa. Your read operations are also not competing for the same processor, memory and storage resources as your OLTP operations.

Readable secondaries create temporary statistics inside the [tempdb] system database to help optimize query performance on that particular readable secondary. If you have multiple readable secondaries servicing different reports it is quite possible for each readable secondary to have a different set of temporary statistics.

Readable secondaries do not block the primary replica from continually updating the secondary database. The readable secondary replicas achieve this by taking advantage of snapshot isolation, which in turn relies on row-versioning. Row-versioning heavily relies on the [tempdb] system database, so make sure it is optimally configured on fast storage.

IMPORTANT **READABLE SECONDARY OVERHEAD**

Because readable secondaries take advantage of row-versioning inside the database engine, they introduce a 14 byte overhead on the primary database. Remember, the primary and secondary databases have to be a binary identical copy of each other. So, the secondary database can never be modified directly. Consequently, when you configure a readable secondary the primary replica starts to automatically add the 14 byte overhead to all data and index pages as they get modified. This can potentially degrade performance, and cause more page splits and fragmentation in your primary database.

Availability groups allow you to fine tune how applications will be able to connect to these read-only replicas. When you configure a replica to be a readable replica you have the following options:

- **Read only** With a read only secondary database any application will be able to connect to the secondary database.
- **Read only intent** With a read only intent secondary database only "modern" applications that support the ApplicationIntent=ReadOnly or Application Intent=ReadOnly connection string parameter will be able to connect to the secondary database.

If you have a number of readable replicas in your Availability Group you can set up routing rules for how applications will be automatically redirected to a readable secondary when they connect to the Availability Group via the listener.

SQL Server 2016 introduced the ability to configure load-balancing across a set of your readable secondary replicas.

Configure Windows clustering

Availability groups rely on the Windows Server Failover Clustering (WSFC) feature to help facilitate high availability and automatic failover. You need to install WSFC on all of the nodes of your failover cluster, and create a cluster before you can configure an Availability Group in SQL Server.

To practice setting up Availability Groups, set up the following VMs in Hyper-V:

1. A domain controller (ADDS) for the SQL.LOCAL domain
2. A SQL Server instance (REPLICA1) joined to SQL.LOCAL
3. A SQL Server instance (REPLICA2) joined to SQL.LOCAL
4. A SQL Server instance (REPLICA3) joined to SQL.LOCAL
5. A Windows file server (STORAGE) joined to the domain:
6. This server is optional
7. It is used for the backup files
8. You could use a share created on the domain controller instead, or either of the SQL Server instances

The following steps show how to install the Windows failover clustering feature

1. Log into the first node that you plan to set up as an Availability Group as a domain administrator.
2. Open up Server Manager.
3. Choose Add Roles And Features from the Manage drop-down list.
4. In the Add Roles And Features Wizard select the Next button.
5. Choose the Role-Based Or Feature-Based Installation type and click on Next.
6. Ensure that the local server is selected in the Server Pool and click on the Next button.
7. Do not install any roles. Click on the Next button in the Select Server Roles page.
8. Select the Failover Clustering check box to install the Failover Clustering feature.

9. The Add Roles And Features Wizard will, by default, want to install the Failover Clustering tools and Powershell modules. Confirm this action by clicking on the Add Features button.
10. Confirm that you are installing Failover Clustering and the related tools before clicking on the Install button to begin the installation.
11. Confirm that the installation was successful and click on the Close button to finish the installation. (A reboot might be required, in which case the wizard will prompt you to do that.)
12. Repeat the above steps on the remaining nodes that will make up the Availability Group replicas. In this chapter, we will be configuring an Availability Group across 3 replicas.

After installing the failover clustering feature on all the planned Availability Group replicas, you need to create a failover cluster.

The following steps show how to install the Windows failover clustering feature:

1. Open Failover Cluster Manager, which has now been installed on your server.
2. Click on the Create Cluster action in the right-most pane. This will start the Create Cluster Wizard.
3. Click on the Next button in the Before You Begin page of the Create Cluster Wizard.
4. Enter the name of the server that you want to add to the failover cluster and click on the Add button. The Create Cluster Wizard will validate the server's existence and add it to the bottom text box using its fully qualified domain name (FQDN).
5. Repeat Step 4 for all of the servers that you wish to add.
6. Click on the Next button as shown in Figure 4-13.

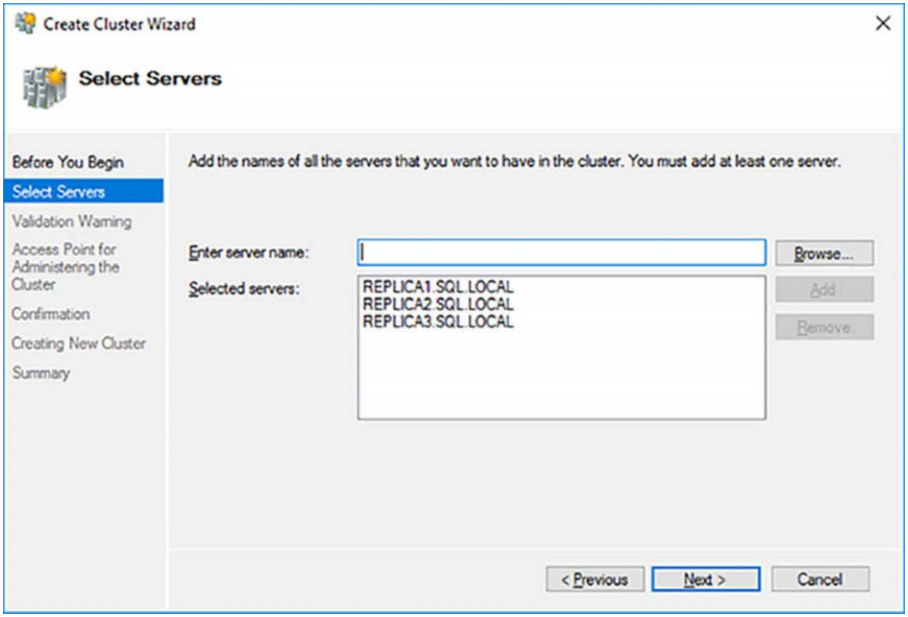

FIGURE 4-13 Selected servers for failover cluster

7. You need to validate that your Windows servers are capable of running a failover cluster that will be supported by Microsoft. Click on the Next button to run the configuration validation tests.
8. Click on the Next button in the Before You Begin page of the Validate A Configuration Wizard.
9. It is a best practice to run all of the cluster validation tests. In your case, as there is no shared stored used in Availability Groups, the validation test might generate some warnings. Click on the Next button to start the validation tests.
10. Review the servers to test and the tests that will be run. Click on the Next button to start the failover cluster validation tests. Figure 4-14 shows the cluster validation tests executing.

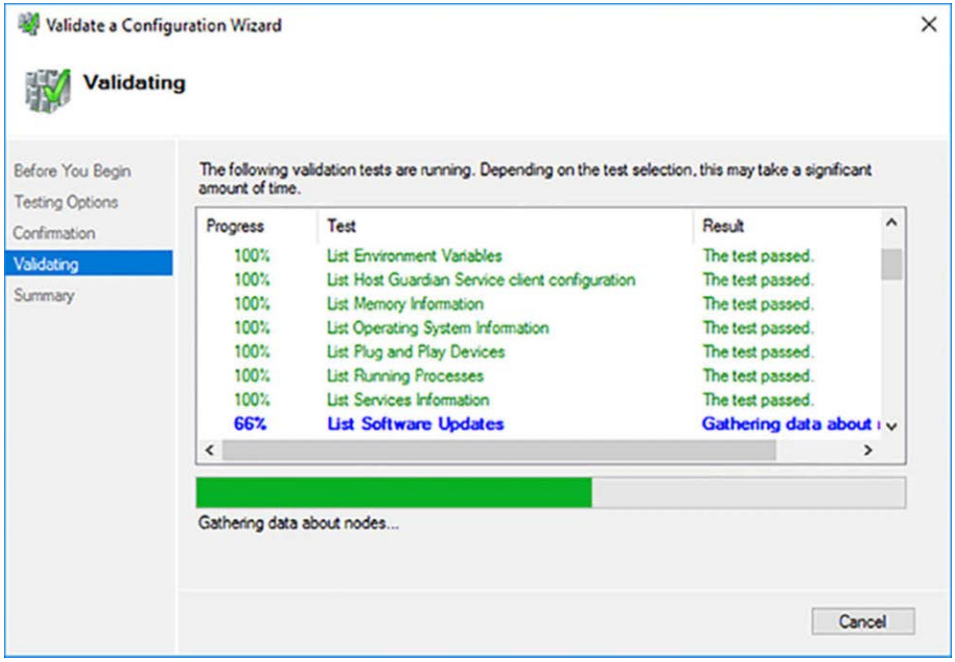

FIGURE 4-14 Cluster validation tests executing

11. As expected, the shared disk validation test has failed, because there are none. Click on the View Report button to see if there are any other problems.
12. Review the Failover Cluster Validation Report, shown in Figure 4-15. In this case the failed storage tests, shown in Figure 4-16, are fine because you will not be using shared disks. The network communication warnings, shown in Figure 4-17, are highlighting the lack of redundancy at the network level between the failover cluster's node. This should be fine. You could, for example, provide redundancy by having multiple NICs in a Windows Server NIC team.

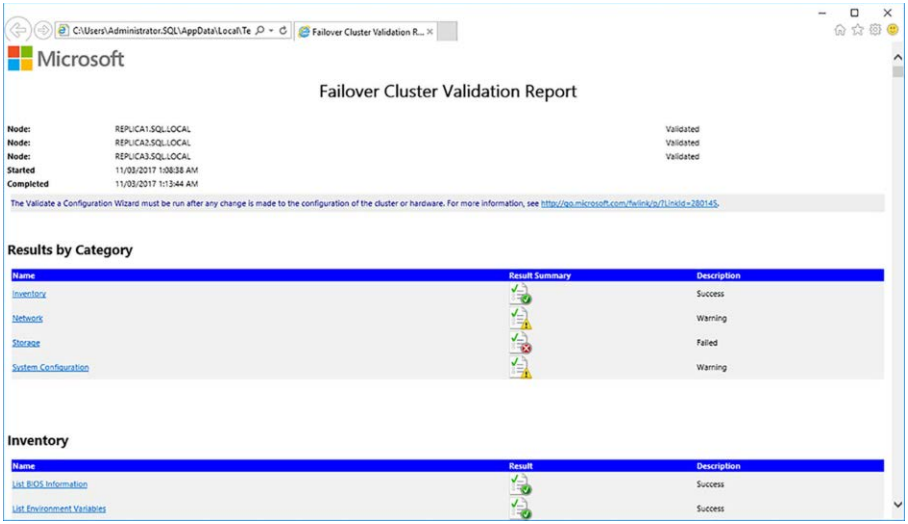

FIGURE 4-15 Failover cluster validation report

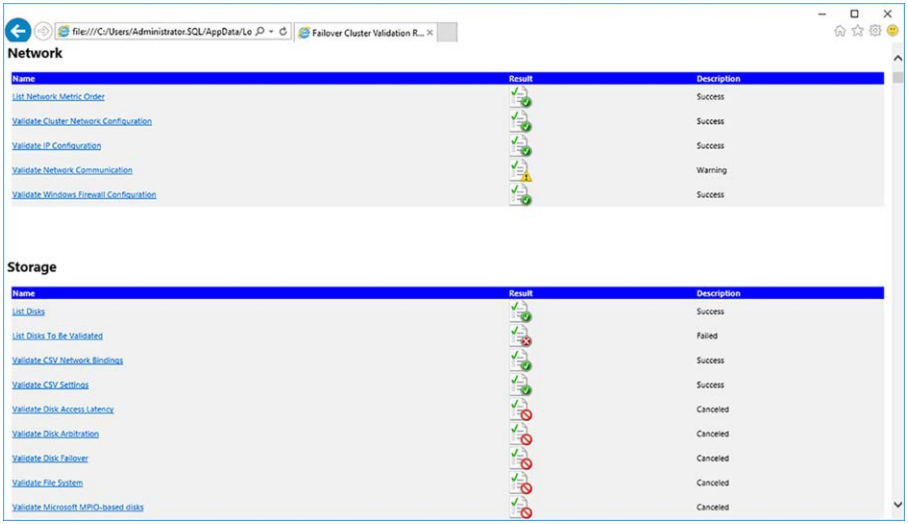

FIGURE 4-16 Failover cluster validation failed storage tests

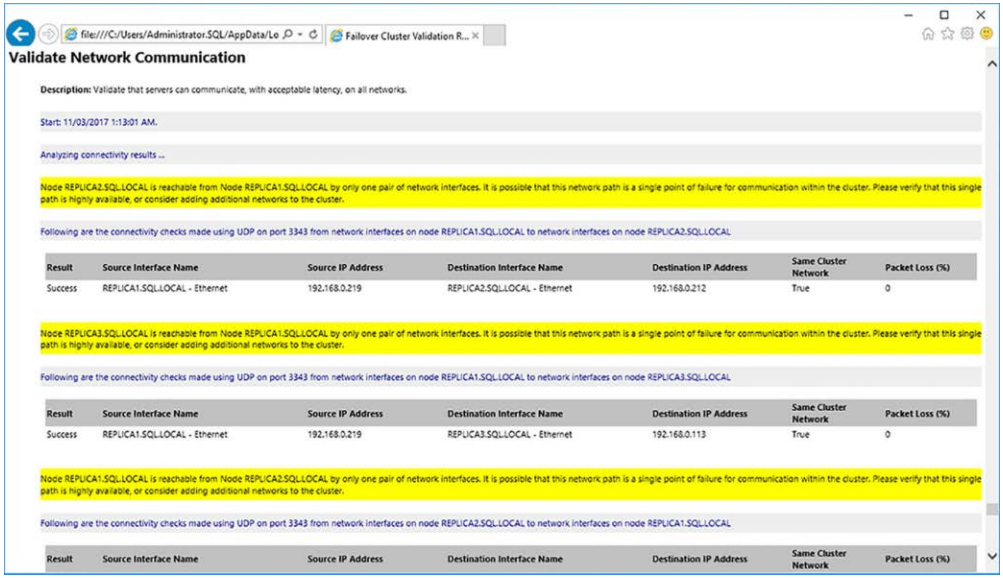

FIGURE 4-17 Failover cluster validation network communication test warnings

13. Address any errors in the Failover Cluster Validation Report, if any.
14. Re-run the Failover Cluster Validation Report, as necessary.
15. Save the Failover Cluster Validation Report. It can be re-run at any time.
16. Close the Failover Cluster Validation Report.
17. Close the Validate a Configuration Wizard by clicking on the Finish button.
18. Provide a NetBIOS name and IP address for the failover cluster, as shown in Figure 4-18.

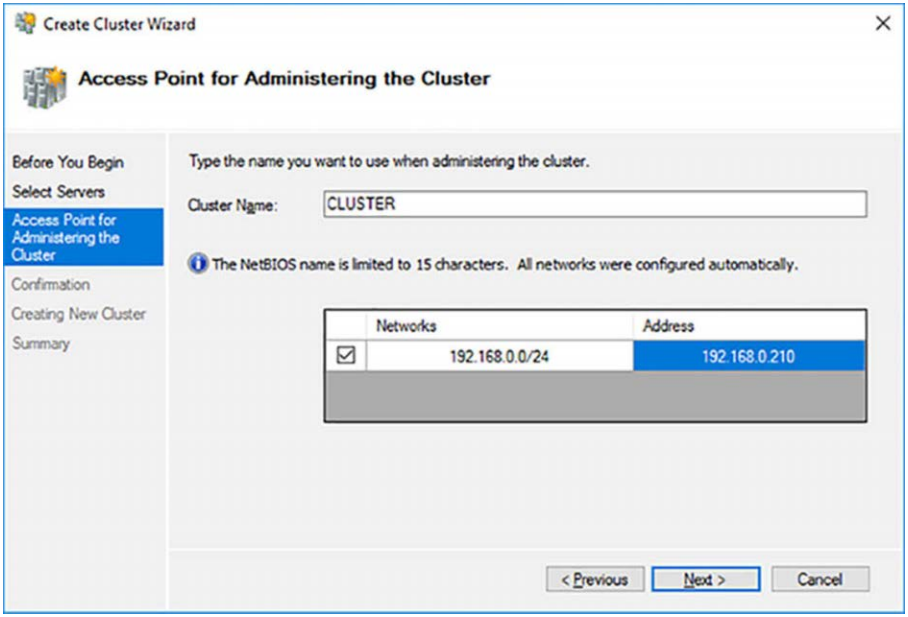

FIGURE 4-18 Availability Group Synchronous Commit

19. Uncheck the Add All Eligible Storage To The Cluster option, review and then confirm the creation of the failover cluster by clicking on the Next button.
20. Wait for the failover cluster to be created.
21. Review the failover cluster creation Summary page. Click on the View Report button to view the detailed failover cluster creation report.
22. Review and save the Create Cluster report looking out for any errors and warnings.
23. Close the Create Cluster report.
24. Click on the Finish button to close the Create Cluster Wizard

You can now leverage your failover cluster to create an Availability Group.

Create an Availability Group

To create an Availability the following prerequisites, need to have been met:

- A SQL Server instance must have been installed on all the servers that you plan to be part of an Availability Group.
- A failover cluster must have been created.
- Availability Groups must be enable for each SQL Server instance.

The following steps show how to enable Availability Groups for a SQL Server instance

1. Open up SQL Server Configuration Manager.
2. Right-click on the SQL Server instance and select Properties.

3. Select the Enable AlwaysOn Availability Group check box, as shown in Figure 4-19. Note the name of the failover cluster that you created earlier.

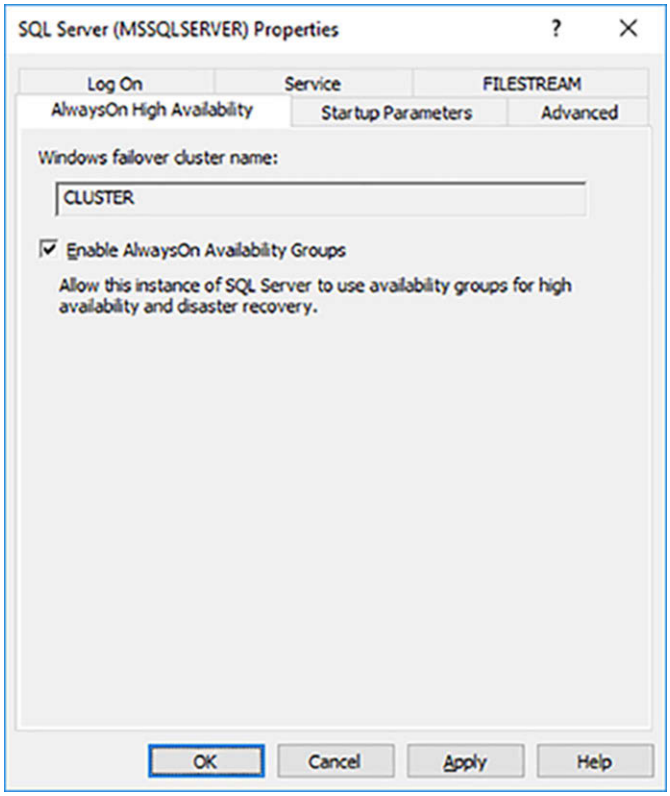

FIGURE 4-19 Enabling Availability Group at the SQL Server instance level

4. Click on the OK button to close the properties dialog box
5. Click on the OK button to close the warning. Note that SQL Server Configuration Manager does not automatically restart SQL Server whenever it is required for a change to take effect.
6. Right-click on the SQL Server instance and restart SQL Server.

You can now install an Availability Group within your SQL Server instances. To be able to add a database to an Availability Group the following pre-requisites must be met:

- The database must be using full recovery model
- A full database backup must have been performed, so that it's transaction log is not in auto-truncate mode.
- The database cannot be in read-only mode
- The database cannot be in single-user mode
- The database cannot be in auto-close mode

- The database cannot be part of an existing Availability Group. Databases can only belong to a single Availability Group at any point in time and space.

TIP **BACKING UP TO NUL DEVICE**

Typically, when setting up an Availability Group you will perform a backup of the database on the primary server and restore it on all secondary replicas. However, you still need to perform a full database backup before you get to that stage of the setup. If you need to perform a full database backup that you do not intend to keep you can perform the required full database backup to a nul device using the BACKUP DATABASE database_name TO DISK = 'nul' syntax. Backing up to a nul device performs a backup operation to nothing, so it is incredibly quick as there is no destination I/O.

The following steps show how to configure an Availability Group with 3 replicas.

1. Open SQL Server Management Studio.
2. Expand the AlwaysOn High Availability folder
3. Right-click on the Availability Groups folder and select New Availability Group Wizard to start the New Availability Group Wizard.
4. Click on the Next button on the Introduction page of the New Availability Group wizard.
5. Enter a name for your Availability Group, as shown in Figure 4-20, and click on the Next button.

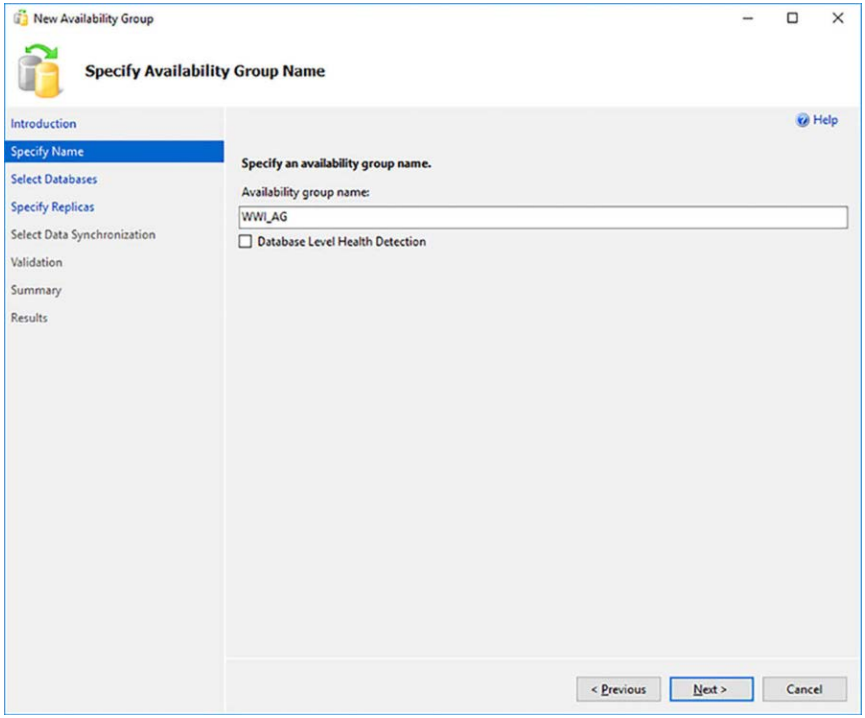

FIGURE 4-20 Availability Group Name

6. Select the Database Level Health Detection if you want the Availability Group to automatically failover if the Database Engine notices that any database within the Availability Group is no longer online. Although not fool-proof this new feature in SQL Server 2016 is worth enabling for Availability Groups that have multiple databases that represent a multi-database solution.
7. The Select Databases page allows you to select which databases will be part of the Availability Group. Only databases that meet the pre-requisites will be selectable. Select your database, as shown in Figure 4-21 and click on the Next button.

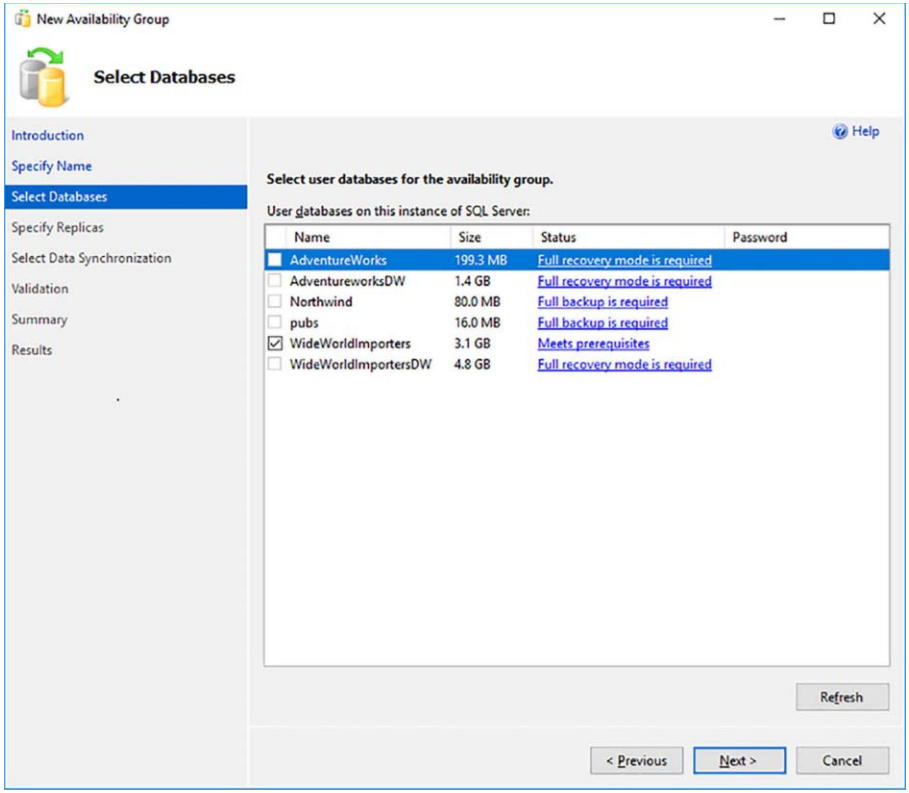

FIGURE 4-21 Availability Group Databases

8. The Specify Replicas page allows you to select up to 8 replicas for your Availability Group. You can select up to 3 replicas that will provide automatic failover, if required. You are further limited to only 3 synchronous replicas. Click on the Add Replica... button to add a secondary replica.
9. Connect to the replica by providing the server name and authentication details.
10. Repeat the above step for any other replicas that you wish to add to the Availability Group.
11. Check the Automatic Failover (Up to 3) check box for all your failover partner replicas, as shown in Figure 4-68. Notice how the replicas are automatically configured to use synchronous mode.
12. Select your readable secondaries, as shown in Figure 4-22. Readable secondaries have the following options:
    - **Yes** When in a secondary role, allow all connections from all applications to access this secondary in a readable fashion.

- **Read-only intent** When in a secondary role, only allow connections from "modern" applications that support the ApplicationIntent=ReadOnly or Application Intent=ReadOnly connection string parameter.

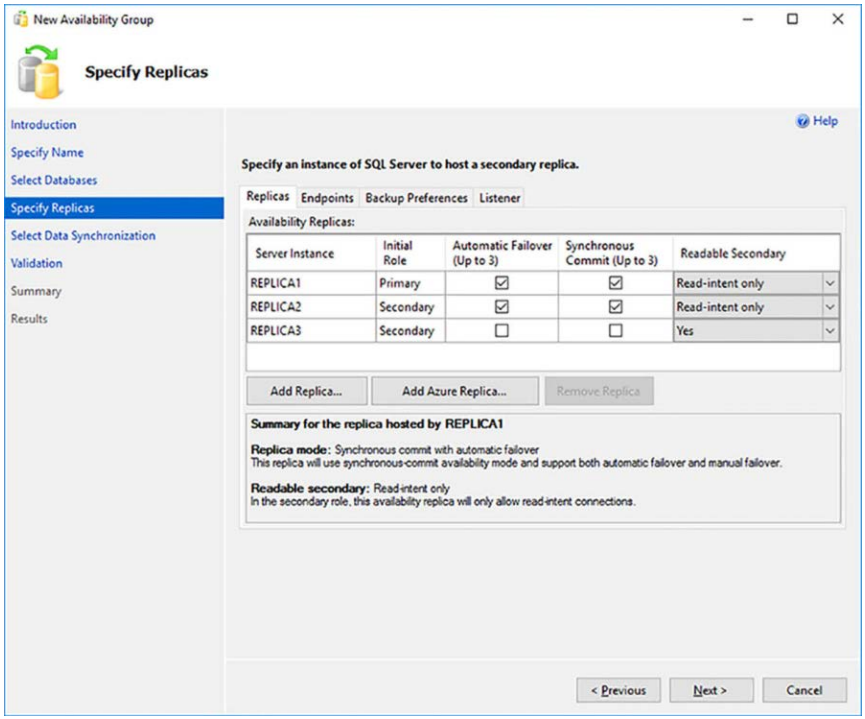

FIGURE 4-22 Availability Group readable secondaries

13. Click on the Next button.
14. Review the endpoint configuration for your replicas, as shown in Figure 4-23. Note that by default the endpoints will be encrypted. The default endpoints are fine in most cases. You will need to change the endpoints if your replicas are hosted on the same Window Operating System Environment (OSE). Click on the Next button when you are done.

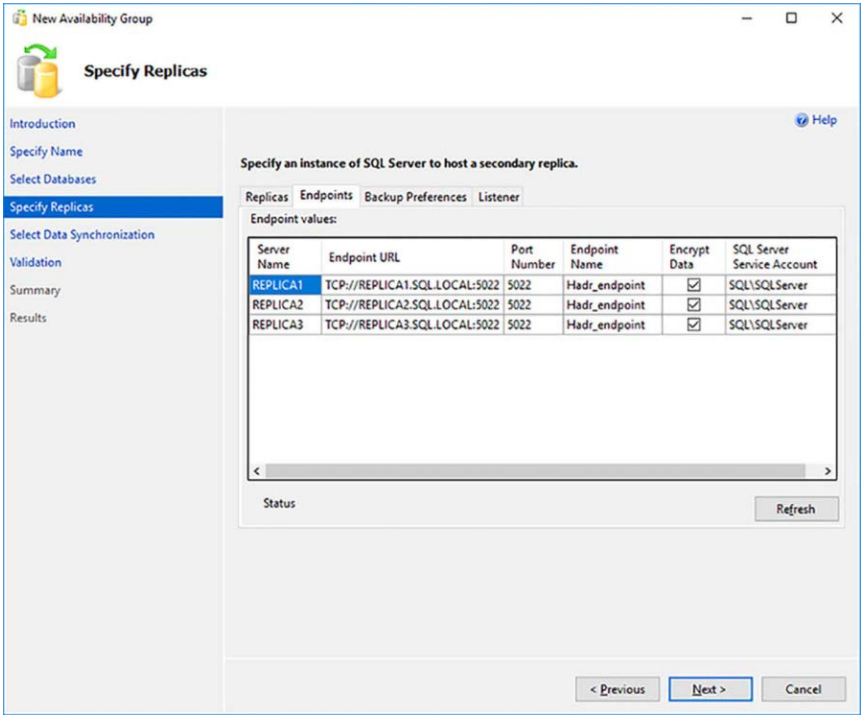

FIGURE 4-23 Availability Group endpoints

15. Define which replicas you want your backups to be performed on and their relative priority weight, as shown in Figure 4-24. If a number of replica can potentially perform the automated backup based on your preferences, the one with the highest priority will perform the backup. With Availability Groups backups can be performed on different replicas depending on where you want them performed. You backup preference choices are:
    - **Prefer Secondary** Automated backups will occur on a secondary replica. If no secondary replica is available, backups will be performed on the primary replica.
    - **Secondary Only** Automated backups for this Availability Group must occur on a secondary replica.
    - **Primary Only** Automated backups for this Availability Group must occur on a primary replica. Don't forget that non copy-only full database backups can only be performed on the primary replica.
    - **Any Replica** Automated backups for this Availability Group can occur on any replica.

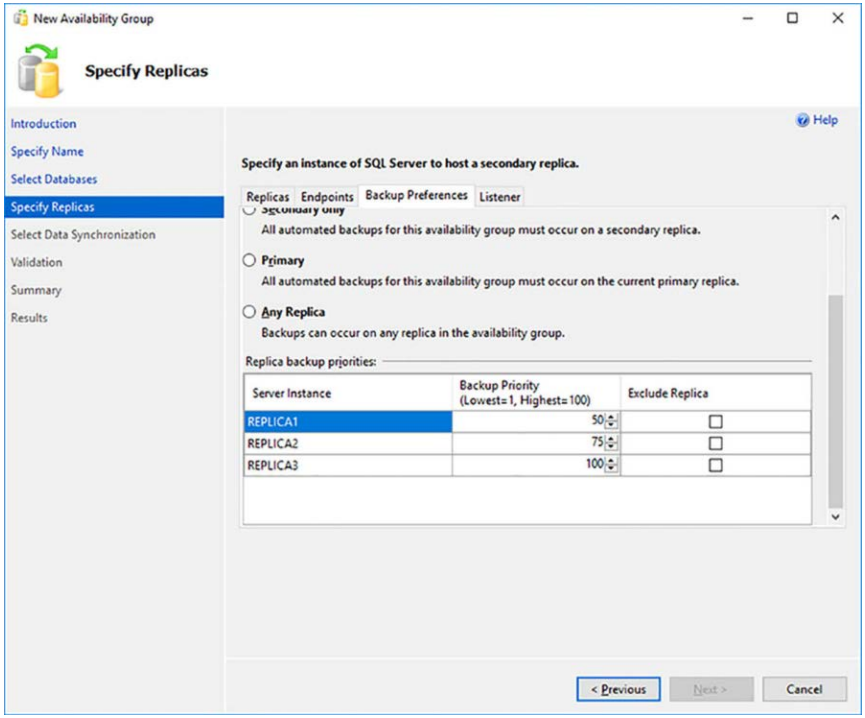

FIGURE 4-24 Availability Group backup preferences

16. Click on the Listener tab.
17. Configure the listener by providing the following information, as shown in Figure 4-25, and then click on the Next button:
    - DNS name
    - IP address
    - Port number

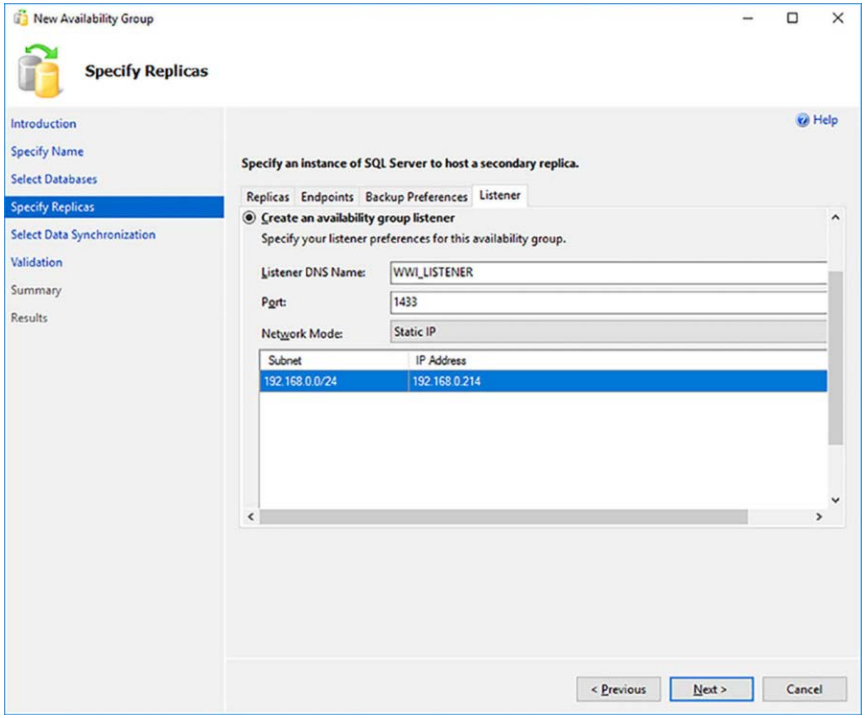

FIGURE 4-25 Availability Group listener configuration

18. The Create New Availability Group Wizard by default will synchronize the database from the primary replica to all of the secondary replicas through backup and restore operations. Provide the shared network location that will be used to store the database backups, as shown in Figure 4-26. Make sure that all replicas have access to this location.

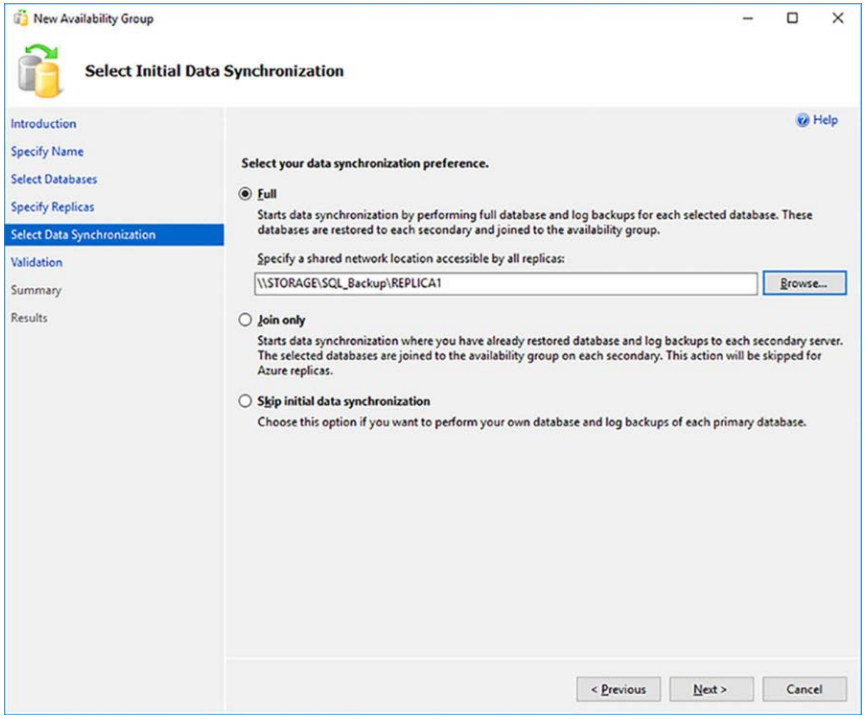

FIGURE 4-26 Availability Group initial synchronization options

NOTE **DIRECT SEEDING**

With the release of SQL Server 2016 you have the option of creating the replica of the primary database on the secondary replicas through the endpoints created by the Availability Group, instead of through backup and restore operations. This is done through the `SEEDING_MODE = AUTOMATIC` **option of the CREATE AVAILABILITY GROUP statement. Direct seeding will not be as efficient as the backup and restore operations, and is designed for specific use cases where the backup/restore process will not work.**

19. Click on the Next button when the Create New Availability Group Wizard finishes validating the Availability Group creation.
20. Review the Availability Group creation summary to make sure that all of the configuration details are correct.
21. Click on the Script drop down list and save the Availability Group creation script for review and change management reasons.
22. Click on the Finish button to start the Availability Group creation.
23. This will take some time.
24. Confirm that the Availability Group was successfully created.

Listing 4-3 shows the Transact-SQL script that was generated to configure the Log Shipping solution.

LISTING 4-3 Availability group configuration

```
--- YOU MUST EXECUTE THE FOLLOWING SCRIPT IN SQLCMD MODE.
:Connect REPLICA1
USE [master]
GO

CREATE ENDPOINT [Hadr_endpoint]
    AS TCP (LISTENER_PORT = 5022)
    FOR DATA_MIRRORING (ROLE = ALL, ENCRYPTION = REQUIRED ALGORITHM AES)
GO

IF (SELECT state FROM sys.endpoints WHERE name = N'Hadr_endpoint') <> 0
BEGIN
    ALTER ENDPOINT [Hadr_endpoint] STATE = STARTED
END
GO

use [master]
GO

GRANT CONNECT ON ENDPOINT::[Hadr_endpoint] TO [SQL\SQLServer]
GO

:Connect REPLICA1
IF EXISTS(SELECT * FROM sys.server_event_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER WITH (STARTUP_STATE=ON);
END
IF NOT EXISTS(SELECT * FROM sys.dm_xe_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER STATE=START;
END
GO

:Connect REPLICA2
USE [master]
GO

CREATE ENDPOINT [Hadr_endpoint]
    AS TCP (LISTENER_PORT = 5022)
    FOR DATA_MIRRORING (ROLE = ALL, ENCRYPTION = REQUIRED ALGORITHM AES)
GO

IF (SELECT state FROM sys.endpoints WHERE name = N'Hadr_endpoint') <> 0
BEGIN
    ALTER ENDPOINT [Hadr_endpoint] STATE = STARTED
END
GO

use [master]
```

```
GO

GRANT CONNECT ON ENDPOINT::[Hadr_endpoint] TO [SQL\SQLServer]
GO

:Connect REPLICA2
IF EXISTS(SELECT * FROM sys.server_event_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER WITH (STARTUP_STATE=ON);
END
IF NOT EXISTS(SELECT * FROM sys.dm_xe_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER STATE=START;
END
GO

:Connect REPLICA3
USE [master]
GO

CREATE ENDPOINT [Hadr_endpoint]
    AS TCP (LISTENER_PORT = 5022)
    FOR DATA_MIRRORING (ROLE = ALL, ENCRYPTION = REQUIRED ALGORITHM AES)
GO

IF (SELECT state FROM sys.endpoints WHERE name = N'Hadr_endpoint') <> 0
BEGIN
    ALTER ENDPOINT [Hadr_endpoint] STATE = STARTED
END
GO

use [master]
GO

GRANT CONNECT ON ENDPOINT::[Hadr_endpoint] TO [SQL\SQLServer]
GO

:Connect REPLICA3

IF EXISTS(SELECT * FROM sys.server_event_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER WITH (STARTUP_STATE=ON);
END
IF NOT EXISTS(SELECT * FROM sys.dm_xe_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER STATE=START;
END
GO

:Connect REPLICA1
USE [master]
GO

CREATE AVAILABILITY GROUP [WWI_AG]
WITH (AUTOMATED_BACKUP_PREFERENCE = SECONDARY,
```

```
DB_FAILOVER = OFF,
DTC_SUPPORT = NONE)
FOR DATABASE [WideWorldImporters]
REPLICA ON N'REPLICA1' WITH (ENDPOINT_URL = N'TCP://REPLICA1.SQL.LOCAL:5022', FAILOVER_
MODE = AUTOMATIC, AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, BACKUP_PRIORITY = 50,
SECONDARY_ROLE(ALLOW_CONNECTIONS = READ_ONLY)),
    N'REPLICA2' WITH (ENDPOINT_URL = N'TCP://REPLICA2.SQL.LOCAL:5022', FAILOVER_MODE
= AUTOMATIC, AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, BACKUP_PRIORITY = 75, SECONDARY_
ROLE(ALLOW_CONNECTIONS = READ_ONLY)),
    N'REPLICA3' WITH (ENDPOINT_URL = N'TCP://REPLICA3.SQL.LOCAL:5022', FAILOVER_MODE
= MANUAL, AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT, BACKUP_PRIORITY = 100, SECONDARY_
ROLE(ALLOW_CONNECTIONS = ALL));
GO

:Connect REPLICA1
USE [master]
GO

ALTER AVAILABILITY GROUP [WWI_AG]
ADD LISTENER N'WWI_LISTENER' (
WITH IP
((N'192.168.0.214', N'255.255.255.0')
)
, PORT=1433);
GO

:Connect REPLICA2
ALTER AVAILABILITY GROUP [WWI_AG] JOIN;
GO

:Connect REPLICA3
ALTER AVAILABILITY GROUP [WWI_AG] JOIN;
GO

:Connect REPLICA1
BACKUP DATABASE [WideWorldImporters] TO  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters.bak' WITH  COPY_ONLY, FORMAT, INIT, SKIP, REWIND, NOUNLOAD,
COMPRESSION,  STATS = 5
GO

:Connect REPLICA2
RESTORE DATABASE [WideWorldImporters] FROM  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters.bak' WITH  NORECOVERY,  NOUNLOAD,  STATS = 5
GO

:Connect REPLICA3
RESTORE DATABASE [WideWorldImporters] FROM  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters.bak' WITH  NORECOVERY,  NOUNLOAD,  STATS = 5
GO

:Connect REPLICA1
BACKUP LOG [WideWorldImporters] TO  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters_20170310165240.trn' WITH NOFORMAT, NOINIT, NOSKIP, REWIND, NOUNLOAD,
COMPRESSION,  STATS = 5
```

```
GO

:Connect REPLICA2
RESTORE LOG [WideWorldImporters] FROM  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters_20170310165240.trn' WITH  NORECOVERY,  NOUNLOAD,  STATS = 5
GO

:Connect REPLICA2
-- Wait for the replica to start communicating
begin try
declare @conn bit
declare @count int
declare @replica_id uniqueidentifier
declare @group_id uniqueidentifier
set @conn = 0
set @count = 30 -- wait for 5 minutes

if (serverproperty('IsHadrEnabled') = 1)
    and (isnull((select member_state from master.sys.dm_hadr_cluster_members where
upper(member_name COLLATE Latin1_General_CI_AS) = upper(cast(serverproperty('ComputerNam
ePhysicalNetBIOS') as nvarchar(256)) COLLATE Latin1_General_CI_AS)), 0) <> 0)
    and (isnull((select state from master.sys.database_mirroring_endpoints), 1) = 0)
begin
    select @group_id = ags.group_id from master.sys.availability_groups as ags where
name = N'WWI_AG'
    select @replica_id = replicas.replica_id from master.sys.availability_replicas
as replicas where upper(replicas.replica_server_name COLLATE Latin1_General_CI_AS) =
upper(@@SERVERNAME COLLATE Latin1_General_CI_AS) and group_id = @group_id
    while @conn <> 1 and @count > 0
    begin
        set @conn = isnull((select connected_state from master.sys.dm_hadr_availability_
replica_states as states where states.replica_id = @replica_id), 1)
        if @conn = 1
        begin
            -- exit loop when the replica is connected, or if the query cannot find the
replica status
            break
        end
        waitfor delay '00:00:10'
        set @count = @count - 1
    end
end
end try
begin catch
    -- If the wait loop fails, do not stop execution of the alter database statement
end catch
ALTER DATABASE [WideWorldImporters] SET HADR AVAILABILITY GROUP = [WWI_AG];
GO

:Connect REPLICA3
RESTORE LOG [WideWorldImporters] FROM  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters_20170310165240.trn' WITH  NORECOVERY,  NOUNLOAD,  STATS = 5
GO

:Connect REPLICA3
```

```
-- Wait for the replica to start communicating
begin try
declare @conn bit
declare @count int
declare @replica_id uniqueidentifier
declare @group_id uniqueidentifier
set @conn = 0
set @count = 30 -- wait for 5 minutes

if (serverproperty('IsHadrEnabled') = 1)
    and (isnull((select member_state from master.sys.dm_hadr_cluster_members where
upper(member_name COLLATE Latin1_General_CI_AS) = upper(cast(serverproperty('ComputerNam
ePhysicalNetBIOS') as nvarchar(256)) COLLATE Latin1_General_CI_AS)), 0) <> 0)
    and (isnull((select state from master.sys.database_mirroring_endpoints), 1) = 0)
begin
    select @group_id = ags.group_id from master.sys.availability_groups as ags where
name = N'WWI_AG'
    select @replica_id = replicas.replica_id from master.sys.availability_replicas
as replicas where upper(replicas.replica_server_name COLLATE Latin1_General_CI_AS) =
upper(@@SERVERNAME COLLATE Latin1_General_CI_AS) and group_id = @group_id
    while @conn <> 1 and @count > 0
    begin
        set @conn = isnull((select connected_state from master.sys.dm_hadr_availability_
replica_states as states where states.replica_id = @replica_id), 1)
        if @conn = 1
        begin
            -- exit loop when the replica is connected, or if the query cannot find the
replica status
            break
        end
        waitfor delay '00:00:10'
        set @count = @count - 1
    end
end
end try
begin catch
    -- If the wait loop fails, do not stop execution of the alter database statement
end catch
ALTER DATABASE [WideWorldImporters] SET HADR AVAILABILITY GROUP = [WWI_AG];
GO
```

EXAM TIP

Make sure you familiarize yourself with the key statements in the Availability Group creation script for the exam.

Configure Quorum Configuration for Availability Group

One of the more important aspects of configuring an Availability Group correctly is to configure the correct quorum configuration. The default Availability Group installation might not configure the optimal quorum configuration. The quorum configuration and forming quorum is the responsible of the WSFC. Consequently, you need to control that at the WSFC level.

Quorum and quorum configuration will be discussed later in this chapter when it covers failover clustering.

Let's assume that the Availability Group with three replicas that we have configured above have the following topology:

- REPLICA1 and REPLICA2 are in one data center
- REPLICA3 is in a separate data center

Let's also assume that REPLICA3 is no longer a failover partner.

In this scenario, you do not want REPLICA3 to participate in the quorum, as it is in a separate data center. There will be a greater latency between it and the other 2 replicas. Likewise, if the network link fails between the data centers it will not be able to communicate with the other replicas. In the worst-case scenario, your entire failover cluster could shut down to protect itself. You do not want REPLICA3 to have a vote.

You are better off by creating an additional witness in the data center where REPLICA1 and REPLICA2 are located.

The following steps show how to change the quorum configuration for your Availability Group:

1. Open SQL Server Management Studio.
2. Connect to your primary replica.
3. Expand the AlwaysOn High Availability folder.
4. Right-click on your Availability Group and select Show Dashboard.
5. Click on the View Cluster Quorum Information link in the top right hand corner of the Availability Group dashboard.
6. Determine the current quorum configuration, as shown in Figure 4-27 and click on the Close button. Initially the Availability Group is using a Node Majority quorum model and replicas REPLICA1, REPLICA2, REPLICA3 all have a vote. You need to change this so that REPLICA3 does not have a vote in the quorum.

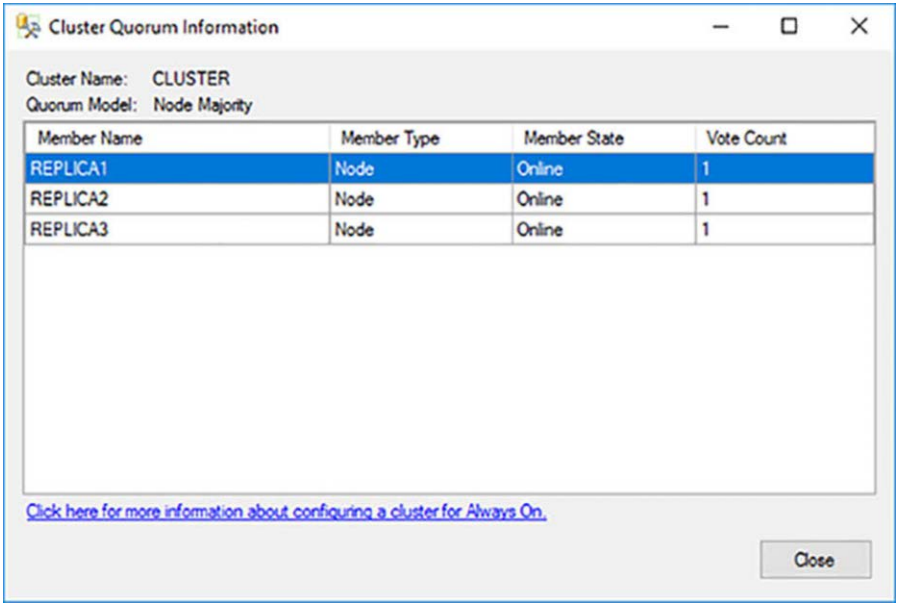

FIGURE 4-27 Initial Availability Group quorum configuration

7. Close the Cluster Quorum Information dialog box.
8. Open Failover Cluster Manager.
9. Connect to the cluster that is being used by the Availability Group.
10. Right click on your cluster, select the More Actions option and then the Configure Cluster Quorum Settings option to start the Configure Cluster Quorum Wizard.
11. Read the welcome page of the Configure Cluster Quorum Wizard and click on the Next button.
12. On the Select Quorum Configuration Option page select the Advanced Quorum configuration option and click on the Next button.
13. On the Select Voting Configuration page select the Select Nodes option, then uncheck REPLICA3 as a voting node, before clicking on the Next button, as shown in Figure 4-28.

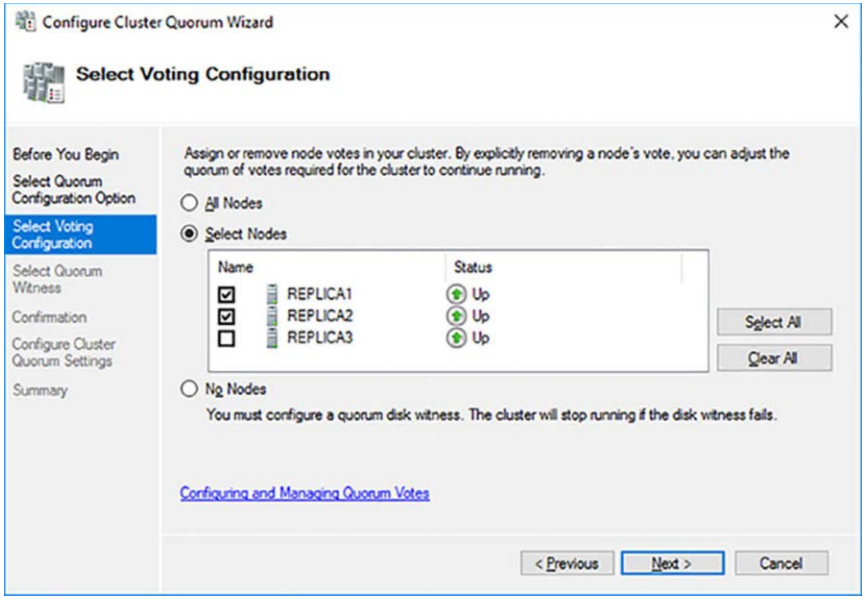

FIGURE 4-28 Select Voting Configuration

14. On the Select Quorum Witness page select the Configure A File Share Witness option and click on the Next button. In this case, you do not have an odd number of replicas in the same data center. Consequently, you need to add a witness to avoid the "split brain" problem, where a cluster cannot form quorum and effectively shuts down.
15. Click on the Browse button to create a file share witness.
16. In the Browse For Shared Folders dialog box type in the name of your file share server and click on the Show Shared Folders button to connect to the file share server and display its shared folders.
17. There are no suitable folders, so click on the New Shared folder button to create a new file share with the appropriate permissions.
18. Configure the following properties and click on the OK button:
    - Share name
    - Local path of shared folder
    - Shared folder permissions
19. Confirm that the file share path is correct and click on the Next button.
20. Review the new quorum settings are correct before clicking on the Next button.
21. Ensure that the new quorum settings have been configured correctly before clicking on the Finish button.
22. Switch back to SQL Server Management Studio and the Availability Group dashboard

23. Click on the View Cluster Quorum Information link in the top right hand corner of the Availability Group dashboard again to show the new quorum model, as shown in Figure 4-29.

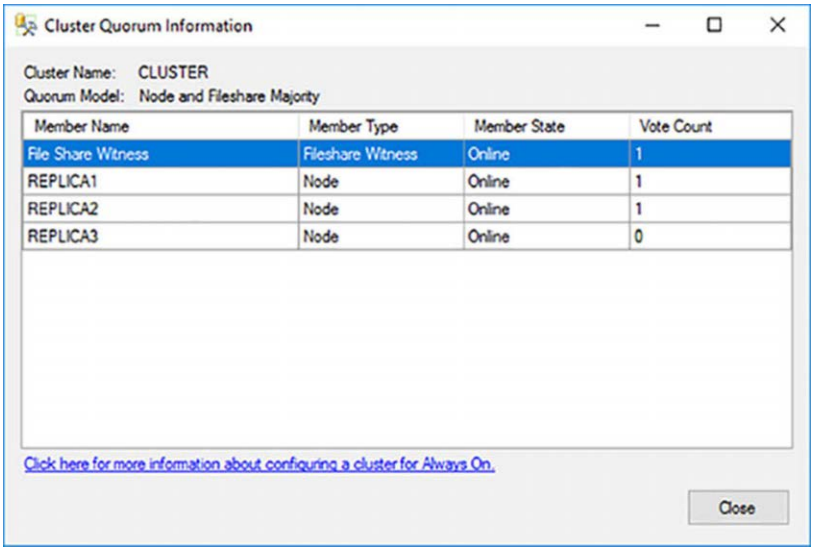

FIGURE 4-29 New Availability Group quorum configuration

24. Confirm that the new quorum model is a Node and Fileshare majority.
25. Confirm that REPLICA3 no longer has a vote.

EXAM TIP

Make sure you understand the different quorum models and which models to use for Availability Groups versus failover clusters for the exam.

Configure read-only routing

One of the major benefits of Availability Groups is their ability to scale out read operations or reporting to readable secondaries. Using read-only routing Availability Groups provides the capability of routing connection requests from applications automatically to a readable secondary.

The following conditions must be true for read-only routing to work:

- The application must connect to the listener and not to the replica directly.
- The application must connect with an explicit read-only request in the connection string.
- A readable secondary replica must exist in the Availability Group.
- Read-only routing rules have been defined by the database administrator.

To define the read-only routing rules you need to configure the following:

- **Read-only Routing URL** The read-only routing URL is used for routing read-intent connection requests to a specific readable secondary replica. It needs to be specified on each replica that will potentially be running as a readable secondary. It takes effect only when the replica is running in the secondary role.
- **Read-only Routing List** The read-only routing list. It dictates the order in which your read-only connection request will be routed. It takes effect only when a replica is running in the primary role.

Listing 4-4 shows you how to set up the read-only routing URLs

LISTING 4-4 Read-only routing URL

```
-- Execute the following statements at the Primary to configure Log Shipping
ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA1' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://REPLICA1.SQL.LOCAL:1433'));
GO

ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA2' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://REPLICA2.SQL.LOCAL:1433'));
GO

ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA3' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://REPLICA3.SQL.LOCAL:1433'));
GO
```

Listing 4-5 shows you how to set up the read-only routing list.

LISTING 4-5 Read-only routing list

```
-- Execute the following statements at the Primary to configure Log Shipping
ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA1' WITH
(PRIMARY_ROLE (READ_ONLY_ROUTING_LIST=(N'REPLICA2',N'REPLICA3')));
GO

ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA2' WITH
(PRIMARY_ROLE (READ_ONLY_ROUTING_LIST=(N'REPLICA1',N'REPLICA3')));
GO

ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA3' WITH
(PRIMARY_ROLE (READ_ONLY_ROUTING_LIST=(N'REPLICA1',N'REPLICA2')));
```

The following steps show you how to test whether read-only routing works:

1. Open SQL Server Management Studio.
2. Click on the Connect option in Object Explorer and choose Database Engine.
3. In the Connect To Server dialog box provide the listener name in the Server Name drop down list and click on the Options button.
4. Click on the Additional Connection Properties tab.
5. Provide the name of the Availability Group database and the read-only intention connection string parameters, as shown in Figure 4-30.

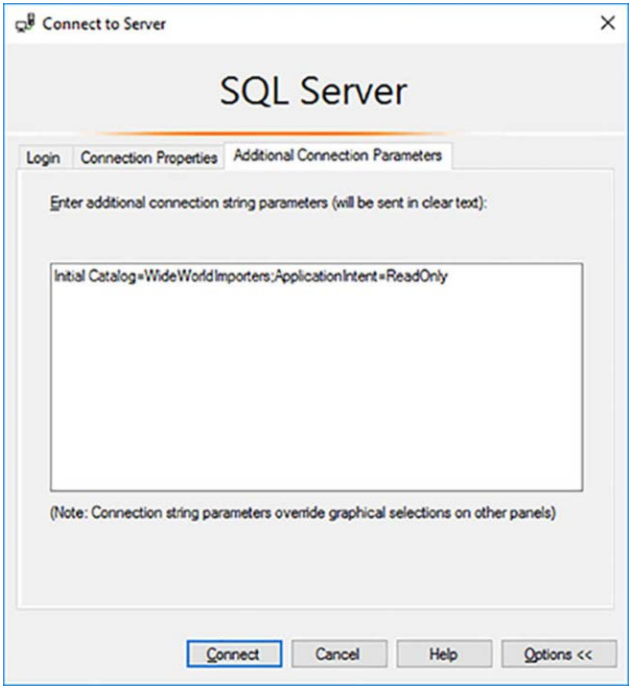

FIGURE 4-30 Read-only intention to connect to listener

IMPORTANT **INITIAL CATALOG**

It is important to specify an Availability Group database in the Initial Catalog connection string setting for read-only routing to work. Otherwise you will connect to your default database in the primary replica. Unless of course it also happens to be a database in the Availability Group that you are intending to connect to, in which case it will work.

6. Click on the listener in Object Explorer.
7. Click on New Query in the tool bar to connect to your listener.

8. Execute the SELECT @@SERVERNAME query. The server name returned should be a secondary replica's and not the primary replica's.
9. Attempt to update a table in the database. You should get an error informing you that you cannot perform this DML operation in a read-only database.

SQL Server 2016 introduced the ability to load balance the read-only replicas. Load balancing uses a round-robin algorithm. To load balance between a set of read-only replica simply enclose the set of read-only replicas with parentheses in the read-only routing list option, as shown in Listing 4-6.

LISTING 4-6 Configure load-balancing across read-only replicas

```
READ_ONLY_ROUTING_LIST = (('REPLICA1','REPLICA2','REPLICA3'), 'REPLICA4', 'REPLICA5')
```

In this example the read-only connection requests will be load balanced between REPLICA1, REPLICA2 and REPLICA3. If none of these replicas are available, REPLICA4 will be used. If it fails, REPLICA5 will be used.

Monitor Availability Groups

Availability Groups support a dashboard that you can use to see the state of your Availability Groups and perform certain tasks, such performing a failover. It shows important performance indicators that will help you to make better operational decisions. Some of the key metrics that it shows includes:

- Synchronization mode and state
- Estimate Data Loss
- Estimated Recovery Time
- Time to restore log

View the Availability Group dashboard by right-clicking on your Availability Group and choosing Show Dashboard.

Figure 4-31 shows the state of the Availability Group from the primary replica. You can view at key metrics such as the send and redo queues, and how long it will take for any replica that is not synchronized to catch up.

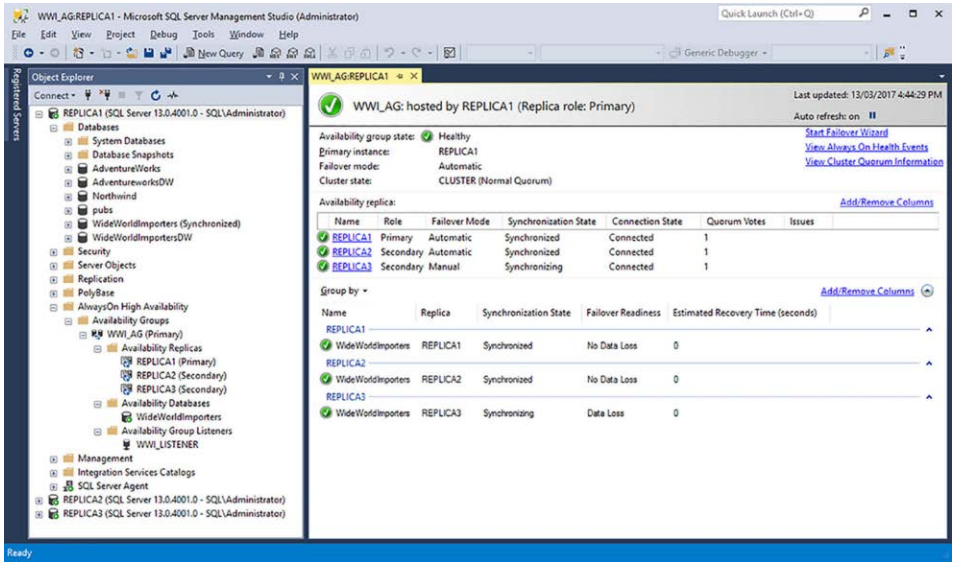

FIGURE 4-31 Availability Group dashboard at the primary replica

You can add the following metrics to the dashboard by right clicking on the column headings and selecting them:

- Issues
- Availability Mode
- Primary Connection Mode
- Secondary Connection Mode
- Connection State
- Operational State
- Last Connect Error No.
- Last Connection Error Description
- Last Connection Error Timestamp
- Quorum Votes
- Member State

You can click on the synchronous secondary replica to see its state within the Availability Group. It will not know about the state of the other replicas. It will be synchronized and ready to fall over. No data loss is possible. In the case of the asynchronous secondary replica it will indicate that data loss is possible. This is always the case with asynchronous replicas.

Manage failover

A failover is a process where the primary replica gives up its role to a failover partner. With Availability Groups the failover is at the Availability Group level. During a "normal" failover no data loss will occur. However, any transactions in flight will be lost and have to be rolled back.

During the failover process, the failover target needs to recover its instance of the databases and bring them online as the new primary databases. This process in certain cases can take a long time.

There are three types of failover:

- **Automatic failover** Automatic failover will occur when the WSFC detects that something has failed or the health of either the Availability Group or database has deteriorated sufficiently, based on the Availability Groups configuration. No data loss is possible.
- **Manual failover** Manual failover occurs when you explicitly perform a failover because you need perform some administrative task, such as patching the Windows operating system or SQL Server. You also fail over an Availability Group if you want it to run on another server's hardware resources. With manual failover no data loss is possible.
- **Forced failover** The RPO defines the maximum acceptable amount of data loss following a disaster incident. With forced failover data loss is possible.

Table 4-3 shows the failover types supported, depending on what synchronization mode the replica is using.

TABLE 4-3 Availability Group Failover options

Failover	Asynchronous MOde	Synchronous Mode	Synchronous Mode with automatic failvover
Automatic Failover	No	No	Yes
Manual Failover	No	Yes	Yes
Forced Failover	Yes	Yes	Yes (same as manual failover)

The following steps show you how to perform a manual failover.

1. Open SQL Server Management Studio.
2. Connect to your primary replica.
3. Expand the AlwaysOn High Availability folder.
4. Right-click on your Availability Group and select Failover to start the Fail Over Availability Group Wizard.
5. Click on the Next button on the Introduction page.
6. Review all of the information in the Select New Primary Replica page to ensure that you are not going to lose data due to failover. Read the warnings. Select the new primary replica, as shown in Figure 4-32, and click on the Next button.

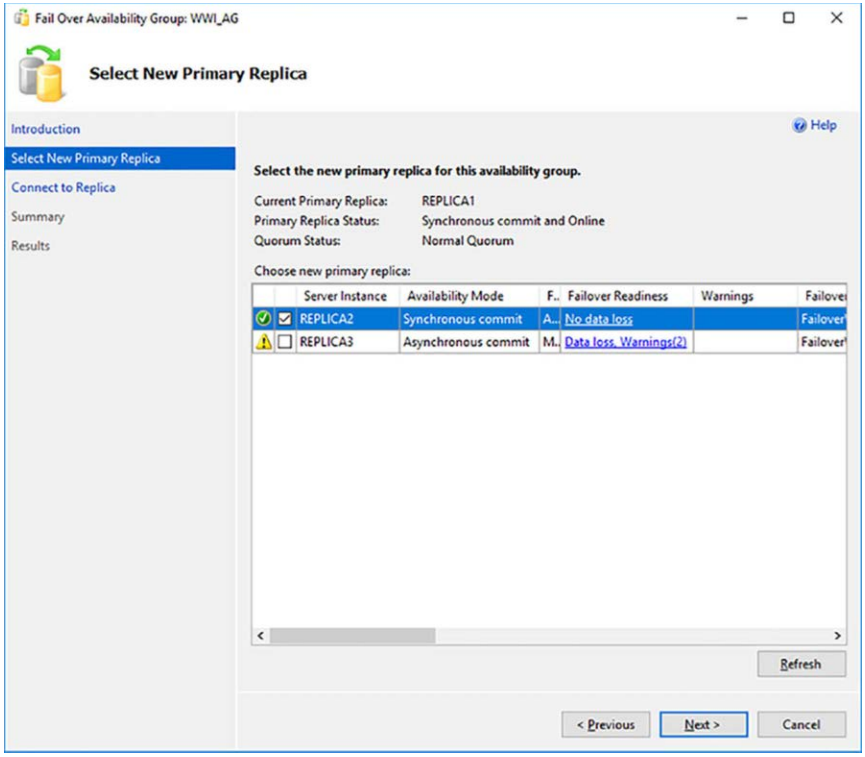

FIGURE 4-32 Specify to failover target

7. Connect to failover target replica and click on the Next button.
8. Review the choices made in the Summary page and click on the Finish button to initiate the fail over.
9. Confirm that the failover has been successful and click on the Close button.

Listing 4-7 shows you how to perform an equivalent failover in Transact-SQL. Note that it has to be performed from the failover target replica, not the primary replica.

LISTING 4-7 Manual fail over with no data loss

```
--- YOU MUST EXECUTE THE FOLLOWING SCRIPT IN SQLCMD MODE.

:Connect REPLICA2

ALTER AVAILABILITY GROUP [WWI_AG] FAILOVER;

GO
```

The following steps show you how to perform a forced failover.

1. Open SQL Server Management Studio.
2. Connect to your primary replica.

3. Expand the AlwaysOn High Availability folder.
4. Right-click on your Availability Group and select Failover to start the Fail Over Availability Group Wizard.
5. Click on the Next button on the Introduction page.
6. This time, in the Select New Primary Replica page, select the asynchronous commit replica as a failover target. The wizard shows that the asynchronous secondary replica is using asynchronous commit and that only fail over with data loss is supported. Furthermore, there are three warnings.
7. Click on the warning link and read the 3 warnings, shown in Figure 4-33.

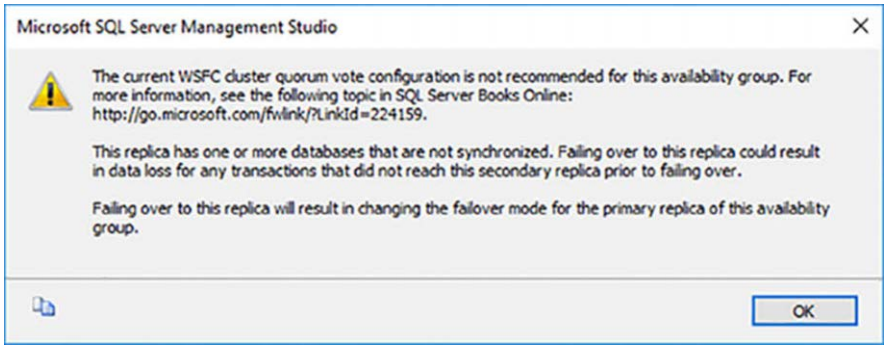

FIGURE 4-33 Fail over warnings

8. Click on the Close button to close the warning dialog box.
9. Click on the Next button in the Select New Primary Replica screen.
10. The next screen in the wizard again warns you about the potential data loss. Select the Click Here To Confirm Failover With Potential Data Loss check box and click on the Next button, as shown in Figure 4-34.

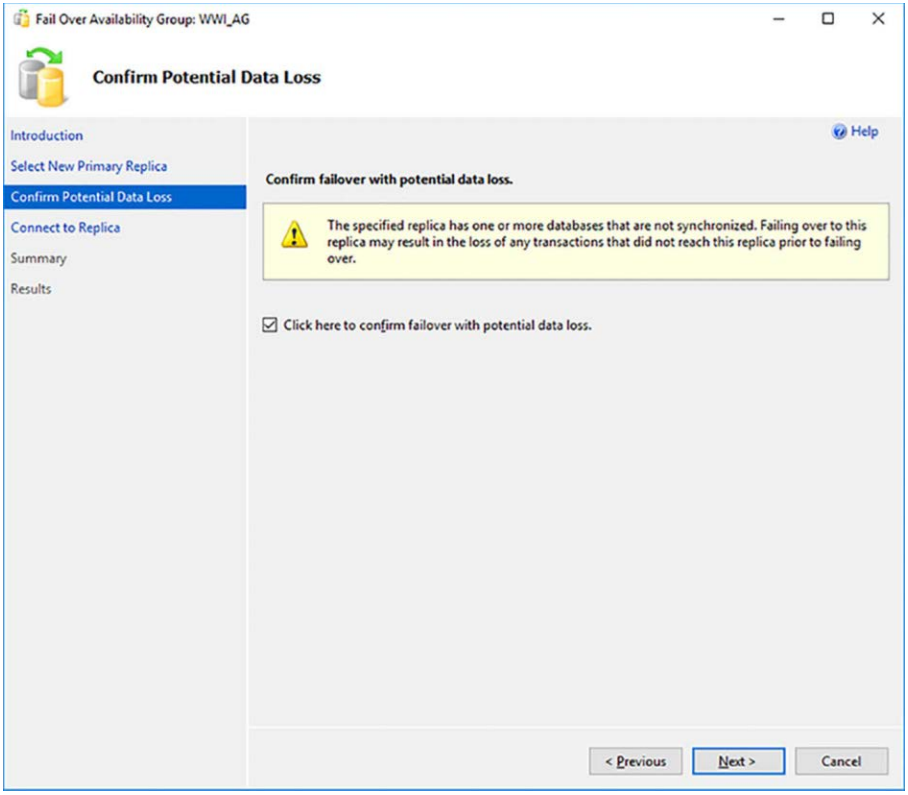

FIGURE 4-34 Potential data loss failover warnings

11. Connect to the asynchronous target in the Connect To Replica screen and click on the Next button.
12. Review the choices made and generate the failover script before clicking on the Finish button to initiate the fail over.
13. Confirm that the failover has been successful and click on the Close button.
14. Confirm on the Action Require link and read the warning, which is identical to the first error in in Figure 4-34 before closing the wizard.

Listing 4-8 shows you how to perform an equivalent forced failover in Transact-SQL.

LISTING 4-8 Forced failover with potential data loss

```
--- YOU MUST EXECUTE THE FOLLOWING SCRIPT IN SQLCMD MODE.
:Connect REPLICA3

ALTER AVAILABILITY GROUP [WWI_AG] FORCE_FAILOVER_ALLOW_DATA_LOSS;
GO
```

Create Distributed Availability Group

Distributed Availability Groups (DAGs) were added in SQL Server 2016 for a number of specific use cases. To best understand where you can use Distributed Availability Groups, it is best to start off with a diagram of what they look like.

Figure 4-35 show the architecture of a Distributed Availability Group. The DAG has the following characteristics:

- The operating system environment (OSE) for the primary WSFC (WSFC1) can be different from the secondary WSFC (WSFC2).
- The health of primary WSFC (WSFC1) is not affected by the health of the secondary WSFC (WSFC2).
  - Each WSFC is responsible for maintaining its own quorum mode.
  - Each WSFC is responsible for its own node voting configuration.
- The data is sent only once between the primary Availability Group (AG1) and the secondary Availability Group (AG2).
  - This is one of the primary benefits of DAGs, especially across WAN links, since otherwise the primary replica in AG1 would have to send the same log records across the network to the three replicas in the secondary Availability Group (AG2) .
- All of the replicas in the secondary Availability Group (AG2) are read-only.
- Automatic failover to the secondary Availability Group (AG2) is not supported.

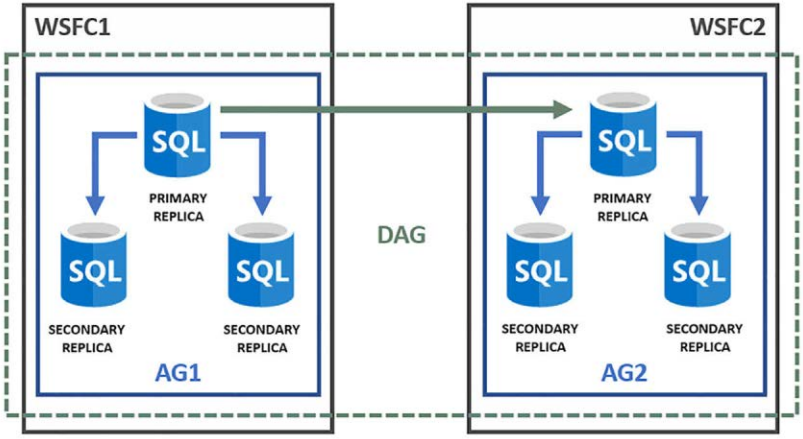

FIGURE 4-35 Distributed Availability Group

To create a distributed Availability Group, perform the following steps:

1. Create an Availability Group for each WSFC.
2. Create a listener for each Availability Group.

3. Create the DAG on the primary Availability Group using the DISTRIBUTED option as shown in Listing 4-9. Note, we will use direct seeding in this example.

LISTING 4-9 Creating an distributed Availability Group on the primary Availability Group

```
CREATE AVAILABILITY GROUP [DAG]
    WITH (DISTRIBUTED)
  AVAILABILITY GROUP ON
    'AG1' WITH (
      LISTENER_URL = 'TCP://AG1_LISTENER:5022',
             AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
      FAILOVER_MODE = MANUAL,
      SEEDING_MODE = AUTOMATIC
    ),
    'AG2' WITH (
      LISTENER_URL = 'TCP://AG2-LISTENER:5022',
      AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
      FAILOVER_MODE = MANUAL,
      SEEDING_MODE = AUTOMATIC
    );
GO
```

4. Join the DAG from the secondary Availability Group, as shown in Listing 4-10.

LISTING 4-10 Joining a distributed Availability Group from the secondary Availability Group

```
ALTER AVAILABILITY GROUP [distributedag]
  JOIN
  AVAILABILITY GROUP ON
    'AG1' WITH (
      LISTENER_URL = 'tcp://ag1-listener:5022',
      AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
      FAILOVER_MODE = MANUAL,
      SEEDING_MODE = AUTOMATIC
    ),
     'AG2' WITH (
      LISTENER_URL = 'tcp://ag2-listener:5022',
      AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
       FAILOVER_MODE = MANUAL,
      SEEDING_MODE = AUTOMATIC
    );
GO
```

Skill 4.5: Implement failover clustering

Failover clustering has been available since SQL Server 6.5, so it is a very stable and manure high availability technology. A SQL Server Failover Cluster Instance (FCI) relies on Windows Clustering, so SQL Server is effectively a cluster aware application stack. However, not all components of SQL Server are cluster aware. For example, Reporting Services cannot be installed as FCI. Always try to deploy SQL Server FCIs on the latest version of the Windows Server operating

system. Microsoft is always improving failover clustering in every Windows release, making it easier to configure and manage, perform better, and more reliable.

This objective covers how to:

- Architect Availability Groups
- Configure Windows clustering
- Create an Availability Group
- Configure read-only routing
- Manage failover
- Create distributed Availability Group

Architect failover clustering

Windows Server 2012 really saw failover clustering come of age, with improvements across the board and in particular with the release of Server Message Block (SMB) 3.0. The benefit that SMB 3.0 brings is that it gives you the ability of locating your database files on shares. SQL Server 2014 introduced the capability of running databases off shares. This capability is commonly overlooked by the industry due to a lack of education and awareness. Failover clustering no longer relies solely on Fiber Channel (FC) or iSCSI protocols.

Compared to Availability Groups, Failover Clustering is a lot easier to implement and administer. This is fundamentally due to the fact that there is only a single set of database files that are hosted by a SQL Server instance and made available to users. Microsoft will continue to invest in failover clustering, so it is a perfectly valid high availability technology that you should assess and use as appropriate.

When designing a failover cluster solution, you should aim to provide redundancy at each level including server hardware, networking hardware, and network infrastructure. Don't forget to leverage the capabilities in the Windows Server operating system, such as NIC teaming and SMB Multichannel.

The failover clustering architecture, shown in Figure 4-36, contains the following elements:

- **Node** A node is a Windows Server instance that is participating in a failover cluster. Failover cluster instances of applications, such as SQL Server, can run on any single node at any given point in time and space. Windows Server Standard Edition only supports 2 node failover clusters. Window Server Datacenter Edition support failover clusters with up to 64 nodes.
- **Shared Storage** Shared Storage is a single instance of a storage subsystem that is accessible by all nodes of the failover cluster. Traditionally, the Shared Storage was located on a SAN that was accessed via Fiber Channel (FC). Since then iSCSI has proved to be more popular as technology has evolved. With the release of Windows Server 2012 and SMB 3.0, you can use SMB shares instead.

NOTE **SINGLE POINT OF FAILURE**

The shared storage represents a single point of failure. Make sure you have the appropriate redundancy at that level.

- **Public Network** A public network is the network that your users will be using to access the SQL Server FCI.
- **Private Network** A private Network is typically a network solely used between the nodes of the failover cluster for cluster network traffic. Strictly speaking, you do not need a dedicated private network, but it represents a best practice that is easy to implement.
- **Windows Server Failover Clustering (WSFC)** Windows Server Failover Clustering (WSFC) is an optional Windows Server feature that you install to build a failover cluster. Think of it as the failover cluster engine.
- **Quorum** The WSFC uses the concept of a *quorum* to determine what state the failover cluster is in; whether a fail over can occur or whether the entire failover cluster should be shut down. It guarantees that only one single SQL Server FCI is accessing the database files so as to avoid data corruption and allow the FCI to start up. The following components can participate (vote) in a quorum configuration:
  - A failover cluster node
  - A disk witness, known as the quorum disk, that is located on the shared storage
  - A file share witness
  - A cloud witness (file share witness hosted in Azure)
- **Failover Cluster Instance (FCI)** A SQL Server Failover Cluster Instance (FCI) is an instance of SQL Server being installed in a failover cluster. You can install a number of FCIs per failover cluster. The number of SQL Server FCIs that are supported are:
  - 25 FCIs when using shared cluster disks
  - 50 FCIs when using SMB file shares
- **Virtual Network Name (VNN)** A virtual network name (VNN) is a virtual NetBIOS name assigned to the SQL Server FCI that is used by users to connect to the SQL Server FCI. NetBIOS names have a 15 character limit. Typically, the first SQL Server FCI that you install is a default instance that can be accessed via its computer name. All subsequent SQL Server FCIs must be named instances and are typically accessed via the computer name and instance name combination.
- **Virtual IP Address (VIP)** A Virtual IP Address is a virtual IP address bound to the VNN.

IMPORTANT **SQL SERVER FCI ONLY RUNS ON ONE NODE**

It is important to understand that a SQL Server FCI only ever runs on one of the nodes in the Failover Cluster. When a fail over occurs the SQL Server binaries need to started on the new node in the failover cluster. This can take time.

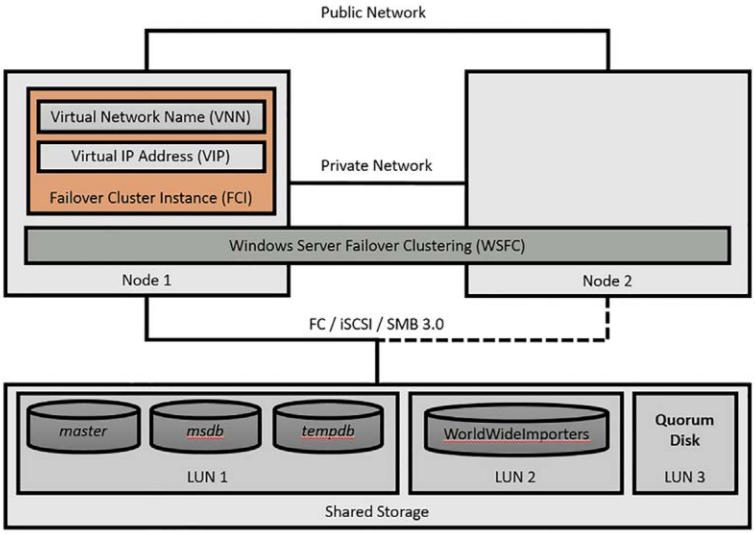

FIGURE 4-36 Failover clustering architecture

NOTE **[TEMPDB] LOCATION ON FAILOVER CLUSTER INSTANCE**

Although all system and user databases must be located on the shared storage in a failover cluster, the [tempdb] system database can be located on the local node's storage. This is possible because the [tempdb] system database is re-created by the database engine whenever the SQL Server instance is started. The benefits of using local storage is to take advantage of much faster (and/or cheaper) local storage, such as PCIe flash storage, and to offload the network/storage I/O off your SAN.

Failover in failover clustering is automatic. Being cluster aware means that a SQL Server FCI has a number of Resource DLLs that monitor the database engine's internals and communicate with the WSFC's Resource Monitors. This communication between the WSFC's Resource Monitors and the SQL Server FCI's Resource DLLs needs to ensure that a fail over is required. This can take some time in certain cases. At a high level the following steps occur in a failover incident:

1. SQL Server FCI fails on a node 1.
2. At this stage the WSFC "suspects" that the SQL Server FCI has failed.
3. The WSFC forms quorum to determine that the SQL Server FCI has failed.
4. The WSFC initiates a fail over.

5. The WSFC starts the SQL Server FCI services on node 2.
6. The SQL Server FCI services get started.
7. The SQL Server FCI's database engine connects to the databases on the shared storage.
8. The SQL Server FCI performs an automatic recovery of the databases.
   - It analyses the database's transactions logs and goes through a redo phase (replaying the transaction log after the last checkpoint).
   - It then goes through the undo phase which rolls back all uncommitted transactions.
   - Any In-memory OLTP tables are loaded into memory from the checkpoint files.
9. After the automatic recovery for a database is completed the database is accessible by users:
   - Until then the database is in the "recovering" state.
   - The database engine performs automatic recovery in order of the database's internal Database Id.

Figure 4-37 shows a high-level overview of a failover occurring in a failover cluster with a subset of the key steps from above:

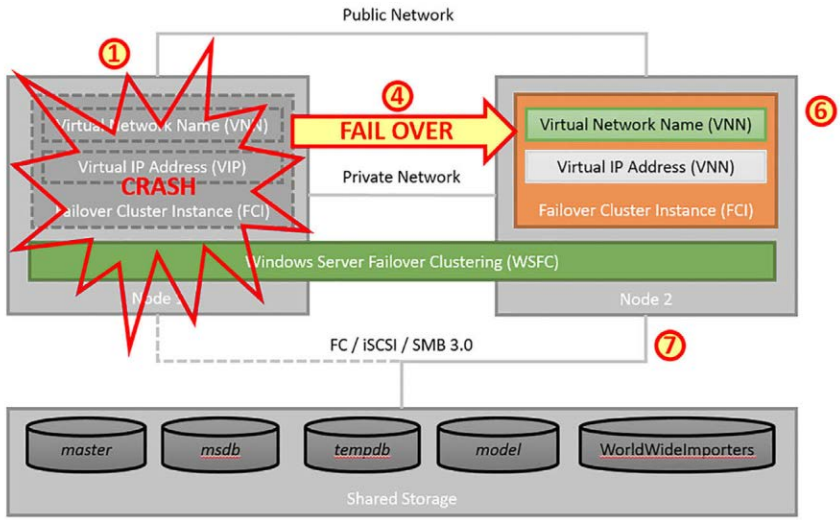

FIGURE 4-37 Failover Clustering Fail Over

The pros of using failover clustering include:

- Support for any edition of SQL Server.
  - SQL Server 2016 Standard Edition only supports 2 node failover clusters.
- Provides automatic failover.
- No data loss is possible as there is only ever a single instance of the database files. There is nothing to synchronize.

- The databases can use any recovery model.
- Scope of protection is at the instance level.
  - This a major benefit of failover clustering over Availability Groups. Because there is only a single instance of the [master] and [msdb] system databases, there is no need to synchronize logins and jobs.
- Very easy to manage.
  - The "passive" node, where no SQL Server FCIs are running, can be easily patched and rebooted.
  - Failing over is very easy as well.
- Fully supports DTC.
- Works with all applications because applications can connect directly to the SQL Server FCI's VNN or VIP.
- Supports running multiple SQL Server FCIs on all Nodes of the failover cluster. This allows you to better use the hardware resources allocated to the Node.

IMPORTANT **LICENSING FAILOVER NODES**

If you are running SQL Server FCIs on a Node of a failover cluster you will have to license SQL Server on that Node. If the Node is doing nothing, there is no need to license that Node.

The cons of using failover clustering include:

- Relies on shared storage, which represents a single point in failure.
- Potentially requires a SAN, which can be expensive or more difficult to maintain.
- An organization's reticence to use newer shared storage technologies, specifically the SMB 3.0 and related technology introduced in Windows Server 2012. IT Managers and Solution Architects are hesitant to use newer technology for no good reason.
- Having to use named instances for all SQL Server FCIs installed after the default instance. Names instances are a bit more difficult to manage and connect to.
- There is no scale out technology natively in failover clustering. It is not a multi-master solution.
- If you do not use the "passive" Node, you are not getting a good return on your investment.
- Not an easy high availability solution to implement between data centers.

NOTE **GEO-CLUSTERING**

Although geo-clustering, or "stretch clustering" as it is commonly referred to, is possible and has been commonly implemented for the last decade, its discussion is outside the scope of this book. Geo-clustering typically requires some sort of storage replication, at the hardware or software level.

Use failover clustering in the following use cases:

- You require an easy to manage high availability solution with no scale out requirements.
- Your databases need to remain in the SIMPLE recovery model. Availability Groups only work for databases that are using the FULL recovery model.
- You have determined that Availability Groups will impact the performance of your database solutions.
- You do not want the complexity of managing the logins, jobs and other related external dependencies between the different vendors.
- You can't use Availability Groups because the applications that will be connecting to the database will have issues with the listener used by Availability Groups.

Configure failover clustering

With every release of Windows, failover clustering improves both the underlying technology and the installation processes. Always try to use the latest version of Windows Server when deploying a failover clustering solution.

Most failover cluster solutions rely on shared storage (such as that provided by a SAN, either a hardware or a software solution). However, since SQL Server 2014 and Windows Server 2012, you can implement failover clusters using shares instead, relying on the SMB 3.0 stack. In this case you might be taking advantage of Windows Server's Scale-Out File Server (SOFS) capabilities, which can provide continuous availability.

MORE INFO **SQL SERVER FAILOVER CLUSTER SUPPORT FOR FILES SHARES**

For more information about using SMB shares for SQL Server FCIs visit: ***https://msdn.microsoft.com/en-us/library/hh759341.aspx and https://technet.microsoft.com/en-us/library/hh831349(v=ws.11).aspx.***

To practice setting up a failover cluster set up the following VMs in Hyper-V:

1. A domain controller (ADDS) for the SQL.LOCAL domain
2. It should be connected to the public network
3. A SQL Server instance (NODE1) joined to SQL.LOCAL
4. It should be connected to the public, private and iSCSI networks through 3 separate NICs
5. A SQL Server instance (NODE2) joined to SQL.LOCAL
6. It should be connected to the public, private and iSCSI networks through 3 separate NICs
7. A file server (STORAGE) joined to the SQL.LOCAL
8. This server will be configured as the shared storage for the failover cluster
9. It should be connected to the public and iSCSI networks through 2 separate NICs

The following steps show how to configure the shared storage that the failover cluster will use.

1. Log into the storage server (STORAGE) that you plan to set up as an iSCSI target.
2. Open up Server Manager.
3. Click on Local Server in the left most pane.
4. Click on Manage drop-down in the top menu bar and choose Add Roles And Features to start the Add Roles And Features Wizard.
5. Click on the Next button in the Before You Begin page.
6. Choose the Role-Based Or Feature-Based Installation in the Select Installation Type page.
7. Ensure your local server is selected in the Server Pool and click on the Next button.
8. In the Server Roles page expand the File And iSCSI Services folder and select iSCSI Target Server. Then click on the Next button.
9. Confirm that the iSCSI Target roles is being installed on the Confirm Installation Selections page and click on the Install button.
10. Confirm that the iSCSI Target roles has been successfully installed, as shown in Figure 4-38 and close the wizard.
11. Select the Failover Clustering check box to install the Failover Clustering feature.
12. You need to set up a number of iSCSI virtual disks for your failover cluster. The SQL Server FCI will use the disks show in Table 4-4.

TABLE 4-4 Failover Cluster shared Disk Properties

Disk Number	VHDX FIle Name	Size	FCI disk letter	Purpose
0	Quorum.vhdx	1GB		Quorum disk for failover cluster
1	SQLData.vhdx	100GB	D:	SQL Server FCI user database data files
2	SQLLog.vhdx	50GB	L:	SQL Server FCI user database transaction log files
3	TempDBData.vhdx	10GB	T:	SQL Server FCI [tempdb] system database data files
4	TempDBLog.vhdx	20GB	U:	SQL Server FCI [tempdb] system database transaction log files

13. Back in Server Manager click on File And Storage Services the left most pane.
14. Click on the iSCSI option.
15. Click on the To Create An iSCSI Virtual Disk, start the New iSCSI Virtual Disk Wizard to start the New iSCSI Virtual Disk Wizard.
16. In the iSCSI Virtual Disk Location choose the appropriate disk volume and click on the Next button.
17. In the iSCSI Virtual Disk Name page provide the Name and Description and click on the Next button.

18. In the iSCSI Virtual Disk Size page configure a Dynamically Expanding disk that is 1GB in size for the quorum disk. You do not need a bigger size than that for a failover cluster's shared quorum disk. Click on the Next button when you are finished.
19. In the Assign iSCSI target page choose the New iSCSI target option and click on the Next button to create a target for the iSCSI disk that you are creating. The 2 nodes of the failover cluster will be the iSCSI targets.
20. In the Specify Target name page provide the following details before clicking on the Next button.
    - Name: SQLCLUSTER
    - Description: SQL Server 2016 Cluster
21. In the Specify Access Servers page click on the Add button to add the first node as an iSCSI initiator.
22. In the Add Initiator ID dialog box configure the first node as an iSCSI initiator by providing its computer name and click on the OK button.
23. In the Add Initiator ID dialog box configure the second node as an iSCSI initiator and click on the OK button.
24. In the Specify Access Servers page make sure you have added the 2 correct nodes, as seen in Figure 4-38 and click on the Next button.

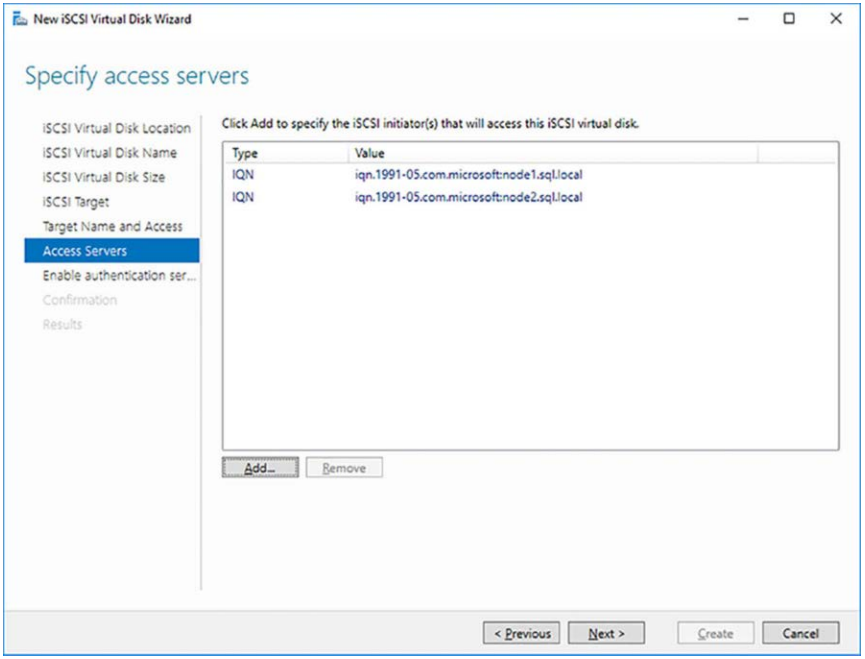

Figure 4-**38** Availability Group Synchronous Commit

25. As there will be no authentication, click on the Next button in the Enable Authentication page.
26. Confirm the properties of the iSCSI virtual disk you are about to create, as shown in Figure 4-39 and click on the Create button.

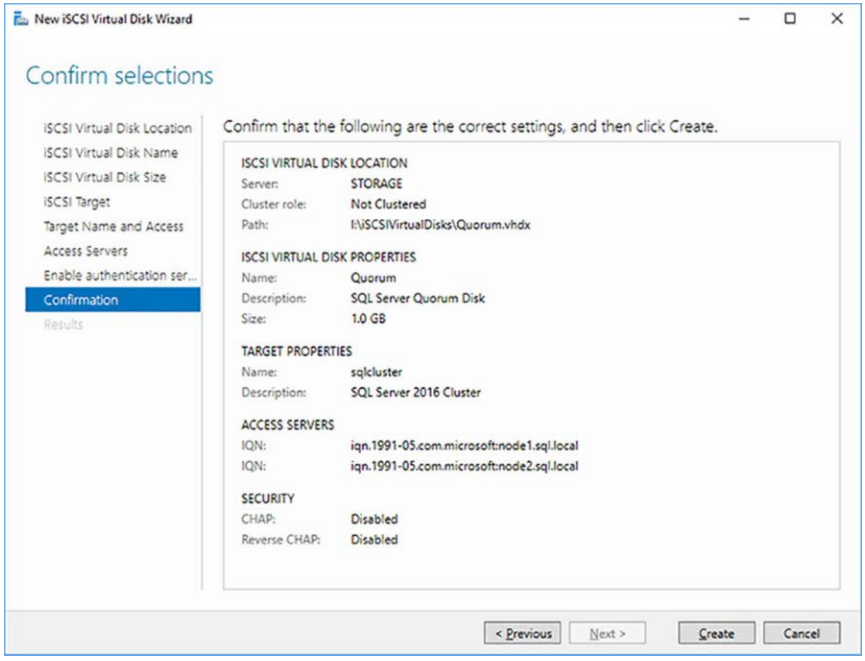

FIGURE 4-39 iSCSI virtual disk confirmation

27. Click on the Close button after the successful creation of your iSCSi virtual disk for the quorum disk of the failover cluster.
28. Repeat the above iSCSI virtual disk creation steps to create a 100GB thinly provisioned iSCSI disk for the databases' data files.
29. Repeat the above iSCSI virtual disk creation steps to create a 50GB thinly provisioned iSCSI disk for the databases' transaction log files.
30. Repeat the above iSCSI virtual disk creation steps to create a 20GB thinly provisioned iSCSI disk for the [tempdb] system database's data files.
31. Repeat the above iSCSI virtual disk creation steps to create a 10GB thinly provisioned iSCSI disk for the [tempdb] system database's transaction log.
32. In Server Manager, you should have 5 iSCSI virtual disks created for the failover cluster, as shown in Figure 4-40.

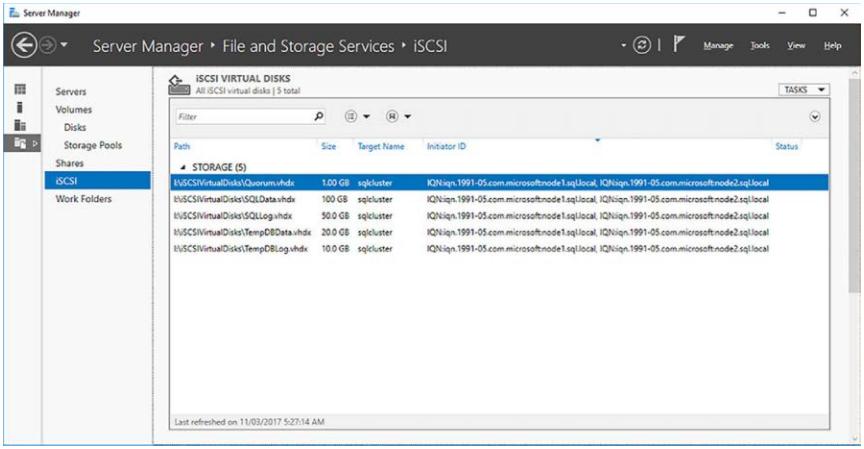

FIGURE 4-40 iSCSI disks configured for failover cluster

33. You need to configure the iSCSI Target Server to only communicate over the dedicated iSCSI network. In Server Manager click on Servers, then right-click on your storage server and select the iSCSI Target Settings option.

34. Select just the iSCSI network, as shown in Figure 4-41 and click on the OK button.

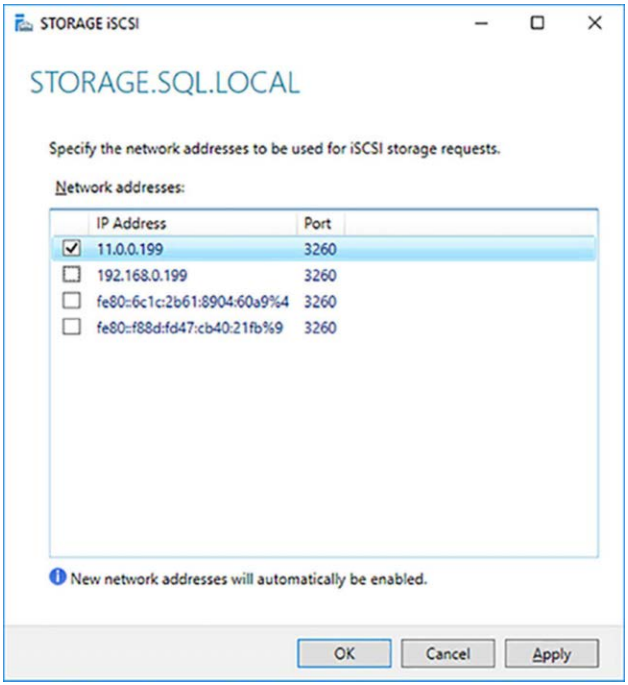

FIGURE 4-41 Isolating iSCSI traffic on dedicated network

You have now created 5 iSCSI LUNs for your failover cluster. You now need to configure your failover cluster. You need to perform the following high-level steps:

- Install and configure the iSCSI Initiator on each Node that will be part of the failover cluster.
- Format the iSCSI disks.
- Install WSFC on all the Nodes that are going to be part of the failover cluster.
- Create a failover cluster with all the Nodes.
- Create a SQL Server FCI by installing it the first Node of the failover cluster.
- Complete the installation of the SQL Server FCI by installing SQL Server on the additional Nodes of the failover cluster and joining the SQL Server FCI installed on the first Node.

The following steps show how to install and configure the iSCSI Initiator on each of the Nodes of the cluster.

1. Log into the first Node that will be part of your failover cluster.
2. Open Server Manager.
3. Select the iSCSI Initiator for the Tools menu.
4. Click on the Yes to confirm that you want the Microsoft iSCSI service to start automatically whenever the computer restarts.
5. Type in the name of your iSCSI target server into the Target text box and click on the Quick Connect button.
6. In the Quick Connect dialog box click on the Connect button and then the Done button so that the Node will be able to connect to the iSCSI Target Server LUNs as required.
7. Confirm that your Node is connected to the iSCSI target server you created earlier and click on the OK button, as shown in Figure 4-42.

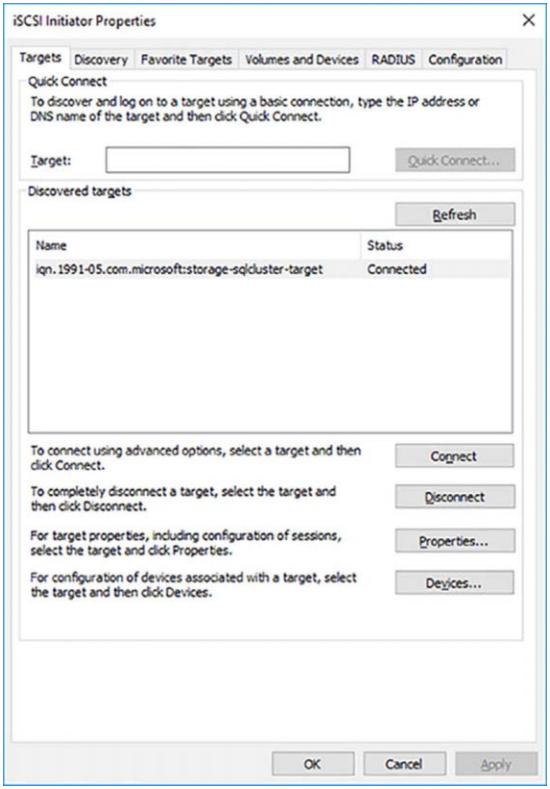

FIGURE 4-42 Successfully connect to iSCISI Target Server

8. Configure the iSCSI initiator on the other nodes that you plan to add to the failover cluster
9. The next step is to format the iSCSI disks using the following properties:
   - NTFS file system.
     - Although ReFS is supported, it not recommended for SQL Server database files due to performance reasons.
   - 64KB allocation unit size.
     - The 1GB quorum disk can be formatted with the default (4KB) allocation unit size.
   - Do not allow the files on the drives to have their contents indexes in addition to file properties.
   - Assign the drive letters as per Table 4-4.

The following steps show how to format the iSCSI disks:

10. Log into the first Node that will be part of your failover cluster.
11. Open Disk Management.

12. Right click on the first iSCSI disk, which should be the 1GB quorum disk and click Online to bring it online.
13. Right-click on the same disk and select the Initialize Disk option.
14. In the Initialize Disk dialog box choose the MBR (Master Boot Record) option and click on the OK button.
15. In Disk Management, right click on the same disk and select the New Simple Volume option to format the disk.
16. In the New Simple Volume Wizard click on the Next button in the welcome screen.
17. In the Specify Volume Size page click on the Next button to format the simple volume with the default maximum possible disk size.
18. In the Assign Drive Letter Or Path screen choose the Do Not Assign A Drive Letter Or Drive path option. The quorum disk for the failover cluster does not require a drive letter to work.
19. Configure the format settings for the quorum disk and click on the Next button.
    - For a production environment, you should normally perform a full format to maximize performance and ensure there is no storage problems.
    - For your lab or development environment you should perform a quick format so as to save actual disk space.
20. Review the disk format settings and click on the Finish button to format the disk.
21. Format the remaining disks using the same steps as above. Remember to format the disks using the NTFS file system and 64KB allocation unit size. Use Figure 4-43 as a guide for the drive letters and volume names.

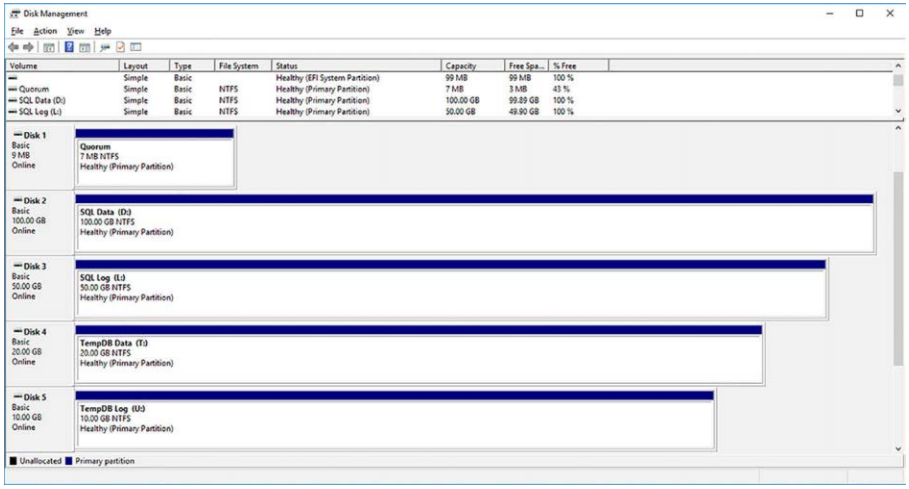

FIGURE 4-43 Failover cluster formatted volumes

You can now set up the failover cluster that the SQL Server FCI will be using. The first step is to install WSFC on all of the Nodes of the failover cluster.

Use the following steps to install WSFC on the first node of your failover cluster.

1. Open up Server Manager on the first Node.
2. Choose Add Roles And Features from the Manage drop-down list.
3. In the Add Roles And Features Wizard click on the Next button.
4. Choose the Role-Based Or Feature-Based Installation and click on Next.
5. Ensure your local server is selected in the Server Pool and click on the Next button.
6. Do not install any roles. Click on the Next button in the Select Server Roles page.
7. Select the Failover Clustering check box to install the Failover Clustering feature.
8. The Add Roles And Features Wizard will, by default, want to install the Failover Clustering tools and Powershell modules. Confirm this action by clicking on the Add Features button.
9. Confirm that you are installing Failover Clustering and the related tools before clicking on the Install button to begin the installation.
10. Confirm the installation was successful and click on the on the Close button to finish.
11. Repeat the WSFC installation on the other Nodes in the failover cluster using the same steps.

After installing the WSFC on all of the Nodes of your failover cluster you are reading to create the cluster. To install a failover cluster, you will need to have rights to modify your AD environment. Consequently, you will need to do one of the following:

- Log in as Domain Administrator when creating the failover cluster.
- Log in as yourself and get the Domain Administrator to run the setup executables as themselves using the Run As capability in the Windows OSE.
- Get the Domain Admin to pre-stage the cluster computer objects in Active Directory Domain Services as described in *https://technet.microsoft.com/en-us/library/dn466519(v=ws.11).aspx*.

The following steps show how to create a failover cluster.

1. Log into your Node as the Domain Administrator.
2. Open Failover Cluster Manager, which has now been installed on your server.
3. Click on the Create Cluster action in the right-most pane. This will start the Create Cluster Wizard.
4. Click on the Next button in the Before You Begin page of the Create Cluster Wizard.
5. Enter the name of the first Node that you want to add to the failover cluster and click on the Add button. The Create Cluster Wizard will validate the server's existence and add it to the bottom text box using its fully qualified domain name (FQDN).
6. Add all the nodes to you cluster, then click on the Next button as shown in Figure 4-44.

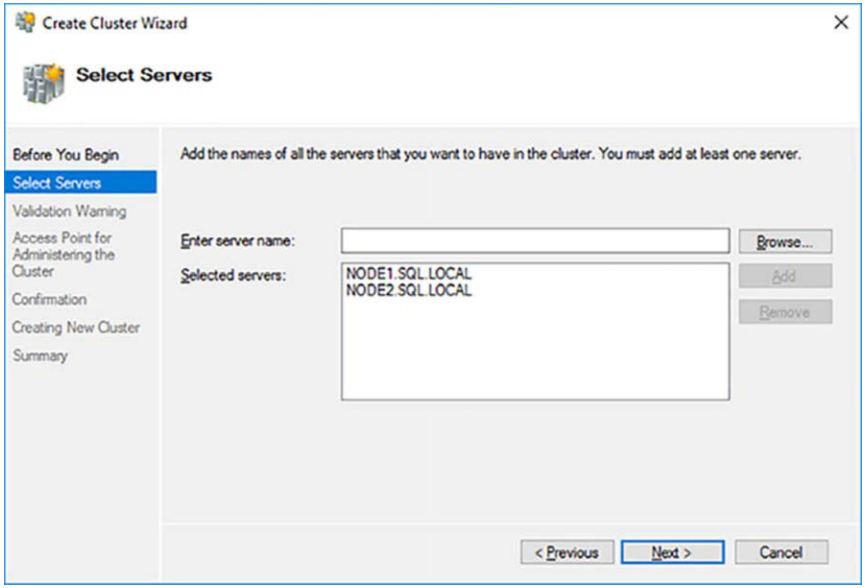

FIGURE 4-44 Selected nodes for failover cluster

7. You need to validate that your nodes are capable of running a failover cluster that will be supported by Microsoft. Click on the Next button to run the configuration validation tests.
8. The Validate A Configuration Wizard will by default automatically run all of the appropriate cluster validation tests for you. Click on the Next button in the Validate A Configuration Wizard.
9. It is a best practice to run all the cluster validation tests. Click on the Next button to start the validation tests, as shown in Figure 4-45.

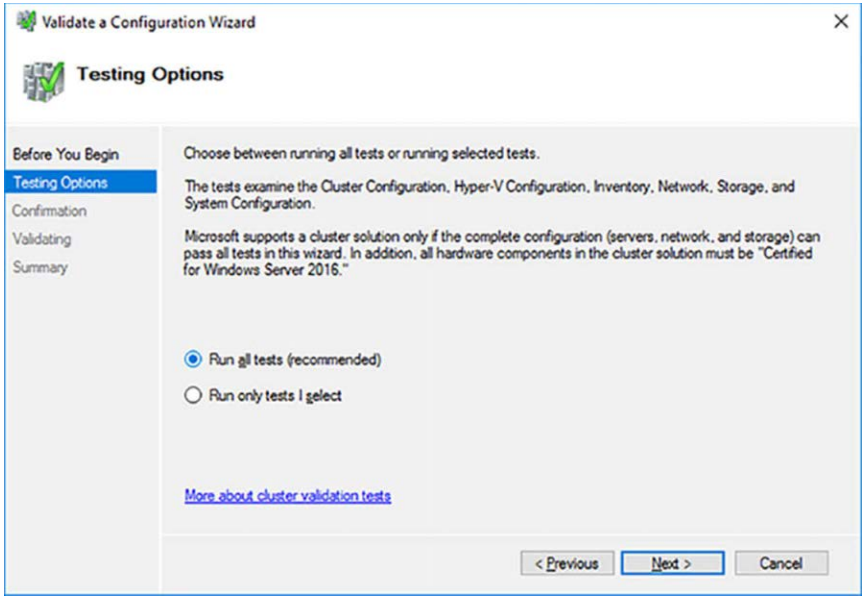

FIGURE 4-45 Running all cluster validation tests

10. Figure 4-46 shows you what tests will be run by default. Note that the Storage Space Direct tests will not be run because you have not installed and configured this feature.

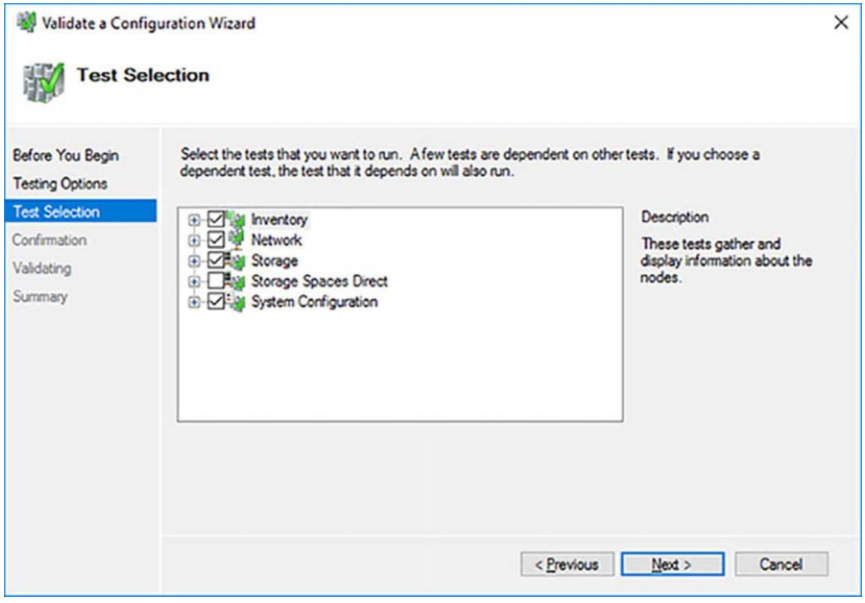

FIGURE 4-46 Possible cluster validation tests

11. Review the servers to test and the tests that will be run. Click on the **Next** button to start the failover cluster validation tests. Figure 4-47 shows the cluster validation tests performing disk failover tests, which are critical to a failover cluster. Note the SCSI-3 persistent reservations tests, and another critical test was successful.

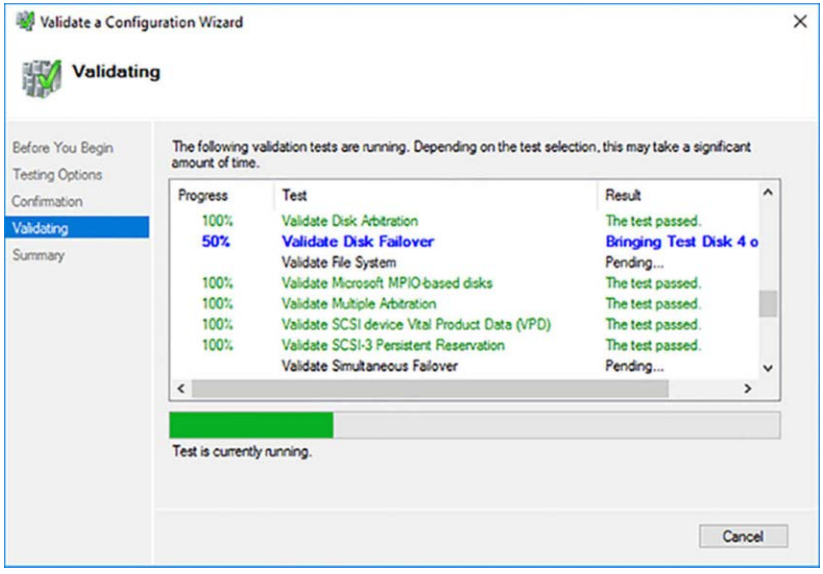

FIGURE 4-47 Cluster validation tests executing disk failover tests

12. Wait for the cluster validation tests to complete. It is not uncommon to have warnings, such as that software patches might be missing. You can fix any issues and re-run the cluster validation issues if that is warranted. Click on the View Report button to see if there are any serious issues.
13. Review the Failover Cluster Validation Report, shown in Figure 4-48. It is a phenomenally good practice to keep it for support reasons.

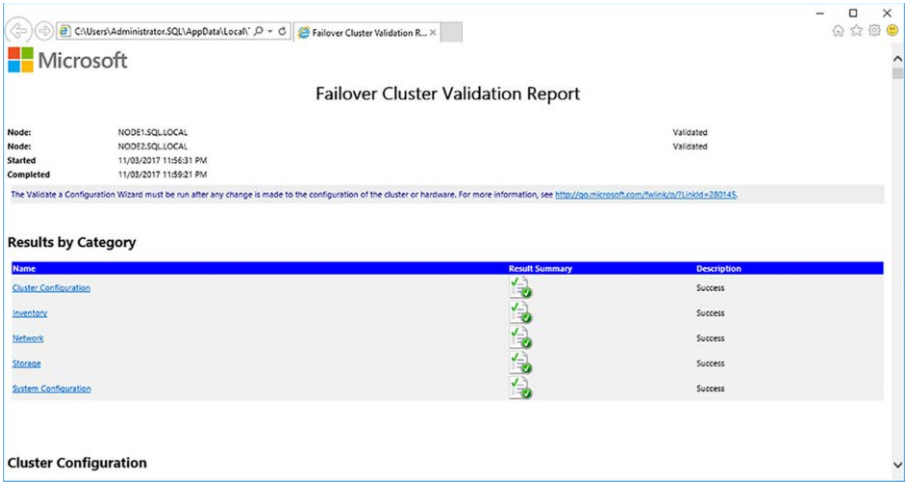

FIGURE 4-48 Successful failover cluster validation report

14. Close the report when you have completed your review.
15. Click on the Finish button to close the Validate A Configuration Wizard.
16. Provide a computer name and IP address for the Client Access Point (CAP), as shown in Figure 4-49, and click on the Next button. The CAP is used to manage the cluster.

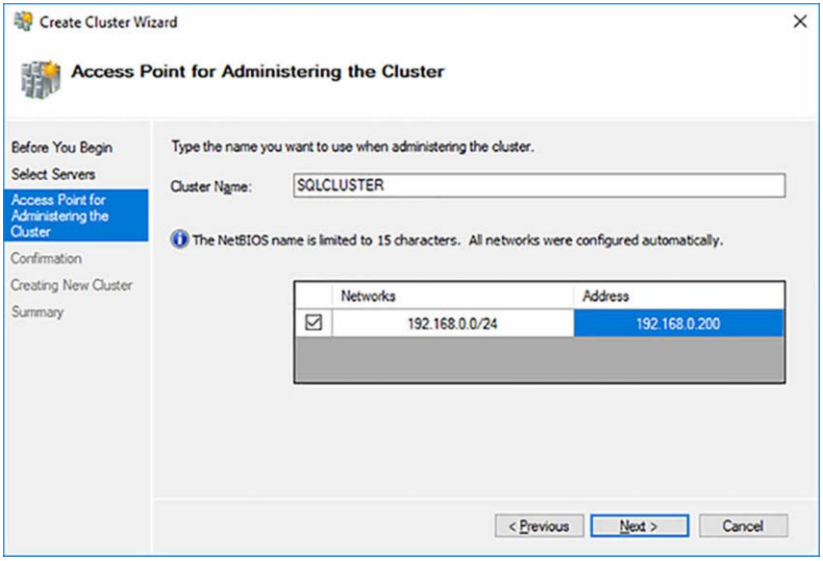

FIGURE 4-49 Client Access Point configuration

17. Check the Add All Eligible Storage To The Cluster option, review and confirm the creation of the failover cluster by clicking on the Next button.

18. Wait for the failover cluster to be created. Figure 4-50 shows one of the most important steps, where the Computer Name Object (CNO) is created in Active Directory (AD).

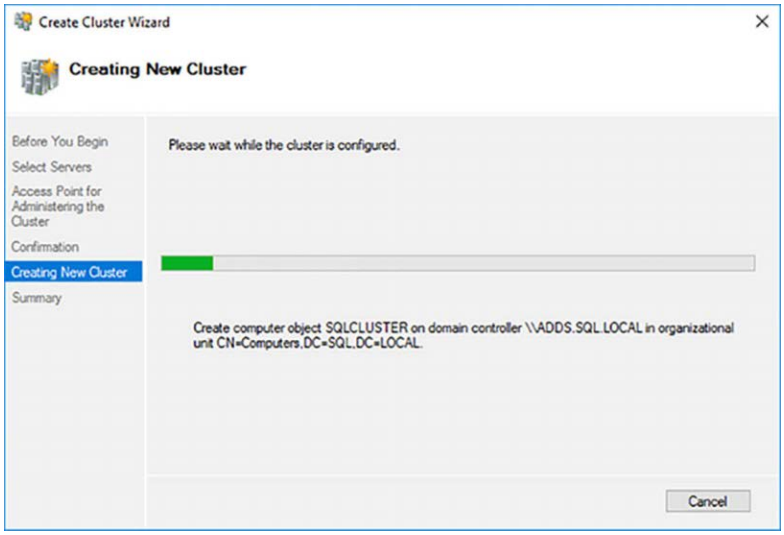

FIGURE 4-50 Creating CNO in AD for failover cluster

IMPORTANT **CREATING CNO IN AD FOR FAILOVER CLUSTER**

You must be logged in as Domain Administrator to create the CNO in AD during the failover cluster installation. If that is not possible you will have to have pre-stage your AD environment as per *https://technet.microsoft.com/library/cc731002(WS.10).aspx*.

19. Review the failover cluster creation Summary page. Click on the View Report button to view the detailed failover cluster creation report.
20. Review and save the Create Cluster report, shown in Figure 4-51, looking out for any errors and warnings.

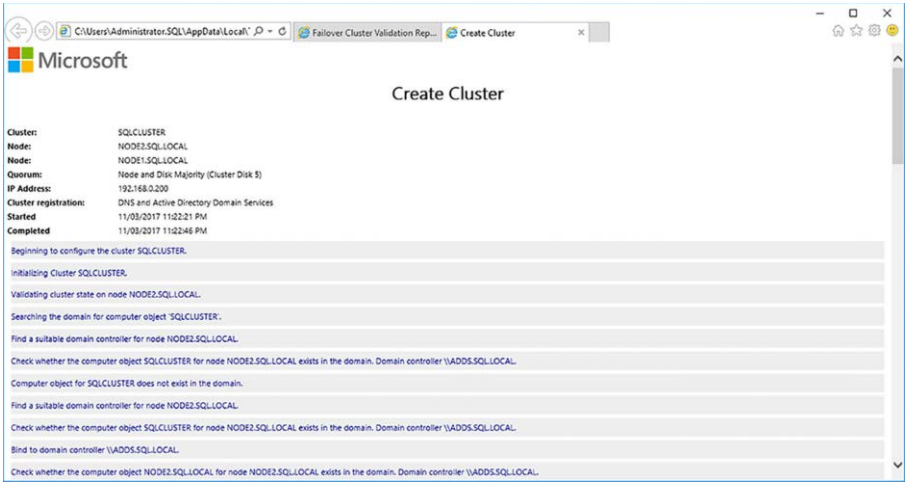

FIGURE 4-51 Create Cluster report

Your failover cluster should now be created. Before you install the SQL Server FCI it is a good idea to change a few elements in your failover cluster to ensure optimal operation and make it easier to manage.

Perform the following steps:

1. In Failover Cluster Manager connect to your failover cluster.
2. Click on the Networks folder.
3. The default configuration is for the networks to be serially identified. It is a best practice to rename them to help troubleshoot and administer your failover cluster correctly. Furthermore, the default is to send cluster traffic across all three networks. In this case, you do no not want cluster traffic to be sent across the iSCSI network. You want it purely for our iSCSI storage traffic.
4. Right-click on Cluster Network 1 (192.168.0.0) and select its properties. Change its properties as shown below:
   - Name: Public Network
   - Allow cluster network communication on this network
   - Allow clients to connect through this network
5. Right click on Cluster Network 2 (10.0.0.0) and select its properties. Change its properties as shown below:
   - Name: Private Network
   - Allow cluster network communication on this network
6. Right click on Cluster Network 1 (11.0.0.0) and select its properties. Change its properties as shown below:

- Name: iSCSI Network
- Do not allow cluster network communication on this network

7. Make sure you cluster networks have been reconfigured as shown in Figure 4-52

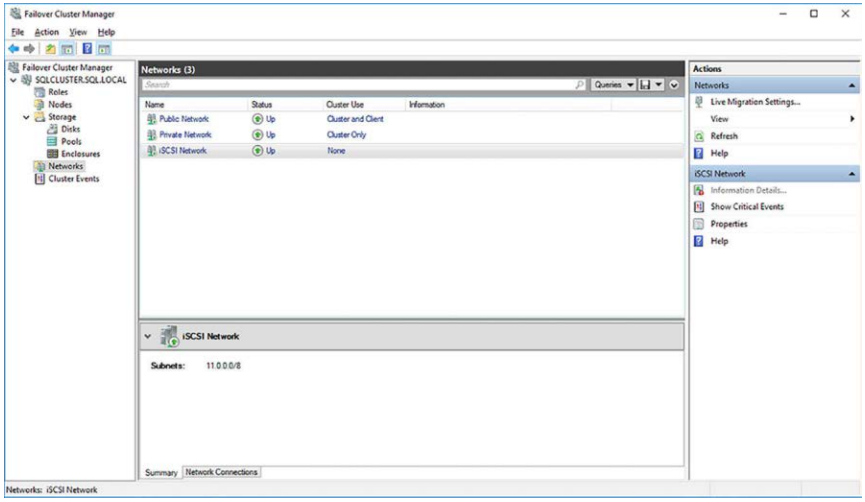

FIGURE 4-52 Re-configured cluster networks

8. Click on the Disks folder. All of the disks have also been named serially. Again, it is a best practice to rename them to help administration and minimize mistakes.
9. Right-click on the 1GB cluster disk being used as a disk witness and select Properties.
10. Rename the cluster disk to "Quorum Disk" to indicate its purpose.
11. Rename all cluster disks, as shown in Figure 4-53, to match their intended purpose.

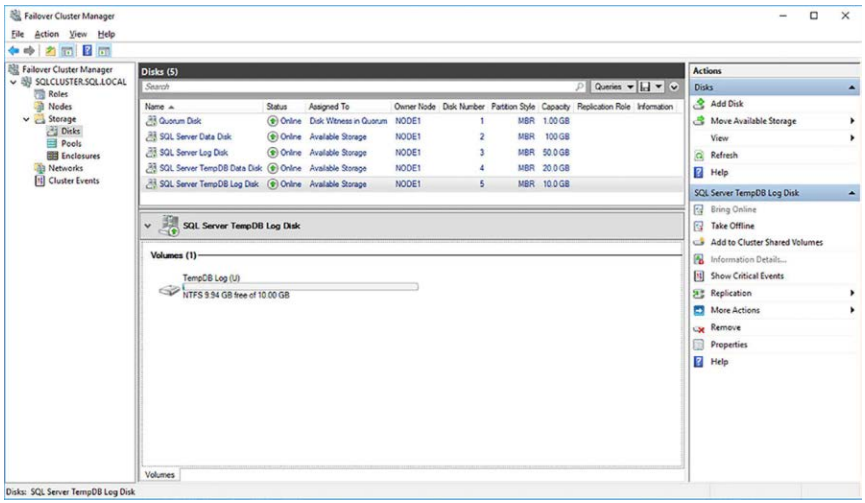

FIGURE 4-53 Renamed cluster disks

Finally, you are ready to install the SQL Server FCI. The process to create SQL Server FCI involves:

- Run the SQL Server setup on the first node to install a SQL Server FCI on the first node
- Run the SQL Server setup on the second node to join it to the SQL Server FCI

Use the following steps to install start the installation of the SQL Server FCI on the failover cluster:

1. Log into Node 1 of the failover cluster as an administrator.
2. Mount the SQL Server Developer ISO and run the setup program.
3. Click on the Installation link in the SQL Server Installation Center.
4. Click on the New SQL Server Failover Cluster Installation link, as shown in Figure 4-54, to start the Install A SQL Server Failover Cluster setup.

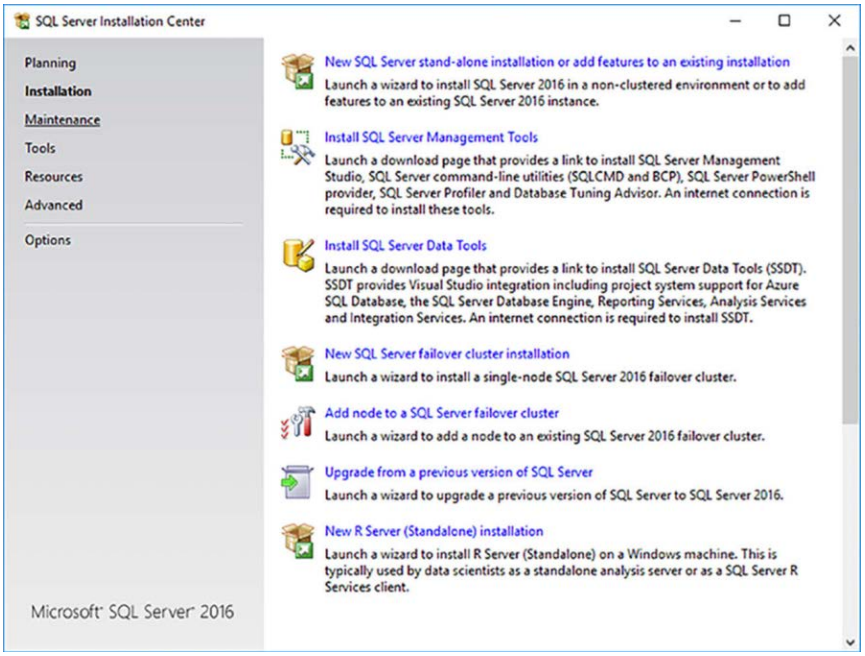

FIGURE 4-54 New SQL Server failover cluster installation

5. In the Product Key page of the Install A SQL Server Failover Cluster setup enter the product key to specify a free edition.
6. In the License Terms page accept the license terms and click on the Next button.
7. In the Global Rules page let the setup engine check to see if there are any blockers for the installation and click on the Next button.

8. In the Microsoft Update page, you can let the setup process check for important updates. Don't. It's easier to manually install any updates. Click on the Next button.
9. Click on the Next button in the Product Updates page.
10. The Install Failover Cluster Rules page, shown in Figure 4-55, runs a number of checks to see if anything would block the FCI install. Review warnings and correct any errors as required. In this case, it is passing through the warning generated by the failover cluster validation done earlier. Click on the Next button when you are ready to proceed to the next step.

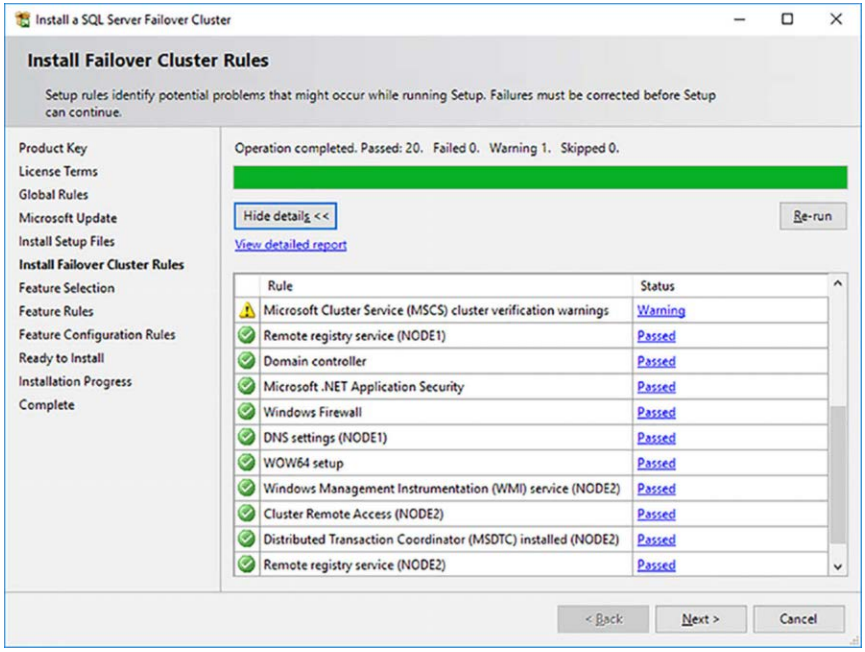

FIGURE 4-55 SQL Server FCI setup install failover cluster rules

11. In the Feature Selection page, shown in Figure 4-56, select the appropriate features. When installing a SQL Server FCI consider the following.
    - The setup process will automatically install the SQL Server Replication, Full-Text, and Semantic Extractions for Search and Data Quality Services.
    - SSRS is not cluster aware.
    - SSIS is not cluster aware.
    - Consider installing SSAS as a separate FCI.

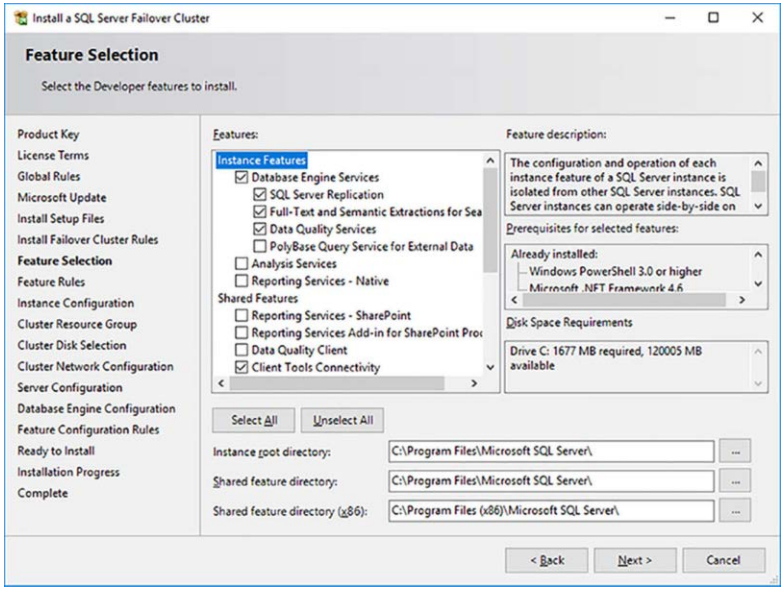

FIGURE 4-56 SQL Server FCI setup feature selection

12. In the Instance Configuration page provide a name for the SQL Server instance, as shown in Figure 4-57 and click on the Next button. In a WSFC you can only install a single default instance. It will be access via its network name. All subsequent instances will be named instances that can be accessed via their network name\instance name.

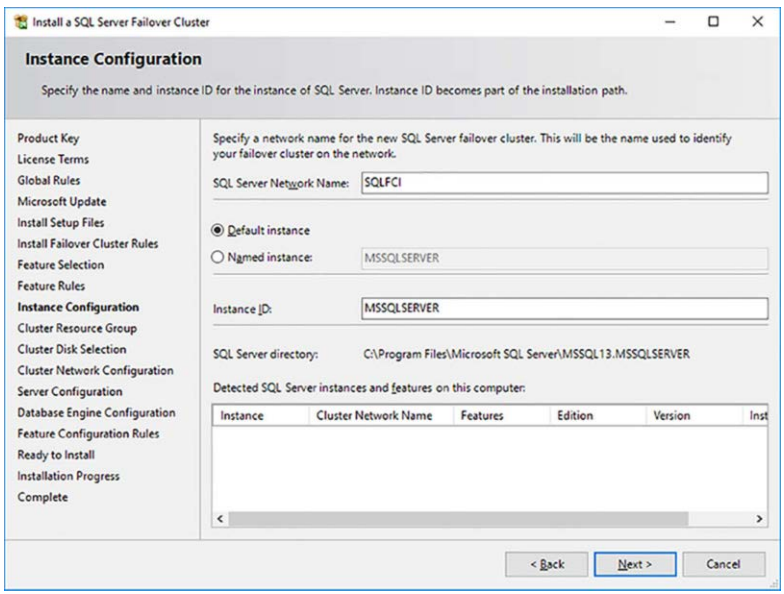

FIGURE 4-57 SQL Server FCI setup instance configuration

13. In the Cluster Resource Group provide a name for the SQL Server Cluster Resource Group name, as shown in Figure 4-58, and click on the Next button. Consider having a naming standard if you plan to install multiple SQL Server FCIs in a failover cluster.

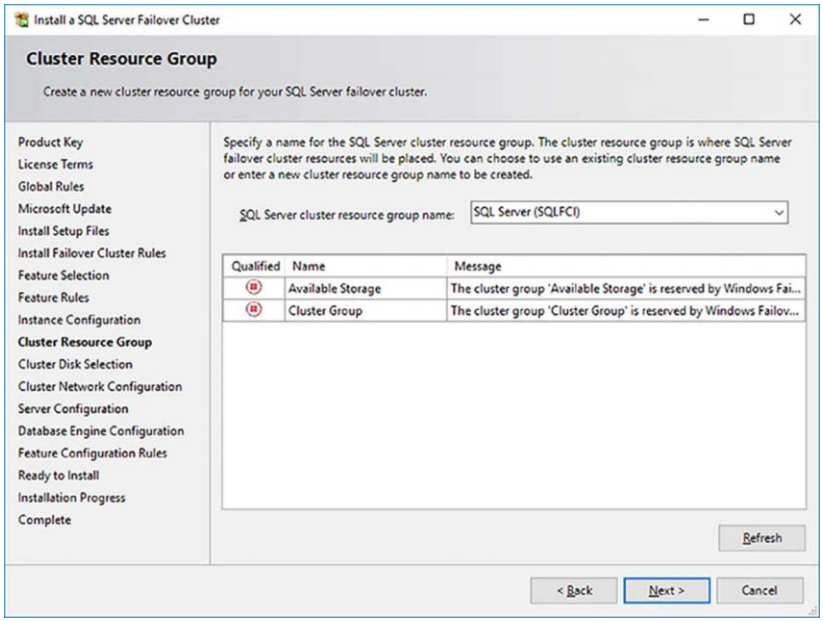

FIGURE 4-58 SQL Server FCI setup cluster resource group

14. Select the cluster disks that your SQL Server FCI will use in the Cluster Disk Selection page, as shown in Figure 4-59, and click on the Next button. Note the benefit of renaming the cluster disks in the failover cluster earlier.

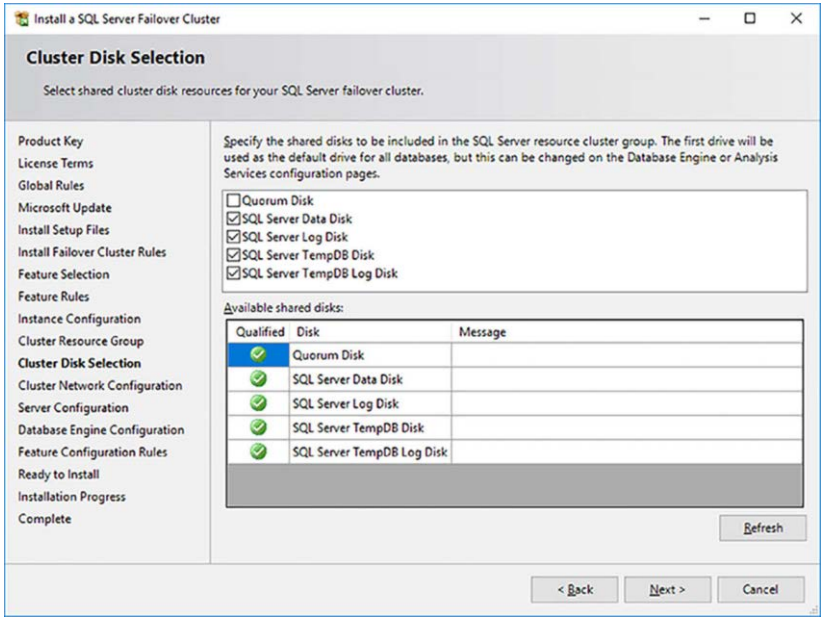

FIGURE 4-59 SQL Server FCI setup cluster disk selection

15. Provide an IP address in the Cluster Network Configuration page, as shown in Figure 4-60,s and click on the Next button.

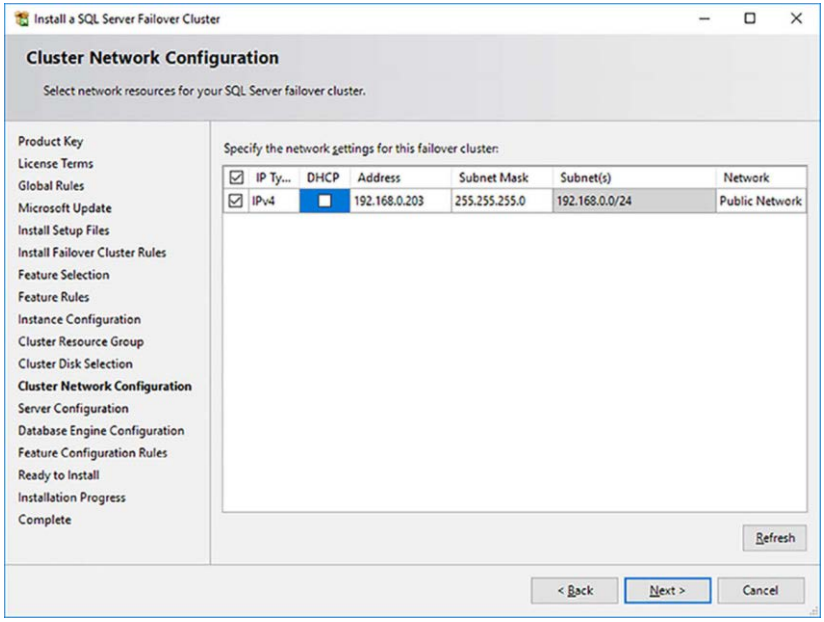

FIGURE 4-60 SQL Server FCI setup cluster network configuration

16. Enter the service account and password details, as shown in Figure 4-61.

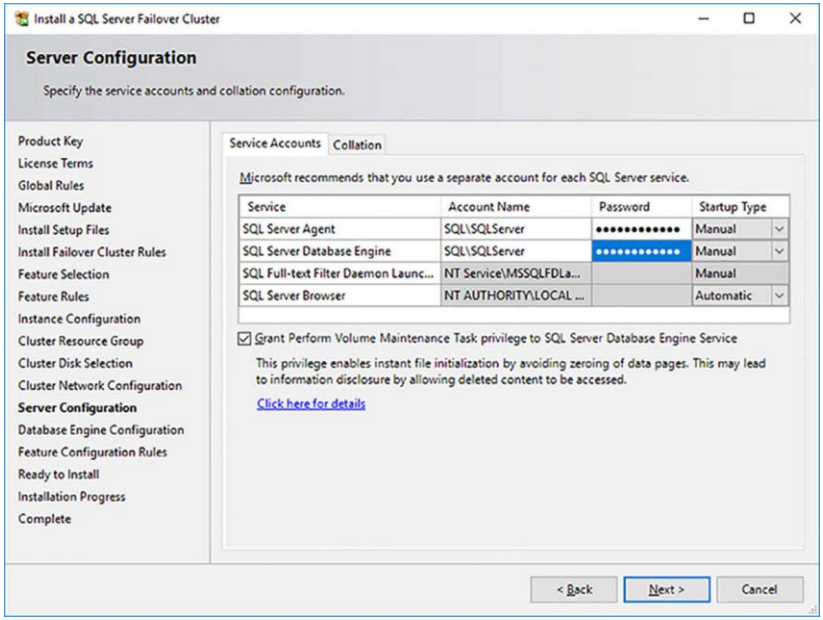

FIGURE 4-61 SQL Server FCI setup service accounts

17. Click on the Collation tab, and enter the required collation.
18. Click on the Next button.
19. In the Database Engine Configuration page configure the Server Configuration details, as shown in Figure 4-62.

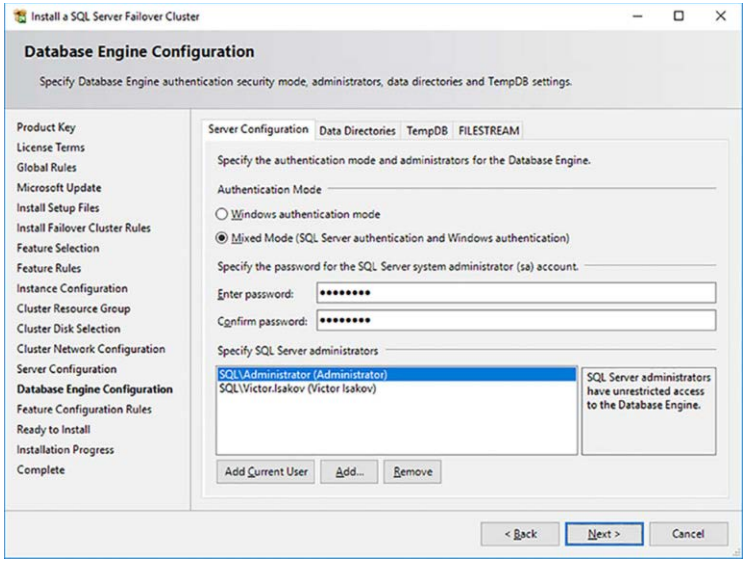

FIGURE 4-62 SQL Server FCI setup server configuration

20. Click on the Data Directories tab and configure the paths for the database and backup paths, as shown in Figure 4-63.

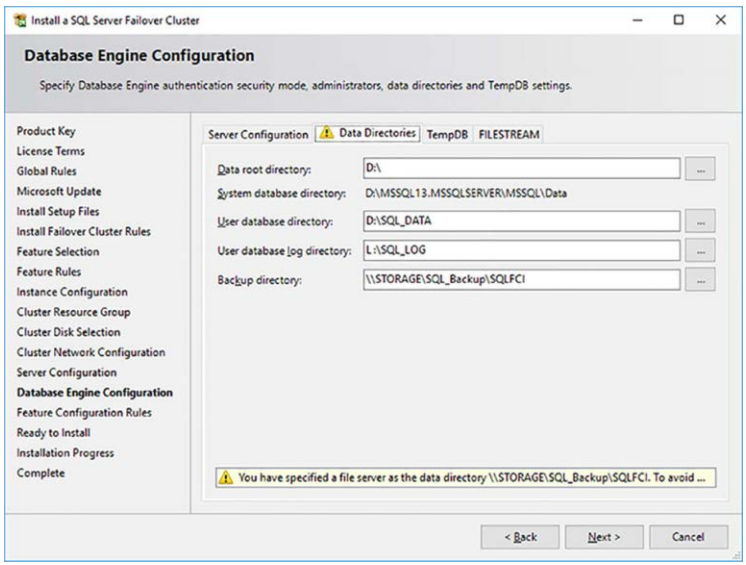

FIGURE 4-63 SQL Server FCI setup data directories

21. Click on the TempDB tab and configure the paths for the [tempdb] system database, as shown in Figure 4-64.

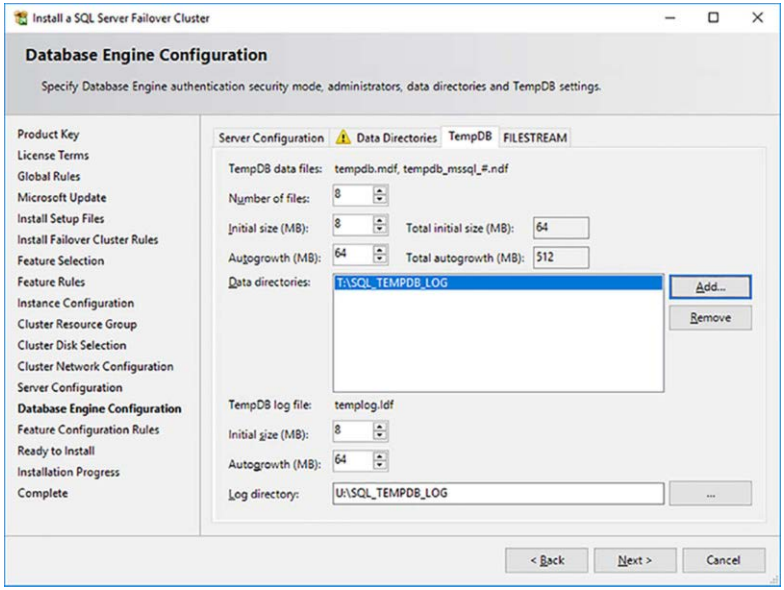

FIGURE 4-64 SQL Server FCI setup TempDB configuration

IMPORTANT **CREATING TEMPDB LOCALLY IN A FCI**

The tempdb system database can be located on local storage in a FCI, since it is automatically recreated each time SQL Server starts up. Locating tempdb on local flash storage represents a great optimization technique for database solutions that heavily utilize it.

22. Click on the FILESTREAM tab and configure your filestream options before clicking on the Next button.
23. In the Feature Configuration Rules page, as shown in Figure 4-65, let the setup engine run its checks and click on the Next button.

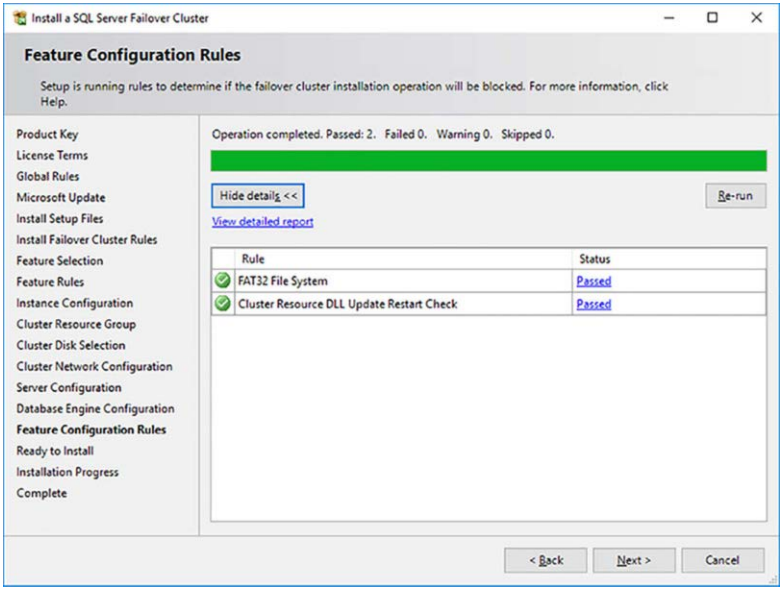

FIGURE 4-65 SQL Server FCI setup feature configuration rules

24. Review the summary of your SQL Server FCI setup and click on the Install button to initiate the installation procedure.
25. Once the setup has completed, review the summary to ensure nothing has gone wrong. Save the summary log for support reasons and click on the Close button to complete close the installer.

You now need to complete the installation of the SQL Server FCI by installing the same configuration on the second Node. Fortunately, this is a lot easier as the installation on the first node has most of the information needed to complete the installation on the second Node, barring the service account passwords.

Use the following steps to complete the installation of the SQL Server FCI on the failover cluster:

1. Log into Node 2 of the failover cluster as an administrator.
2. Mount the SQL Server Developer ISO and run the setup program.
3. Click on the Installation link in the SQL Server Installation Center.
4. Click on the Add Node To A SQL Server Failover Cluster link, as shown in Figure 4-54, to start the Install A SQL Server Failover Cluster setup.

In the Product Key page of the Install A SQL Server Failover Cluster setup enter the product key to specify a free edition, like for Node 1, and click on the Next button.

1. In the License Terms page accept the license term and click on the Next button.
2. In the Global Rules page let the installer check to see if there are any blockers for the installation and click on the Next button.

3. In the Microsoft Update page, like for Node 1, click on the Next button for Node 1.
4. Click on the Next button in the Product Updates page, for Node 1.
5. In the Install Setup Files page let the installer install the required setup files and click on the Next button.
6. The Add Node Rules page, shown in Figure 4-66, runs a number of checks to see if anything would block the Node being added to the FCI. Review and warnings and correct any errors as required. Click on the Next button when you are done.

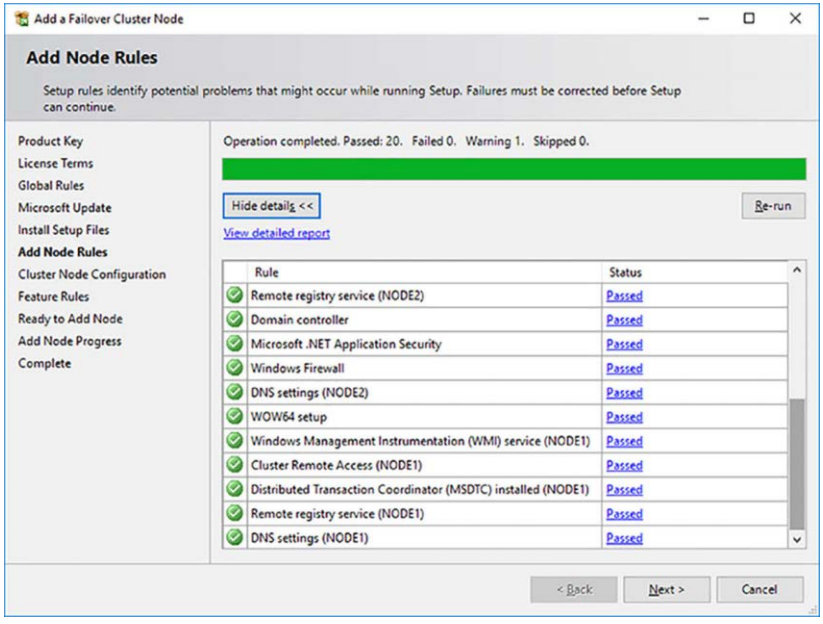

FIGURE 4-66 SQL Server FCI setup install add node rules

7. In the Cluster Node Configuration page, shown in Figure 4-67, the installer shows you details of the SQL Server FCI you are going to become part of. Remember that you can have multiple SQL Server FCIs in a failover cluster. In this case, there is only one SQL Server FCI. Click on the Next button when you have reviewed the page.

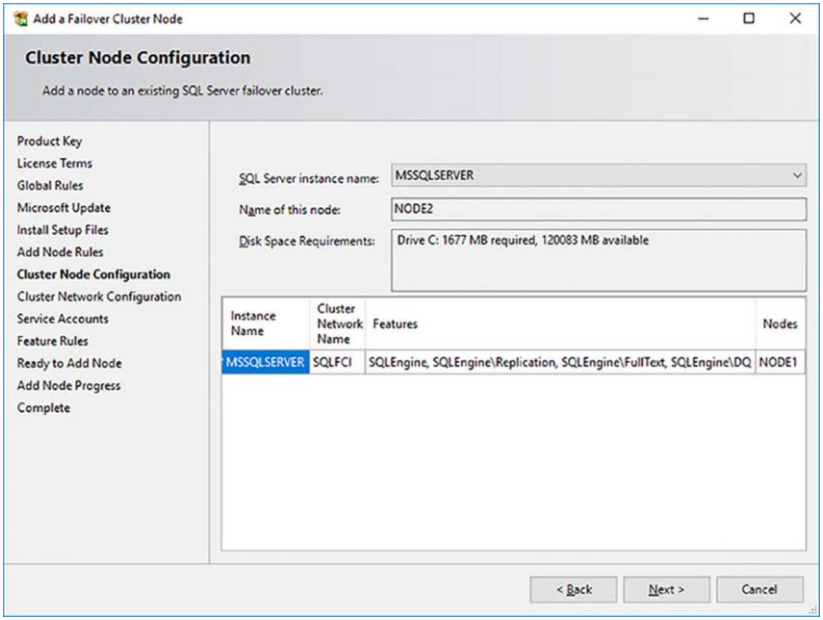

FIGURE 4-67 SQL Server FCI setup cluster node configuration

8. In the Cluster Network Configuration page the installer shows you details of the SQL Server FCI's network configuration Click on the Next button.
9. In the Service Accounts page provide the same passwords for the credentials configured for Node 1 and click on the Next button.
   - You should ensure that you have configured the Grant Perform Volume Maintenance Task Privilege To SQL Server Database Engine Service all Nodes of the failover cluster.
10. The Feature Rules page, shown in Figure 4-197, checks to see if there are any blocking processes for configuring the SQL Server FCI. Click on the Next button to proceed to the next step.
11. In the Ready To Add Node page review what will be installed and click on the Install button to engage the completion of the SQL Server FCI installation.
12. Save the setup log for support reasons and click on the Close button.

In general, there is nothing further to configure after you create your SQL Server FCI. However, you should familiarize yourself with the failover cluster and SQL Server FCI, especially if they have been deployed using new versions of Windows Server and SQL Server.

Figure 4-68 shows the SQL Server FCI that you have created. The bottom pane shows all the resources of the SQL Server FCI.

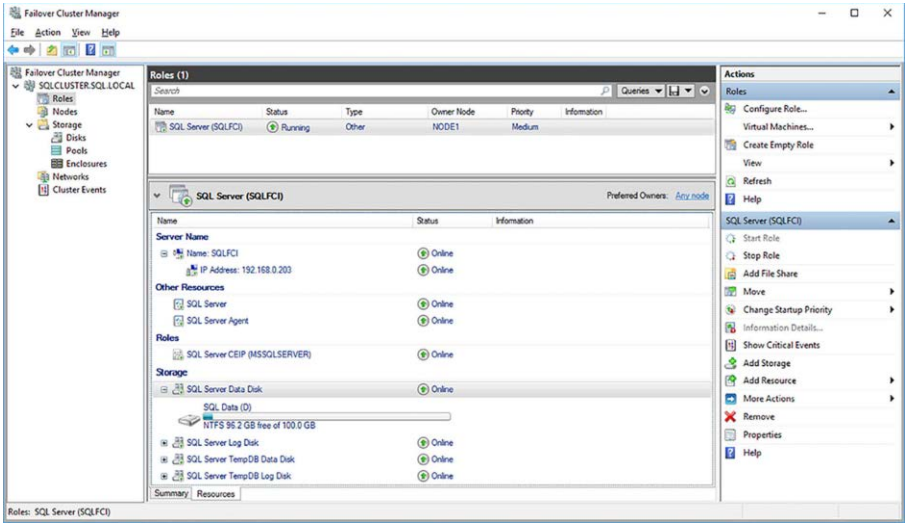

FIGURE 4-68 SQL Server FCI resources

To see the dependencies between these resources, perform the following steps:

1. Open Failover Cluster Manager.
2. Connect to your failover cluster.
3. Click on the Roles folder.
4. Right-click on your SQL Server FCI and click on the More Actions menu option.
5. Select the Show Dependency Report.
6. The SQL Server FCI Dependency Report will be shown in a web browser. Scroll down the report until you see the graphical representation of the dependencies between the various cluster resource of your SQL Server FCI, as shown in Figure 4-69. Note, for example, how the SQL Server Agent can't be brought online until the SQL Server Database Engine is online. Also note how all the physical disks have to be brought online with the Network Name before SQL Server can be started.

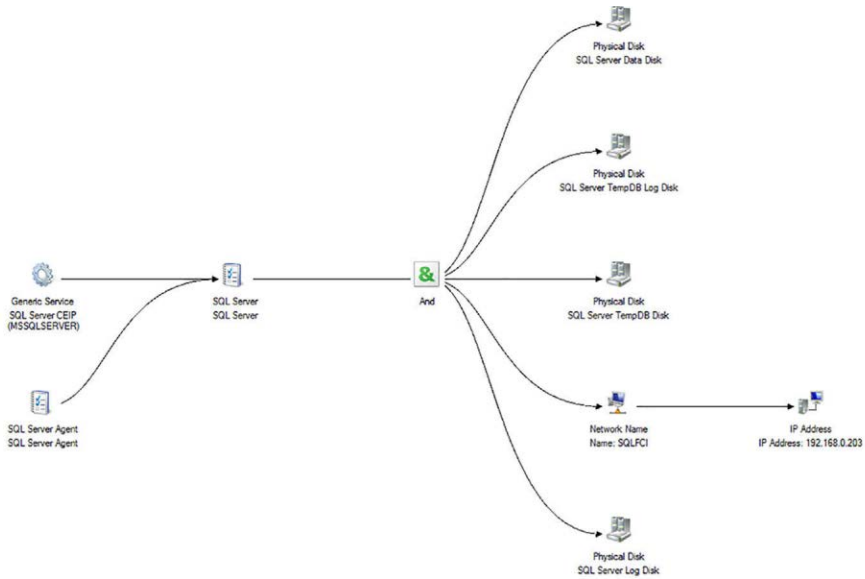

FIGURE 4-69 SQL Server FCI dependency report

You might have noticed from the above figures above that there is a new resource that gets installed with a SQL Server 2016 FCI, the SQL Server CEIP. Figure 4-70 shows the SQL Server CEIP cluster resource properties.

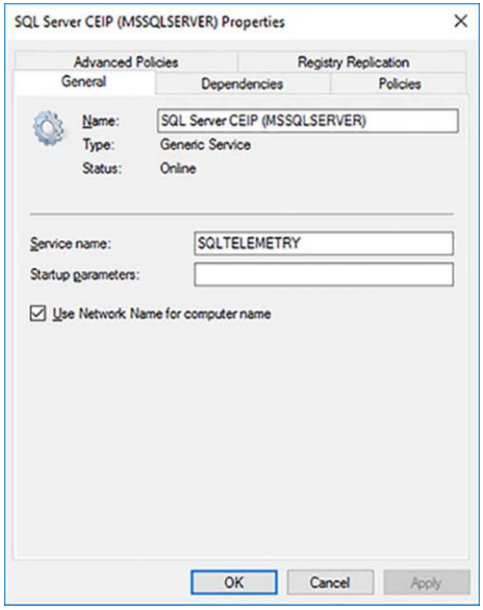

FIGURE 4-70 SQL Server CEIP resource properties

The SQL Server CEIP is the telemetry service that gets installed now by default with SQL Server 2016 and by default automatically transmits information about your installation experience, as well as other usage and performance, to Microsoft.

MORE INFO **SQL SERVER TELEMETRY SERVICE**

For more information about how to configure SQL Server 2016 to send feedback to Microsoft visit: *https://support.microsoft.com/en-us/help/3153756/how-to-configure-sql-server-2016-to-send-feedback-to-microsoft.*

Configure Quorum Configuration

Quorum in a failover cluster is how the various elements of your WSFC vote to decide whether a SQL Server FCI can be started or failed over. The concept of quorum in your WSFC is critical to the functioning of your SQL Server FCIs. Without quorum, the WSFC will go offline as a precautionary measure and your SQL Server FCIs will also be taken offline.

The quorum configuration controls what different elements can participate in the decision as to whether a WSFC can form quorum. Typically, all Nodes in your failover cluster will have a vote in the quorum. You can add additional quorum witnesses to help form quorum and avoid the "split brain" problem, where there is not a majority of votes formed. In general, you want to have an odd number of voters in your WSFC configuration.

Figure 4-71 shows how you can control what Nodes of your failover cluster can participate in the quorum voting. In certain cases, as we saw with Availability Groups, you might not want a Node to have a vote in your quorum. With a Node witness, each Node has the cluster database located locally. When a change is made to the failover cluster is it considered committed when it has been applied to the local cluster database on behalf of the Nodes (rounding down) plus one.

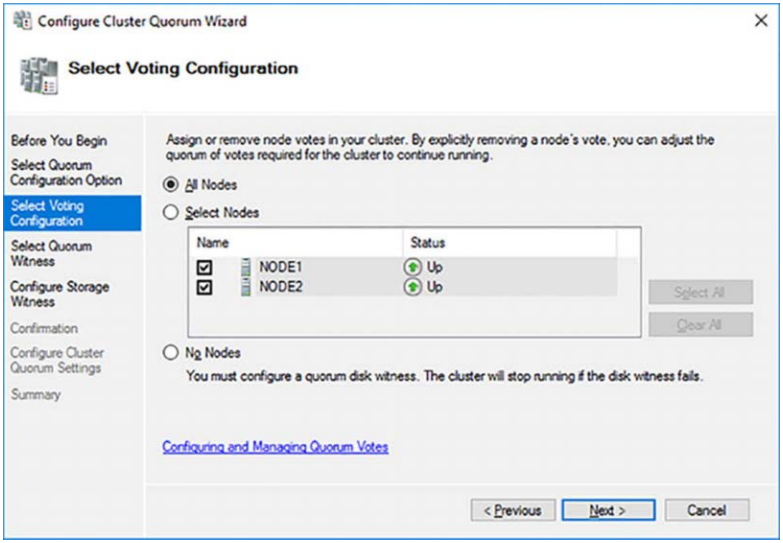

FIGURE 4-71 Configuration of voting nodes for quorum

You can add additional witness voters to the Node voters in your WSFC. This can help ensure that you have an odd number of voters in your quorum configuration. It also allows you to correctly place the witness in your existing infrastructure.

Generally, you should always configure a witness. Since Windows Server 2012 failover clustering has supported dynamic quorum, and thus dynamic witnesses. Dynamic quorum modifies the vote allocation to nodes dynamically in your failover cluster, as circumstances change, as in the case of 2 nodes in a 5 node failover cluster being shut down. With a dynamic witness, if there is an odd number of votes, the quorum witness does not have a vote. If there is an even number of votes, the quorum witness has a vote.

If you are using shared storage, you can take advantage of a disk witness. A disk witness is a dedicated LUN that stores a copy of the cluster database. Figure 4-72 shows the configuration for the disk witness. Always try to use a disk witness over other witness in the case where you are using shared storage in your failover cluster, as it is more robust than other types of witnesses. Remember that a disk witness LUN does not require a drive letter. A disk witness only needs 512MB of disk storage.

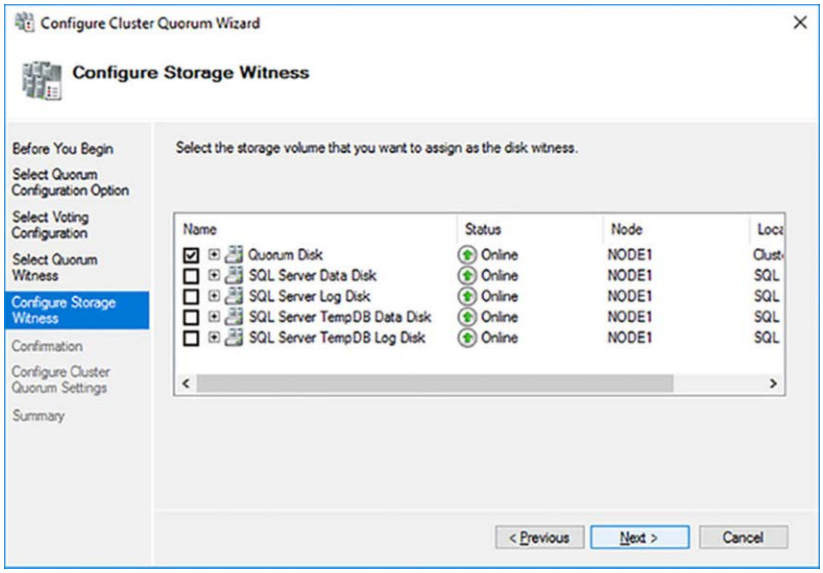

FIGURE 4-72 Configuration of disk witness in WSFC

Figure 4-73 shows the file share witness option. In the case of a file share witness the cluster database is not stored there. The file share witness only keeps track of which Node has the most updated cluster database in the witness.log file. This can lead to a scenario where only a single Node and the file share witness survive, but the failover cluster will not be able to come online if the surviving node does not have the most up to date version of the cluster database because this would cause a "partition in time." That is why the disk witness was recommended over the file share witness. You should use the file share witness when you do not have shared storage or where you have a multisite cluster with replicated storage.

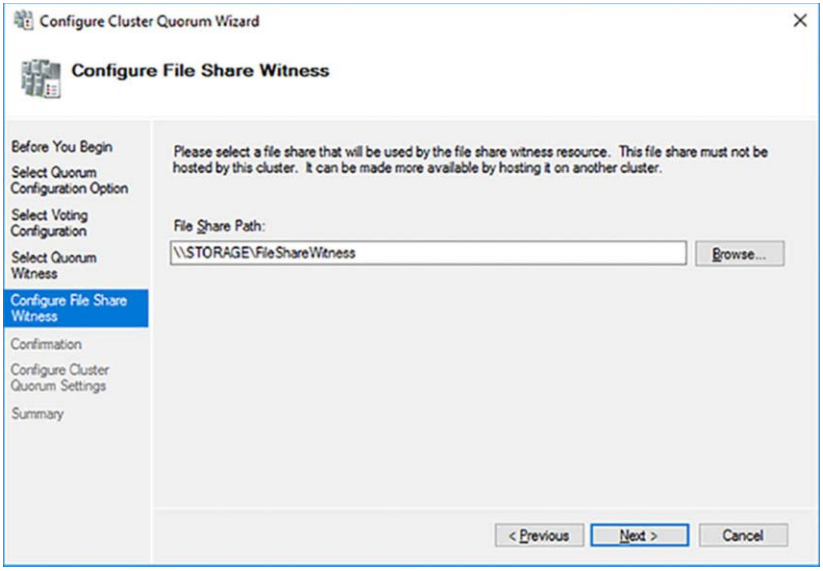

FIGURE 4-73 Configuration of file share witness in WSFC

Figure 4-74 shows the cloud witness, which was added with Windows Server 2016. It is fundamentally a file share witness, except that it is hosted in Microsoft Azure. Its primary use case is where you have two data centers and ideally need to place the witness in a third data center.

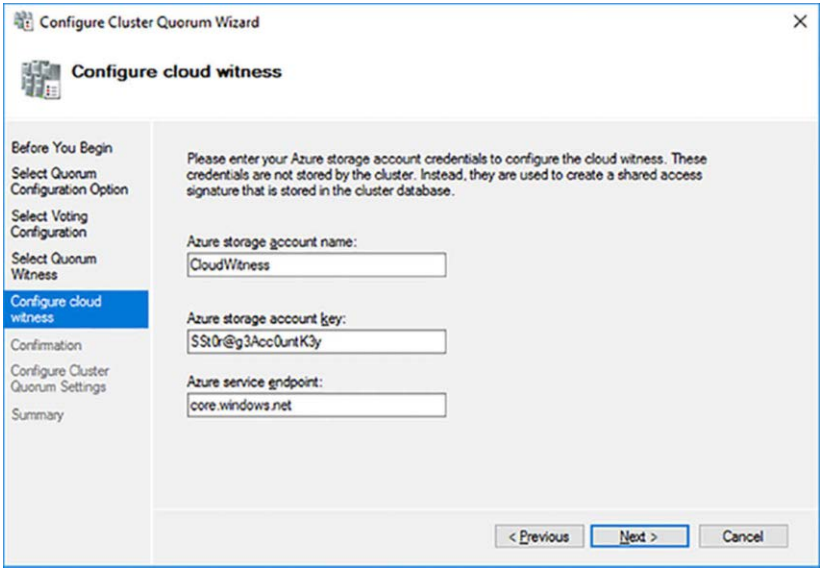

FIGURE 4-74 Configuration of cloud witness in WSFC

Manage Shared Disks

Disks (LUNs) attached to a failover cluster work differently from disks attached to a stand-alone server environment. A number of health monitoring checks are performed on a failover cluster managed disks. If any of these checks fail the WSFC will assume there is a problem and take appropriate action, including:

- Try to restart the resources and mount the disk on same node.
- Assume failover ownership of the disk.
- Try to bring the disk online on another Node.

The following file system level checks are performed on disks managed by WSFC:

- **LooksAlive** A quick check is performed every 5 seconds to verify the disk is still available.
- **IsAlive** A complete check is performed every 60 seconds to verify the disk and the file system can be accessed.

Additionally, the following device level checks are performed by the Clusdisk.sys driver:

- **SCSI Reserve** A SCSI Reserve command is sent to the LUN every 3 seconds to ensure that only the owning node has ownership and can access the disk.
- **Private Sector** Perform a read/write operation to sector 12 of the LUN every 3 seconds to ensure that the device is writable.

Sometimes you need to perform certain administrative or maintenance tasks on your clustered disks that require exclusive access to the disk, such as with the CHKDSK /F or FORMAT operations. In such cases, you do not the health monitoring checks to fail and trigger a failover.

To perform such administrative or maintenance tasks on your failover cluster's shared disks you first need to place the disk into maintenance mode. This can be done in Failover Cluster Manager by right clicking on the disk, selecting More Actions and then Turn On Maintenance Mode.

Configure Cluster Shared Volumes

Clustered Shared Volumes (CSV) is a new clustered file system in Windows Server that is a layer of abstraction above the NTFS file system in a WSFC environment. It allows all Nodes in the failover cluster to read and write to the CSV volume. CSV leverages the investments Microsoft have made in SMB 3.0, such as SMB Direct and SMB Multichannel.

SQL Server 2014 was the first version of SQL Server to support CSVs. However, CSVs are not commonly deployed with SQL Server in the industry. This poor adoption is mostly like due to a lack of awareness in the industry of CSV and its related benefits.

The Cluster Shared Volume architecture, shown in Figure 4-75, contains the following elements:

- **Coordinator Node** The Coordinator node is the node of your failover cluster on which the NTFS volume is mounted. All meta data operations from the other nodes

in your failover cluster are orchestrated through this coordinator node using SMB 3.0. Meta data operations in SQL Server include opening and closing a database, creating a database, and auto-growing a database. Such meta data operations are relatively rare.

- **CSV Proxy File System** The CSV Proxy File System is mounted on all nodes of the failover cluster. All read and write operations are sent directly through these proxies to the shared storage. This direct I/O is not even hitting the NTFS stack. If a Node cannot communicate directly to the shared storage it can communicate with the CSV Proxy File System using SMB 3.0 at the block level.
- **CSVFS** The Clustered Share Volume File System (CSVFS) is the clustered file system that spans all nodes of the failover cluster. It is effectively the layer of abstraction that sits on top of the NTFS file system.
- **NTFS Stack** The NTFS stack is used for all meta data operations to maintain consistency at the file system level.

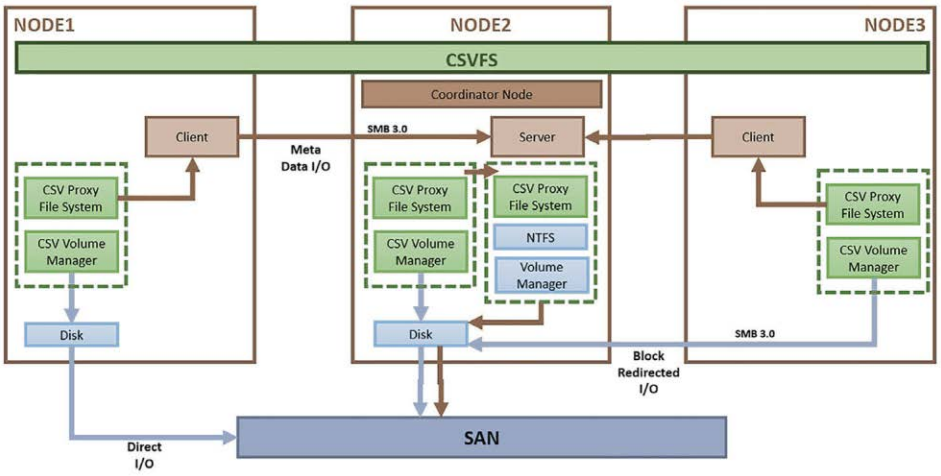

FIGURE 4-75 Cluster Share Volumes architecture

The benefits of CSV include:

- Faster failover times because there are no physical disks that need to be unmounted/mounted by the WSFC.
- Improved resilience in the case a data path fails. A Node is now able to redirect its block level I/O to the coordinator node. With the benefits of SMB 3.0, including SMB multichannel and SMB Direct (RDMA), there should be no/minimal performance impact.
- Your failover cluster no longer relies upon drive letters. You can only have as many cluster disks as the alphabet allows (24 in most cases). In the case of CSVs you are no longer replying on drive letters.

- Zero downtime with CHKDSK operations. Effectively you can provide disk repairs without any SQL Server down time.
- Easier administration as you are able to manage the underlying storage from any node. CSVFS provides the same abstraction layer across all nodes of the failover cluster.

The following steps show you have to implement CSVs in your SQL Server FCI.

1. Log into your storage server as administrator.
2. Provision another 2 iSCSI virtual disks as your iSCSI targets.
3. Log into Node 1 of the failover cluster as an administrator.
4. Open Disk Management.
5. Online, initialize and format the two new disks as NTFS volumes.
6. Open Failover Cluster Manager and connect to your failover cluster.
7. Right-click on the Disks folder and select the Add Disk option.
8. Select both new disks in the Add Disks To A Cluster dialog box and click on the OK button.
9. Rename both new cluster disks to something more meaningful.
10. Convert the cluster disks to Cluster Shared Volumes by right clicking on each cluster disk and selecting the Add To Cluster Shared Volumes option.
11. Confirm that the disks are not Cluster Shared Volumes and the they are using the CSVFS filesystem, as shown in Figure 4-76.

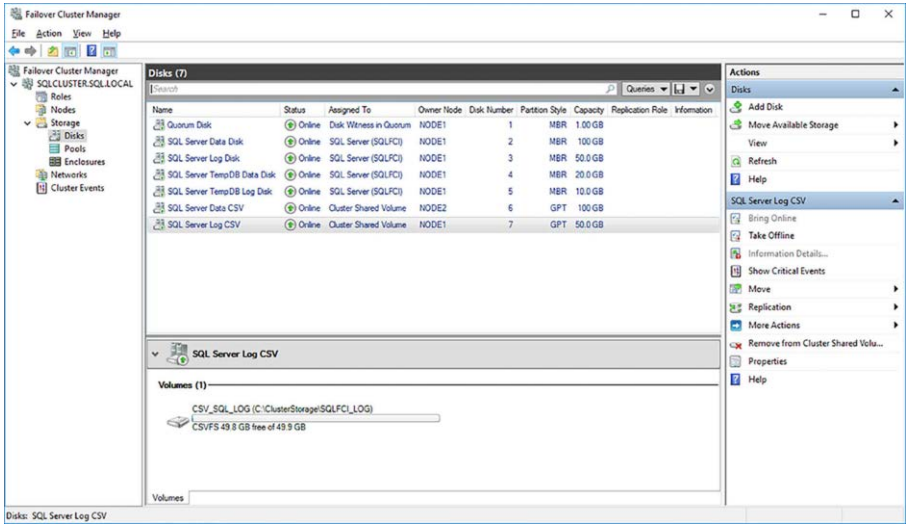

FIGURE 4-76 Cluster Shared Volumes and CSVFS filesystem

12. Open File Explorer and navigate to the C:\ClusterStorage root CSV folder as shown in Figure 4-77.

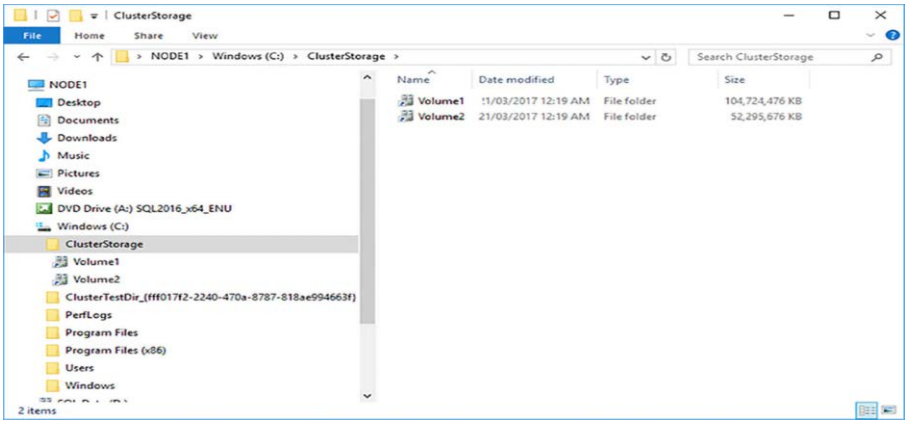

FIGURE 4-77 C:\ClusterStorage root CSV location

13. Rename the two volume folders to something more meaningful.
14. Create a subdirectory under both mount points for the SQL Server FCI to store the database files.
15. Open SQL Server Management Studio and connect to the SQLFCI instance.
16. Create a new database using the CSV database paths, as shown in Figure 4-78.

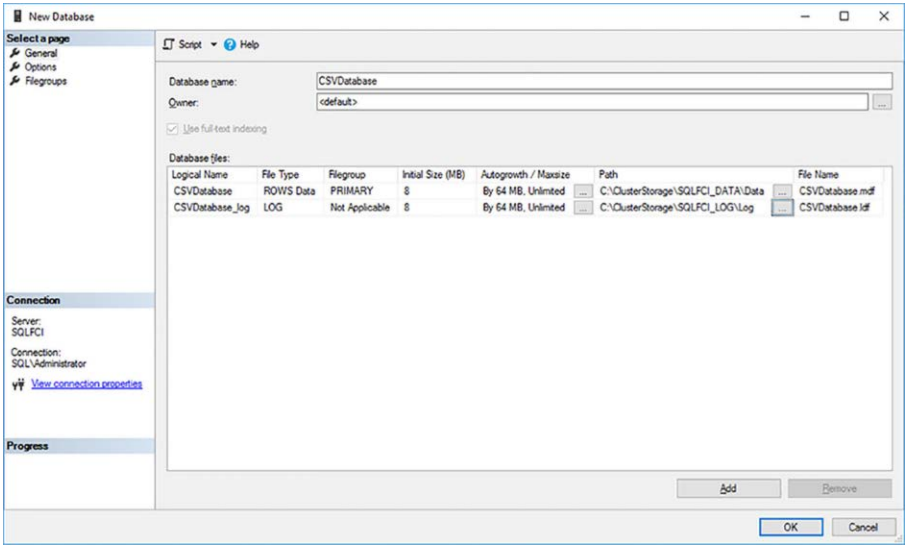

FIGURE 4-78 CSV database paths for SQL Server FCI

17. Switch to File Explorer and confirm the database files have been created in the CSV folder namespace.
18. Log into Node 2 of your failover cluster.
19. Open File Explorer and navigate to the same directory location used in Step 16.
20. Confirm you can see the database files there as well.
21. Switch back to Node 1.
22. Switch back to Failover Cluster Manager.
23. Generate a new Dependency Report.
24. Confirm you cannot see the CSVs in the dependencies, unlike the physical disks.

Consider using CSVs in your next SQL Server 2016/2017 failover cluster solution because they offer a number of advantages over traditionally deployed shared disks.

Thought experiment

In this thought experiment, demonstrate your skills and knowledge of the topics covered in this chapter. You can find answers to this thought experiment in the next section.

You work as a Database Administrator for World Wide Importers. You need to design a disaster recovery and high availability strategy for your multi-database solution that is used internally. Your company has a primary office in Sydney and another office in Wagga Wagga.

The multi-database solution has the following characteristics:

- The main OLTP database is 400GB in size
- There is a 200GB database which is used for auditing and logging records
- There are 5 more databases that are used. They are all under 50GB in size.
- All databases currently use the full recovery model.
- All databases are hosted on a single SQL Server instance
- OLAP and real-time reports are impacting performance of the OLTP transactions

You have an existing 2 Node failover cluster based in Sydney that hosts a vendor database solution. This database solution does not support Availability Groups.

Management has asked you to solve the following business problems:

- The business requires a high availability solution that supports automatic failover.

- The databases should be highly available in Sydney's data center.
- If that data center in Sydney fails the disaster recovery solution should have Wagga Wagga with an RTO of 2 hours and RPO of 15 minutes.
- Management wants to reduce the impact of running reports on the OLTP transactions. Analysts in Wagga Wagga want to report off a "snapshot" of the database at close of business on the previous day.

Question 1

Management wants you to create a high-availability solution strategy for the multi-database solution that meets their requirements. What high availability solution should you use:

1. Create an Availability Group with 4 replicas:
   - 3 replicas in Sydney
   - 2 synchronous replicas in Sydney will be used as failover partners
   - 1 readable synchronous replica in Sydney for OLAP reporting
   - 1 asynchronous secondary replica in Wagga Wagga
2. Create an Distributed Availability Group with 4 replicas:
   - 3 replicas in Sydney
   - 2 synchronous replicas in Sydney will be used as failover partners
   - 1 readable synchronous replica in Sydney for OLAP reporting
   - 1 asynchronous secondary replica in Wagga Wagga
3. Create a 3 node failover cluster:
   - 2 nodes will be based in Sydney
   - 1 node will be based in Wagga Wagga
4. Create an Availability Group in Sydney. Use log shipping between Sydney and Wagga Wagga:
   - 3 replicas in Sydney
   - 2 synchronous replicas in Sydney will be used as failover partners
   - 1 readable synchronous replica in Sydney for OLAP reporting
   - Perform log backups every 15 minutes

Question 2

Management wants to extend the failover cluster to Wagga Wagga. They plan to add two more nodes to the cluster in Wagga Wagga. What quorum configuration should you use?

1. Use a node majority quorum with no witness.
2. Use a node majority quorum with a file share witness.
3. Use a node majority with a cloud witness
4. Use a node majority with a disk witness.

Question 3

After implementing an Availability Group for your 400GB OLTP database you notice that the physical disk that has been provisioned for the database's MDF file is running out of space. Management has provisioned a new 1TB PCIe SSD for the database on all replicas. You need to ensure that the database does not run out of space with minimal downtime while maintaining high availability. What should you do?

1. Perform the following actions:
   - Take the database offline
   - Detach the database
   - Move the MDF files to the new storage
   - Attach the database
2. Perform the following actions:
   - Suspend Availability Group
   - Backup the database
   - Drop the database
   - Restore database to new storage
   - Resume Availability Group
   - Wait for secondary replicas to replicate your changes
3. Perform the following actions:
   - Remove the 400GB database from the Availability Group
   - Detach the database
   - Move the database file to the new storage
   - Attach the database
   - Drop the 400GB database from all secondary replicas
   - Add the database back to the Availability Group with the Direct Seeding option
4. Perform the following actions:
   - Add a new secondary file for the database on the new storage

Thought experiment answers

This section contains the solution to the thought experiment. Each answer explains why the answer choice is correct.

Question 1

1. **Correct answer: D**
   - A. **Incorrect:** You can't meet your snapshot reporting with a replica in Wagga Wagga.
   - B. **Incorrect:** You can't meet your snapshot reporting with a replica in Wagga Wagga. A DAG with only one replica in Wagga Wagga is effectively the same as option A.
   - C. **Incorrect:** You cannot scale out or offload reporting using failover clustering.
   - D. **Correct:** Availability Group in Sydney provides high availability and offloads reporting. Log shipping provides the snapshot reporting.

Question 2

2. **Correct answer: C**
   - A. **Incorrect:** With only 2 nodes at each site the cluster might shut down if the WAN link goes down and a node in Sydney fails.
   - B. **Incorrect:** A fileshare witness in either data center might prevent quorum if the WAN link has problems.
   - C. **Correct:** A cloud witness is designed for such scenarios where you do not have a third data center.
   - D. **Incorrect:** A disk witness in either data center might prevent quorum if the WAN link has problems

Question 3

3. **Correct answer: D**
   - A. **Incorrect:** You cannot take a database offline when it is part of an Availability Group.
   - B. **Correct:** You cannot drop a database when it is part of an Availability Group.
   - C. **Incorrect:** This will not meet your high availability and time constraint requirements.
   - D. **Correct:** This will allow the database to use the new space while maintaining high availability and incur no downtime.

Chapter summary

- High availability is not equivalent to disaster recovery.
- Log Shipping is not a high availability technology.
- Log Shipping supports multiple secondary servers.
- With Log Shipping, users cannot access the secondary database when a log backup is being restored.
- Failover clustering and Availability Groups rely on the Windows Server Failover Cluster (WSFC) feature.
- Failover clustering and Availability Groups support automatic failover.
- The scope of protection in Availability Groups is at the database level.
- Availability Groups support three failover partners with SQL Server 2016.
- Each replica in an Availability Group maintains its own version of the database.
- Availability Groups support asynchronous and synchronous communication between the replicas.
- With an Availability Groups you can only perform a manual fail over to a synchronous replica.
- A forced failover in an Availability Group to an asynchronous replica can result in data loss.
- Availability Groups support readable secondaries.
- Availability Groups are the only high-availability technology that allows you to scale out your database solution.
- You can offload reports, read operations, database consistency checks and backup operations to a readable secondary.
- SQL Server 2016 Standard Edition supports Basic Availability Groups, which is intended to replace Database Mirroring.
- Distributed Availability Groups are designed to be used between data centers where you want to minimize the network usage across the WAN link between your data centers.
- SQL Server 2016 provides limited support for DTC in Availability Groups.
- Availability Groups are more complicated to maintain and administer as they do not automatically synchronize logins, SQL Server Agent jobs and other external database dependencies between replicas.
- You can install a number of SQL Server Failover cluster instances on a Windows failover cluster.
- The scope of protection in a failover cluster is at the instance level.
- WSFC no longer require a domain.

- Failover clustering uses shared storage which represents a single point of failure.
- Failover clustering will work with SMB 3.0 file shares as a location for your database files.
- Cluster Shared Volumes (CSVs) represent a new clustered file system in Windows Server.
- SQL Server 2014 added support for (CSVs).
- CSV remove the dependency on physical disks, which reduces the fail over time in a failover cluster.

Index

A

actions 206
Activity Monitor 182–183, 188
administrative accounts
  Azure SQL Database 49
Advanced Encryption Standard New Instructions (AES-NI) 21
alerts
  custom 245
  SQL Agent 243–245
ALLOW_SNAPSHOT_ISOLATION isolation level 186
ALTER DATABASE statement 93, 104
ALTER INDEX DISABLE statement 225
ALTER INDEX REBUILD statement 220, 226
ALTER INDEX REORGANIZE statement 220, 225–226
ALTER INDEX statement 220
ALTER TABLE REBUILD statement 220, 226
Always Encrypted (AE) 9–20
application roles 36
asymmetric keys 3, 4, 27
  encrypting backups with 25
asynchronous commit mode 289, 291–292
auditing
  Azure SQL Database 57–61
  blob 59–60
  configuration 50–61
  database audit specification 51
  implementing 54–55
  management of 56–57
  policies 57–58
  querying SQL Server audit log 55–56
  server audit specification 51
  SQL Server 51–58
audit logs 55–56, 57, 60–61
authentication
  Azure Active Directory 49
  Azure SQL Database 49
  SQL 29, 49
  Windows 29
authenticator parameter 6
Auto Create Incremental Statistics 232
Auto Create Statistics 232
automatic failover 268, 293, 327
Auto Update Statistics 231–233
Auto Update Statistics Asynchronously 232
availability 266–267. *See also* high availability
Availability Groups 269
  architecture 287–298
  automatic failover in 293
  backup preferences 310–311
  Basic 293–294
  configuration 307–319
  creating 304–318
  Distributed 331–332
  enhancements to 296–297
  failover management 327–330
  implementing 287–332
  Initial Catalog connection 324
  listener preferences 311–312
  listeners 288
  log stream compression 293–294
  monitoring 325–326
  pros and cons of 293–295
  quorum configuration 319–322
  readable secondary replicas 297–298
  read-only routing configuration 322–325
  synchronization modes 289–293
  use cases 296–297
  Windows Server Failover Clustering 298–304
avg_page_space_used_in_percent 219

Azure
  backing up databases to 85–87
Azure Active Directory Authentication 49
Azure AD admin 49
Azure Portal 60
Azure SQL Database
  audit configuration 57–61
  user options configuration 49–50
Azure Storage Explorer 61
Azure Trust Center 57

B

backup automation 106–131
  maintenance plan 116–131
BACKUP command 107
backup compression 21, 68
backup destination 71
backup devices 71, 72–73
backup encryption 25–26
backup logs 74
BACKUP LOG statement 95
backups
  alerts for failed 131–140
  Availability Groups 310–311
  differential 78–79, 94
  file 80
  full 74–79
  log 79–80, 90–105
  mirrored 83
  options 81–83
  partial 81
  tail-log 97–100, 142
  to Azure 85–87
  to nul device 306
  types of 67–68, 70–71
  VLDBs 87–90
backup set 71
BACKUP statement 81–83
backup strategy 66–140
  backup automation configuration 106–131
  backup operations 67–69, 70–90
  designing 66–70
  evaluating potential 69–70
  factors in 68–69
  failed backup alerts 131–140
Basic Availability Groups 293–294
blob auditing 59–60
Blocked By column filter 183
blocked process reports 184
blocking activity 183–186
block predicates 42
bottlenecks 254
buffer pool extension (BPE) 22
Bulk Change Map (BCM) 92
BULK_LOGGED recovery models 91, 93–94

C

cached plans 215–216
certificates 3, 6, 8
  backing up 24, 26
Change Data Capture (CDC) 12, 50
channels 205
Check Database Integrity task 117
CHECKSUM option 81
checksum page verification 154
Clean Up History task 117
Client Access Point (CAP) 350
cloud witness 370
CLR. *See* Common Language Runtime (CLR)
Clusdisk.sys driver 371
Clustered Shared Volumes (CSV) 371–375
Clustered Share Volume File System (CSVFS) 372
Column Encryption Key (CEK) 12
Column Encryption Setting=Enabled; option 10
column-level encryption 2–9
Column Master Key (CMK) 10, 12
columns
  encrypted 8
  helper 4
columnstore indexes 225–226
Common Criteria 51
Compatibility Level 232
compression
  backup 21, 68
  log stream 293–294
  VLDBs 87
COMPRESSION option 82
Computer Name Object (CNO) 351
configuration
  auditing 50–61
  Availability Groups 307–319
  backup automation 106–131

Clustered Shared Volumes 371–375
data access 28–50
database mail 235–241
data collector 188–196
dynamic data masking 47–48
encryption 2–28
Extended Events 205–214
failed backup alerts 131–140
failover clustering 338–367
log shipping 275–284, 286
maintenance plan 116–131
operators 131–132
policy based management 247–253
Query Store 198–202
quorum 319–322, 367–369
read-only routing 322–325
row-level security 41–46
Windows clustering 298–304
Configure Management Data Warehouse Wizard 190
connections
encryption of 26–27
CONTINUE_AFTER_ERROR option 81
Coordinator node 371–372
COPY_ONLY option 81
CREATE INDEX statement 220
cross-database ownership chaining 32
CSV. *See* Cluster Shared Volume
CSV Proxy File System 372
CTEs. *See* common table expressions (CTEs)
current sessions
monitoring 180–183
custom roles 36–37

D

DAGs. *See* Distributed Availability Groups
damaged databases
tail-log backups on 100
data
querying. *See* queries
data access 1
Azure SQL Database 49–50
configuration 28–50
custom roles 36–37
database object permissions 38–41
dynamic data masking 47–48
row-level security 41–46
user creation 29–35
database activity monitoring 180–196
current sessions 180–183
data collector 188–196
identifying sessions causing blocking activity 183–186
identifying sessions consuming tempdb resources 186–188
database audit specification 51
database backups. *See* backups
database checkpoint 74
database consistency checks 163–167
Database Console Command (DBCC) 164–167, 170
database corruption
identifying 167–169
recovery from 169–173
database encryption key (DEK) 22
database files 68
database integrity
consistency checks 163–167
database corruption 167–173
management of 163–173
database mail
components of 236–237
configuration 235–241
logs 241
database mail accounts 236, 238
database mail profiles 236
Database Master Key (DMK) 3
database mirroring 269, 293
Database Properties page 94–95
database recovery models
configuration 91–95
database roles 29, 32, 34–35
Azure SQL Database 49–50
user-defined 36
databases
emergency repairs 171–173
indexes 218–226
maintenance plan for 116–131
number of 268
partial availability 89
restoring 141–163
size 268
very large 87–90
database size 68
database snapshots
performing 84–85
reverting 148
database user mappings 33–34

data collector
  configuration 188–196
  information collected by 189
  role-based security 189–190
data compression 87
Data Definition Language (DDL) 50
data encryption. *See* encryption
DATA_FLUCH_INTERVAL_SECONDS option 198
data loss 1, 65, 67, 92
Data Manipulation Language (DML) 50
data modifications 68
data protection API (DPAPI) 3
data volumes
  identifying available space on 253
DAX volume 105
DBCC CHECKCONSTRAINTS operation 170
DBCC CHECKDB command 164–166, 169
DBCC CHECKFILEGROUP command 166
DBCC CHECKTABLE command 166
DBCC commands 164–167
DBCC error states 171
DBCC OPENTRAN command 102
DBCC SHOW_STATISTICS command 228
dbmanager role 49
dc_admin role 189
dc_operator role 189
dc_proxy role 189
DEADLOCK_GRAPH trace event 185
deadlocks 184–186, 217–218
default system_health session 185
DEK. *See* database encryption key
delayed durability 103–104
DENY statement 39
DESCRIPTION option 82
DETAILED mode 219
deterministic encryption 11
differential backups 68, 70, 78–79, 94
direct seeding 313
disaster recovery
  log shipping 271–287
  solution design 270
  vs. high availability 265
Disaster Recovery Plan (DRP) 65
  backup operations 67–69
  documentation 142
  recovery objectives, defining 67
Disk Usage data collector 192–194
disk witness 368
Distributed Availability Groups (DAGs) 331–332
DMK. *See* Database Master Key
downtime
  planned 267
  unplanned 267–268
DROP INDEX statement 220, 225
dynamic data masking (DDM) 47–48
dynamic management views (DMVs) 180–182, 214–215, 219, 221–222, 223–224

E

EKM. *See* Extensible Key Management
emergency repairs 171–173
encryption 1
  Always Encrypted 9–20
  authenticators 6
  backup 25–26
  column-level 2–9
  configuration 2–28
  deterministic 11
  for connections 26–27
  hierarchy 3
  layers 2
  randomized 11
  system functions 4
  Transparent Database Encryption 21–25
  troubleshooting errors 27–28
ENCRYPTION option 82
error logs 27
error messages 168
events 205–206
event tracing for Windows (ETW) framework 205
ExecuteSql() function 251–252
Execute SQL Server Agent Job task 117
Execute T-SQL Statement Task 117
execution plans 203
  identifying problematic 214–216
EXPIREDATE option 82
Extended Events 179, 185, 205–214
  architecture 205–207, 215
  engine 207
  session creation 207–214
  troubleshooting server health using 217–218
  use cases 205
Extensible Key Management (EKM) 2, 9

F

failed backup alerts 131–140
failover clustering 269, 287, 294, 298–304
  architecture 333–338
  automatic failure in 335
  Client Access Point 350
  Clustered Shared Volumes 371–375
  Computer Name Object 351
  configuration 338–367
  failover in 335–336
  implementing 332–374
  installation of SQL Server FCI 362–367
  pros and cons of 336–337
  quorum configuration 367–369
  shared disk management 371
  use cases 338
  validation tests 347–350
Failover Cluster Instance (FCI) 332–334, 354–367
failover management 327–330
failover partner 288
failover speed 268
failover types 327
failure actions 246–247
Fiber Channel (FC) 333
file backups 70, 80
filegroups
  restoring 161–162
FILELISTONLY statement 143
file share witness 369–370
filtered statistics 235
filter predicates 42
firewalls 49
fixed database roles 34
fixed server roles 32–33
forced failover 327, 330
FORMAT option 82
fragmentation, index 218–221
full backups 68, 70, 74–79
FULL recovery models 91–92

G

GRANT statement 39

H

Hardware Security Module (HSM) 2
HEADERONLY statement 143
heaps 220
helper columns 4
high availability
  Availability Groups 287–332
  failover clustering for 287, 298–304, 332–374
  log shipping 271–287
  number of nines 266–267
  Proof-of-Concept 269
  solution design 266–269
  vs. disaster recovery 265
High-Availability (HA) technologies 65
high availability technologies
  SQL Server 269
HSM. *See* Hardware Security Module

I

incremental backups 68
incremental statistics 235
INCREMENTAL_STATS 234
indexes
  columnstore 225–226
  dropping and recreating 220
  identify and repair fragmentation 218–221
  managing 218–226
  missing 216, 221–223
  rebuilding 220, 229
  reorganization 220
  scanning modes 219
  underutilized 223–225
Initial Catalog 324
INIT option 82
in-memory tables
  consistency checks 164, 171
INTERVAL_LENGTH_MINUTES option 198
iSCSI protocol 333

K

keywords 206
KILL statement 102–103

L

LABELONLY statement 143
large object (LOB) storage 186
LIMITED mode 219

live query statistics 216
log backup chains 94, 96–97
log backups 70, 79–80, 90–105
log cache 103
loginmanager role 50
logins
  creating 29–35
  mapping to users 33–34
  roles for 31–35
  SQL authentication 29
  Windows authentication 29
LOG_REUSE_WAIT_DESC column 101
log sequence number (LSN) 71, 74
log shipping 269
  architecture 271–274
  configuration 275–284, 286
  customizing 284
  implementing 271–287
  monitoring 284–287
  pros and cons of 273–274
  use cases 274
log stream compression 293–294

M

Maintenance Cleanup Task 117
maintenance plan
  configuration 116–131
  notifications 133–137
Maintenance Plan Wizard 116–131
management data warehouse (MDW) 188, 190
manual failover 327–330
maps 206
marked transactions 160–161
masking data 47–48
MAX_STORAGE_SIZE_MB option 198
mdw_admin role 190
mdw_reader role 190
mdw_writer role 190
MEDIADESCRIPTION option 83
media family 71
media header 72
MEDIANAME option 83
media set 71, 73
memory
  NVDIMM 104–105
memory buffers 198
memory-optimized filegroup 149
Microsoft Azure Blob Storage 85–86
Microsoft Azure Key Vault 9
Microsoft Distributed Transaction Coordinator (MSDTC) 295
Microsoft SQL Server Backup to Microsoft Azure Tool 86
minimally logged operations 92
mirrored backups 83
MIRROR TO clause 83
missing indexes 216, 221
MOVE option 146
Multiple Active Results Sets (MARS) sessions 186

N

NAME option 83
NO_CHECKSUM option 81
NO_COMPRESSION option 82
nodes 333
NO_ENCRYPTION option 82
NOFORMAT option 82
NOINDEX option 166
NOINIT option 82
non-volatile memory (NVDIMM) 104–105
NORECOVERY option 95, 146
NOSKIP option 83
Notify Operator Task 117
NO_TRUNCATE option 95
NTFS stack 372
nul devices
  backups to 306

O

OBJECT_NAME system function 169
object permissions 38–41
offload reporting 274
On-Line Transactional Processing (OLTP) database 143–144
online-transaction processing (OLTP) environment 218
Operating System Environment (OSE) 309
OPERATION_MODE option 198
operators
  configuration 131–132
  creating 242–243
  managing 242–243
orphaned log experiment 97–99
outdated statistics 227–231

P

packages 205
page corruption detection 154
page recovery 154–156
partial availability 89
partial backups 71, 81
partial-restore sequence 149, 151
PBM. *See* policy based management
performance bottlenecks 254
performance degradation
  identifying cause of 254–259
Performance Monitor 258–259
performance problems 180
permissions 32
  custom roles for 36–37
  database object 38–41
persistent memory 104–105
PHYSICAL_ONLY option 166
piecemeal restores 148–154
planned downtime 267
point-in-time recovery 69, 157–161
policy based management (PBM) 247–253
Power BI 61
pre-defined policies 251
predicates 206
primary database 288
primary replica 288
private keys
  backing up 26
private network 334
Proof-of-Concept (POC) 269, 287
public network 334

Q

queries
  statistics 227–228
QUERY_CAPTURE_MODE option 199
Query Detail History report 195–196
query monitoring 197–218
  Extended Events 205–214
  identifying problematic execution plans 214–216
  live 216
  Query Store 197–204
  troubleshooting server health 217–218
Query Statistics History report 194–195
Query Store 179, 188, 197–204, 214, 257
  analyzing and managing 204
  configuration 198–202
  execution plans 203
  memory buffers 198
  schema 202
  system catalog views 200–202
  uses of 197–198
quorum 334
quorum configuration
  Availability Group 319–322
  failover clustering 367–369

R

randomized encryption 11
readable secondary replicas 288, 297–298, 308–309
READ_COMMITTED_SNAPSHOT isolation level 186
read-only routing 322–325
read operations
  offloading 297
Rebuild Index task 117
Recovery Level Objective (RLO) 67, 270
recovery model 268
recovery models 69
  changing 93
  default 91
  querying 94
recovery objectives 67, 69
RECOVERY option 146
Recovery Point Objective (RPO) 67, 270
Recovery Time Objective (RTO) 67, 270
Reorganize Index task 117
REPAIR_ALLOW_DATA_LOSS option 171–173
REPAIR_REBUILD repair operation 170–171
REPLACE option 146
restores, database 141–163
  automating and testing 162–163
  emergency repairs 171–173
  filegroups 161–162
  options 146–147
  page recovery 154–156
  piecemeal 148–154
  point-in-time recovery 157–161
  process of 145–148
  restore operations 145–148
  restore strategy 141–145
  reverting database snapshots 148
  testing 144

RESTORE statement 143, 145–146, 146–147, 155–156, 157
RESTORE VERIFYONLY operation 143
RETAINDAYS option 83
RETAINSDAYS option 82
REVOKE statement 39
RLS. *See* row-level security
role-based security 189–190
roles
  creating and maintaining 35
  custom 36–37
  database 32, 34–35
  server 31, 32–33, 36–37
root cause analysis
  database corruption 169
  page corruption 155
row-level security (RLS) 41–46

S

SAMPLED mode 219
scalability 268
Scale-Out File Server (SOFS) 338
SCSI Reserve command 371
secondary database 288
secondary replicas 288, 297
Secure Sockets Layer (SSL) 27
security
  auditing 50–61
  Azure SQL Database 49–50
  dynamic data masking 47–48
  role-based 189–190
  row-level 41–46
  transport layer 27
security predicates 42
sequence number 71
Server admin 49
server audit specification 51
Server Message Block (SMB) 3.0 333
server roles 31, 32–33
  user-defined 36, 36–37
Service Broker event notifications 185
Service Level Agreement (SLA) 266
Service Master Key (SMK) 3
sessions
  causing blocking activity 183–186
  consuming tempdb resources 186–188
  Extended Events 207–214
  monitoring current 180–183
Setup Data Collection Sets wizard 191–192
shared disk management 371
shared storage 333, 338
Shrink Database task 117
SIMPLE recovery models 91, 93
SIZE_BASED_CLEANUP_MODE option 199
SKIP option 83
SMK. *See* Service Master Key
sp_add_operator 242
[sp_describe_parameter_encryption] system stored procedure 10
sp_send_dbmail 240–241
sp_set_session_context 43
SQL authentication 29, 49
sqllogship.exe 283, 284
SQL Server
  auditing 50–57
  editions 269
  encryption
    Always Encrypted 9–20
    backup encryption 25–26
    configuration 2–9
    connections 26–27
    Transparent Data Encryption 21–25
  high availability technologies 269
  login creation 29–35
  log shipping 271–272
  performance condition alerts 244
SQL Server Agent 189
  alerts 243–245
  failure actions 246–247
  jobs 245
  scheduling backup through 114–116
SQL Server Configuration Manager 26
SQL Server instances 179–264
  database activity monitoring 180–196
  enabling Availability Groups for 304–317
  Failover Cluster Instance 332–333, 354–367
  indexes 218–226
  monitoring 235–259
  primary replicas 288
  query monitoring 197–218
  secondary replicas 288
  statistics management 227–235
SQL Server Managed Backup 85
SQL Server Management Studio 94–95, 114, 275

SQL Server Management Studio (SSMS) 61
SQL Server Profiler 185
STALE_QUERY_THRESHOLD_DAYS option 199
STANDBY option 95, 147
statistics
  auto update of 231–233, 234
  filtered 235
  for large tables 233–235
  incremental 235
  managing 227–235
  outdated 227–231
  updating 229
STATS option 83
STOPAT clause 157
STOPATMARK clause 157
STOP_ON_ERROR option 81
symmetric keys 3, 4
synchronous commit mode 289, 290–291
sys.dm_db_file_space_usage 186
sys.dm_db_index_usage_stats 223
sys.dm_db_missing_index_columns 221
sys.dm_db_missing_index_details 221
sys.dm_db_missing_index_groups 222
sys.dm_db_missing_index_group_stats 222
sys.dm_db_session_space_usage 187
sys.dm_db_task_space_usage 187
sys.dm_exec_cached_plans 214
sys.dm_exec_connections 180
sys.dm_exec_query_plan 214
sys.dm_exec_query_stats 214–215
sys.dm_exec_requests 181
sys.dm_exec_sessions 181
sys dm_exec_session_wait_stats 181
sys.dm_exec_sql_text 215
[sys] [dm_io_virtual_file_stats] 259
[sys].[dm_os_wait_stats] 258
sys.dm_os_waiting_tasks 181
sys.dm_plan_attributes 215
sys.dm_tran_active_snapshot_database_transactions 187
sys.dm_tran_session_transactions 181
sys.dm_tran_version_store 187
sys.fn_get_audit_file 61
sys.fn_get_audit_file system function 56
sys.indexes 224
sys.query_context_settings 200
sys.query_store_plan 200
sys.query_store_query 200
sys.query_store_query_text 201
sys.query_store_runtime_stats 201
sys.query_store_runtime_stats_interval 202
system functions
  encryption 4
system_health 217–218

T

tables
  rebuilding 220
  statistics for large 233–235
table valued function (TVF) 42
Tabular Data Stream (TDS) protocol 254
tail-log backups 71, 97–100, 142
targets 206
TDE. *See* Transparent Database Encryption
telemetry 214
tempb related errors 188
tempdb resources 186–188
tempdb system database 361
TLS. *See* Transport Layer Security
TORN_PAGE_DETECTION 154
trace events 205–214
transactional replication 269
transaction log backups 90–105
transaction log chains 96–97
transaction logs 69
  managing full 100–103
  with delayed durability 103–104
  with persistent memory 104–105
transaction queries
  killing long-running 102–103
Transact-SQL statements 81–83
Transparent Database Encryption (TDE) 21–25
Transport Layer Security (TLS) 27
troubleshooting
  encryption errors 27–28
  performance degradation 254–259
  server health 217–218
types 207

U

undocumented [fn_dblog] function 158
unplanned downtime 267–268
UPDATE STATISTICS statement 229
Update Statistics task 117
user-defined database roles 36

user-defined server roles 36, 36–37, 36–37
user options
  Azure SQL Database 49–50
users
  creating 29
  mapping logins to 33–34

V

Validate A Configuration Wizard 347–350
VERIFYONLY statement 143
very large databases (VLDBs) 268
  back up of 87–90
  configuring primary data file for 88
  creating 88
Virtual IP Address (VIP) 334
virtual network name (VNN) 334

W

wait stats 257
Windows authentication 29
Windows event logs 27
Windows Performance Monitor 257
Windows Server Failover Clustering
    (WSFC) 298–304, 334
WMI events 243–244
work files 186
work tables 186
write ahead logging (WAL) protocol 103
WSFC. *See* Windows Server Failover Clustering