

Microsoft  
Official  
Course



**DP-300T00**

Administering Relational  
Databases on Microsoft  
Azure

**DP-300T00**  
**Administering Relational**  
**Databases on Microsoft Azure**

---

## II Disclaimer

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2019 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <http://www.microsoft.com/trademarks><sup>1</sup> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

---

<sup>1</sup> <http://www.microsoft.com/trademarks>

## MICROSOFT LICENSE TERMS

### MICROSOFT INSTRUCTOR-LED COURSEWARE

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to your use of the content accompanying this agreement which includes the media on which you received it, if any. These license terms also apply to Trainer Content and any updates and supplements for the Licensed Content unless other terms accompany those items. If so, those terms apply.

**BY ACCESSING, DOWNLOADING OR USING THE LICENSED CONTENT, YOU ACCEPT THESE TERMS.  
IF YOU DO NOT ACCEPT THEM, DO NOT ACCESS, DOWNLOAD OR USE THE LICENSED CONTENT.**

**If you comply with these license terms, you have the rights below for each license you acquire.**

#### 1. DEFINITIONS.

1. "Authorized Learning Center" means a Microsoft Imagine Academy (MSIA) Program Member, Microsoft Learning Competency Member, or such other entity as Microsoft may designate from time to time.
2. "Authorized Training Session" means the instructor-led training class using Microsoft Instructor-Led Courseware conducted by a Trainer at or through an Authorized Learning Center.
3. "Classroom Device" means one (1) dedicated, secure computer that an Authorized Learning Center owns or controls that is located at an Authorized Learning Center's training facilities that meets or exceeds the hardware level specified for the particular Microsoft Instructor-Led Courseware.
4. "End User" means an individual who is (i) duly enrolled in and attending an Authorized Training Session or Private Training Session, (ii) an employee of an MPN Member (defined below), or (iii) a Microsoft full-time employee, a Microsoft Imagine Academy (MSIA) Program Member, or a Microsoft Learn for Educators – Validated Educator.
5. "Licensed Content" means the content accompanying this agreement which may include the Microsoft Instructor-Led Courseware or Trainer Content.
6. "Microsoft Certified Trainer" or "MCT" means an individual who is (i) engaged to teach a training session to End Users on behalf of an Authorized Learning Center or MPN Member, and (ii) currently certified as a Microsoft Certified Trainer under the Microsoft Certification Program.
7. "Microsoft Instructor-Led Courseware" means the Microsoft-branded instructor-led training course that educates IT professionals, developers, students at an academic institution, and other learners on Microsoft technologies. A Microsoft Instructor-Led Courseware title may be branded as MOC, Microsoft Dynamics, or Microsoft Business Group courseware.
8. "Microsoft Imagine Academy (MSIA) Program Member" means an active member of the Microsoft Imagine Academy Program.
9. "Microsoft Learn for Educators – Validated Educator" means an educator who has been validated through the Microsoft Learn for Educators program as an active educator at a college, university, community college, polytechnic or K-12 institution.
10. "Microsoft Learning Competency Member" means an active member of the Microsoft Partner Network program in good standing that currently holds the Learning Competency status.
11. "MOC" means the "Official Microsoft Learning Product" instructor-led courseware known as Microsoft Official Course that educates IT professionals, developers, students at an academic institution, and other learners on Microsoft technologies.
12. "MPN Member" means an active Microsoft Partner Network program member in good standing.

13. "Personal Device" means one (1) personal computer, device, workstation or other digital electronic device that you personally own or control that meets or exceeds the hardware level specified for the particular Microsoft Instructor-Led Courseware.
  14. "Private Training Session" means the instructor-led training classes provided by MPN Members for corporate customers to teach a predefined learning objective using Microsoft Instructor-Led Courseware. These classes are not advertised or promoted to the general public and class attendance is restricted to individuals employed by or contracted by the corporate customer.
  15. "Trainer" means (i) an academically accredited educator engaged by a Microsoft Imagine Academy Program Member to teach an Authorized Training Session, (ii) an academically accredited educator validated as a Microsoft Learn for Educators – Validated Educator, and/or (iii) a MCT.
  16. "Trainer Content" means the trainer version of the Microsoft Instructor-Led Courseware and additional supplemental content designated solely for Trainers' use to teach a training session using the Microsoft Instructor-Led Courseware. Trainer Content may include Microsoft PowerPoint presentations, trainer preparation guide, train the trainer materials, Microsoft One Note packs, classroom setup guide and Pre-release course feedback form. To clarify, Trainer Content does not include any software, virtual hard disks or virtual machines.
2. **USE RIGHTS.** The Licensed Content is licensed, not sold. The Licensed Content is licensed on a **one copy per user basis**, such that you must acquire a license for each individual that accesses or uses the Licensed Content.
- 2.1 Below are five separate sets of use rights. Only one set of rights apply to you.
    1. **If you are a Microsoft Imagine Academy (MSIA) Program Member:**
      1. Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.
      2. For each license you acquire on behalf of an End User or Trainer, you may either:
        1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User who is enrolled in the Authorized Training Session, and only immediately prior to the commencement of the Authorized Training Session that is the subject matter of the Microsoft Instructor-Led Courseware being provided, **or**
        2. provide one (1) End User with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
        3. provide one (1) Trainer with the unique redemption code and instructions on how they can access one (1) Trainer Content.
      3. For each license you acquire, you must comply with the following:
        1. you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,
        2. you will ensure each End User attending an Authorized Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Authorized Training Session,
        3. you will ensure that each End User provided with the hard-copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End

User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,

4. you will ensure that each Trainer teaching an Authorized Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Authorized Training Session,
5. you will only use qualified Trainers who have in-depth knowledge of and experience with the Microsoft technology that is the subject of the Microsoft Instructor-Led Courseware being taught for all your Authorized Training Sessions,
6. you will only deliver a maximum of 15 hours of training per week for each Authorized Training Session that uses a MOC title, and
7. you acknowledge that Trainers that are not MCTs will not have access to all of the trainer resources for the Microsoft Instructor-Led Courseware.

**2. If you are a Microsoft Learning Competency Member:**

1. Each license acquire may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.
2. For each license you acquire on behalf of an End User or MCT, you may either:
  1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User attending the Authorized Training Session and only immediately prior to the commencement of the Authorized Training Session that is the subject matter of the Microsoft Instructor-Led Courseware provided, **or**
  2. provide one (1) End User attending the Authorized Training Session with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
  3. you will provide one (1) MCT with the unique redemption code and instructions on how they can access one (1) Trainer Content.
3. For each license you acquire, you must comply with the following:
  1. you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,
  2. you will ensure that each End User attending an Authorized Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Authorized Training Session,
  3. you will ensure that each End User provided with a hard-copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,

4. you will ensure that each MCT teaching an Authorized Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Authorized Training Session,
5. you will only use qualified MCTs who also hold the applicable Microsoft Certification credential that is the subject of the MOC title being taught for all your Authorized Training Sessions using MOC,
6. you will only provide access to the Microsoft Instructor-Led Courseware to End Users, and
7. you will only provide access to the Trainer Content to MCTs.

**3. If you are a MPN Member:**

1. Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.
2. For each license you acquire on behalf of an End User or Trainer, you may either:
  1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User attending the Private Training Session, and only immediately prior to the commencement of the Private Training Session that is the subject matter of the Microsoft Instructor-Led Courseware being provided, **or**
  2. provide one (1) End User who is attending the Private Training Session with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
  3. you will provide one (1) Trainer who is teaching the Private Training Session with the unique redemption code and instructions on how they can access one (1) Trainer Content.
3. For each license you acquire, you must comply with the following:
  1. you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,
  2. you will ensure that each End User attending an Private Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Private Training Session,
  3. you will ensure that each End User provided with a hard copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,
  4. you will ensure that each Trainer teaching an Private Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Private Training Session,

5. you will only use qualified Trainers who hold the applicable Microsoft Certification credential that is the subject of the Microsoft Instructor-Led Courseware being taught for all your Private Training Sessions,
6. you will only use qualified MCTs who hold the applicable Microsoft Certification credential that is the subject of the MOC title being taught for all your Private Training Sessions using MOC,
7. you will only provide access to the Microsoft Instructor-Led Courseware to End Users, and
8. you will only provide access to the Trainer Content to Trainers.

**4. If you are an End User:**

For each license you acquire, you may use the Microsoft Instructor-Led Courseware solely for your personal training use. If the Microsoft Instructor-Led Courseware is in digital format, you may access the Microsoft Instructor-Led Courseware online using the unique redemption code provided to you by the training provider and install and use one (1) copy of the Microsoft Instructor-Led Courseware on up to three (3) Personal Devices. You may also print one (1) copy of the Microsoft Instructor-Led Courseware. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.

**5. If you are a Trainer.**

1. For each license you acquire, you may install and use one (1) copy of the Trainer Content in the form provided to you on one (1) Personal Device solely to prepare and deliver an Authorized Training Session or Private Training Session, and install one (1) additional copy on another Personal Device as a backup copy, which may be used only to reinstall the Trainer Content. You may not install or use a copy of the Trainer Content on a device you do not own or control. You may also print one (1) copy of the Trainer Content solely to prepare for and deliver an Authorized Training Session or Private Training Session.

2. If you are an MCT, you may customize the written portions of the Trainer Content that are logically associated with instruction of a training session in accordance with the most recent version of the MCT agreement.
3. If you elect to exercise the foregoing rights, you agree to comply with the following: (i) customizations may only be used for teaching Authorized Training Sessions and Private Training Sessions, and (ii) all customizations will comply with this agreement. For clarity, any use of "customize" refers only to changing the order of slides and content, and/or not using all the slides or content, it does not mean changing or modifying any slide or content.

- 2.2 **Separation of Components.** The Licensed Content is licensed as a single unit and you may not separate their components and install them on different devices.
- 2.3 **Redistribution of Licensed Content.** Except as expressly provided in the use rights above, you may not distribute any Licensed Content or any portion thereof (including any permitted modifications) to any third parties without the express written permission of Microsoft.
- 2.4 **Third Party Notices.** The Licensed Content may include third party code that Microsoft, not the third party, licenses to you under this agreement. Notices, if any, for the third party code are included for your information only.
- 2.5 **Additional Terms.** Some Licensed Content may contain components with additional terms, conditions, and licenses regarding its use. Any non-conflicting terms in those conditions and licenses also apply to your use of that respective component and supplements the terms described in this agreement.

3. **LICENSED CONTENT BASED ON PRE-RELEASE TECHNOLOGY.** If the Licensed Content's subject matter is based on a pre-release version of Microsoft technology ("Pre-release"), then in addition to the other provisions in this agreement, these terms also apply:
  1. **Pre-Release Licensed Content.** This Licensed Content subject matter is on the Pre-release version of the Microsoft technology. The technology may not work the way a final version of the technology will and we may change the technology for the final version. We also may not release a final version. Licensed Content based on the final version of the technology may not contain the same information as the Licensed Content based on the Pre-release version. Microsoft is under no obligation to provide you with any further content, including any Licensed Content based on the final version of the technology.
  2. **Feedback.** If you agree to give feedback about the Licensed Content to Microsoft, either directly or through its third party designee, you give to Microsoft without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft technology, Microsoft product, or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its technology, technologies, or products to third parties because we include your feedback in them. These rights survive this agreement.
  3. **Pre-release Term.** If you are an Microsoft Imagine Academy Program Member, Microsoft Learning Competency Member, MPN Member, Microsoft Learn for Educators – Validated Educator, or Trainer, you will cease using all copies of the Licensed Content on the Pre-release technology upon (i) the date which Microsoft informs you is the end date for using the Licensed Content on the Pre-release technology, or (ii) sixty (60) days after the commercial release of the technology that is the subject of the Licensed Content, whichever is earliest ("Pre-release term"). Upon expiration or termination of the Pre-release term, you will irretrievably delete and destroy all copies of the Licensed Content in your possession or under your control.
4. **SCOPE OF LICENSE.** The Licensed Content is licensed, not sold. This agreement only gives you some rights to use the Licensed Content. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the Licensed Content only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the Licensed Content that only allows you to use it in certain ways. Except as expressly permitted in this agreement, you may not:
  - access or allow any individual to access the Licensed Content if they have not acquired a valid license for the Licensed Content,
  - alter, remove or obscure any copyright or other protective notices (including watermarks), branding or identifications contained in the Licensed Content,
  - modify or create a derivative work of any Licensed Content,
  - publicly display, or make the Licensed Content available for others to access or use,
  - copy, print, install, sell, publish, transmit, lend, adapt, reuse, link to or post, make available or distribute the Licensed Content to any third party,
  - work around any technical limitations in the Licensed Content, or
  - reverse engineer, decompile, remove or otherwise thwart any protections or disassemble the Licensed Content except and only to the extent that applicable law expressly permits, despite this limitation.
5. **RESERVATION OF RIGHTS AND OWNERSHIP.** Microsoft reserves all rights not expressly granted to you in this agreement. The Licensed Content is protected by copyright and other intellectual property

laws and treaties. Microsoft or its suppliers own the title, copyright, and other intellectual property rights in the Licensed Content.

6. **EXPORT RESTRICTIONS.** The Licensed Content is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the Licensed Content. These laws include restrictions on destinations, end users and end use. For additional information, see [www.microsoft.com/exporting](http://www.microsoft.com/exporting).
7. **SUPPORT SERVICES.** Because the Licensed Content is provided "as is", we are not obligated to provide support services for it.
8. **TERMINATION.** Without prejudice to any other rights, Microsoft may terminate this agreement if you fail to comply with the terms and conditions of this agreement. Upon termination of this agreement for any reason, you will immediately stop all use of and delete and destroy all copies of the Licensed Content in your possession or under your control.
9. **LINKS TO THIRD PARTY SITES.** You may link to third party sites through the use of the Licensed Content. The third party sites are not under the control of Microsoft, and Microsoft is not responsible for the contents of any third party sites, any links contained in third party sites, or any changes or updates to third party sites. Microsoft is not responsible for webcasting or any other form of transmission received from any third party sites. Microsoft is providing these links to third party sites to you only as a convenience, and the inclusion of any link does not imply an endorsement by Microsoft of the third party site.
10. **ENTIRE AGREEMENT.** This agreement, and any additional terms for the Trainer Content, updates and supplements are the entire agreement for the Licensed Content, updates and supplements.
11. **APPLICABLE LAW.**
  1. United States. If you acquired the Licensed Content in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.
  2. Outside the United States. If you acquired the Licensed Content in any other country, the laws of that country apply.
12. **LEGAL EFFECT.** This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the Licensed Content. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.
13. **DISCLAIMER OF WARRANTY. THE LICENSED CONTENT IS LICENSED "AS-IS" AND "AS AVAILABLE." YOU BEAR THE RISK OF USING IT. MICROSOFT AND ITS RESPECTIVE AFFILIATES GIVES NO EXPRESS WARRANTIES, GUARANTEES, OR CONDITIONS. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT AND ITS RESPECTIVE AFFILIATES EXCLUDES ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.**
14. **LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. YOU CAN RECOVER FROM MICROSOFT, ITS RESPECTIVE AFFILIATES AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO US\$5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.**

This limitation applies to

- anything related to the Licensed Content, services, content (including code) on third party Internet sites or third-party programs; and
- claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential, or other damages.

**Please note: As this Licensed Content is distributed in Quebec, Canada, some of the clauses in this agreement are provided below in French.**

**Remarque : Ce le contenu sous licence étant distribué au Québec, Canada, certaines des clauses dans ce contrat sont fournies ci-dessous en français.**

**EXONÉRATION DE GARANTIE.** Le contenu sous licence visé par une licence est offert « tel quel ». Toute utilisation de ce contenu sous licence est à votre seule risque et péril. Microsoft n'accorde aucune autre garantie expresse. Vous pouvez bénéficier de droits additionnels en vertu du droit local sur la protection dues consommateurs, que ce contrat ne peut modifier. La ou elles sont permises par le droit locale, les garanties implicites de qualité marchande, d'adéquation à un usage particulier et d'absence de contrefaçon sont exclues.

**LIMITATION DES DOMMAGES-INTÉRÊTS ET EXCLUSION DE RESPONSABILITÉ POUR LES DOMMAGES.** Vous pouvez obtenir de Microsoft et de ses fournisseurs une indemnisation en cas de dommages directs uniquement à hauteur de 5,00 \$ US. Vous ne pouvez prétendre à aucune indemnisation pour les autres dommages, y compris les dommages spéciaux, indirects ou accessoires et pertes de bénéfices.

Cette limitation concerne:

- tout ce qui est relié au le contenu sous licence, aux services ou au contenu (y compris le code) figurant sur des sites Internet tiers ou dans des programmes tiers; et.
- les réclamations au titre de violation de contrat ou de garantie, ou au titre de responsabilité stricte, de négligence ou d'une autre faute dans la limite autorisée par la loi en vigueur.

Elle s'applique également, même si Microsoft connaissait ou devrait connaître l'éventualité d'un tel dommage. Si votre pays n'autorise pas l'exclusion ou la limitation de responsabilité pour les dommages indirects, accessoires ou de quelque nature que ce soit, il se peut que la limitation ou l'exclusion ci-dessus ne s'appliquera pas à votre égard.

**EFFET JURIDIQUE.** Le présent contrat décrit certains droits juridiques. Vous pourriez avoir d'autres droits prévus par les lois de votre pays. Le présent contrat ne modifie pas les droits que vous confèrent les lois de votre pays si celles-ci ne le permettent pas.

Revised April 2019



# Contents

■	<b>Module 0 Introduction</b>	1
	Start Here	1
■	<b>Module 1 The Role of the Azure Database Administrator</b>	3
	Module Introduction	3
	Azure Data Platform Roles	4
	Azure Database Platforms and Options	7
	SQL Server Compatibility Level	17
	Azure Preview Features	19
■	<b>Module 2 Plan and Implement Data Platform Resources</b>	23
	Module Introduction	23
	Deploying SQL Server in a Virtual Machine	24
	Deploying a Platform as a Service Solution	40
	Deploying MariaDB, MySQL, and PostgreSQL on Azure	59
	Module Summary	67
■	<b>Module 3 Implement a Secure Environment</b>	71
	Module Introduction	71
	Configure Database Authentication	72
	Configure Database Authorization	80
	Implement Security for Data at Rest	86
	Implement Security for Data in Transit	99
	Implement Compliance Controls for Sensitive Data	102
	Module Summary	107
■	<b>Module 4 Monitor and Optimize Operational Resources</b>	111
	Module-Introduction	111
	Baselines and Performance Monitoring	112
	Major Causes of Performance Issues	132
	Configuring Resources for Optimal Performance	141
	User Database Configuration	148
	Performance-related maintenance tasks	153
	Module Summary	158
■	<b>Module 5 Optimize Query Performance</b>	163
	Module-Introduction	163
	Understanding SQL Server Query Plans	165



Explore Performance-based Database Design .....	182
Evaluating Performance Improvements .....	197
Module Summary .....	202
<b>Module 6 Automation of Tasks</b> .....	205
Module Introduction .....	205
Setting up Automatic Deployment .....	206
Defining Scheduled Tasks with the SQL Server Agent .....	216
Configuring Extended Events .....	227
Managing Azure PaaS Resources by Using Automated Methods .....	236
Module Summary .....	247
<b>Module 7 Planning and Implementing a High Availability and Disaster Recovery Environment</b> .....	251
Module Introduction .....	251
High Availability and Disaster Recovery Strategies .....	252
IaaS Platform and Database Tools for HADR .....	266
PaaS Platform and Database Tools for HADR .....	275
Database Backup and Recovery .....	279
Module Summary .....	286

# Module 0 Introduction

## Start Here

## About this Course

### Course Description

This course provides students with the knowledge and skills to administer a SQL Server database infrastructure for cloud, on-premises and hybrid relational databases and who work with the Microsoft PaaS relational database offerings. Additionally, it will be of use to individuals who develop applications that deliver content from SQL-based relational databases.

**Level:** Intermediate

### Audience

The audience for this course is data professionals managing data and databases who want to learn about administering the data platform technologies that are available on Microsoft Azure.

The Azure Database Administrator implements and manages the operational aspects of cloud-native and hybrid data platform solutions built on Azure Data Services and SQL Server. The Azure Database Administrator uses a variety of methods and tools to perform day-to-day operations, including applying knowledge of using T-SQL for administrative management purposes.

This role is responsible for management, availability, security and performance monitoring and optimization of modern relational database solutions. This role works with the Azure Data Engineer role to manage operational aspects of data platform solutions.

### Prerequisites

Successful Azure Database Administrators start this role with experience on operating systems, virtualization, cloud infrastructure, storage structures, and networking. They should also have knowledge of basic relational database concepts and knowledge of querying with the SQL language. Understanding the following concepts will be beneficial:

- Understanding of on-premises virtualization technologies, including: VMs, virtual networking, and virtual hard disks.

- Understanding of network configuration, including TCP/IP, Domain Name System (DNS), virtual private networks (VPNs), firewalls, and encryption technologies.
- Understanding of SQL Server database organization and database creation, including the configuration and purpose of data files and log files, and the purpose and use of the transaction log.
- Basic knowledge of the SQL language, particular the Microsoft T-SQL dialect.
- Understanding of table and index structures, usefulness of indexes, and familiarity with the main data types used in SQL Server tables.

**Expected learning**

- Administer Relational Databases on Microsoft Azure using the Azure portal, Cloud Shell, Azure PowerShell, CLI, and ARM templates.
- Deploy and configure resources, including databases and database servers using manual methods.
- Configure database authentication and authorization.
- Implement security policies for data at rest and in transit.
- Monitor server performance to identify major causes of performance problems after establishing a baseline.
- Configure resources, including databases, servers and networks, for optiml performance.
- Monitor and tune query performance.
- Manage performance-related maintenance tasks including updating of statistics and defragmentation.
- Configure database auto-tuning features.
- Deploy resources using automated deployment scripts.
- Create and manage tasks, alerts and notifications.
- Determine optimum HADR strategy that will satisfy RPO and RTO requirements.
- Plan and configure a disaster recovery solution.

# Module 1 The Role of the Azure Database Administrator

## Module Introduction

### Module Introduction

This module explores the role of a database administrator in the world of Azure. It also provides some foundational information relevant to the overall content. This includes a review of the various SQL Server-based options (SQL Server in a VM, Managed Instances, and Azure SQL Database.) Students will learn why compatibility level is a crucial concept when working with SQL databases in Azure. Students are also introduced to other database platforms available on Azure in addition to those based on SQL Server, in particular PostgreSQL and MySQL.

### Learning Objectives

After taking this module, you will:

- Understand the role of Azure Database Administrator as it fits in with other data platform roles
- Be able to describe the key differences between the SQL Server-based database options in Azure and other open-source database platforms available on Azure
- Describe the difference between versions and compatibility levels
- Know how to enable and disable preview features

# Azure Data Platform Roles

## Introduction

**Prerequisites:** Students attending this course or taking this exam should have a solid understanding of the principals of SQL Server Database Administration, and a beginner level knowledge of the Azure Data Platform options, including Azure SQL Database, Azure SQL Database Managed Instance, Azure Virtual Machines, and Azure MariaDB/MySQL/PostgreSQL database. It is also expected that you have a familiarity with the T-SQL language, executing queries and evaluating the results.

The role of Azure Database Administrator is just one of the roles that can be filled by professionals working with the Microsoft data platform. There are five different role-based certifications offered by Microsoft for people whose main job responsibilities deal with cloud-focussed data.

**Azure Data Engineer:** Azure Data Engineers design and implement the management, monitoring, security, and privacy of data using the full stack of Azure data services to satisfy business needs.

**Azure Data Analyst:** Data Analysts enable businesses to maximize the value of their data assets by using Microsoft Power BI. As a subject matter expert, Data Analysts are responsible for designing and building scalable data models, cleaning and transforming data, and enabling advanced analytic capabilities that provide meaningful business value through easy-to-comprehend data visualizations.

**Azure Data Scientist:** The Azure Data Scientist applies their knowledge of data science and machine learning to implement and run machine learning workloads on Azure; in particular, using Azure Machine Learning Service.

**Azure Artificial Intelligence Engineer:** Azure AI Engineers use Cognitive Services, Machine Learning, and Knowledge Mining to architect and implement Microsoft AI solutions involving natural language processing, speech, computer vision, bots, and agents

**Azure Database Administrator:** The Azure Database Administrator implements and manages the operational aspects of cloud-native and hybrid data platform solutions built on Microsoft Azure data services and Microsoft SQL Server. The Azure Database Administrator uses a variety of methods and tools to perform day-to-day operations, including applying knowledge of using T-SQL for administrative management purposes.

As the name of this course and certification implies, the material covered is Azure-focused and the examples, demos and lab exercises will be looking at either a pure Azure-based environment or a hybrid environment, involving both Azure and on-premises data. Note that 'hybrid' can also refer to an environment that involves Azure and data managed or stored by a different cloud provider.

This does not mean that if you are working in an environment that only has on-premises databases and applications that you can't benefit from this training. As we'll see, a SQL Server running in an Azure virtual machine is for almost all intents and purposes equivalent to an on-premises SQL Server. Almost everything you'll learn about working with SQL Server in an Azure VM will be applicable to all your SQL Servers.

Let's go into a bit more detail into the tasks that an Azure database administrator will need to be responsible for, as suggested in the above description. We'll look at these tasks module by module.

## Plan and Implement Data Platform Resources

Module 2 will look at the methods for deploying data platform resources in Azure. It will discuss options for both upgrading and migrating existing SQL databases to Azure. The module goes into detail on the differences between the various platform offerings and what you need to know to implement resources on both an IaaS platform and a PaaS platform.

It is expected that you understand the basic structure of a SQL Server database, and understand the purpose of the databases data and transaction log files. You should be able to use the basic SQL Server tools including Azure Data Studio, SQL Server Management Studio, and SQL Server Configuration Manager.

## Implement a Secure Environment

Module 3 covers aspects of securing your database environment, including both authentication (determining who you are) and authorization (determining what you have the right to do) within your database environment. The tasks discussed also include protecting the data with encryption, both for data at-rest and in transit, implementing compliance controls to hide sensitive data within your database, and assessing other threats to the security of your data.

It is expected that you have a basic understand of security principles and why security is such a crucial topic. You should know the difference between SQL Server security for authentication and Windows or Active Directory authentication.

## Monitor and Optimize Operational Resources

Module 4 teach you about resource optimization for your databases, whether they're created using IaaS or PaaS services. The module also covers monitoring server and hardware resources and discusses the importance of establishing a performance baseline. You will learn how to interpret performance metrics for the most critical resources. The module covers tuning for database issues that aren't directly related to individual query performance (which is covered in Module 5.)

It is expected that you have some familiarity with Windows Performance Monitoring and what the main performance-related resources are. It is also helpful to understand the concepts of resource contention, locking, blocking and deadlocks. Knowledge of dynamic management objects and their usefulness is also expected.

## Optimize Query Performance

Module 5 explore various query plan viewing options to identify problem areas in execution plans. You will also use Query Store (available in SQL Server 2016+ and Azure SQL Database) to extract plans that you did not directly execute. This module looks at several useful dynamic management objects that provide information into query performance and the use of indexes. You will explore how changing index structures can affect performance and will look at possible changes in the queries themselves, including the use of hints.

It is assumed that you have a basic understanding of index structures and how indexes are useful in helping the SQL Server's execution engine find the data you are interested in. Knowledge of dynamic management objects will also be useful.

## Automate Tasks

Module 6 provides details regarding the automating tasks in order to simplify the DBA's job. Automation methods include scheduling tasks for regular maintenance jobs, as well as multi-instance administration and configuration of notifications for task success or failure or non-completion.

It is expected that you have a basic understanding of the benefits of automation. Familiarity with the SQL Server Agent and its components is also assumed. A basic familiarity with PowerShell will also be very helpful.

## Plan and Implement a High Availability and Disaster Recovery Environment

Module 7 covers configuring, testing, and managing a solution for high availability and disaster recovery (HADR) in Azure, for both Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS) deployments. This module will not only cover basic requirements, but also the various options available to achieve HADR.

The more background you have in Azure high availability options the more you'll be able to understand the high availability options for your data platform. It is expected that you understand why and when high availability is important. It is assumed you know why backups are important for disaster recovery even with a high availability solution in place. To further understand SQL Server disaster recovery, an understanding of the importance and usage of the transaction log will be very helpful.

# Azure Database Platforms and Options

## Introduction

Although Azure platforms and options are described in detail in later modules, in this lesson we'll cover a high-level view to provide necessary background for the modules that follow. The range of options for deploying SQL Server on Azure are wide and comprehensive, and challenging for many administrators to navigate though. Many Database Administrators are tasked with not just maintaining and supporting SQL Server, but also providing assistance with making architectural decisions as they relate to the relational database management system (RDBMS) platform.

The decisions vary by the nature of the application, whether the application is developed in-house or by an independent software vendor (ISV). While most of the Azure Data Platform offerings are quite similar in their implementation, there are nuanced differences which require changes in application code which may not be possible if you do not own the application code.

## Learning Objectives

In this section, you will learn about:

- Options for SQL Server in an Azure VM
- Deployment Options for SQL Server on Azure
- Azure SQL Platform as a Service Offerings
- Azure open source Database offerings

## Understanding Azure Services

In general, cloud services can be broken down into two sets of services: Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) offerings. IaaS services typically consist of virtual machines, storage, and virtual networking components, and are largely managed by the user in terms of patching and software. On the other hand, PaaS services have a larger percentage of management tasks which are handled by the cloud provider. In addition, PaaS services tend to be more flexible in scaling. In the Azure Data Platform, PaaS services include Azure SQL Database, Azure SQL Database managed instance, and Azure MySQL/MariaDB/PostgreSQL.

## SQL Server in an Azure VM IaaS

Many applications will require a VM running SQL Server. Some reasons for this include:

1. **Older Versions of SQL Server**—If an application requires an older version of SQL Server for vendor support, running inside a VM is the best option for those applications, because it allows for the application to be supported by that vendor.
2. **Use of other SQL Server Services**—While Analysis Services and to an extent Integration Services (through the use of Azure Data Factory) are available as PaaS offerings, many users maximize their licensing by running SQL Server Analysis Services, Integration Services, and/or Reporting Services on the same machine as the database engine.
3. **General Application Incompatibility**—This is a little bit of a catch-all, but for example, Azure SQL Database does not support cross-database querying, while Managed Instance does. Some applications may require additional services to be co-located with the database instance in a manner that is not compatible with a PaaS offering.

In addition to the above reasons, many organizations find moving into VMs to be an easier migration path for their initial foray into the cloud. The networking and storage configurations are perceived to be easier to implement than a PaaS solution. With this in mind, SQL Server offers a number of offerings to maximize your productivity using VMs in Azure.

## *SQL Server IaaS Agent Extension*

When you deploy an SQL Server VM from the Azure Marketplace, as seen in Figure 1, part of the process installs the IaaS Agent Extension.

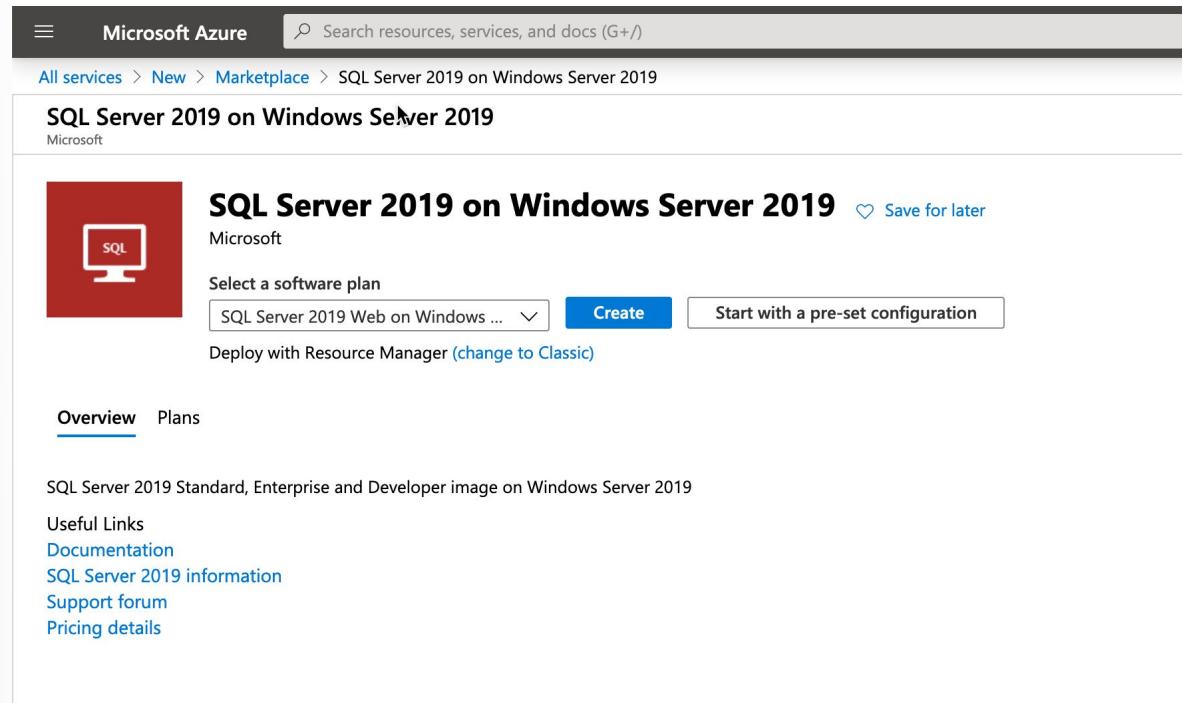


Figure 1 SQL Server VM Creation from Azure Marketplace

Extensions are code that is executed on your VM post-deployment, typically to perform post deployment configurations, like installing anti-virus features, or installing a Windows feature. The SQL Server IaaS Agent Extension provides three key features that can reduce your administrative overhead.

- **SQL Server Automated Backup**
- **SQL Server Automated Patching**
- **Azure Key Vault Integration**

In addition to these features, the extension allows you to view information about your SQL Server's configuration and storage utilization as shown in Figure 2.

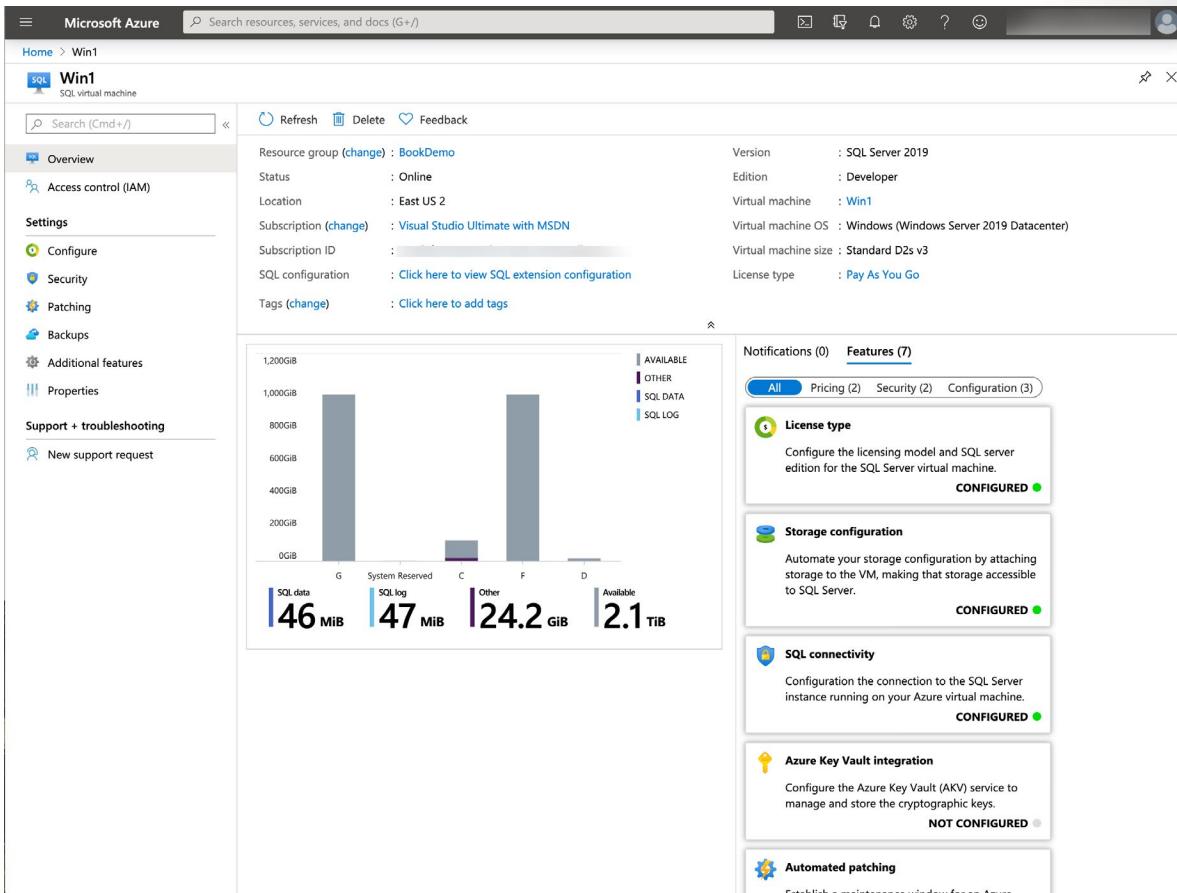


Figure 2 Image of SQL Virtual Machine Configuration in Azure Portal

## Versions of SQL Server Available on Azure VMs

Microsoft keeps images of all supported versions of SQL Server available in the Azure Marketplace. If you have a need for an older version, that is covered by an extended support contract, you must install your own SQL Server binaries.

## Features to Support SQL Server on IaaS

In recent releases of SQL Server, Microsoft has introduced many features to support running SQL Server in an Azure VM. We are going to focus on two features related to backup—Backup to URL and Azure Backup. Backup to URL allows you to use standard backup syntax to backup your databases to Azure storage, while Azure Backup for SQL VMs offers a complete enterprise backup solution that automatically handles your backups across your infrastructure.

## Deployment Options for Azure VMs

All resources in Azure share a common provider known as Azure Resource Manager which acts as a management and deployment service for Azure. While there are numerous ways to deploy Azure resources, ultimately, they all end up going into JSON documents known as Azure Resource Manager (ARM) templates, which is itself is one of the deployment options for Azure Resources. The main difference between the processes is that ARM Templates are explicitly a declarative deployment approach which

describe the desired structure and state of the resources to be deployed, whereas the other methods can all be described as imperative which uses procedural models to explicitly specify a process to be executed. In large scale deployment the declarative approach is better and should be followed.

## Azure Resource Manager Templates

Azure ARM Templates have the benefit of being able to deploy a full set of resources in one single declarative template. This includes the ability to build dependencies into the templates, as well as using parameters to change deployment values at deployment time. Once you have a template, you can deploy it a number of ways including an Azure DevOps pipeline, or through the custom deployments blade in the Azure Portal.

## High Availability in Azure

The Azure platform is designed to be fault tolerant and quickly recover from service faults and transient errors. In fact, many organizations see higher levels of availability in single VM (which with premium storage is 99.9% uptime) deployments than they previously experienced in their on-premises environments. This will be covered in full detail in Module 7, but you will learn about the build blocks here. Azure offers a number of specific features to support high availability including availability sets, availability zones, and load balancing techniques which you will learn more about in subsequent modules.

## Overview of Azure Storage

Azure offers a fully redundant object-based storage model. This will be covered in more detail in Module 4, but there few things to be aware of in designing and deploying VM architecture. In terms of VMs there are four types of storage you can use:

- Standard Storage
- Standard SSD Storage
- Premium Storage
- Ultra Disk

For production SQL Server data and transaction log files, you should only use Premium Storage and Ultra Disk. With premium storage you will see latencies in the range of 5-10 ms on a properly configured system, and with Ultra Disk you may have sub millisecond latency but will likely see 1-2ms workloads in the real world. You can use standard storage is for your database backups. The performance is adequate for most backup and restore workloads.

## Azure SQL Database Managed Instance for Modernization and Migration

While many organizations initially migrate to Azure using IaaS offerings, the platform as a service (PaaS) service offering allows for additional benefits. One key benefit is that you no longer have to have to install or patch SQL Server as that is performed by the service. Additionally, consistency checking, and backups are also part of the managed service, and there are additional security and performance tools that are included in the PaaS offerings.

## Azure SQL Managed Instance Features

Azure SQL Managed Instance allows for easy migration paths for existing applications by allowing restores from on-premises backups. Unlike Azure SQL Database, which is designed around single database structures, Managed Instance provides an entire SQL Server instance, allowing up to 100 databases, as well as providing access to the system databases. Managed Instance provides other features that are not available in Azure SQL Database, including cross-database queries, common language runtime (CLR) and along with the msdb system database, it allows the use of SQL Agent.

## Managed Instance Offerings

Managed Instance comes in two tiers of service—General Purpose and Business Critical. Both service tiers support the same feature set, and the main differences between the two tiers relate to performance and availability. The only feature differences between the two tiers are that Business Critical supports In-Memory OLTP and readable secondary replicas. Business critical also includes more memory per core, and uses direct attached storage (as opposed to networked) which offers lower storage latency.

## Hybrid Licensing Options for Azure PaaS Services

Microsoft offers several benefits to SQL Server licensees. For both Azure SQL Database and Azure SQL Database Managed Instances, taking advantage of your existing licenses can reduce the cost of running the PaaS offering.

For each core of Enterprise Edition with Active Software Assurance, you are eligible for one vCore of Azure SQL Database or Managed Instance Business Critical, and eight vCores of general purpose. For each core of Standard Edition with Software Assurance that you own, you are eligible for one vCore of General Purpose. This can reduce the total license costs by up to 40%. Effectively, you will only be paying for the compute and storage costs, and not the software licensing costs.

## High Availability Architecture in Azure PaaS

Azure SQL Database and Managed Instance have similar high availability architectures, which guarantee 99.99% percent uptime. Windows and SQL Server updates are handled by the backend infrastructure, generally without any impact to your application, though it is important to place retry logic into your application. The high availability solution is automatic and built-in to the platform, and is designed so that committed data is never lost to failure, and your database(s) have no single point of failure. The solution used to guarantee this availability depends on your service tier and will be discussed in full detail in Module 7.

## Connectivity Architecture

Connections to Azure SQL Database and Managed Instance are made through TDS endpoints. While the routing and security on these connections differ, the fundamental architecture is that there is a gateway component which is where connections are handled and routed to the database service. This gateway component is also deployed in a highly available fashion.

## Backup and Restore

Managed backup provides a fully managed backup service that takes full, differential, and log backups on a regular basis. You can also manually take copy only backups of the databases to Azure storage.

## *Automatic Tuning*

Running in a PaaS service allows for you to take advantage of additional compute resources on Azure to allow value added services to run. One of the best of these features, is automatic database tuning.

Automatic tuning currently includes the following features:

- Identify Expensive Queries
- Forcing Last Good Execution Plan
- Adding Indexes
- Removing Indexes

The Azure services uses a combination of built-in intelligence and advanced heuristics to determine the best indexes for your query patterns. These indexes are tested on a shadow copy of your database and then ultimately implemented into your database. All the aforementioned tuning features have the ability to be turned off in the event you would like to have more control over your environment.

## *Migrating to Azure SQL Database Managed Instance*

Migrating to Managed Instances is relatively easy given the near complete feature set of SQL Server.

There are a couple of ways to execute migrations:

- Restoring a backup
- Using the Database Migration Service

Backup and restore will incur more downtime, as it is not possible to restore with norecovery and apply log backups. The Database Migration Service is a managed service that connects to both your on-premises (or Azure VM) SQL Server to Managed Instance with near zero down time. Effectively it acts like an automated log shipping process which means you can keep your target databases in sync, right up to the point of cutover.

## **Azure SQL DB for Cloud-Native Apps**

While the Managed Instance service is designed to make for easy migration of existing applications, the Azure SQL Database service offering is aimed at new application development as it gives developers a great deal of flexibility in building new application services, and granular deployment options.

There are several configurations in which to deploy Azure SQL Database:

- Single Database
- Elastic Pools
- Hyperscale
- Serverless

All of these configurations share a common pricing model which is moving from the traditional database transaction unit (DTU) model, to the vCore model that Managed Instance uses. This vCore model allows for hybrid licensing, and offers other performance options like running Azure SQL Database on M or F series Azure VM which allow for more memory or faster CPUs.

## *Single Database*

This is simplest and original deployment model of Azure SQL Database. Once your server is deployed, you add a database to it, and can then connect your application to that database. You manage each of

your databases individually from scale and data size perspective. Each database deployed in this model (even if to the same logical server) has its own dedicated resources.

## *Elastic Pools*

Elastic pools are the Azure SQL Database equivalent of defining multiple databases in the same SQL Server instance. From a resource perspective, you deploy to an elastic pool, and resources are shared between multiple databases in the pool. This can dramatically lower the cost for a software as a service application model, since resources are shared between databases in the pool. Single databases can be added or removed from the elastic pool with only brief downtime at the end of the operation. Additionally, you can rescale a pool itself with minimal downtime. Elastic pools are best suited for workloads that have a low average utilization with non-concurrent spikes in workload.

## *Hyperscale*

Azure SQL Database has been limited to 4 TB of storage per database for many years. This is due to a physical limitation of the Azure infrastructure. Azure SQL Database Hyperscale changes the paradigm and allows for databases to be 100 TB and beyond. Hyperscale introduces new horizontal scaling techniques that uses advanced techniques to add compute nodes as the data sizes grow. The cost of Hyperscale is the same as the cost of Azure SQL Database; however there is a per terabyte cost for storage.

## *Serverless*

Despite its name, Azure SQL Database serverless does require you to have a server with your database. The serverless option can best be thought of as an auto-scale and auto-close solution for Azure SQL Database. It is very effective for lowering the costs in development and testing environments. The per hour/per vCore pricing for serverless is higher than normal Azure SQL Database, but the database and the billing will pause after an hour of inactivity.

Azure SQL Database, like virtual machines, can be deployed using ARM templates, PowerShell, Azure CLI, or the Azure Portal.

## *Network Options for Azure SQL Database*

Azure SQL Database by default has a public internet endpoint. Access to this endpoint can be controlled via firewall rules, or limited to specific Azure networks, using features like Virtual Network endpoints or Private Link, which you will learn about in later modules.

## *Scaling Elastic Pools*

One of the key benefits of elastic pools is that since resources are shared between a group of databases, the administrator does not need to manage the performance level of each databases individually. In the legacy DTU model, you assigned a default min/max DTU setting for each database in your pool, and had the ability to customize those settings for individual or group of databases. In the vCore model there is a similar paradigm which allows to assign as little as a quarter of a vCore to each database for a minimum up to all of the vCores as a maximum value. You can also increase the number of vCores or DTUs available to your pool, but this action is not immediate, and will incur a single short downtime, as the process to increase the size performs a migration task.

## *Serverless Autoscale*

Similar to elastic pool, the serverless tier of Azure SQL Database can automatically increase scale as workload increases. You have the ability to define a minimum and maximum number of vCores associated with each database. While most scale-up operations happen seamlessly sometime a database service is required to move to another host because of capacity constraints which may require a brief connection outage. For applications that are using serverless, you should ensure that the application connection is fault tolerant due to both scaling events, and startup events.

## **Open Source Databases on Azure SQL Platform**

Microsoft also offers the most popular open-source database platforms MySQL, MariaDB, and PostgreSQL on the Azure platform. The architecture and features are similar to that of the Azure SQL Database, but running different database engines. An additional benefit of these solutions is that they tightly integrate with other services like Azure Web Apps.

## *Network Connectivity in Azure MySQL and Azure Postgres*

Similar to Azure SQL Database there is a firewall that can be set for specific IP address or range of IP addresses, or you can configure a virtual network endpoint to allow connections from an Azure Virtual Network.

## *Query Store on Postgres*

One of the value-added features of PostgreSQL on the Azure platform is the query store, which is very similar to the query store feature in Azure SQL Database. The implementation is slightly different, but the query store keeps track of both query execution runtime statistics and wait stats. The query store data is stored in the `azure_sys` database on your Postgres server, in the `query_store` schema.

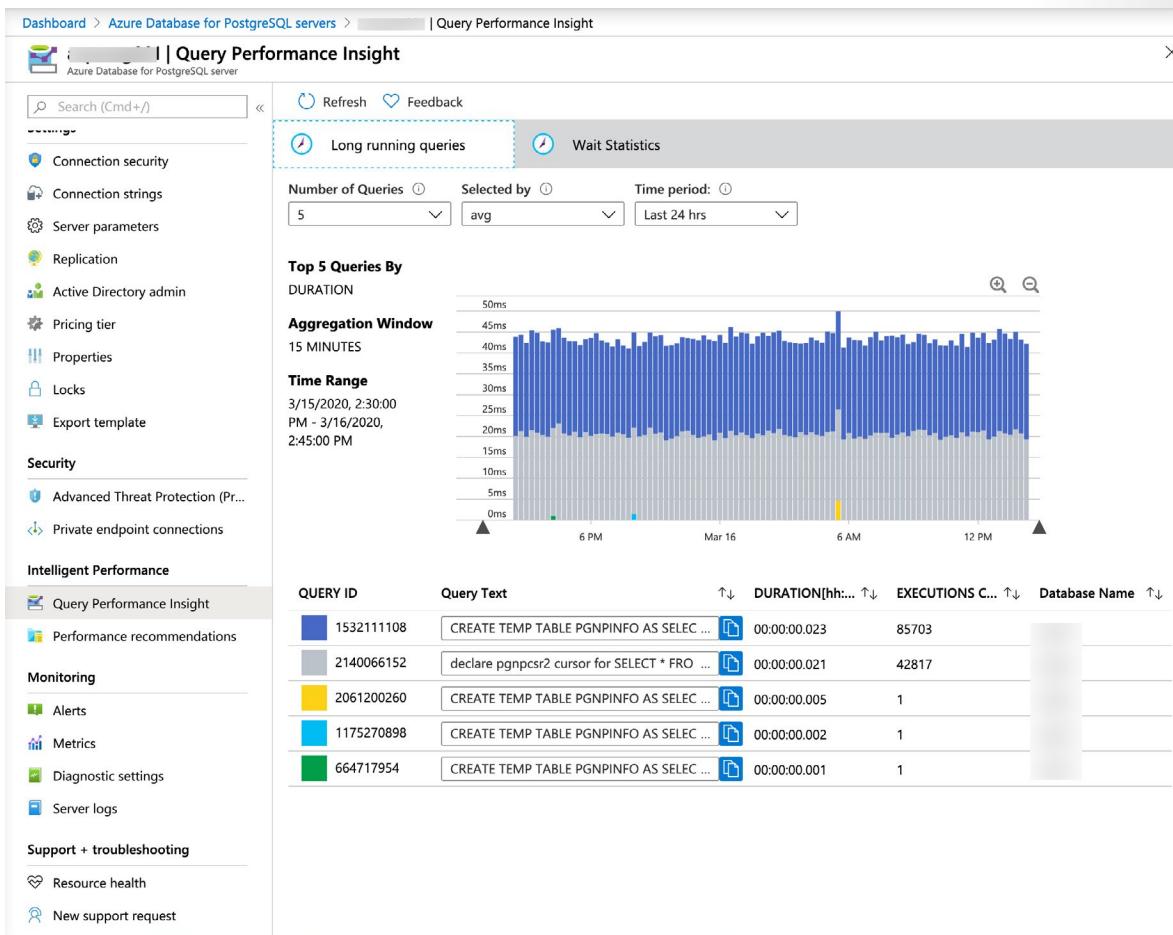


Figure 4 Query Performance Insight in Azure Portal for PostgreSQL

As seen in figure 4, the query store data drives the Query Performance Insight dashboard in the Azure Portal, offering a quick way to identify expensive queries in your service.

## Summary and Knowledge Check

The Azure platform offers you multiple options for configuring your database solutions. Whether its SQL Server running on an Azure VM with 416 cores and 12 terabytes of RAM, or a single core PostgreSQL database that costs less than a cup of coffee every day, you have options for how to run your database workloads to meet all of your performance and budget needs. The Azure Resource Manager platform allows for flexible programmatic and graphical options for deploying resources.

### Question 1

An Azure SQL Database Managed Instance represents what kind of cloud service?

- Platform as a Service (PAAS)
- Infrastructure as a Service (IAAS)
- Software as a Service (SAAS)

## Question 2

*Which of the following is NOT a valid reason for migrating your database into an IaaS environment?*

- Your applications need to run older versions of SQL Server, such as SQL Server 2016.
- You want Azure to manage all the upgrades, patching and server configuration.
- You need to use other SQL Server services with your application, such as SQL Server Analysis Services (SSAS), Integration Services (SSIS) and Reporting Services (SSRS)

## Question 3

*Which of the following is NOT an Azure SQL Database deployment option?*

- Availability Zones
- Serverless
- Elastic Pools

# SQL Server Compatibility Level

## Introduction

Historically, software vendors who build software for SQL Server have certified their software to run on a specific version of the database engine. For example, SharePoint 2016 was only certified to run on SQL Server 2014. This hampered many organizations who would potentially have to run on an older, or even unsupported version of the SQL Server Database Engine in order to maintain support from their application vendor.

## Learning Objectives

In this lesson you will learn:

- How SQL Server Compatibility Level affects database behavior
- Microsoft's support policy for SQL Server
- How to certify an application based on compatibility level

## SQL Server Compatibility Level

SQL Server compatibility level has always been a database level setting. Setting compatibility level to a specific version allows for specific T-SQL constructs and keywords to be used and the compatibility level also determines certain query optimizer behaviors. For example, if you had a database using compatibility level 100 (associated with SQL Server 2008) and migrated to SQL Server 2019, the execution plan shapes and query syntax support should remain the same as they did on SQL Server 2008.

## *Support Policy for SQL Server*

Microsoft has an extremely generous support policy for SQL Server—releases are supported for five years in primary support, and then five additional years in extended support. During the first five years, Microsoft updates all releases with enhanced capabilities, closes feature gaps, and addresses performance, functional, and security bugs. After a release moves into extended support Microsoft will only address security bugs.

While this support policy is quite generous, there are many benefits to running on the latest release of SQL Server including enhancements to performance, security, availability, and query functionality. This is further enhanced by the one to two-year release cadence of SQL Server, and the evergreen nature of the Azure SQL Database services, which means it never needs to be patched or upgraded and new features are added and fixes are applied automatically.

In order to take advantage of newer releases of SQL Server but maintain vendor support for applications, Microsoft has recommended that application vendors certify applications at a specific compatibility level, instead of for a particular software version. This process, called compatibility certification, allows for an application to run in Azure SQL Managed Instance or the latest release of SQL Server, while maintaining its vendor supported compatibility level.

Microsoft includes query plan shape protection, which means your query execution plans and their performance should be nearly the same (on similar hardware) and will treat changes in query plan shape at the same compatibility level as bugs in the project. This removes one of the big risks of upgrading SQL Server: optimizer changes causing degradation in query performance. Microsoft still recommends upgrading to a newer compatibility level when possible but will support databases on older compatibility levels as long as the release of SQL Server that you are running on is a supported release of SQL Server.

## Summary and Knowledge Check

Microsoft has struck a balance with supportability for the SQL Server platform and allowing vendor applications to continue to run as expected with its policy of basing support on compatibility level. You can upgrade your applications databases with the comfort that your performance level should be the same or better in the new release.

### Question 1

*Which version of SQL Server does compatibility level 100 equate to?*

- SQL Server 2008
- SQL Server 2014
- SQL Server 2019

### Question 2

*What does Microsoft guarantee if you upgrade versions of SQL Server but maintain the same compatibility level, on similar hardware?*

- Execution Plan Shape
- Elapsed time of queries
- Syntax compatibility

### Question 3

*Where do you change the compatibility level settings?*

- The individual database properties page
- The server settings page
- Using a trace flag through the Configuration Manager

# Azure Preview Features

## Introduction

Azure is constantly evolving, and new features are literally released hourly, but it may take some time before new features are generally available through the preview process. While preview features can benefit your applications, there are some tradeoffs around supportability.

## Learning Objectives

In this lesson you will learn the following:

- How the Azure preview feature process works
- The differences between private and public preview
- The support policy of preview features

## Azure Preview Features

Azure offers preview features that are designed for non-production usage. It is important to note that previews are subject to reduced or different service terms. Preview features may not be available in all regions, may have limited service level agreements (SLA), and may have limited functionality. You should not use preview features in a production application, unless you are working directly with the product team to ensure support.

Preview features are broken down into public and private preview. Public preview features are opted into in the portal but are available to everyone. Typically, private preview features will require that Microsoft add your subscription to an allow list for a given feature. You can enable public preview features by going to the Azure portal preview page at <https://azure.microsoft.com/en-us/updates/?status=inpreview>. Some features may require further opt-in at the individual resource, the public preview experience is not consistent across Azure services.

## Summary and Knowledge Check

Using Azure preview features comes with some risk as the support policy and available regions may not match your requirements. However, it can get you early access to start building your applications to meet future requirements.

### Question 1

*How do you get access to a private preview?*

- Gain access directly from Microsoft
- Check a box in the Azure Portal
- File a support case

# Answers

## Question 1

An Azure SQL Database Managed Instance represents what kind of cloud service?

- Platform as a Service (PAAS)
- Infrastructure as a Service (IAAS)
- Software as a Service (SAAS)

*Explanation*

## Question 2

Which of the following is NOT a valid reason for migrating your database into an IaaS environment?

- Your applications need to run older versions of SQL Server, such as SQL Server 2016.
- You want Azure to manage all the upgrades, patching and server configuration.
- You need to use other SQL Server services with your application, such as SQL Server Analysis Services (SSAS), Integration Services (SSIS) and Reporting Services (SSRS)

*Explanation*

## Question 3

Which of the following is NOT an Azure SQL Database deployment option?

- Availability Zones
- Serverless
- Elastic Pools

*Explanation*

## Question 1

Which version of SQL Server does compatibility level 100 equate to?

- SQL Server 2008
- SQL Server 2014
- SQL Server 2019

*Explanation*

## Question 2

What does Microsoft guarantee if you upgrade versions of SQL Server but maintain the same compatibility level, on similar hardware?

- Execution Plan Shape
- Elapsed time of queries
- Syntax compatibility

*Explanation*

**Question 3**

Where do you change the compatibility level settings?

- The individual database properties page
- The server settings page
- Using a trace flag through the Configuration Manager

*Explanation*

**Question 1**

How do you get access to a private preview?

- Gain access directly from Microsoft
- Check a box in the Azure Portal
- File a support case

*Explanation*



---

## Module 2 Plan and Implement Data Platform Resources

### Module Introduction

#### Module Introduction

This module introduces methods for deploying data platform resources in Azure. You will learn about options for both upgrading and migrating existing SQL databases to Azure. You will learn how to set up Azure resources to host SQL Server on a Virtual Machine, a Managed Instance, Azure SQL Database and either PostgreSQL or MySQL. You will learn how to determine which options are best based on specific requirements including the High Availability and Disaster Recovery (HADR) needs. They will learn to calculate resource requirements and create templates for their deployments.

### Learning Objectives

In this module, you will learn how to:

- Deploy resources using manual methods
- Recommend an appropriate database offering based on requirements
- Configure resources
- Evaluate and implement a strategy for moving a database to Azure

# Deploying SQL Server in a Virtual Machine

## Introduction

The most common reason for deploying SQL Server in an Azure Virtual Machine (VM) is because you want an easy, straightforward method to migrate an existing on-premises SQL Server into the cloud. Understanding the options and methods for deploying SQL Server in an Azure VM is critical to ensure a successful migration. Infrastructure as a Service (IaaS) allows for greater flexibility in configuration; however this means that you, as a database administrator, must plan carefully to properly configure VM sizing, storage, and networking options to ensure adequate performance for your workloads.

This lesson will focus on ways to provision and deploy Microsoft SQL Server into an Azure virtual machine as well as provide clear and concrete information on the various options when performing a migration.

## Learning Objectives:

In this lesson you will

- Explore the basics of SQL Server in an Infrastructure as a Service (IaaS) Offering
- Learn the available options for provisioning and deployment
- Examine methods for migration to Azure SQL Server virtual machine

## Overview of Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) allows the administrator to have more granular access over specific settings of the underlying infrastructure than the other Azure offerings. While the management of the underlying server and network hardware is managed by the Azure platform, you still retain access to the virtual storage, virtual networking configuration and any additional software you might install within the virtual machine, including Microsoft SQL Server.

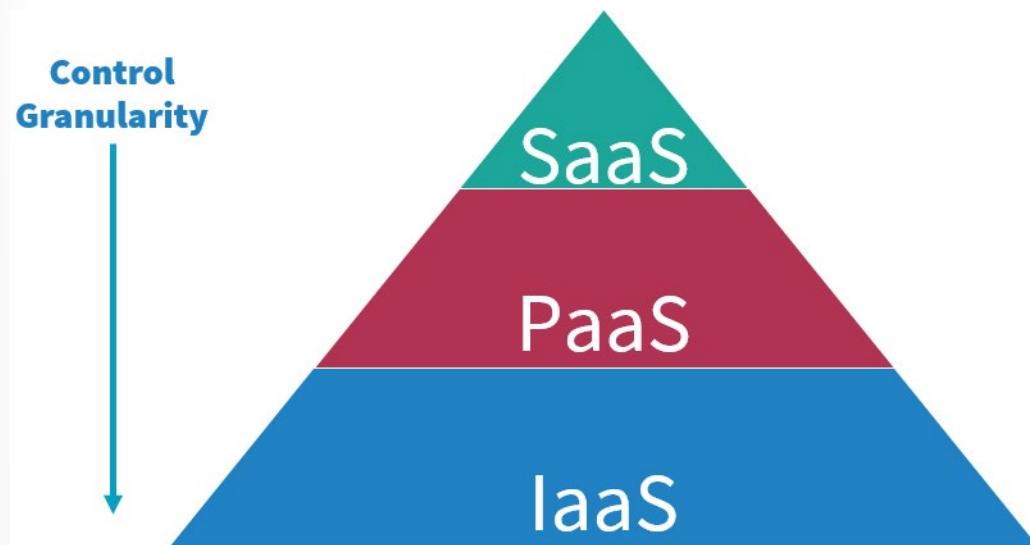


Figure 1 Granularity of Control of SaaS, PaaS, and IaaS options

Figure 1 illustrates the increased control you have using IaaS, compared to the other Azure SQL offerings. While the exact configuration options vary between service offerings, typically in SaaS offerings the administrator is responsible only for user security and possibly data management. When using PaaS

services, the operating system (OS) and other software are managed by the cloud provider—a good example of this is the Azure Database platform where the operating system and RDBMS are installed and configured by Microsoft, allowing you to quickly start building database applications. IaaS solutions are the most open ended; you are responsible for OS patching, as well as optimal configuration of your network and storage options. With an IaaS deployment, you are also responsible for software configuration.

For IaaS solutions running in Azure, Microsoft will manage any resource below the operating system, including the physical servers, storage and physical networking. The database administrator is responsible for configuration of your SQL Server instances running on the operating system.

Some of your applications may not be suited for other Azure offerings, such as Azure SQL Database, because they require specific operating conditions—whether that be a specific combination of SQL Server and Windows versions for vendor support purposes, or additional software that needs to be installed alongside of SQL Server. SQL Server paired with the Azure IaaS platform provides the required control options for many organizations, whether it be specific feature like CLR or replication, or the use of Active Directory (as opposed to Azure Active Directory) authentication. Another requirement is that some applications install software alongside SQL Server which requires direct access to the underlying operating system which is not supported in a PaaS model. These organizations and their applications can obtain the advantages of moving to a cloud service without losing critical capabilities that their organization requires.

## *SQL Server Licensing Models*

There are several different options related to how SQL Server is licensed when utilizing the Azure IaaS offering.

If you aren't participating in the Microsoft Software Assurance (SA) program, you can deploy an image from the Azure Marketplace containing a preconfigured SQL Server, and pay-per-minute for the use of SQL Server. This is referred to as the Pay as you Go model and the cost of the SQL Server license is included with the cost of the virtual machine.

If you are participating in the Microsoft Software Assurance (SA) program you have more flexibility in how your SQL Server is licensed:

1. You can use the previous method and pay-per-minute by deploying a virtual machine image containing a SQL Server from the Azure Marketplace
2. You can Bring Your Own License (BYOL) when deploying the virtual machine that doesn't contain a preconfigured SQL Server instance. This means that you already have purchased a valid SQL Server license for your on-premises infrastructure. This license can be applied to the virtual machine to ensure that you are properly licensed. You must report the usage of licenses to Microsoft by using the License Mobility verification form within 10 days of implementing the virtual machine.

When choosing this method, you can manually install SQL Server through media you have obtained, or you can choose to upload a virtual machine image to Azure.

In addition to flexible licensing options for SQL Server, there is also Window Server licensing options that can be taken advantage of known as the Azure Hybrid Use Benefit (AHUB). Similar to applying a SQL Server license you already have purchased, you are able to take advantage of Windows Server licenses you already own.

Reserving a virtual machine for one to three years provides another option for cost savings. This commitment does not require an upfront payment and can be billed monthly. Using the reservation option can be beneficial if you know the workloads are going to be persisted. The cost savings can be significant, especially for larger VMs.

## Virtual Machine Families

When deploying to an Azure virtual machine, there are several series, or "Families", of virtual machine sizes that can be selected. Each series is a combination of memory, CPU, and storage that meet certain requirements. For example, the series that are compute optimized have a higher CPU to memory ratio. Having multiple options allows you to select an appropriate hardware configuration for the expected workload. The following six series each have various sizes available, the details of which are fully described in the Azure Portal when you choose the option to select your VM size.

**General 1purpose<sup>2</sup>** - These VMs provide a balanced ration of CPU to memory. This VM class is ideal for testing and development, small to medium database servers, and low to medium traffic web servers.

**Compute optimized<sup>3</sup>** – Compute optimized VMs have a high CPU-to-memory ratio and are good for medium traffic web servers, network appliances, batch processes, and application servers. These VMs can also support machine learning workloads that cannot benefit from GPU-based VMs.

**Memory optimized<sup>4</sup>** -. These VMs provide high memory-to-CPU ratio. These VMs cover a broad range of CPU and memory options (all the way up to 4 TB of RAM) and are well suited for most database workloads.

**Storage optimized<sup>5</sup>** – Storage optimized VMs provide fast, local, NVMe storage that is ephemeral. This makes them a good candidate for scale-out data workloads such as Cassandra. It is possible to use them with SQL Server, however since the storage is ephemeral, you would need to ensure you configured data protection using a feature like Always On Availability Groups or Log Shipping.

**GPU<sup>6</sup>** – Azure VMs with GPUs are targeted at two main types of workloads—naturally graphics processing operations like video rendering and processing, but also massively parallel machine learning workloads that can take advantage of GPUs.

**High performance 7compute<sup>8</sup>** – High Performance Compute workloads support applications that can scale horizontally to thousands of CPU cores. This support is provided by high performance CPU and remote direct memory access (RDMA) networking that provides low latency communications between VMs.

The easiest way to see the sizing options within each series is through the Azure Portal. From the blade for creating a VM, you can click the option to "Select Size" and see a list like the one shown in Figure 2.

---

<sup>1</sup> <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-general>

<sup>2</sup> <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-general>

<sup>3</sup> <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-compute>

<sup>4</sup> <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-memory>

<sup>5</sup> <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-storage>

<sup>6</sup> <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-gpu>

<sup>7</sup> <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-hpc>

<sup>8</sup> <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-hpc>

Select a VM size										
VM size ↑↓	Offering ↑↓	Family ↑↓	vCPUs ↑↓	RAM (GiB) ↑↓	Data disks ↑↓	Max IOPS ↑↓	Temporary storage (GiB) ↑↓	Premium disk support ↑↓		
A6	Standard	General purpose	4	28	8	8x500		No		
A7	Standard	General purpose	8	56	16	16x500		No		
A8_v2 ⓘ	Standard	General purpose	8	16	16	16x500	80	No		
A8m_v2 ⓘ	Standard	General purpose	8	64	16	16x500	80	No		
B12ms ⓘ	Standard	General purpose	12	48	16	4320	96	Yes		
B16ms ⓘ	Standard	General purpose	16	64	32	4320	128	Yes		
B11s ⓘ	Standard	General purpose	1	0.5	2	160	4	Yes		
B1ms ⓘ	Standard	General purpose	1	2	2	640	4	Yes		
B1s ⓘ	Standard	General purpose	1	1	2	320	4	Yes		
B20ms ⓘ	Standard	General purpose	20	80	32	4320	160	Yes		
B2ms ⓘ	Standard	General purpose	2	8	4	1920	16	Yes		
B2s ⓘ	Standard	General purpose	2	4	4	1280	8	Yes		
B4ms ⓘ	Standard	General purpose	4	16	8	2880	32	Yes		
B8ms ⓘ	Standard	General purpose	8	32	16	4320	64	Yes		
D1_v2 ⓘ	Standard	General purpose	1	3.5	4	4x500	50	No		
D11_v2 ⓘ	Standard	Memory optimized	2	14	8	8x500	100	No		
D12_v2 ⓘ	Standard	Memory optimized	4	28	16	16x500	200	No		
D13_v2 ⓘ	Standard	Memory optimized	8	56	32	32x500	400	No		
D14_v2 ⓘ	Standard	Memory optimized	16	112	64	64x500	800	No		
D15_v2 ⓘ	Standard	Memory optimized	20	140	64	64x500	1000	No		
D16_v3 ⓘ	Standard	General purpose	16	64	32	32x500	400	No		
D16s_v3 ⓘ	Standard	General purpose	16	64	32	25600	128	Yes		

Figure 2: A partial list of the VM sizes available through the Azure Portal

Figure 2 shows just a small set of the series and size possibilities. Note that for each option, you can see the number of Virtual CPUs, the amount of RAM, the number of Data disks, the Max IOPS, the temporary storage provided and whether or not Premium storage is supported.

## High Availability

One of the major benefits of cloud computing is that platform high availability is part of the architecture. Azure provides a high-level of built hardware, storage, and networking redundancy. High availability for a system is typically measured as a percentage of uptime per year. In Table 1, you can see what those numbers translate into in terms of time.

Availability %	Downtime per Year
<b>99% ("two nines")</b>	3.65 days
<b>99.5% ("two and a half nines")</b>	1.83 days
<b>99.9% ("three nines")</b>	8.77 hours
<b>99.95% ("three and a half nines")</b>	4.38 hours
<b>99.99% ("four nines")</b>	52.60 minutes
<b>99.995% ("four and a half nines")</b>	26.30 minutes
<b>99.999% ("five nines")</b>	5.26 minutes

Table 1 Availability Percentage and Downtime per Year

A single Azure Virtual Machine provides three nines (99.9%) of high availability when used in conjunction with Azure managed storage. This means that the service guarantees the availability of the virtual machine up to 99.9% of the time, which translates into a downtime of no more than 8.77 hours each year.

In addition to the default number of nines, there are additional features that you can include with your SQL Server in Azure virtual machine to ensure that you achieve the maximum amount of up time.

# Azure Platform Availability

Beyond its built-in high availability, the Azure platform offers two options for providing higher levels of availability for VM and some PaaS workloads. Availability Zones and Availability Sets protect your workloads from planned maintenance activity and potential hardware failures.

## Availability Zones

Availability Zones are unique physical locations within a region. Each zone is made up of one or more datacenters equipped with independent power, cooling, and networking. Within Azure regions that support Availability Zones, you can specify in which zone you want the virtual machine to reside when you choose to use Available Zones during VM creation. There are three availability zones within each supported Azure region. Availability zones provide high availability against data center failures when you deploy multiple VMs into different zones. In addition, they also provide a means for Microsoft to perform maintenance (using a grouping called an update domain) within each region by only updating one zone at any given time. You can spread out your virtual machine ecosystem across three zones in the region. Utilizing availability zones in conjunction with your Azure virtual machines raises your uptime to four nines (99.99%) which equates to a maximum of 52.60 minutes of downtime per year. You can identify which Azure regions support Availability Zones at [docs.microsoft.com<sup>9</sup>](https://docs.microsoft.com/en-us/azure/availability-zones/az-overview). If Availability Zones are available in your region, and your application can support the minimal cross-zone latency, Availability Zones will provide the highest level of availability for your application.

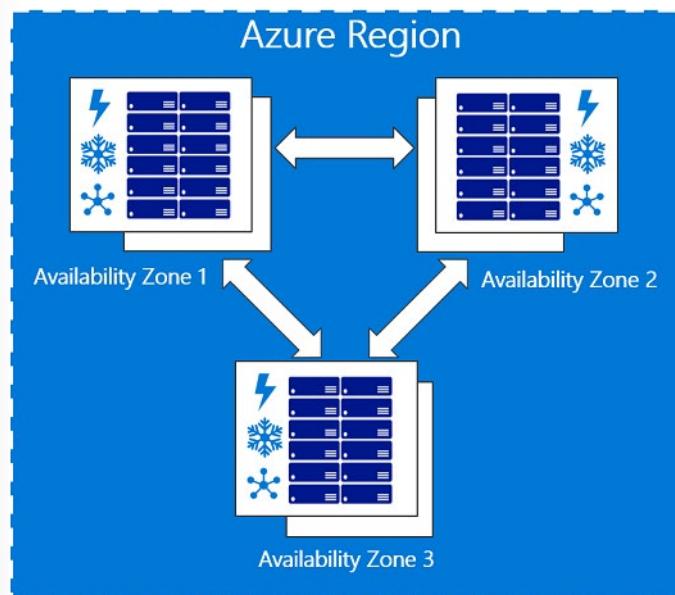


Figure 3 Azure Availability Zones

In Figure 3, you can see the availability zone configuration. When you deploy a VM into a region with an availability zone you will be presented with the option to deploy in Zone 1, 2, and 3. These zones are logical representations of physical data centers, which means a deployment to Zone 1 in one subscription, does not mean that Zone 1 represents the same data center in another subscription.

---

<sup>9</sup> <https://docs.microsoft.com/en-us/azure/availability-zones/az-overview>

## *Availability Sets*

Availability sets are similar to availability zones, except instead of spreading workloads across data centers in a region, they spread workloads across servers and racks in a data center. Since nearly all workloads in Azure are virtual, you can use availability sets in order to guarantee that the two VMs containing your Always On Availability Group members are not running on the same physical host. Availability sets can provide up to 99.95% availability, and should be used when Availability Zones are unavailable in a region, or an application cannot tolerate intra-zone latency.

## **SQL Server High Availability Options**

SQL Server provides two major options for high availability: Always On Availability Groups and Failover Cluster Instances. In addition, using Availability Groups for your Virtual Machines also provides the option to allow for disaster recovery.

### *Always On Availability Groups (AG)*

Always On Availability Groups can be implemented between two or more (up to a maximum of nine) SQL Server instances running on Azure virtual machines or across an on-premises data center and Azure. In an Availability Group database transactions are committed to the primary replica, and then the transactions are sent either synchronously or asynchronously to all secondary replicas. The physical distance between the servers (i.e. whether or not they are in the same Azure region) dictates which availability mode you should choose. Generally, if the workload requires the lowest possible latency or the secondary replicas are geographically spread apart, asynchronous availability mode is recommended. If the replicas are within the same Azure region and the applications can withstand some level of latency, synchronous commit mode will help to ensure that each transaction is committed to one or more secondaries prior to allowing the application to continue. Always On Availability groups provide both high availability and disaster recovery, because a single availability group can support both synchronous and asynchronous availability modes. You should note that the unit of failover for an availability group is a group of databases, and not the entire instance.

### *SQL Server Failover Cluster instances*

If you need to protect the entire instance, you could use a SQL Server Failover Cluster Instance (FCI), which provides high availability for an entire instance, in a single region, but does not provide disaster recovery without being combined with another feature like Availability Groups or Log Shipping (which will be covered in detail in module 7). FCIs also require shared storage which can be provided on Azure by using shared file storage or using Storage Spaces Direct on Windows Server.

For Azure workloads Availability Groups are the preferred solution for newer deployments, because the shared storage requirement of FCIs increases the complexity of deployments. However, for migrations from on-premises solutions, an FCI may be required for application support.

## **SQL Server Disaster Recovery Options**

While the Azure platform offers 99.9% up time by default, disasters can still occur and affect application uptime. It is imperative that you have a proper disaster recovery plan in place when doing any type of migration. Azure offers us several methods to ensure that your SQL Server on a virtual machine is protected in the event of a disaster. There are two components to this—Azure platform options like geo-replicated storage for backups and Azure Site Recovery (which is an all-encompassing disaster recovery solution for all of your workloads), and SQL Server specific offerings like Availability Groups and backups.

## Native SQL Server Backups

Backups are considered the life blood of any database administrator and it is not any different when working with a cloud solution. With SQL Server on an Azure virtual machine, you have granular control of when backups occur and where they are stored. You can utilize SQL agent jobs to backup directly to a URL linked to Azure blob storage. Azure provides the option to use geo-redundant storage(GRS) or read-access geo-redundant storage (RA-GRS) to ensure that your backup files are safely tucked away across the geographic landscape.

Additionally as part of the Azure SQL VM service provider, you can have your backups automatically managed by the platform.

## Azure Backup for SQL Server

The Azure Backup solution requires an agent to be installed on the virtual machine. The agent then communicates with an Azure service that manages automatic backups of your SQL Server databases. Azure Backup also provides a central location that you can utilize to manage and monitor the backups to ensure meeting any specified RPO/RTO metrics.

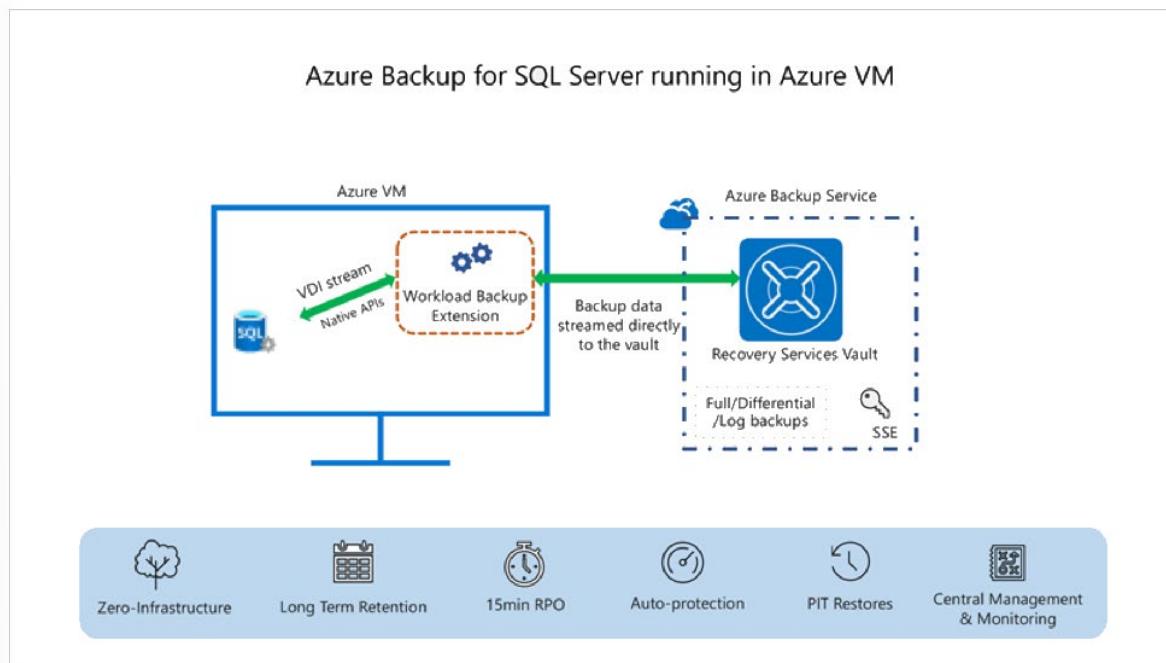


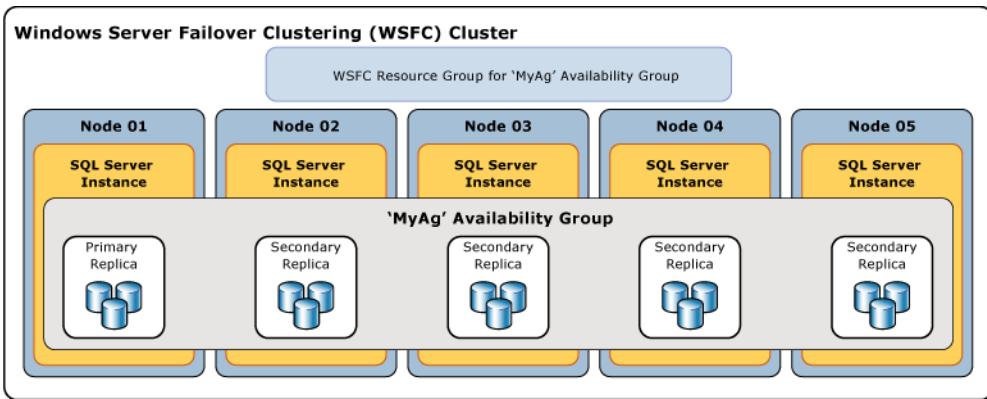
Figure 4 Azure Backup for SQL Server Architecture

As shown in Figure 4, the Azure Backup solution is a comprehensive, enterprise backup solution that provides long-term data retention, automated management and additional data protection. This option costs more than simply performing your own backups, or using the Azure resource provider for SQL Server, but does offer a more complete backup feature set.

## Availability Groups

In addition to the high availability scenarios described above, Always On Availability Groups can be used for disaster recovery purposes. You can implement up to nine replicas of a database across Azure regions, and stretch this architecture even further using Distributed Availability Groups, which will be covered in module 7. This ensures that a viable copy of your database(s) is in another location beyond the primary

region. By doing so you help to ensure that your data ecosystem is protected against natural disasters as well as human made ones.



*Figure 5 AlwaysOn Availability Group Configuration*

Figure 5 shows a logical diagram of an Always On Availability Group, running on a Windows Server Failover Cluster. There is one primary and four secondary replicas. In this scenario all five replicas could be synchronous, or some combination of synchronous and asynchronous replicas. You should make note that as mentioned above, the unit of failover is the group of databases and not the instance. While a failover cluster instance provides HA at an instance level, it does not provide disaster recovery.

## Azure Site Recovery (ASR)

Azure Site Recovery is a low-cost solution that will perform block level replication of your Azure virtual machine. This service offers various options, including the ability to test and verify your disaster recovery strategy. This solution is best utilized for stateless environments (e.g. web servers) versus transactional database virtual machines. Azure Site Recovery supports SQL Server environments, but currently has a relatively high recovery point objective (up to an hour) making it better suited for migrations with some allowed downtime. A more common use case is to have all your stateless applications using Azure Site Recovery in conjunction with Availability Groups to provide a lower RPO.

## Provisioning and Deployment

The Azure ecosystem offers several different methods in which to provision a new SQL Server instance on an Azure virtual machine. Each method provides different capabilities, such as the ability to have a repeatable process (via scripting languages like PowerShell) or a method to accomplish the goal without needing an understanding of programming constructs, via the Azure portal.

## Azure Marketplace

The Azure Marketplace is essentially a centralized location that provides the ability to create Azure resources based on a predesigned template. For example, you can quickly create a SQL Server 2019 instance on Windows Server 2019 with a couple of clicks of the mouse along with some basic information, such as the virtual machine name as well as some SQL Server configuration information. Once provided, Azure Resource Manager (ARM) will initiate the creation of the virtual machine and within minutes it will be up and running.

The blade for SQL Server 2019 on Windows Server 2019 in the Azure marketplace is shown in figure 6. This blade gives you the option of pre-set configurations that support OLTP or Data Warehouse workloads and allow you to specify storage, patching, and backup options.



Figure 6 Azure Portal SQL Server VM Creation

The disadvantage of using the portal to create Azure resources is that it is not an easily repeatable process. However, it is very easy to get started with the portal, and using the portal a new administrator can quickly get up and running.

## Storage Considerations

SQL Server requires good storage performance to deliver robust application performance, whether it be an on-premises instance or installed in an Azure VM. Azure provides a wide variety of storage solutions to meet the needs of your workload. While Azure offers various types of storage (blob, file, queue, table) in most cases SQL Server workloads will use Azure managed disks. The exceptions are that a Failover Cluster Instance can be built on file storage and backups will use blob storage. Azure managed disks act as a block-level storage device that is presented to your Azure VM. Managed disks offer a number of benefits including 99.999% availability, scalable deployment (you can have up to 50,000 VM disks per subscription per region), and integration with availability sets and zones to offer higher levels of resiliency in the event of failure.

Azure managed disks all offer two types of encryption. Azure Server side encryption is provided by the storage service and acts as encryption-at-rest provided by the storage service. Azure Disk Encryption uses BitLocker on Windows, and DM-Crypt on Linux to provide OS and Data disk encryption inside of the VM. Both technologies integrate with Azure Key Vault and allow you to bring your own encryption key.

Each VM will have at least two disks associated with it: the operating system disk and the temporary disk.

**Operating System disk** – Each virtual machine will require an operating system disk which contains the boot volume. This would be the C: drive in the case of a Windows platform virtual machine, or /dev/sda1 on Linux. The operating system will be automatically installed on the operating system disk.

**Temporary disk** – Each virtual machine will include one disk used for temporary storage. This storage is intended to be used for data that does not need to be durable, such as page files or swap files. Because the disk is temporary, you should not use it for storing any critical information like database or transaction log files as they will be lost during maintenance or a reboot of the virtual machine. This drive will be mounted as D:\ on Windows, and /dev/sdb1 on Linux.

Additionally, you can and should add additional data disks to your Azure VMs running SQL Server.

Data disks – The term data disk is used in the Azure Portal, but in practice these are just additional managed disks added to a VM. These disks can be pooled to increase the available IOPs and storage capacity, using Storage Spaces on Windows or Logical Volume Management on Linux.

Furthermore, each disk can be one of several types:

- **Standard HDD** were the original storage offering on Azure and offer cost-effective storage for non I/O intensive workloads. A common use case for standard disks are for SQL Server backups.
- **Standard SSD** is a solid state drive and will have similar latency and IOPS to the standard HDD drives at sizes up to 4 TB; however, this type offers significant performance gains at larger volumes. Standard SSDs do offer guaranteed performance levels where the Standard HDD disks do not.
- **Premium SSD** is the most commonly used type of disk for SQL Server workloads. They are available in all regions and support a wide variety of VM types. Premium disks strike a good balance between price and performance.
- **Ultra SSD** provides the lowest latency (sub-millisecond) and the highest potential IOPs. Ultra SSD allows you to configure IOPs, storage volume, and bandwidth independently for more granular cost control.

The best practices for SQL Server on Azure recommend using Premium Disks pooled for increased IOPs and storage capacity. Data files should be stored in their own pool with read-caching on the Azure disks. Transaction log files will not benefit from this caching, so those files should go into their own pool without caching. TempDB can optionally go into its own pool, or using the VM's temporary disk, which offers low latency since it is physically attached to the physical server where the VMs are running. Properly configured Premium SSD will see latency in single digit milliseconds. For mission critical workloads that require latency lower than that, you should consider Ultra SSD.

## *Deploying via PowerShell/CLI*

PowerShell is an object-oriented programming language that can be used to quickly and efficiently create a virtual machine (as well as other resources) within the Azure ecosystem. Utilizing PowerShell gives you granular control over various aspect of the virtual machine such as size, name, IP address, and even storage.

```

1 # Variables for common values
2 $resourceGroup = "myResourceGroup"
3 $location = "westeurope"
4 $vmName = "myVM"
5
6 # Create user object
7 $cred = Get-Credential -Message "Enter a username and password for the virtual machine."
8
9 # Create a resource group
10 New-AzResourceGroup -Name $resourceGroup -Location $location
11
12 # Create a virtual machine
13 New-AzVM ` 
14     -ResourceGroupName $resourceGroup ` 
15     -Name $vmName ` 
16     -Location $location ` 
17     -ImageName "Win2016Datacenter" ` 
18     -VirtualNetworkName "myVnet" ` 
19     -SubnetName "mySubnet" ` 
20     -SecurityGroupName "myNetworkSecurityGroup" ` 
21     -PublicIpAddressName "myPublicIp" ` 
22     -Credential $cred ` 
23     -OpenPorts 3389

```

\*Figure 7 PowerShell sample to create a Resource Group and an Azure VM\*

Figure 7 shows a PowerShell script that defines parameters and creates an Azure resource group and a virtual machine with a predefined virtual network.

Azure Command Line Interface (CLI) is another method of using scripting to create Azure resources, including virtual machines. Its simplistic nature allows you to accomplish the same goal as with PowerShell in fewer lines of actual code.

```
1 az group create --name myResourceGroup --location eastus
2
3 az vm create \
4   --resource-group myResourceGroup \
5   --name myVM \
6   --image win2016datacenter \
7   --admin-username azureuser
```

Figure 8 Azure CLI to Create a Resource Group and a VM

As Figure 8 shows, the CLI is less verbose in terms of the amount of code required to create new resources.

## Deploying using ARM Templates

Azure ARM Templates have the benefit of being able to deploy a full set of resources in one single declarative template. This includes the ability to build dependencies into the templates, as well as using parameters to change specific values at deployment time. Once you have a template, there are several ways with which you can deploy it including an Azure DevOps pipeline, or through the custom deployments blade in the Azure Portal. You can use the Azure Portal to export an ARM template to JSON for future deployment.

```
1  {
2    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
3    "contentVersion": "1.0.0.0",
4    "parameters": {
5      "adminUsername": {
6        "type": "string",
7        "metadata": {
8          "description": "Username for the Virtual Machine."
9        }
10     },
11     "adminPassword": [
12       {
13         "type": "securestring",
14         "metadata": {
15           "description": "Password for the Virtual Machine."
16         }
17       ],
18       "dnslabelPrefix": {
19         "type": "string",
20         "metadata": {
21           "description": "Unique DNS Name for the Public IP used to access the Virtual Machine."
22         }
23     },
24     "windowsOSVersion": {
25       "type": "string",
26       "defaultValue": "2016-Datacenter",
27       "allowedValues": [
28         "2008-R2-SP1",
29         "2012-Datacenter",
30         "2012-R2-Datacenter",
31         "2016-Nano-Server",
32         "2016-Datacenter-with-Containers",
33         "2016-Datacenter",
34         "2019-Datacenter"
35       ],
36       "metadata": {
37         "description": "The Windows version for the VM. This will pick a fully patched image of this given Windows version."
38     }
39   }
```

Figure 9 Example ARM Template

Figure 9 is an example of an ARM template that can be used to deploy a group of Azure resources repeatedly. ARM templates will be covered in further detail in Module 6. Templates can be deployed directly from the Azure portal, or via scripting languages that reference the templates like PowerShell, the Azure CLI, or through DevOps workflows like Azure DevOps.

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"

.setLocation = Read-Host -Prompt "Enter the location (i.e. centralus)"

New-AzResourceGroup -Name $resourceGroupName -Location $location

New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName `

-TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-tem-
plates/master/101-storage-account-create/azuredploy.json

az deployment group create --resource-group myresourcegroup `

--TemplateUri "https://raw.githubusercontent.com/Azure/azure-quick-
start-templates/master/101-sql-logical-server/azuredploy.json"
```

Figure 10 ARM Template Deployment using PowerShell and Azure CLI

Figure 10 highlights the PowerShell and Azure CLI code to deploy an Azure Resource Group and prompts the user for the resource group name and the target Azure region.

## Migration

Azure Infrastructure as a Service (IaaS) solutions are frequently used when migrating your on-premises environment to the cloud. There are several tools available to help with the migration process. This next section looks at some of the tools and methods for migration.

### Azure Migrate Server Tool

This migration tool provides a centralized location to assess and migrate on-premises servers, infrastructure, applications and data to Azure. It will provide discoverability and proper assessments of your servers regardless of whether they are physical or VMWare/Hyper-V virtual machines.

Azure Migrate will also help to ensure that you select the appropriate size of virtual machine so that workloads will have enough resources available. In addition, the tool will provide a cost estimation so that you can budget accordingly.

In order to utilize the Azure Migrate tool, you must deploy a light-weight appliance which can be deployed on a virtual or physical machine. Once the on-premises servers are discovered, the appliance will continually send metadata about each server (along with performance metrics) to Azure Migrate, which resides in the cloud.

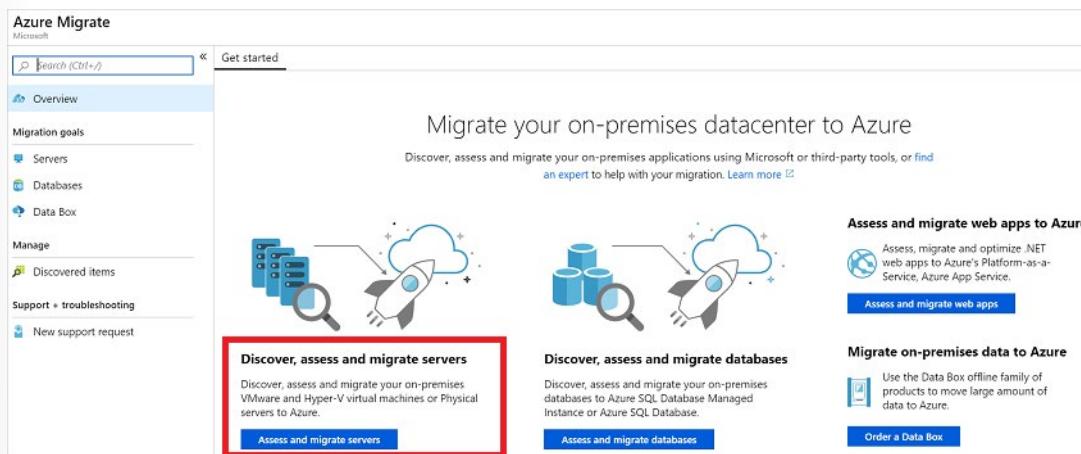


Figure 11 Azure Migrate portal options

As shown in the Figure 11, the Azure Migrate experience can be kicked off from the portal to begin your migration process. The service consists of a unified migration platform which provides a single portal to track your entire migration to Azure.

There are several additional tools you can use to map your server estate and identity compatibility with your target Azure platform:

- MAP Toolkit—The Microsoft Assessment and Planning Toolkit automatically gathers an inventory of all of the SQL Servers in your network, gathering version and server information and providing reports on the information gathered.
- Database Experimentation Assistant—This tool can be used to evaluate version upgrades of SQL Server by checking syntax compatibility and provides a platform to evaluate query performance on the target version.

## *Data Migration Assistant*

The MAP toolkit and Database Experimentation assistant can help you identify your databases and highlight any incompatibilities or potential performance issues in your database, but the Data Migration Assistant (DMA) is a comprehensive toolkit that assesses, identifies new features you can use to benefit your application, and ultimately performs the migration. This tool can be used to migrate between versions of SQL Server, from on-premises to an Azure VM or Azure SQL Database or Azure SQL Managed Instance.

One of the main benefits of the DMA is the ability to assess queries both from Extended Event trace files and SQL queries from an external application, for example T-SQL queries in the C# application code for your application. You can generate a full report using a C# source and upload the migration assessment to the DMA. The DMA mitigates the risk of moving to a newer version of SQL Server or to Azure SQL Database.

## *Azure Database Migration Service*

The Azure Database Migration Service (DMS) is designed to support a wide mix of different migration scenarios with different source and target databases, and both offline (one-time) and online (continuous data sync) migration scenarios. The DMS supports migrating to and from SQL Server to Azure SQL Database and Azure SQL Managed Instance. The offline source and target pairs are shown in Table 2 below:

Target	Source
<b>Azure SQL DB</b>	SQL Server
	RDS SQL
	Oracle
<b>Azure SQL DB MI</b>	SQL Server
	RDS SQL
	Oracle
<b>Azure SQL VM</b>	SQL Server
	Oracle
<b>Azure Cosmos DB</b>	MongoDB
<b>Azure DB for MySQL</b>	MySQL
	RDS MySQL
<b>Azure DB for PostgreSQL</b>	PostgreSQL
	RDS PostgreSQL

Table 2 Offline Migration Source/Targets for DMS

The source and target pairs for online migration are shown in Table 3 below:

Target	Source
<b>Azure SQL DB</b>	SQL Server
	RDS SQL
	Oracle
<b>Azure SQL DB MI</b>	SQL Server
	RDS SQL
	Oracle
<b>Azure SQL VM</b>	SQL Server

Target	Source
	Oracle
<b>Azure Cosmos DB</b>	MongoDB
<b>Azure DB for MySQL</b>	MySQL
	RDS MySQL
<b>Azure DB for PostgreSQL</b>	PostgreSQL
	RDS PostgreSQL
	Oracle

Table 3 Online Migration Source/Targets for DMS

The Data Migration Service has a few prerequisites that are common across migration scenarios. You need to create a virtual network in Azure, and if your migration scenarios involve on-premises resources, you will need to create a VPN or ExpressRoute connection from your office to Azure. There are a number of network ports that are required for connectivity. Once the prerequisites are in place, the time to complete migration will depend on the data volume and rate of change in the databases in question.

There are a number of traditional, more manual approaches to migrating databases to Azure including backup and restore, log shipping, replication, and adding an Availability Group replica in Azure. Several of these technologies will be discussed in Module 7. These solutions were not designed primarily for performing migrations, but they can be used for that purpose. The technique you use for physically migrating your data will depend on the amount of downtime you can sustain during the migration process.

## Summary and Knowledge Check

Azure provides a great deal of flexibility for deploying SQL Server to an Azure virtual machine. The hybrid licensing options reduce your cost and allow for additional protection for high availability and disaster recovery as part of your software assurance benefits. Azure Resource Manager offers a number of ways to deploy your resources in both a programmatic and graphical fashion. Azure and SQL Server include several tools to make it easy to migrate your data.

### Question 1

*What type of storage offers the lowest latency in Azure?*

- Ultra Disk
- Premium Disk
- Standard SSD

### Question 2

*Which tool would you use to assess and migrate your databases from an on-premises SQL Server to an Azure VM?*

- Microsoft Assessment and Planning Toolkit
- Database Experimentation Assistant
- Data Migration Assistant

## Question 3

*To reduce the cost of an Azure SQL Server VM you intend to run full time for 3 years, which option should you choose?*

- Azure Reserved VM Instances
- Availability Set
- Pay as You Go Licensing

# Deploying a Platform as a Service Solution

## Introduction

The Platform as a Service (PaaS) offering for SQL Server can be a great solution for certain workloads. The PaaS offering provides less granular control over the infrastructure and delegates management of the underlying components (memory, CPU, storage, operating system, etc) to the cloud provider, namely Microsoft.

This lesson will focus on ways to provision and deploy both Azure SQL Database and SQL Server Managed Instances, as well as provide clear and definitive guidance on the various options when performing a migration to this platform.

## Learning Objectives

In this lesson you will:

- Gain an understanding SQL Server in a Platform as a Service (PaaS) offering
- Understand PaaS provisioning and deployment options
- Understand elastic pools
- Examine Managed Instances
- Configure a template for PaaS deployment

## Overview of Platform as a Service (PaaS)

Platform as a Service (PaaS) provides a complete development and deployment environment in the cloud which can be used for simple cloud-based applications as well as for advanced enterprise applications.

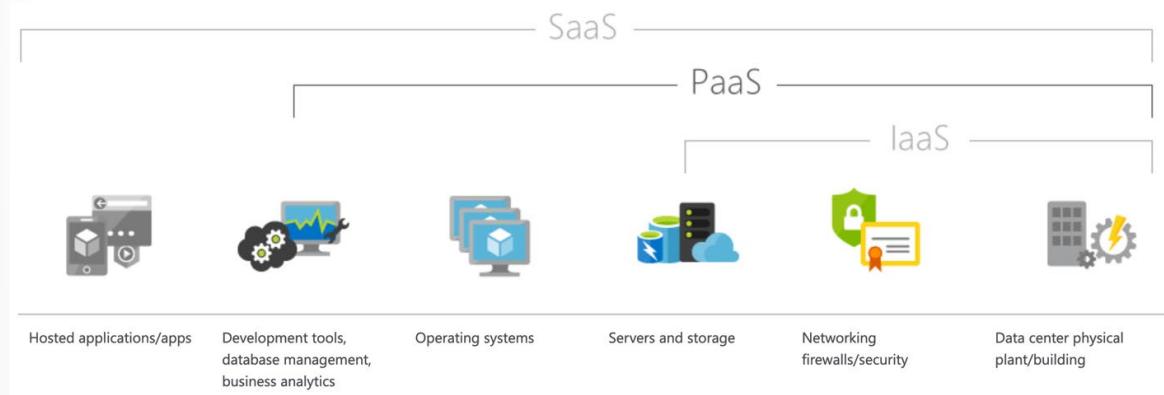


Figure 12 Platform Management for PaaS Solutions

As shown in figure 12, PaaS includes cloud provided operating system and management of your solution. For example, in Azure SQL Database and Azure SQL Managed Instance, the high availability, OS and SQL Server features, and backups are all provided by the Azure platform.

We'll explore Azure SQL database in this section. Azure SQL Database is available in three different deployment options:

- A single database—A single database that is billed and managed on a per database level

- Elastic Pools—A group of databases that are managed together and share a common set of resources
- Hyperscale—a single database offering that allows databases to scale much beyond the 4 TB limit of an Azure SQL Database

Elastic Pool will be discussed later within this module. Even in the cloud all services are backed by physical hardware. The Azure SQL Database allows you to choose from two different purchasing models:

- Database Transaction Unit (DTU) – DTUs are calculated based on a formula combining compute, storage, and I/O resources.
- vCore – The vCore model allows you to purchase a specified number of vCores based on your given workloads. This is the default purchasing model when purchasing Azure SQL Database resources. vCore databases have a specific relationship between the number of cores and the amount of memory and storage provided to the database.

Azure SQL Database also includes:

- Automatic backups
- Automatic patching
- Built-in high availability
- SQL Server feature enhancements without the need to upgrade software

## *Service Tier Options*

PaaS comes in several different service tiers. Each tier has varying capabilities which allow you to have a wide range of options when choosing this platform.

The DTU model is available in three different service tiers:

- Basic
- Standard
- Premium

You can purchase vCore databases in three different service tiers as well:

- General Purpose – This tier is for general purpose workloads. It is backed by Azure premium storage. It will have higher latency than Business Critical
- Business Critical – This tier is for high performing workloads offering the lowest latency of either service tier. This tier is backed by local SSDs instead of Azure blob storage. It also offers the highest resilience to failure as well as providing a built-in read-only database replica that can be used to off-load reporting workloads
- Hyperscale—Hyperscale databases can scale far beyond the 4 TB limit the other Azure SQL Database offerings and have a unique architecture that supports databases of up to 100 TB

## *Backups*

One of the most important features of the Platform as a Service offering is backups. In this case, backups are performed automatically without any intervention from you. Backups are stored in Azure blob geo-redundant storage and by default are retained for between seven and 35 days, based on the service tier of the database. Basic and vCore databases default to seven days of retention, and on the vCore databases this can be adjusted by the administrator. This can be extended by configuring long-term retention (LTR), which would allow you to retain backups for up to 10 years. In order to provide redun-

dancy, you are also able to utilize read-accessible geo-redundant blob storage. This storage would replicate your database backups to a secondary region. It would also allow you to read from that secondary region if needed. It is worth stating that manual backups of databases is not permitted and the platform will deny any request to do so.

Database Backups are taken on a given schedule:

- Full – Once a week
- Differential – Every 12 hours
- Log – Every 5-10 minutes depending on transaction log activity

This backup schedule should meet the needs of most recovery point/time objectives (RPO/RTO) however each customer should evaluate whether they meet your business requirements.

If the need to restore a database arises, there are several options available. Due to the nature of Platform as a Service, you cannot manually restore a database utilizing conventional methods, such as issuing the T-SQL command RESTORE DATABASE. This will not work.

Regardless of which restore method is implemented, it is not possible to restore over an existing database. If a database needs to be restored, the existing database must be dropped or renamed prior to initiating the restore. Furthermore, keep in mind that depending on the platform service tier, restore times could fluctuate. It is recommended that you test the restore process to obtain baseline metrics on how long a restore could potentially take.

The available restore options are:

Restore using the Azure portal –Using the Azure portal you have the option of restoring a database to the same Azure SQL Database server, or you can use the restore to create a new database on a new server in any Azure region.

Restore using scripting Languages – Both PowerShell and Azure CLI can be utilized in order to restore a database.

## *Active Geo-Replication*

Geo-replication is a business continuity feature that asynchronously replicates a database to up to four secondary replicas. As transactions are committed to the primary (and its replicas within the same region), the transactions are sent to the secondaries to be replayed. Since this is done asynchronously, the calling application does not have to wait for the secondary replica to commit the transaction prior to SQL Server returning control to the caller.

The secondary databases are readable and can be used to offload read-only workloads, thus freeing up resources for transactional workloads on the primary or placing data closer to your end users. Furthermore, the secondary databases can be in the same region as the primary or in another Azure region.

With geo-replication you can initiate a failover either manually by the user or from the application. If a failover occurs, you potentially will need to update application connection strings to reflect the new endpoint of what is now the primary database.

## *Failover Groups*

Failover groups are built on top of the technology used in geo-replication, but provide a single endpoint for connection. The major reason for using failover groups is that the technology provides endpoints which can be utilized to route traffic to the appropriate replica. Your application can then connect after a failover without connection string changes.

## Serverless

The name "Serverless" can be a bit confusing as you still deploy your Azure SQL Database to a logical server, to which you connect. Azure SQL Database serverless is a compute tier that will automatically scale up or down the resources for a given database based on demand. If the workload no longer requires compute resources, the database will become "paused" and you will not be charged during the period when the database is in this state. When a connection attempt is made , the database will "resume" and become available. Resuming the database is not instantaneous.

Another difference between serverless and the normal vCore model of Azure SQL Database is that with serverless you can specify a minimum and maximum number of vCores. Memory and I/O limits are proportional to the range that is specified.

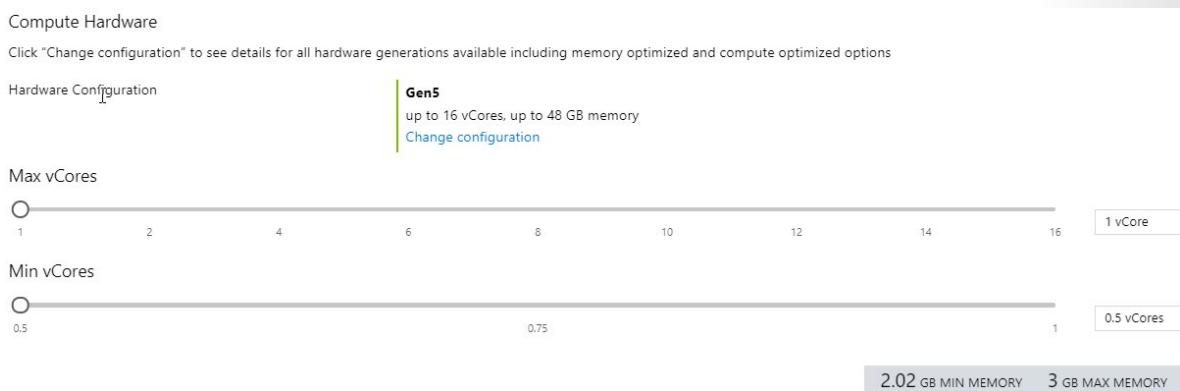


Figure 13 The Azure SQL Database Serverless Settings in the Azure Portal

Figure 13 shows the configuration screen for a serverless database in the Azure portal. You have the option have a minimum of half of a vCore all the way up to 16 vCores. You should note that databases that are not deployed as serverless are referred to as "provisioned".

The setting to control pausing is referred to as the autopause delay and has a minimum value of 60 minutes and a maximum value of 7 days. If the database has been idle for that period of time it will then pause. Once the database has been inactive for the specified amount of time, it will be paused until a subsequent connection is attempted. Any applications using serverless should be configured to handle connection errors and include retry logic, as connecting to a paused database will generate a connection error.

Serverless is not fully compatible with all features in Azure SQL Database as some features are running background processes all the time. These features include:

- Geo-replication
- Long-term backup retention
- A job database in elastic jobs (see module 6 for more detail on elastic jobs)
- The sync database in SQL Data Sync (Data Sync is a service that replicates data between a group of databases)

## Hyperscale

Azure SQL Database has been limited to 4 TB of storage per database for many years. This is due to a physical limitation of the Azure infrastructure. Azure SQL Database Hyperscale changes the paradigm and allows for databases to be 100 TB or more. Hyperscale introduces new horizontal scaling techniques to add compute nodes as the data sizes grow. The cost of Hyperscale is the same as the cost of Azure SQL

Database; however, there is a per terabyte cost for storage. You should note that once an Azure SQL Database is converted to Hyperscale, you cannot convert it back to a "regular" Azure SQL Database.

## Deploying a Singleton DB

We'll look at several methods for deploying a singleton Azure SQL Database.

### Deploying via the Portal

The process to create a singleton database through the Azure portal is straightforward. While in the portal, on the left-hand navigation menu, select "SQL Databases". In the resulting slide out dialog, click "Create" as shown in Figure 14 below.

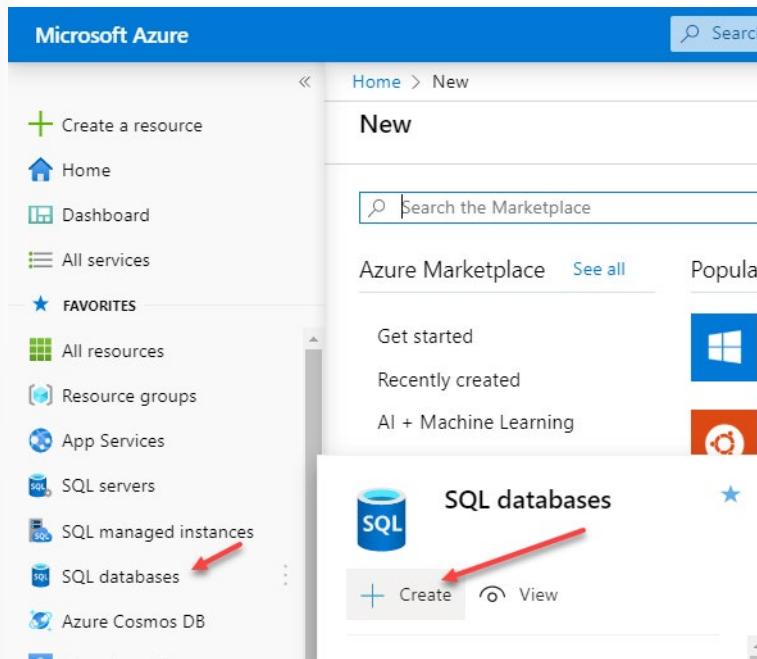


Figure 14 The Azure Portal Azure SQL Database Deployment screen

In the blade in Figure 15, below, you'll notice that the subscription should already be provided for you. You will need to supply the following information:

- Resource Group – If there is an existing resource group that you wish to use, you may select it from the drop-down list. You can click on the "Create new" option if you wish to create a new resource group for this Azure SQL Database.
- Database Name – You must provide a database name
- Server – Each database must reside on a logical server. If you have one already in existence in the appropriate region, you may choose to use it. Otherwise, you can click on the "Create new" and follow the prompts to create a new logical server to house the database.
- Determine whether to use an elastic pool, which is described later in this module.
- Determine the appropriate compute resources needed. By default, it will be a Gen5, 2vCore, with 32GB of storage until something else is selected. Click on "Configure database" to view alternate configuration options.

**Create SQL Database**

Microsoft

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#)

**Project details**

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

Subscription \* ⓘ Dev-Test-Lab

Resource group \* ⓘ Contoso

[Create new](#)

**Database details**

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name \* Widgets

Server ⓘ contososql1 (West US)

[Create new](#)

Want to use SQL elastic pool? \* ⓘ  Yes  No

Compute + storage \* ⓘ

**General Purpose**  
Gen5, 2 vCores, 32 GB storage  
[Configure database](#)

**Review + create** [Next : Networking >](#)

Figure 15 Create SQL Database blade of Azure Portal

Figure 16 shows the Portal blade in which you can configure the database options. Here you will notice that the service tier is General Purpose and compute tier is Provisioned. Provisioned implies that the compute resources are pre-allocated and billed per hour based on the number of configured vCores. The other option is Serverless which was discussed previously. Serverless is billed per second based on the number of vCores utilized.

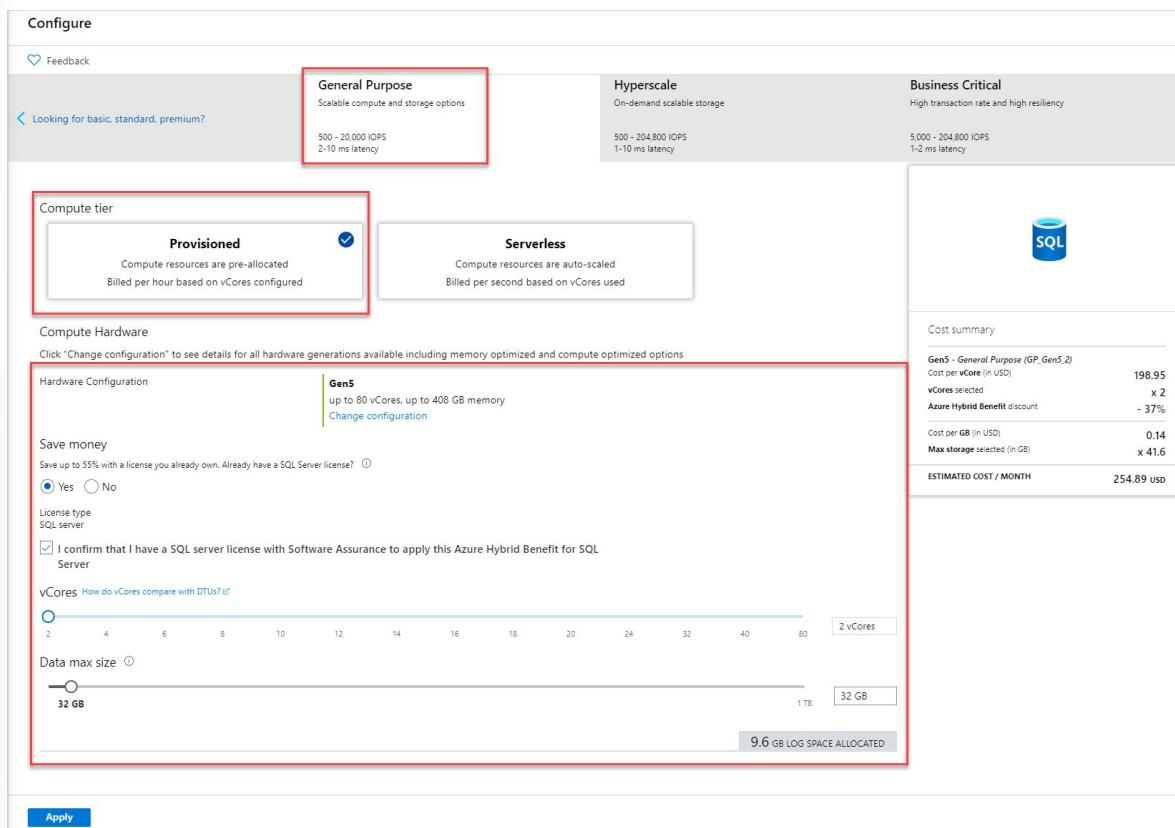


Figure 16 Service Tier selection in Azure Portal

## Deploying an Azure SQL Database via PowerShell/CLI

You can also deploy your database using Azure PowerShell or the Azure CLI. Figure 17 shows the PowerShell example where you are creating a new resource group, and defining an admin called SqlAdmin and then creating a new server, database, and firewall rule.

```
# Connect-AzAccount

# The SubscriptionId in which to create these objects
$SubscriptionId = ''

# Set the resource group name and location for your server
$resourceGroupName = "myResourceGroup-$((Get-Random))"

.setLocation = "westus2"

# Set an admin login and password for your server

$adminSqlLogin = "SqlAdmin"

$password = "ChangeYourAdminPassword1"
```

```
# Set server name - the logical server name has to be unique in the system

$serverName = "server-$($Get-Random)"

# The sample database name

$dabataseName = "mySampleDatabase"

# The ip address range that you want to allow to access your server

$startIp = "0.0.0.0"

$endIp = "0.0.0.0"

# Set subscription

Set-AzContext -SubscriptionId $subscriptionId

# Create a resource group

$resourceGroup = New-AzResourceGroup -Name $resourceGroupName -Location
$location

# Create a server with a system wide unique server name

$server = New-AzSqlServer -ResourceGroupName $resourceGroupName `

-ServerName $serverName `

-Location $location `

-SqlAdministratorCredentials $(New-Object -TypeName System.Management.
Automation.PSCredential -ArgumentList $adminSqlLogin, $(ConvertTo-Secure-
String -String $password -AsPlainText -Force))

# Create a server firewall rule that allows access from the specified IP
range

$serverFirewallRule = New-AzSqlServerFirewallRule -ResourceGroupName $re-
sourceGroupName `

-ServerName $serverName `
```

```
-FirewallRuleName "AllowedIPs" -StartIpAddress $startIp -EndIpAddress  
$endIp  
  
# Create a blank database with an S0 performance level  
  
$database = New-AzSqlDatabase -ResourceGroupName $resourceGroupName `  
-ServerName $serverName `  
-DatabaseName $databaseName `  
-RequestedServiceObjectiveName "S0" `  
-SampleName "AdventureWorksLT"
```

*Figure 17 Deploying an Azure SQL Database using PowerShell*

The Azure CLI can also be used to deploy an Azure SQL Database as shown below in Figure 18.

```
#!/bin/bash  
  
# set execution context (if necessary)  
  
az account set --subscription <replace with your subscription name or id>  
  
# Set the resource group name and location for your server  
  
resourceGroupName=myResourceGroup-$RANDOM  
  
location=westus2  
  
# Set an admin login and password for your database  
  
adminlogin=ServerAdmin  
  
password=`openssl rand -base64 16`  
  
# password=<EnterYourComplexPasswordHere1>  
  
# The logical server name has to be unique in all of Azure  
  
servername=server-$RANDOM
```

```
# The ip address range that you want to allow to access your DB

startip=0.0.0.0

endip=0.0.0.0


# Create a resource group

az group create \
    --name $resourceGroupName \
    --location $location


# Create a logical server in the resource group

az sql server create \
    --name $servername \
    --resource-group $resourceGroupName \
    --location $location \
    --admin-user $adminlogin \
    --admin-password $password


# Configure a firewall rule for the server

az sql server firewall-rule create \
    --resource-group $resourceGroupName \
    --server $servername \
    -n AllowYourIp \
    --start-ip-address $startip \
    --end-ip-address $endip
```

```
# Create a database in the server

az sql db create \
    --resource-group $resourceGroupName \
    --server $servername \
    --name mySampleDatabase \
    --sample-name AdventureWorksLT \
    --edition GeneralPurpose \
    --family Gen4 \
    --capacity 1 \

# Echo random password

echo $password
```

Figure 18 Deploying an Azure SQL Database using CLI

## Deploying Azure SQL Database Using ARM Templates

Another method for deploying resources is as mentioned earlier is using an ARM template. An ARM template gives you the most granular control over your resources, and Microsoft provides a GitHub repository called "Azure-Quickstart-Templates" which hosts ARM templates that you can reference in your deployments. A PowerShell example of deploying a GitHub based template is shown in Figure 19:

```
#Define Variables for parameters to pass to template

$ projectName = Read-Host -Prompt "Enter a project name"

$ location = Read-Host -Prompt "Enter an Azure location (i.e. centralus)"

$ adminUser = Read-Host -Prompt "Enter the SQL server administrator user-
name"

$ adminPassword = Read-Host -Prompt "Enter the SQL server administrator
password" -AsSecureString

$ resourceGroupName = "${projectName}rg"
```

```
#Create Resource Group and Deploy Template to Resource Group

New-AzResourceGroup -Name $resourceGroupName -Location $location

New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName `

-TemplateUri "https://raw.githubusercontent.com/Azure/azure-quick-`

start-templates/master/101-sql-logical-server/azureddeploy.json" `

-administratorLogin $adminUser -administratorLoginPassword $adminPassword

Read-Host -Prompt "Press [ENTER] to continue ..."
```

*Figure 19 Deploying a GitHub-based template*

## Elastic Pools

Elastic pools are a deployment option in which you purchase Azure compute resources (CPU, memory, and storage) that is then shared among multiple databases defined as belonging to the same pool. An easy comparison to an on-premises SQL Server is that an elastic pool is like a SQL Server instance that has multiple user databases. By using elastic pools, you can easily manage pool resources while at the same time potentially saving costs. Elastic pools also facilitate easy scalability up to the set limits such that if a single database within the pool needs resources due to an unpredictable workload, the resources are there. If the entire pool needs additional resources, a simple slider option within the Azure portal will facilitate scaling the elastic pool up or down.

### Creating New Pools

Creating new SQL elastic pools from the Azure portal is straightforward. From the portal click “Create a Resource” and then search for “SQL Elastic database pool” and you will see the screen in Figure 20.

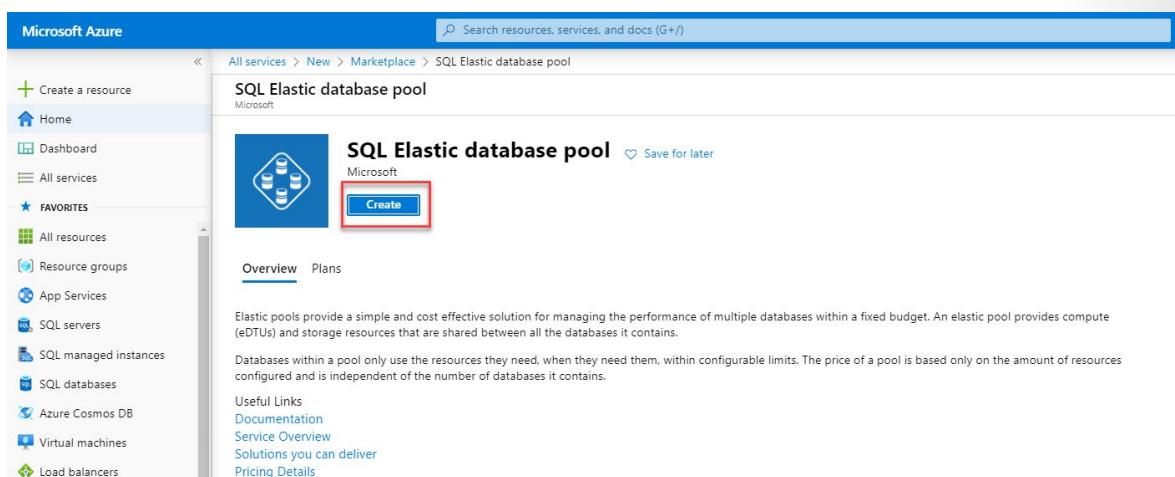


Figure 20 The Create Elastic Pool screen in the Azure Portal

Click "Create" shown in Figure 20 in order to launch the screen in Figure 21.

**Create SQL Elastic pool**

Microsoft

Basics Tags Review + create

Create a SQL Elastic pool with your preferred configurations. Elastic pools provide a simple and cost effective solution for managing the performance of multiple databases within a fixed budget. Complete the Basic tab, then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#)

**Project details**

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

Subscription \* ⓘ Dev-Test-Lab

Resource group \* ⓘ Contoso

Create new

**Elastic pool details**

Enter required settings for this pool, including picking a logical server and configuring the compute and storage resources.

Elastic Pool Name \* Enter an elastic pool name

Server \* ⓘ contososql1 (West US)

Create new

Compute + storage \* ⓘ

**General Purpose**  
Gen5, 2 vCores, 32 GB, 0 databases  
[Configure elastic pool](#)

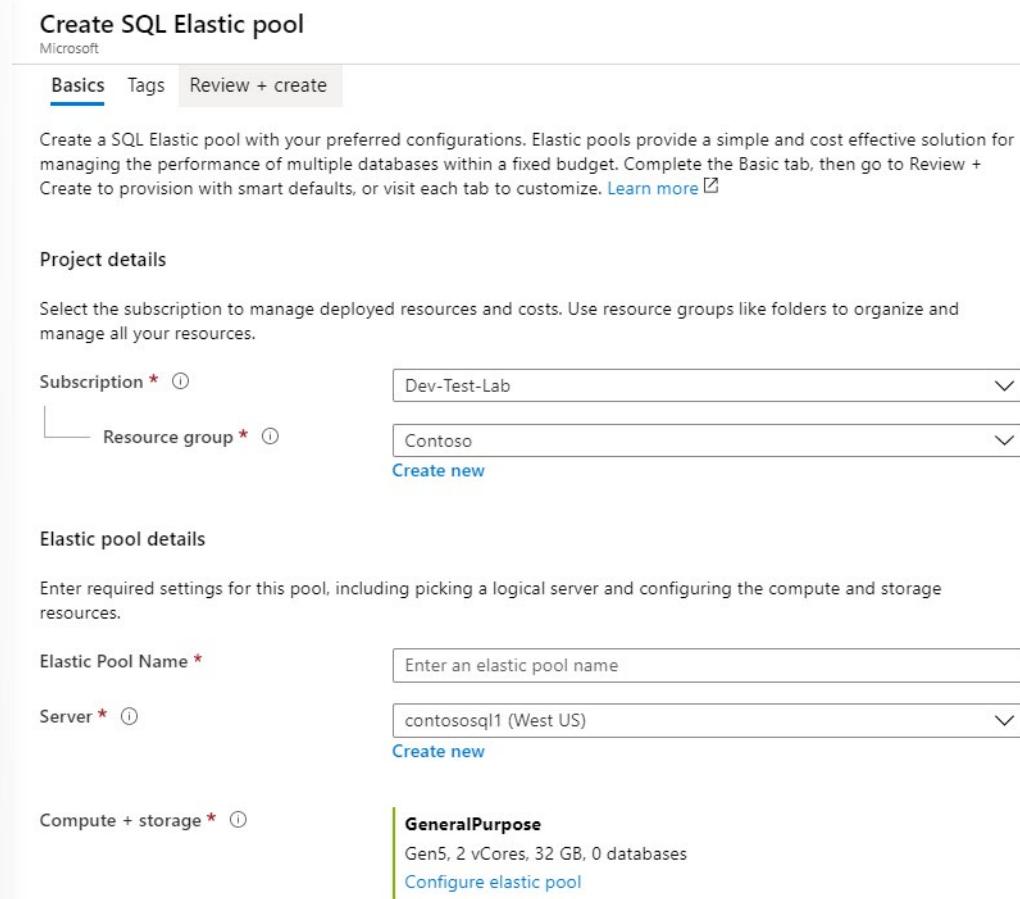


Figure 21 Elastic Pool Creation in Azure Portal

## Adding a database to an existing pool

Using the Azure portal, locate the pool to which you are adding a database, as showing in Figure 22.

Configure

Feedback

Looking for basic, standard, premium?

Pool settings Databases Per database settings

+ Add databases Revert selected

Search to filter databases...

Database name	Pricing tier	Data space used
Currently, there are no databases selected to be added to the pool. To add databases, click 'Add databases' above.		

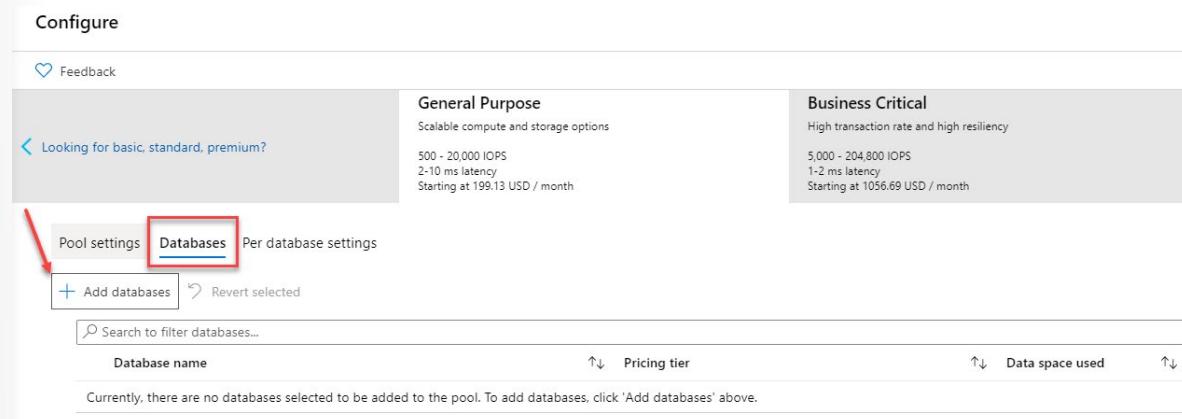


Figure 22 Adding a Database to an Elastic Pool in Azure Portal

Figure 23 shows the process for selecting which database(s) you wish to add to the pool.

The screenshot shows a modal dialog titled "Add databases". At the top right is a close button (X). Below it is a "Select all" checkbox and a "Search to filter databases..." input field. On the right, it displays "Selected/Total databases 1/4". The main area contains a table with columns: Database name, Pricing tier, and Database size. Two rows are listed:

Database name	Pricing tier	Database size
<input checked="" type="checkbox"/> AdventureWorks2014	General Purpose: Serverless, Gen5, ...	272 MB
<input type="checkbox"/> AdventureWorksDW2014	General Purpose: Serverless, Gen5, ...	111 MB

Figure 23 Elastic Pool database addition

Click Apply on the screen shown in Figure 24.

The screenshot shows the "Configure" screen for an elastic pool. At the top left is a "Configure" button. Below it is a "Feedback" section with a "Looking for basic, standard, premium?" link. The main area has tabs: "Pool settings" (disabled), "Databases" (selected), and "Per database settings". Under "Databases", there are buttons for "+ Add databases" and "Revert selected", and a "Search to filter databases..." input field. A table lists databases with columns: Database name, Pricing tier, and Data space used. One row, "AdventureWorks2014", is highlighted with a red box. At the bottom is a large blue "Apply" button, also highlighted with a red box.

Figure 24 The Final Screen to add a database to an elastic pool

Click Apply one more time and the database will be added to the elastic pool.

## Managing pool resources

The Azure portal delivers a wealth of information regarding the state and health of the elastic pool. You can view resource utilization and see which database is consuming the most resources. This can be helpful for diagnosing performance issues or identify a database that might not be a good fit for the pool, such as when one database is consuming the vast majority of pool resources. Figure 25 shows an elastic pool with even resource utilization.

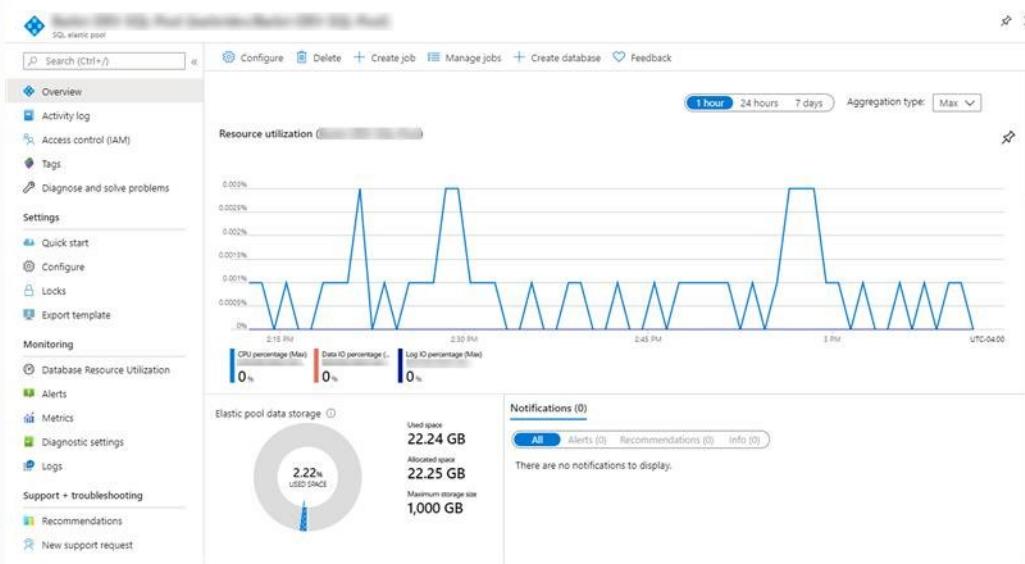


Figure 25 The Resource Utilization of an Elastic Pool

If you need to adjust the pool to decrease or increase resources allocated to the pool, you can make that change via the Configure option in the Settings section of the Elastic Pool management blade.

From that blade, you can quickly and easily adjust:

- Pool size including DTUs, vCores, and storage size.
- Service Tier
- Resources per database
- Which databases are included in the pool, by adding or removing them.

As shown in Figure 26, you can adjust numerous settings in the Elastic Pool. Many of these changes can be made online, including the min and max DTUs or vCores per database. You can change the size of total size of the pool or add and remove databases from the pool as needed, but you should be aware that active connections will be dropped as the resizing completes.

The screenshot shows the Azure Portal interface for configuring an elastic pool. On the left, a sidebar lists various management options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Quick start, Configure (which is selected), Locks, and Export template. The main content area is titled '/ElasticPoolDemo) | Configure'. It shows the 'Basic' section with two plans: 'Standard' (for less demanding workloads, starting at 75.02 USD/month) and 'Premium' (for IO-intensive workloads, starting at 112.38 USD/month). A link 'vCore-based purchasing options' leads to customization. Below this, the 'Pool settings' tab is active, showing an eDTU slider set to 50 DTU, a data max size of 4.88 GB, and a cost summary indicating an estimated monthly cost of 75.02 USD.

Figure 26 The Azure Portal page showing Elastic Pool Configuration

Probably the most useful feature is the ability to monitor Database Resource Utilization, as shown in Figure 27. This features allows you to easily see how databases are performing within the pool.

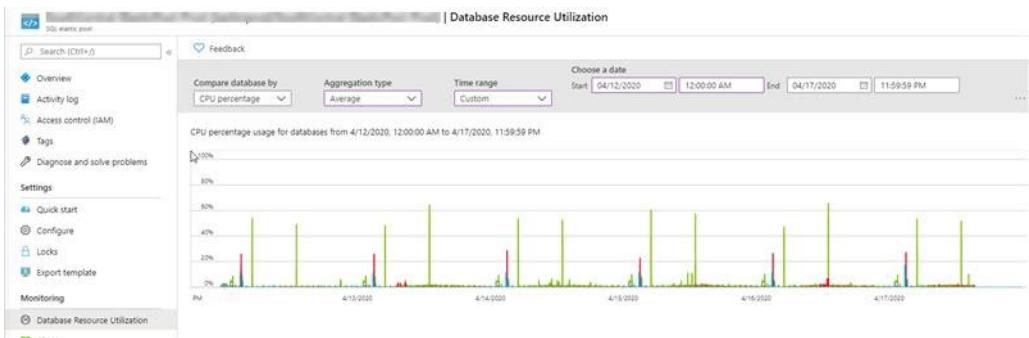


Figure 27 Utilization per database in an Elastic Pool from the Azure Portal

An elastic pool is a good fit for multi-tenant databases where each tenant has its own copy of the database. The workload should also be balanced across databases so as not to allow one database to monopolize all the pool's resources.

## Azure SQL Managed Instance

Managed instance (MI) is another Platform as a Service database offering from Microsoft. While Azure SQL Database is an offering that is typically designed around applications that were built-in the cloud, Managed Instance is designed for easy migration of on-premises SQL Server databases into a managed PaaS offering. , Azure SQL MI has nearly identical functionality to SQL Server, including SQL Agent and

access to tempdb, unlike Azure SQL Database which has little access to functionality outside of a single database. However, the operating system and hardware resources remain abstracted away from the user.

Azure SQL Managed Instance is a fully functional SQL Server instance that is almost 100% compatible with your on-premises ecosystem including features like cross-database query and common language runtime (CLR). The service uses the same infrastructure as Azure SQL Database and includes all the benefits of the PaaS service such as automatic backups, automatic patching, and built-in high availability, just to name a few.

## *Options*

There are two service tiers available when creating an Azure SQL Managed Instance, and they are the same as Azure SQL Database vCore model (managed instance is purchased using the vCore model), Business Critical and General Purpose. There are minimal functionality differences between the two tiers—the main two are that Business Critical includes In-Memory OLTP and offers a readable secondary, neither of which is available with the General Purpose tier. Both tiers offer the same levels of availability and allow for independent configuration of storage and compute.

## *High Availability*

Because Azure SQL Database Managed Instance is backed by the PaaS Service, it has high availability baked into the product. A standalone Managed Instance offers a 99.99% Service Level Agreement (SLA) which guarantees at most 52.60 minutes of downtime per year. The architecture is the same as Azure SQL Database with General Purpose which use storage replication for availability, and Business critical using multiple replicas.

## *Backups*

Automatic backups are also automatically configured for Managed Instance. One key difference between Managed Instance and Azure SQL Database is that with MI you can manually make a copy-only backup of a database. You must backup to a URL, as access to the local storage is not permissible. You can also configure long-term retention (LTR) for retaining automatic backups for up to 10 years in geo-redundant Azure blob storage.

Database backups occur on the same schedule as with Azure SQL Database. These schedules are not adjustable.

- Full – Once a week
- Differential – Every 12 hours
- Transaction Log – Every 5-10 minutes depending on transaction log usage

Restoring a database to a Managed Instance is also similar to the process with Azure SQL Database. You can use:

- Azure Portal
- PowerShell
- Azure CLI

However, there are some limitations when restoring. In order to restore from one instance to another, both instances must reside within the same Azure subscription as well as the same Azure region. You also cannot restore the entire managed instance, only individual databases within the Managed Instance itself.

As with Azure SQL Database, you cannot restore over an existing database. The existing database must either be dropped or renamed prior to restoring it from backup. Since Managed Instance is a fully functional SQL Server instance, you can execute a RESTORE command whereas with Azure SQL Database that is not possible. However, since it is a PaaS service, there are limitations.

- You must restore from an URL endpoint. You do not have access to local drives.
- You can use the following options (in addition to specifying the database):
  - FILELISTONLY
  - HEADERONLY
  - LABELONLY
  - VERIFYONLY
- Backup files containing multiple log files cannot be restored
- Backup files containing multiple backup sets cannot be restored
- Backups containing In-Memory/FILESTREAM cannot be restored

By default, the databases in a managed instance are encrypted using Transparent Data Encryption (TDE) with a Microsoft managed key. In order to take a user-initiated copy only backup, you must turn off TDE for the specific database. You will learn more about TDE in Module 3. If a database is encrypted, you can restore it, however, you will need to ensure that you have access to either the certificate or asymmetric key that was used to encrypt the database. If you do not have either of those two items, you will not be able to restore the database to a Managed Instance.

## *Disaster Recovery*

Azure SQL Database Managed Instance offers auto-failover groups as a means to implement disaster recovery. You should note that this feature protects the entire managed instance and all of the databases contained within it, not just specific databases. This process asynchronously replicates data from the Azure SQL Managed Instance to a secondary; however, it is currently limited to the paired Azure region of the primary copy, and only one replica is allowed.

Much like Azure SQL Database, auto-failover groups offer read-write and read-only listener endpoints which facilitate easy connection string management. In the event of a failover, application connection strings will automatically be routed to the appropriate instance. While fairly consistent with Azure SQL Database, these endpoints follow a slightly different format, <fog-name>.zone\_id.database.windows.net whereas Azure SQL Database is in the <fog-name>.secondary.database.windows.net format.

Each managed instance, primary and secondary, must be within the same DNS zone. This will ensure that the same multi-domain certificate can be used for client connection authentication between either of the two instances in the same failover group. This can be facilitated by specifying a "DNS Zone Partner" through various methods such as the Azure portal, PowerShell, or Azure CLI.

## **Summary and Knowledge Check**

### **Summary and Knowledge Check**

In this lesson you learned about the platform as a service options for SQL Server on Azure. Azure SQL Database is a flexible platform, well suited for software as service applications and has a number of platform options for large databases, multiple databases, or development environments that do not need

to be online 24x7. Azure SQL Database Managed Instance is a PaaS version of SQL Server that allows you to easily move your on-premises environments into a managed service. Managed Instance also supports many server-level capabilities that are not available with Azure SQL Database.

## Question 1

*You need to migrate a set of databases that use distributed transactions from on-premises SQL Server. Which of the following options should you choose?*

- Azure SQL Managed Instance
- Azure SQL Database
- Azure SQL Database Hyperscale

## Question 2

*You are building a new cloud database that you expect to grow to 50 TB. Which is the best option for your database?*

- Azure SQL Database Hyperscale
- Azure SQL Database Managed instance
- Azure SQL Database Serverless

## Question 3

*You are building a database for testing purposes that will be used less than 8 hours a day and is expected to be 20 GB in size. What is your most cost-effective option?*

- Azure SQL Database Serverless
- Azure SQL Database Elastic Pools
- Azure SQL Database Managed instance

# Deploying MariaDB, MySQL, and PostgreSQL on Azure

## Introduction

When development speed is of the essence, open source database platforms usually are unmatched when it comes delivering a fully managed ready to roll database for applications. Many developers prefer open source systems for this very reason. Microsoft has rounded out their relational database offerings by providing open source database platforms such as MySQL, MariaDB, and PostgreSQL. These platforms are a great compliment to the existing Azure SQL Server.

## Learning Objectives

After completing this lesson, you will be able to:

- Understand the service tiers available for open source database platforms
- Deploy MySQL, MariaDB, and PostgreSQL
- Configure high availability option for MySQL and MariaDB
- Configure Scale Our reads for MySQL and MariaDB
- Describe the SSL Options for MySQL and MariaDB

## Overview of Open Source Offerings

The Azure ecosystem offers three different open source database platforms. Each platform has different attributes that facilitate moving to the cloud quickly and painlessly. Each of these services comes with native high availability, automatic patching, automatic backups and the highest level of security protection. These offerings are fully supported by Microsoft from the service all the way through the database engine.

Azure Database for MySQL is a full managed enterprise grade database service that is ready to easily lift and shift customer environments to the Azure cloud. Customers can use their existing frameworks and languages to ensure that a migration doesn't disrupt any business activity. In addition, the service has built-in high availability and dynamic scaling which helps to meet any fluctuation in performance demands.

Azure Database for MariaDB is similar to the MySQL offering. It also allows for continued use of frameworks and languages of your choice. High availability and dynamic scaling are also provided by the service to ensure that customer demands are met within a moment's notice.

Azure Database for PostgreSQL helps customers to build large scalable applications. The service allows for horizontal scaling and is available with Hyperscale which allows for unparalleled performance. It also integrates with several native features such as geospatial support, rich indexing and JSONB along with other extensions. Azure Database for PostgreSQL Hyperscale is ideal for multi-tenant applications, with minor code changes to allow for data sharding.

## Service Tiers

There are three service tiers for each offering. Each tier has an ideal workload for which it is designed and allows you to choose from a variety of performance options.

- Basic – This tier is best for light workloads that need minimal compute and I/O performance.

- General Purpose – This tier is great for most workloads requiring scalable I/O throughput along with a healthy balance of compute and memory.
- Memory Optimized – This tier is suitable for workloads that demand high performance and require in-memory speed for quick processing of transactions along with higher concurrency.

The service tiers offered by the open source platforms provide a wide range of performance options and allow you to choose the best one for your given workload. The following versions shown in Table 4 are available in the service for each database.

Database	Supported Versions
MariaDB	10.2-10.3
MySQL	5.6-8.0
Postgres	9.5-11

Table 4 Support Versions of MariaDB, MySQL, and PostgreSQL in Azure Database

While the Azure service will perform minor upgrades, in order to execute a major upgrade you will be need to restore from a backup into a the new version.

## Deploying PostgreSQL

Just like the Azure SQL offerings, deployment for MariaDB, MySQL and PostgreSQL is supported using all of the standard Azure methods including the Azure Portal, PowerShell, ARM templates, and the Azure CLI. There is limited support for PowerShell cmdlets for these platforms.

### Single Database

In the Azure portal, click on Create a Resource on the main portal blade as shown. Search for Azure Database for PostgreSQL and click create. The portal will launch the screen shown in Figure 28.

Select Single Server or Hyperscale. You will learn more about Hyperscale later in the lesson, but it is typically used for large-scale databases that scale out across multiple nodes. Click the appropriate Create button.

The screenshot shows the 'Select Azure Database for PostgreSQL deployment option' page. It features two main sections: 'Single server' and 'Hyperscale (Citus) server group'. Each section includes a brief description, a 'Create' button, and a 'Learn more' link.

Deployment Option	Description	Create Button	Learn more Link
Single server	Best for broad range of traditional transactional workloads. Enterprise ready, fully managed community PostgreSQL server with up to 64 vCores, optional geospatial support, full-text search and more.	Create	Learn more
Hyperscale (Citus) server group	Ideal for multi-tenant applications and real-time analytical workloads that need sub-second response. Supports both transactional/operational workloads and hybrid transactional analytics workloads.	Create	Learn more

Figure 28 Azure Database for PostgreSQL Creation Page

You will supply the Resource Group, Server Name, Region, Version, Username and Password as shown in Figure 29.

**Single server**  
Microsoft

**Basics** Tags Review + create

Create an Azure Database for PostgreSQL server. [Learn more](#)

**Project details**

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

Subscription \*

Resource group \*  [Create new](#)

**Server details**

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name \*

Data source \*

Location \*

Version \*

Compute + storage [Configure server](#)

**Administrator account**

Admin username \*

Password \*

Confirm password \*

[Review + create](#) [Next : Tags >](#)

Figure 29 Configuration Page for Azure Database for PostgreSQL

Click on Configure Server. Select the appropriate service tier for your applications and/or workloads as shown in Figure 30. Click ok.

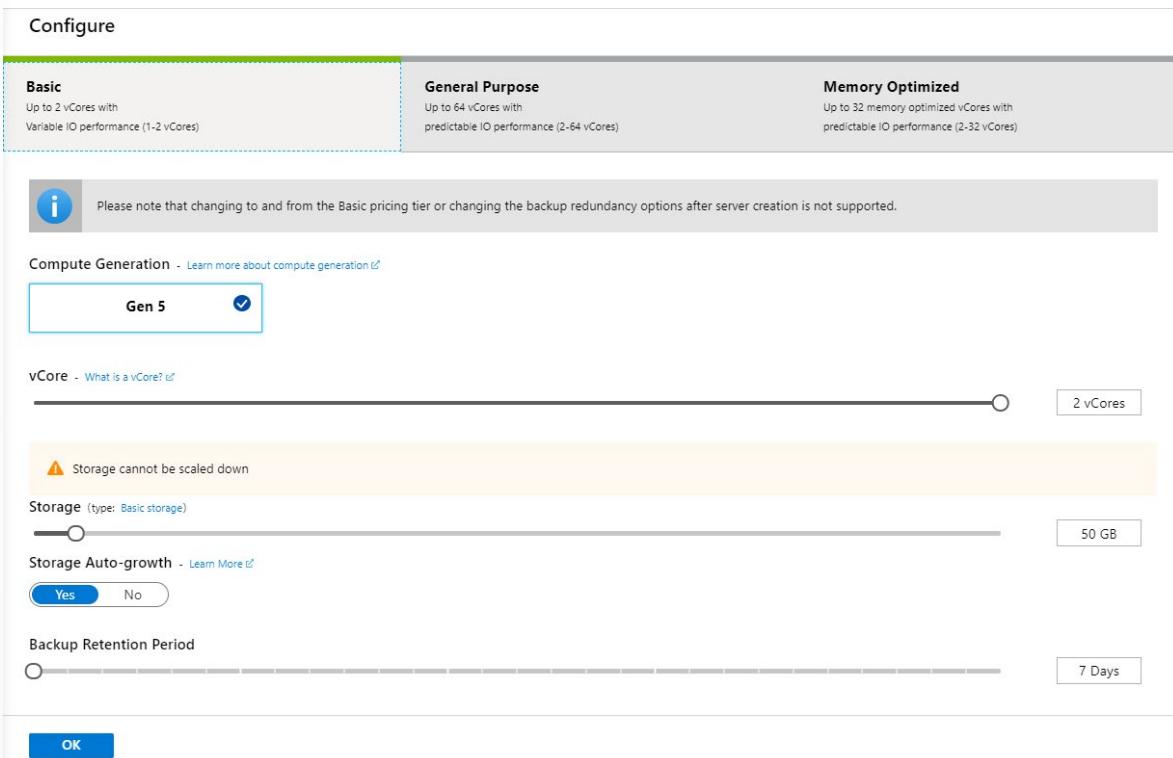


Figure 30 Creation and Tier Selection for Azure Database for PostgreSQL

Figure 30 also shows you that you can configure the amount of storage and properties of the storage. As mentioned above you can use the Azure CLI to deploy Azure Database for PostgreSQL as shown here.

```
az postgres server create --resource-group myresourcegroup --name mydemoserver `

--location westus --admin-user myadmin --admin-password <server_admin_password> `

--sku-name GP_Gen4_2 --version 9.6
```

## Azure Database for PostgreSQL Hyperscale

While Hyperscale shares a name with Azure SQL Database and while they both offer horizontal scalability supporting very large data volumes, the Hyperscale technology for PostgreSQL is implemented differently. Hyperscale allows the servers for Azure Database for PostgreSQL (called nodes) to work together in a "shared nothing" architecture design. Nodes are added to a server group, and each server group has a coordinator node, and multiple workers nodes. Applications send their queries to the coordinator node which relays the query to relevant worker nodes and gathers the results.

Hyperscale databases are sharded, which means some the data in a single table can be split across multiple nodes, using a type of table called distributed tables. This allows for both parallelization and distribution of queries across nodes. The worker nodes and coordinator nodes can be scaled independently of each other.

Creating a Hyperscale PostgreSQL deployment is different than deploying a single instance of the service. In this case, Hyperscale allows you to deploy additional worker nodes along with a coordinator node. You can deploy up to 20 worker nodes by default (this is a soft limit; for additional nodes, you can contact Microsoft support). You can also configure high availability for each node to ensure that any disruptions have minimal impact on your applications. You will learn more about this high availability configuration in Module 7.

## Deploying MySQL/MariaDB

When deploying MySQL or MariaDB you have similar options to PostgreSQL; you can deploy using the Azure Portal, Azure CLI, or ARM templates. There are similar sizing and availability choices as detailed below.

### *HA Configuration*

High availability for both Azure Database for MySQL and Azure Database for MariaDB comes packaged in with the service so there is less administrative work that needs to be done. The service provides a guaranteed high level of availability, offering an uptime of 99.99%. This equates to a maximum of 52.60 minutes of downtime per year.

Transactions on either platform are written synchronously to storage. If a node interruption occurs, the database server will automatically create a new node and subsequently attach the storage to the new node. Any transactions in flight are not committed and active connections to the database are dropped. As mentioned with Azure SQL Database, it is important to ensure that applications connecting to the database service support retry logic, also known as connection resiliency, in their database connections.

### *Scale Out Reads Using Read Replicas*

The Azure Service makes it easy to scale our read workloads using read replicas for MySQL and MariaDB. The example below illustrates the procedure for MySQL however MariaDB uses a similar process.

To enable replication click on Replication in the Settings section of the MySQL blade in the Azure portal as shown in Figure 31.

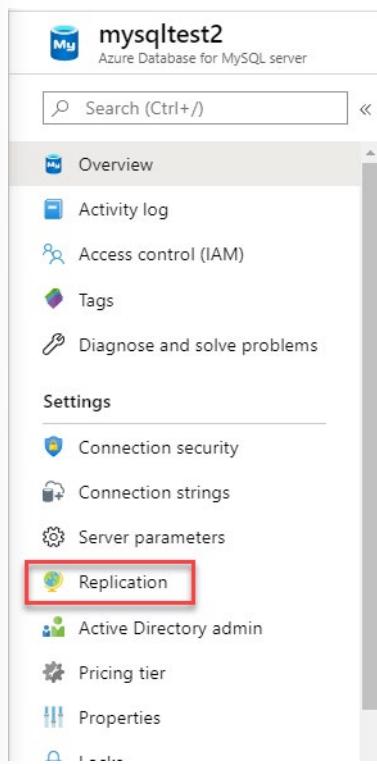


Figure 31 MySQL blade in the Azure Portal highlighting Replication

Click Add Replica as shown in Figure 32.

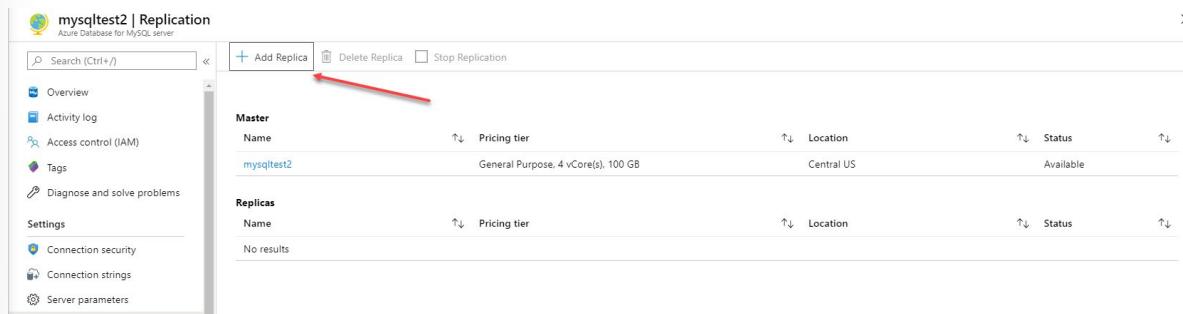


Figure 32 Add Replica in MySQL

Provide a new server name and specify which Azure region it should reside in, as shown in Figure 33.

The screenshot shows the 'MySQL server' configuration page in the Azure portal. It includes fields for 'Restore to new server \*' (mysqltest2west), 'Location \*' ((US) West US), and 'Pricing tier' (General Purpose, 4 vCores, 100 GB storage, estimated cost per month \$18.54 USD). A 'Supported Locations' link is also present.

Figure 33 Configuring a replica in MySQL

Click Ok.

## Migration to Azure Database for MariaDB, MySQL, and PostgreSQL

Many organizations are migrating their Oracle databases to Azure PostgreSQL to reduce licensing costs. Migrating to the Azure Database platform is made easier by the Azure Database Migration Service (DMS). The DMS supports both homogenous (e.g. MySQL in a VM to Azure Database for MySQL) and heterogeneous (e.g. Oracle in a VM to Azure Database for PostgreSQL) migrations. DMS performs an initial load of your on-premises database or database running in an Azure VM to Azure Database, and then continuously syncs new database transactions to the Azure target. When you are ready to cut over to the target Azure service, you can stop the replication, and switch the connection strings in your application to the Azure Database.

## Summary and Knowledge Check

The open source offerings on the Azure Database for MariaDB, MySQL, and PostgreSQL platform cover the most popular open source offerings and offer value-added services like built-in high availability and backups in addition to easy scaling up and down for the flexibility your application needs.

### Question 1

*You are deploying a mission critical MySQL database to support an e-commerce site which depends on low latency. What should you configure your application to do in order to handle transient errors?*

- Connection resiliency
- Secure Socket Layer (SSL)
- An Azure Virtual Network

## Question 2

*Which of the following is a benefit of Azure Database for PostgreSQL?*

- Managed automated backups
- Automatic Query Tuning
- Automated cross-region disaster recovery

## Question 3

*What's the process to upgrade a major version of Azure Database for MySQL?*

- The Service performs the update for you
- Create a dump and restore to a server at the higher version
- Change the version of the server in the portal

## Module Summary

### Module Summary

The Azure Data Platform offerings are very extensive, ranging from running SQL Server databases on VMs, to a fully managed service like Azure SQL Database and even including the most popular open-source database platforms like MySQL and PostgreSQL. IaaS offerings may offer more flexibility in configuration but they do require proper configuration of storage and networking. PaaS offerings are easier to get started with and manage, but may face application incapability. Azure SQL Database offerings can support the full range of small to very large databases with Hyperscale offering support for volumes up to 100 TB. Managed Instances offers nearly feature complete compatibility with SQL Server, but in a fully managed PaaS service. Azure Database for MariaDB, MySQL, and PostgreSQL build on the Azure database platform and add enhanced features for PostgreSQL like Hyperscale.

### Additional Resources

Azure VM Sizes

<https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes?toc=/azure/virtual-machines/windows/toc.json>

Azure QuickStart GitHub Repository

<https://github.com/Azure/azure-quickstart-templates>

# Answers

## Question 1

What type of storage offers the lowest latency in Azure?

- Ultra Disk
- Premium Disk
- Standard SSD

*Explanation*

*Ultra Disk offers the lowest latency of the three storage types*

## Question 2

Which tool would you use to assess and migrate your databases from an on-premises SQL Server to an Azure VM?

- Microsoft Assessment and Planning Toolkit
- Database Experimentation Assistant
- Data Migration Assistant

*Explanation*

*The data migration assistant provides both tooling to assess and execute a migration*

## Question 3

To reduce the cost of an Azure SQL Server VM you intend to run full time for 3 years, which option should you choose?

- Azure Reserved VM Instances
- Availability Set
- Pay as You Go Licensing

*Explanation*

*Azure Reserved VM Instance can offer significant cost saving over pay as you go licensing.*

## Question 1

You need to migrate a set of databases that use distributed transactions from on-premises SQL Server. Which of the following options should you choose?

- Azure SQL Managed Instance
- Azure SQL Database
- Azure SQL Database Hyperscale

*Explanation*

*Azure SQL Managed instance is the only offering here that supports cross-database transactions*

**Question 2**

You are building a new cloud database that you expect to grow to 50 TB. Which is the best option for your database?

- Azure SQL Database Hyperscale
- Azure SQL Database Managed instance
- Azure SQL Database Serverless

*Explanation*

*Azure SQL Database Hyperscale is the only option that supports 50 TB of data*

**Question 3**

You are building a database for testing purposes that will be used less than 8 hours a day and is expected to be 20 GB in size. What is your most cost-effective option?

- Azure SQL Database Serverless
- Azure SQL Database Elastic Pools
- Azure SQL Database Managed instance

*Explanation*

*Serverless will reduce the cost significantly for a database that is only used during business hours.*

**Question 1**

You are deploying a mission critical MySQL database to support an e-commerce site which depends on low latency. What should you configure your application to do in order to handle transient errors?

- Connection resiliency
- Secure Socket Layer (SSL)
- An Azure Virtual Network

*Explanation*

*Connection resiliency should be built into your application to handle transient failures, like failover or database resizing*

**Question 2**

Which of the following is a benefit of Azure Database for PostgreSQL?

- Managed automated backups
- Automatic Query Tuning
- Automated cross-region disaster recovery

*Explanation*

*Automated backup is a feature of the service.*

**Question 3**

What's the process to upgrade a major version of Azure Database for MySQL?

- The Service performs the update for you
- Create a dump and restore to a server at the higher version
- Change the version of the server in the portal

*Explanation*

*It is required to create a backup and restore to upgrade to a new major version.*

## Module 3 Implement a Secure Environment

### Module Introduction

#### Module Introduction

This module explores the practices of securing your SQL Server Database as well as an Azure SQL database. This includes a review of the various SQL Server-based options as well as the various Azure options for securing Azure SQL Database as well as the databases with reside within Azure SQL Database. Students will lean why security is crucial when working with databases.

#### Learning Objectives

### Learning Objectives

After completing this module, you will:

- Understand the differences between Windows, SQL Server and Azure Active Directory Authentication
- Be able to describe and configure both data-at-rest encryption solutions as well as data-in-transit encryption solutions
- Be able to implement a data sensitivity solution.

# Configure Database Authentication

## Introduction

Azure SQL Database has several authentication options which are different from the authentication options of SQL Server. This is because Azure SQL Database and Azure SQL Managed Instance rely on Azure Active Directory instead of Windows Server Active Directory. In this lesson you will learn about the differences between those authentication mechanisms.

## Learning Objectives

In this lesson you will learn about:

- Authentication Options for Azure SQL Database
- Security Principals
- Roles in Azure SQL Database

## What's the Difference Between Active Directory and Azure Active Directory

A common question is what is the difference between Azure Active Directory and Windows Server Active Directory, which we'll refer to simply as 'Active Directory'. This is especially confusing for new administrators because Azure Active Directory interacts with Active Directory. Both solutions provide authentication services and identity management, but in different ways—Active Directory uses a protocol called Kerberos to provide authentication using tickets, and it is queried by the Lightweight Directory Access Protocol (LDAP). Azure Active Directory uses HTTPS protocols like SAML and OpenID Connect for authentication and uses OAuth for authorization.

The two services have different use cases—for example, you cannot join a Windows Server to an Azure Active Directory domain and work together in most organizations to provide a single set of user identities. A service called Azure Active Directory Connect connects your Active Directory identities with your Azure Active Directory.

## Authentication and identities

Both on-premises SQL Server installations as well as SQL Server installations within Azure Virtual Machines support two modes of authentication: SQL Server authentication and Windows Authentication. When using SQL Server authentication, SQL Server-specific login name and password information is stored within SQL Server, either in the master database, or in the case of contained users, within the user database. Using Windows Authentication, users connect to the SQL Server using the same Active Directory account they use to log into their computer (as well as accessing file shares and applications.)

Active Directory authentication is considered to be more secure because SQL Server authentication allows for login information to be seen in plain text while being passed across the network. In addition, Active Directory authentication makes it easier to manage user turnover. If a user leaves the company and you use Windows authentication, the administrator would only have to lock the single Windows account of that user, instead of identifying each occurrence of a SQL login.

Azure SQL Database similarly supports two different modes of authentication, SQL Server authentication and Azure Active Directory authentication. SQL Server authentication is the same authentication method that has been supported in SQL Server since it was first introduced, where user credentials are stored

within either the master database or the user database.. Authentication via Azure Active Directory allows the user to enter the same username and password which is used to access other resources such as the Azure portal or Microsoft 365.

As mentioned above, Azure Active Directory can be configured to sync with the on-premises Active Directory. This allows users to have the same usernames and passwords to access on-premises resources as well as Azure resources. Azure Active directory adds on additional security measures by allowing the administrator to easily configure multi-factor authentication (MFA).

With MFA enabled on an account, after the correct username and password is supplied, a second level of authentication is required. By default, MFA can be configured to use the Windows Authenticator application which will then send a push notification to the phone. Additional options for the default MFA action include sending the recipient a text message with an access code, or having the user enter an access code that was generated with the Microsoft Authenticator application. If a user has MFA enabled, they have to use the Universal Authentication with MFA option in Azure Data Studio and SQL Server Management Studio as shown in Figure 1.

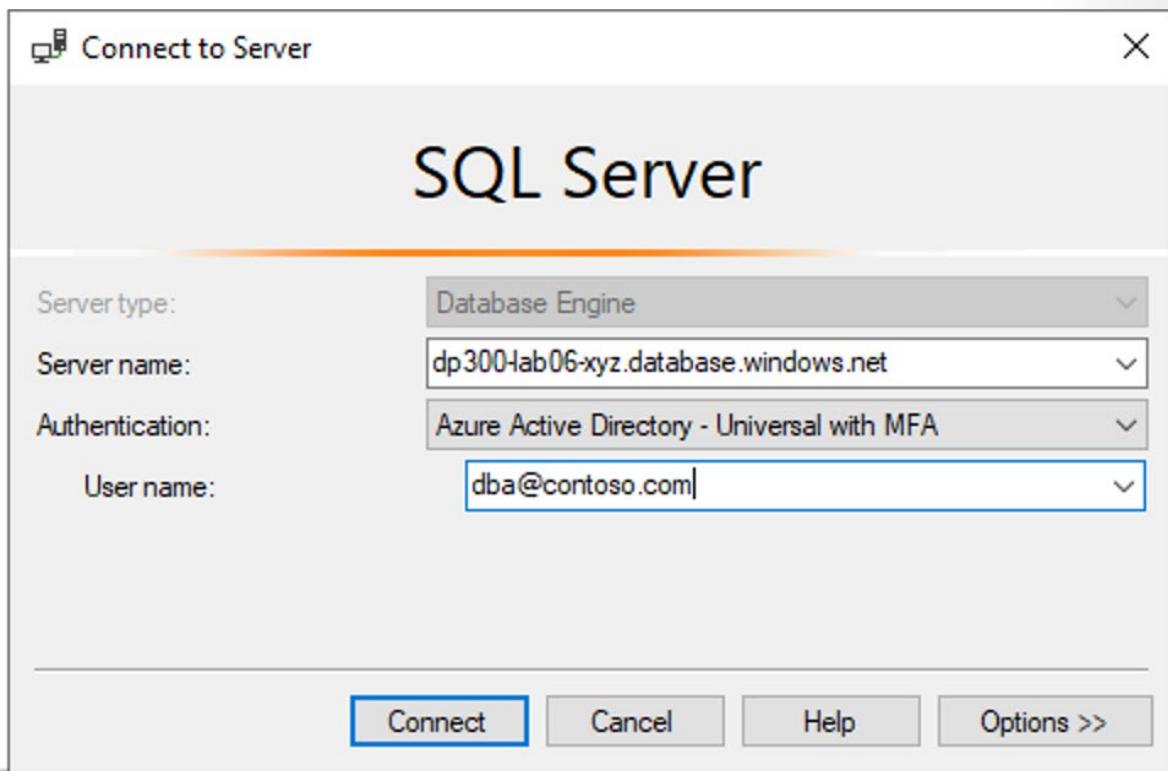


Figure 1 Universal Authentication in SQL Server Management Studio

Both Azure SQL Database for SQL Server and Azure Database for PostgreSQL support configuring the server which is hosting the database to use Azure Active Directory Authentication as shown in Figure 2..

The screenshot shows the Azure portal interface for managing an Azure SQL server. The server name is 'dp300-lab06-xyz'. On the left, there's a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and Settings. The main pane displays various settings for the server, including its resource group ('DP-300-RG'), status ('Available'), location ('East US'), subscription ('Contoso Ltd'), and tags. A specific setting, 'Active Directory admin', is highlighted with a red box and set to 'DBA Team'. Other settings shown include 'Firewalls and virtual net...' and 'Server name'.

Figure 2 Azure Active Directory Admin Configuration for Azure SQL server

This login allows the admin access to all of the databases in the server. It is a best practice to make this account an Azure Active Directory group, so access is not dependent on a single login. In Figure 2, that group has been called DBA team. The Azure Active Directory Admin account grants special permissions and allows the account or group that holds that permission to have sysadmin like access to the server and all of the databases within the server. The admin account is only set using Azure Resource Manager and not at the database level. In order to change the account or group, you have to use the Azure Portal, PowerShell, or CLI.

## Security Principles

*Security Principals* are entities that can request SQL Server resources and to which you can (usually) grant permissions. There are several sets of security principals in SQL Server. Security principals exist at either the server level or the database level and can be either individuals or collections. Some sets have a membership controlled by the SQL Server administrators, and some have a fixed membership.

Logins and Server Roles are the server-level security principals we will be discussing. New logins can be added by administrators, but new server roles cannot be added.

At the database level, we'll look at users, database roles, application roles.

## Schemas and Securables

Before we look at the details of security principals, we need to understand the concepts of securables and schemas. SQL Server and Azure SQL Database have three scopes for securables. Securables are the resources within the database to which the authorization system manages access. For example, a table is a securable. To simplify access control SQL Server contains securables in nested hierarchies called scopes. The three securable scopes are the server, the database, and the schema. A schema is a collection of objects within your database which allows objects to be grouped into separate namespaces.

Every user has a default schema. If a user tries to access an object without specifying a schema name, as in: `SELECT name FROM customers`, it is assumed the object is in the user's default schema. If there is no such object in the default schema, SQL Server will check to see if the object is in the pre-defined `dbo` schema. If there is no object of the specified name in either the user's default schema, or in the `dbo` schema, the user will receive an error message. It is considered best practice to always specify the schema name when accessing objects, so the previous select would be something like:

`|SELECT name FROM SalesSchema.customers`. If a user has not been given a default schema, the default default schema is `dbo`.

By default, if no schema is specified when a user creates an object, SQL Server will attempt to create it in the user's default schema. If the user has not been granted permission to create objects in their default schema, the object cannot be created.

## Logins and Users

No matter the mode of authentication that is used, a login name used to access your SQL database is setup as a login within the instance. Those logins are setup at the instance level of SQL Server and stored in the master database. However, you can configure contained users which are added at the database level. These users can be configured as SQL Server Authentication users as well as either Windows Authentication users or Azure Active Directory users (depending on which platform you are using). In order to create these users, the database must be configured for partial containment, which is configured by default in Azure SQL Database, and optionally configurable in SQL Server.

These users only have access to the database that the user is setup with. For the purposes of Azure SQL Database, it is considered a best practice to create users at the scope of the user database, and not in the master database as shown below.

```
CREATE USER [dba@contoso.com] FROM EXTERNAL PROVIDER;
```

```
GO
```

The create user statement is executed in the context of the user database. In the example above, the user is an Azure Active Directory user as indicated with the FROM EXTERNAL PROVIDER syntax.

If logins are created at the instance level in SQL Server, a user should then be created within the database which maps the user to the server-based login as shown in the following example.

```
USE Master
```

```
GO
```

```
CREATE LOGIN demo WITH PASSWORD = 'Pa55.w.rd'
```

```
GO
```

```
USE WideWorldImporters
```

```
GO
```

```
CREATE USER demo FROM LOGIN demo
```

```
GO
```

The login is first created in the master database, and then in the WideWorldImporters database a user is created to map to that user. Logins are used to access the SQL Server or the Azure SQL Database, but to do any work within a database, the login must be mapped to a username. The username is used for all authentication, as we'll see in Lesson 2.

Logins and usernames are the most important security principals you need to be aware of, but the next section describe some of the other concepts and terms when dealing with authorization.

## Database Roles

As you can imagine, database security can get quite complicated for applications with many users. In order to make it easier for both administrators and auditors, most database applications use role-based security. Roles are effectively security groups that share a common set of permissions. This allows a set of roles to be created for a given application. Some examples of roles would be administrators who had full access to all of the databases and servers, reporting users who only read the database, and an application account that had access to write data into the database. The roles can be defined when the application is designed, and then users can be assigned to those roles as they need access to the database. Role based access control or RBAC is a common architecture across computer systems and is how authorization is managed in Azure Resource Manager.

SQL Server and Azure SQL Database both include built-in roles that are defined by Microsoft, and also provide the option to create custom roles. Custom roles can be created at the server or database level. However, server roles cannot be granted access to objects within a database directly. Server roles are only available in SQL Server and Azure SQL Managed Instance, not in Azure SQL Database.

Within a database, permissions can be granted to the users that exist within the database. If multiple users all need the same permissions this can be accomplished by creating a database role within the database and granting the needed permissions to this role. Users can be added a member of the database role. This will allow the members of the database role to inherit the permissions of the database role and allow them whatever access has been configured for the database role.

```
CREATE USER [DP300User1] WITH PASSWORD = 'Pa55.w.rd'  
GO  
  
CREATE USER [DP300User2] WITH PASSWORD = 'Pa55.w.rd'  
GO  
  
CREATE ROLE [SalesReader]  
GO  
  
ALTER ROLE [SalesReader] ADD MEMBER [DP300User1]  
GO  
  
ALTER ROLE [SalesReader] ADD MEMBER [DP300User2]  
GO  
  
GRANT SELECT, EXECUTE ON SCHEMA::Sales TO [SalesReader]  
GO
```

In the above example, you can see that two users are created, and then a role called SalesReader is created. The two new users are added to the newly created role, and then finally the role is granted SELECT and EXECUTE permissions on the Sales schema. This means any user who is in that role can select from any object in the Sales schema, as well as execute any stored procedure in the schema.

## Application Roles

Application roles can also be created within a SQL Server database or Azure SQL Database. Unlike database roles, users are not made members of an application role. An application role is activated by the user, by supplying the pre-configured password for the application role. Once the role is activated the permissions which are applied to the application role are applied to the user until that role is deactivated.

## Built-In Database Roles

Microsoft SQL Server contains several fixed database roles within each database for which the permissions are predefined. Users can be added as members of one or more roles. These roles give their members a pre-defined set of permissions. These roles work the same within Azure SQL Database and SQL Server.

Users that need to create other users within the database can be granted membership in the role db\_accessadmin. This role does not grant access to the schema of any of the tables, nor does it grant access to the data within the database.

Users that need to backup a database in a SQL Server or Managed Instance can be made members of the role db\_backupoperator. The role db\_backupoperator does not confer any permissions in an Azure SQL Database .

Users that need the ability to read from every table and view within the database can be made members of the role db\_datareader.

Users that need the ability to INSERT, UPDATE, and DELETE data from every table and view within the database can be made members of the role db\_datawriter.

Users who need the ability to create or modify objects within the database can be made members of the role db\_ddladmin. Members of this role can change the definition of any object, of any type, but members of this role are not granted access to read or write any data within the databases.

The role db\_denydatareader can be used for users who need to be prevented from reading data from any object in the database, when those users have been granted rights through other roles or directly.

The role db\_denydatawriter can be used for users who need to be prevented from writing data to any object in the database, when those users have been granted rights through other roles or directly.

Users who need administrative access to the database can be made members of the role db\_owner. Members of the db\_owner role can perform any action within the database and they cannot be blocked by placing the members in any other database roles or by denying access to objects. Membership in this database role should be limited to only trusted users.

Users who need to be able to grant access to other users within the database can be made members of the role db\_securityadmin. Members of this role are not specifically granted access to the data within the database; however members of this role can grant themselves access to the tables within the database. Membership in this database role should be limited to only trusted users.

All users within a database are automatically members of the public role. By default, this role has no permissions granted to it. Permissions can be granted to the public role, but you should consider carefully whether that is really something you want to do. Granting permissions to the public role would grant these permissions to any user, including the guest account, if the guest account was enabled.

The built-in database roles do meet the needs of many applications; however with applications that require more granular security (for example, when you only want to grant access to a specific subset of tables) a custom role is often a better choice. You should also note that users in roles like db\_owner can always see all of the data in the database. Applications should take advantage of encryption options like

Always Encrypted to protect sensitive data from privileged users. You will learn more about encryption later in the module.

Azure SQL Database has two additional roles that are defined in the master database of Azure SQL server.

The role dbmanager within the master database allows its members to create additional databases within the Azure SQL Database environment. This role is the equivalent of the dbcreator fixed server role in an on-premises Microsoft SQL Server.

The role loginmanager within the master database allows its members to create additional logins at the server level. This role is the equivalent of the securityadmin fixed server role in an on-premises Microsoft SQL Server.

## Fixed Server Roles

In addition to database roles, SQL Server and Azure SQL Managed Instance both provide several fixed server roles. These roles assign permissions at the scope of the entire server. Server level principals, which include SQL Server logins, Windows accounts, and Windows group can be added into fixed server roles. The permissions for fixed server roles are predefined, and no new server roles can be added. The fixed server roles are:

**Sysadmin** — Members of the sysadmin role can perform any activity on the server.

**Serveradmin** — Members of the serveradmin role can change server-wide configuration settings (for example Max Server Memory) and can shutdown the server.

**Securityadmin** — Members of the securityadmin role can manage logins and their properties (for example, changing the password of a login). The members can also grant and revoke server and database level permissions. This role should be treated as being equivalent to the sysadmin role.

**Processadmin** — Members of the processadmin role can kill processes running inside of SQL Server.

**Setupadmin** — Members of the setupadmin role can add and remove linked servers using T-SQL.

**Bulkadmin** — Members of the bulkadmin role can run the BULK INSERT T-SQL statement.

**Diskadmin** — Members of the diskadmin role have the ability to manage backup devices in SQL Server.

**Dbcreator** — Members of the dbcreator role have the ability to create, restore, alter, and drop any database.

**Public** — Every SQL Server login belongs to the public user role. Unlike the other fixed server roles, permissions can be granted, denied, or revoked from the public role.

## Summary and Knowledge Check

### Question 1

*Which protocol is used by Azure Active Directory for Authorization?*

- Kerberos
- LDAP
- OAuth

## Question 2

*Which database stores the information about logins in SQL Server?*

- master
- model
- msdb

## Question 3

*Which role allows users to create users within a database*

- db\_datareader
- db\_accessadmin
- db\_securityadmin

# Configure Database Authorization

## Introduction

This module explores the practices of granting permissions and what the various permissions do within a database. This module also explores the concept of “least privilege”. While the built-in roles in SQL Server and other database engines provide broad brushes of security privileges, many applications need more granular security on database objects.

## Learning Objectives

After completing this lesson, you will:

1. Understand permissions within an Azure SQL Database
2. Understand permission tree within SQL Server
3. Understand the concept of least privilege

## Database and Object Permissions

All Relational Database Management platforms have 4 basic permissions which control data manipulation language (DML) operations. These permissions are SELECT, INSERT, UPDATE, and DELETE. These permissions apply to all SQL Server platforms as well as Azure SQL Database for MySQL and Azure SQL Database for PostgreSQL. All of these permissions can be granted, revoked or denied on tables and views. If a permissioned is granted using the GRANT statement, then the permission is given to the user or role referenced in the GRANT statement. Users can also be denied permissions using the DENY command. If a user is granted a permission and denied the same permission, the DENY will always supersede the grant, and the user will be denied access to the specific object as shown in Figure 3.

The screenshot shows a SQL Server Management Studio (SSMS) interface. In the top pane, there is a code editor containing the following T-SQL script:

```
GRANT SELECT ON dbo.Company TO Demo  
GO  
DENY SELECT ON dbo.Company TO Demo  
GO  
EXECUTE AS USER = 'Demo'  
SELECT Name, Address FROM dbo.Company
```

In the bottom pane, under the 'Messages' tab, the output is displayed in red text:

```
Msg 229, Level 14, State 5, Line 17  
The SELECT permission was denied on the object 'Company', database 'WideWorldImporters', schema 'dbo'.
```

Below the messages, the completion time is shown:

```
Completion time: 2020-05-13T14:42:28.8361616-07:00
```

Figure 3 A T-SQL Example of a DENY overriding a GRANT

In the above example, the user Demo is granted SELECT and then denied SELECT permissions on the dbo. Company table. When the user tries to execute a query that selects from the dbo.Company table, the user receives an error that SELECT permissioned was denied.

## What Is Execute As User

The EXECUTE AS [user], or EXECUTE AS [login] (only available in SQL Server and Azure SQL Managed Instance) commands allow for the user context to be changed. As subsequent commands and statements will be executed using the new context with the permissions granted to that context.

If a user has a permission and the user no longer needs to have that permission, permissions can be removed (either grants or denies) using the REVOKE command. The revoke command will remove any GRANT or DENY permissions for the right specified to the user specified.

## Table and View Permissions

Tables and views represent the objects on which permissions can be granted within a database. Within those tables and views, you can additionally restrict the columns that are accessible to a given security principal (user or login). SQL Server and Azure SQL Database also include row-level security which can be used to further restrict access.

The SELECT permission, when granted, will allow the user to view the data within the object (table or view). When denied, the user will be prevented from viewing the data within the object.

The INSERT permission, when granted, will allow the user to insert data into the object. When denied, the user will be prevented from inserting data into the object.

The UPDATE permission, when granted, will allow the user the update data within the object. When denied, the user will be prevented from updating data in the object.

The DELETE permission, when granted, will allow the user to delete data within the object. When denied, the user will be prevented from deleting data from the object.

Azure SQL Database and Microsoft SQL Server have additional permissions which can be granted, revoked or denied as needed.

The CONTROL permission grants all rights to the objects. This allows the user who has this permission to perform any action they wish against the object, including deleting the object.

The REFERENCES permission grants the user the ability to view the foreign keys on the object. This permission is also available in Azure SQL Database for MySQL and Azure SQL Database for PostgreSQL.

The TAKE OWNERSHIP permission allows the user the ability to take ownership of the object.

The VIEW CHANGE TRACKING permission allows the user to view the change tracking setting for the object.

The VIEW DEFINITION permission allows the user to view the definition of the object.

## Function and Stored Procedure permissions

Like tables and views, functions and stored procedures have several permissions which can be granted or denied.

The ALTER permission grants the user the ability to change the definition of the object.

The CONTROL permission grants the user all rights to the object.

The EXECUTE permission grants the user the ability to execute the object. This permission can be granted to Azure SQL Database for MySQL and Azure SQL Database for PostgreSQL.

The VIEW CHANGE TRACKING permission allows the user to view the change tracking setting for the object.

The VIEW DEFINITION permission allows the user to view the definition of the object.

## Ownership Chains

A concept called chaining applies to permissions which allows users to inherit permissions from other objects. The most common example of this is a function or stored procedure that accesses a table during its execution. If the procedure has the same owner as the table, the stored procedure is able to be executed and access the table, even though the user does not have rights to access the table directly. This is because the user inherits the rights to access the table from the stored procedure, but only for the duration of the execution of the stored procedure, and only within the context of the stored procedures execution.

In the example below, run as a database owner or server administrator, a new user is created and added as a member of a new *SalesReader* role, which is then granted permission to select from any object and execute any procedure in the Sales schema. A stored procedure is then created in the Sales schema which accesses a table in the Production schema. .

The example then changes content to be the new user and an attempt is made to select directly from the table in the Production schema.

```
USE AdventureWorks2016;
GO
CREATE USER [DP300User1] WITH PASSWORD = 'Pa55.w.rd';
GO
CREATE ROLE [SalesReader];
GO
ALTER ROLE [SalesReader] ADD MEMBER [DP300User1];
GO
GRANT SELECT, EXECUTE ON SCHEMA::Sales TO [SalesReader];
GO
CREATE OR ALTER PROCEDURE Sales.DemoProc
as
SELECT P.Name, Sum(SOD.LineTotal) as TotalSales ,SOH.OrderDate
FROM Production.Product P
INNER JOIN Sales.SalesOrderDetail SOD on SOD.ProductID = P.ProductID
INNER JOIN Sales.SalesOrderHeader SOH on SOH.SalesOrderID = SOD.SalesOrde-
```

```
    rID  
  
    GROUP BY P.Name, SOH.OrderDate  
  
    ORDER BY TotalSales DESC;  
  
    GO  
  
  
    EXECUTE AS USER = 'DP300User1';  
  
  
  
    SELECT P.Name, Sum(SOD.LineTotal) as TotalSales ,SOH.OrderDate  
  
    FROM Production.Product P  
  
    INNER JOIN Sales.SalesOrderDetail SOD on SOD.ProductID = P.ProductID  
  
    INNER JOIN Sales.SalesOrderHeader SOH on SOH.SalesOrderID = SOD.SalesOrde-  
rID  
  
    GROUP BY P.Name, SOH.OrderDate  
  
    ORDER BY TotalSales DESC;
```

The above query results in an error that the user *DP300User1* does not have SELECT permission, because the role that the user belongs to does not have any privileges in the Production schema. Now we can try to execute the stored procedure:

```
EXECUTE AS USER = 'DP300User1';  
  
EXECUTE Sales.DemoProc;
```

The *DP300User1* has EXECUTE permission on the stored procedure in the *Sales schema*, because the role the user belongs to has EXECUTE permission on the *Sales schema*. Because the table has the same owner as the procedure, we have an unbroken ownership chain, and the execution will succeed and results will be returned.

Permission changes do not apply when dynamic SQL is being used within stored procedures. The reason that dynamic SQL breaks the permission chain is due to the fact that the dynamic SQL is executed outside of the context of the calling stored procedure. You can see this behavior by changing the above stored procedure to execute using dynamic SQL as shown below.

```
CREATE OR ALTER PROCEDURE Sales.DemoProc  
  
AS  
  
DECLARE @sqlstring NVARCHAR(MAX)
```

```
SET @sqlstring = 'SELECT P.Name, Sum(SOD.LineTotal) as TotalSales ,SOH.
OrderDate
FROM Production.Product P
INNER JOIN Sales.SalesOrderDetail SOD on SOD.ProductID = P.ProductID
INNER JOIN Sales.SalesOrderHeader SOH on SOH.SalesOrderID = SOD.SalesOrde-
rID
GROUP BY P.Name, SOH.OrderDate'
EXECUTE sp_executesql @sqlstring
GO
--
EXECUTE AS USER = 'DP300User1'
EXECUTE Sales.DemoProc
```

The *DP300User1* will receive an error that the user does not have SELECT permission on the *Production.Product* table, just like the user tried to execute the query directly. Because of this permission chains to not apply and the user account which is executing the dynamic SQL must have rights to the tables and views which are being used by the code within the dynamic SQL.

## What is Dynamic SQL

Dynamic SQL is a concept where a query is built programmatically. This allows T-SQL statements to be generated within a stored procedure or a query itself. A very simple example is shown below.

```
SELECT 'BACKUP DATABASE ' + name + ' TO DISK =''\\backup\sql1\' + name +
'.bak'''
FROM sys.databases
```

The above statement will generate a list of T-SQL statements to backup all of the database on the server. Typically, this generated T-SQL will be executed using *sp\_executesql* or passed to another program to execute.

## Least Privilege

The principle of least privilege is fairly simple. The basic idea behind the concept is that users and applications should only be given the permissions needed in order for them to complete the task. This means that applications should only have permissions that they need to do in order to complete the task at hand.

As an example, if an application accesses all data through stored procedures, then the application should only have the permission to execute the stored procedures, with no access to the tables.

## Summary and Knowledge Check

### Question 1

*Which permission allows the user to perform any option against a database object?*

- Control
- Delete
- View Definition

### Question 2

*Which database object can be granted insert access?*

- Functions
- Tables
- Procedures

### Question 3

*What feature allows a user to execute a stored procedure even if she does not have permission to access the tables referenced in the stored procedure?*

- Ownership chaining
- Principle of least privilege
- Granular security

# Implement Security for Data at Rest

## Introduction

This lesson explores the encryption options available within Microsoft SQL Server, Azure SQL Database, Azure SQL Database for MySQL and Azure SQL Database for PostgreSQL. Each of the various platforms support different database encryption options. In this lesson students will explore these data encryption options and how to configure them.

## Learning Objectives

After taking this lesson, you will:

- Understand the data encryption options available in the various platforms
- Understand how to configure data at rest encryption for Microsoft SQL Server
- Understand how to configure data at rest encryption for Azure SQL Database
- Understand how to configure data at rest encryption for Azure Database for PostgreSQL
- Understand how to configure object level encryption for Microsoft SQL Server and Azure SQL Database
- Understand what Always Encrypted is used for and how to configure it
- Understand what Always Encrypted Enclaves are used for
- Understand what Dynamic Data Encryption is used for and how to configure it
- Understand how to configure SQL Server to use Azure Key Vault

## Transparent Data Encryption

Microsoft SQL Server's Transparent Data Encryption (TDE) encrypts all the data within a target database. The data in the database is encrypted as the data is written to the data page and decrypted when the data page in memory is accessed. The end result is that all data pages on disk are encrypted. In the event of any database file being accessed by someone who is not authorised, the database file will not be readable. Database backups will also be encrypted as a backup operation just copies the data pages from the database file to the backup device. No decryption is done during the backup operation.

With Azure SQL Database enabling TDE is extremely simple. Databases that are created in Azure SQL Database after May 2017 have TDE enabled automatically. Databases that were created before May 2017 will have TDE disabled by default and TDE will need to be manually enabled on these databases. When using Azure SQL Managed Instance, databases which were created after February 2019 have TDE enabled. Databases created before February 2019 will have TDE disabled. Enabling TDE within an Azure SQL Database database is simply a matter of editing the database within the Azure Portal. From the "Transparent Data Encryption" pane select to enable data encryption.

The screenshot shows the Azure portal interface for managing database encryption. At the top, there are three buttons: 'Save' (with a blue checkmark), 'Discard' (with a red X), and 'Feedback'. Below this, a shield icon with a yellow padlock is displayed next to the text: 'Transparent data encryption encrypts your databases, backups, and logs at rest without any changes to your application. To enable encryption, go to each database.' A blue 'Learn more' button is below the text. Underneath, a section titled 'Data encryption' has a blue 'ON' button and a white 'OFF' button. Further down, 'Encryption status' is shown with a green checkmark and the word 'Encrypted'.

*Figure 4 Transparent Data Encryption Settings for an Azure SQL Database*

By default, databases within Azure SQL Database are encrypted using a Microsoft provided certificate. Microsoft Azure does provide a Bring Your Own Key option which allows you to use a certificate which was created by your company and uploaded to Azure. If your company removes the certificate from Azure then the database connections will be closed, and there will be no access to the database.

Enabling TDE within a Microsoft SQL Server Database is an easy process as only a few T-SQL commands are required. This involves setting a master key within the master database using the CREATE MASTER KEY ENCRYPTION command. After that, a certificate is created in the master database which will be used for the encryption using the CREATE CERTIFICATE command. A database encryption key is then created within the database which allows you to enable TDE with the CREATE DATABASE ENCRYPTION KEY command. Once the encryption key is created, it needs to be enabled using the ALTER DATABASE command. The entire set of commands is shown below.

```
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Pa55.w.rd';
GO
CREATE CERTIFICATE MyServerCert
WITH SUBJECT = 'TDEDemo_Certificate';
GO
USE TDE_Demo;
GO
```

```
CREATE DATABASE ENCRYPTION KEY  
  
WITH ALGORITHM = AES_256 ENCRYPTION BY SERVER CERTIFICATE MyServerCert;  
  
GO  
  
ALTER DATABASE TDE_Demo SET ENCRYPTION ON;  
  
GO
```

Once TDE is enabled it will take some time in order to encrypt the database as each database page must be read and encrypted and written back to disk. The larger the database the longer this operation will take. This process is a background process and is run at a low priority in order to not overload the IO or the CPU of the system.

Once the certificate that will be used by TDE has been created, it must be manually backed up and stored in a safe place. SQL Server integrates with Enterprise Key Managers (EKMs) in order to manage encryption keys. An example of an EKM is Azure Key Vault; you can find instructions on integrating SQL Server TDE and Azure Key Vault in the additional resources section of this module. Managing the certificate is extremely important, because if the certificate is lost and the database needs to be restored from a backup, the restore will fail, as the database cannot be read. To use TDE with databases in an Always On Availability Group, the certificate used to encrypt the database must be backed up and restored to the other servers within the Availability Group which will be hosting copies of the database.

## What is Encryption at Rest

It is important to understand exactly what encryption at rest entails. Encryption at rest does not inherently encrypt data within the database. It provides protection against someone restoring a backup to an unsecured server or making a copy of a database and transaction log file and attaching it to another unsecured server. Encryption at rest does not protect data within a database from user access or prevent data exfiltration by a malicious user.

### *Encryption at Rest for MySQL and PostgreSQL*

Azure SQL Database for MySQL and Azure SQL Database for PostgreSQL do not have a TDE like process. However, the Azure environment in which they are run supports a disk encryption method. This process encrypts the databases that are stored on the disk, so that if the disk hosting the database was compromised the data on it would be unreadable. This service offers a Bring Your Own Key service, much like TDE within Azure SQL Database does. This allows you to upload your own certificate and have this certificate be used for the encryption.

## Key Vault Encryption and Disk Encryption for Virtual Machines

SQL Server, either within an Azure Virtual Machine or on-premises, supports using Azure Key Vault to store certificates for features such as Transparent Data Encryption, Backup Encryption, or Always Encrypt-

ed. While this configuration is somewhat complex in an on-premises environment (the documentation link is in additional resources), is quite easy using the SQL VM Resource Provider as shown in Figure 5.

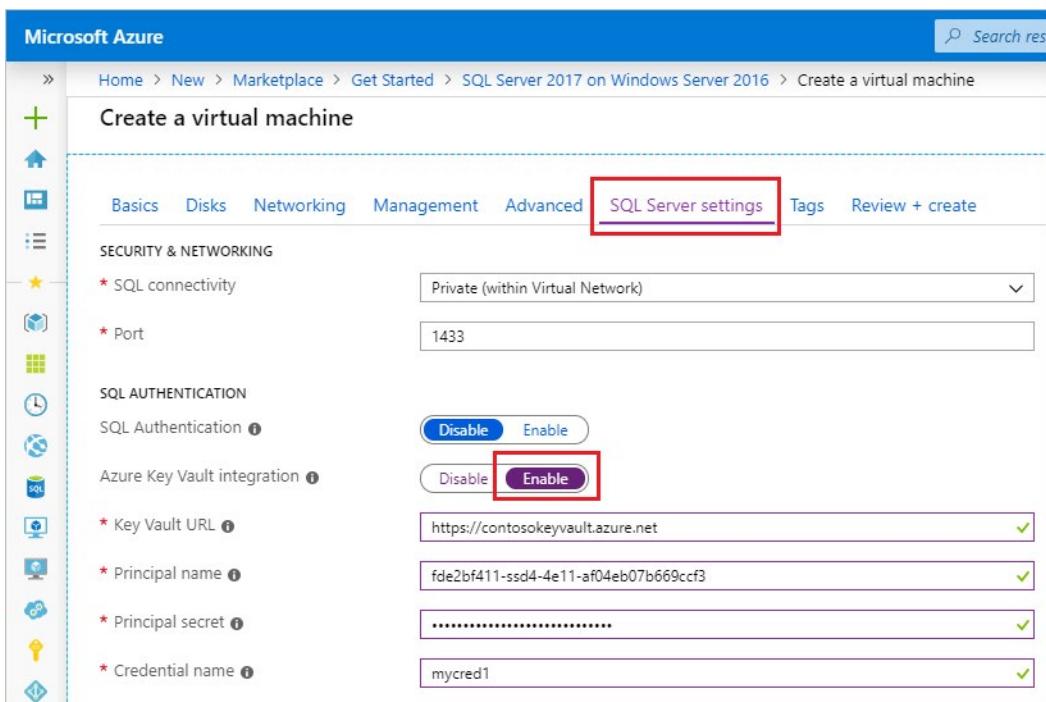


Figure 5 Azure SQL VM Resource Provider Azure Key Vault configuration

In order to configure the Azure Key Vault integration, the Key Vault URL, the Principal name, the Principal secret and the name of the credential. This task can be done at VM creation or to an existing VM.

Configuring SQL Server to connect to Azure Key Vault first requires creating a normal SQL Server Login within the instance. Next a Credential needs to be created and mapped to the login. For the identity of the credential the name of the key vault should be used. For the secret of the credential use the application ID from Azure Key Vault. Once the credential is created an asymmetric key can be created within the Azure Key Vault. An asymmetric key can then be created within the SQL Server database which maps to the Azure Key Vault asymmetric key using the CREATE ASYMMETRIC KEY command with the FROM PROVIDER syntax. Once an asymmetric key is created within the database that key can be used for TDE, or Backup Encryption or Always Encrypted.

## Azure Disk Encryption

In addition to these SQL Server security features, Azure VMs include an additional layer of security, Azure Disk Encryption—a feature that helps protect and safeguard data and meet organization and compliance commitments. If you are using TDE, your data is protected by multiple layers of encryption. Azure Disk Encryption and encryption of the SQL Server database files and backup.

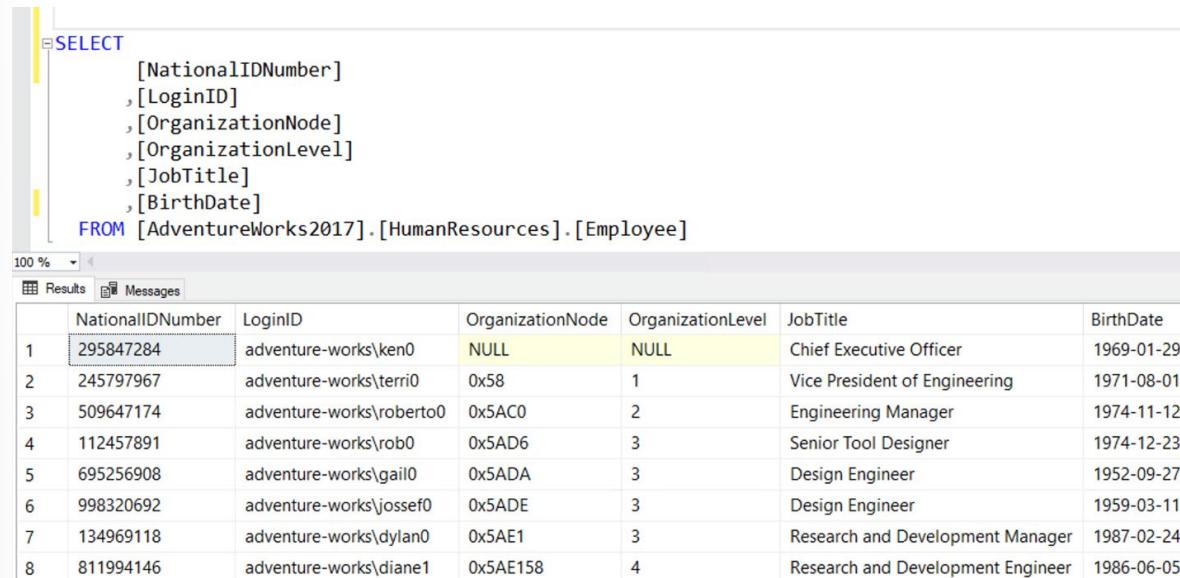
## What is Azure Key Vault

Azure Key Vault is a tool used for storing and accessing secrets. Whether they be passwords, certificates or keys, Key Vault acts as a secure area for those secrets to be accessed in a secure fashion, typically programmatically. Key Vault data has its own RBAC policies, separate from the Azure subscription. This means someone who is in the role of subscription admin will not have access to the Key Vault unless explicitly granted.

## Object Level Encryption

In addition to supporting encryption at rest through transparent data encryption, SQL Server supports encrypting data within columns using Always Encrypted. Once data is encrypted, the application accessing the database must have the correct certificate in order to view the plain text values of the data."

Always Encrypted allows for the encryption of data within the client application automatically as needed based on the settings within the Microsoft SQL Server database which tell the application what the encryption settings on the database column are. Always Encrypted is based on a master encryption key as well as a column encryption key. This allows each column to be encrypted using a different encryption key for maximum data protection. Always Encrypted has a variety of key stores which can store the certificate which is used by encryption. Here's an example of enabling Always Encrypted. As shown in Figure 5, you can see that *NationalIDNumber* and *BirthDate* are both in plain text.



The screenshot shows a SQL Server Management Studio window with a query editor and a results grid. The query is:

```
SELECT [NationalIDNumber], [LoginID], [OrganizationNode], [OrganizationLevel], [JobTitle], [BirthDate] FROM [AdventureWorks2017].[HumanResources].[Employee]
```

The results grid displays eight rows of data from the Employee table, showing the NationalIDNumber, LoginID, OrganizationNode, OrganizationLevel, JobTitle, and BirthDate columns. The NationalIDNumber and BirthDate columns are displayed in plain text.

	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	BirthDate
1	295847284	adventure-works\ken0	NULL	NULL	Chief Executive Officer	1969-01-29
2	245797967	adventure-works\terri0	0x58	1	Vice President of Engineering	1971-08-01
3	509647174	adventure-works\roberto0	0x5AC0	2	Engineering Manager	1974-11-12
4	112457891	adventure-works\rob0	0x5AD6	3	Senior Tool Designer	1974-12-23
5	695256908	adventure-works\gail0	0x5ADA	3	Design Engineer	1952-09-27
6	998320692	adventure-works\jossef0	0x5ADE	3	Design Engineer	1959-03-11
7	134969118	adventure-works\dylan0	0x5AE1	3	Research and Development Manager	1987-02-24
8	811994146	adventure-works\diane1	0x5AE158	4	Research and Development Engineer	1986-06-05

Figure 6 Query from Unencrypted table

The next few images will show you how we can encrypt both of these columns using Always Encrypted. The encryption could be done using T-SQL, but in this example, you will see the wizard from SQL Server Management Studio. You can reach the wizard by right clicking on the table name in Object Explorer as shown in Figure 6.

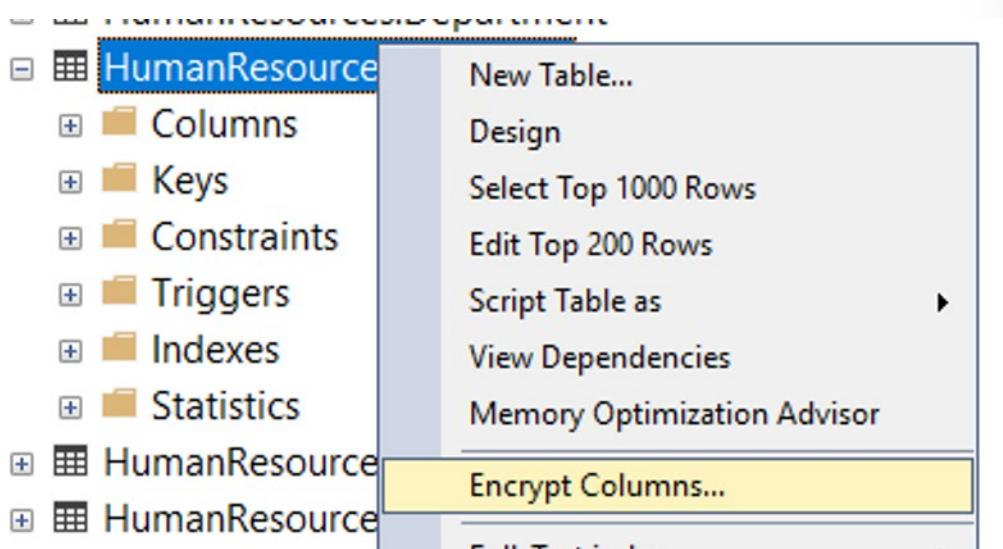


Figure 7 Launching the Encryption Wizard in SQL Server Management Studio

When you click on Encrypt Columns the wizard will launch.

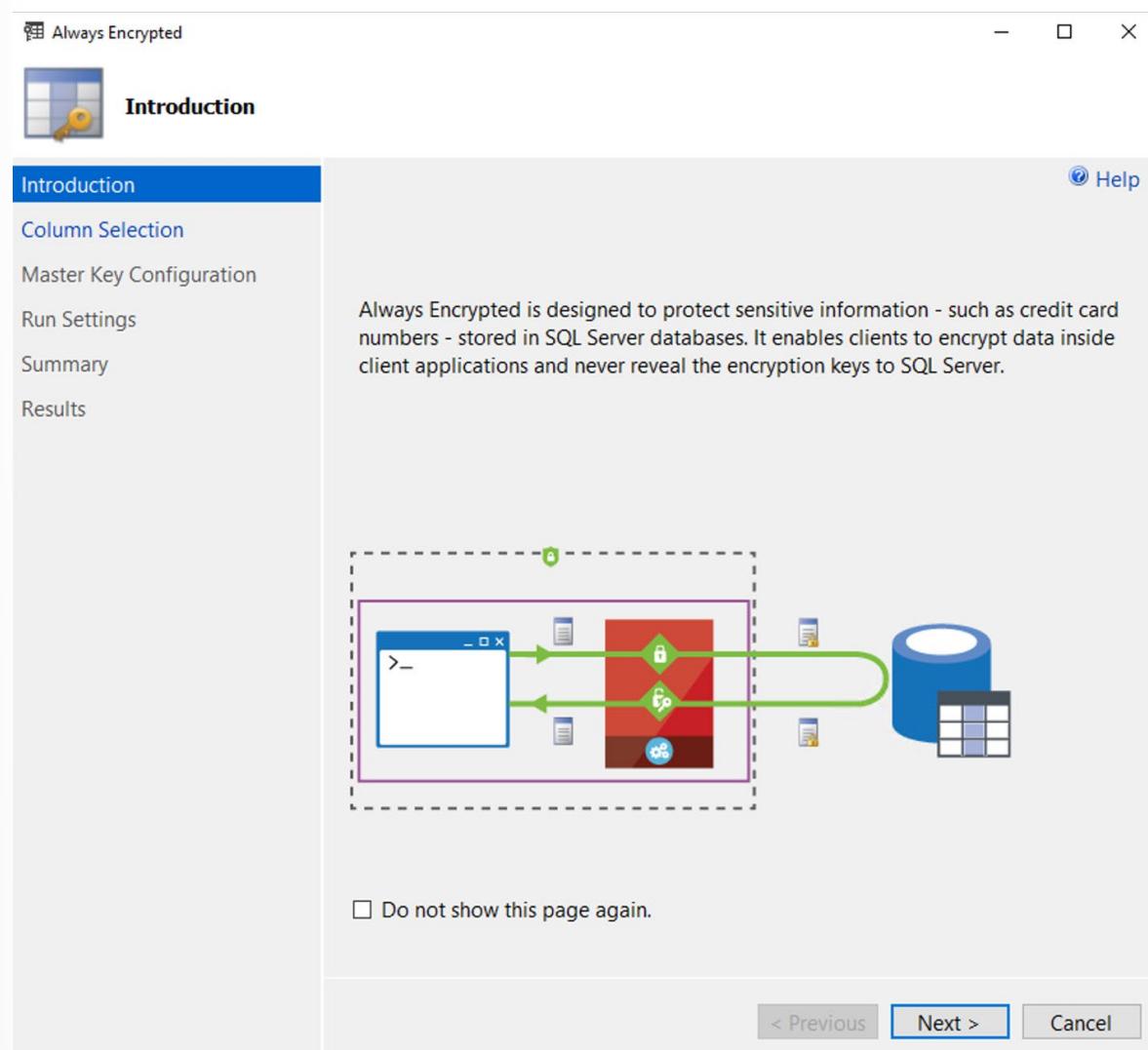


Figure 8 Always Encrypted Wizard launch screen

In Figure 8, you will see the Always Encrypted launch screen. Click next to choose the columns you want to encrypt.

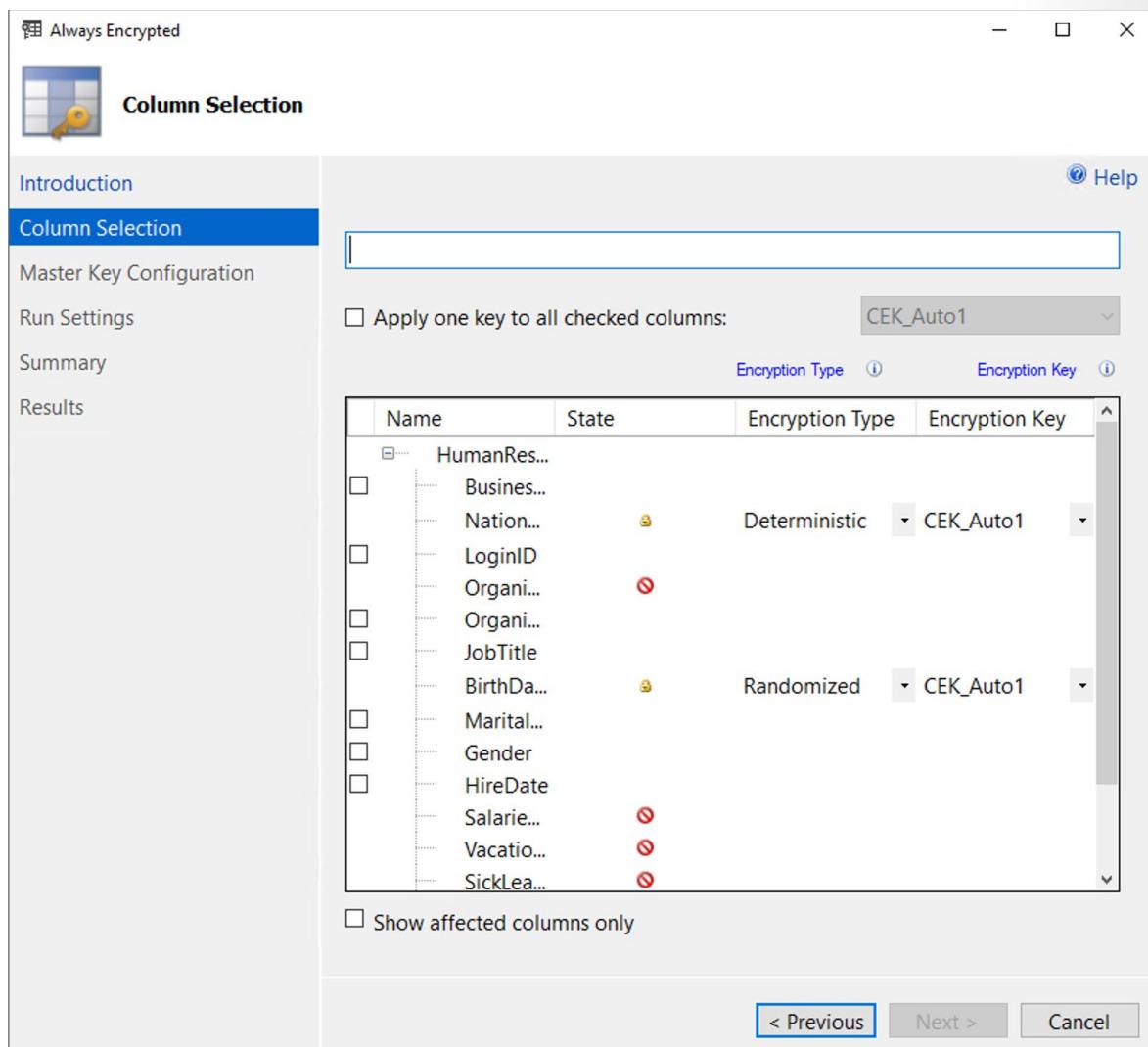


Figure 9 Always Encrypted Column selection

In Figure 9, note that there are two different types of encryption specified. The *NationalIDNumber* column is encrypted with Deterministic encryption, and the *BirthDate* column is encrypted using Randomized encryption. Randomized encryption is more secure than Deterministic encryption, but is more limited. The type of encryption cannot be changed after the column is created. You would use Randomized encryption for columns that had a small number of well-known distinct values that could potentially be guessed by someone with access to the encrypted values. An example of a potentially guessable column would a three-digit credit card verification code.

Using Always Encrypted with Randomized encryption is more limited because the randomization means that the same value is not always encrypted the same way. The only thing you can do with columns with Randomized encryption is to return them in your results. With Deterministic encryption, a given value always is encrypted to the same string so we can compare columns to a constant using equality and inequality operators, and we can compare columns with other columns to do joins, grouping, and indexing.

Always Encrypted with secure enclaves (supported in SQL Server 2019) addresses these limitations and enables pattern matching, comparison operations and indexing on columns using Randomized encryption.

Another thing to note from Figure 9 is that the wizard is generating a column encryption key which is the key that actually performs the data encryption. Each column being encrypted may have its own key, or as shown here, you can use the same key to encrypt both columns.

After identifying the columns you are encrypting you can click next and you will see the Master Key Configuration screen shown in Figure 10.

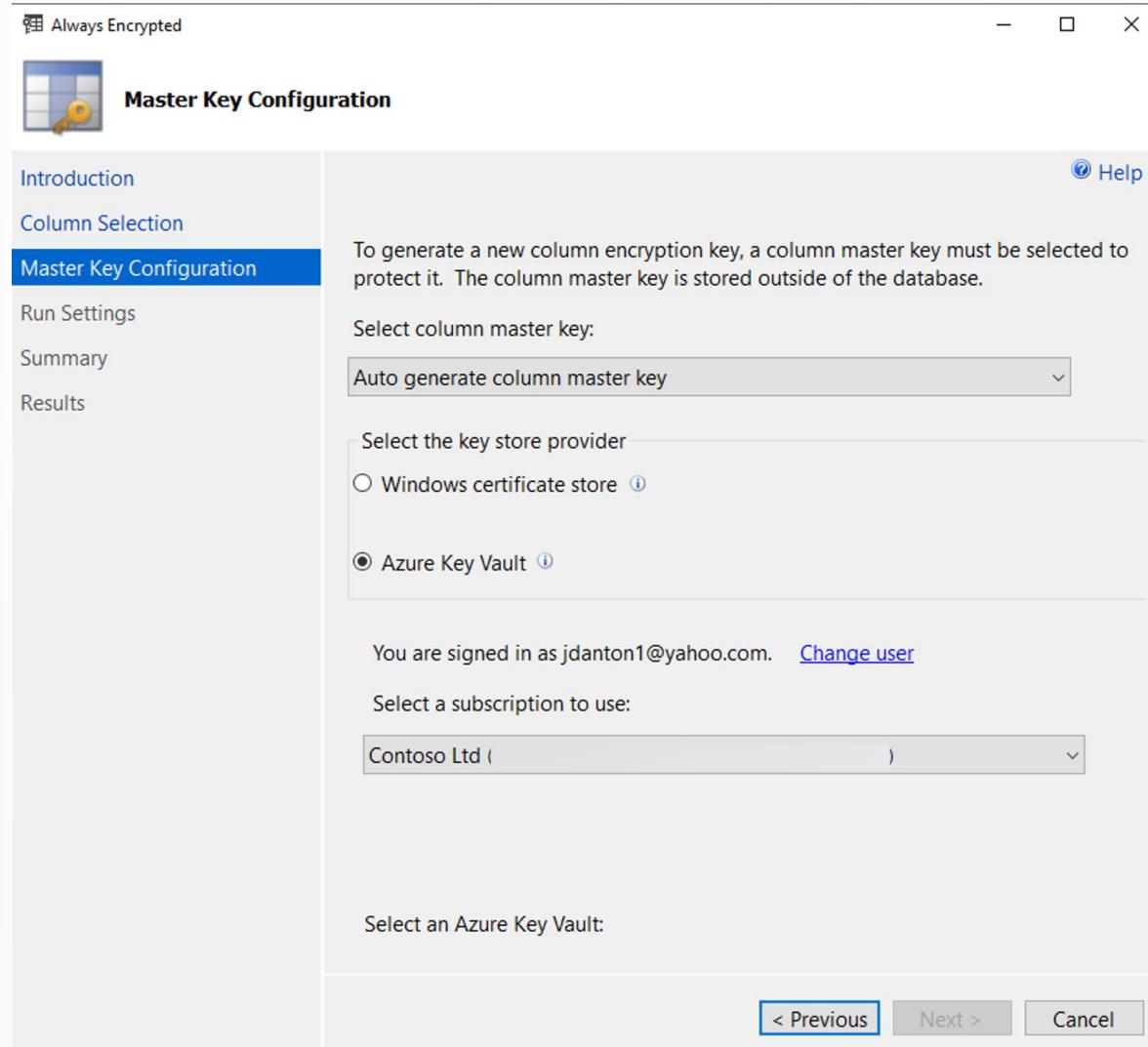
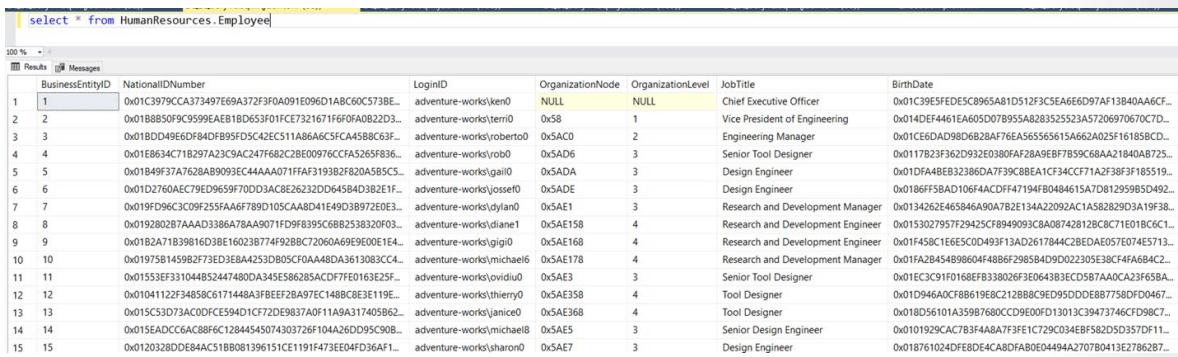


Figure 10 Master Key Configuration

In this screen, you create the column master key, which is used to encrypt the column encryption keys. You can supply your own key, if you are using T-SQL to encrypt the columns. This key must be stored in a key store such as the Windows Certificate Store, Azure Key Vault, or a hardware security module. The database engine never stores the column master key, and only contains the metadata about where it is stored. This is the feature that protects data access from users who have full access to the database.

For the highest level of security, the key should be stored within a 3<sup>rd</sup> party key store such as Azure Key Vault. You should also never generate the keys on the server hosting your database, as the key could potentially be extracted from memory on that server. In the example in Figure 10, the key is being stored in Azure Key Vault. After clicking next, the wizard provides you the option to either finishing the encryption process now, or to generate a PowerShell script. Once you complete the process, the data will appear as encrypted ciphertext to anyone querying the data without the key as shown in Figure 11.



The screenshot shows a SQL Server Management Studio (SSMS) window displaying the results of a query against the 'HumanResources.Employee' table. The results grid contains 15 rows of data, each with columns for BusinessEntityID, NationalIDNumber, LoginID, OrganizationNode, OrganizationLevel, JobTitle, and BirthDate. The data is heavily redacted with numerous black bars, demonstrating the use of Always Encrypted.

		BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	BirthDate
1	1	0x01C397CCAA73497E69A372F3FOA091E096D1ABC60C573BE...	adventure-works\ken0	NULL	NULL	Chief Executive Officer	0x01C39E5FED5C8965A1D512F3C5E66E6D97AF13B40AA6...	
2	2	0x01B8B50F9C9599EAE1B0D653F01FC7326716F0FA0822D3...	adventure-works\tim0	0x58	1	Vice President of Engineering	0x014DEF4461EA605D07B955A828352523A57206970670C7...	
3	3	0x01BDD49E6DF84DF95FD5C42EC511A86C5FC4A5B8C63F...	adventure-works\yoberto0	0x5AC0	2	Engineering Manager	0x01CE6DAD98D6B28AF76EA65565615A662A025F16185BCD...	
4	4	0x01E8634C71B297A23C9CA247F682C2BE00976CFA5265F836...	adventure-works\rob0	0x5AD6	3	Senior Tool Designer	0x0117B23F362D932E0380FAF28A9E8FT859C66AA21B40AB725...	
5	5	0x01B49F37AT7628AB9093EC44AAAD71FFAF3193B2F620A585C5...	adventure-works\gail0	0x5ADA	3	Design Engineer	0x01DFA48EB323860DA7F39C88EA1CF34CC771A2F38F3F185519...	
6	6	0x01D27640AC79E965970D0JACE8E2622D064584D282E1...	adventure-works\josef0	0x5ADE	3	Design Engineer	0x0186FF58BAD1064FACDF47194FB0484615A7D08129595D492...	
7	7	0x019FD96C3C09F255FA6A6789D105CA8A041E49D38972E0E3...	adventure-works\dylan0	0x5AE1	3	Research and Development Manager	0x0134262E465846A90A7B2E134A22092AC1A582829303A19F38...	
8	8	0x019280287AAAD3386A78A9071FDF8395C6B25382D0F03...	adventure-works\diane1	0x5AE158	4	Research and Development Engineer	0x0153027957729425CF8949093CA0A8742812B8C71E01BC6C1...	
9	9	0x01B2A71B39816C38E16023877491928BC2060A69E9E001E14...	adventure-works\gigi0	0x5AE168	4	Research and Development Engineer	0x01F458C1E65C0D4931F1AD2617844C28EDAE057E045713...	
10	10	0x01975B145982773ED3E8A4253D05CFOA48BD43613085CC4...	adventure-works\michael0	0x5AE178	4	Research and Development Manager	0x01FA2B454898604F4886F2985D49D022305E38CF4FA684C2...	
11	11	0x01553E7331044852447480D0A345E586285ACDF7FE0163E25F...	adventure-works\ovidiu0	0x5AE3	3	Senior Tool Designer	0x01EC3C91F0168EF338380263F0E043B3ECD587AA0CA23F658A...	
12	12	0x01041122F34B58C617448A3F3BEF2B9A7C14B8C8E3119E...	adventure-works\thierry0	0x5AE358	4	Tool Designer	0x01D94640C88B6198EBC212B8C9E095DDE887758DFD0467...	
13	13	0x015CS3D73AC0DCE594D1C772DE9837A0F11A9A317405B62...	adventure-works\janice0	0x5AE368	4	Tool Designer	0x018D56101A359876800CCD9E00FD13013C39473746FCF98C7...	
14	14	0x015EADCCGAC88F6C12844545074303726F104A266D95C90B...	adventure-works\michael8	0x5AE5	3	Senior Design Engineer	0x0101929CAC7B3F4ABA7F73F1C729C034EBF582D5357D11...	
15	15	0x0102382DDE84C51BB081396151CE1191F473E04FD364F1...	adventure-works\sharon0	0x5AE7	3	Design Engineer	0x018761024DFE8DE4C48DFA80E04494A2707B0413E27862B7...	

Figure 11 Employees Table with Encrypted Data

In order to decrypt data from an Always Encrypted column, your application needs an Always Encrypted driver to connect to the database. The application has access to the key store where the Always Encrypted keys are stored, and the application can then retrieve the data. Data that is written back to the database is encrypted at the client through the driver.

In addition to the driver, the application's connection string need to have the setting "Column Encryption Setting=enabled" placed within it. This will cause a metadata lookup to be made for each column that is used by the application. To minimize these metadata lookups the application needs to be modified by updating the *SqlCommandColumnEncryptionSetting* on the *SqlConnection* objects within the .NET application. Values of either Disabled, Enabled or *ResultSet* can be specified. These settings must be set for each database query that the application submits.

The Disabled setting will prevent the application from executing any metadata queries against the database for that query.

The Enabled setting will cause the application to execute the metadata queries for all columns in the SELECT clause of the T-SQL statement as well as the WHERE clause of the T-SQL statement.

The *ResultSet* setting will cause the metadata queries to be executed for only the values within the SELECT statement.

So to determine which setting of the *SqlCommandColumnEncryptionSetting* should be used, you need to know where in the statement the encrypted columns occur.

## What are Secure Enclaves?

Starting with SQL Server 2019, Always Encrypted supports a feature called secure enclaves, which allows more robust querying of encrypted data. A secure enclave is a secured region of memory within the SQL Server process which acts as trusted execution environment for processing encrypted data. This enclave appears as black box to SQL Server, and it is not possible to view any data or code, even with a debugger. Figure 12 shows the architecture of this process.

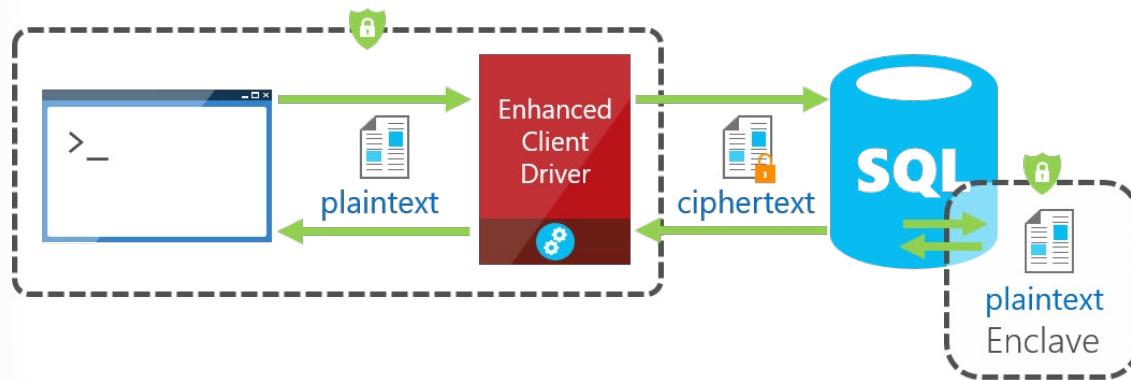


Figure 12 Secure Enclaves Architecture

Secure enclaves allow pattern matching and range comparison on data that is stored in columns using Randomized encryption.

## Dynamic Data Masking

Dynamic Data Masking allows users to view secure data without viewing the entire value. This allows users who do not need to see sensitive data such as credit card numbers, tax identification numbers, etc. to view the column which contains the data, but without seeing the actual data which is stored in the table. It is important to note that Dynamic Data Masking is a presentation layer feature, and that unmasked data will always be seen by administrators. The best use case of Dynamic Data Masking is to mask data from application users who have no direct access to the database.

Dynamic Data Masking can be implemented in the Azure Portal, or using T-SQL as shown in Figure 13.

```
ALTER TABLE [Application].[People] ALTER COLUMN [PhoneNumber] ADD MASKED WITH (FUNCTION = 'partial(0,"XXX-XXX-",4)')
ALTER TABLE [Application].[People] ALTER COLUMN [EmailAddress] ADD MASKED WITH (FUNCTION = 'email()')

EXECUTE AS USER = 'DDMDemo'

SELECT [PersonID]
      ,[FullName]
      ,[PhoneNumber]
      ,[EmailAddress]
  FROM [WideWorldImporters].[Application].[People]
 WHERE PhoneNumber is NOT NULL
```

Results

PersonID	FullName	PhoneNumber	EmailAddress
1	Kayla Woodcock	XXX-XXX-0102	kXXX@XXXX.com
2	Hudson Onslow	XXX-XXX-0102	hXXX@XXXX.com
3	Isabella Rupp	XXX-XXX-0102	iXXX@XXXX.com
4	Eva Muirden	XXX-XXX-0102	eXXX@XXXX.com
5	Sophia Hinton	XXX-XXX-0102	sXXX@XXXX.com
6	Amy Trefl	XXX-XXX-0102	aXXX@XXXX.com
7	Anthony Grosse	XXX-XXX-0102	aXXX@XXXX.com

Figure 13 Dynamic Data Maintain

In the above example, both the phone number and email address are hidden from a user named DDM-Demo who only has SELECT access on the table. The user is allowed to see the last four digits of the phone number as its masked using a 'partial' function that replaces all but the last four digits in the

column. This is considered to be a custom function. In addition to T-SQL, if you are using Azure SQL Database, you can create dynamic masking rules in the Azure Portal as shown in Figure 14.

The screenshot shows the 'Add masking rule' interface in the Azure Portal. At the top, there are 'Add', 'Discard', and 'Delete' buttons. Below that, the 'Mask name' field contains 'HumanResources\_Employee\_NationalID ...'. The 'Select what to mask' section includes dropdowns for 'Schema' (HumanResources), 'Table' (Employee), and 'Column' (NationalIDNumber (nvarchar)). Under 'Select how to mask', the 'Masking field format' dropdown is set to 'Default value (0, xxxx, 01-01-1900)', which is highlighted with a blue border. Other options listed in the dropdown menu include 'Credit card value (xxxx-xxxx-xxxx-1234)', 'Email (aXXXX@XXXX.com)', 'Number (random number range)', and 'Custom string (prefix [padding] suffix)'.

Figure 14 Add Masking Rule Screen in Azure Portal

You can reach the add masking rule screen by navigating to your database in the Azure Portal and selecting Dynamic Data Masking in the security section of the main blade for your database.

Dynamic Data Masking supports a variety of masking patterns that can be used. These include Default, Credit Card, Social Security Number, Random Number, and Custom Text.

The **default** masking option fully masks the data in the column without exposing any part of the values to the user. The user would see XXXX for string values, 0 for numbers, and 01.01.1900 for date values.

The **Credit Card** masking option masks all but the final four characters allowing users to view the final four digits. This can be useful for customer service agents who need to view the last four digits of a credit card number but who do not need to see the entire number. The data is shown in the usual format of a credit card number XXXX-XXXX-XXXX-1234.

The **Social Security Number** option masks all but the final four characters, with the masked data showing as the United States Social Security Number in the format XXX-XX-1234.

The **Random Number** masking option should be used on numeric columns. It shows a random number as the masked value instead of the actual value. Each time the record is queried a different number is displayed.

The **Custom Text** masking option allows for any rules that are needed to be specified. This allows text to be masked with any value, and to display a custom number of characters at either end of the masked value. If the value being masked is equal to or less than the number of characters which the mask specifies are to be displayed, then only the masked character is displayed.

A good use case of Dynamic Data Masking is for exporting a copy of your production database to a lower environment for development purposes, which may have fewer security restrictions. If you execute the export process using the credentials of a lower privileged user who doesn't have UNMASK permissions, the data will be exported in its masked format.

## Summary and Knowledge Check

### Question 1

*Which security object is required in order to enable transparent data encryption?*

- Credential
- Master Key
- Login

### Question 2

*Which feature prevents members of the sysadmin role from viewing the values of data in a table?*

- Always Encrypted
- Dynamic Data Masking
- Transparent Data Encryption

### Question 4

*Which is a valid location for Always Encrypted keys?*

- Azure Key Vault
- Azure Automation
- Azure Blob Storage

# Implement Security for Data in Transit

## Introduction

This module explores the practices of securing the firewalls of an Azure SQL Server. It will also show you how Always Encrypted is used to protect data in transit.

### Learning Objectives

After taking this lesson, you will:

- Understand the difference between database and instance firewalls in Azure SQL Database
- Understand how Always Encrypted protects data in transit
- Understand how Secure Enclaves are used by SQL Server

## Firewalls

Firewalls are used to prevent unauthorized users from accessing the protected resources. Each Azure SQL Database maps to a public IP address which is hosted by Microsoft. Each Azure region will have one or more public IP addresses where you can reach your database gateway, which will then take you to your database. In order to protect your database and your data, Azure provides built-in firewalls to limit access. In Azure SQL Database there are two sets of firewall rules, server level firewall rules and database level firewall rules. Both server and database level firewalls use IP Address rules instead of SQL Server Logins. This allows all users at the same public IP Address to access the SQL Server. At most companies this will be the outbound IP address of the company..

Server level firewalls are configured to allow users to connect to the master database and all databases on the instance. Database level firewalls are used to grant or block specific IP Addresses from accessing specific databases.

Server level firewall rules can be configured using the Azure portal or using the `sp_set_database_firewall_rule` stored procedure from within the master database. Database level firewall rules are configured through T-SQL only using the `sp_delete_database_firewall_rule` stored procedure from within the user database. Upon connection, Azure SQL Database will look first for a server level firewall rule in the master database and then a database level firewall rule, if the connection string specified a database name. If either of these exist, the connection will be completed. If neither exist and the user is connecting through SQL Server Management Studio or Azure Data Studio, if the user authenticates to the database, they will be prompted to create a firewall rule as shown in Figure 15 below.

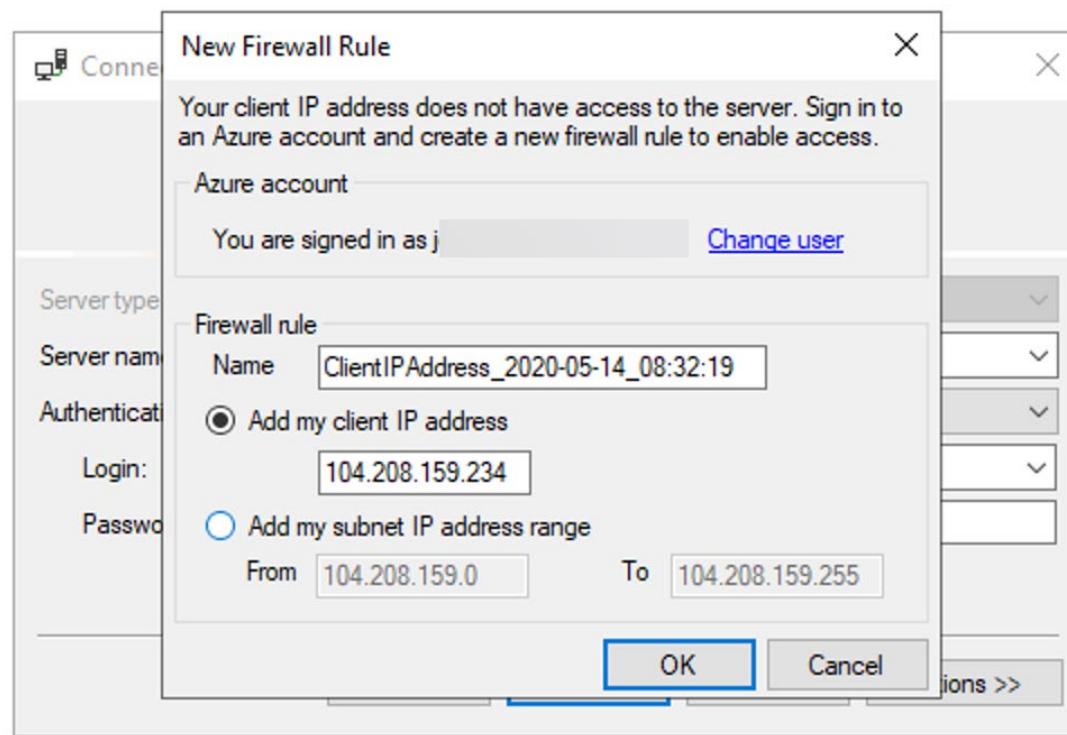


Figure 15 New Firewall Rule Screen from SQL Server Management Studio

## Virtual Network Endpoints

Virtual network endpoints allow traffic from a specific Azure Virtual Network. These rules apply at the server level, not just the database level. Additionally, the service endpoint applies to only one region, which is the underlying endpoint's region. One additional concern is that the virtual network that is connecting to the Azure SQL Database must have outbound access to the public IP address for Azure SQL Database, which can be configured using service tags for Azure SQL Database.

## Private Link

The Private Link feature allows you connect to Azure SQL Database (and other PaaS offerings) using a private endpoint. A private endpoint allows for a connection to your Azure SQL Database to go entirely over the Azure backbone network and not over the public internet. This feature provides a private IP address on your Virtual Network. Another feature of private link is that it allows for Azure Express Route connections through that circuit. Private link offers several additional benefits include cross-region private connectivity and protection against data leakage by only allowing connections to specific resources.

## Summary and Knowledge Check

### Question 1

*Which feature provides a private IP address for an Azure SQL Database?*

- Network Endpoints
- Private Link
- Database Firewall

### Question 2

*Which technique can be used to create database firewall rules in Azure SQL Database?*

- Running a PowerShell script
- Running a Azure CLI script
- Executing a T-SQL statement

### Question 3

*Which features allows for ExpressRoute connectivity to Azure SQL Database?*

- Private Link
- Network Endpoints
- Linked Server

# Implement Compliance Controls for Sensitive Data

## Introduction

This module explores the practices of setting compliance controls on the data that is stored within the SQL Server database. We will also review the Advanced Threat Protection options within the Azure environment.

## Learning Objectives

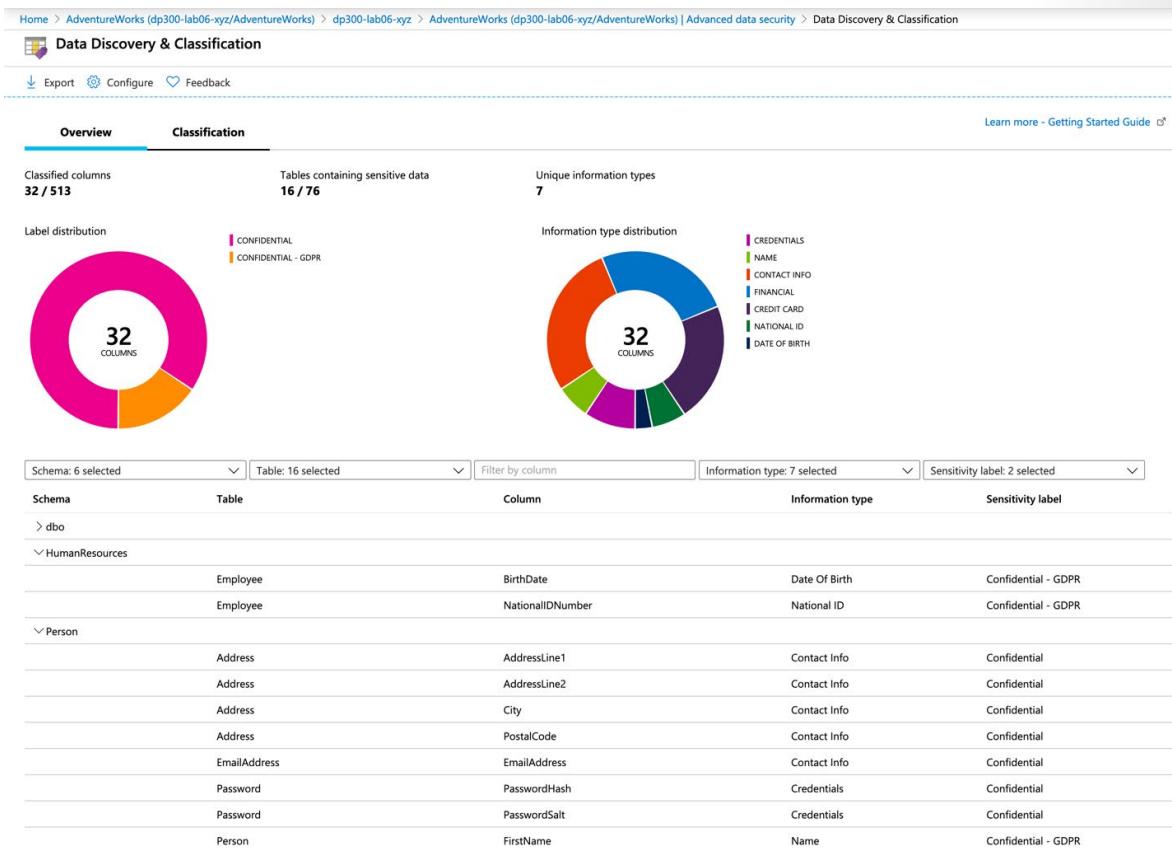
After taking this module, you will understand:

- How data should be classified
- Why data classification should be done
- What Azure Threat Protection does
- Why Azure Threat Protection should be used

## Data Classification

Confidential data stored within a Microsoft SQL Server or Azure SQL Database should be classified within the database. This allows the SQL Server users as well as other applications to know the sensitivity of the data that is being stored. Data classification with the database is done on a column by column basis. It is possible for a single table to have some columns be public, some columns be confidential and some columns be highly confidential. Data classification was first introduced into SQL Server Management Studio and used extended properties of objects to store its data classification information. Starting with SQL Server 2019 (and in Azure SQL Database) this metadata is now stored in a catalog view called sys.sensitivity\_classifications.

The Azure portal provides a management pane for data classification of your Azure SQL Database as shown in Figure 16 below. You can reach this screen by clicking Data Discovery and Classification in the Advanced Data Security screen, which is in the Security section of the main blade for your Azure SQL Database.



*Figure 16 Data Discovery and Classification in the Azure Portal*

In both the Azure Portal and SQL Server Management Studio, you can configure data classification. The classification engine scans your database and locates columns with names that indicate that the column could have sensitive information. An example would be that a column named email which would be classified by default as containing personally identifiable information. You should verify and validate this data, since it is based on column name, so a column named column1 that contained email addresses would not be classified as personally identifiable information.

Columns can be classified using the sensitivity wizard in SQL Server Management Studio, from the Advanced Data Security screen in the Azure Portal for Azure SQL Database, or by using the ADD SENSITIVITY CLASSIFICATION T-SQL command as shown below:

```
ADD SENSITIVITY CLASSIFICATION TO
[Application].[People].[EmailAddress]
WITH (LABEL='PII', INFORMATION_TYPE='Email')
GO
```

Classification of data allows you to easily identify the sensitivity of data within the database. Knowing what columns contain sensitive data allows for easier audits and allows you to more easily identify which columns are good choices for data encryption. This will allow other employees within the company to make better decisions on how to handle the data which is available within the database.

## Advanced Threat Protection

Microsoft offers a suite of protections for Azure SQL Database databases as part of the Advanced Threat Protection (ATP) feature. ATP monitors the connections to the Azure SQL Database databases and the queries which are executed against the database. You can configure ATP from the Azure Portal as shown in Figure 17. This screen is reached from the Security section of your databases main blade.

The screenshot shows two side-by-side configuration pages. The left page is titled 'Server settings' and shows 'dp300-lab06-xyz'. It has a 'Save' button, a 'Discard' button, and a 'Feedback' link. Below this is a 'ADVANCED DATA SECURITY' section with an 'ON' button. A note below it says: 'Advanced Data Security costs 15 USD/server/month. It includes Data Discovery & Classification, Vulnerability Assessment and Advanced Threat Protection. We invite you to a trial period for the first 30 days, without charge.' The right page is titled 'Advanced Threat Protect...' and shows a list of threat types with checkboxes: All, SQL injection, SQL injection vulnerability, Data exfiltration, Unsafe action, Brute Force, and Anomalous client login. The 'All' checkbox is checked. The background of the right page has a blue header bar with 'Learn more - Advanced Threat Protection alerts' and a close button.

Figure 17 Advanced Threat Protection in the Azure Portal.

To get maximum benefit out of Advanced Threat Protection you will want to enable auditing on your databases. Auditing will allow for deeper investigation into the source of the problem if ATP detects an anomaly. ATP supports alerts for the following threats:

**Vulnerability to SQL injection**—This alert looks for T-SQL code coming into your database that may be vulnerable to SQL injection attacks. An example would be a stored procedure call that did not sanitize user inputs.

**Potential SQL injection**—This alert is triggered when an attacker is actively attempting to execute a SQL injection attack.

**Access from unusual location**—This alert is triggered when a user logs in from an unusual geographic location.

**Access from unusual Azure data center**—This alert is looking for attacks from an Azure data center that is not normally accessed.

**Access from unfamiliar principal**—This alert is raised when a user or applications log in to a database that they have not previously accessed.

**Access from a potentially harmful application**—This alert detects common tools that are used to attack databases.

**Brute force SQL credentials**—This alert is triggered when there a high number of login failures with different credentials.

## What is SQL Injection

SQL Injection is one of the most common methods used for data breaches. The core of the attack is that an SQL command is appended to the back end of a form field in the web or application front end (usually through a website), with the intent of breaking the original SQL Script and then executing the SQL script that was injected into the form field. This SQL injection most often happens when you have dynamically generated SQL within your client application. The core reason behind an SQL Injection attack comes down to poor coding practices both within the client application and within the database stored procedures. Many developers have learned better development practices, but SQL Injection is still a big problem due to both the number of legacy applications still being used and newer applications built by developers who didn't take SQL Injection seriously while building the application.

As an example, assume that the front-end web application creates a dynamic SQL statement as shown below:

```
SELECT * FROM Orders WHERE OrderId=25
```

This T-SQL is created when the user goes to the sales order history portion of the company's website and enters 25 into the form field for the order ID number. However, suppose the user enters more than just an ID number, for example "25; DELETE FROM Orders;"

In that case the query sent to your database would be as shown below:

```
SELECT * FROM Orders WHERE OrderID=25; delete from Orders; DELETE FROM Orders;
```

The way the query in the above example works is that the SQL database is told via the semicolon ";" that the statement has ended and that there is another statement that should be run. The database then processes the next statement as instructed, which would result in the deletion of all values from the Orders table.

While the initial SELECT query is run as normal, and without any error being generated, when you look at the Orders table, you won't see any records because the second query in that batch to delete all the rows was also executed.

One technique used to prevent SQL Injection attacks is to inspect the text of the parameters, or values entered into the form fields, looking for various keywords. However, this only provides minimum protection as there are many, many ways to force these attacks to work. Some of these techniques include passing in binary data, having the database engine convert the binary data back to a text string, and then executing the string. This can be proven by running the T-SQL statement shown below

```
DECLARE @v varchar(255)

SELECT @v = cast(0x73705F68656C706462 as varchar(255))

EXEC (@v)
```

When data is being accepted from a user, either a customer or an employee, one good way to ensure that the value won't be used for an SQL Injection attack is to validate that the data that was entered is of the expected data type. If a number is expected, the client application should ensure that there is in fact a number being returned. If a text string is expected, then ensure that the text string is of the correct length, and it does not contain any binary data within it. The client application should be able to validate all data being passed in from the user, either by informing the user of the problem and allowing the user to correct the issue, or by crashing gracefully in such a way that an error is returned and no commands are sent to the database or the file system.

While fixing your application code should always be the priority, in some cases that may not be possible, so having Advanced Threat Protection can provide an additional layer of protection for your sensitive data.

## Summary and Knowledge Check

### Question 1

*Where is the data from data classification stored in SQL Server 2019?*

- In the extended properties for each object
- In the sys.sensitivity\_classifications catalog view
- In the sys.all\_columns catalog view

### Question 2

*Which of the following threats is analyzed by Advanced Threat Protection?*

- Weak passwords
- Open Firewall rules
- SQL Injection

### Question 3

*Which attack type is commonly associated with dynamic SQL?*

- SQL Injection
- Brute Force
- Data Exfiltration

## Module Summary

### Module Summary

Security is always an important concern but given the importance of databases in your application stack, configuring proper database security is of the essence. SQL Server and Azure SQL Database allow you easily encrypt your data files and backups at rest and encrypt the data within your tables using Always Encrypted. You can also use the Data Classification feature to help you better understand the data stored in your database and monitor data access. Advanced Threat Detection can protect you from a variety of attacks, such as SQL Injection.

# Answers

## Question 1

Which protocol is used by Azure Active Directory for Authorization?

- Kerberos
- LDAP
- OAuth

*Explanation*

## Question 2

Which database stores the information about logins in SQL Server?

- master
- model
- msdb

*Explanation*

## Question 3

Which role allows users to create users within a database

- db\_datareader
- db\_accessadmin
- db\_securityadmin

*Explanation*

## Question 1

Which permission allows the user to perform any option against a database object?

- Control
- Delete
- View Definition

*Explanation*

## Question 2

Which database object can be granted insert access?

- Functions
- Tables
- Procedures

*Explanation*

**Question 3**

What feature allows a user to execute a stored procedure even if she does not have permission to access the tables referenced in the stored procedure?

- Ownership chaining
- Principle of least privilege
- Granular security

*Explanation*

**Question 1**

Which security object is required in order to enable transparent data encryption?

- Credential
- Master Key
- Login

*Explanation*

**Question 2**

Which feature prevents members of the sysadmin role from viewing the values of data in a table?

- Always Encrypted
- Dynamic Data Masking
- Transparent Data Encryption

*Explanation*

**Question 4**

Which is a valid location for Always Encrypted keys?

- Azure Key Vault
- Azure Automation
- Azure Blob Storage

*Explanation*

**Question 1**

Which feature provides a private IP address for an Azure SQL Database?

- Network Endpoints
- Private Link
- Database Firewall

*Explanation*

**Question 2**

Which technique can be used to create database firewall rules in Azure SQL Database?

- Running a PowerShell script
- Running a Azure CLI script
- Executing a T-SQL statement

*Explanation*

**Question 3**

Which features allows for ExpressRoute connectivity to Azure SQL Database?

- Private Link
- Network Endpoints
- Linked Server

*Explanation*

**Question 1**

Where is the data from data classification stored in SQL Server 2019?

- In the extended properties for each object
- In the sys.sensitivity\_classifications catalog view
- In the sys.all\_columns catalog view

*Explanation*

**Question 2**

Which of the following threats is analyzed by Advanced Threat Protection?

- Weak passwords
- Open Firewall rules
- SQL Injection

*Explanation*

**Question 3**

Which attack type is commonly associated with dynamic SQL?

- SQL Injection
- Brute Force
- Data Exfiltration

*Explanation*

## Module 4 Monitor and Optimize Operational Resources

### Module-Introduction

#### Module introduction

This module will teach you about resource optimization for your databases created using either IaaS or PaaS services. The module also covers monitoring server and hardware resources. It will familiarize you with the various tools available for monitoring performance and establishing a baseline. You will learn how to interpret performance metrics for the most critical resources. You will also learn how to troubleshoot database performance using Azure SQL Database Intelligent Insights.

#### Learning Objectives

#### Learning Objectives

After this module, you will be able to:

- Monitor activity and compare to a baseline
- Define maintenance tasks related to performance
- Identify major causes of performance problems
- Configure resources for optimal performance
- Configure a user database for optimal performance

# Baselines and Performance Monitoring

## Introduction

A major part of the job of a database administrator is proper performance monitoring. This task does not change when moving to a cloud platform. While Azure offers tools for monitoring, you may lack some specific controls around hardware that you would have in an on-premises environment which makes understanding how to identify and resolve performance bottlenecks while in Azure SQL that much more critical.

## Learning Objectives

- Understanding methods to review potential performance issues
- Identify critical Azure metrics
- Learn how to collect metrics for an established baseline
- Understand Azure SQL Database Intelligent Insights

## What is a Baseline?

A baseline is a collection of data measurements that helps you understand the normal “steady state” of your application or server’s performance. Having the data collected over time allows you to identify changes from normal state. Baselines can be as simple as a chart of CPU utilization over time, or complex aggregations of metrics to offer granular level performance data from specific application calls. The granularity of your baseline will depend on the criticality of performance of your database and application.

## Performance Monitoring Tools

Azure provides multiple methods to monitor the performance of your resources and create a baseline. Each method can be tailored for a specific metric. In this section we will examine Azure monitoring tools. The metrics that you can monitor will vary depending on the type of Azure resource you are monitoring. For example, depending on the Azure SQL and SQL Server on an Azure VM, will have different metrics available in the Azure portal.

The following set of examples are focused on an Azure Virtual Machine. When you deploy an Azure Virtual Machine from the Azure marketplace, an agent is installed in the VM which provides a basic set of operating system metrics that are presented to you in the Azure portal. This agent provides metrics into a service called Azure Monitor which is a comprehensive platform monitoring solution that collects and displays a standard set of metrics from Azure resources. In the case of a VM, the default metrics captured are CPU, network utilization, and disk read and write operations. You can capture additional metrics beyond what is captured in Azure monitor by enabling Monitoring Insights for your VM as shown in Figure 1.

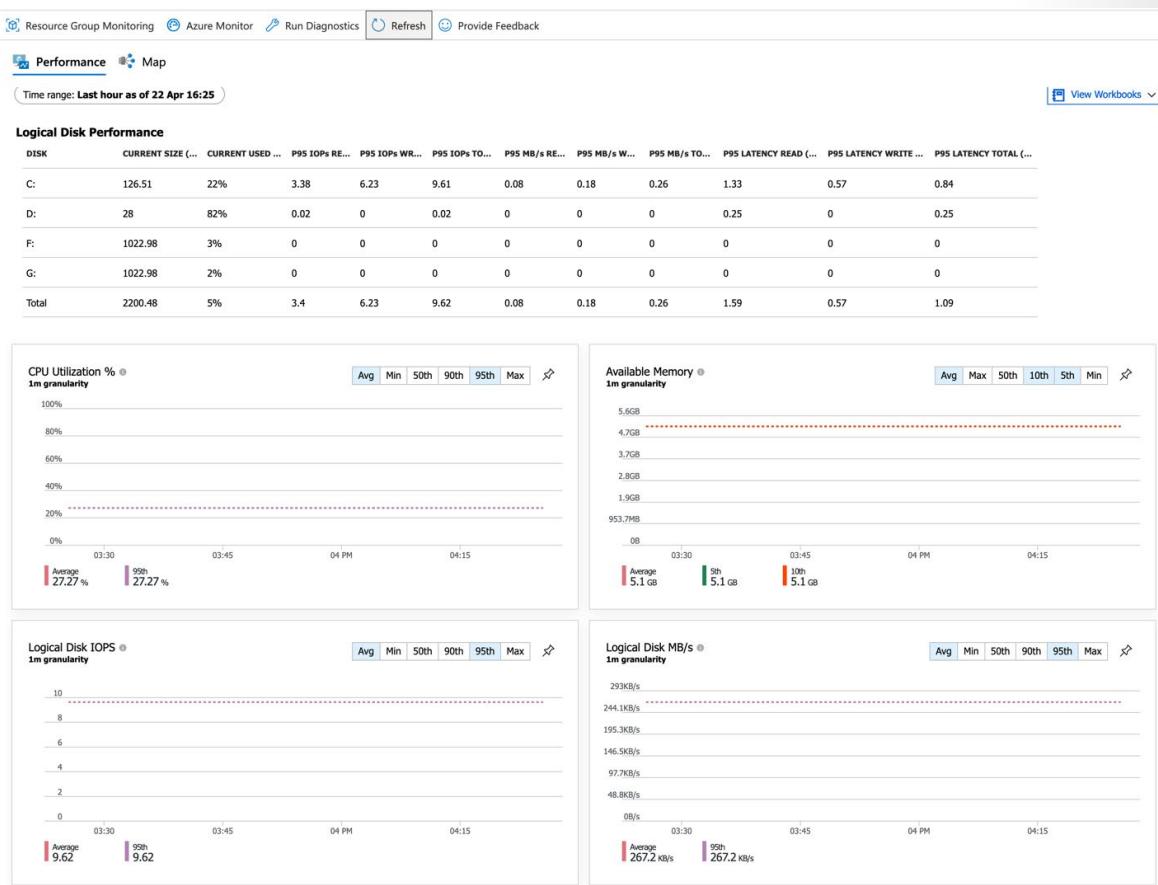


Figure 1 Azure Insights for an Azure VM

It is worth noting that these metrics pertain to the operating system, not SQL Server. You'll notice that the namespace for each metric is the virtual machine host, not SQL Server.

This means that you are unable to view SQL Server specific metrics from within the portal. For detailed SQL Server specific metrics you will need to gather them from the virtual machine itself.

Azure Monitoring Insights allows you to collect additional data points like storage latency, available memory, and disk capacity. These Azure Monitor Insights can be one way of viewing a baseline of performance for your Azure VM including I/O performance, memory and CPU utilization. This data is stored in an Azure Log Analytics workspace. Azure Log Analytics is the primary tool in Azure for storing and querying log files of all kinds. Log Analytics is queried by a SQL-like language called Kusto Query Language (KQL).

If you create a VM with one of the pre-configured SQL Server images in the Azure marketplace, you will also get the SQL VM resource provider as shown in Figure 2. You can launch this screen in the Azure Portal by clicking on the SQL Server configuration option in the settings section of the main blade for an Azure VM.

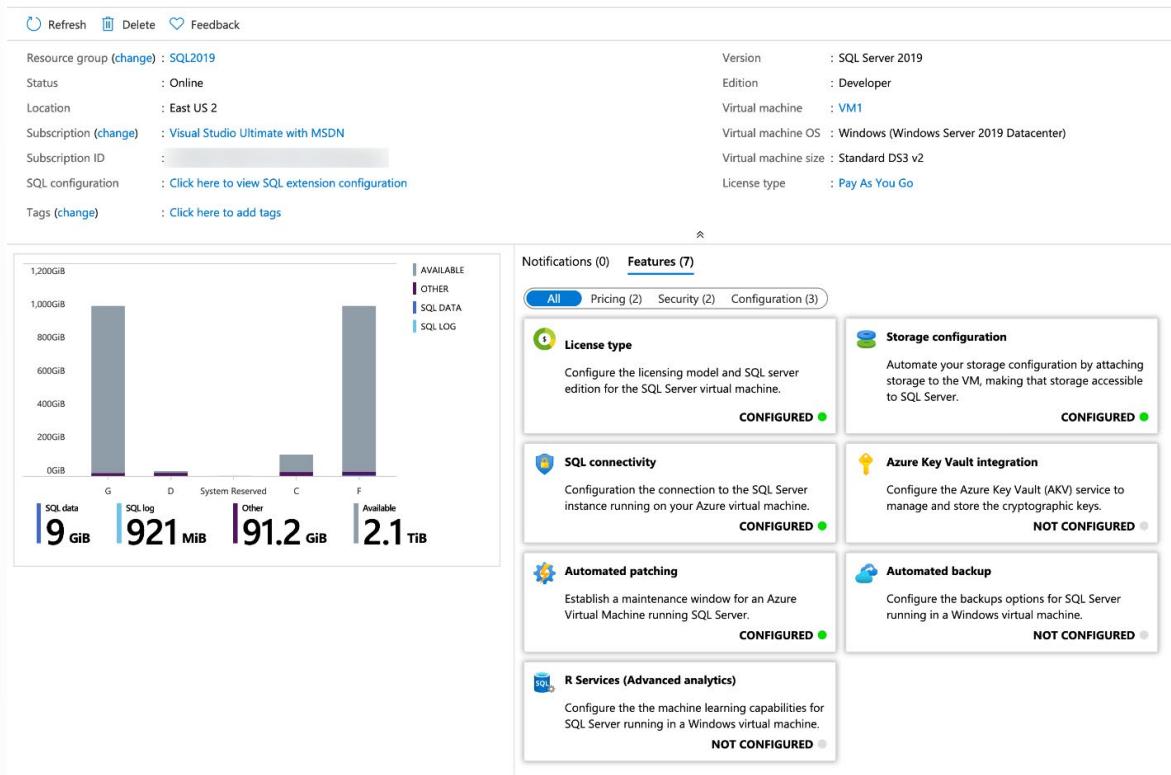
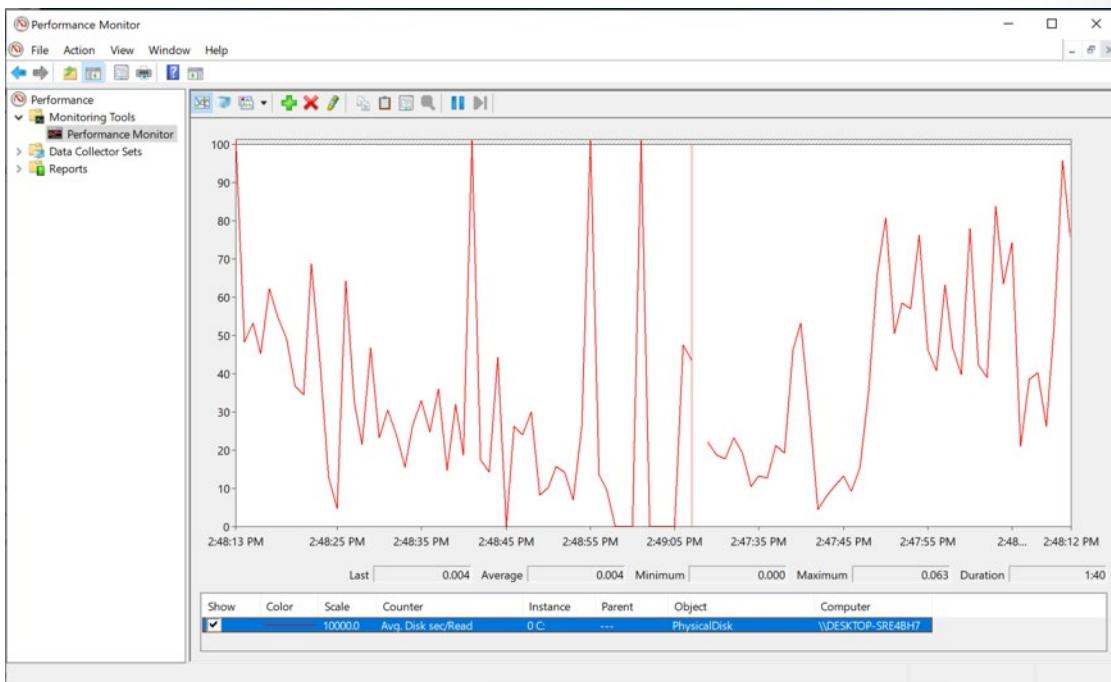


Figure 2 SQL VM Resource Provider screen in the Azure Portal

This dashboard allows you to see how much space your database files and transaction log file are consuming, and allows you manage the features provided by the resource provider like automated patching and storage configuration. You can manually install the SQL Resource Provider for other installations of SQL Server on Azure VMs that were not defined as part of the VM.

## Perfmon with Azure SQL Virtual Machine

Whether you are using an on-premises server or on an Azure virtual machine, the Windows Server platform has a native tool called Performance Monitor (commonly shortened to perfmon after the name of its executable file) that allows you to easily and routinely monitor performance metrics. Perfmon operates with counters for both the operating systems and installed programs. When SQL Server is installed on the operating system the database engine creates its own group of specific counters.



*Figure 3 Performance Monitor in Windows Server*

Figure 3 shows the reporting interface of Performance Monitor, with a single counter being collected. This screen is reached from launching Performance Monitor in Windows and shows a live tracker of a specific performance counter. In many cases you will capture multiple counters in the same session. Perfmon data can be stored locally and analyzed, but in larger environments you can forward performance monitor results into Azure Monitor, where you can have a single view across many servers.

## Critical Metrics

You have seen about how to collect data in both Azure Monitor and Windows Performance Monitor. You will now learn how to create metrics in Azure Monitor which allow you to trigger alerts or execute automated error responses.

### Review of Azure Metrics

The Azure Monitor service includes the ability to track various metrics about the overall health of a given resource. Metrics are gathered at regular intervals and are the gateway for alerting processes that will help to resolve issues quickly and efficiently. Azure Monitor Metrics is a powerful subsystem that allows you to not only analyze and visualize your performance data, but to also trigger alerts that notify administrators or automated actions that can trigger an Azure Automation runbook or a webhook. You also have the option to archive your Azure Metrics data to Azure Storage, since active data is only stored for 93 days.

### Creating Metric Alerts

Utilizing the Azure portal, you can create alert rules, based on defined metrics, in the overview section of the Azure Monitor blade. Azure Monitor Alerts can be scoped in three ways. For example, using Azure Virtual Machines as an example you can specify the scope as:

- A list of virtual machines in one Azure region within a subscription

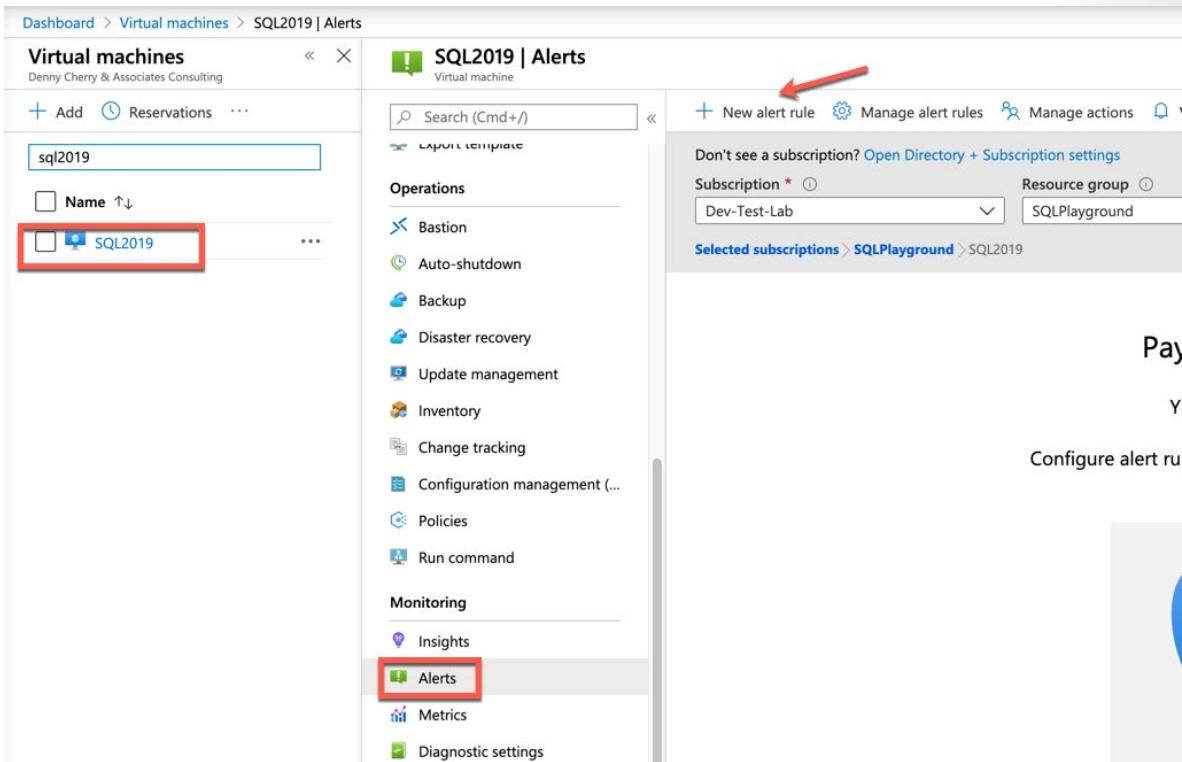
- All virtual machines (in one Azure region) in one or more resource groups in a subscription
- All virtual machines (in one Azure region) in one subscription

In this manner you can create a alert rule based on resources contained within resource groups as shown in Figure 4.

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation menu is open, with the 'Monitor' option highlighted by a red box. The main content area is titled 'Monitor | Alerts' and displays a list of monitoring categories: Overview, Activity log, Alerts (which is also highlighted by a red box), Metrics, Logs, Service Health, Workbooks, Insights, Settings, Support + Troubleshooting, and Usage and estimated costs. At the top right, there are buttons for 'New alert rule' (with a red arrow pointing to it), 'Manage alert rules', 'Manage actions', 'View classic alerts', 'Refresh', and a smiley face icon. Below these buttons, it says 'Don't see a subscription? Open Directory + Subscription settings'. It shows the 'Subscription' dropdown set to 'Dev-Test-Lab' and the 'Resource group' dropdown set to '33 selected'. A section titled 'Selected subscriptions' shows '33 selected' resource groups. To the right, there is a chart titled 'All is good! You have 1 alert' showing a pie chart and some bars. Below the chart are links: 'Manage alert rules (2)' and 'The classic alerts can be accessed here'.

Figure 4 Creating a Monitoring Alert in the Azure Portal

The example shown in Figure 5 below reflects a virtual machine named SQL2019 on which we are creating an alert that is at the scope of the individual virtual machine.



*Figure 5 Creating an Alert Rule on a Specific Resource*

Regardless the scope of the alert, the creation process is the same.

From the alerts screen, shown in Figure 5, click on New Alert Rule. If an alert is created from within the scope of a resource, the resource values should be populated for you. In Figure 5, you can see that the resource is the SQL2019 virtual machine, the subscription is Dev-Test-Lab and the resource group in which it resides is SQLPlayground.

Under the Condition section, click Add, shown in Figure 6.

The screenshot shows the 'Create rule' page in the Azure portal. At the top, the navigation path is 'Dashboard > Virtual machines > SQL2019 | Alerts > Create rule'. The main title is 'Create rule' with a subtitle 'Rules management'. The 'RESOURCE' section shows 'SQL2019' selected from a dropdown menu. To the right, the 'HIERARCHY' shows 'Dev-Test-Lab > SQLPlayground'. Below this, the 'CONDITION' section has a note: 'No condition defined, click on 'Add' to select a signal and define its logic' and a 'Select' button. The 'ACTIONS GROUPS (optional)' section includes an 'Action group name' field set to 'No action group selected', a 'Contain actions' checkbox, and 'Add' and 'Create' buttons. A tooltip banner states: 'Action rules (preview) allows you to define actions at scale as well as suppress actions. Learn more about this functionality by clicking on this banner.' In the bottom section, 'ALERT DETAILS' include an 'Alert rule name' field with a sample value 'Percentage CPU greater than 70' and a 'Description' field with a placeholder 'Specify alert description here...'. The entire interface is clean with a light blue header and white background.

Figure 6 Create Rule screen in Azure Alerts

Select the metric that you wish to alert on. Figure 7 shows Percentage CPU which you will then see selected in Figure 8.

**Configure signal logic**

Choose a signal below and configure the logic on the next screen to define the alert condition.

Signal type	Monitor service
All	All

Displaying 1 - 20 signals out of total 50 signals

Search by signal name

Signal name	Signal type	Monitor service
Percentage CPU	Metric	Platform
Network In Billable (Deprecated)	Metric	Platform
Network Out Billable (Deprecated)	Metric	Platform
Disk Read Bytes	Metric	Platform
Disk Write Bytes	Metric	Platform
Disk Read Operations/Sec	Metric	Platform
Disk Write Operations/Sec	Metric	Platform

Figure 7 Selecting a Signal for an Alert

The alerts can be configured in a static manner (e.g. alert when CPU goes over 95%) or in a dynamic fashion using Dynamic Thresholds. Dynamic Thresholds learn the historical behavior of the metric and alert when the resources are operating in an abnormal manner. This can detect seasonality in your workloads and adjust the alerting accordingly.

If Static alerts are used, you must provide a threshold for the selected metric. In this example 80 percent was provided. This means that if the CPU utilization exceeds 80 percentage over a given period, an alert will be fired and react as specified.

Both types of alerts offer Boolean operators such as greater than or less than. Along with Boolean operators, there are aggregate measurements to select from such as average, minimum, maximum, count, average, and total. With these options available, it's easy to construct a flexible alert that will suit just about any enterprise level alerting.

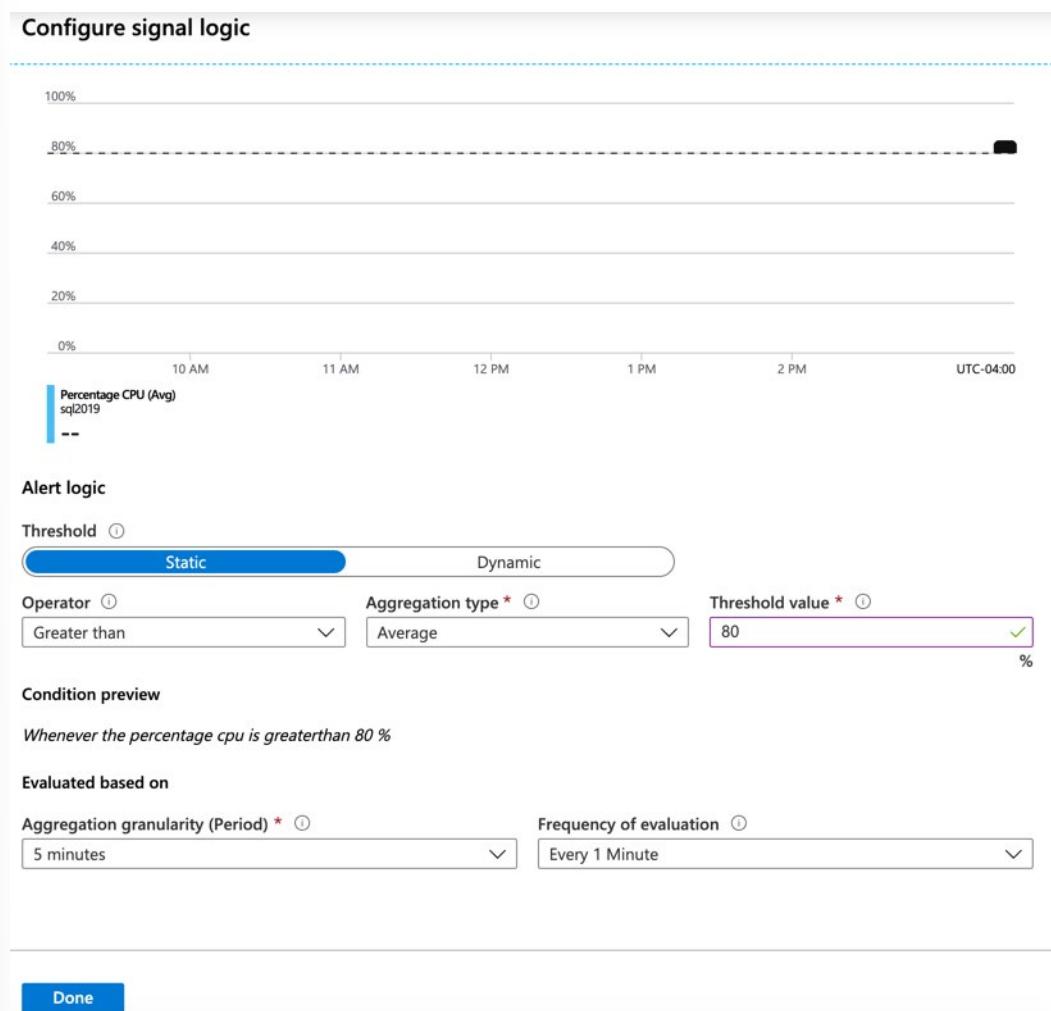


Figure 8 Configuring Logic Alert for a High CPU Rule

After creating the alert, in order to notify administrators or launch an automation process, an action group needs to be configured. You should note that defining an action group is optional, and if one is not configured the alert will just log the notification to storage with no further action is taken. You can create a new action group from the metrics screen shown in Figure 6, by clicking Add next to Action Groups. You will then see the dialog in Figure 9.



Figure 9 Create an action group

Once you click Create Action group, you will see the screen shown in Figure 10. You will name the action group and define an alert and the response. In this example, the administrator is going to be emailed in the event of the alert's condition being triggered.

**Add action group**

Action group name *	①	CPU Alert for SQL1	✓
Short name *	①	CPUSQL1	✓
Subscription *	①	Contoso Ltd	▼
Resource group *	①	Default-ActivityLogAlerts (to be created)	▼
<b>Actions</b>			
Action name *	Action Type *	Status	Configure
EmailOperator	Email/SMS message/P...	✓	Edit details X
Unique name for the action	Select an action type	▼	

[Azure Privacy Statement](#)  
[Azure Alerts Pricing](#)

**Info** Have a consistent format in emails, notifications and other endpoints irrespective of monitoring source. You can enable per action by editing details. Click on the banner to learn more ↗

*Figure 10 Configuring an alert*

You can configure the email or SMS details as can be seen in Figure 11. You can reach this screen either by clicking Details under Configure in Figure 10, or by adding a new action which will also bring up the configuration screen.

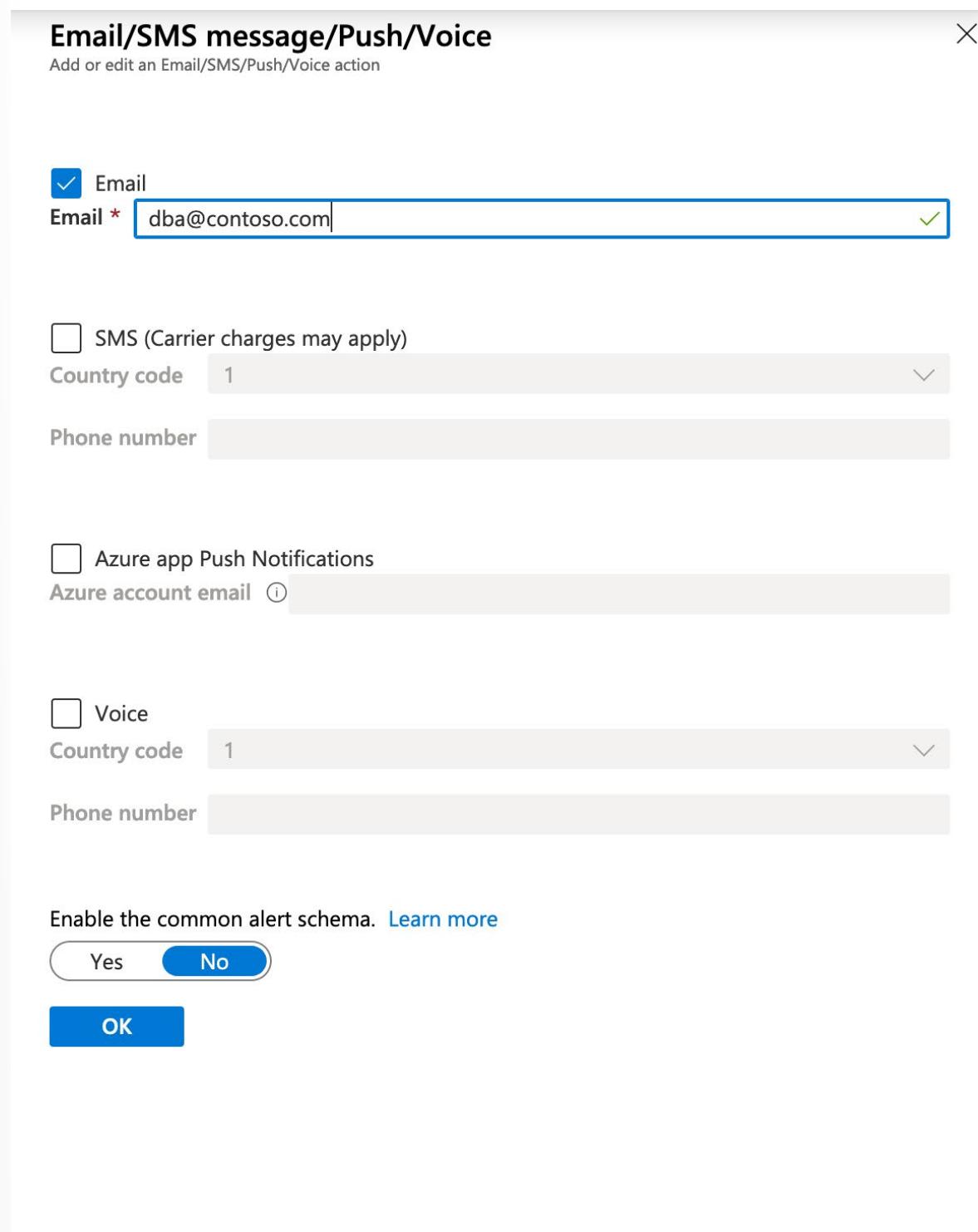


Figure 11: Configuring email/SMS

With an action group, there are several ways in which you can respond to the alert. The following options are available for defining the action to take:

- Automation Runbook
- Azure Function

- Email Azure Resource Manager Role
- Email/SMS/Push/Voice
- ITSM
- Azure Logic App
- Secure Webhook
- Webhook

There are two categories to these actions—notification which means notifying an administrator or group of administrators of an event, and automation, which is taking a defined action to respond to a performance condition.

## Reviewing Older Performance Data

One of the benefits of utilizing the Azure Monitor is the ability to easily and quickly review past metrics that were gathered. If you examine a resource, you'll note a datetime picker in the upper right-hand corner, as seen in Figure 12. Azure Monitor Metrics will be retained for 93 days, after which they are purged, however you do have the option to archive them to Azure Storage.

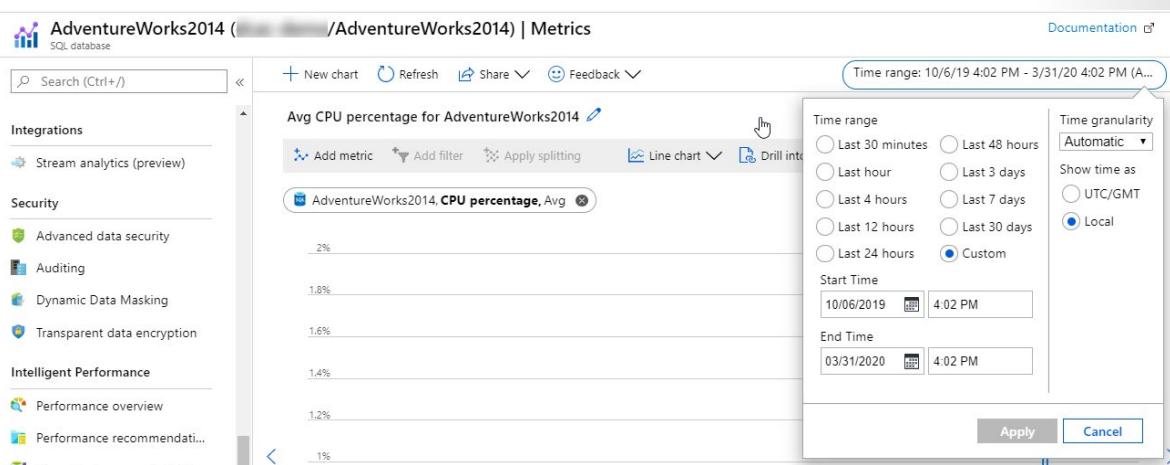


Figure 12 Metrics view for an Azure SQL Database

You are also able to select a smaller window of time such as the last 30 minutes, last hour, last 4 hours, or last 12 hours as an example. The flexibility of Azure monitor allows administrators to quickly identify issues as well as to potentially diagnose past issues.

## SQL Server Metrics that Matter

Microsoft SQL Server is an extremely well instrumented piece of software that collects a great deal of performance metadata. The database engine has metrics that can be monitored to help identify and improve performance related issues. Some operating system metrics are only viewable from within performance monitor while others can be accessed through T-SQL queries, in particular, by selecting from the dynamic management views (DMVs). There are some metrics that are exposed in both locations so knowing where to identify specific metrics is important. One example of data that can only be captured from DMVs is data and transaction log file read/write latency as exposed in sys.dm\_os\_volume\_stats. On the other hand, an example of an OS metric that is not available directly through SQL Server is the seconds per disk read and write for the disk volume. Combining these two metrics can help you gain better understand if a performance issue is related to database structure or a physical storage bottleneck.

## Establishing a Baseline

With any type of application workload, it is imperative to establish a working baseline. A baseline will help you identify if an ongoing issue should be considered within normal parameters or has exceeded given thresholds. Without a baseline, every issue encountered could be considered normal and therefore not require any additional intervention.

### *Correlating SQL Server and Operating System Performance*

When deploying SQL Server on an Azure virtual machine, it's critical to correlate the performance of SQL Server with that of the underlying operating system. Note that if you are using Linux as the operating system, you will need to install InfluxDB, Collectd, and Grafana to capture data similar to Windows Performance Monitor. These services collect data from SQL Server and provide a graphical interface to review the data. Utilizing these tools on Linux or Performance Monitor on Windows can be used in conjunction looking at SQL Server specific data such as SQL Server wait statistics, which will be described below and discussed in more detail in Module 5. Using these tools together will allow you to identify bottlenecks in hardware or code. The following Performance Monitor counters are a sampling of useful Windows metrics, and can allow you to capture a good baseline for a SQL Server workload:

**Processor(\_Total)% Processor Time**—This counter measures the CPU utilization of all of the processors on the server. It is a good indication of the overall workload, and when used in conjunction with other counters, can identify problems with query performance

**Paging File(\_Total)% Usage**—In a properly configured SQL Server, memory should not page to the paging file on disk. However, in some configurations you may have other services running that consume system memory and lead to the operating system paging memory to disk resulting in performance degradation.

**PhysicalDisk(\_Total)\Avg. Disk sec/Read and Avg. Disk sec/Write**—This is a good metric for how the storage subsystem is working. Your latency values in most cases should not be above 20ms, and with Premium Storage you should see values less than 10ms.

**System\Processor Queue Length**—This number indicates the number of threads that are waiting for the time on the processor. If it is greater than zero, it indicates CPU pressure, indicating your workload could benefit from more CPUs.

**SQLServer:Buffer Manager\Page life expectancy**—Page life expectancy indicates how long SQL Server expects a page to live in memory. There is no proper value for this setting. Older documentation refers to 300 seconds as proper, but that was written in a 32-bit era when servers had far less RAM. You should monitor this value over time, and evaluate sudden drops. This could indicate poor query patterns, external memory pressure (for example, the server running a large SSIS package) or could just be normal system processing like running a consistency check on a large database.

**SQLServer:SQL Statistics\Batch Requests/sec**—This counter is good for evaluating how consistently busy a SQL Server is over time. Once again there is no good or bad value, but you can use this in conjunction with % Processor time to better understand your workload and baselines.

**SQLServer:SQL Statistics\SQL Compilations/sec and SQL Re-Compilations/sec**—These counters will be updated when SQL Server has to compile or recompile an execution plan for a query because there is no existing plan in the plan cache, or because a plan was invalidated because of a change. This can indicate T-SQL with recompile query hints, or be indicative of memory pressure on the plan cache caused by either a lot of ad-hoc queries or simple memory pressure.

These are just a sample of the available performance monitor counters that are available to you. While the above counters provide a good baseline of performance, you may need to examine more counters to identify specific performance problems.

## Wait Statistics

When a thread is being executed and is forced to wait on an unavailable resource, SQL Server keeps track of these metrics. This information is easily identifiable via the dynamic management view (DMV) `sys.dm_os_wait_stats`. This information is important to understanding the baseline performance of your database, and can help you identify specific performance issues both with query execution and hardware limitations. Identifying the appropriate wait type and corresponding resolution will be critical for resolving performance issues. Wait statistics are available across the Azure SQL platform. You will learn more details about wait statistics in Module 5.

## Azure SQL Database Intelligent Insights

One of the benefits of using Azure SQL Database is that the baseline performance collection that is built into the Azure platform. Beyond the simple Azure Monitor data collection, Azure SQL Database Intelligent Insights is a component of Azure SQL Database that allows you to analyze performance of your queries. This feature is built using data from the Query Store which is enabled in your Azure SQL Database at creation time. The Query Store automatically gathers query performance metrics including runtime statistics and execution plan history. It also retains this set of metrics over time (the duration is dependent on the storage option you choose) which allows you to investigate performance issues that you may have encountered in the past. Azure SQL Database Intelligent Insights uses Query Store data that was exposed to the Azure portal for viewing as shown in Figure 13. You can access this dashboard by clicking Query Performance Insight in the Intelligent Performance section of the main blade of your Azure SQL Database. Query Store will be discussed in detail later in this module.

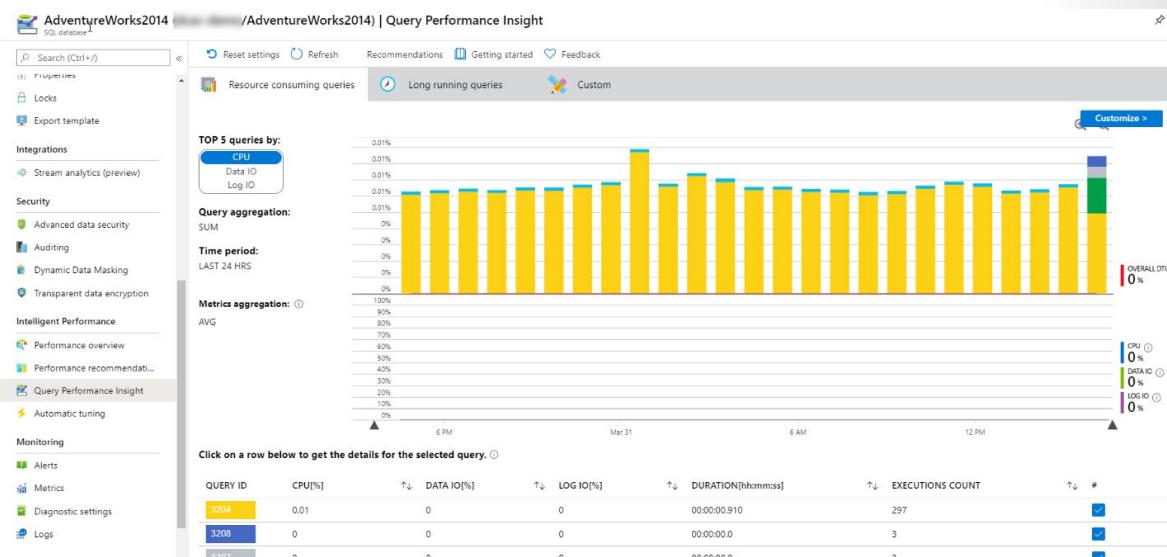


Figure 13 The Azure Portal Query Performance Insights

In order to enable Intelligent Insights, you need to add diagnostic settings to your database as shown in Figure 14.

The screenshot shows the 'Diagnostics settings' configuration page in the Azure portal. At the top, there are navigation links: Home > | Diagnostic settings > Diagnostics settings. Below the navigation is a toolbar with Save, Discard, Delete, and Provide feedback buttons. A note states: 'A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur.' It includes a link to 'Learn more about the different log categories and contents of those logs'. A 'Diagnostic settings name' input field is present. The main area is divided into 'Category details' and 'Destination details'. Under 'Category details', there are two sections: 'log' and 'metric'. The 'log' section contains checkboxes for: SQLInsights, AutomaticTuning, QueryStoreRuntimeStatistics, QueryStoreWaitStatistics, Errors, DatabaseWaitStatistics, Timeouts, Blocks, and Deadlocks. The 'metric' section contains checkboxes for: Basic, InstanceAndAppAdvanced, and WorkloadManagement. Under 'Destination details', there is a checked checkbox for 'Send to Log Analytics'. To the right, there are dropdown menus for 'Subscription' (Visual Studio Ultimate with MSDN) and 'Log Analytics workspace' (DefaultWorkspace-424d0f78-5980-4d31-98ec-...). There are also unchecked checkboxes for 'Archive to a storage account' and 'Stream to an event hub'.

Figure 14 Azure SQL Database Diagnostic Configuration

## Intelligent Insights Storage Options

You have three options for where to store the Intelligent Insights data, and the data is stored in a different format in each one. If you choose Azure storage, your data is stored in extended events format with an XEL extension. Those files can only be viewed on the Azure SQL server where they were created. If you choose an Event Hub, the data is stored in Avro format, which is a binary JSON format used for event reporting. Finally, if you use Log Analytics as a destination, as the example in Figure 14 does, your data is stored in Log Analytics, and can be queried using the Kusto Query Language.

## SQL Insights in Log Analytics

The other benefit of storing your data in Log Analytics is the Azure SQL Analytics dashboard shown in Figure 15.



Figure 15 Azure SQL Analytics Dashboard

The dashboard shown in Figure 15 can be reached by navigating to your Log Analytics workspace and then clicking Workspace summary in the general section of the Overview blade. You can then click through to Azure SQL Analytics. Azure SQL Analytics is a cloud monitoring solution that brings together performance metrics at scale and across multiple subscriptions in a single view. In addition to visualization and data collection, it has built-in intelligence for troubleshooting activities. It also allows for custom monitoring alerts and rules that provide flexibility in being able to quickly identify issues and resolve them. It supports Azure SQL Database, both single instance and elastic pools, as well as Azure SQL Managed Instances.

## Identify problematic queries

Identifying which queries are consuming the most resources is the first step in any database performance tuning endeavor. In older versions of SQL Server, this required extensive tracing and a series of complex SQL scripts, which could make the process of data gathering quite cumbersome.

Azure SQL Database offers a tool called Query Performance Insight, that allows the administrator to quickly identify expensive queries. You can navigate to Query Performance Insight in the main blade for your Azure SQL Database in the Intelligent Performance section.

When you launch Query Performance Insight you'll discover three buttons to allow you to filter for long running queries, top resource consuming queries, or a custom filter. The default value is Resource Consuming Queries which is shown Figure 16. This tab will show you the top 5 queries sorted by the

particular resource that you select on the left. In this case, it was sorted by CPU. You also have the additional options of sorting by Data IO and Log IO metrics.

You can drill into individual queries by clicking on the row within the lower grid. Each row will be identified with a unique color that correlates to the color within the bar graph above it.

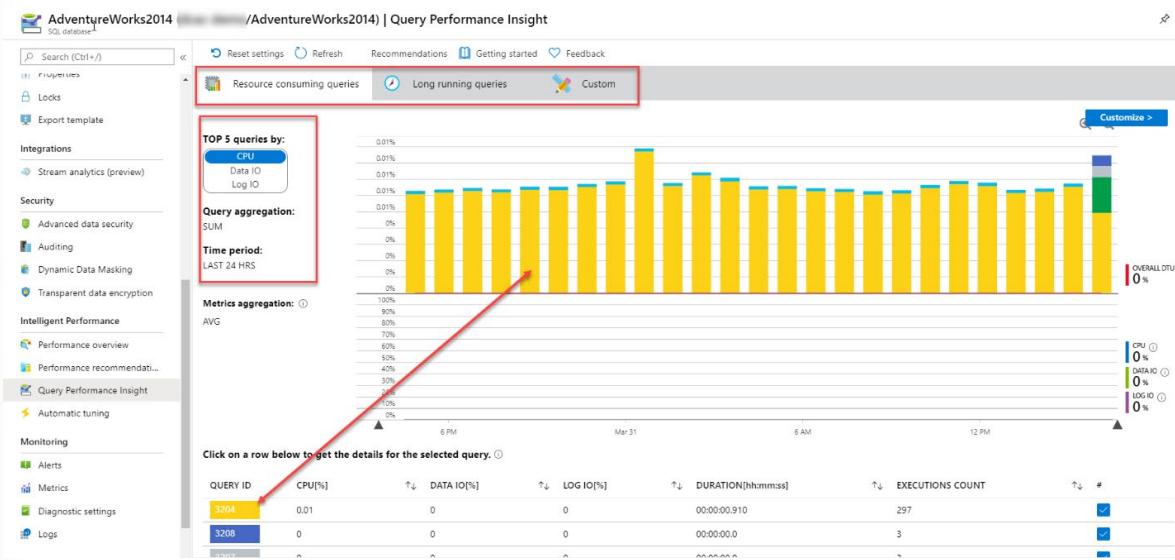


Figure 16 Query Performance Insights

Switching to Long Running Queries as shown in Figure 17, you can see a similar layout as before. In this case, the metrics are limited to the top 5 queries sorted by duration from the previous 24 hours and is a sum aggregation. As with top resource consuming queries, you can examine specific queries by clicking on the row within the grid below the graph.

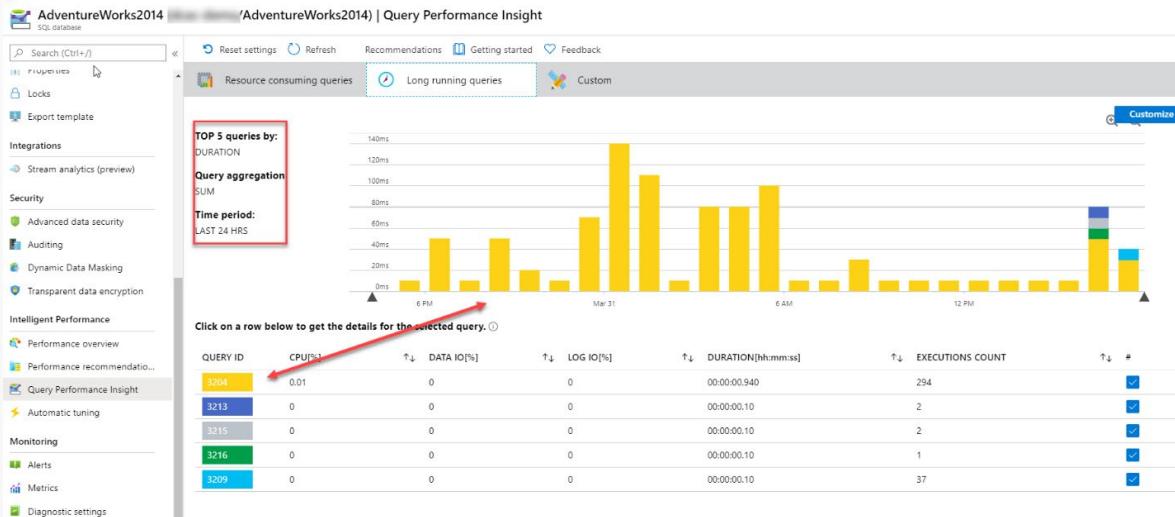


Figure 17 Performance of a Query Across Multiple Executions in Query Performance Insight

Switching to the custom tab, there is a little more flexibility compared to the other two options.

Within this tab we can further define how we wish to examine performance data. It offers us several drop-down menus that will drive the visual representation of the data, as seen in Figure 18. The key metrics are CPU, Log IO, Data IO, and memory, which are the aspects of database performance, the upper limits of which are determined by the service tier and compute resources of your Azure SQL Database.

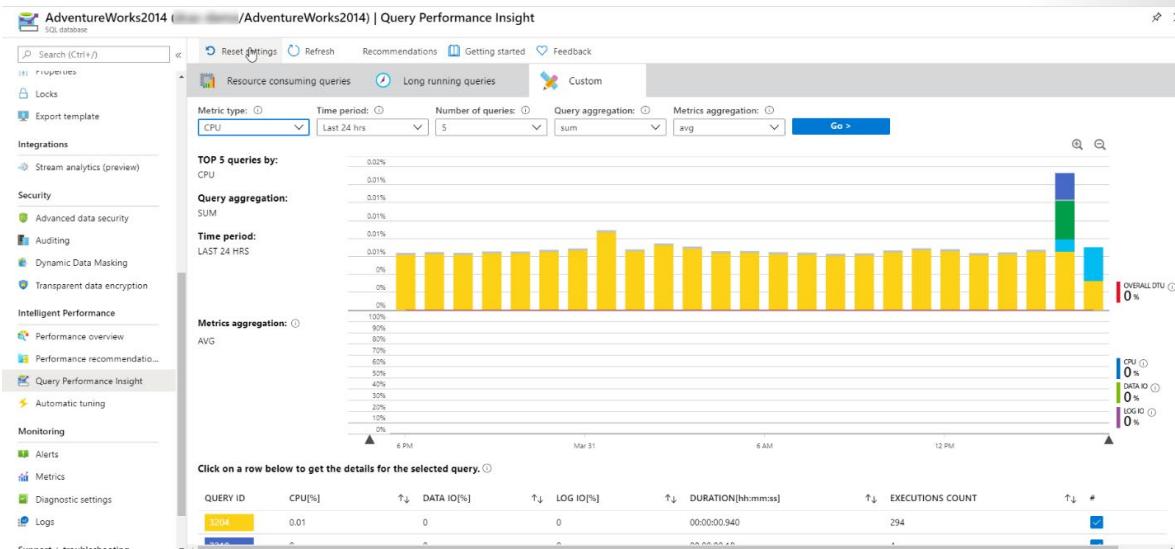


Figure 18 Custom Dashboard in Query Performance Insight

If we drill into an individual query, we will be able to see the query Id and the query itself, as well as the query aggregation type and associated time period, as seen in Figure 19. Furthermore, the query ID also correlates to the query ID located within the Query Store. Metrics gleaned from Query Performance Insights can then be easily located within the Query Store itself for deeper analysis or possibly problem resolution if needed.

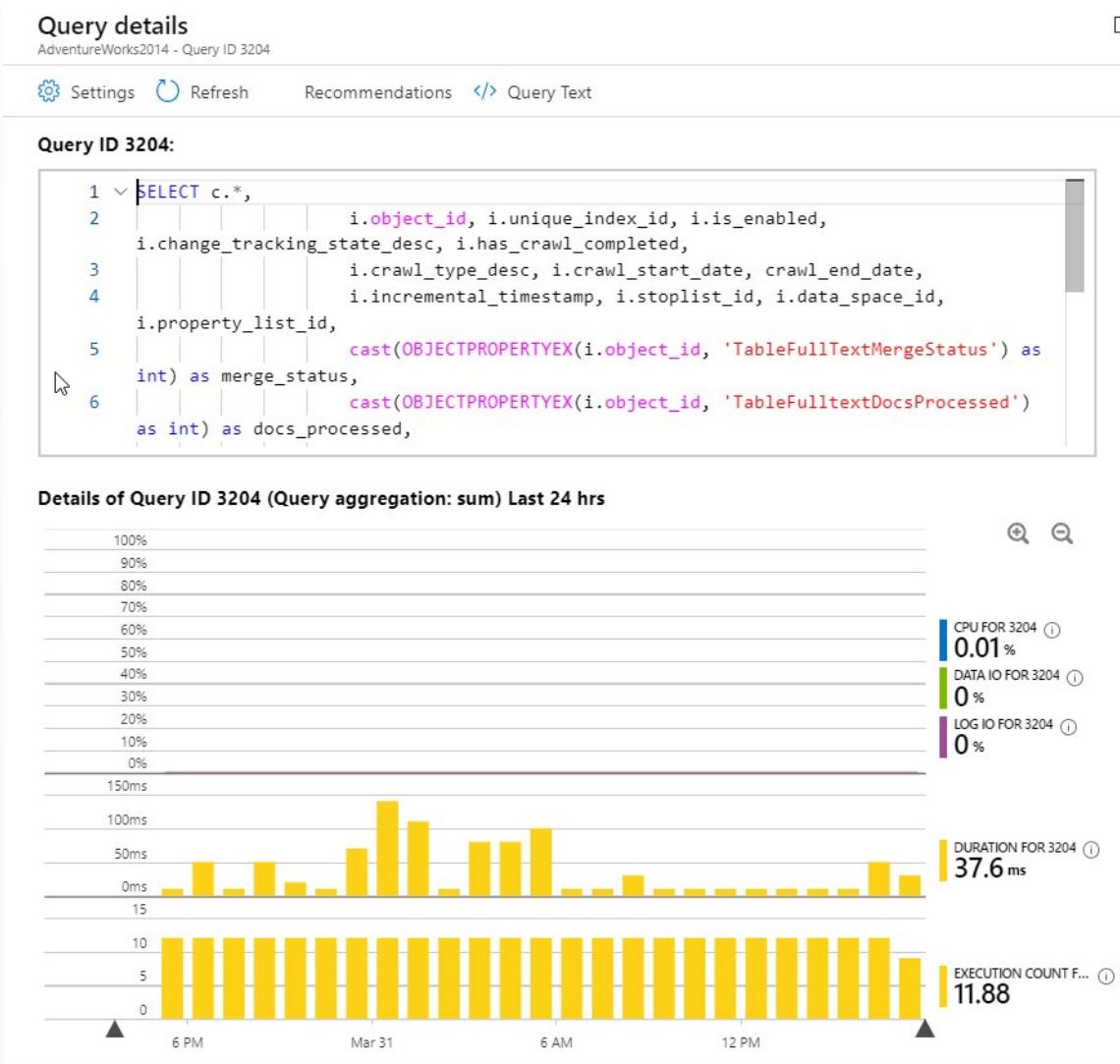


Figure 19 Details of Query ID 3204 in Query Performance Insight

While Query Performance Insight does not show the query's execution plan, you can quickly identify that query and use the information to extract the plan from the Query Store in your database.

## Summary and knowledge check

### Summary and Knowledge Check

Having a solid understand of the baseline workload of your server are database are critical to understanding performance anomalies.

The Azure platform allows you to use Azure Monitor to collect baseline performance data about both PaaS and IaaS resources. Within Windows, Performance Monitor allows you to collect detailed performance information about your SQL Server.

Azure SQL Database includes additional detailed query performance information through Query Performance Insight.

## Question 1

*Which Intelligent Insights options provides SQL Insights?*

- Log Analytics
- Azure Storage
- Event Hub

## Question 2

*Which Performance Monitor counter reflects how long SQL Server expects to retain data in memory?*

- Page Life Expectancy
- Processor Queue Length
- Paging File Usage

## Question 1

*If you want to see the sizes of your SQL Server Databases running in an Azure VM which tool should you use?*

- The SQL VM Resource Provider
- Azure Monitor
- Intelligent Insights

# Major Causes of Performance Issues

## Introduction

A common scenario that the DBA faces is when a user or group of users report problems with the performance of a report or operation. Typically, this happens well after the problem occurs, sometimes a day or two after they ran the query. In the past, it was difficult or impossible to retrieve data related to query execution, because the data was stored in memory caches and was very transient.

In order to gather this data with SQL Server you needed to have a third-party tool or build a custom monitoring solution which actively collected and maintained performance information. Third party or custom solutions are no longer necessary starting with SQL Server 2016 in which Microsoft introduced the Query Store, which acts as a data recorder for the database engine along with other additional functionality. The Query Store is focused on the query performance and changes in execution plans. In this lesson you will learn about the Query Store, as well as how to identify blocking and locking activity which can cause unpredictable query runtimes. You will also learn about disk fragmentation.

## Learning Objectives

In this lesson, you will:

- Understand the purpose and benefits of the Query Store
- Investigate the available reports and data in the Query Store
- See the impact of forcing execution plan selection
- Learn how blocking and locking work in the SQL Server database engine
- Learn about database isolation levels
- Gain understanding of database fragmentation and its impact

## Query Store

The Query Store can be thought of as a flight data recorder for SQL Server. It collects and permanently stores and aggregates performance information, that in older versions of SQL Server was either transient or not collected at all. The Query Store captures run time information such as duration, logical I/O, CPU usage amongst other metrics for query executions in a specific user database. It also captures the estimated execution plan for each execution, which can allow you to quickly detect an execution plan that has regressed in performance. The catalog views that make up the Query Store are stored in the user database, and are shown in Figure 20.

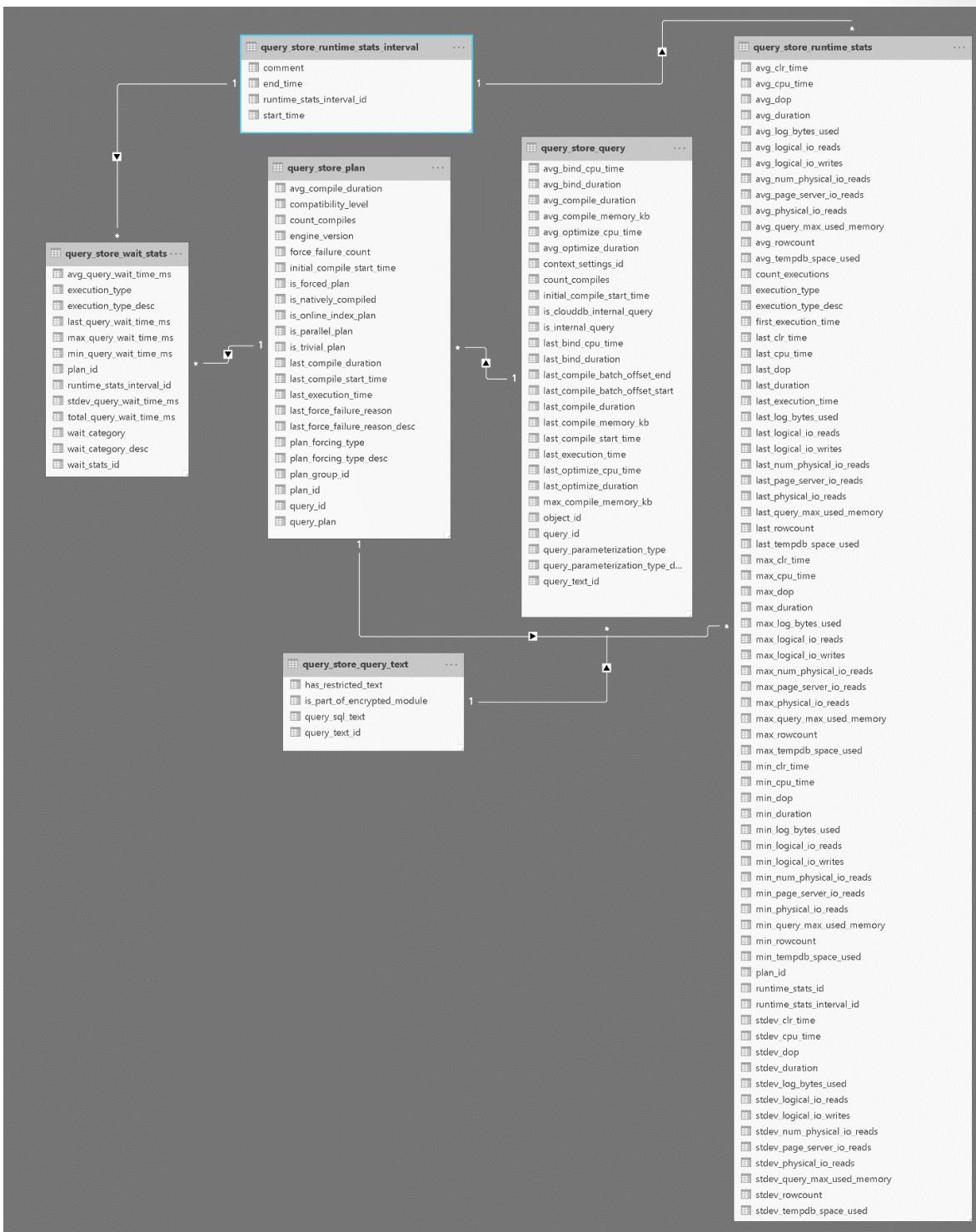


Figure 20 Relationships between Query Store Catalog Views

There are two types of data being captured. One is the data about the query itself (number of executions, the plan it used in execution, the query text) and the other is the performance information, which collects runtime statistics for each execution of the query. Each query execution has a large number of runtime metrics captured for each execution of the query. This gives the DBA a powerful tool to perform analysis of the data.

## Query Store Reports

SQL Server Management Studio offers a number of out of the box reports that allow you to quickly gather information from the Query Store. The current list of built in reports is shown in Figure 21. You can reach this menu by using SQL Server Management Studio, clicking on a user database in Object Explorer, and clicking on the Query Store folder to expand it.

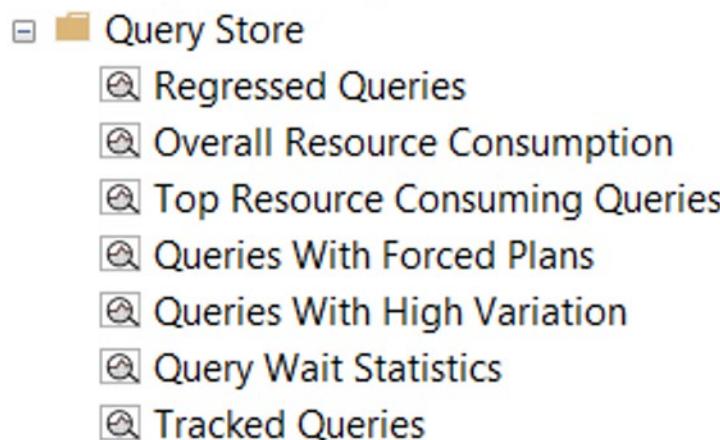


Figure 21 Query Store Reports in SSMS 18

Most of these reports can be filtered by the following execution metrics:

- CPU Milliseconds
- Duration
- Logical Reads
- Logical Writes
- Physical Reads
- CLR Time
- Degree of Parallelism
- Memory Consumption
- Row Count
- Log Memory Used
- TempDB Memory Use
- Wait Time

In addition to these metrics, you can choose a statistic to further refine your data.

- Average
- Maximum
- Minimum
- Standard Deviation
- Total

You can further filter based on a time interval, which allows you to retroactively troubleshoot a performance problem that may have happened several days ago. The built-in reports include:

**Regressed Queries**—This shows any queries where execution metrics have degraded in the period of time of interest (last hour, day, and week). This is a useful report for evaluating the impact of minor or major changes in server configuration or database schema.

**Overall Resource Consumption**—This report allows you to quickly observe the most impactful queries in your database. This report allows click through to the "Top Resource Consuming Queries" report, which allows you to gather execution plan information.

**Top Resource Consuming Queries**—This report shows the query and query plan for the most impactful queries in a database for a time period. This allows you to observe if a query has multiple execution plans, and whether or not those plans have high variability in performance.

**Queries with Forced Plans**—This report contains information about plan forcing, and any plan forcing failures (a situation where a forced execution plan was not honored).

**Queries with High Variation**—This report showcases queries that have a high degree of variance between executions and can be sorted by any of the above execution metrics.

**Query Wait Statistics**—This report allows you to see waits statistics aggregated, and drill-through to get further information on queries that spent the most time waiting. You should note this wait information is aggregated and not as detailed as what you might observe in the `sys.dm_os_wait_stats` DMV.

**Tracked Queries**—This report is filtered by `query_Id` and allows you to view the performance of a specific query and its execution plans. You can manually enter a `query_id` or you can add queries from the regressed or top resource consuming query reports. The `query_id` can be captured from the catalog view `sys.query_store_query`.

## *Performance Overhead of Query Store*

Collecting data in any system has an execution cost of CPU cycles, memory, and disk utilization that is known as observer overhead. The Query Store is designed to minimize the impact of its data collection. Data for the Query Store is written to memory for each new query, and each execution of an existing query. If this was written to disk for each execution, the performance overhead would be significant, so SQL Server uses a setting called **DATA\_FLUSH\_INTERVAL\_SECONDS** to control the frequency of flushing the Query Store data to disk. By default, the data is flushed every fifteen minutes, but this is a user configurable setting per database. This process of collecting the query and runtime information and writing it to disk is shown Figure 22.

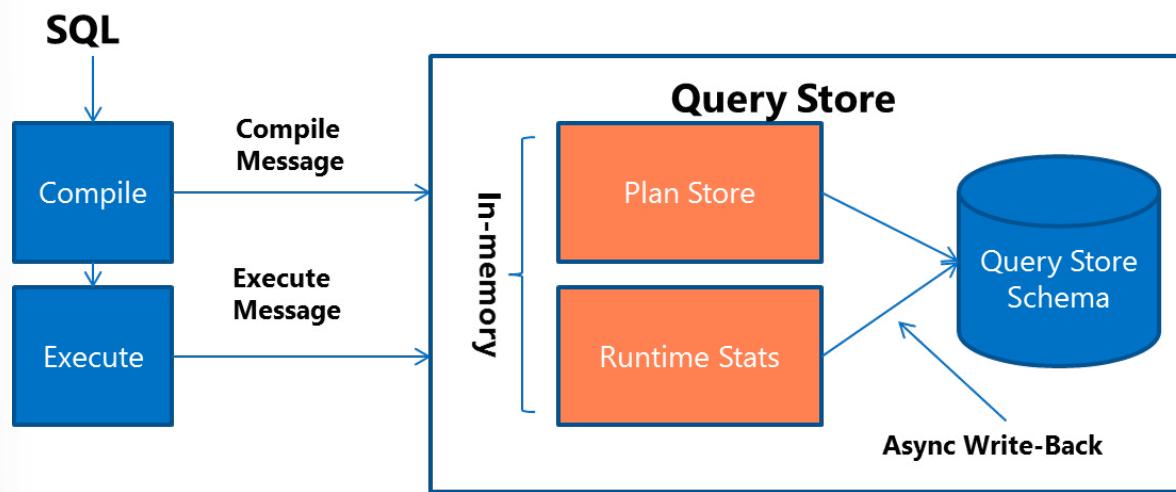


Figure 22 Process for writing data into the Query Store

The other important setting to note in the Query Store options is **Max Size (MB)**, which sets the amount of storage for the data collected. The default value is 100 MB and is commonly increased to around 1-2 GB depending on the volume of queries executed against a database. Some workloads that have a high number of unique ad-hoc queries, which is characteristic of applications written in Entity Framework, may see very high data volume. If the size of the data stored on disk exceeds the **Max Size (MB)** the Query Store will go into read-only mode until more space is added, or cleanup happens. The default value for time-based cleanup is 30 days, but the **size based cleanup mode** option will remove older queries as the Query Store approaches its max size. You will want to strike a balance between the amount of data you keep and the amount of space being consumed on disk.

The other setting you should note is **Query\_Capture\_Mode** which defaults to Auto, which means queries with insignificant compilation time and duration are ignored, along with infrequent queries. This default was changed in SQL Server 2019, and in Azure SQL. The older default was All, which captures all queries executed. There are also options of none (collect no queries) and Custom, which was also introduced in SQL Server 2019, which allows you to use metrics such as execution count, compile CPU time, and execution time to limit which queries are captured. This is particularly useful for a database where most of the queries are unique, as these unique queries can cause the Query Store to grow rapidly in size.

## Plan Forcing in the Query Store

Another benefit of the Query Store is the ability to force a given execution plan for a query. Plan forcing in the Query Store also drives the automatic tuning feature that was introduced in SQL Server 2017, which uses the last known good execution plan for a given query after a performance regression occurs. Since the query can store multiple execution plans for a given query, you can have the database engine force a known good plan for a given query. Plan forcing should be used for queries that have suddenly changed execution plans and have experienced significant regression in execution time. Plan forcing offers a quick mitigation for a performance problem, but you should always investigate what caused the performance regression and look to resolve the cause of the execution plan variability. Those fixes could be adding an index or looking to rewrite part of a query.

## Blocking and Locking in SQL Server

One feature of relational databases is locking, which is essential to maintain the atomicity, consistency, and isolation properties of the ACID model. All RDBMSs will block actions that would violate the consist-

ency and isolation of writes to a database. When programming in SQL, programmers are responsible for starting and ending transactions at the right point, in order to ensure the logical consistency of their data. In turn, the database engine provides locking mechanisms that also protect the logical consistency of the tables affected by those queries. These actions are a foundational part of the relational model.

On SQL Server, blocking occurs when one process holds a lock on a specific resource (row, page, table, database), and a second process attempts to acquire a lock with an incompatible lock type on the same resource. Typically, locks are held for a very short period, and when the process holding the lock releases it, the blocked process can then acquire the lock and complete its transaction.

SQL Server locks the smallest amount of data needed to successfully complete the transaction. This behavior allows maximum concurrency. For example, if SQL Server is locking a single row, all other rows in the table are available for other processes to use, so a lot of concurrent work can go on. However, each lock requires memory resources, so it's not cost-effective for one process to have thousands of individual locks on a single table. SQL Server tries to balance concurrency with cost. One technique used is called lock escalation. If SQL Server needs to lock more than 5000 rows on a single object in a single statement, it will escalate the multiple row locks to a single table lock.

Locking is normal behavior and happens many times during a normal day. Locking only becomes a problem when it causes blocking that is not quickly resolved. There are two types of performance issues that can be caused by blocking:

- A process holds locks on a set of resources for an extended period of time before releasing them. These locks cause other processes to block, which can degrade query performance and concurrency.
- A process gets locks on a set of resources, and never releases them. This problem requires administrator intervention to resolve.

Another blocking scenario is deadlocking, which occurs when one transaction has a lock on a resource, and another transaction has a lock on a second resource. Each transaction then attempts to take a lock on the resource which is currently locked by the other transaction. Theoretically, this scenario would lead to an infinite wait, as neither transaction could complete. However, the SQL Server engine has a mechanism for detecting these scenarios and will kill one of the transactions in order to alleviate the deadlock, based on which transaction has performed the least of amount of work that would need to be rolled back. The transaction that is killed is known as the deadlock victim. Deadlocks are recorded in the *system\_health* extended event session which is enabled by default.

It is important to understand the concept of a transaction. By default, SQL Server and Azure SQL Database are in autocommit mode, which means the changes made by the statement below would automatically be written to the database's transaction log upon completion.

```
INSERT INTO DemoTable (A) VALUES (1);
```

In order to allow developers to have more granular control over their application code, SQL Server also allows you to explicitly control your transactions. The query below would take a lock on a row in the DemoTable table that would not be released until a subsequent command to commit the transaction was added.

```
BEGIN TRANSACTION
```

```
INSERT INTO DemoTable (A) VALUES (1);
```

The proper way to write the above query is as follows:

```
BEGIN TRANSACTION
```

```
INSERT INTO DemoTable (A) VALUES (1);

COMMIT TRANSACTION
```

The COMMIT TRANSACTION command explicitly commits a record of the changes to the transaction log. The changed data will eventually make its way into the data file asynchronously. These transactions represent a unit of work to the database engine. If the developer forgets to issue the COMMIT TRANSACTION command, the transaction will stay open and the locks will not be released. This is one of the main reasons for long running transactions. They're not really long-running, but they've just not been handled properly.

The other mechanism the database engine uses to help the concurrency of the database is row versioning. When a row versioning isolation level is enabled the database engine maintains versions of each modified row in TempDB. This is typically used in mixed use workloads, in order to prevent reading queries from blocking queries that are writing to the database.

## *Isolation Levels*

SQL Server offers several isolation levels to allow you to define the level of consistency and correctness you need guaranteed for your data. Isolation levels let you find a balance between concurrency and consistency. The isolation level does not affect the locks taken to prevent data modification, a transaction will always get an exclusive lock on the data that is modifying. However, your isolation level can affect the length of time that your locks are held. Lower isolation levels increase the ability of multiple user process to access data at the same time, but increase the data consistency risks that can occur. The isolation levels in SQL Server are as follows:

**Read uncommitted**—This is the lowest isolation level available. Dirty reads are allowed, which means one transaction may see changes made by another transaction that have not yet been committed.

**Read committed**—This level allows a transaction to read data previously read, but not modified by another transaction without waiting for the first transaction to finish. This level also releases read locks as soon as the select operation is performed. This is the default SQL Server level.

**Repeatable Read**—This level keeps read and write locks that are acquired on selected data until the end of the transaction.

**Serializable**—This is the highest level of isolation where transactions are completely isolated. Read and write locks are acquired on selected data and not released until the end of the transaction.

SQL Server also includes two isolation levels that include row-versioning.

**Read Committed Snapshot**—In this level read operations take no row or page logs, and the engine presents each operation with a transactionally consistent snapshot of the data as it existed at the start of the query. This level is typically used when users are running frequent reporting queries against an OLTP database, in order to prevent the read operations from blocking the write operations.

**Snapshot**—This level provides transaction level read consistency through row versioning. This level is vulnerable to update conflicts. If a transaction running under this level reads data modified by another transaction, an update by the snapshot transaction will be terminated and roll back. This is not an issue with read committed snapshot isolation.

Isolation levels are set for each session with the T-SQL SET command, as shown:

```
SET TRANSACTION ISOLATION LEVEL

{ READ UNCOMMITTED
```

```
| READ COMMITTED  
| REPEATABLE READ  
| SNAPSHOT  
| SERIALIZABLE  
}
```

There is no way to set a global isolation level all queries running in a database, or for all queries run by a particular user. It is a session level setting.

## *Monitoring for Blocking Problems*

Identifying blocking problem can be troublesome as they can be sporadic in nature. There is a DMV called *sys.dm\_tran\_locks*, which can be joined with *sys.dm\_exec\_requests* in order to provide further information on locks that each session is holding. A better way to monitor for blocking problems is to do so on an ongoing basis using the Extended Events engine. In the additional resources session of this module, you can see of an example extended events session that will track locks held and correlate them with queries so you can monitor queries over time.

Blocking problems typically fall into two categories:

- Poor transactional design. As shown above, a transaction that has no COMMIT TRANSACTION will never end. While that is a very simple example, trying to do too much work in a single transaction or having a distributed transaction which uses a linked server connection, can lead to unpredictable performance.
- Long running transactions caused by schema design. Frequently this can be an update on a column with a missing index, or poorly designed update query.

Monitoring for locking-related performance problems allows you to quickly identify performance degradation related to locking.

## **Fragmentation in Data Files**

Fragmentation on storage occurs when a collection of data that should be stored together is broken up into many pieces that are not contiguous on disk. Historically this was problematic on systems with hard disk drives, which performed much better when executing sequential reads and writes (reading or writing a set of data in contiguous sectors on the disk). Modern solid state devices (SSDs) reduce the impact of fragmentation, at least at the operating system level, but it can still be impactful within the data files used by SQL Server.

Fragmentation occurs when indexes (both clustered and nonclustered) have pages in which the logical ordering of the index, which is based on the data value of the index key, does not match the physical ordering of the pages. You will learn more about the structure of indexes in Module 5. Fragmentation is caused by the engine modifying pages whenever insert, update or delete operations take places, and there is no longer space on the page for the new value, causing the page to split. This can degrade performance, particularly of scan operations because additional I/O is required to retrieve the data to which the index points. Fragmentation can be reduced by using the fillfactor setting when creating an index to allow free space for inserted and updated records.

## Summary and knowledge check

### Summary and Knowledge Check

The Query Store helps you quickly identify your most expensive queries, and find any changes in performance. By automating the collection of query plan and execution runtime, the Query Store provides a powerful complete data collection. SQL Server and Azure SQL provide locking and blocking in order to manage concurrency and ensure data consistency. You can adjust the isolation levels in SQL Server to help manage concurrency. Finally, fragmentation can occur in indexes over time as inserts and deletes happen.

#### Question 1

*Which isolation level should you choose if you want to prevent users reading data from blocking users writing data?*

- Serializable
- Read Committed Snapshot Isolation
- Repeatable Read

#### Question 2

*Which DMV shows sessions holding locks?*

- sys.dm\_os\_wait\_stats
- Sys.dm\_tran\_locks
- Sys.dm\_exec\_requests

#### Question 3

*Which Query Store catalog view provides the Query ID to allow for query tracking?*

- Sys.query\_store\_plan
- Sys.query\_store\_runtime\_statistics
- Sys.query\_store\_queries

# Configuring Resources for Optimal Performance

## Introduction

One of the challenges administrators face in the cloud is balancing costs and performance. Both Azure and SQL Server provide many options for configuration to meet the needs of small and large workloads. Choosing the right storage and sizing your virtual machine are critical steps in meeting the performance needs of your applications and balancing cloud costs. Proper configuration of SQL Server resources like TempDB which can easily become a performance bottleneck, and Resource Governor, which can be used to manage multi-tenant workloads, are also important for properly maintaining your server performance.

In this lesson you will:

- Understand your options for configuration of Azure Storage
- Learn how to configure TempDB data files in SQL Server
- Learn how to choose the right type of VM for SQL Server workloads
- Understand the use cases and configuration of Resource Governor in SQL Server

## Azure Storage

As mentioned earlier in the lesson, storage performance is a critical component of an I/O heavy application like a database engine. Azure offers a broad array of storage options and can even build your storage solution to meet your workload requirements.

Azure Storage is a highly scalable, secure storage platform that offers a range of solutions to meet the needs of many applications. Because the focus of this course is databases, you will learn about the aspects of blob storage that are applicable to SQL Server workloads, which are disk, file, and blob storage. You should note that all the above types of storage support encryption at rest with either a Microsoft managed or a user-defined encryption key.

**Blob Storage**—Blob storage is what is known as object-based storage and includes cold, hot, and archive storage tiers. In a SQL Server environment, blob storage will typically be used for database backups, using SQL Server’s backup to URL functionality.

**File Storage**—File storage is effectively a file share that can be mounted inside a VM, without the need to set up any hardware. SQL Server can use File storage as a storage target for a failover cluster instance.

**Disk Storage**—Azure managed disks offer block storage that is presented to a VM. These are managed just like a physical disk in an on-premises server, except that they are virtualized. There are several performance tiers within managed disks depending on your workload. This is the most commonly used storage for SQL Server data and transaction log files.

## Azure Managed Disks

Azure Managed Disks is block-level storage that is presented to Azure VMs. Block level storage refers to raw volumes of storage that are created and can be treated as an individual hard drive. These block devices can be managed within the operating system, and the storage tier is not aware of the contents of the disk. The alternative to block storage is object storage, where files and their metadata are stored on the underlying storage system. Azure Blob Storage is an example of an object storage model. While object storage works well for many modern development solutions, most workloads running in virtual machines will use block storage.

The configuration of your managed disks is important to the performance of your SQL Server workloads. If you are moving from an on-premises environment it is important to capture metrics like **average disk seconds/read** and **average disk seconds/write** from Performance Monitor as detailed earlier. Another metric to capture is the I/O Operations per Second, which can be captured using the **SQL Server:Resource Pool Stats Disk Read and Write IO/sec** counters, which show you how many IOPs SQL Server is serving at its peak. It is important to understand your workloads—you will want to design your storage and VM (this is important because each Azure VM type has an IOPs limit) to meet the needs of those workload peaks without incurring significant latency.

Azure managed disks come in four types:

**Ultra Disk**—Ultra disks support high-IO workloads for mission critical databases with extremely low latency.

**Premium SSD**—Premium SSD disks are high-throughput and low latency and can meet the needs of most database workloads running in the cloud

**Standard SSD**—Standard SSDs are designed for lightly used dev/test workloads or web servers that do a small amount of IO, and require predictable latency.

**Standard HDD**—Standard HDDs are suitable for backups and file storage that is infrequently accessed.

Typically, production SQL Server workloads will use either Ultra Disk or Premium SSD, or some combination of the two. Ultra Disks are typically used where you are looking for sub-millisecond latency in response time. Premium SSDs typically have single digit millisecond response time, but have lower costs, and more flexibility in design. Premium SSDs also support read-caching which can benefit read-heavy database workloads by reducing the number of trips to the disk. The read cache is stored on the local SSD (the D:\ drive on Windows or /dev/sdb1/ on Linux) which can help reduce the number of round trips to the actual disk.

## *Striping Disks for Maximum Throughput*

One of the ways to get more performance and volume out of Azure disks is to stripe your data across multiple disks. This does not apply to Ultra Disk, as you can scale IOPs, throughput, and maximum size independently on a single disk. However, with Premium SSDs it can be beneficial to scale both IOPs and storage volume. In order to stripe disks in Windows, you simply add the number of disks you would like to the VM, and then create a pool using Storage Spaces in Windows. You should not configure any redundancy for your pool (which would limit your performance) as the redundancy is provided by the Azure framework, which keeps three copies of all disks in synchronous replication to protect against a disk failure. When you create a pool, your pool has the sum of the IOPs and the sum of the volume of all the disks in your pool. This means if you used 10 P30 disks which are 1 TB and have 5000 IOPs per disk, you would have a 10 TB volume with 50,000 IOPs available.

## *SQL Server Storage Configuration Best Practices*

There are few recommendations for best practices for SQL Server on Azure VMs and their storage configuration:

- Create a separate volume for data and transaction log files
- Enable read caching on the data file volume
- Do not enable any caching on the log file volume
- Plan for an additional 20% of IOPs and throughput when building your storage for your VM to handle workload peaks

- Use the D: drive (the locally attached SSD) for TempDB files because TempDB is recreated upon server restart, so there is no risk of data loss
- Enable instant file initialization to reduce the impact of file-growth activities.
- Move trace file and error log directories to data disks
- For workloads requiring storage latency under 1ms, you should consider using Ultra Disk over Premium SSD.

## Azure SQL VM Resource Provider

One way to reduce the complexity of building storage for your SQL Server VM is to use the SQL Server templates in the Azure Marketplace which allow you to configure your storage as part of your deployment as shown in Figure 23. You can configure the IOPs as need and the template will perform the work of creating your storage spaces pools within Windows.

**Configure storage**

Storage optimization: General, Transactional processing (selected), Data warehousing

**Data storage**  
These disks will be attached to your virtual machine as data disks and will be stored in storage as page blobs.

Disk type	Size (GiB)	Max IOPS	Max throughput	Number of disks
1024 GiB, Premium SSD (...)	1024	5000	200	1

**Log storage**  
Transaction logs are a critical component of the database as they record all transactions and database modifications made by each transaction.

Shared drive space *	Log drive location *	Disk type *
Use a separate drive for lo...	G:\log	Premium SSD

Disk type	Size (GiB)	Max IOPS	Max throughput	Number of disks
1024 GiB, Premium SSD (...)	1024	5000	200	1

**TempDb storage**  
The tempDb system database is a global resource that is available to all users connected to the instance of SQL Server. It is used to store temporary user objects and internal objects created by the database engine.

Shared drive space *	TempDb drive location *
Use local SSD drive	D:\tempDb

Figure 23 SQL Server VM Disk Configuration

This resource provider also supports adding TempDB to the local SSD drive and creates a scheduled task to create the folder on startup.

## TempDB Configuration

SQL Server uses TempDB extensively beyond simply storing user defined temporary tables. Work tables that are used to store intermediate query results, sorting operations, and the version store for row versioning are among just a few of the uses for TempDB. Because of this utilization it is important both to place TempDB on the lowest latency storage possible, and to properly configure its data files.

Prior to SQL Server 2016, TempDB defaulted to having only one data file. This meant that there could be contention for multiple processes trying to access system pages of the TempDB database (this scenario is fully detailed in Module 5). One common solution to this contention problem was to enable trace flag 1118 which changed the way extents were allocated. Another common best-practice recommendation was to create multiple TempDB data files. Because SQL Server uses a proportional fill algorithm for databases with multiple data files, it was also important to ensure that those files were the same size and grew at the same rate. To support this many DBAs used trace flag 1117 which forced all databases with multiple data files to grow them at the same rate.

## What is Proportional Fill?

If you were inserting one gigabyte of data into a SQL Server database with two data files, you would expect the size of each of your data files to increase by roughly 512 megabytes. However, this is not necessarily the case, as SQL Server will insert data into data files in different volumes based on the size of the data file. In the above example, if your data files were both two gigabytes in size, you would expect the even distribution of data. However, if one of your data files was ten gigabytes, and the other was one gigabyte, roughly 900 MB would go into the ten gigabyte file, and 100 MB into the one gigabyte file. While this behavior occurs in any database, with the write-intensive nature of TempDB, an uneven write pattern could cause a bottleneck on the largest file, as more of the writes would happen there.

### *TempDB Configuration in SQL Server*

SQL Server 2016 changed this behavior, by simply detecting the number of CPUs available at setup, and configuring the proper number of files, up to 8, and sizing the data files evenly. Additionally, the behaviors of trace flags 1117 and 1118 are built-in to the database engine, but only for TempDB. For TempDB heavy workloads there may be benefits to increasing the number of TempDB files beyond eight, to the number of CPUs on your machine. You will learn more about the details of TempDB contention in Module 5.

## Azure VM Sizing

As discussed in Module 2, there are many size options for Azure VMs. For SQL Server workloads the main characteristics to look for are the amount of memory available, and the number of IOPs the virtual machine can perform. Note that Azure has an option of using constrained core VMs, which can be useful for the following reason. Typically Azure will use a fixed ratio of CPU cores to memory and SQL Server is licensed by the core. You may have workloads that require large amounts of memory, but not require the number of CPUs allocated to get that amount of memory. With constrained cores you have the option of not utilizing all the VMs cores while still getting the full amount of memory. You should note there is no cost difference for constrained core VMs; you are using pay as you go licensing for your VM workloads.

Most SQL Server production workloads will run on the general purpose or memory-optimized families of Azure VMs. Larger workloads requiring more memory and/or CPU resources will land in memory-optimized VMs, but many production applications can run comfortably on general purpose VMs. Another dimension to evaluate when sizing your VM is the number of IOPs the VM supports—this is the upper limit of IOPs that your VM will perform regardless of its storage configuration.

Azure supports resizing your VM. This operation does require a restart; however, restarting a VM is typically a very fast process. In some cases (this depends on what VM type you are switching to and from) you may need to deallocate your VM and then resize. This operation does extend the duration of the outage but should not take more than a few minutes.

## Managing SQL Server Environments

While some SQL Servers or Azure SQL Managed Instances only support one application's databases (this configuration is commonly seen in mission critical applications), many servers support databases for a multiple applications with differing performance requirements and different peak workload cycles. Balancing these differing requirements can be challenging to the administrator. One of the ways to balance server resources is to use Resource Governor, which was introduced to SQL Server 2008.

## Resource Governor in SQL Server

Resource governor is a feature in SQL Server and Azure SQL Managed Instance that allows you to granularly control how much CPU, physical IO, and memory resources can be used by an incoming request from an application. Resource Governor is enabled at the instance level and allows you to define how connection are treated by using a classifier function which subdivides sessions into workload group. Each workload group is configured to use a specific pool of system resources.

### *Resource Pools*

A resource pool represents physical resources available on the server. SQL Server always has two pools, default and internal, even when resource governor is not enabled. The internal pool is used by critical SQL Server functions and cannot be restricted. The default pool, and any resource pools you explicitly define, can be configured with limits on the resources it can use. You can specify the following limits for each non-internal pool:

- Min/Max CPU Percent
- Cap of CPU Percent
- Min/Max Memory Percent
- NUMA Node Affinity
- Min/Max IOPs Per Volume

You should note that changes to a pool only effect new sessions, not existing ones. This means that a change to a pool will not help you restrict the resources of a long-running process. The exception to this is external pools which are used in conjunction with SQL Server Machine Learning Services, which are external to SQL Server and can be limited by a pool change.

You should note that with the exception of min/max CPU percent, all of the other resource pool settings represent hard limits and cannot be exceeded. Min/Max CPU percentage will only apply when there is CPU contention. For example, if you have a maximum of 70%, if there is available CPU cycles the workload may use up to 100%. If there are other workloads running, the workload will be restricted to 70%.

### *Workload Group*

A workload group is a container for session requests based on their classification by the classifier function. Like resource pools there are two built-in groups, default and internal, and each workload group can only belong to one resource pool. However, a resource pool can host multiple workload groups. All connections go into the default workload group, unless they passed into another user-defined group by

the classifier function. And by default, the default workload group uses the resources assigned to the default resource pool.

## *Classifier Function*

The classifier function is run at the time a connection is established to the SQL Server instance and classifies each connection into a given workload group. If the function returns a NULL, default, or the name of the non-existent workload group the session is transferred into the default workload group. Since the classifier is run at every connection, it should be tested for efficiency. Exhibit 1 shows a sample classifier function that classifies users based on their user name.

```
CREATE FUNCTION dbo.RGClassifier()

RETURNS SYSNAME

WITH SCHEMABINDING

AS

BEGIN

DECLARE @WorkloadGroup AS SYSNAME

IF(SUSER_NAME() = 'ReportUser')

SET @WorkloadGroup = 'ReportServerGroup'

ELSE IF (SUSER_NAME() = 'PrimaryUser')

SET @WorkloadGroup = 'PrimaryServerGroup'

ELSE

SET @WorkloadGroup = 'default'

RETURN @WorkloadGroup

END
```

### *Exhibit 1 Sample Classifier Function for Resource Governor*

You can increase the complexity of the function definition shown in Example 1, but you should verify that the user login performance is minimally impacted.

## *Resource Governor Use Cases*

Resource governor is used primarily in multi-tenant scenarios where a group of databases share a single SQL Server instance, and performance needs to be kept consistent for all users of the server. You can also use Resource Governor to limit the resources used by maintenance operations like consistency checks and index rebuilds, to try to guarantee sufficient resources for user queries during your maintenance windows.

## Summary and knowledge check

### Summary and Knowledge Check

Proper storage configuration is important to the performance of your Azure VMs. SQL Server should run on premium disk storage or ultra disk for optimal performance. You can use the Resource Provider for SQL Server to automate the creation of your storage for you SQL Server VMs on Azure. You can used Resource Governor to manage differing workloads in the same SQL Server.

#### Question 1

*Which type of storage should be used in conjunction with Azure VMs for SQL Server data files?*

- Table Storage
- Blob Storage
- Disk Storage

#### Question 2

*Which of the following can be limited using Resource Governor?*

- Buffer pool allocation
- Write IOPs
- Recompilation

#### Question 3

*Which is an option from the SQL Server Resource Provider for Azure VMs?*

- Storage configuration
- Changing Max Degree of Parallelism
- Maintenance Plans

# User Database Configuration

## Introduction

In recent versions of SQL Server Microsoft has moved more configuration options to the database level, giving you more granularity in how your databases behave. Along with those options, they have introduced features as part of the intelligent query processing that allow the query optimizer to make better choices when it generates query execution plans. Azure Database for PostgreSQL and MySQL also offer intelligent options to help you better understand the performance of your databases.

## Learning Objectives

After this lesson you will:

- Understand Database Scoped Configuration Options
- Understand the features of Intelligent Query Processing
- Know your performance monitoring options in Azure Database for PostgreSQL and MySQL

## Database Scoped Configuration Options

SQL Server has always had configuration options that were set at the database level. For example, the recovery model has always been a database setting, but as more complex features have been introduced to the database, more options have been added. Many of these options are tied to the compatibility level of the database, which is itself a database level configuration option. Database configuration options break down into two groups, with a minor difference:

- Options configured by the ALTER DATABASE SCOPED CONFIGURATION syntax in T-SQL
- Options configured by the ALTER DATABASE syntax in T-SQL

There is no significance to the different ways to set these options. Options that are set using ALTER DATABASE include:

- Database Recovery Model—Whether the database is in full or simple recovery model
- Automatic Tuning Option—Whether to enable the force last good plan as shown in the next lesson
- Auto Create and Update Statistics—Allows the database to create and update statistics and allows for the option of asynchronous statistics updates
- Query Store Options—The Query Store options described in Lesson 2 are configured here
- Snapshot Isolation—You can configure snapshot isolation and read committed snapshot isolation

The above settings are a subset of the configurable options. The full list is linked in the additional resources section of this document.

Database scoped options were introduced in SQL Server 2016, and allow configuration of several options that were formerly configured at the server level. Some of the options include:

- Maximum Degree of Parallelism—This setting allows for a database to configure its own MaxDOP setting and override the server's setting.
- Legacy Cardinality Estimation—This setting allows for the database to use the older cardinality estimator. Some queries may have degraded performance under the newer cardinality estimator, introduced in SQL Server 2014, and may benefit from this setting. You should note that if you use this

option in conjunction with a newer compatibility level, you can still get the benefits of Intelligent Query Processing in compatibility level 140 or 150.

- Last Query Plan Stats—This allows you to capture the values of the last actual execution plan for a query. This feature is only active in compatibility level 150.
- Optimize for Ad Hoc Workloads—This option uses the optimizer to store a stub query plan in the plan cache. This can help reduce the size of the plan cache for workloads that have a lot of single use queries.

There are additional settings available, and they are documented in the additional resources below.

## *Database Compatibility Level*

Each database has its own compatibility level, which controls the behavior of the query optimizer for that database. You can manage this setting when upgrading SQL Server to ensure that your queries have similar execution plans to the older version. As you learned in Module 1, Microsoft will support running on an older compatibility level for an extended period. You should attempt to move towards newer compatibility levels, as many of the new performance features in Intelligent Query Processing are only available under compatibility level 140 or 150.

# **Intelligent Query Processing**

As mentioned above, in SQL Server 2017 and 2019, and of course Azure SQL, Microsoft has introduced many new features into compatibility levels 140 and 150. Many of these features correct what were formerly anti-patterns like using user defined scalar value functions and using table variables. These features break down into a few families of features:

- Adaptive Query Processing
- Table Variable Deferred Compilation
- Batch Mode on Rowstore
- Scalar UDF Inlining
- Approximate Query Processing

## *Adaptive Query Processing*

Adaptive query processing includes a number of options that make query processing more dynamic, based on the execution context of a query. This includes several features that enhance the processing of queries.

- Adaptive Joins—the database engine defers choice of join between hash and nested loops based in the number of rows going into the join. Adaptive joins currently only work in batch execution mode.
- Interleaved Execution—Currently this feature supports multi-statement table valued functions (MST-VF). MSTVFs prior to SQL Server 2017, carried a fixed row estimate of 1 or 100 rows (dependent on version SQL Server), which could lead to suboptimal query plans if they had a much larger row count than that. With interleaved execution an actual row count is gathered from the MSTVF before the rest of the plan is generated.
- Memory Grant Feedback—SQL Server generates a memory grant in the initial plan of the query, based on row count estimates from statistics. In cases of data skew, this can lead to either over- or under-estimates of row counts, which can cause over-grants of memory that decrease concurrency, or under-grants which can cause the query to spill data to TempDB. With Memory Grant Feedback, SQL

Server detects these conditions and decreases or increases the amount of memory granted to the query to either avoid the spill or overallocation.

These features are all automatically enabled under compatibility mode 150 and requires no other changes to enable.

## *Table Variable Deferred Compilation*

Like MSTVs, table variables in SQL Server execution plans carried a fixed row count of one row. Much like MSTVs, this led to poor performance when the variable had a much larger row count than expected. With SQL Server 2019, table variables are now analyzed and have an actual row count. Deferred compilation is similar in nature to interleaved execution for MSTVs, except that it is performed at the first compilation of the query rather than dynamically within the execution plan.

## *Batch Mode on Row Store*

Batch execution mode was introduced to SQL Server 2012 in conjunction with columnstore indexes. Batch execution mode allows data to be processed in batches instead of row by row. Queries that incur significant CPU costs for calculations and aggregations will see the largest benefit from this processing model. By separating batch processing and columnstore indexes, more workloads can benefit from batch mode processing.

## *Scalar User-Defined Function Inlining*

In past releases of SQL Server scalar functions performed poorly for several reasons. Scalar functions were executed iteratively (effectively one per row), did not have proper costing in an execution plan, and did not allow parallelism in a query plan. With UDF inlining, these functions are transformed into scalar subqueries in place of the UDF operator in the execution plan. This can create very significant gains in performance for queries that involve scalar function calls.

## *Approximate Count Distinct*

A common data warehouse query pattern is to execute a distinct count of orders or users. This query pattern is can be very expensive against a large table. Approximate count distinct introduces a much faster approach to gathering a distinct count by grouping rows. This function guarantees a 2% error rate with a 97% confidence interval.

# **Query Store in Azure Database for MySQL and PostgreSQL**

One of the value-added features in Azure Database for MySQL and PostgreSQL is an implementation of the Query Store in each database. While a slightly different implementation of the feature than in Azure SQL, the Query Store offers insightful actionable performance information to the DBA.

## *Query Store in Azure Database for MySQL*

The Query Store in MySQL allows you to track query performance over time and quickly identify the longest running and most expensive queries in your databases. The data for all users, databases and queries is stored in the **mysql** schema database in your instance. Like Azure SQL Database, the Azure

Portal includes a Query Performance Insight dashboard which highlights expensive queries and wait statistics for your instance, as shown in Figure 24.

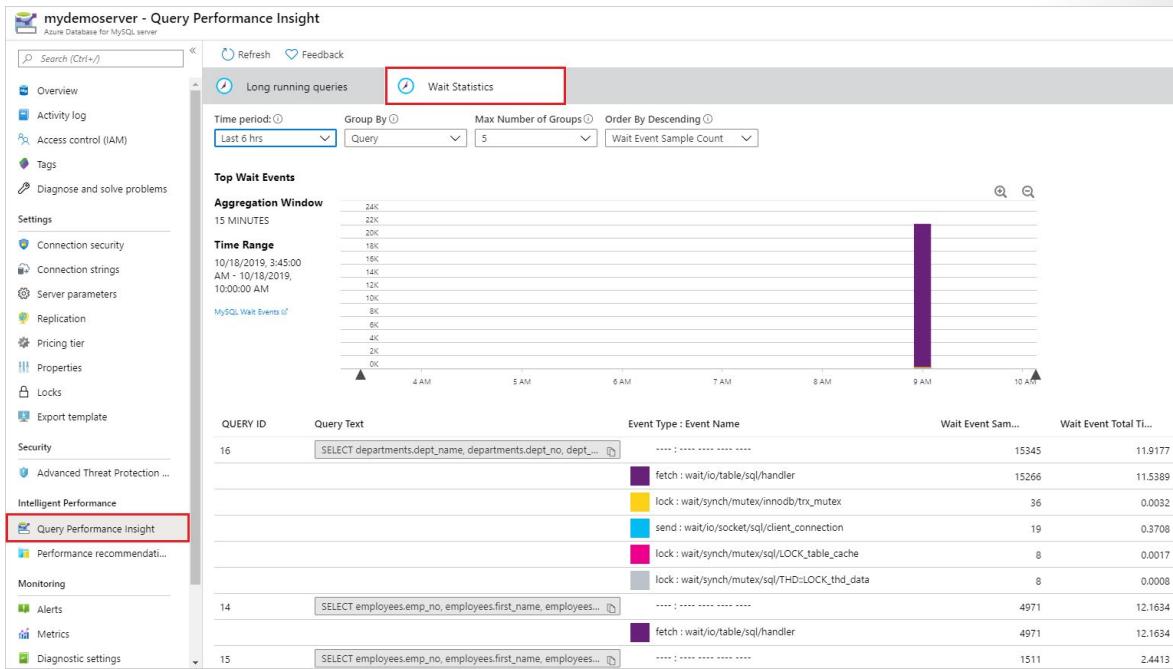


Figure 24 Query Performance Insight Wait Statistics View for MySQL

The Azure Portal also includes a performance recommendations section based on the data in the Query Store.

## Query Store in Azure Database for PostgreSQL

In a similar implementation to Azure Database for MySQL, PostgreSQL stores its Query Store information in a database named **azure\_sys** in your instance. You can also capture wait statistics. The feature is not enabled by default and should be enabled by the DBA in the server parameter settings for your instance. Similar to the Query Store in Azure SQL, you can change the capture mode from all queries, to top queries, or no queries. You can also adjust the data retention period and frequency of wait stats sampling. Just like Azure SQL and Azure Database for MySQL, when you enable the Query Store you can access Query Performance Insights and performance recommendations in the Azure Portal.

## Summary and knowledge check

### Summary and Knowledge Check

SQL Server and Azure SQL have added many features in recent releases to offer improved performance. Many of these features can be enabled at the individual database level, and may also be controlled using the compatibility mode of the database. Azure Database for MariaDB/MySQL/PostgreSQL includes the Query Store which helps you identify problematic queries.

## Question 1

*Which intelligent query processing feature allows for faster calculations of a large number of rows?*

- Batch Mode on Row Store
- Approximate Count Distinct
- Interleaved Execution

## Question 2

*Which component of resource governor allows you to configure limits on system resources?*

- Workload Groups
- Classifier Functions
- Resource Pools

## Question 3

*Which database setting affects the way the query optimizer generates execution plans?*

- Recovery Model
- Optimize for Ad-Hoc Workloads
- Compatibility Level

# Performance-related maintenance tasks

## Introduction

Even when your database is in the cloud, whether you are using a PaaS or an IaaS solution, on-going performance related maintenance tasks are critical to the overall success of your applications. Whether it is a SQL Server instance running in a virtual machine or Azure SQL Database, you need to ensure that your statistics are current, and your indexes are well organized.

## Learning Objectives

After this lesson you will:

- Understand related maintenance tasks related to indexing and statistics
- Gain insight on plan forcing
- Obtain knowledge on the auto-tuning feature in Azure

## Index maintenance tasks

Beyond just proper indexing, index maintenance is an important part of performance, especially for queries which scan tables or indexes. The query optimizer utilizes statistical information from the indexes to attempt to build the most optimal execution plan. While this execution plan is usually "good enough" having healthy indexes and statistics will ensure that any given plan will perform at optimal efficiency. Index maintenance should be performed regularly as data in your databases changes over time. You may choose to change your index maintenance strategy, based on the rate of change of your data.

### *Rebuild and Reorganize*

As discussed earlier in this document, index fragmentation occurs when logical ordering within index pages does not match the physical ordering. This can occur during routine data modification statements such as UPDATE, DELETE, and INSERT. Due to the nature of index fragmentation, this can introduce performance issues because of the additional I/O that is required to locate the data which is being referenced by the pointers within the index pages.

A reorganization of an index is an online operation that will defragment the leaf level of the index (both clustered and nonclustered). This defragmentation process will physically reorder the leaf-level pages to match the logical order of the nodes from left to right. During this process, the index pages are also compacted based on the configured fillfactor value.

A rebuild can be either online or offline depending on the command executed or the edition of SQL Server being utilized. An offline rebuild process will drop and re-create the index itself. If you can do so online, a new index will be built in parallel to the existing index. Once the new index has been built, the existing one will be dropped and then the new one will be renamed to match the old index name. Keep in mind that the online version will require additional space as the new index is built in parallel to the existing index.

The common guidance for index maintenance is:

- >5% but <30% Reorganize the index
- >30% Rebuild the index

You should use this a guidance, but not a hard and fast rule. Depending on your workload and data, you may need to be more aggressive, or in some cases you may be able to defer index maintenance for databases that mostly perform queries that seek specific pages.

Starting with SQL Server 2017, Microsoft introduced the ability to have resumable rebuild operations. This provides more flexibility in controlling how much of an impact a rebuild operation might impose on a given instance. With SQL Server 2019, the ability to control an associated maximum degree of parallelism was introduced further providing more granular control to database administrators.

## SQL Server on an Azure VM

With SQL Server being installed within an Azure virtual machine you have access to scheduling services such as the SQL Agent or the Windows Task Scheduler. These automation tools can assist in keeping the amount of fragmentation within indexes to a minimum. With larger databases, a balance between a rebuild and a reorganization of indexes must be found to ensure optimal performance. The flexibility provided by SQL Agent or Task Scheduler allows you to run custom jobs. You will learn more about these options in Module 6.

## Azure SQL Database

Due to the nature of Azure SQL Database, you do not have access to SQL Server Agent nor Windows Task Scheduler. This means that index maintenance must be controlled from outside of the database. There are three ways to accomplish this:

- Azure Automation Runbooks
- SQL Agent Job from SQL Server in an Azure VM
- Azure SQL Elastic Jobs

This will be covered in detail in Module 6.

## Azure SQL Managed Instance

As with SQL Server on an Azure VM, you can schedule jobs on an Azure SQL Managed Instance via the SQL Server Agent. This will provide flexibility to execute code specifically designed to reduce fragmentation within the indexes in the database.

## Statistics Maintenance Tasks

When doing performance tuning in Azure SQL, understanding the importance of statistics is critical. Statistics are stored in the user database as binary large objects (blobs) that contain statistical information about the distribution of data values in one or more columns of a table or indexed view. The query optimizer uses column and index statistics to understand the distribution of data in order to determine cardinality. Cardinality estimates are then used by the query optimizer to generate the execution plan. Furthermore, cardinality estimates help to determine what type of operation (e.g. Index seek or scan) is employed in order to retrieve the data requested.

## Automatic Tuning

As mentioned earlier in the module, the Query Store provides database administrators with in-depth insights on query plans and performance metrics. By default, execution plans evolve over time due to schema changes, index modifications, or changes to the data that cause updates to the statistics. This evolution can cause queries to perform poorly as the execution plan no longer meets the demands of the

given query. As mentioned earlier in the module, the Query Store in Azure SQL features a plan forcing option to always force a given execution plan.

## Auto-tuning Features

SQL Server 2017 introduced a feature called automatic tuning. Automatic tuning allows for the gathering and applying machine learning against performance metrics to provide suggested improvements or even allow for self-correction. Automatic tuning, whether on-premises or in the cloud, allows you to identify issues caused by query execution plan regression. Additionally, in Azure SQL Database you have the option to further improve query performance by index tuning. Azure SQL Database automatic tuning can identify indexes that should be added or even removed from the database to enhance query performance.

## Automatic Plan Correction

By utilizing Query Store, the database engine can identify when query execution plans have regressed in their performance. While you can manually identify a regressed plan through the user interface, this can be done automatically via the database engine.

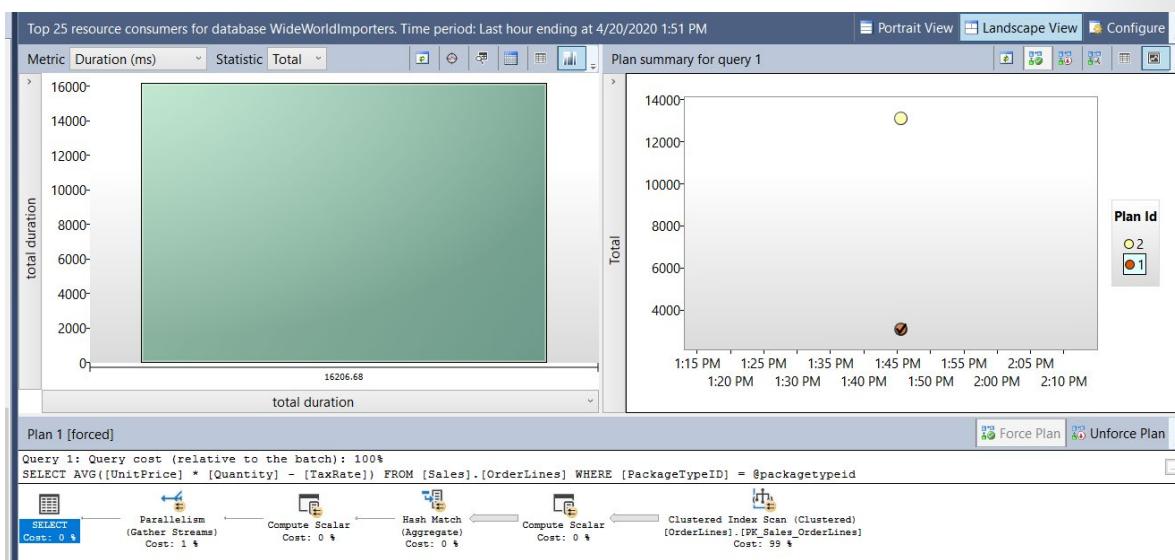


Figure 25 Query Store view of Automatic Plan Correction

In the example in Figure 25, you can see a check mark on Plan ID 1, which means that the plan has been forced. Once the feature has been enabled, the database engine will automatically force any recommended query execution plan where the number of errors in the previous plan is higher than the recommended plan, the estimated CPU gain is greater than ten seconds, or a plan was forced and continues to be better than the previous one. The plan will revert back to the last known good plan after fifteen executions of the query.

When plan forcing occurs automatically, the database engine will apply the last known good plan and will also continue to monitor query execution plan performance. If the forced plan does not perform better than the previous plan, it will be subsequently unforced and force a new plan to be compiled. If the forced plan continues to outperform the previously bad plan, it will remain forced until such time as a recompile occurs.

You can enable automatic plan correction via a T-SQL query, as shown below. Note that Query Store must be enabled and must be in Read-Write mode for the command to succeed. If either of those two criteria are not met, the ALTER statement will fail.

```
ALTER DATABASE [WideWorldImporters] SET AUTOMATIC_TUNING (FORCE_LAST_GOOD_PLAN = ON);
```

You can examine the automatic tuning recommends through a dynamic management view (DMV), `sys.dm_db_tuning_recommendations`, which is available in SQL Server 2017 or higher and is also available in Azure SQL Database solutions. This DMV provides information such as reasons as to why the recommendation was provided, the type of recommendation, the state of the recommendation, as well as others. To confirm that automatic tuning is enabled for a database, there is a DMV `sys.database_automatic_tuning_options` that can be queried.

## Automatic Index Management

As mentioned earlier Azure SQL Database can perform automatic index tuning. Over time, the database will learn about existing workloads and provide recommendations on adding or removing indexes in order to provide better performance. Like forcing improved query plans, the database can be configured to allow for automatic index creation or removal depending on existing index performance, as shown below in Figure 26.

The screenshot shows the 'Auto-tuning Options' section of the Azure portal. It includes a general information banner, inheritance settings, a note about inheritance, configuration options, and a detailed table of tuning options.

Azure SQL Database built-in intelligence automatically tunes your databases to optimize performance. Click here to learn more about automatic tuning.

Inherit from: [Server](#) Azure defaults Don't inherit

**Note:** The database is inheriting automatic tuning configuration from the server. You can set the configuration to be inherited by going to: [Server tuning settings](#)

Configure the automatic tuning options [Edit](#)

Option	Desired state	Current state
FORCE PLAN	<input type="radio"/> ON <input type="radio"/> OFF <input checked="" type="radio"/> INHERIT	<b>ON</b> Inherited from server
CREATE INDEX	<input type="radio"/> ON <input type="radio"/> OFF <input checked="" type="radio"/> INHERIT	<b>OFF</b> Inherited from server
DROP INDEX	<input type="radio"/> ON <input type="radio"/> OFF <input checked="" type="radio"/> INHERIT	<b>OFF</b> Inherited from server

Figure 26 Auto-tuning Options

When enabled, the Performance Recommendations blade will identify indexes that can be created or dropped depending on query performance. Remember this feature is not available for on-premises databases and only available for Azure SQL Database.

Creating new indexes can consume resources and timing of the index creations is critical to ensure a no negative impact is felt on your workloads. Azure SQL Database will monitor windows of time to implement new indexes to avoid causing performance issues. If resources are needed for existing workloads and potentially not available to create an index, the tuning action is postponed until such time as it has resources available. If a newly created index does not yield an increase in query performance, it will be dropped quickly. This monitoring process will validate that any actions taken only helps performance and not degrade it. If an index is dropped and query performance noticeably degrades, the recently dropped index will be re-created automatically.

## Summary and knowledge check

### Summary and Knowledge Check

Maintaining indexes and statistics can help provide consistent performance for your queries. SQL Server and Azure SQL Databases automatically update statistics based on the modification of a percentage of rows using a sliding scale. Azure SQL Database provides an additional layer of automation by automating the creation of new indexes and the removal of unused indexes.

#### Question 1

*Which platform supports automatic index management?*

- Azure SQL Managed Instance
- Azure SQL Database
- SQL Server in an Azure VM

#### Question 2

*Which of the following actions triggers automatic plan choice activity to occur?*

- An execution plan recompilation
- An execution plan change that has caused an estimated CPU gain of 10 seconds
- A new index is created

#### Question 3

*Which DVM shows the status of a plan updated by automatic tuning?*

- sys.dm\_db\_tuning\_recommendations
- sys.dm\_db\_automatic\_tuning\_options
- sys.dm\_exec\_query\_plan\_stats

## Module Summary

### Module Summary

Database Performance is often the bottleneck for many applications, and the more information the administrator has the better he or she can isolate the root cause of performance degradation. Having a baseline of performance data is a critical component of that—it is hard to identify anomalous performance if you do not understand what your ‘normal’ performance is. Azure and the Query Store provide you with many options and a great deal of data to help you understand why performance has changed. It is helpful to understand blocking and locking in your database as blocking problems can lead to very unpredictable performance. How you configure your VMs and their storage is also a fundamental element of how your workloads perform. Finally, Azure SQL provides a number of database configuration options to meet the needs of your workloads.

# Answers

## Question 1

Which Intelligent Insights options provides SQL Insights?

- Log Analytics
- Azure Storage
- Event Hub

*Explanation*

## Question 2

Which Performance Monitor counter reflects how long SQL Server expects to retain data in memory?

- Page Life Expectancy
- Processor Queue Length
- Paging File Usage

*Explanation*

## Question 1

If you want to see the sizes of your SQL Server Databases running in an Azure VM which tool should you use?

- The SQL VM Resource Provider
- Azure Monitor
- Intelligent Insights

*Explanation*

## Question 1

Which isolation level should you choose if you want to prevent users reading data from blocking users writing data?

- Serializable
- Read Committed Snapshot Isolation
- Repeatable Read

*Explanation*

## Question 2

Which DMV shows sessions holding locks?

- sys.dm\_os\_wait\_stats
- Sys.dm\_tran\_locks
- Sys.dm\_exec\_requests

*Explanation*

**Question 3**

Which Query Store catalog view provides the Query ID to allow for query tracking?

- Sys.query\_store\_plan
- Sys.query\_store\_runtime\_statistics
- Sys.query\_store\_queries

*Explanation*

**Question 1**

Which type of storage should be used in conjunction with Azure VMs for SQL Server data files?

- Table Storage
- Blob Storage
- Disk Storage

**Question 2**

Which of the following can be limited using Resource Governor?

- Buffer pool allocation
- Write IOPs
- Recompilation

**Question 3**

Which is an option from the SQL Server Resource Provider for Azure VMs?

- Storage configuration
- Changing Max Degree of Parallelism
- Maintenance Plans

**Question 1**

Which intelligent query processing feature allows for faster calculations of a large number of rows?

- Batch Mode on Row Store
- Approximate Count Distinct
- Interleaved Execution

**Question 2**

Which component of resource governor allows you to configure limits on system resources?

- Workload Groups
- Classifier Functions
- Resource Pools

**Question 3**

Which database setting affects the way the query optimizer generates execution plans?

- Recovery Model
- Optimize for Ad-Hoc Workloads
- Compatibility Level

**Question 1**

Which platform supports automatic index management?

- Azure SQL Managed Instance
- Azure SQL Database
- SQL Server in an Azure VM

**Question 2**

Which of the following actions triggers automatic plan choice activity to occur?

- An execution plan recompilation
- An execution plan change that has caused an estimated CPU gain of 10 seconds
- A new index is created

**Question 3**

Which DVM shows the status of a plan updated by automatic tuning?

- sys.dm\_db\_tuning\_recommendations
- sys.dm\_db\_automatic\_tuning\_options
- sys.dm\_exec\_query\_plan\_stats



# Module 5 Optimize Query Performance

## Module-Introduction

### Module Introduction

#### Optimize Query Performance

Query execution plans are potentially the most important aspect of database performance. Improving bad plans is certainly an area where a small amount of effort can bring huge improvements. While hardware issues can limit query performance, improving hardware usually yields performance improvements in the 10-20% range, at most. More commonly database administrators encounter queries that are not optimized, have stale or missing statistics, have missing indexes, or poor database design choices that lead to the database engine doing more work than is necessary to return results for a given query. Improving the plans can sometimes yield performance improvements in the 100-200% range or even more, meaning that after improving a plan with better indexes or statistics, a query could run twice or three times as fast!

This module provides details on how to analyze individual query performance and determine where improvements can be made.

In this module, you will explore various query plan viewing options to identify problem areas in execution plans. You will also use Query Store (available in SQL Server 2016+ and Azure SQL Database) to extract plans that you did not directly execute. You will evaluate potential performance improvements using Dynamic Management Objects to gather query performance information. You will learn about wait statistics and how they can help you analyze performance. You will explore how changing index structures can affect performance and will look at possible changes in the queries themselves, including the use of hints.

This module covers query performance tools and techniques relevant to Azure SQL Database and Azure Managed Instance, as well as SQL Server, whether installed in an Azure VM or on premises.

## **Learning Objectives:**

After completing this module, you will be able to:

- Analyze query plans and identify problem areas
- Evaluate potential query improvements
- Review table and index design
- Determine whether query or design changes have had a positive effect

# Understanding SQL Server Query Plans

## Introduction

The most important skill you should acquire in database performance tuning is being able to read and understand query execution plans. The plans explain the behavior of the database engine as it executes queries and retrieves the results.

### Learning Objectives:

In this section, you will:

1. Generate and save execution plans
2. Compare the different types of execution plans
3. Understand how and why query plans are generated

## Types of Query Plans

It is helpful to have a basic understanding of how database optimizers work before taking a deeper dive into execution plan details. SQL Server uses what is known as cost-based query optimizer. The query optimizer calculates a cost for multiple possible plans based on the statistics it has on the columns being utilized, and the possible indexes that can be used for each operation in each query plan. Based on this information, it comes up with a total cost for each plan. Some complex queries can have thousands of possible execution plans. The optimizer does not evaluate every possible plan, but uses heuristics to determine plans that are likely to have good performance. The optimizer will then choose the lowest cost plan of those evaluated for a given query.

Because the query optimizer is cost-based, it is important that it has good inputs for decision making. This means that the statistics SQL Server uses to track the distribution of data in columns and indexes need be kept up to date, or it can cause suboptimal execution plans to be generated. SQL Server automatically updates its statistics as data changes in a table; however, more frequent updates may be needed for rapidly changing data. The engine uses a number of factors when building a plan including combability level of the database, row estimates based on statistics and available indexes.

When a user submits a query to the database engine, the following process happens:

1. The query is parsed for proper syntax and a parse tree of database objects is generated if the syntax is correct
2. The parse tree from Step 1 is taken as input to a database engine component called the Algebrizer for binding. This step validates that columns and objects in the query exist and identifies the data types which are being processed for a given query. This step outputs a query processor tree which is in the input for step 3.
3. Because query optimization is a relatively expensive process in terms of CPU consumption, the database engine caches execution plans in a special area of memory called the plan cache. If a plan for a given query already exists, that plan is retrieved from the cache. The queries whose plans are stored in cache will each have a hash value generated based on the T-SQL in the query. This value is referred to as the query\_hash. When looking for a plan in cache, the engine will generate a query\_hash for the current query and then look to see if it matches any existing queries in the plan case.

4. If the plan does not exist, the Query Optimizer then uses its cost-based optimizer to generate several execution plan options based on the statistics about the columns, tables, and indexes that are used in the query, as described above. The output of this step is a query execution plan.
5. The query is then executed using an execution plan that is pulled from the plan cache, or a new plan generated in step 4. The output of this step is the results of your query.

Let's look at an example. Consider the following query:

```
SELECT orderdate  
  
    , avg(salesAmount)  
  
FROM FactResellerSales  
  
WHERE ShipDate = '2013-07-07'  
  
GROUP BY orderdate;
```

In this example SQL Server will check for the existence of the OrderDate, ShipDate and SalesAmount columns in the table FactResellerSales. If those columns exist it will then generate a hash value for the query and examine the plan cache for a matching hash value. If there is plan for a query with a matching hash the engine will try to reuse that plan, if not it will examine the statistics is has available on the OrderDate and ShipDate columns. The WHERE clause referencing the ShipDate column is what is known as the predicate in this query. If there is a nonclustered index that includes the ShipDate column SQL Server will most likely include that in the plan, if the costs are lower than retrieving data from the clustered index. The optimizer will then choose the lowest cost plan of the available plans and execute the query.

Query plans combine a series of relational operators to retrieve the data, and also capture information about the data such as estimated row counts. Another element of the execution plan is the memory required to perform operations such as joining or sorting data, which is known as the memory grant. The memory grant is a good example of the importance of statistics. If SQL Server thinks an operator is going to return 10,000,000 rows, when it is only returning 100, a much larger amount of memory is granted to the query. This can cause a twofold problem—the query may encounter a RESOURCE\_SEMAPHORE wait, which indicates that query is waiting for SQL Server to allocate it a large amount of memory. SQL Server defaults to waiting for 25 times the cost of the query (in seconds) before executing, up to 24 hours. When the query is executed, if there is not enough memory available the query will spill to tempdb, which is much slower than operating in memory.

The execution plan also stores other metadata about the query, including, but not limited to, the database compatibility level, the degree of parallelism of the query, and the parameters that are supplied if the query is parameterized.

Query plans can be viewed either in a graphical representation or in a text-based format. The text-based options are invoked with SET commands and apply only to the current connection. Text-based plans can be viewed anywhere you can run TSQL queries.

Most DBAs prefer to look at plans graphically, because a graphical plan allows you to see the plan as a whole, including what's called the 'shape' of the plan, much more easily. There are several ways you can view and save graphical query plans. The most common tool used for this purpose is SQL Server Management Studio, but estimated plans can also be viewed in Azure Data Studio. There are also third-party tools that support viewing graphical execution plans.

There are three different types of execution plans that can be viewed.

**Estimated Execution Plan:** This is the execution plan as generated by the query optimizer. The metadata and size of query memory grant are based on estimates from the statistics as they exist in the database at the time of query compilation. To see a text-based estimated plan run the command SET SHOWPLAN\_ALL ON before running the query. When you run the query, you will see the steps of the execution plan, but the query will NOT be execute, and you will not see any results. The SET option will stay in effect until you set it OFF.

**Actual Execution Plan:** This is same plan as the estimated plan; however this plan also contains the execution context for the query, which includes the estimated and actual row counts, any execution warnings, the actual degree of parallelism (number of processors used) and elapsed and CPU times used during the execution. To see a text-based actual plan run the command SET STATISTICS PROFILE ON before running the query. The query will execute, and you get the plan and the results.

**Live Query Statistics:** This plan viewing option combines the estimated and actual plans into an animated plan that displays execution progress through the operators in the plan. It refreshes every second and shows the actual number of rows flowing through the operators. The other benefit to Live Query Statistics is that it shows the handoff from operator to operator which may be helpful in troubleshooting some performance issues. Because the type of plan is animated, it is only available as a graphical plan.

## Estimated vs Actual Plans

The topic of actual versus estimated execution plans can be confusing. As mentioned above, the difference is that the actual plan includes runtime statistics that are not captured in the estimated plan. The operators used, and order of execution will be the same as the estimated plan in nearly all cases. The other consideration is that in order to capture an actual execution plan the query has to be actually executed, and this may be extremely time consuming, or not be possible (e.g. an UPDATE statement that can only be run once). However, if you need to see query results as well as the plan, you'll need to use one of the actual plan options.

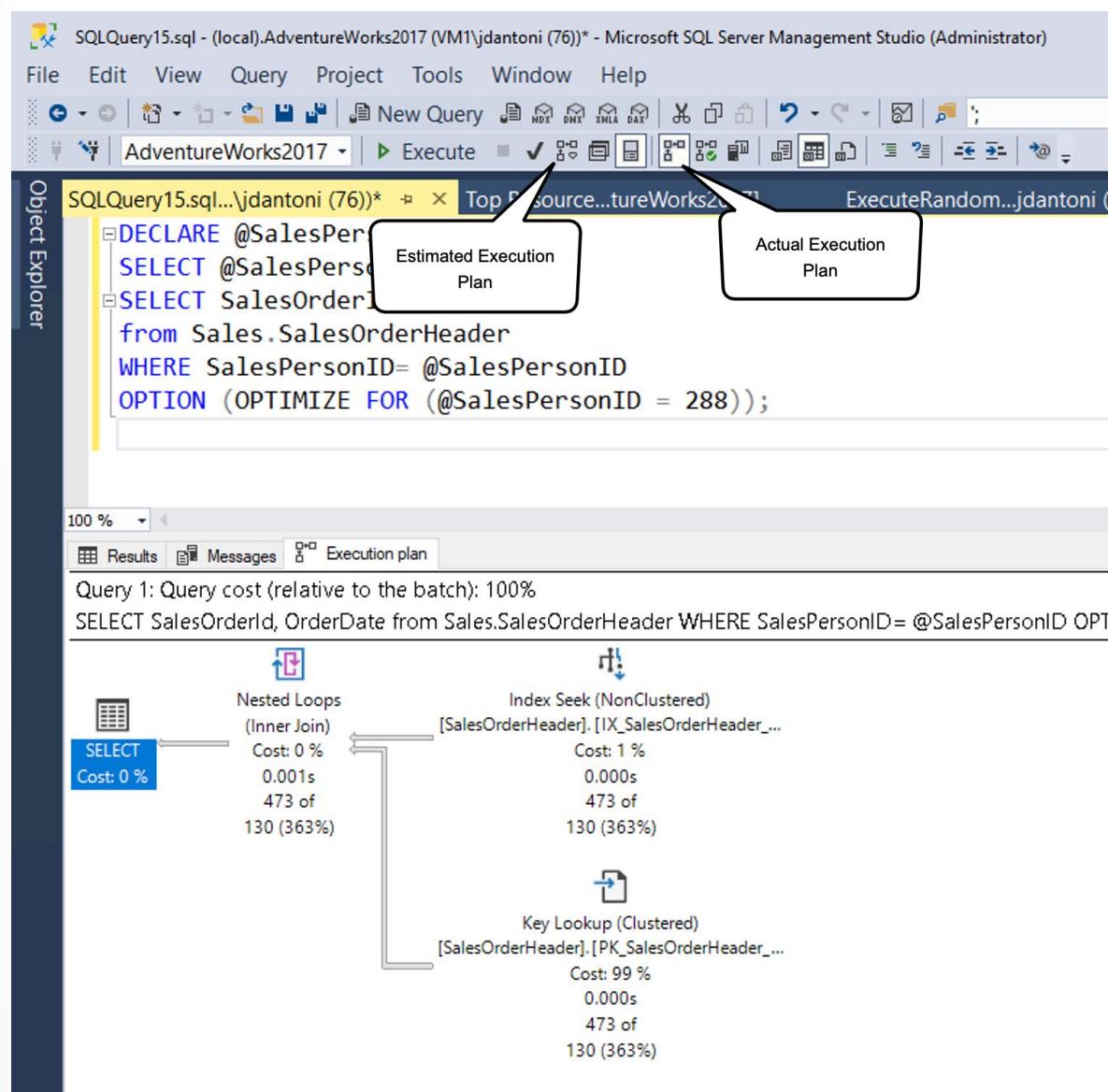


Figure 1 An estimated execution plan generated in SQL Server Management Studio

As shown in Figure 1, you can generate an estimated plan in SSMS by clicking the button pointed to by the estimated query plan box (or using the keyboard command Control+L). You can generate the actual plan by either clicking the icon shown or the keyboard command Control+M) and executing the query (unlike an estimated query plan which does not require the query to be executed). The two option buttons work a bit differently. The Estimated Query Plan button responds immediately to whatever query is highlighted (or the entire workspace, if nothing is highlighted). The Actual Execution plan button is a toggle. Once you click it, it says on and displays the Actual plan for all queries. To stop displaying plans, you can click the button again which will turn it off.

You should note there is overhead to both executing a query and generating an estimated execution plan, so this should be done carefully in a production environment.

For the most part you can use the estimated execution plan while writing your query, to understand its performance characteristics, identify missing indexes (to be covered in greater detail later in this lesson), or detect query anomalies. The actual execution plan is best used to understand the runtime performance

of the query, and most importantly gaps in statistical data that cause the query optimizer to make suboptimal choices based on the data it has available.

## Reading a Query Plan

Execution plans show you what tasks the database engine is performing while retrieving the data needed to satisfy a query. Let's dive into the plan in Figure 2.

First, the query itself is shown in Example 1.

```
SELECT [stockItemName]
      ,[UnitPrice] * [QuantityPerOuter] AS CostPerOuterBox
      ,[QuantityOnHand]
  FROM [Warehouse].[StockItems] s
  JOIN [Warehouse].[StockItemsHoldings] sh ON s.StockItemID = sh.StockItemID
 ORDER BY CostPerOuterBox;
```

### *Example 1 Example SQL Query with a join*

This query is joining the StockItems table to the StockItemHoldings table where the values in the column StockItemID are equal. So the database engine has to first identify those rows before it can process the rest of the query.

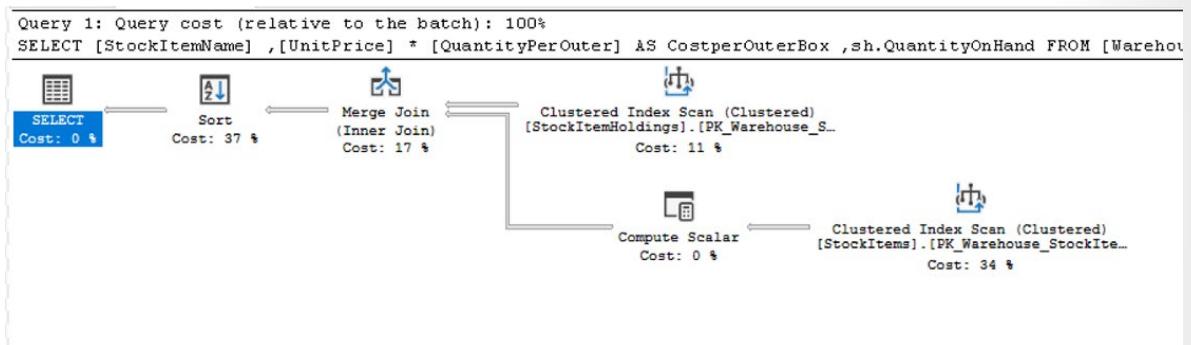


Figure 2 Query Execution Plan

Each icon in the plan shows a specific operation which represents the various actions and decisions that make up an execution plan. The SQL Server database engine has over 100 query operators that can make up on an execution plan. You will notice that under each operator icon, there is a cost percentage relative to the total cost of the query. This represents a ratio to the total, not the actual number, so even an operation that shows a cost of 0% still represents some cost. In fact, 0% is usually just due to rounding, because the graphical plan costs are always shown as whole numbers, and the real percentage is something less than 0.5%.

The flow of execution in an execution plan is from right to left, and top to bottom, so in the case of the plan in Figure 2, the Clustered Index Scan operation against the StockItemHoldings.PK\_Warehouse\_StockItemHoldings clustered index is the first operation in the query. The widths of the lines that connect the operators are based on the estimated number of rows of data that flow onward to the next operator. A very thick arrow is an indicator of large operator to operator transfer and may be indicative of an oppor-

tunity to tune a query. You can also hold your mouse over an operator and see additional information in a ToolTip as shown in Figure 3.

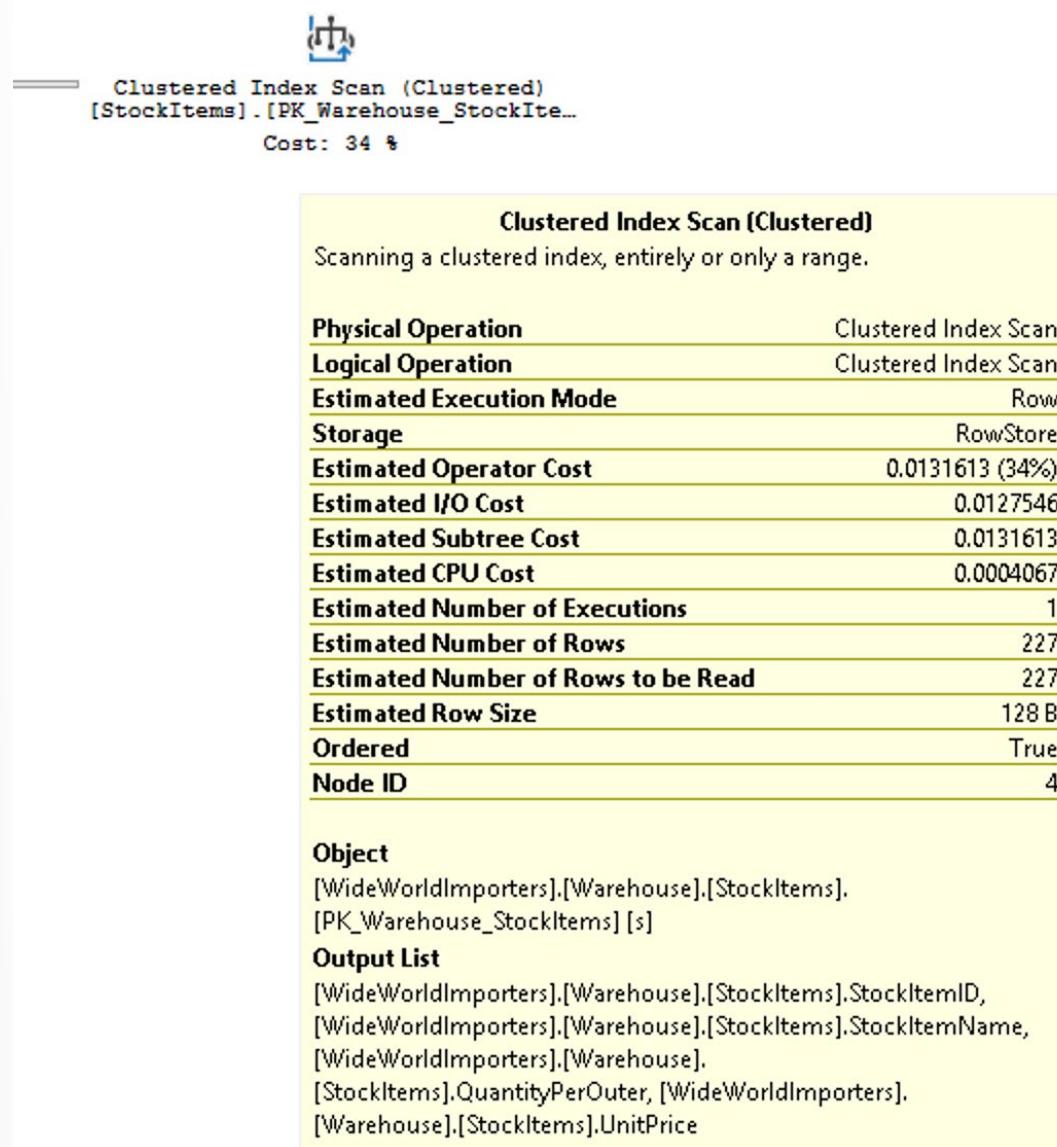


Figure 3 ToolTip for the Clustered Index Scan Operation on the StockItems table

The tooltip highlights the cost and estimates for the estimated plan, and for an actual plan will include the comparisons to the actual rows and costs. Each operator also has properties that will show you more than the tooltip does. If you right-click on a specific operator you can select the Properties option from the context menu to see the full property list as shown in Figure 4. This option will open up a separate Properties pane in SQL Server Management Studio, which by default is on the right side. Once the Properties pane is open, clicking on any operator will populate the Properties list with properties for that operator. Alternatively, you can open the Properties pane by clicking on View in the main SQL Server Management Studio menu and choosing Properties.

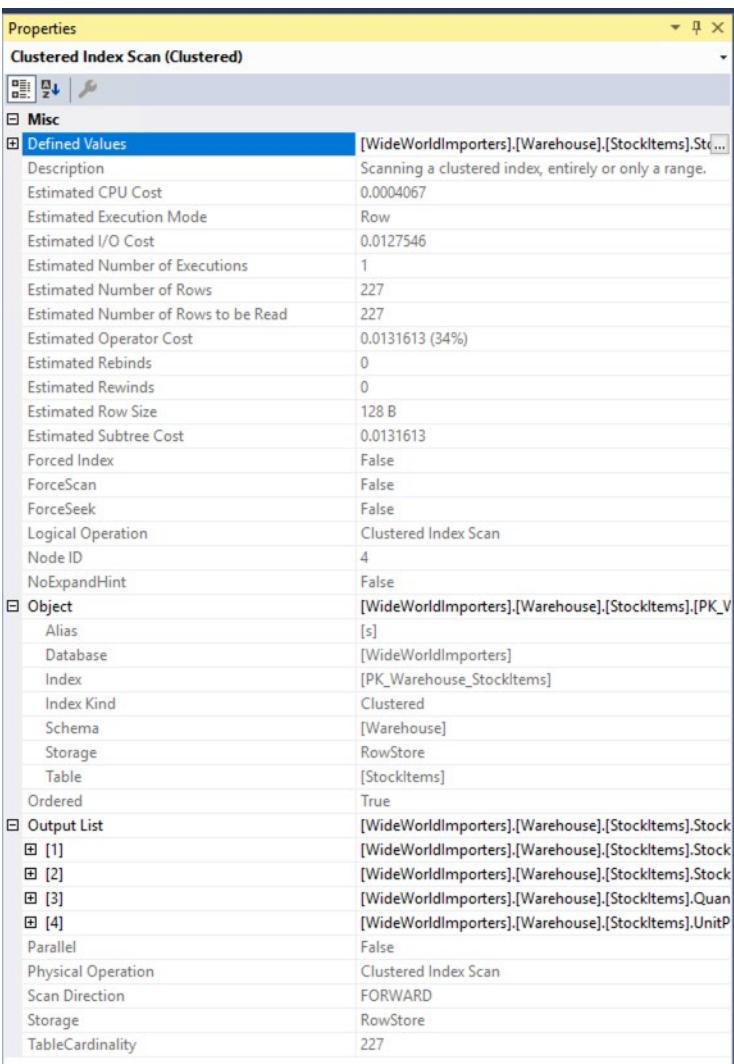


Figure 4 Properties for Operator

The Properties pane includes some additional information and shows the output list which provides details of the columns being passed to the next operator. Examining these columns, in conjunction with a clustered index scan operator can indicate that an additional nonclustered index might be needed to improve the performance of the query. Since a clustered index scan operation is reading the entire table, in this scenario a non-clustered index on the StockItemID column in each table could be more efficient.

## Lightweight Query Profiling

As mentioned above, capturing actual execution plans, whether using SSMS or the Extended Events monitoring infrastructure (to be covered in more detail later in this module) can have a large amount of overhead, and is typically only done in live site troubleshooting efforts. Observer overhead, as it is known, is the cost of monitoring a running application. In some scenarios this can be just a few percentage points of CPU utilization, but in other cases like capturing actual execution plans, it can slow down individual query performance significantly. The legacy profiling infrastructure in SQL Server's engine could produce up to 75% overhead for capturing query information, whereas the lightweight profiling infrastructure has a maximum overhead of around 2%.

Starting with SQL Server 2014 SP2 and SQL Server 2016, Microsoft introduced lightweight profiling and enhanced it with SQL Server 2016 SP1 and all later versions. In the first version of this feature, lightweight profiling collected row count and I/O utilization information (the number of logical and physical reads and writes performed by the database engine to satisfy a given query). In addition, a new extended event called `query_thread_profile` was introduced to allow data from each operator in a query plan to be inspected. In the initial version of lightweight profiling, using the feature requires trace flag 7412 to be enabled globally.

In newer releases (SQL Server 2016 SP2 CU3, SQL Server 2017 CU11, and SQL Server 2019), if lightweight profiling is not enabled globally, you can use the `USE HINT` query hint with `QUERY_PLAN_PROFILE` to enable lightweight profiling at the query level. When a query that has this hint completes execution, a `query_plan_profile` extended event is generated, which provides an actual execution plan. You can see an example of a query with this hint in Example 2.

```
SELECT [stockItemName]
      , [UnitPrice] * [QuantityPerOuter] AS CostPerOuterBox
      , [QuantityOnHand]
FROM [Warehouse].[StockItems] s
JOIN [Warehouse].[StockItems] sh ON s.StockItemID = sh.StockItemID
ORDER BY CostPerOuterBox
OPTION(USE HINT ('QUERY_PLAN_PROFILE'));
```

*Example 2 SQL Query with the 'Query\_Plan\_Profile' Hint*

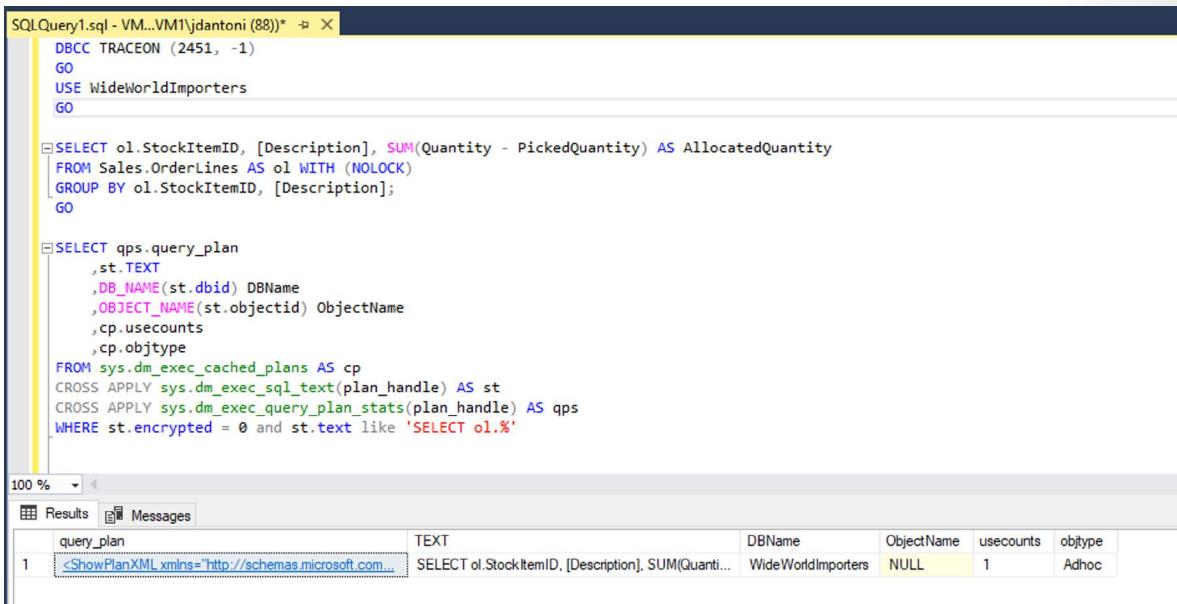
## Last Query Plans Stats

SQL Server 2019 and Azure SQL Database support two further enhancements to the query profiling infrastructure. First, lightweight profiling is enabled by default in both SQL Server 2019 and Azure SQL Database and Managed Instance, and is available as a database scoped configuration option, called `LIGHTWEIGHT_QUERY_PROFILING`, so you can disable it for any of your user databases independent of each other. Second, there is a new dynamic management function called `sys.dm_exec_query_plan_stats` which can show you the last known actual query execution plan for a given plan handle. This is an opt-in feature and requires trace flag 2451 to be enabled server-wide, or it can be enabled by using a database scoped configuration option, called `LAST_QUERY_PLAN_STATS`, for a particular database. It can also be combined other objects to get the last execution plan for all cached queries as shown in Example 3.

```
SELECT *
FROM sys.dm_exec_cached_plans AS cp
CROSS APPLY sys.dm_exec_sql_text(plan_handle) AS st
CROSS APPLY sys.dm_exec_query_plan_stats(plan_handle) AS qps;
GO
```

*Example 3 SQL Query to Return last actual execution plan for all plans in the Plan Cache*

This functionality lets you quickly identify the runtime stats for the last execution of any query in your system, with minimal overhead. Figure 5 shows how to retrieve the plan. If you click on the execution plan XML, which will be the first column of results, will display the execution plan shown in Figure 6.



```
SQLQuery1.sql - VM...VM1\jdantoni (88)* -> X
DBCC TRACEON (2451, -1)
GO
USE WideWorldImporters
GO

SELECT ol.StockItemID, [Description], SUM(Quantity - PickedQuantity) AS AllocatedQuantity
FROM Sales.OrderLines AS ol WITH (NOLOCK)
GROUP BY ol.StockItemID, [Description];
GO

SELECT qps.query_plan
, st.TEXT
, DB_NAME(st.dbid) DBName
, OBJECT_NAME(st.objectid) ObjectName
, cp.uscounts
, cp.objtype
FROM sys.dm_exec_cached_plans AS cp
CROSS APPLY sys.dm_exec_sql_text(plan_handle) AS st
CROSS APPLY sys.dm_exec_query_plan_stats(plan_handle) AS qps
WHERE st.encrypted = 0 and st.text like 'SELECT ol.%'
```

query_plan	TEXT	DBName	ObjectName	uscounts	objtype
<a href="#">&lt;ShowPlanXML&gt;</a> xmlns="http://schemas.microsoft.com/	SELECT ol.StockItemID, [Description], SUM(Quant...	WideWorldImporters	NULL	1	Adhoc

Figure 5 Retrieving the Actual Execution Plan for a Query

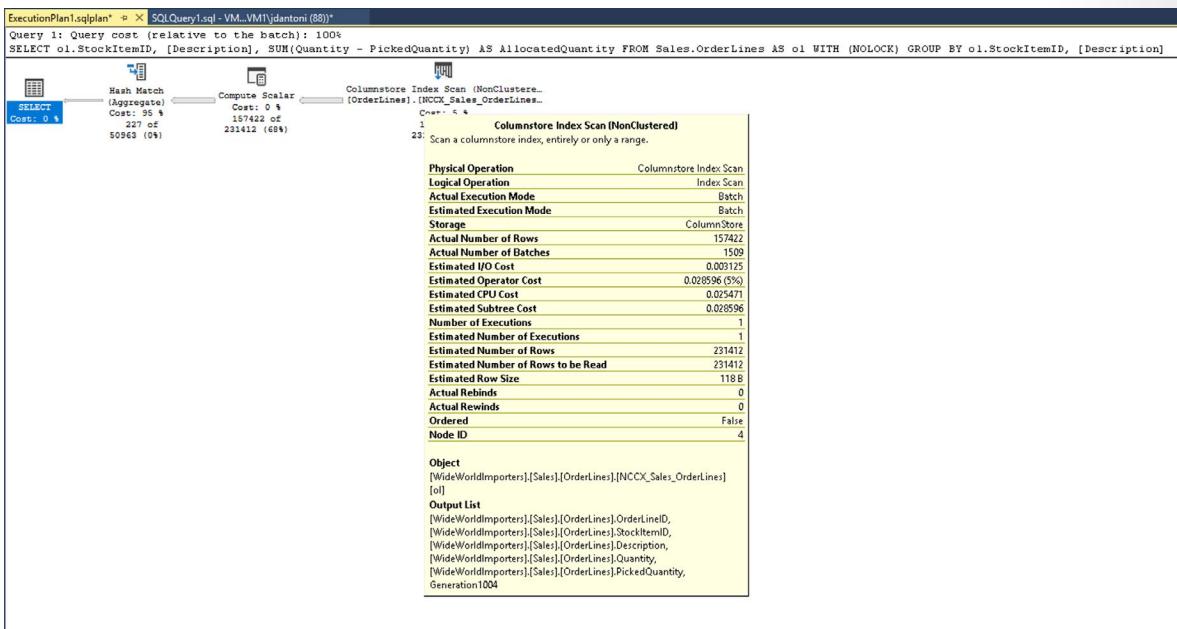


Figure 6 Execution Plan retrieved from SQL Server 2019

As you can see from the properties of the Columnstore Index Scan in Figure 6, the plan retrieved from the cache has actual number of rows retrieved in the query.

# Identifying Problematic Query Plans

The path most DBAs take to troubleshoot query performance is to first identify the problematic query (typically the query consuming the highest amount of system resources), and then retrieve that query's execution plan. There are two scenarios—one is that the query consistently performs poorly. This can be caused by a few different problems: hardware resource constraints (though this typically will not affect a single query running in isolation), a suboptimal query structure, database compatibility settings, missing indexes, or poor choice of plan by the query optimizer. The second scenario is that the query performs well for some executions, but not others. This can be caused by a few other factors—the most common being data skew in a parameterized query which will have an efficient plan for some executions, and a poor one for other executions. The other common factors in inconsistent query performance are blocking, where a query is waiting on another query to complete in order to gain access to a table, or hardware contention.

Let's look at each of these potential problems in more detail.

## Hardware Constraints

For the most part, hardware constraints will not manifest themselves with single query executions but will be evident when production load is applied and there is a limited number of CPU threads and a limited amount of memory to be shared among the queries. When you have CPU contention it will usually be detectable by observing the performance monitor counter '% Processor Time' which measures the CPU usage of the server. Looking deeper into SQL Server, you may see SOS\_SCHEDULER\_YIELD and CXPACKET wait types when the server is under CPU pressure. However, in some cases with extremely poor storage system performance, even single executions of a query that is otherwise optimized can be extremely slow. Storage system performance is best tracked at the operating system level using the performance monitor counters 'Disk Seconds/Read' and 'Disk Seconds/Write' which measure how long an I/O operation takes to complete. SQL Server will write to its error log in the event of extremely poor storage performance (if an I/O takes longer than 15 seconds to complete). If you look at wait statistics and see a high percentage of PAGEIOLATCH\_SH waits in your SQL Server this can be another indication of a storage system performance issue. Typically, hardware performance is examined at a high level, early in the performance troubleshooting process, because it is relatively easy to evaluate. You can refer to the baseline information in Module 4 for more information on hardware.

Note that most database performance issues can be attributed to sub-optimal query patterns, but in many cases running inefficient queries will put undue pressure on your hardware. For example, missing indexes could lead to CPU, storage, and memory pressure by retrieving more data than is required to process the query. It is recommended that you address suboptimal queries and tune them, before addressing hardware issue. We'll start looking at query tuning next.

## Suboptimal Query Constructs

Relational databases perform best when executing set-based operations. This means performing data manipulation operations (INSERT, UPDATE, DELETE and SELECT) in sets—where work is done on a set of values and produces either a single value or a result set. The alternative to set-based operations is to perform row-based work, using a cursor or a while loop—this is known as row-based processing, and its cost increases linearly with the number of rows impacted. That linear scale is particularly problematic as data volumes grow for an application.

While detecting suboptimal use of row-based operations with cursors or WHILE loops is very important, there are other SQL Server anti-patterns that you should be able to recognize. Table valued functions (TVF), particularly multi-statement table valued functions, caused problematic execution plan patterns

prior to SQL Server 2017. Many developers like to use multi-statement table valued functions because they can execute multiple queries within a single function and aggregate the results into a single table. However, anyone writing T-SQL code needs to be aware of the possible performance penalties for using TVFs.

SQL Server has two types of table-valued functions, inline and multi-statement. If you use a an inline TVF, the database engine treats it just like a view. Multi-statement TVFs are treated just like another table when processing a query. Because TVFs are dynamic and as such, SQL Server does not have statistics on them, it used a fixed row count when estimating the query plan cost. This can be fine, if the number of rows is small, however if the TVF returns thousands or millions of rows, the execution plan could be very inefficient.

Another anti-pattern has been the use of scalar functions, which have similar estimation and execution problems. Microsoft has made a lot of headway on improving the performance of the aforementioned patterns with the introduction of Intelligent Query Processing in SQL Server 2017 and Azure SQL Database, under compatibility levels 140 and 150.

## SARGAbility

The term SARGable in relational databases refers to a predicate (WHERE clause) in a specific format that can leverage an index to speed up execution of a query. Predicates in the correct format are called 'Search Arguments' or SARGs. In the case of SQL Server, using a SARG means that the optimizer will evaluate using a nonclustered index on the column referenced in the SARG for a SEEK operation, instead of scanning the entire index (or the entire table) to retrieve a value.

Note that the presence of SARG does not guarantee the use of an index for a SEEK. The optimizer's costing algorithms could still determine that the index was too expensive. This could be the case if a SARG refers to a very large percentage of rows in a table. The absence of a SARG does mean that the optimizer will not even evaluate a SEEK on a nonclustered index.

Some examples of expressions that are not SARGs (sometimes said to be non-sargable) are those that include a LIKE clause with a wildcard at the beginning of the string to be matched, e.g. WHERE lastName LIKE '%SMITH%', or using functions on a column e.g. WHERE CONVERT(CHAR(10), CreateDate,121) = '2020-03-22'. These queries with non-sargable expressions are typically identified by examining execution plans for index or table scans, where seeks should otherwise be taking place. In Figures 7 and 8, you can see this in action.

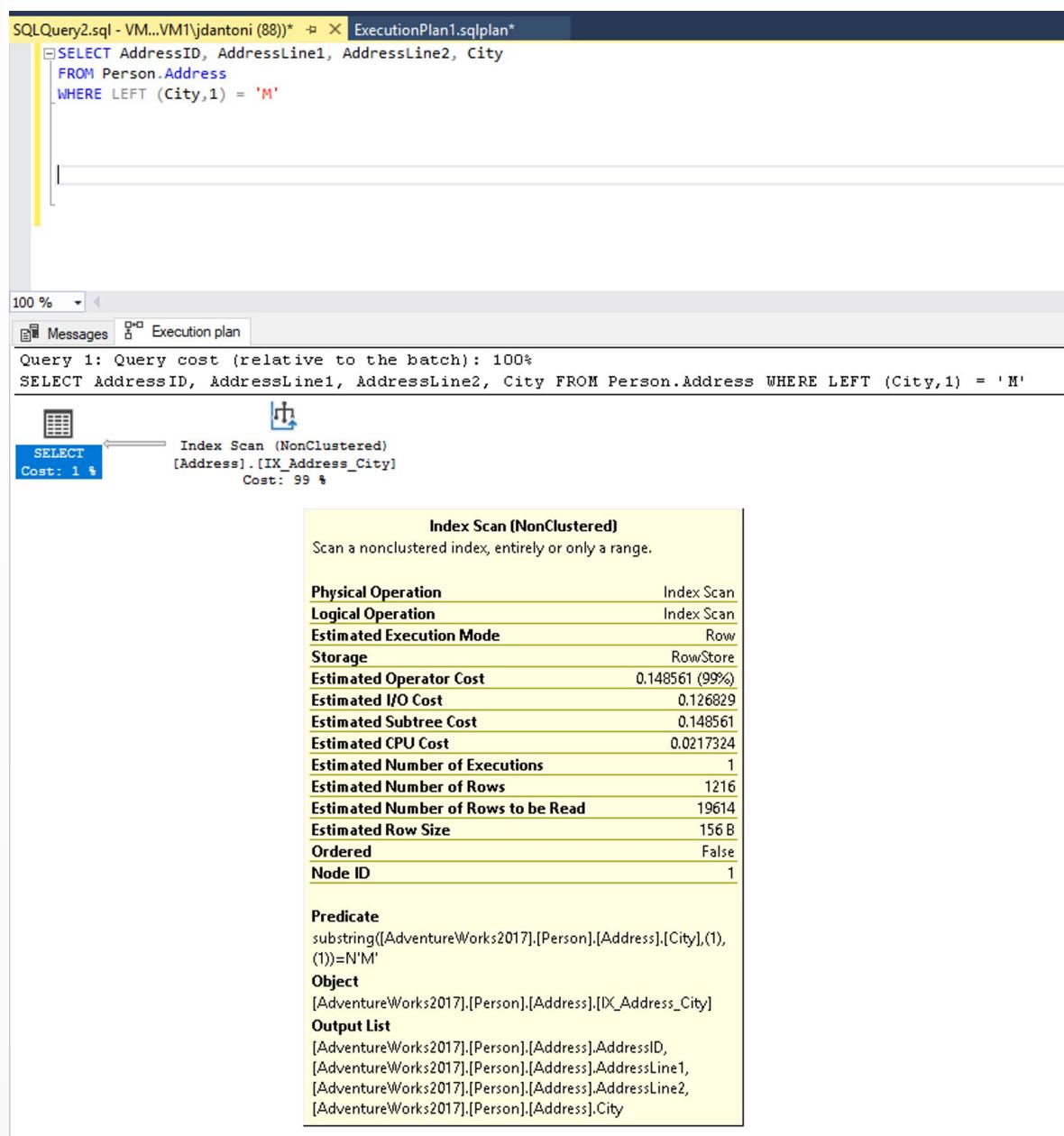


Figure 7 Query and Execution Plan using a non-SARGable Function

There is an index on the City column that is being used in the WHERE clause of the query and while it is being used in this execution plan in Figure 7, you can see the index is being scanned, which means the entire index is being read. The LEFT function in the predicate makes this expression non-SARGable. The optimizer will not evaluate using an index seek on the index on the City column.

This query could be written to use a predicate that is SARGable. The optimizer would then evaluate a SEEK on the index on the City column. An index SEEK, in this case, would read a much smaller set of rows, as shown in Figure 8.

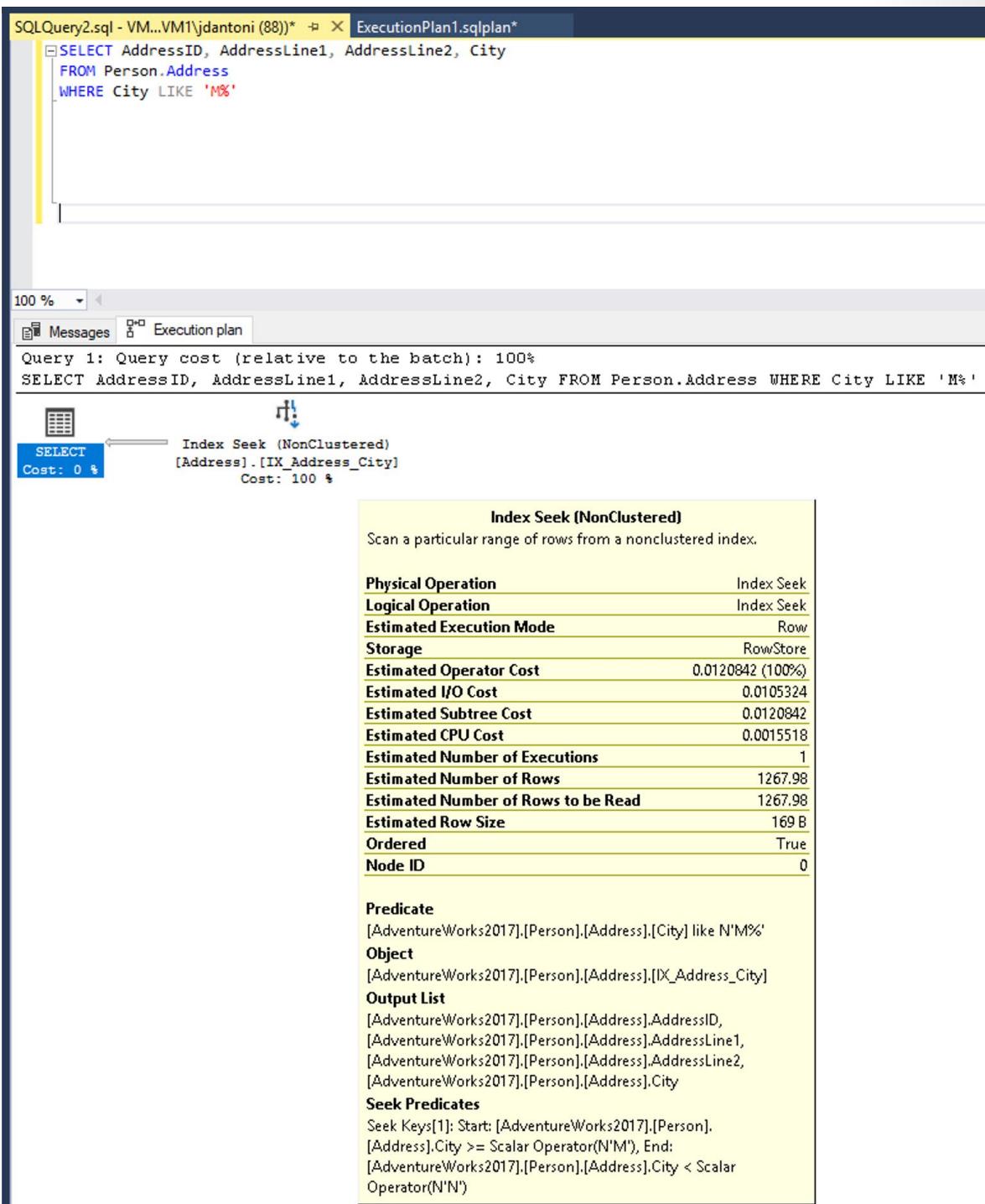


Figure 8 A Query and Execution Plan with a SARGable Predicate

By changing the LEFT function into a LIKE, and index SEEK is used. You should note that this LIKE does not have a wildcard on the left, so it is looking for cities that begin with M. , if it was "two-sided" or started with a wildcard ('%M%' or '%M') it would be non-SARGable. The seek operation is estimated to return 1267 rows, or approximately 15% of the estimate for the query with the non-SARGable predicate.

Some other database development anti-patterns are treating the database as a service rather than a data store. Using a database to convert data to JSON, manipulate strings, or perform complex calculations can

lead to excessive CPU use and increased latency. Queries that try to retrieve all records and then perform computations in the database can lead to excessive IO and CPU usage. Ideally, you should use the database for data access operations and optimized database constructs like aggregation.

## Missing Indexes

The most common performance problems we see as database administrators are due to a lack of useful indexes causing the engine to read far more pages than necessary to return the results of a query. While indexes are not free in terms of resources (adding additional indexes to a table can impact write performance and consume space) the performance gains they offer can offset the additional resource costs many times over. Frequently execution plans with these performance issues can be identified by the query operator "Clustered Index Scan" or the combination of the "Nonclustered index seek" and "Key Lookup" (which is more indicative of missing columns in an existing index). The database engine attempts to help with this problem by reporting on missing indexes in execution plans. The names and details of the indexes which have been deemed potentially useful are available through a dynamic management view called `sys.dm_db_missing_index_details`. There are also other DMVs in SQL Server like `sys.dm_db_index_usage_stats` and `sys.dm_db_index_operational_stats` which highlight the utilization of existing indexes. It may make sense to drop an index that is not used by any queries in the database. The missing index DMVs and plan warnings should only be used as a starting point for tuning your queries—it's important to understand what your key queries are and build indexes to support those queries. Creating all missing indexes without evaluating indexes in the context of each other is not recommended.

## Missing and Out of Date Statistics

You have learned about the importance of column and index statistics to the query optimizer. It is also important to understand conditions that can lead to out of date statistics, and how this can manifest itself in SQL Server. SQL Server and Azure SQL Database and Managed Instance default to having auto-update statistics set to ON. Prior to SQL Server 2016, the default behavior of auto-update statistics was to not update statistics until the number of modifications to columns in the index was equal to about 20% of the number of rows in a table. Because of this behavior, you could have data modifications that were significant enough to change query performance, but not update the statistics. Any plan that used the table with the changed data would be based on out of date statistics and would frequently be suboptimal. Prior to SQL Server 2016, you had the option of using trace flag 2371, which changed the required number of modifications to be a dynamic value, so as your table grew in size, the percentage of row modifications that was required to trigger a statistics update got smaller. Newer versions of SQL Server and Azure SQL Database and Managed Instance support this behavior by default. There is also a dynamic management function called `sys.dm_db_stats_properties` which shows you the last time statistics were updated and the number of modifications that have been made since the last update, which allows you to quickly identify statistics that might need to be manually updated.

## Poor Optimizer Choices

While the query optimizer does a very good job of optimizing most queries, there are some edge cases where the cost-based optimizer may make impactful decisions that are not fully understood. There are a number of ways to address this including using query hints, trace flags, execution plan forcing, and other adjustments in order to reach a stable and optimal query plan. Microsoft has a support team that can help troubleshoot these scenarios.

In the below example from the AdventureWorks2017 database, a query hint is being used to tell the database optimizer to always use a city name of Seattle. This will not guarantee the best execution plan

for all city values, but it will be predictable. Note that the value of 'Seattle' for @city\_name will only be used during optimization. During execution the actual supplied value ('Ascheim') will be used.

```
DECLARE @city_name nvarchar(30);

DECLARE @postal_code nvarchar(15);

SELECT @city_name = 'Ascheim';

SELECT @postal_code = 86171;

SELECT * FROM Person.Address

WHERE City = @city_name AND PostalCode = @postal_code

OPTION (OPTIMIZE FOR (@city_name = 'Seattle'));
```

*Example 4—A T-SQL Query Batch with the OPTIMIZE\_FOR query hint.*

As seen in Example 4, the query uses a hint (the OPTION clause) to tell the optimizer to use a specific variable value to build its execution plan.

## Parameter Sniffing

Earlier in this module you learned about how SQL Server caches query execution plans for future use. Since the execution plan retrieval process is based on the hash value of a query, the query text has to be identical for every execution of the query for the cached plan to be used. In order to support multiple values in the same query, many developers use parameters, usually passed in through stored procedures, as seen in Example 5.

```
CREATE PROC GetAccountID (@Param INT)

AS

<other statements in procedure>

SELECT accountid FROM CustomerSales WHERE sales > @Param;

<other statements in procedure>

RETURN;

-- Call the procedure:

EXEC GetAccountID 42;
```

*Example 5—A procedure with a parameter*

Queries can also be explicitly parameterized using the procedure sp\_executesql. However, explicit parameterization of individual queries is usually done through the application with some form (depending on the

API) of PREPARE and EXECUTE. When the database engine executes that query for the first time, it will optimize the query based on the initial value of the parameter, in this case, 42. This behavior, called parameter sniffing, allows for the overall workload of compiling queries to be reduced on the server. However, in the event that there a lot of data skew—for example a table that had 10 million records, and 99% of those records had an ID of 1, and the other 1% were unique numbers, query performance would vary widely, based on which ID was initially used to optimize the query. This wildly fluctuating performance is indicative of data skew and is not an inherent problem with parameter sniffing. This is a fairly common performance problem that you should be aware of and understand the options for alleviating the issue. There a few ways to address this problem, but they each come with tradeoffs:

- Use the RECOMPILE hint in your query, or the WITH RECOMPILE execution option in your stored procedures. This will cause the query or procedure to be recompiled every time it is executed, which will increase CPU utilization on the server but will always use the current parameter value
- You can use the OPTIMIZE FOR UNKNOWN query hint. This will cause the optimizer to choose to not sniff parameters and compare the value with column data histogram. This will not get you the best possible plan but will allow for a consistent execution plan.
- Rewrite your procedure or queries by adding logic around parameter values to only RECOMPILE for known troublesome parameters. In the example below, if the SalesPersonID parameter is NULL, the query will be executed with the OPTION (RECOMPILE).

```
CREATE OR ALTER PROCEDURE GetSalesInfo (@SalesPersonID INT = NULL)
AS
DECLARE @Recompile BIT = 1
,@SQLString NVARCHAR(500)

IF @SalesPersonID IS NULL
SET @Recompile = 1

SELECT @SQLString = N'SELECT SalesOrderID, OrderDate FROM Sales.SalesOrder-
Header WHERE SalesPersonID = @SalesPersonID'

IF @Recompile = 1
BEGIN
SET @SQLString = @SQLString + N' OPTION(RECOMPILE)'
END
```

```
EXEC sp_executesql @SQLString  
  
,N'@SalesPersonID INT'  
  
,@SalesPersonID = @SalesPersonID  
  
GO
```

This is a good solution but it does require a fairly large development effort, and a firm understanding of your data distribution. It also may require maintenance as the data changes.

## Summary and Knowledge Check

Understanding execution plans and how indexes are used are the fundamentals of database performance. At the same time, you need to understand patterns and anti-patterns for relational databases and aim for good query choices and proper indexing.

### Question 1

*Which type of execution plan is stored in the plan cache?*

- Estimated execution plan
- Actual execution plan
- Live Query Stats

### Question 2

*Which DMV should you use to find index utilization?*

- sys.dm\_db\_index\_usage\_stats
- sys.dm\_db\_missing\_index\_details
- sys.dm\_exec\_query\_plan\_stats

### Question 3

*Which of the following wait types would indicate excessive CPU consumption?*

- SOS\_SCHEDULER\_YIELD
- RESOURCE\_SEMAPHORE
- PAGEIOLATCH\_SH

# Explore Performance-based Database Design

## Introduction

Database design is a very important aspect of database performance, even though it's not always under the control of the database administrator. You may be working with third- party vendor applications that you did not build. . Whenever possible, it's important to design your database properly for the workload, whether it's an online transaction processing (OLTP) or data warehouse workload. Many design decisions, such as choosing the right datatypes, can make large differences in the performance of your databases.

## Learning Objectives:

In this section, you will:

1. Understand normal form and how it affects database design
2. Choose appropriate datatypes for your database
3. Understand types of indexes

## Normalization

Database normalization is a design process used to organize a given set of data into tables and columns in a database. Each table should contain data relating to a specific 'thing' and only have data that supports that same 'thing' included in the table. The goal of this process is to reduce duplicate data contained within your database, to reduce the performance impact of database inserts and updates. For example, a customer address change is much easier to implement if the only place the customer address is stored in the Customers table. The most common forms of normalization are first, second, and third normal form and are described below.

## First Normal Form

First normal form has the following specifications:

- Create a separate table for each set of related data
- Eliminate repeating groups in individual tables
- Identify each set of related data with a primary key

In this model you should not use multiple columns in a single table to store similar data. For example, if product can come in multiple colors, you should not have multiple columns in a single row containing the different color values. Table 1 below (ProductColors) is not in first normal form as there are repeating values for color. For products with only one color, there is wasted space. And what if a product came in more than three colors? Rather than having to set a maximum number of colors, we can recreate the table as shown in Table 2, ProductColor. We also have a requirement for first nomal form that there is a unique key for the table, which is column (or columns) whose value uniquely identifies the row. Neither of the columns in Table 2 is unique, but together, the combination of ProductID and Color is unique. When multiple columns are needed, we call that a composite key.

ProductID	Color1	Color2	Color3
1	Red	Green	Yellow
2	Yellow		
3	Blue	Red	

ProductID	Color1	Color2	Color3
4	Blue		
5	Red		

Table 1: ProductColors

ProductID	Color
1	Red
1	Green
1	Yellow
2	Yellow
3	Blue
3	Red
4	Blue
5	Red

Table 2: ProductColor

Table 3, ProductInfo, is in first normal form because each row refers to a particular product, there are no repeating groups, and we have the column ProductID to use as a Primary Key.

ProductID	ProductName	Price	ProductionCountry	ShortLocation
1	Widget	15.95	United States	US
2	Foop	41.95	United Kingdom	UK
3	Glombit	49.95	United Kingdom	UK
4	Sorfin	99.99	Republic of the Philippines	RepPhil
5	Stem Bolt	29.95	United States	US

Table 3: ProductInfo

## Second Normal Form

Second normal form has the following specification, in addition to those required by first normal form:

- If the table has a composite key, all attributes must depend on the complete key, not just part of it.

Second normal form is only relevant to tables with composite keys, like ProductColor in Table 2 above. Consider the case where the ProductColor table also includes the product's price. This table has a composite key on ProductID and Color, because only using both column values can we uniquely identify a row. If a product's price does not change with the color, we might see data as shown in Table 4.

ProductID	Color	Price
1	Red	15.95
1	Green	15.95
1	Yellow	15.95
2	Yellow	41.95
3	Blue	49.95
3	Red	49.95
4	Blue	99.95

ProductID	Color	Price
5	Red	29.95

Table 4: ProductColorPrice

Table4 is NOT in second normal form. The price value is dependent on the ProductID but not on the Color. There are three rows for ProductID 1, so the price for that product is repeated 3 times. The problem with violating second normal form is that if we have to update the price, we have to make sure we update it everywhere. If we update the price in the first row, but not the second or third, we would have something called an 'update anomaly'. After the update, we wouldn't be able to tell what the actual price for ProductID 1 was. The solution is to move the Price column to a table that has ProductID as a single column key, because that is the only column that Price depends on. For example, we could use Table 3 to store the Price.

Note that if the price for a product was different based on the color of the product, then Table 4 would be in second normal form, because the price would depend on both parts of the key: ProductID and Color.

## Third Normal Form

Third normal form is typically the aim for most OLTP databases. Third normal form has the following specification, in addition to those required by second normal form:

- All nonkey columns are non-transitively dependent on the primary key.

The transitive relationship implies that one column in a table is related to other columns, through a second column. In the case of dependence, when you say that a column is dependent on another column, it means that the value of one can be derived from the other. For example, your age can be determined from your date of birth, making your age dependent on your date of birth. Refer back to Table 3, ProductInfo. This table is in second normal form, but not in third. The ShortLocation column is dependent on the ProductionLocation column which is not the key. Like second normal form, violating third normal form can lead to update anomalies. If we updated the ShortLocation in one row, without updating in all the rows where that location occurred, we would end up with inconsistent data. To prevent this, we could create a separate table to store country names and their shortened forms.

## Denormalization

While the 3<sup>rd</sup> normal form is theoretically desirable, it is not always possible for all data. In addition, a completely normalized database does not always give you the best performance. Normalized data frequently requires multiple join operations to get all the necessary data returned in a single query. There is a tradeoff between normalizing data where the number of joins required to return query results has high CPU utilization, and denormalized data that has fewer joins and less CPU required, but opens up the possibility of update anomalies.

Note that denormalized data is not the same as unnormalized. For denormalization, we start by designing tables that are normalized. Then we can add additional columns to some tables to reduce the number of joins required, but as we do so, we are aware of the possible update anomalies. We then make sure we have triggers or other kinds of processing that will make sure that when we perform an update, all the duplicate data is also updated.,

Denormalized data can be more efficient to query, especially for read heavy workloads like a data warehouse. In those cases, having additional columns may offer better query patterns and/or more simplistic queries.

## Star Schema

While most normalization is aimed at OLTP workloads, data warehouses have their own modeling structure, which is usually a **denormalized** model. This design uses fact tables, which record measurements or metrics for specific events like a sale, and joins them to dimension tables, which are smaller in terms of row count, but may have a large number of columns to describe the fact data. Some example dimensions would include inventory, time, and/or geography. This design pattern is used to make the database easier to query and offer performance gains for read workloads.

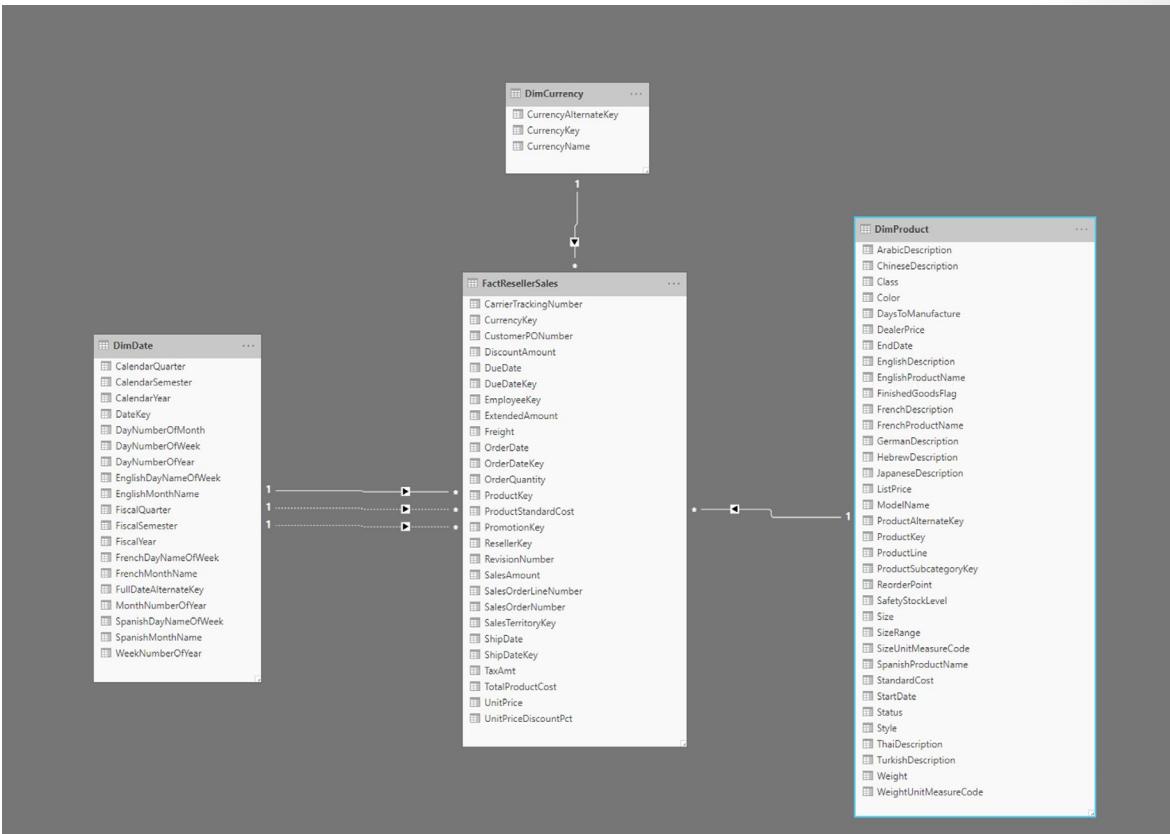


Figure 9 A Sample Star Schema

Figure 9 shows an example of a star schema, including a *FactResellerSales* fact table, and dimensions for date, currency, and products. The fact table contains data related to the sales transactions, and the dimensions only contain data related to a specific element of the sales data. For example, the *FactResellerSales* table contains only a *ProductKey* to indicate which product was sold. All of the details about each product is stored in the *DimProduct* table, and related back to the fact table with the *ProductKey* column.

Related to star schema design is a snowflake schema, which uses a set of more normalized tables for a single business entity. Figure 10 shows an example of a single dimension for a snowflake schema. The Products dimension is normalized and stored in three tables called *DimProductCategory*, *DimProductSubcategory*, and *DimProduct*.

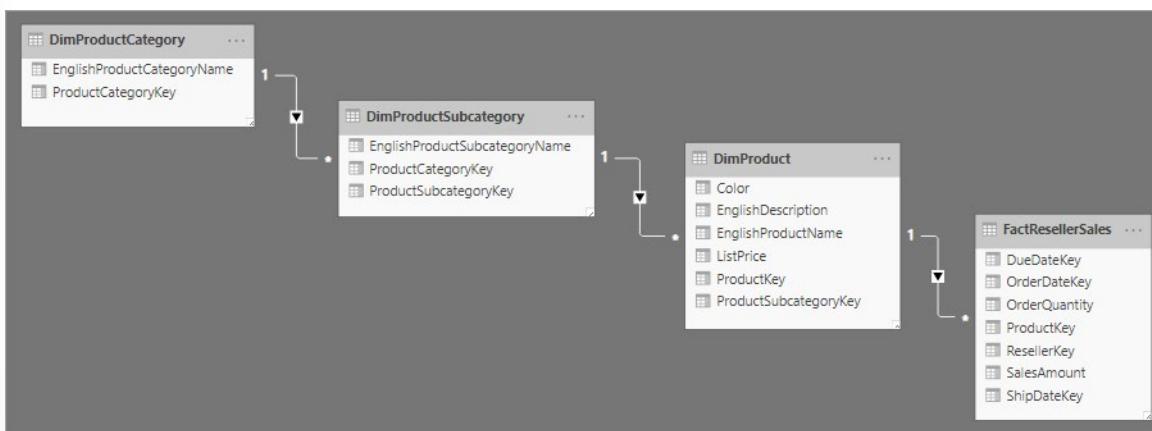


Figure 10 Sample Snowflake Schema

The main difference between star and snowflake schemas is that the dimensions in a snowflake schema are normalized to reduce redundancy, which saves storage space. The tradeoff is that your queries require more joins which can increase your complexity and decrease performance.

## Choosing Appropriate Data Types

SQL Server offers a wide variety of data types to choose from, and your choice can affect performance in a number of ways. While SQL Server can convert some data types automatically (we call this an ‘implicit conversion’, conversion can be costly and can also negatively impact query plans. The alternative is an explicit conversion, where you use the CAST or CONVERT function in your code to force a data type conversion. Additionally, choosing data types that are much larger than needed can cause wasted space and require more pages than is necessary to be read. It is important to choose the right data types for a given set of data—which will reduce the total storage required for the database and improve the performance of queries executed. Figure 11 indicates in which cases SQL Server can do an implicit conversion and in which cases you must explicitly convert datatypes in your code. Also note that in some cases, conversions are not possible at all. For example, a date cannot be converted to a bit. Conversions can negatively impact query performance by causing index scans where seeks would have been possible, and additional CPU overhead from the conversion itself.

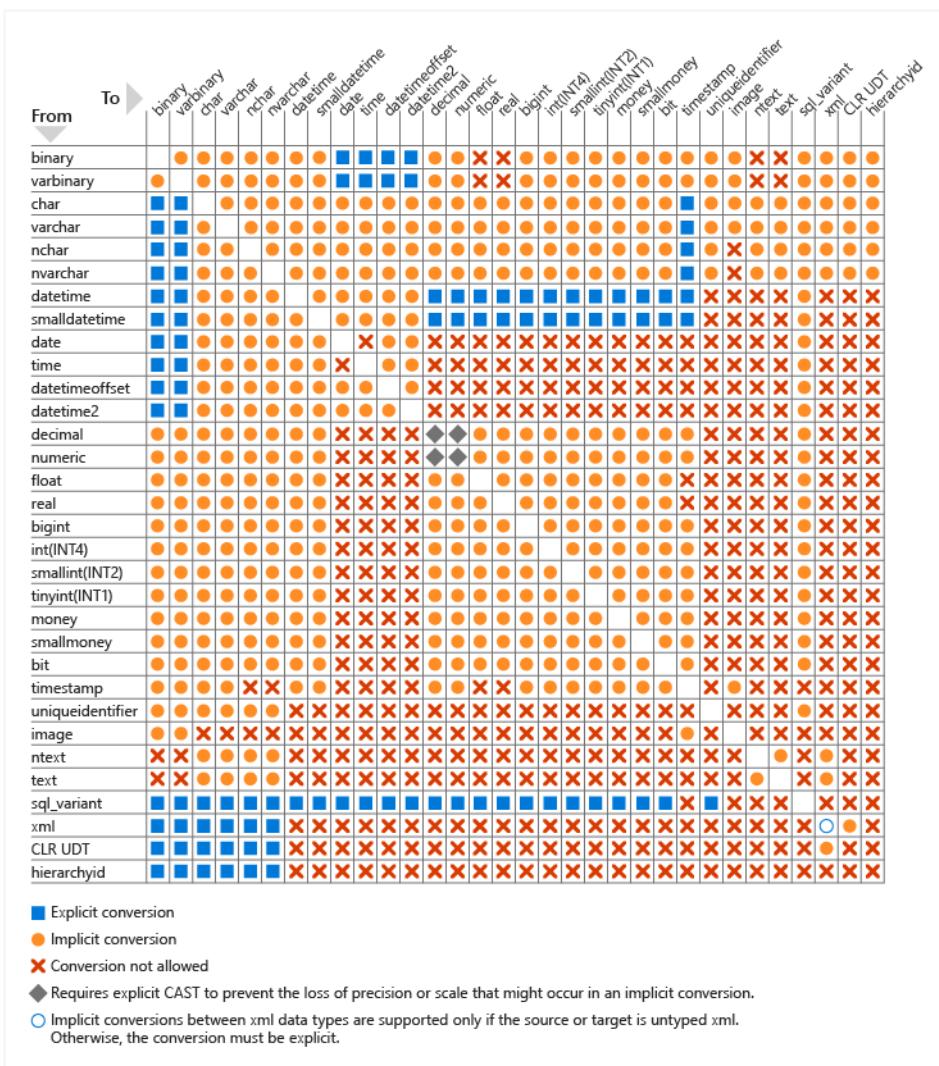


Figure 11 Chart of Type Conversions in SQL Server and Azure SQL

SQL Server offers a set of system supplied data types for all data that can be used in your tables and queries, and also allows the creation of user defined data types in either T-SQL or the .NET framework.

## Designing Indexes

SQL Server has several index types to support different types of workloads. At a high level, an index can be thought of as an on-disk structure that is associated with a table or a view, that enables SQL Server to more easily find the row or rows associated with the index key (which consists of one or more columns in the table or view), compared to scanning the entire table.

### Clustered Indexes

A common DBA job interview question is to ask the candidate the difference between a clustered and nonclustered index—as indexes are a fundamental data storage technology in SQL Server. A clustered index is the underlying table, stored in sorted order based on the key value. There can only be one clustered index on a given table, because the rows can be stored in one order. A table without a clustered index is called a heap, and heaps are typically only used as staging tables. An important performance

design principle is to keep your clustered index key as narrow as possible. When considering the key column(s) for your clustered index you should consider columns that are unique or that contain many distinct values. Another property of a good clustered index key is for records that are accessed sequentially, and are used frequently to sort the data retrieved from the table. Having the clustered index on the column used for sorting can prevent the cost of sorting every time that query executes, because the data will be already stored in the desired order.

Note that when we say that the table is 'stored' in a particular order, we are referring to the logical order, not necessarily the physical, on-disk order. Indexes have pointers between pages, and the pointers help create the logical order. When scanning an index 'in order', SQL Server follows the pointers from page to page. Immediately after creating an index, it is most like also stored in physical order on the disk, but after you start making modifications to the data, and new pages need to be added to the index, the pointers will still give us the correct logical order, but the new pages will most likely not be in physical disk order.

## Nonclustered Indexes

Nonclustered indexes are a separate structure from the data rows. A nonclustered index contains the key values defined for the index, and a pointer to the data row that contains the key value. You can add additional nonkey columns to the leaf level of the nonclustered index to cover more columns using the included columns feature in SQL Server. You can create multiple nonclustered indexes on a table.

An example of when you need to add an index or add columns to an existing nonclustered index is shown in Figure 12.

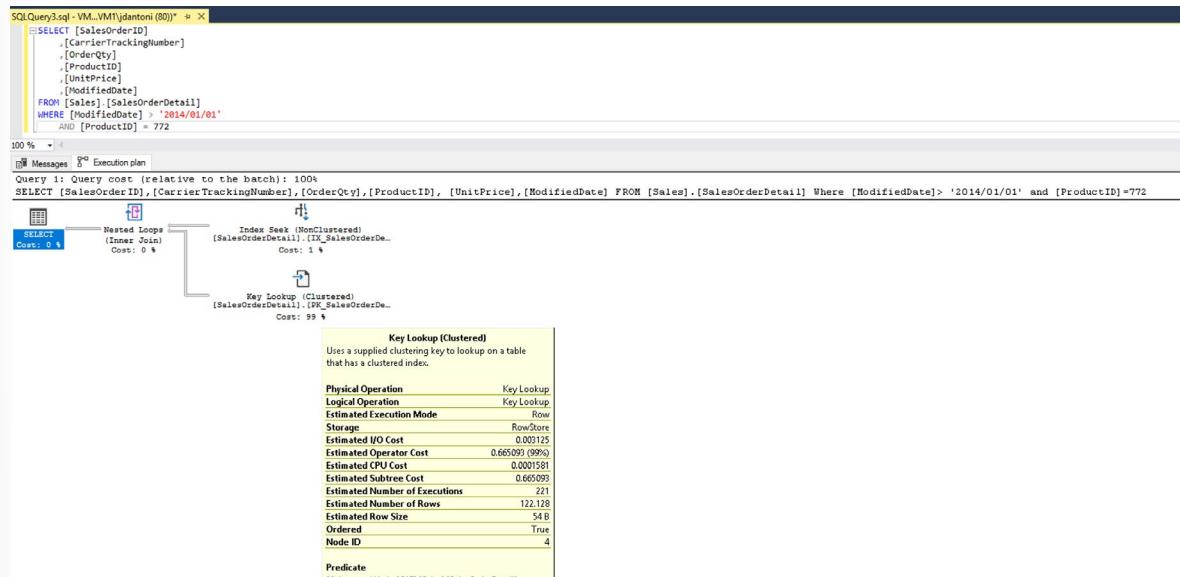


Figure 12 Query and Query Execution Plan with a Key Lookup operator

The query plan indicates that for each row retrieved using the index seek, additional data will need to be retrieved from the clustered index (the table itself). There is a nonclustered index, but it only includes the product column. If you add the other columns in the query to a nonclustered index as shown in Figure 13, you can see the execution plan change to eliminate the key lookup.

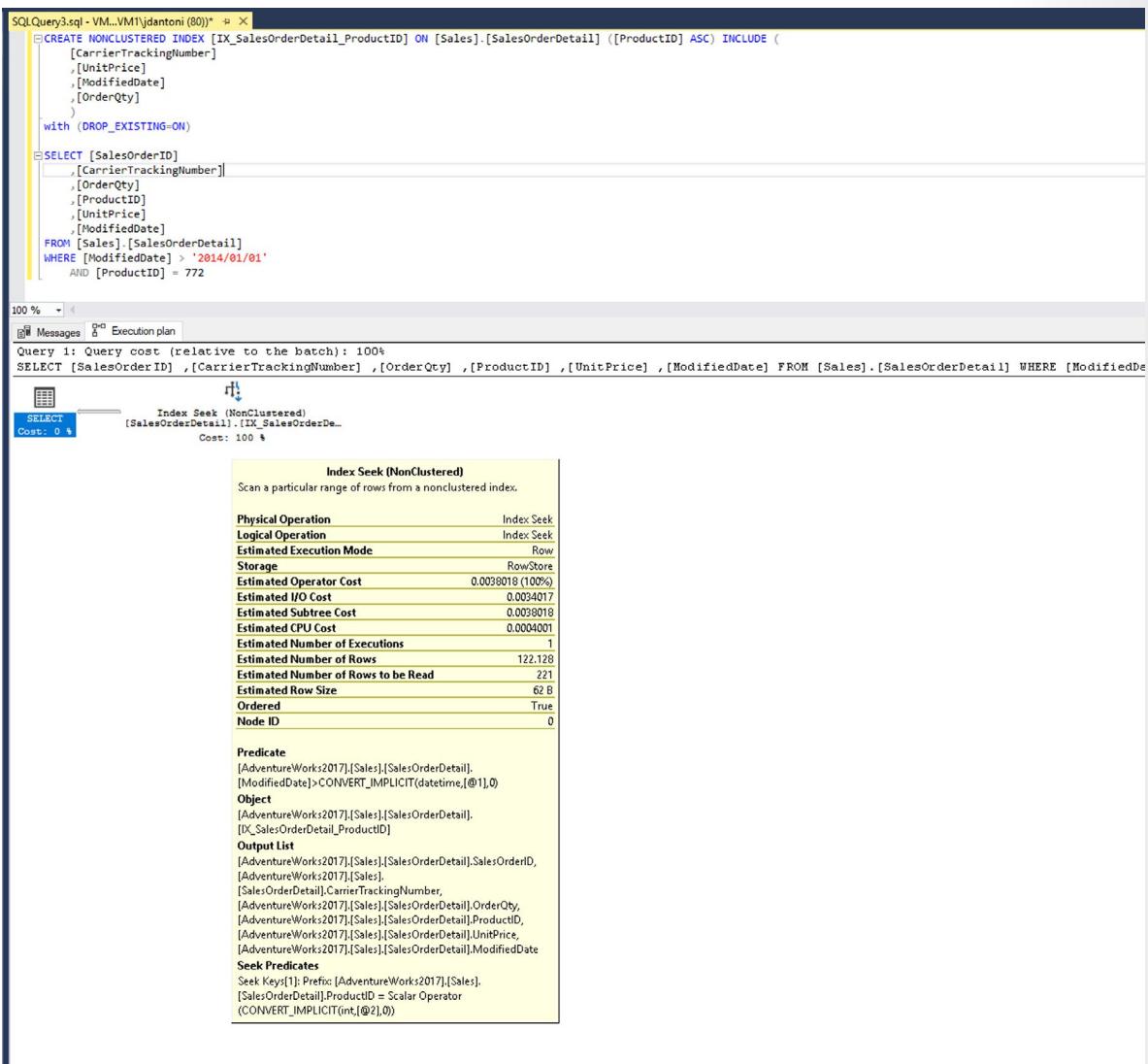


Figure 13 Changing the Index and the Query Plan with No Key Lookup

The index created in Figure 13 is an example of a covering index, where in addition to the key column you are including additional columns to cover the query and eliminate the need to access the table itself.

Both nonclustered and clustered indexes can be defined as unique, meaning there can be no duplication of the key values. Unique indexes are automatically created when you create a PRIMARY KEY or UNIQUE constraint on a table.

The focus of this section is on b-tree indexes in SQL Server—these are also known as row store indexes. The general structure of a b-tree is shown in Figure 14.

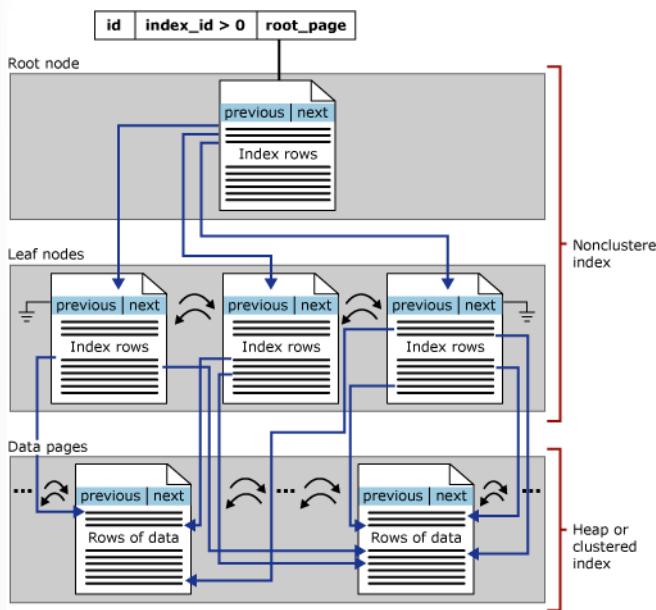


Figure 14 The B-tree architecture of an index in SQL Server and Azure SQL

Each page in an index b-tree is called an index node, and the top node of b-tree is called the root node. The bottom nodes in an index are called leaf nodes and the collection of leaf nodes is the leaf level.

Index design is a little bit art and a little bit science. A narrow index with few columns in its key requires less time to update and has lower maintenance overhead; however it may not be useful for as many queries as a wider index that includes more columns. You may need to experiment with several indexing approaches based on the columns selected by your application's queries. The query optimizer will generally choose what it considers to be the best existing index for a query; however, that does not mean that there is not a better index that could be built.

Properly indexing a database is a complex task. When planning your indexes for a table, you should keep a few basic principles in:

1. Understand the workloads of the system. A table that is used mainly for insert operations it will benefit far less from additional indexes than a table used for data warehouse operations that are 90% read activity.
2. Understand what queries are run most frequently , and optimize your indexes around those queries
3. Understand the data types of the columns used in your queries. Indexes are ideal for integer data types, or unique or non-null columns.
4. Create nonclustered indexes on columns that are frequently used in predicates and join clauses, and keep those indexes as narrow as possible to avoid overhead.
5. Understand your data size /volume – A table scan on a small table will be a relatively cheap operation and SQL may decide to do a table scan simply because it is easy (trivial) to do. A table scan on a very large table would be very costly.

Another option SQL Server provides is the creation of filtered indexes. Filtered indexes are best suited to columns in large tables where a large percentage of the rows have the same value in that column. A practical example would be an employee table as shown below that stored the records of all employees, including ones who had left or retired.

```

CREATE TABLE [HumanResources].[Employee] (
    [BusinessEntityID] [int] NOT NULL,
    [NationalIDNumber] [nvarchar](15) NOT NULL,
    [LoginID] [nvarchar](256) NOT NULL,
    [OrganizationNode] [hierarchyid] NULL,
    [OrganizationLevel] AS ([OrganizationNode].[GetLevel]()),
    [JobTitle] [nvarchar](50) NOT NULL,
    [BirthDate] [date] NOT NULL,
    [MaritalStatus] [nchar](1) NOT NULL,
    [Gender] [nchar](1) NOT NULL,
    [HireDate] [date] NOT NULL,
    [SalariedFlag] [bit] NOT NULL,
    [VacationHours] [smallint] NOT NULL,
    [SickLeaveHours] [smallint] NOT NULL,
    [CurrentFlag] [bit] NOT NULL,
    [rowguid] [uniqueidentifier] ROWGUIDCOL NOT NULL,
    [ModifiedDate] [datetime] NOT NULL" title="" target="_blank" data-generated='>Employee</a>

```

In this table, you will note that there is a column called CurrentFlag, which would indicate if an employee was currently employed. This example uses the bit datatype, indicating only two values, 1 for currently employed and 0 for not currently employed. A filtered index with a WHERE CurrentFlag = 1, on the CurrentFlag column would allow for efficient queries of current employees.

You can also create indexes on views, which can provide significant performance gains when views contain query elements like aggregations and/or table joins.

## Columnstore Indexes

Columnstore indexes were introduced in SQL Server 2012 and offer improved performance for queries that run large aggregation workloads. This type of index was originally targeted at data warehouses, but over time columnstore indexes have been used in a number of other workloads in order to help solve query performance issues on large tables. As of SQL Server 2014, there are both nonclustered and clustered columnstore indexes. Like b-tree indexes, a clustered columnstore index is the table itself stored in a special way, and nonclustered columnstore indexes are stored independently of the table. Clustered columnstore indexes inherently include all the columns in a given table. However, unlike rowstore clustered indexes, clustered columnstore indexes are NOT sorted.

Nonclustered columnstore indexes are typically used in two scenarios, the first is when a column in the table has a data type that is not supported in a columnstore index. Most data types are supported but XML, CLR, sql\_variant, ntext, text, and image are not supported in a columnstore index. Since a clustered columnstore always contains all the columns of the table (because it IS the table), a nonclustered is the only option. The second scenario is a filtered index—this scenario is used in an architecture called hybrid transactional analytic processing (HTAP), where data is being loaded into the underlying table, and at the same time reports are being run on the table. By filtering the index (typically on a date field), this design allows for both good insert and reporting performance.

Columnstore indexes are unique in their storage mechanism, in that each column in the index is stored independently. This offers a two-fold benefit—a query using a columnstore index only needs to scan the columns needed to satisfy the query, reducing the total IO performed, and it allows for greater compression, since data in the same column is likely to be similar in nature.

Columnstore indexes perform best on analytic queries that scan large amounts of data, like fact tables in a data warehouse. Starting with SQL Server 2016 you can augment a columnstore index with an additional b-tree nonclustered index, which can be helpful if some of your queries do lookups against singleton values.

Columnstore indexes also benefit from batch execution mode, which refers to processing a set of rows (typically around 900) at a time versus the database engine processing those rows one at time. Instead of loading each record independently and processing them, the query engine computes the calculation in that group of 900 records. This processing model reduces the number of CPU instructions dramatically. This works best for a query like the following:

```
SELECT SUM(Sales) FROM SalesAmount;
```

Batch mode can provide up to a 10x performance increase over traditional row processing. SQL Server 2019 also includes batch mode for rowstore data. While batch mode for rowstore does not have the same level of read performance as a columnstore index, analytical queries may see up to a 5x performance improvement.

The other benefit columnstore indexes offer to data warehouse workloads is an optimized load path for bulk insert operations of 102,400 rows or more. While 102,400 is the minimum value to load directly into the columnstore, each collection of rows, called a rowgroup, can be up to approximately 1,024,000 rows. Having fewer, but fuller, rowgroups makes your SELECT queries more efficient, because fewer row groups need to be scanned to retrieve the requested records. These loads take place in memory and are directly loaded to the index. For smaller volumes, data is written to a b-tree structure called a delta store, and asynchronously loaded into the index. This is shown in Figure 15.

The screenshot shows a SQL Server Management Studio (SSMS) interface. The top window is titled "SQLQuery3.sql - VM...VM1\jdantoni (52)\*". It contains the following T-SQL code:

```
SET STATISTICS TIME ON

INSERT INTO FactResellerSales_CCI_Demo
SELECT TOP 1024000 *
FROM FactResellerSalesXL_CCI

INSERT INTO FactResellerSales_Page_Demo
SELECT TOP 1024000 *
FROM FactResellerSalesXL_CCI
```

Below this is the "Messages" tab, which displays the execution results:

```
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 6 ms.

SQL Server Execution Times:
    CPU time = 7609 ms, elapsed time = 7902 ms.

(1024000 rows affected)

SQL Server Execution Times:
    CPU time = 17031 ms, elapsed time = 19685 ms.

(1024000 rows affected)

Completion time: 2020-04-22T14:02:07.5526154+00:00
```

Figure 15 Columnstore Index Load Example

In this example, the same data is being loaded into two tables, FactResellerSales\_CCI\_Demo and FactResellerSales\_Page\_Demo. The FactResellerSales\_CCI\_Demo has a clustered columnstore index, and the FactResellerSales\_Page\_Demo has a clustered b-tree index with two columns and is page compressed. As you can see each table is loading 1,024,000 rows from the FaceResellerSalesXL\_CCI table. By using the SET STATISTICS TIME ON option, SQL Server keeps track of the elapsed time of the query execution. Loading the data into the columnstore table took roughly 8 seconds, where loading into the page compressed table took nearly 20 seconds. In this example, all the rows going into the columnstore index are loaded into a single rowgroup.

You should note that if you load fewer rows (less than 102,400) of data into a columnstore index, it is loaded in b-tree structure known as a delta store. The database engine moves this data into the column-

store index using an asynchronous process called the tuple mover. Having open delta stores can impact the performance of your queries, because reading those records is less efficient than reading from the columnstore. You can also reorganize the index with the COMPRESS\_ALL\_ROW\_GROUPS option in order to force the delta stores to be added and compressed into the columnstore indexes.

## Data Compression

As mentioned above, one of the key benefits to columnstore indexes is the ability to compress your data. In addition to columnstore compression, SQL Server offers a few other options for compressing data. While SQL Server still stores compressed data on 8KB pages, when the data is compressed, more rows of data can be stored on a given page, which allows the query to read fewer pages. This has a twofold benefit: it reduces the amount of physical I/O performed and it allows more rows to be stored in the buffer pool, making more efficient use of memory. The tradeoffs to compression are that it does require a small amount of CPU overhead, however in most cases the storage I/O benefits far outweigh any additional processor usage.

The screenshot shows a SQL Server Management Studio window with two tabs: 'SQLQuery6.sql - dca...14 (jdantoni (107))' and 'SQLQuery4.sql - VM...VM1\jdantoni (78)'. Both tabs contain identical T-SQL code:

```
SELECT COUNT(*)
FROM Production.TransactionHistory
WHERE TransactionDate > '2008-01-01'

SELECT COUNT(*)
FROM Production.TransactionHistory_Page
WHERE TransactionDate > '2008-01-01'
```

The results pane shows the execution output:

```
(1 row affected)
Table 'TransactionHistory'. Scan count 1, logical reads 992, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row affected)
Table 'TransactionHistory_Page'. Scan count 1, logical reads 273, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

Completion time: 2020-04-22T14:55:13.3744311+00:00
```

Figure 16 Query against non-compressed and page compressed table

Figure 16 shows this performance benefit —these tables have same underlying indexes; the only difference is that the clustered and nonclustered indexes on the Production.TransactionHistory\_Page table are page compressed. The query against the page compressed object performs 27% fewer logical reads than the query that uses the uncompressed objects.

Compression is implemented in SQL Server at the object level. Each index or table can be compressed individually, and you have the option of compressing partitions within a partitioned table or index. You

can evaluate how much space you will save by using the `sp_estimate_data_compression_savings` system stored procedure. Prior to SQL Server 2019, this procedure did not support columnstore indexes, or columnstore archival compression.

## Row Compression

Row compression is fairly basic and does not incur much overhead; however, it does not offer the same amount of compression (measured by the percentage reduction in storage space required) that page compression may offer. Row compression basically stores each value in each column in a row in the minimum amount of space needed to store that value. It uses a variable-length storage format for numeric data types like integer, float, and decimal, and it stores fixed-length character strings using variable length format.

## Page Compression

Page compression is a superset of row compression, as all pages will initially be row compressed prior to applying the page compression. Then a combination of techniques called prefix and dictionary compression are applied to the data. Prefix compression eliminates redundant data in a single column, storing pointers back to the page header. After that step, dictionary compression searches for repeated values on a page and replaces them with pointers, further reducing storage. Note that the more redundancy in your data, the greater the space savings when you compress your data.

## Columnstore Archival Compression

Columnstore objects are always compressed, however, they can be further compressed using archival compression, which uses the Microsoft XPRESS compression algorithm on the data. This is best used for data that is infrequently read but needs to be retained for regulatory or business reasons. While this data is further compressed, the CPU cost of decompression tends to outweigh any performance gains from IO reduction.

## Summary and Knowledge Check

Database design is a critical component of overall performance. The SQL Server platform offers a number of options for building indexes and compressing data for better performance. Using the proper data types and storage mechanisms can also help the performance of your applications.

### Question 1

*What type of database design should you use for a data warehouse when you want to reduce the data volume of your dimensions?*

- Snowflake schema
- Star Schema
- 3rd normal form

## Question 2

*What is the minimum number rows do you need to bulk insert into a columnstore index?*

- 102,400
- 1,000,000
- 1000

## Question 3

*Which compression type offers the highest level of compression?*

- Columnstore Archival
- Page Compression
- Row Compression

# Evaluating Performance Improvements

## Introduction

One of the challenges of the DBA's role is to evaluate the impact of changes they make to code or data structures on a busy, production system. While tuning a single query in isolation offers easy metrics such as elapsed time or logical reads, making minor tweaks on a busy system may require deeper evaluation. SQL Server and Azure SQL are deeply instrumented software and allow for monitoring at several levels.

## Dynamic Management Views and Functions

SQL Server provides several hundred dynamic management objects. These objects contain system information that can be used to monitor the health of a server instance, diagnose problems, and tune performance. Dynamic management views and functions return internal data about the state of the database or the instance. Dynamic Management Objects can be either views (DMVs) or functions (DMFs), but most people use the acronym DVM to refer to both types of object. There are two levels of DMVs: server scoped and database scoped. Server scoped objects require VIEW SERVER STATE permission on the server, and database scoped objects require the VIEW DATABASE STATE permission within the database. The names of the DMVs are all prefixed with sys.dm\_ followed by the functional area and then the specific function of the object. SQL Server supports three categories of DMVs:

- Database-related dynamic management objects
- Query execution related dynamic management objects
- Transaction related dynamic management objects

You can see a large number of queries to monitor server and database performance on the page in the docs called Monitoring Performance Azure SQL Database Using Dynamic Management Views, which is linked in the additional resources section of this module.. You should note that for older versions of SQL Server (2012 and 2014) where the query store is not available, you can use the view sys.dm\_exec\_cached\_plans in conjunction with the functions sys.dm\_exec\_sql\_text and sys.dm\_exec\_query\_plan to return information about execution plans. However, unlike with Query Store, you will not be able to see changes in plans for a given query. You can read more about getting cached plans from the documentation, as also referenced in the additional resources section of this module.

Azure SQL Database has a slightly different set of the DMVs available than SQL Server; some objects are available only in Azure SQL Database, while other objects are only available in SQL Server. Some are scoped at the server level and are not applicable in the Azure model (the waits\_stats DMV below is an example of this), while others are specific to Azure SQL Database, like sys.dm\_db\_resource\_stats and provide Azure specific information that is not available in (or relevant to) SQL Server.

## Wait Statistics

One holistic way of monitoring server performance is to evaluate what the server is waiting on—SQL Server is instrumented with a wait tracking system, which monitors each running thread and logs what resources the thread is waiting on. Wait statistics are broken down into three types of waits: resource waits, queue waits, and external waits.

- Resource waits occur when a worker thread in SQL Server requests access to a resource that is currently being used by a thread. Examples resources waits are locks, latches, and disk I/O waits.
- Queue waits occur when a worker thread is idle and waiting for work to be assigned. Example queue waits are deadlock monitoring and deleted record cleanup.

- External waits occur when SQL Server is waiting on an external process like a linked server query to complete. An example of an external wait is a network wait related to returning a large result set to a client application.

This data is aggregated in the DMV sys.dm\_os\_wait\_stats (and in sys.dm\_db\_wait\_stats in Azure SQL Database) and is also tracked for active sessions in sys.dm\_exec\_session\_wait\_stats. These statistics (SQL Server tracks over 900 wait types) allow the DBA to get an overview of the performance of the server, and to readily identify configuration or hardware issues. This data is generally persisted from the time of instance startup, but the data can be cleared as needed to observe changes.

Wait statistics are evaluated as a percentage of the total waits on the server as shown in Figure 17.

Wait Type	Wait Percentage	Avg Wait_Sec
1 REDO_THREAD_PENDING_WORK	24.75	0.1016
2 CXPACKET	17.19	0.0021
3 CXCONSUMER	12.24	0.0090
4 PARALLEL_REDO_TRAN_TURN	10.60	0.0029
5 SOS_SCHEDULER_YIELD	10.06	0.0022
6 BPSORT	3.80	0.0126
7 PAGEIOLATCH_SH	3.60	0.0029
8 BACKUPIO	2.01	0.0110
9 HTDELETE	1.83	0.1573
10 LATCH_EX	1.81	0.0047

Figure 17 Top 10 Waits on a SQL Server by Percentage

The results of this query from sys.dm\_os\_wait\_stats shows the wait type, and the aggregation of Percent of time waiting (Wait Percentage) and the average wait time in seconds for each wait type. In this case the server has Always On Availability Groups in place, as indicated by the REDO\_THREAD\_PENDING\_WORK and PARALLEL\_REDO\_TRAN\_TURN wait types. The relatively high percentage of CXPACKET and SOS\_SCHEDULER\_YIELD waits indicates that this server is under some CPU pressure.

There are many wait types that relate to specific and common SQL Server performance issues:

- RESOURCE\_SEMAPHORE waits—this wait type is indicative of queries waiting on memory to become available, and may indicate excessing memory grants to some queries. This is typically observed by long query runtimes or even time outs. These waits can be caused by out of date statistics, missing indexes, and excessive query concurrency.
- LCK\_M\_X waits—frequent occurrences of this wait type can indicate a blocking problem, that can be solved by either changing to the READ COMMITTED SNAPSHOT isolation level, or making changes in indexing to reduce transaction times, or possibly better transaction management within T-SQL code.
- PAGEIOLATCH\_SH waits—this wait type can indicate a problem with indexes (or a lack of useful indexes), where SQL Server is scanning too much data. Alternatively, if the wait count is low, but the wait time is high, it can indicate storage performance problems. You can observe this by analyzing the data in the waiting\_tasks\_count and the wait\_time\_ms in the sys.dm\_os\_wait\_stats DMV, to calculate an average wait time for a given wait type.

- SOS\_SCHEDULER\_YIELD waits—this wait type can indicate high CPU utilization which is generally correlated with either very high number of large scans, or missing indexes, often in conjunction with high numbers of CXPACKET waits as shown above in Figure 17.
- CXPACKET waits—if this wait type is high it can indicate improper configuration. Prior to SQL Server 2019, the Max Degree of Parallelism (MaxDOP) default setting is to use all available CPUs for queries. Additionally, the cost threshold for parallelism (CTfP) setting defaults to 5, which can lead to small queries being executed in parallel, which can limit throughput. Lowering MaxDOP and increasing CTfP can reduce this wait type, but the CXPACKET wait type can also indicate high CPU utilization, which is typically resolved through index tuning.
- PAGEIOLATCH\_UP—This wait type on data pages 2:1:1 can indicate TempDB contention on Page Free Space (PFS) data pages. Each data file has one PFS page per approximately 64MB of data. This wait is typically caused by only having one TempDB file, as prior to SQL Server 2016 the default behavior was to use one data file for TempDB. The best practice is to use one file per CPU core up to eight files. It is also important to ensure your TempDB data files are the same size and have the same autogrowth settings to ensure they are used evenly. SQL Server 2016 and higher control the growth of TempDB data files to ensure they grow in a consistent, simultaneous fashion.

In addition to the aforementioned DMVs, starting with SQL Server 2017, the Query Store tracks waits associated with a given query. This data is not tracked at the same granularity as the data in the DMVs but can provide a nice overview of what a query is waiting on.

## Index Tuning

The most common (and most effective) method for tuning T-SQL queries is to evaluate and adjust your indexing strategy. Properly indexed databases perform fewer IOs to return query results, which reduces pressure on IO and storage, and even allows for better memory utilization. It is also important to keep in mind the read/write ratio of your queries. A heavy write workload may indicate that the cost of writing rows to additional indexes is not of much benefit. An exception would be if the workload performs mainly update operations which also need to do lookup, in which case additional indexes or columns added to an existing index may be beneficial. Your goal should always be to get the most benefit out of the smallest number of indexes on your tables.

A common performance tuning approach is as follows:

1. Evaluate existing index usage using sys.dm\_db\_index\_operational\_stats and sys.dm\_db\_index\_usage\_stats
2. Consider eliminating unused and duplicate indexes—this should be done carefully, as some indexes may only be used during monthly/quarterly/annual operations, and may be important for those processes. You may also consider creating indexes to support those operations just before the operations are scheduled, to reduce the overhead of having otherwise unused indexes on a table.
3. Review and evaluate expensive queries from the Query Store, or Extended Events capture, and work to manually craft indexes to better serve those queries.
4. Create the index(s) in a non-production environment, and test query execution and performance and observe performance changes. It is important to note any hardware differences between your production and non-production environments, as the amount of memory and the number of CPUs could impact your execution plan.
5. After testing carefully, implement the changes to your production system.

You should also verify the column order of your indexes—the leading column drives column statistics and usually determines whether the optimizer will choose the index. Ideally, the leading column will be selective and used in the WHERE clause of many of your queries. This is a good use of a change control

process for tracking changes that could impact application performance. Finally, before dropping indexes, save the code in your source control, so the index can be quickly recreated in the event that an infrequently run query requires the index to perform well.

## Index Maintenance

As data is inserted, updated, and deleted from indexes the logical ordering in the index will no longer match the physical ordering inside of the pages, and between the pages, making up the indexes. Also, over time the data modifications can cause the data to become scattered or fragmented in the database. This fragmentation can degrade query performance when the database engine needs to read additional pages in order to locate needed data.

The SQL Server and Azure SQL platforms offer DMVs that allows you to detect fragmentation in your objects using the following DMVs. The most commonly used DMVs for this purpose are: sys.dm\_db\_index\_physical\_stats, for b-tree indexes, and sys.dm\_dm\_colum\_store\_row\_group\_physical\_stats, for columnstore indexes.

There are two options for index maintenance: reorganization, and rebuilding. Reorganization defragments an index by physically reordering the leaf-level index pages to match the logical sorted order of the leaf nodes and compacts the index pages based on the index's fill factor setting. Rebuilding an index drops and recreates the pages of the index. The typical guidelines for deciding to rebuild versus reorganizing an index is that when fragmentation is greater than 30% that the index should be rebuilt. However, the actual values vary from case to case. If an index is used heavily for scan operations, removing excess pages may help significantly, however there will be limited benefits for seek operations. Reorganization has the benefit of being an online activity, whereas rebuilding requires a special ONLINE option. Indexes that have less than 5% fragmentation do not need to be defragmented, because the cost of defragmentation is greater than the performance benefits that could be gained. Whenever you are considering either rebuilding or reorganizing, you should verify that doing so does provide performance improvements.

SQL Server 2017 introduced resumable index maintenance operations. This option allows index maintenance operations to be paused, or take place in a time window, and be resumed later. A good example of where to use resumable index operations is to reduce the impact of index maintenance in a busy production environment, as you can perform rebuild operations during a specific maintenance window giving you more control over the process.

One other thing to note is that index rebuilds cause the statistics on the index to be updated, which can further help performance. Index reorganization does not update statistics.

## Columnstore Maintenance

Columnstore index fragmentation is not reported in same manner as b-tree index fragmentation. You should examine the sys.dm\_dm\_colum\_store\_row\_group\_physical\_stats for deleted rows—which will be caused by both update and delete operations. A good metric to measure fragmentation in a columnstore index is based on the deleted rows in the index. If 20% or more of the rows are deleted, you should consider reorganizing your index. In a test environment, verify whether a reorganization improves the performance of the workloads using the columnstore index.

## Summary and Knowledge Check

### Question 1

*What type of index is best used on a data warehouse fact table?*

- Clustered Columnstore
- Nonclustered Columnstore
- Clustered b-tree

### Question 2

*Which DMV provides information about server level wait statistics?*

- sys.dm\_db\_index\_physical\_stats
- sys.dm\_os\_wait\_stats
- sys.dm\_exec\_session\_wait\_stats

### Question 3

*Which DMV can you use to capture the last Actual Execution Plan for a given query?*

- sys.dm\_exec\_cached\_plans
- sys.dm\_exec\_query\_plan
- sys.dm\_exec\_query\_plan\_stats

## Module Summary

### Module Summary

Database performance tuning is huge topic. The basis of understanding performance starts with database design, and is then strongly tied to query execution. Being able to understand execution plans and how they impact performance can help you address both design and indexing issues in your database. Proper indexing will allow your databases to operate efficiently. Wait statistics allow you to holistically look at a server and identify potential performance bottlenecks, which helps you isolate your performance troubleshooting efforts.

### Additional Resources

Monitoring performance Azure SQL Database using dynamic management view

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-monitoring-with-dmvs>

Documentation for sys.dm\_os\_wait\_stats and database wait types

<https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-wait-stats-transact-sql?view=sql-server-ver15>

Columnstore Index Fragmentation Guidance

<https://docs.microsoft.com/en-us/sql/relational-databases/indexes/reorganize-and-rebuild-indexes?view=sql-server-ver15#considerations-specific-to-reorganizing-a-columnstore-index>

SQL Server Index Architecture and Design Guide

<https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver15>

Xpress Compression Algorithm

[https://docs.microsoft.com/en-us/openspecs/windows\\_protocols/ms-xca/a8b7cb0a-92a6-4187-a23b-5e14273b96f8](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-xca/a8b7cb0a-92a6-4187-a23b-5e14273b96f8)

# Answers

## Question 1

Which type of execution plan is stored in the plan cache?

- Estimated execution plan
- Actual execution plan
- Live Query Stats

*Explanation*

*The estimated execution plan is stored in the plan cache*

## Question 2

Which DMV should you use to find index utilization?

- sys.dm\_db\_index\_usage\_stats
- sys.dm\_db\_missing\_index\_details
- sys.dm\_exec\_query\_plan\_stats

*Explanation*

*sys.dm\_db\_index\_usage\_stats shows the read and write operations against each index*

## Question 3

Which of the following wait types would indicate excessive CPU consumption?

- SOS\_SCHEDULER\_YIELD
- RESOURCE\_SEMAPHORE
- PAGEIOLATCH\_SH

*Explanation*

*The SOS\_SCHEDULER\_YIELD wait is the only one of these wait types that is associated with CPU*

## Question 1

What type of database design should you use for a data warehouse when you want to reduce the data volume of your dimensions?

- Snowflake schema
- Star Schema
- 3rd normal form

*Explanation*

*Snowflake schema would reduce the data volume*

**Question 2**

What is the minimum number rows do you need to bulk insert into a columnstore index?

- 102,400
- 1,000,000
- 1000

*Explanation*

*102,400 is the minimum number of rows to bulk insert into a columnstore index, fewer rows will use a normal insert operation*

**Question 3**

Which compression type offers the highest level of compression?

- Columnstore Archival
- Page Compression
- Row Compression

*Explanation*

*Columnstore archival provides the highest level of compression in SQL Server.*

**Question 1**

What type of index is best used on a data warehouse fact table?

- Clustered Columnstore
- Nonclustered Columnstore
- Clustered b-tree

*Explanation*

*A clustered columnstore index will provide the best performance for a data warehouse fact table*

**Question 2**

Which DMV provides information about server level wait statistics?

- sys.dm\_db\_index\_physical\_stats
- sys.dm\_os\_wait\_stats
- sys.dm\_exec\_session\_wait\_stats

*Explanation*

*A clustered columnstore index will provide the best performance for a data warehouse fact table*

**Question 3**

Which DMV can you use to capture the last Actual Execution Plan for a given query?

- sys.dm\_exec\_cached\_plans
- sys.dm\_exec\_query\_plan
- sys.dm\_exec\_query\_plan\_stats

*Explanation*

*In SQL Server 2019 you can query sys.dm\_exec\_query\_plan\_stats to get the last actual execution plan for a query.*

# Module 6 Automation of Tasks

## Module Introduction

### Module Introduction

A common goal for database administrators in many environments is to automate as many of their repetitive tasks. This can be as simple as using scripting to automate a backup process, and as complex as building a fully automated alerting system. This module provides details of automating tasks to simplify the DBA's job. Methods include scheduling tasks for regular maintenance jobs, as well as multi-instance administration and configuration of notifications for task success or failure or non-completion.

### Learning Objectives

At the completion of this module, you will be able to:

- Deploy resources using automated deployment scripts
- Create scheduled tasks
- Create notifications and alerts
- Configure automation for PaaS services

# Setting up Automatic Deployment

## Introduction

Unlike the on-premises world, which requires cabling and racking of hardware in order to deploy a new database server, one of the major benefits of cloud computing is that system resources are abstracted behind an API. In the case of Azure, this deployment and management layer is called Azure Resource Manager (ARM). ARM offers a consistent deployment mechanism called ARM templates which are JavaScript Object Notation (JSON) documents that can be used for parameterized deployment.

## Learning Objectives

After this lesson you will understand:

- How to deploy an ARM template using PowerShell and CLI
- How to deploy an ARM Template using Azure DevOps
- The structure of an ARM template

## Automatic Deployment of Azure Resources

Azure ARM Templates have the benefit of being able to deploy a full set of resources in one single declarative template. This includes the ability to build dependencies into the templates, as well as using parameters to change deployment values at deployment time. Once you have a template, you can deploy it several ways including using an Azure DevOps pipeline, or through the custom deployments blade in the Azure Portal. The benefit of these deployments is that they use a declarative model which defines what should be created and then the ARM framework determines how to deploy it. The alternative to the declarative model is the imperative model. Imperative frameworks include PowerShell and the Azure CLI which follow a prescriptive order of tasks to be executed.

## Declarative and Imperative Models?

A common term used around cloud computing deployments is “infrastructure as code” which means all of your resources are defined a set of scripts that are stored in source control and can easily be deployed to a new environment. While stateful resources like databases are not deployed as frequently as application code, by defining your infrastructure, you ensure that resources are deployed in a consistent fashion, reducing the configuration risk and the impact of human error. As mentioned above, there are two programming models that are used for cloud deployments: imperative and declarative.

In imperative programming you are defining a set of prescriptive tasks for the target system to execute. A simple example of this model is using a batch script to install SQL Server and its prerequisites. In declarative programming you are describing a set of resources to be defined, typically by a framework. A simple example of this would be a CREATE TABLE statement which describes the columns and keys to be built by the SQL Server engine which acts as the framework for building the table.

## *ARM Templates*

ARM templates allow you to create and deploy an entire infrastructure in a declarative framework. For example, you can deploy not only a virtual machine, but its network and storage dependencies in one document. Resource manager also supports orchestration, which manages the deployment of interdependent resources so that they are created in the correct order. For example, a VM is dependent on the

existence of a virtual network, so the framework will deploy the network (or check for the existence of the network) before attempting to build the VM. ARM templates also support extensibility, which allow you to run PowerShell or Bash scripts on your resources after they are deployed.

## PowerShell

PowerShell provides a core module known as Az, which has child resource providers for nearly all Azure services. For example, Az.Compute would cover Azure Virtual Machines. PowerShell is more commonly used for resource modification and status retrieval. While it is possible to create resources using PowerShell, it is not typically used for complex deployments. PowerShell can also be used to deploy ARM templates, so in a sense it supports both declarative and imperative models.

## Azure CLI

The Azure Command Line Interface, or CLI, is similar to PowerShell in that it can be used either imperatively or declaratively. Much like PowerShell and ARM templates, the Azure CLI provides a mechanism to deploy or modify Azure Resources. You should note that some commands for Azure PostgreSQL and Azure MySQL Databases are only available in the Azure CLI.

## Azure Portal

The Azure Portal is a graphical interface to Azure ARM. Any resources you build and deploy using the Portal will have an ARM template that you can capture by clicking "Export Template" in the Settings Blade, as shown in Figure 1.

```

1 {
2   "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3   "contentVersion": "1.0.0.6",
4   "parameters": {
5     "virtualMachine_Win1_name": {
6       "defaultValue": "Win1",
7       "type": "String"
8     },
9     "disk_Win1_DataDisk_1_externalId": {
10       "defaultValue": "subscriptions/424d8f78-5900-4c31-98ec-624616db8e74/resourceGroups/BookDemo/providers/Microsoft.Compute/disks/win1_DataDisk_1",
11       "type": "String"
12     },
13     "disk_Win1_DataDisk_2_externalId": {
14       "defaultValue": "subscriptions/424d8f78-5900-4c31-98ec-624616db8e74/resourceGroups/BookDemo/providers/Microsoft.Compute/disks/win1_DataDisk_2",
15       "type": "String"
16     },
17     "disk_Win1_DataDisk_3_externalId": {
18       "defaultValue": "subscriptions/424d8f78-5900-4c31-98ec-624616db8e74/resourceGroups/BookDemo/providers/Microsoft.Compute/disks/win1_DataDisk_3",
19       "type": "String"
20     },
21     "networkInterface_win1222_externalId": {
22       "defaultValue": "subscriptions/424d8f78-5900-4c31-98ec-624616db8e74/resourceGroups/BookDemo/providers/Microsoft.Network/networkInterfaces/win1222",
23       "type": "String"
24     }
25   },
26   "variables": {},
27   "resources": [
28     {
29       "type": "Microsoft.Compute/virtualMachines",
30       "name": "Win1"
31     }
32   ]
33 }

```

Figure 1 Azure Portal Generated Deployment Template

The Azure Portal is usually the easiest way to get started when first learning about the Azure platform. Organizations (and DBAs) typically move into a more automated model of deployment as their Azure estate and experience grows.

## *Deploying an Azure ARM Template—PowerShell and CLI*

With PowerShell you have several options for the scope of your deployment. You can deploy to a resource group, a subscription, a Management Group (a collection of subscriptions under the same Azure template and commonly used in large enterprise deployments), or a tenant. ARM templates are parameterized, and you will need to pass in parameters, either inline or through the use of a parameter file as shown in the example below.

```
New-AzResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName  
ExampleResourceGroup  
  
-TemplateFile c:\MyTemplates\azuredeploy.json  
  
-TemplateParameterFile c:\MyTemplates\storage.parameters.json
```

The parameter and template file can also be stored in a git repo, Azure Blob Storage, or any other place where it is accessible from the deploying machine.

## *Deploying an ARM Template with Azure CLI*

The Azure CLI allows the same options for deployment scope as you have with PowerShell. Also, like with PowerShell, you can use a local or remote parameter file and template, as shown in the example below.

```
az deployment group create --resource-group SampleRG --template-file '\path\template.json'
```

## *Using Azure DevOps to Deploy Templates*

In Azure DevOps, deployments are carried out using Azure Pipelines. Azure Pipelines are a fully featured continuous integration and continuous delivery service (CI/CD), which allows you to automate the build, testing, and deployment of your code. You can deploy Azure resources using ARM templates in two ways... The first method calls a PowerShell script as shown above. The second approach defines tasks which stage your artifacts (the templates themselves and any required secrets) and then deploys the templates. One task stages the artifacts and the other tasks deploys the templates.

## **Continuous Integration and Continuous Delivery (CI/CD)**

Continuous integration is a development methodology that focused on making small changes to code and frequent code check-ins to the version control system. This provides an automated way to build, package, and test applications. Having the framework in place facilitates more frequent check-ins, and allows better collaboration between developers, with the goal of improving code quality. Continuous delivery builds on the continuous integration and automates the delivery of code changes to the underlying infrastructure.

## **Creating an Azure ARM Template**

An ARM template is a JSON (JavaScript Object Notation) document that describes the resources that will be deployed within an Azure Resource Group. The structure of these templates is shown below.

```
{  
  
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploy-  
  mentTemplate.json#",  
  
  "contentVersion": "",  
  
  "apiProfile": "",  
  
  "parameters": { },  
  
  "variables": { },  
  
  "functions": [ ],  
  
  "resources": [ ],  
  
  "outputs": { }  
  
}
```

The schema file, which is referenced on the first line the above example describes the version of the template language. This file is supplied by Microsoft to define the Azure API. The content version number is only defined by you as needed, and should align with your internal versioning standard. The API profile serves as a collection of API versions for this environment. This is especially useful when deploying resources to different environments like national clouds and commercial clouds, which may have differing resource providers. Parameters are values that are provided to the template in order to customize resources at deployment time, whereas variables are values that are used as JSON fragments to simplify template language expressions. You can also include user defined functions within the template. The resources component defines what resources are getting defined in the template. The resources are the resource types that you are deploying. Exhibit 1 shows the example resources section from an ARM template to deploy an Azure SQL Database.

```
"resources": [  
  
  {  
  
    "name": "[variables('sqlServerName')]",  
  
    "type": "Microsoft.Sql/servers",  
  
    "apiVersion": "2014-04-01-preview",  
  
    "location": "[parameters('location')]",  
  
    "tags": {  
  
      "displayName": "SqlServer"  
  
    },  
  
  },  
  
]
```

```
"properties": {  
  
    "administratorLogin": "[parameters('sqlAdministratorLogin')]",  
  
    "administratorLoginPassword": "[parameters('sqlAdministratorLoginPass-  
word')]",  
  
    "version": "12.0"  
  
},  
  
"resources": [  
  
{  
  
    "name": "[variables('databaseName')]",  
  
    "type": "databases",  
  
    "apiVersion": "2015-01-01",  
  
    "location": "[parameters('location')]",  
  
    "tags": {  
  
        "displayName": "Database"  
  
    },  
  
    "properties": {  
  
        "edition": "[variables('databaseEdition')]",  
  
        "collation": "[variables('databaseCollation')]",  
  
        "requestedServiceObjectiveName": "[variables('databaseServiceObjective-  
Name')]"  
  
    },  
  
    "dependsOn": [  
  
        "[variables('sqlServerName')]"  
  
    ],  
  
    "resources": [  
  
{  
  
        "comments": "Transparent Data Encryption",
```

```
"name": "current",

"type": "transparentDataEncryption",

"apiVersion": "2014-04-01-preview",

"properties": {

    "status": "[parameters('transparentDataEncryption')]"}

    } , 

    "dependsOn": [
        "[variables('databaseName')]"
    ]
}

]
}
```

*Exhibit 1: The example resources section from an ARM template to deploy an Azure SQL Database*

One of the most important things to note about the structure of the resources section of the above template is the **dependsOn** option—this allows you to build a dependency structure into your template. A simple example of this is in the template above—the Azure SQL Database is dependent on the existence of an Azure SQL Database server, and Transparent Data Encryption is dependent on the existence of the database. This can be used to build even more complex dependency structures for complex application deployments.

It is important to understand the difference between variables and parameters. Parameters can accept values from outside the template by user interaction, a file, or CI/CD pipelines. Variables can be defined within the template and are used for simplification, since you can define a complicated expression once, and then use it throughout the template. An example of variable definition in an ARM template is shown in Exhibit 2 below.

```
"variables": {

    "storageName": "[concat(toLower(parameters('storageNamePrefix')), uniqueString(resourceGroup().id))]"}
```

*Exhibit 2 Example variable for storageName*

By defining this expression in the variables section of your template, you can reuse it by referencing the variable in other parts of the template in the following manner: "[variables('storageName;')]". Variables let you avoid repeating a complicated expression throughout your template, by using the variable name.

## Source Control for Templates

ARM Templates are an example of infrastructure as code. Since all hardware resources are abstracted behind a set of APIs, this allows for your entire infrastructure to be another component of your application code. Just like application or database code, it is important to protect and version this code. In addition to the internal version in the template, your source control system should version your templates. You can also configure your templates to be automatically deployed from GitHub. As shown in Figure 2.

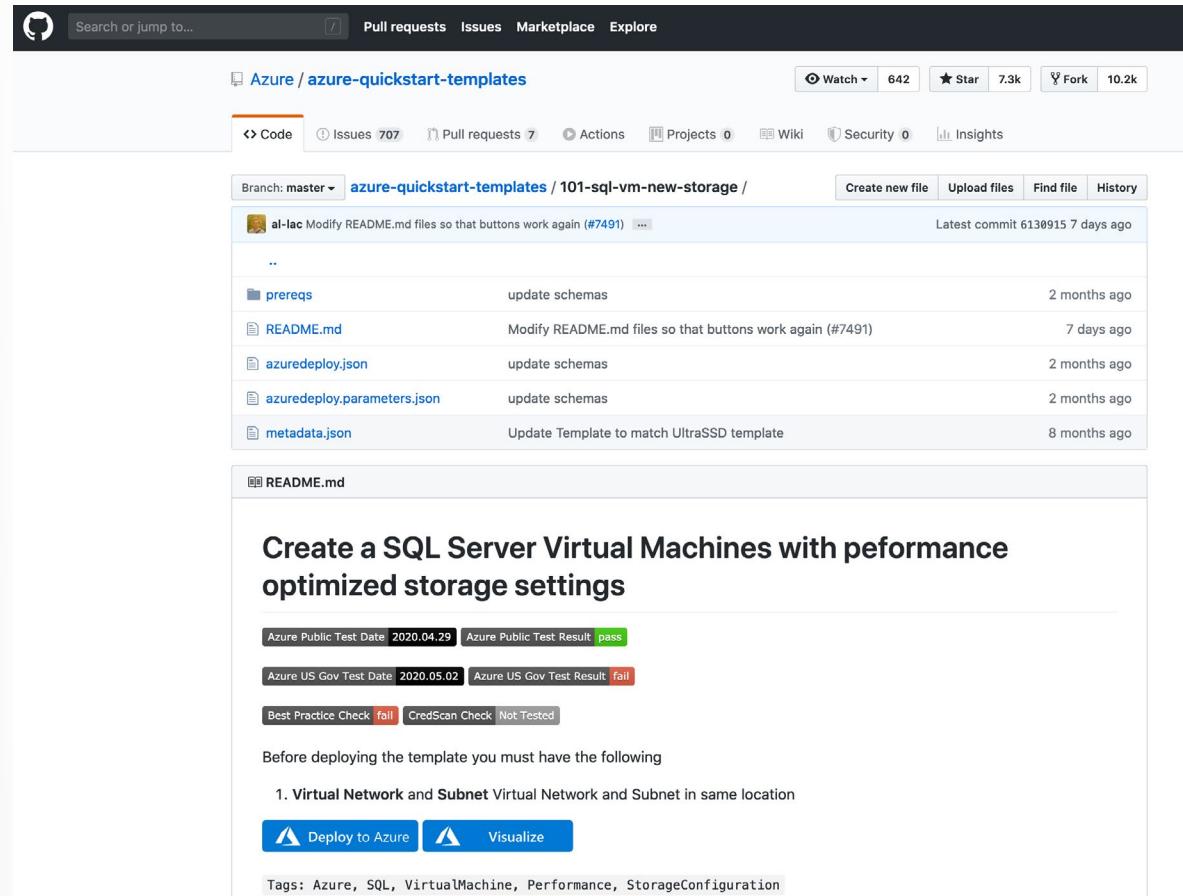


Figure 2 GitHub Azure Quickstart Template

## Deploying an Azure ARM Template from the Portal

In most cases the database administrator will not be writing their own ARM templates from scratch. You may either build them from the Azure Portal or using a template from the quick start templates that are provided by Microsoft on GitHub. If you click on the deploy to GitHub button you will login to the Azure portal and be presented with the following screen, shown in Figure 3.

The screenshot shows the Microsoft Azure ARM Deployment screen for a "SQL Server VM with performance optimized storage settings" quickstart template. The template contains 5 resources. The deployment form includes fields for Subscription (Contoso Ltd), Resource group (Create new), and Location ((US) East US 2). Under SETTINGS, the Virtual Machine Name is set to "myVM", Size to "Standard\_DS13\_v2", and Workload Type to "General". Storage parameters include 1 data disk at path F:\SQLData and 1 log disk at path G:\SQLLog. The Admin Password field is empty. A "Purchase" button is visible at the bottom.

*Figure 3 Azure ARM Deployment screen*

The template parameters are defined by the parameters section of the template file. In this deployment example, there are only two VM sizes defined (the default value is a Standard\_DS13\_v2).

If you click the edit template button, you will see the JSON defining the template, which would allow you to change the values to meet your requirements.

```
"virtualMachineSize": {

    "type": "String",

    "defaultValue": "Standard_DS13_v2",
```

```
"allowedValues": [  
    "Standard_DS13_v2",  
    "Standard_DS3_v2"  
,  
    "metadata": {  
        "description": "The virtual machine size."  
    }  
}
```

After filling in the required parameters in the deployment screen, you click Purchase and deploy your template. You will perform this activity in the lab for this module.

## Summary and Knowledge Check

The Azure platform provides many options for deploying resources. You have the option to deploy either from Azure DevOps or through command line scripting options. ARM templates provide you the flexibility to granularly deploy Azure resources in a repeatable consistent fashion. It is considered a best practice to manage your infrastructure as code and implement source control for it. This has the side benefit of providing more consistent deployments.

### Question 1

*What language are ARM templates written in?*

- JSON
- C#
- T-SQL

### Question 2

*If you want to pass in the region for a resource group deployment which option should you include in your template?*

- Parameter
- Variable
- Output

## Question 3

*Which element of a template allows for you to build dependencies into resources?*

- dependsOn
- concat
- apiVersion

# Defining Scheduled Tasks with the SQL Server Agent

## Introduction

Database systems need regular maintenance which includes tasks like making backups and updating statistics. Maintenance may also include regularly scheduled jobs that execute against a database. Some common examples of these jobs would be to extract, transform, and load data from a transaction processing system into a data warehouse. In SQL Server and Azure SQL Managed Instance, the SQL Server Agent service allows you to schedule jobs to perform these maintenance tasks (as well as providing other management functions). For Azure SQL Database and Azure Database for MariaDB/MySQL/Postgres, there are other options available for scheduling maintenance operations, including Azure Automation, which you will learn about below.

After this lesson you will understand:

- What Maintenance Activities you should perform on your databases
- How to configure notifications and alerts on SQL Server Agent jobs and SQL Server
- How to configure notifications alerts based on performance monitor values

## SQL Server Maintenance

Typical activities that you can schedule for regular SQL Server maintenance include:

- Database and transaction log backups
- Database Consistency Checks
- Index maintenance
- Statistics updates

You learned about the importance of backups and index/statistics maintenance in modules 4 and 5. Database consistency checks, also known as CHECKDB (for the command DBCC CHECKDB) are of equal importance, as it is the only way to check an entire database for corruption. Depending on the size of your databases and your uptime requirements, you may perform all of these activities nightly. More commonly in production systems, the maintenance operations are spread out over the course of week, as both index maintenance and consistency checks are very I/O intensive operations and they are typically done during weekend hours. Similarly, many DBAs stagger backups of large databases, and only do one full backup a week. Differential and transaction log backups can then be used to manage point in time recovery, as will be discussed in Module 7. SQL Server offers a built-in way to manage all of these tasks using Maintenance Plans. Maintenance plans create a workflow of the tasks to support your databases. Maintenance plans are created as Integration Services packages which allow you to schedule your maintenance activities. Many DBAs also use open source scripts to perform database maintenance, to allow for more flexibility and control of maintenance activities.

## *Best Practices for Maintenance Plans*

In addition to allowing you to perform database maintenance, maintenance plans provide options to allow you to prune data from the msdb database, which acts as the data store for the SQL Server Agent. Maintenance plans also allow you to specify that older database backups should be removed from disk. This helps your SQL Server by reducing the size of your backup volume and helps manage the size of the msdb database. You should ensure that your backup retention period is longer than your consistency

check window. This means if you run a consistency check weekly, you should retain eight days of backups. (Note: The backup operation will not detect corruption in a database, so it is possible to have corruption within a backup file). Maintenance plan activities are scheduled as SQL Server Agent jobs for execution.

## *Creating a Maintenance Plan*

You can create a maintenance plan using SQL Server Management Studio as shown below, starting with Figure 2. You should note that in the example in Figure 2, multiple maintenance tasks are combined in one maintenance plan. The best practice would be to create a maintenance plan for each type of task—and possibly even for a specific database on your server. For example, you might create a maintenance plan to backup system databases and another maintenance plan to backup user databases. You could also have another maintenance plan for special handling of the backup of one very large user database. Figure 2 and the following examples show the creation a maintenance plan using the maintenance plan wizard.

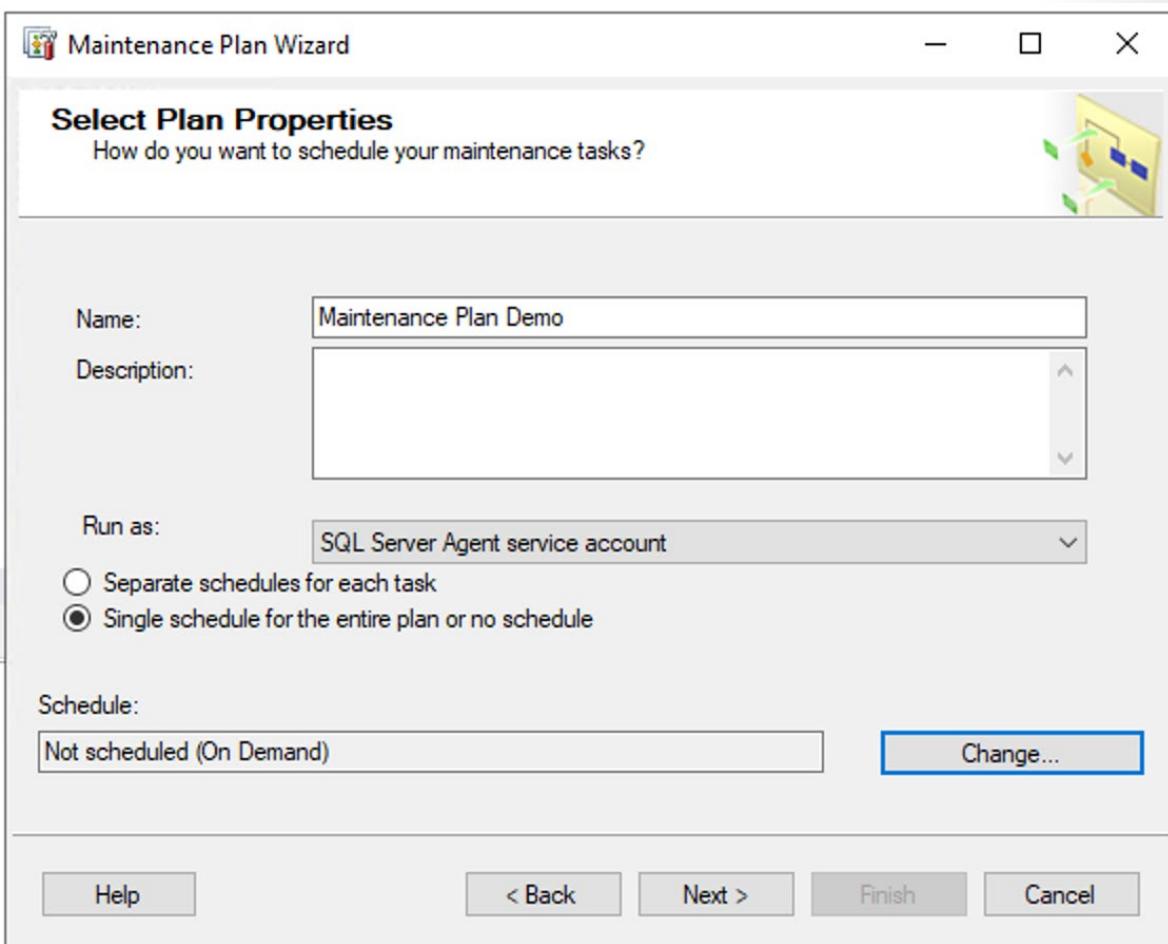


Figure 4 Maintenance Plan Wizard

Figure 2 shows the first screen of the maintenance plan wizard from SQL Server Management Studio (SSMS). There are a couple of things you should note—you need to specify a name for your maintenance plan, and you need to specify a run-as account. Typically, most maintenance operations will run as the SQL Server Agent service account, but you may need to have a task run as a different account for security purposes. For example, if you need to backup to a file share that only a specific account has access to,

you would need to run that specific plan as a different user. This concept is known a proxy user and is another component of the SQL Server Agent.

## What is a Proxy Account?

A proxy account is an account with stored credentials that can be used by the SQL Server Agent to execute steps of a job as a specific user. The login information for this user is stored as a credential in the SQL Server instance. Proxy accounts are typically used when very granular security rights are needed for specific steps of a SQL agent job.

You also define a schedule for this job—schedules are a component of the job system in the msdb system database. SQL Server Agent jobs and schedules have a many to many relationship. Each job can contain multiple schedules, and the same schedule can be assigned to multiple jobs. The maintenance plan wizard, however, does not allow creation of independent schedules. It creates a specific schedule for each maintenance plan, as shown below in Figure 3.

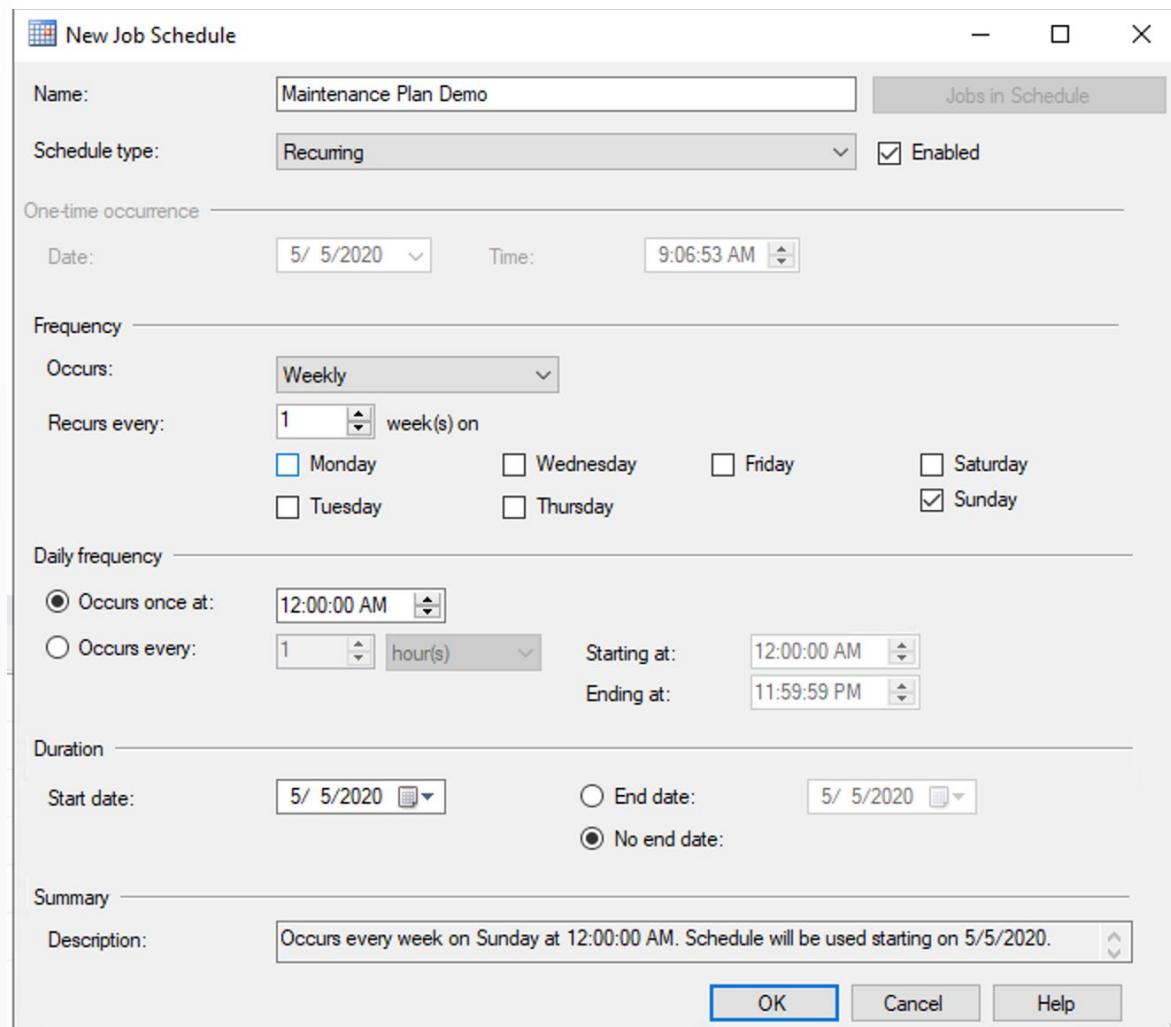


Figure 5 Job Schedule Creation in Maintenance Plan Wizard

The above schedule is for a weekly execution, but you also have the option to create a schedule with hourly or daily recurrence. The next step in this process is to add maintenance tasks to the plan.

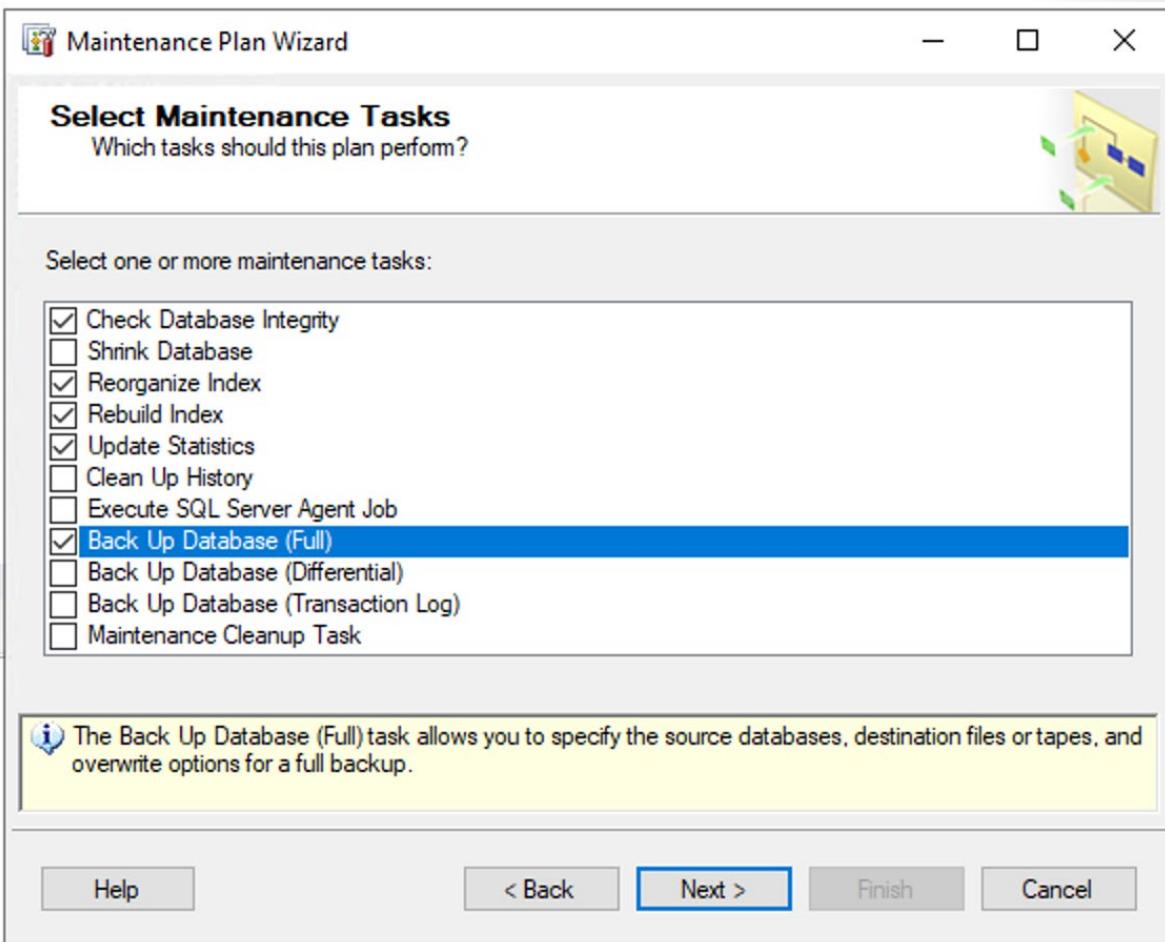


Figure 6 Maintenance Plan Tasks Addition Screen

Figure 4 shows the maintenance tasks addition screen. This is where you choose the operations to be performed by your maintenance plan. The options are:

**Check Database integrity**—This task executes the DBCC CHECKDB commands which validates the contents of each database page to ensure its logical and physical consistency. This task should be performed on a regular basis (daily or weekly), and it should align to your backup retention window. To ensure corruption is not carried over to your backups, make sure you are successfully completing a consistency check before discarding any prior backups.

**Shrink Database:** This task reduces the size of database or transaction log file by moving data into free space on pages. When enough space is consumed the free space can be returned to the file system.

**Note: It is recommended that you never execute this action as part of any regular maintenance as it leads to severe index fragmentation which can harm database performance. The operation itself is also very I/O and CPU intensive and can severely impact your system performance.**

\*\*Reorganize/Rebuild Index—\*\*These operations have been discussed in modules 4 and 5. To review, this task will check the level of fragmentation in a database's indexes, and may either rebuild or reorganize the index based on the user-defined level of fragmentation. You should note that rebuilding an index updates the statistics on the index.

\*\*Update Statistics—\*\*This task updates the column and index statistics that are used by SQL Server to build query execution plans. It is important that the statistics accurately reflect the data stored in tables

so that the query optimizer can make the best decisions in building execution plans. This task allows you to choose which tables and indexes are scanned, and the percentage or number of rows scanned. The default sampling rate is acceptable for most objects, though you may wish to capture more detailed statistics for specific tables.

**\*\*Cleanup History—**\*\*This task deletes history of backup and restore operations from the msdb database as well the history of SQL Server agent jobs. This task is used to manage the size of the msdb database.

**Execute SQL Server Agent Job**—This task is used to execute a user-defined SQL Server Agent job.

**\*\*Backup Database (Full/Differential/Log)—**\*\*This task is used to backup databases on a SQL Server instance. A full backup backs up the entire database, and serves as the starting point for a restore (you need a full backup in order to completely restore a database). Differential backups backup the pages in the database that have changed since the last full backup, and are typically used to provide an incremental restore point. Transaction log backups backup the active pages in your transaction log, and allow you to define your recovery point objective. Transaction log backups cannot be performed on databases in SIMPLE recovery mode.

Here's an example of the use of different kinds of backups: If you took a full backup on Sunday, and a differential each weeknight, and you wanted to restore to your database to noon on Thursday, you would only need to restore Sunday's full backup and Wednesday's differential, followed by the transaction log backups from the point of Wednesday's differential backup until Thursday at noon. You will learn more about backup and restore in Module 6.

**\*\*Maintenance Cleanup Tasks—**\*\*This task removes old files related to maintenance plans, including text reports from maintenance plan execution, and backup files. It only removes backups on files in the folders specified, so any subfolders must be specifically listed or they will be skipped.

Each task has a scope of user databases, system databases, or a custom selection of databases. Additionally, each task has its own specific configuration options.

Once you finish creating the Maintenance Plan, you will be presented with the details of the entire plan, as shown in Figure 5. You can get back to this view in SQL Server Management Studio by expanding the Management node, then expanding the Maintenance Plans node, right clicking on this Maintenance Plan and selecting Modify.

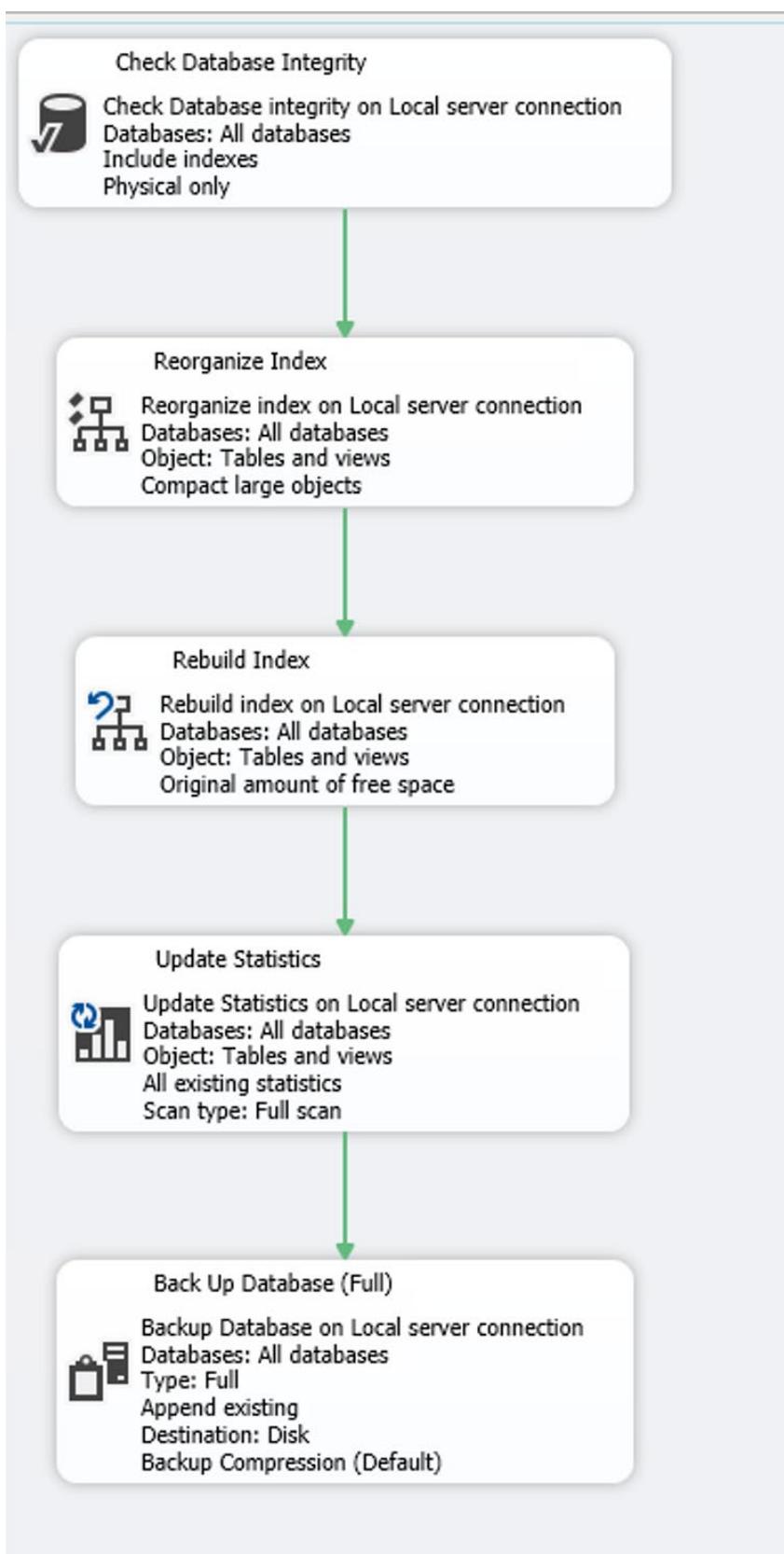


Figure 7 Complete Maintenance Plan in SSMS

Upon creation the plan will appear as a job in the SQL Server Agent. If you added a schedule either during the creation process or after, that job will be executed and the maintenance tasks will be performed.

## Multi-Server Automation

In a multi-server environment, the SQL Server Agent provides the option of designating one server as a master server that can execute jobs on other servers, designated as target servers. The master server stores a master copy of the jobs and distributes the jobs to the target servers. Target servers connect to the master server periodically to update their schedule of jobs. This allows you to define one job and deploy it across your enterprise. A good example of this would be configuring database maintenance across your environment. You could create a set of maintenance plan tasks just once and allow them to be pushed out to group of target servers, to ensure consistent deployment.

## Task Status Notifications

One important part of automation is providing notifications in the event of job failure or if certain system errors are encountered. SQL Server Agent provides this functionality through a group of objects. Alerting is most commonly done via email using the Database Mail functionality of SQL Server. The other agent objects that are used in this workflow are:

- Operators—alias for people or group who receive notifications
- Notifications—notify an operator of the completion, success or failure of a job.
- Alerts—are assigned to an operator, for either a notification or a defined error condition

### *Operators*

Operators act an alias for a user or group of users that have been configured to receive notifications of job completion, or to be informed of alerts have been sent to the error log. An operator is defined as an operator name and contact information. Typically, an operator will map to a group of people using an email group. Having multiple people in the email group provides redundancy so that a notification is not missed if someone is unavailable. Groups are also beneficial if an employee leaves the organization; the single person can be removed from the email group and you do not have to update all of your instances. To send email to an operator you need to enable the email profile of the SQL Server Agent as shown in Figure 6 below.

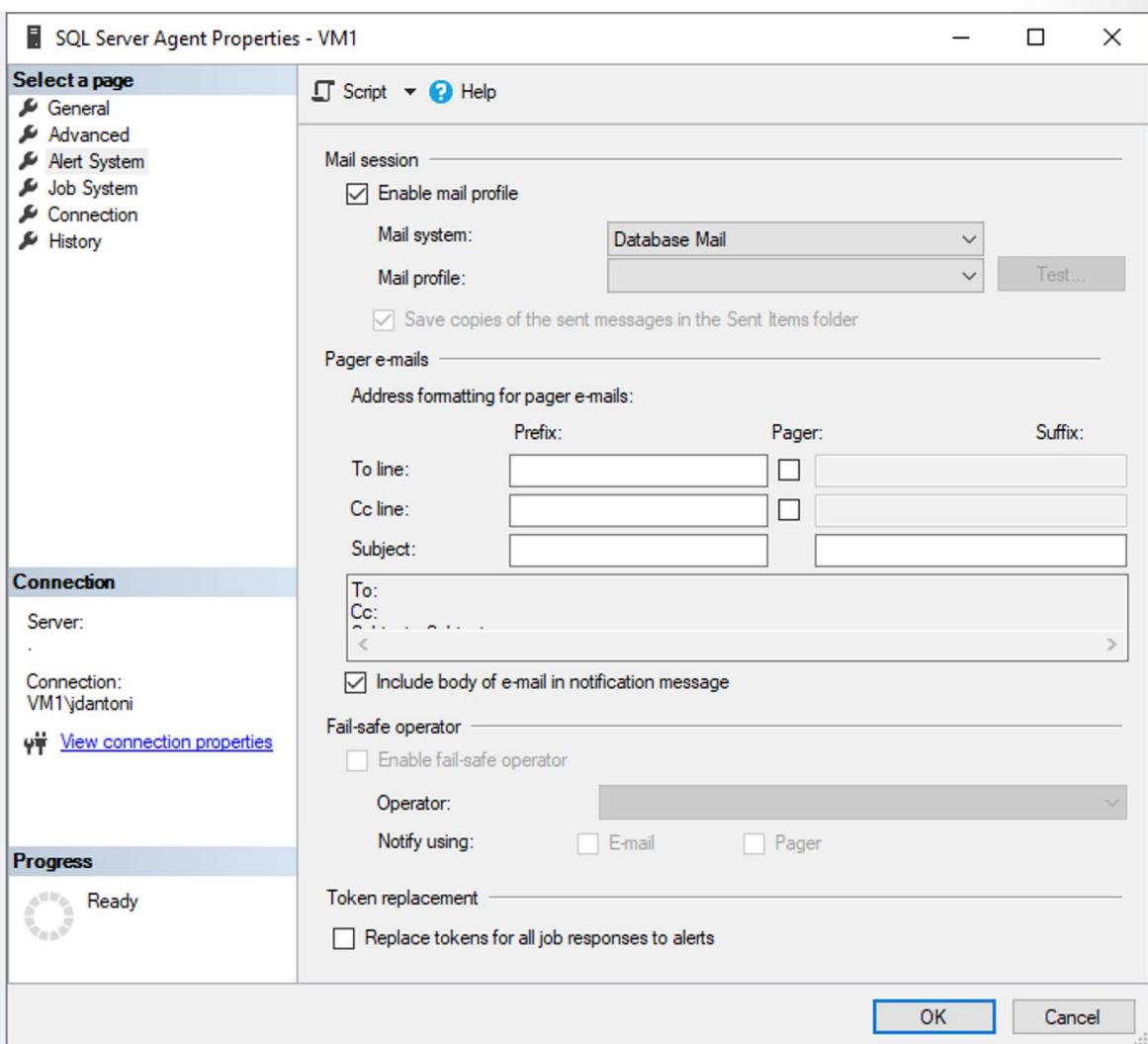


Figure 8 SQL Server Agent Mail Profile Configuration

## Notifications

Notification of completion is part of each SQL Server Agent job. You have the option of sending a notification on Job completion, failure, or success. Most DBAs notify on failure only, to avoid an influx of notifications for successful jobs. Notifications have a dependency on an operator existing in order to send a notification as shown in Figure 7.

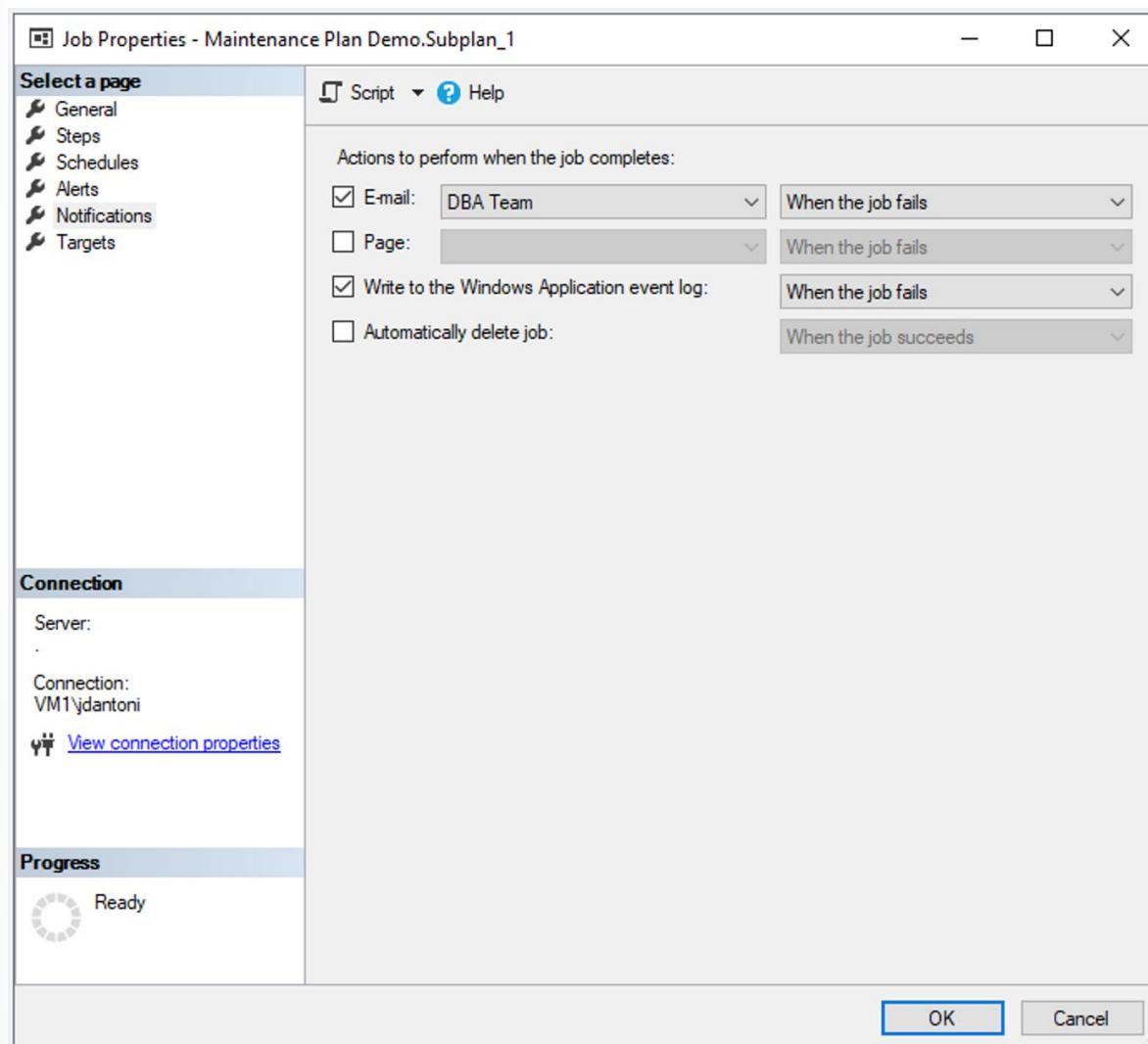


Figure 9 Assigning a Notification recipient for Job Failure

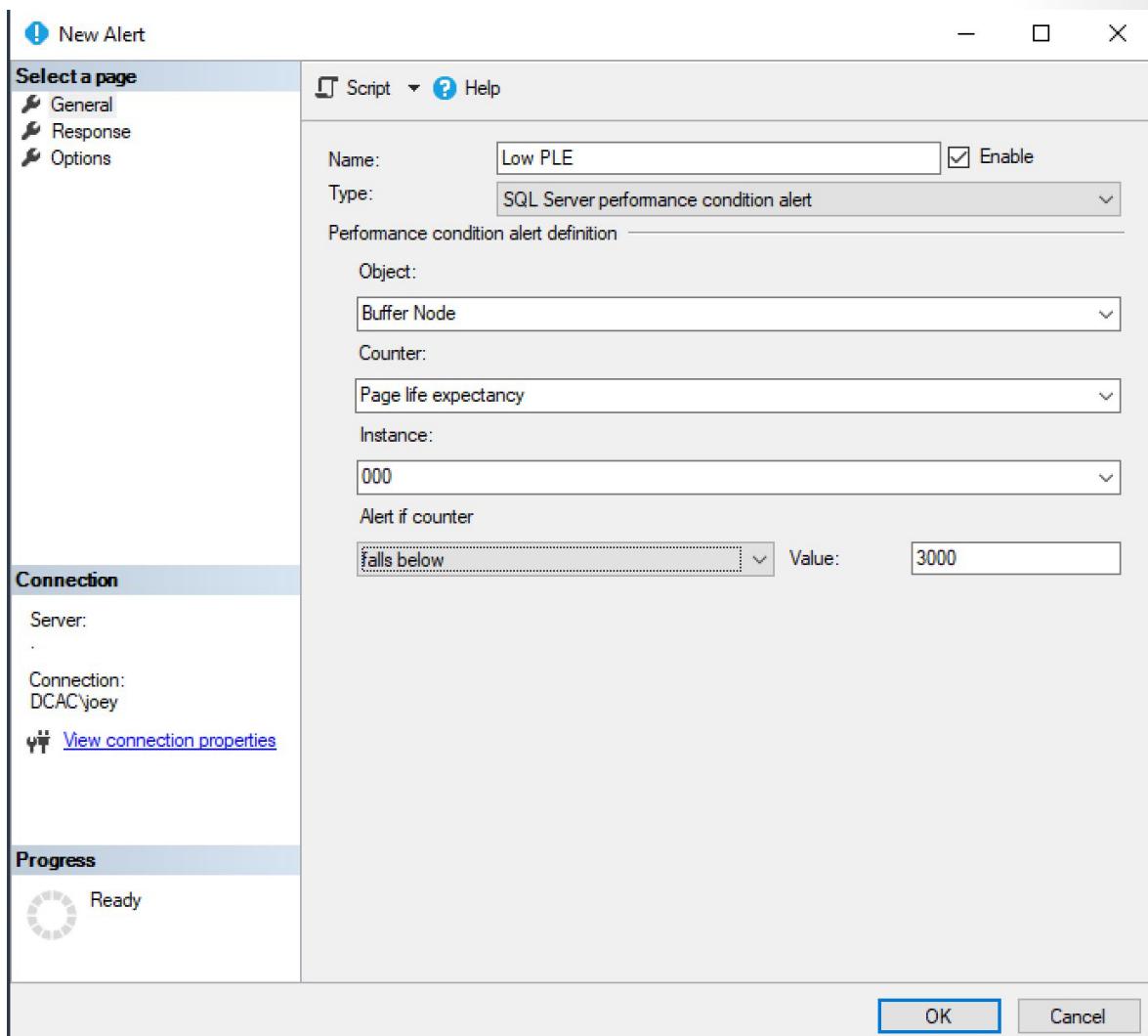
## Alerts

SQL Server Agent alerts allow you to be proactive with monitoring of your SQL Server. The agent reads the SQL Server error log and when it finds an error number for which an alert has been defined, it notifies an operator. In addition to monitoring the SQL Server errorlog, you can set up alerts to monitor SQL Server Performance conditions, as well as Windows Management Instrumentation (WMI) events. You can specify an alert to be raised in response to one or more events. A common pattern is to raise an alert on all SQL Server errors of level 16 and higher, and then add alerts for specific event types related to critical storage errors or Availability Group failover. Another example would be to alert on performance conditions such as high CPU utilization or low Page Life Expectancy. You will learn how to implement alerts and notifications in the next section.

## Notifications Based on Metrics

Another common use case for alerts is that DBAs may want to be notified in the event of certain server conditions. For example, if CPU utilization is over 90% for a period of five minutes, or Page Life Expectan-

cy drops below a certain value. This is accomplished by creating performance condition alerts as shown in Figure 8. These conditions are based on the Windows Performance Monitor (perfmon) metrics that are tracked within the SQL Server database engine. You can reach the screen shown in Figure 8 by right-clicking SQL Server Agent (if it is running) and choosing New|Alert..



*Figure 10 SQL Server Agent Metric Alert Configuration*

You have options for how to respond to the performance condition—you can notify an operator via email, which is the most common approach, or you can execute another SQL Server Agent job, which could resolve the problem. Executing another SQL Server Agent job is most commonly used in the scenario where the condition is well-known, and easily handled without manual intervention. A good example of this would be to create an alert on for SQL Server storage error conditions (errors 823, 824, 825), and then to execute a job to perform a database consistency check. The notifications for these alerts use the same SQL Server Agent subsystem.

## Summary and Knowledge Check

The SQL Server Agent provides a robust mechanism for managing and maintaining your SQL Server and Azure SQL Managed instance deployments. In addition to providing job scheduling, it provides alerting and some monitoring for your databases. Maintenance plans can be used to provide backups, index and

statistics maintenance and log management activity. You should note that the SQL Server Agent is only available on SQL Server and Azure SQL Managed Instance, but not Azure SQL Database. Finally, the most important thing you can do as a DBA is ensure that your databases are backed up, and that you test your database backups regularly by making sure they can be successfully restored.

## Question 1

*What has to be configured before the SQL Server Agent can send e-mail?*

- a mail profile
- An agent job
- An alert

## Question 2

*Which system database stores SQL Server Agent jobs and their information?*

- MSDB
- Master
- Model

## Question 3

*Which operation recalculates the statistics on an index?*

- Rebuild
- Reorganize
- Shrinking a file group

# Configuring Extended Events

## Introduction

One of the ways you can benefit from Azure is using the built-in monitoring of your resources that the platform provides, combined with options that the Azure platform offers for handling and responding to events. It's also important to understand SQL Server's event handling system, called Extended Events and to be familiar with how you can leverage it to perform extensive monitoring of your systems.

## Learning Objectives

In this lesson you will learn:

- How to monitor for server/database configuration changes with Extended Events
- How to use Extended Events for Performance Analysis

## What are Extended Events and Why Do I Need to Know About Them?

Extended events are a lightweight and very extensive diagnostic system that is built into SQL Server and Azure SQL Database and Managed Instance. Extended Events allow you collect additional information about the internal operations of your databases. Historically, DBAs used a tool call Profiler to trace inbound queries and gather execution plans to identity to problematic queries and their execution plans. Extended Events builds on the functionality of Profiler by allowing you to trace queries and by exposing additional data (events) that you can monitor. Some examples of issues you might troubleshoot with Extended Events include:

- Troubleshooting blocking and deadlocking performance issues
- Identifying long-running queries
- Monitoring Data Definition Language (DDL) operations
- Logging missing column statistics
- Observing Memory Pressure in your database
- Long-running physical I/O operations

The extended event framework also allows you to use filters to limit the amount of data you collect in order to reduce the overhead of data collection, and allows you to more easily identity your performance problem by targeting your focus onto specific areas.

## What Can I Monitor with Extended Events?

Extended Events cover the full surface area of SQL Server, and are divided into four channels, which define the audience of an event.

- Admin—Admin events are targeted for end users and administrators. The events included indicate a problem within a well-defined set of actions an administrator can take. An example of this is the generation of an XML deadlock report to help identity the root cause of the deadlock.
- Operational—Operational events are used for analysis and diagnostics or common problems. These events can be used to trigger an action or task based on an occurrence of the event. An example of an

operational event would be a database in an availability group changing state, which would indicate a failover.

- Analytic—Analytic events are typically related to performance events and are published in high volume. Tracing stored procedure or query execution would be an example of an analytic event.
- Debug—Debug events are not necessarily fully documented and should only be used when troubleshooting in conjunction with Microsoft support.

Events are added to sessions which can host multiple events. Typically, multiple events are grouped together in a session to capture a related set of information.

## How to Create an Extended Events Session

Here you will see the basic process of creating an Extended Events session using the New Session dialog from SQL Server Management Studio. You can get to this screen by expanding the Management node in SSMS, expanding the Extended Events node, right-clicking on Sessions and selecting New Session.

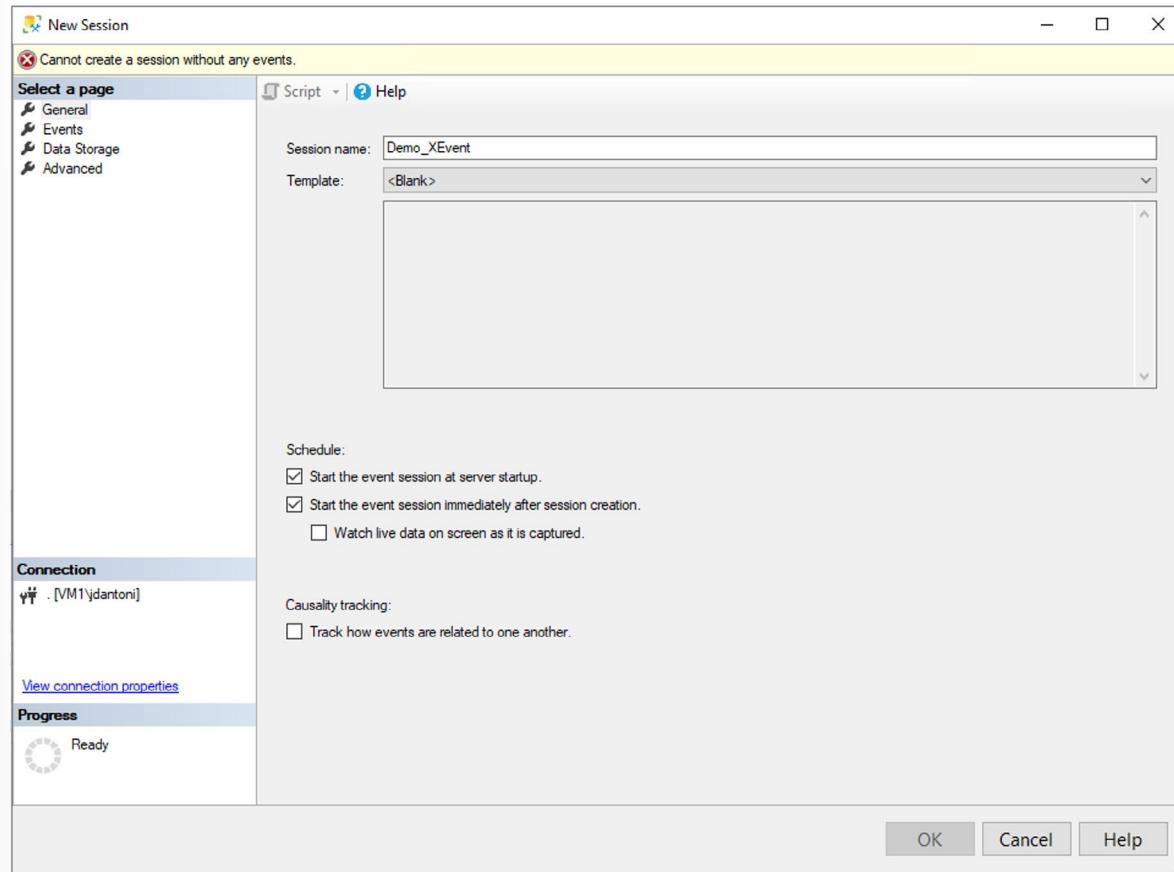


Figure 11 Creating an Extended Events Session

Figure 9 shows the first creation screen in the new Extended Events session. You must first name the session. SQL Server provides numerous templates which are grouped into the following categories:

- Locks and Blocks
- Profiler Equivalents
- Query Execution

- System Monitoring

These predefined templates can allow you quickly get started with using Extended Events for monitoring. In this example, you will see events manually added to the session to walk through all of the options, but when you are getting started, using a template can be an easy way to create a basic session. You have a couple of check box options for when to start this session. You can choose to have your new session start whenever the server starts, and you can also choose to start the session as soon as it's been created. Event sessions can be started and stopped by the administrator at any time through the Extended Events node in SQL Server Management Studio. You also have the option of enabling causality tracking, which adds a globally unique identifier (GUID) and sequence number to the output of each event, which allows you to easily step through the order that the events occurred.

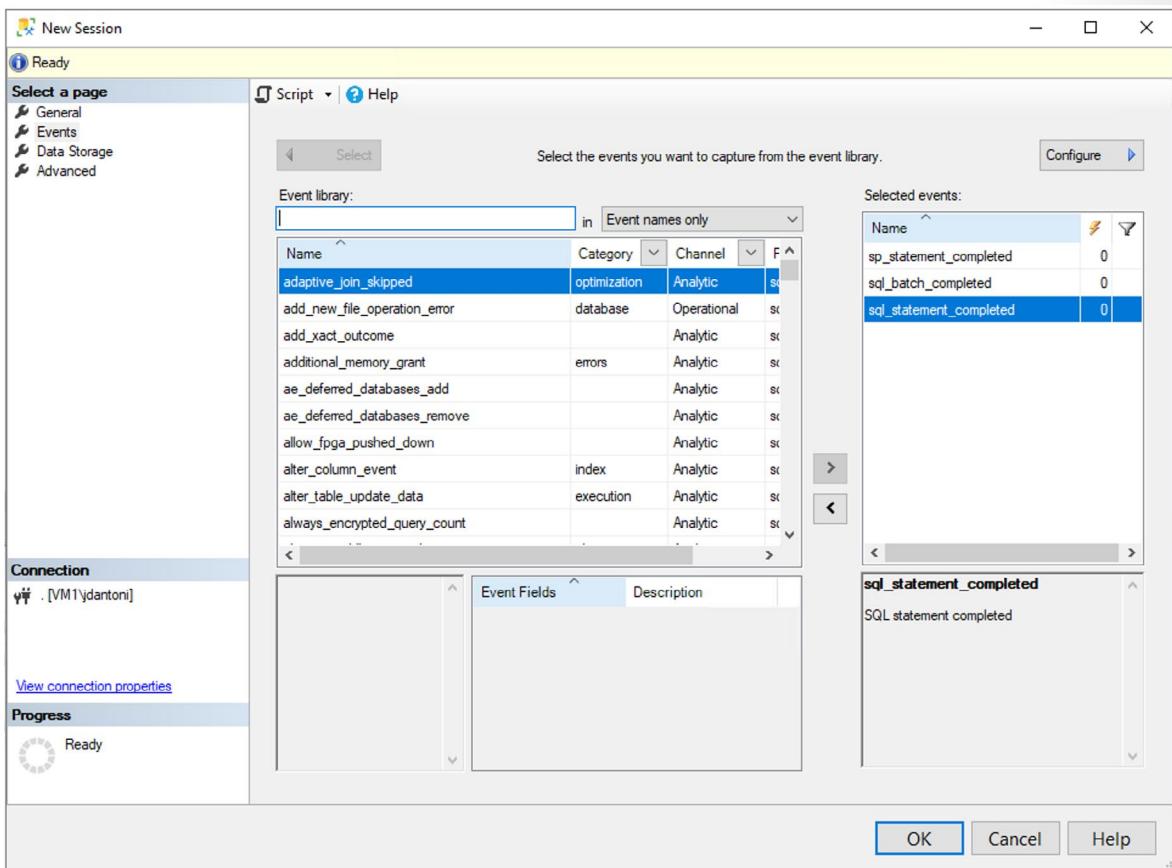


Figure 12 Event Selection in SQL Server Management Studio

Figure 10 shows the screen where you add the events to your session. An event represents a point of interest within the code of database engine—these can represent purely internal system operations, or they can be associated with user actions like query execution. In the above example, you can see that three events (`sp_statement_completed`, `sql_batch_completed`, `sql_statement_completed`) have been added to this event session. By default, this session would capture all instances of these events taking place on your instance. You can limit collection by clicking the configure button.

Event configuration options:		
	Name	Description
<input type="checkbox"/>	callstack	Collect the current call stack
<input type="checkbox"/>	client_app_na...	Collect client application name
<input type="checkbox"/>	client_connec...	Collects the optional identifier provided at connec...
<input type="checkbox"/>	client_hostname	Collect client hostname
<input type="checkbox"/>	client_pid	Collect client process ID
<input type="checkbox"/>	collect_cpu_c...	Collect the current CPU's cycle count
<input type="checkbox"/>	collect_curren...	Collect the current Windows thread ID
<input type="checkbox"/>	collect_syste...	Collect the current system time with 100 microse...
<input type="checkbox"/>	compile_plan...	Collect compiled plan guid. Use this to uniquely i...
<input type="checkbox"/>	context_info	Collect the same value as the CONTEXT_INFO(...)
<input type="checkbox"/>	cpu_id	Collect current CPU ID
<input type="checkbox"/>	create_dump...	Create mini dump including all threads
<input type="checkbox"/>	create_dump...	Create mini dump for the current thread
<input type="checkbox"/>	database_id	Collect database ID
<input type="checkbox"/>	database_name	Collect current database name
<input type="checkbox"/>	datapool_ddl_...	Collect guid datapool DDL query
<input type="checkbox"/>	debug_break	Break the process in the default debugger
<input type="checkbox"/>	distributed_pl...	Collect distributed plan step

Figure 13 Global Event Selection

The event configuration screen allows for defining what data you are collecting as it relates to your events. Global fields, shown in Figure 11, allow you to choose the data you are collecting, when your event occurs. Global fields are also known as actions, as the action is to add additional data fields to the event. These fields represent the data that is collected when the extended event occurs, and are common across most extended events. Figure 12 shows the Filter options for an extended event.

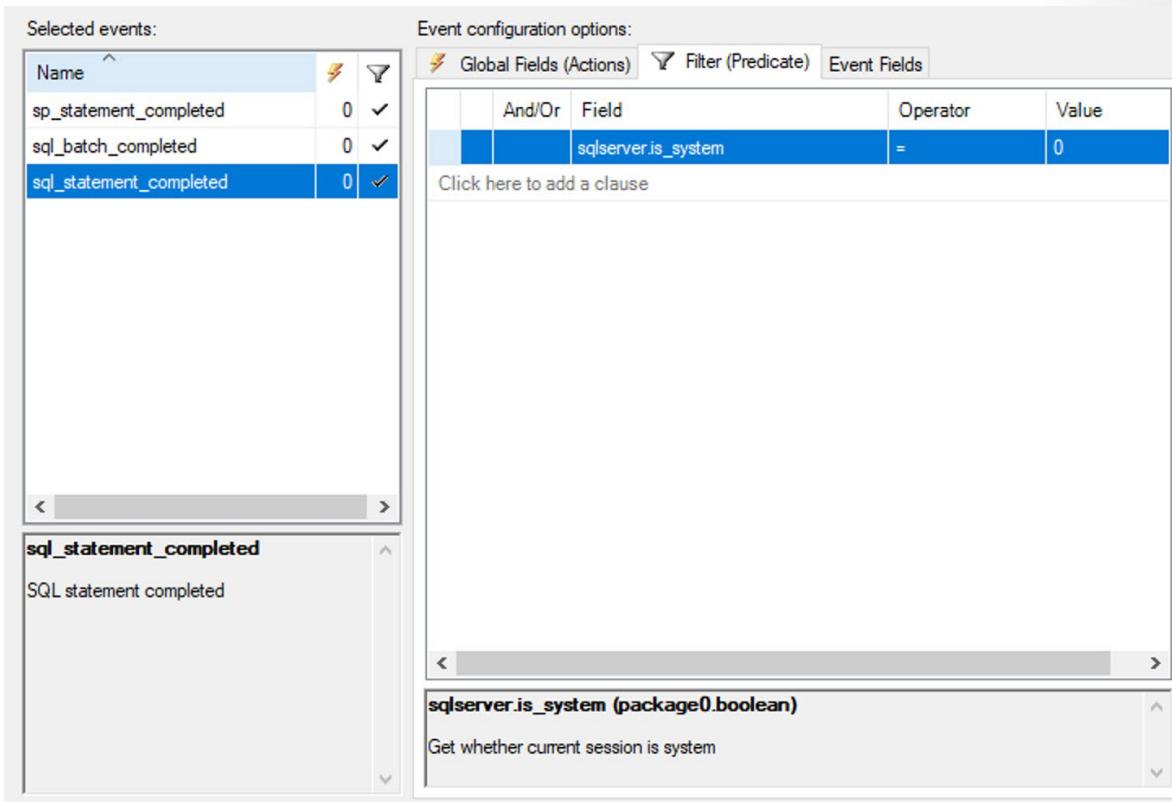


Figure 14 Event Filter in SQL Server Management Studio

Filters are a powerful feature of Extended Events that allow you to use granular control to capture only the specific occurrences of the event you want to capture. In this example, you can see that filter is being applied on the field `sqlserver.is_system` where it is equal to zero, which indicates that the query is not an internal operation. In other words, the session will not capture completion of statements that are submitted by system connections. We only want to capture statements submitted by users or user applications.

Filters apply to a single field on a single event. If you want to make sure that you aren't tracing system activities for any events, you'll need a separate filter for each: for the `sql_statement_completed` event (shown), for the `sql_batch_completed` event, and for the `sp_statement_completed` event (which is a statement run inside a stored procedure).

It is recommended that you configure a filter for each event that you are capturing. This helps improve the efficiency of data collection and allows you to narrow the focus of your search.

Figure 13 shows the event fields that are collected. These are specific to the event being triggered and can include optional fields for collection. In the above event, you can see the optional collection options are `statement` and `parameterized_plan_handle`. In this example, on the `statement` field has been selected for collection.

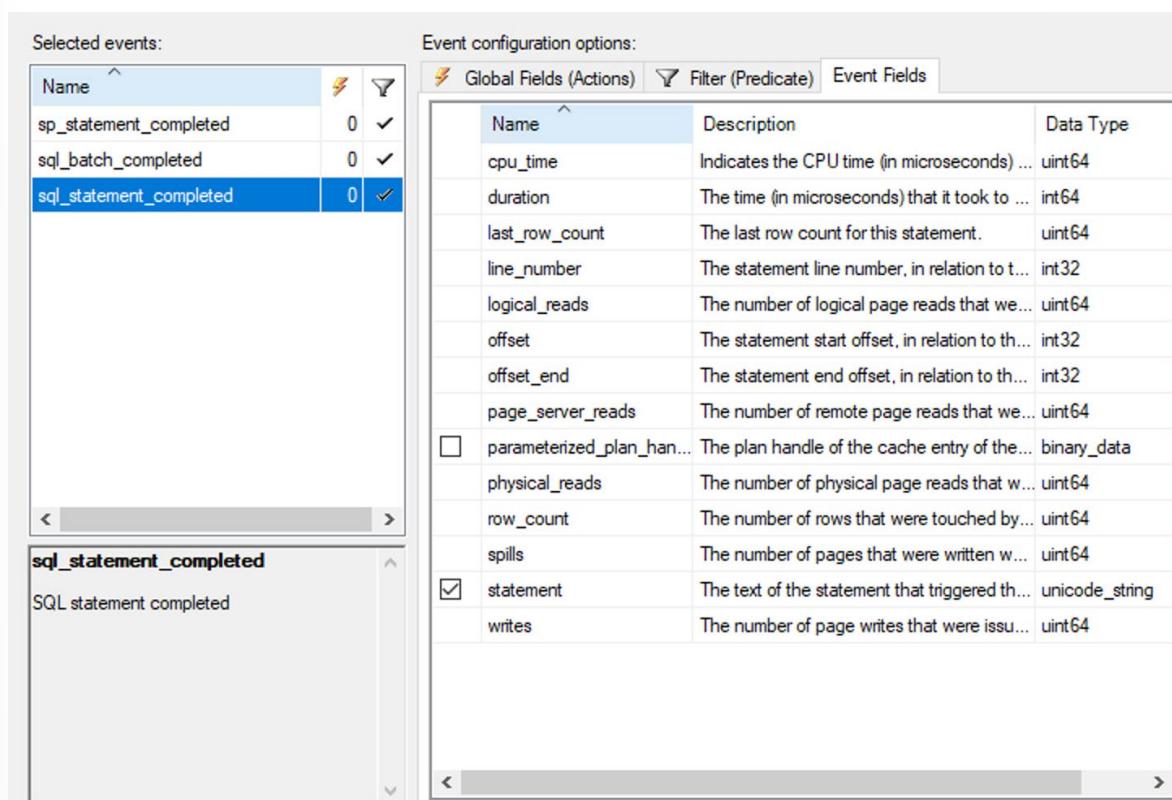


Figure 15 Event Fields Selection

Once you have defined an event session, you will define a storage target, as shown in Figure 14.

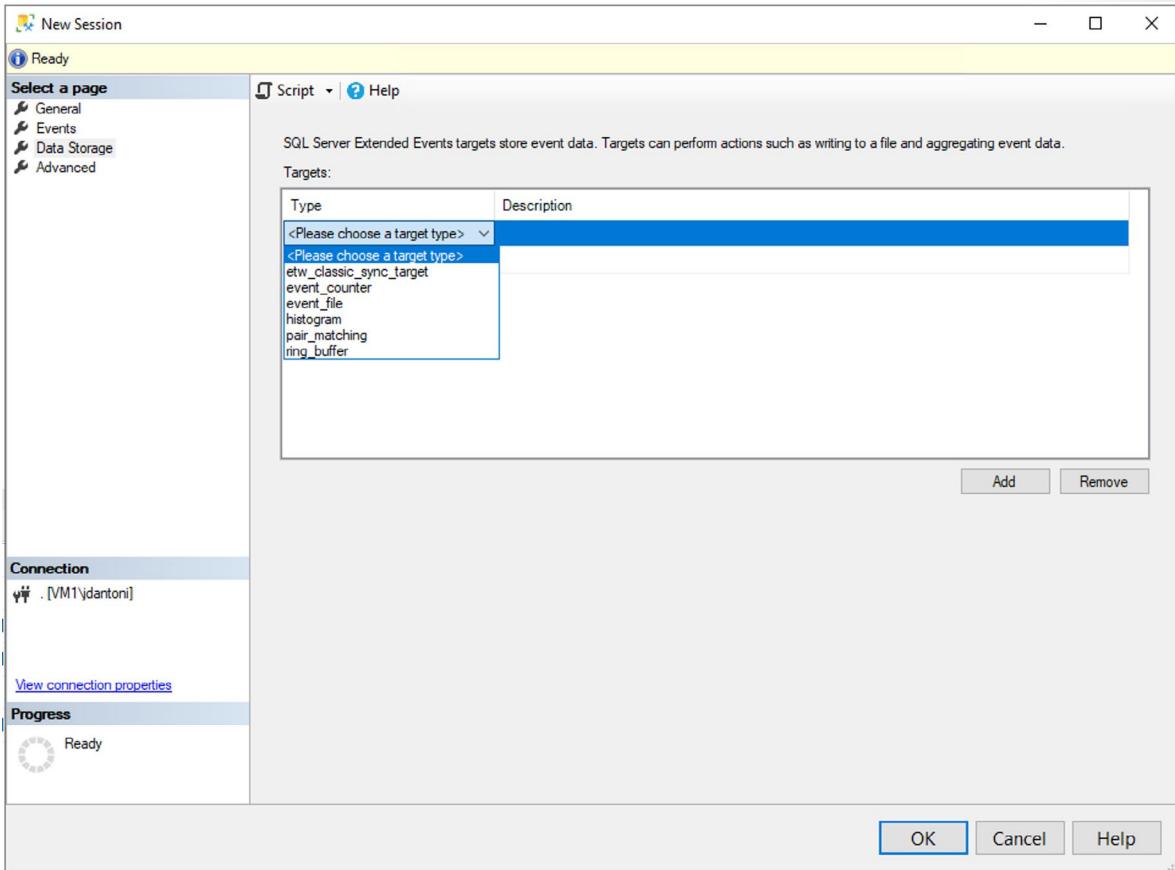


Figure 16 Storage Target Selection for Extended Events

An extended event session has a target—a target can be simply thought of as a place for the engine to keep track of occurrences of an event. Two of the more common targets are *event file* which is a file on the file system that can store events (the size of the file and the number of files is configurable to control runaway growth), and in Azure SQL PaaS offerings this data is written to blob storage. Another common target is the *ring buffer* which is within SQL Server's memory. The *ring buffer* is most commonly used for live observation of an event session, as it's a circular buffer and data is not persisted beyond a session. Most targets process data asynchronously, which means that the event data is written to memory before being persisted to disk. The exception is the Event Tracing for Windows target (ETW) and Event Counter targets which are processed synchronously.

The following table contains information and uses for each type of Extended Events target.

Target	Description	Processing
<b>Event Counter</b>	Counts all events that occurred during an Extended Event session. This is used to obtain information about workload characteristics about a workload without the overhead of a full event collection.	Synchronous
<b>Event File</b>	Writes event session output from memory onto persistent file on disk.	Asynchronous

Target	Description	Processing
<b>Event Pairing</b>	Many events that generally occur in pairs (e.g. lock acquire, lock release), and this collection can be used to identify when those events do not occur in a matched set.	Asynchronous
<b>Event Tracing for Windows (ETW)</b>	Used to correlate SQL Server events with the Windows OS event data.	Synchronous
<b>Histogram</b>	This is similar to event counter, which counts the occurrences of an event. The difference is that the histogram can count based on a specific event column or action.	Asynchronous
<b>Ring Buffer</b>	Used to hold data in memory. Data is not persisted to disk and maybe frequently flushed from the buffer	Asynchronous

## Creating Extended Events Sessions with T-SQL

There are two options for creating an Extended Event session. You can create a session programmatically using T-SQL or you can use the GUI in SQL Server Management Studio, as shown in figures 9-14.. However, using T-SQL to deploy event sessions is more easily repeated and should be part of automation routines. The T-SQL for a sample event session is shown below.

```
IF EXISTS (SELECT * FROM sys.server_event_sessions WHERE name='test_session')

DROP EVENT session test_session ON SERVER;

GO

CREATE EVENT SESSION test_session
ON SERVER

ADD EVENT sqlos.async_io_requested,
      ADD EVENT sqlserver.lock_acquired
      ADD TARGET package0.etw_classic_sync_target
      (SET default_etw_session_logfile_path = N'C:\demo\traces\sqletw.etl' )
      WITH (MAX_MEMORY=4MB, MAX_EVENT_SIZE=4MB);

GO
```

Event sessions can be scoped to a server or a database. In the example shown above, you are adding two events and using the Event Tracing for Windows (ETW) path, with a file location. After you create the session, you'll have to start it. You can do this using T-SQL and ALTER the session using the STATE option, or you can use the Extended Events | Session node in SQL Server Management Studio. You can also have the session start at server startup, which is common for lightweight event sessions that monitor over time.

## Summary and Knowledge Check

The Extended Events engine in Azure SQL is an extremely powerful monitoring system that allows you to capture very granular information about activity in your databases and servers. The monitoring solutions on the Azure platform allow you to easily configure powerful monitoring for your environment and provide automated responses to error conditions.

### Question 1

*Which Extended Events target only writes to memory and is not persisted ?*

- Ring Buffer
- Target File
- Event Tracing for Windows

### Question 2

*Where do you implement a filter in your Extended Event Session?*

- On the Event
- On the Storage Target
- On a Global Field

### Question 3

*Which scenario can you troubleshoot using Extended Events?*

- A server restart
- A long running query
- A failed backup

# Managing Azure PaaS Resources by Using Automated Methods

## Introduction

You have learned about some of the capabilities of the SQL Server Agent. However, if you are using Azure SQL Database or Azure Database for MariaDB/MySQL/PostgreSQL, you will have to find alternative scheduling mechanisms, as neither the msdb database nor the SQL Server agent is available. In this lesson you will learn about Azure Automation and Elastic Jobs, two approaches for automating jobs in PaaS. You will also learn about Azure policy and how it can be used to manage your subscription and costs.

In this lesson you will learn about:

- The benefits of Azure Policy
- The capabilities of Azure Automation
- How to use Elastic Jobs

## Implementing Azure Policy

Group Policies or GPOs have been used by Windows server administrators for a long time, to manage security, provide consistency across the Windows Server environment in your organization. Some example of group policies include enforcement of password complexity, mapping shared network drives and configuring networked printers.

Azure offers similar features in ARM using Azure Policy. Policy provides a level of governance over your Azure subscriptions which enforce rules and controls over your Azure resources. Some examples of how you might use this include limiting the regions you can deploy a resource to, enforcing naming standards, or controlling resource sizes. Azure provides many example policies that you can use or you can define custom policies using JSON.

Policies are assigned to a specific scope, which could be a management group (a group of subscriptions that are managed together), a subscription, a resource group, or even an individual resource. Most commonly policy will be applied at the subscription or resource group level. Individual policies can be grouped using a structure known as initiatives, which are sometimes called policy sets. Policies have a scope of assignment which can be defined at the individual resource, the resource group, the subscription, or a management group (a group of subscriptions managed together), or all of the subscriptions in a given tenant.

Another example of how you might implement Azure Policy is tagging of resources. Azure Tags, which are described below, store metadata about Azure resources in key-value pairs, and are commonly used to highlight environment type (test, QA, or production) or cost center for a given resource. A policy that required all resources to have a tag for environment and cost center would cause an error and block the deployment of any Azure resource that did not have the required tags.

## Why Would I Have Multiple Azure Subscriptions?

Organizations use multiple subscriptions for several reasons, including budget management, security, or isolation of resources. One example of this would be an organization that has both internal and customer facing resources. The internal resources could exist in one subscription, and the customer resources in

another, for easier separation of billing and for isolation of the internal resources. These subscriptions may be managed together in a management group which allows you to manage policy and compliance across subscriptions.

## Why Should I Tag Azure Resources?

Tags are simply metadata that are used to better describe your Azure resources. These tags are stored as key:value pairs and appear in the Azure Portal associated with your Azure resources. Since they are associated with the resource, when you use PowerShell or Azure CLI commands, you can filter your commands based on tags. In that sense, you can think of them like a WHERE clause in a SQL query. A very basic example is shown below:

```
$rg=(get-AzResourceGroup)

$rg=($rg|where-object {($_.tags['Use'] -ne 'Internal')}).ResourceGroupName
```

In this code sample on the second line, you can see that the list of resource groups is being filtered by the tag called 'Use', and will return only those resource groups where that tag does not have a value of 'External'. Tags can be applied in the Azure Portal, or programmatically via PowerShell, Azure CLI, or as part of ARM template deployment. Tags can also be applied at the subscription, resource group, or individual resource level. Tags can also be modified at any time. Azure supports applying up to 15 tags to each Azure resource.

Tags are also included in Azure billing information, so tagging by cost center means it can be much easier for management to breakdown the Azure charges. Tags are in the overview section of the blade for every Azure Resource. To add tags to a resource using the Azure Portal, click tags, and enter the key and value for your tag. Click save after you apply the tags to your resources.

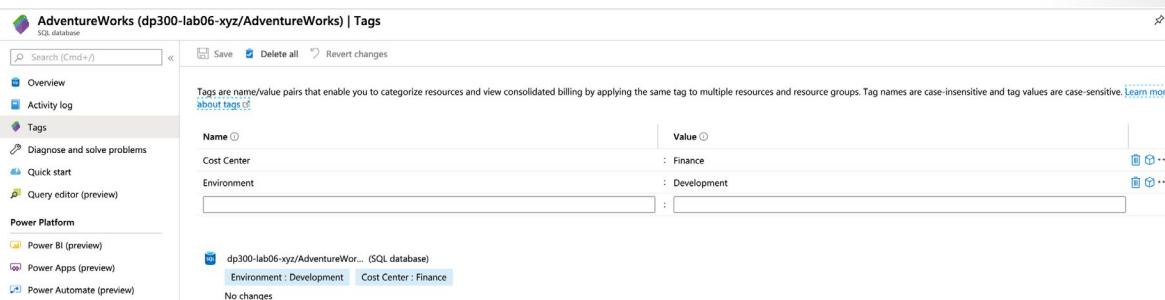


Figure 17 Adding Tags to an Azure SQL Database

You can also use PowerShell or the CLI to add Tag. The PowerShell example is below:

```
$tags = @{"Dept"="Finance"; "Status"="Normal"}

$resource = Get-AzResource -Name demoStorage -ResourceGroup demoGroup

New-AzTag -ResourceId $resource.id -Tag $tags
```

The Azure CLI example is below:

```
az resource tag --tags 'Dept=IT' 'Environment=Test' -g examplegroup -n
examplevnet `
```

```
--resource-type "Microsoft.Network/virtualNetworks"
```

## Azure Automation

Azure offers several ways to automate processes. Azure Functions and Logic Apps are both Azure services that enable serverless workloads. Both services create workflows that are a collection of steps to execute complex tasks. For example, a Logic App can be created to populate a table in an Azure SQL Database when an entry is made in a SharePoint list. A full explanation of these services is beyond the scope of this course, but is linked in the additional resources section of this module.

For more complete control and granularity of your automation, Azure Automation allows for process automation, configuration management, full integration with Azure platform options (such as role-based access control and AAD) and can manage Azure and on-premises resources. One of the unique benefits of Azure Automation is that it can manage resources within Azure or on-premises VMs. For example, if you have a VM that is normally kept in a down state for cost savings (except when it needs to be used), you have the ability within Azure Automation, using a feature called hybrid runbooks, to execute a script to start the VM, then kick off a SQL Server backup from within the VM, and finally shut down the VM.

### *Overview of Azure Automation Components*

Azure Automation supports both automation and configuration management activities. In this module, we are going to focus on the automation components, but you should be aware that Automation can also be used to manage server updates and desired state configuration. The components of Automation you will need to use to execute automated tasks are as follows:

**Runbooks** - Runbooks are the unit of execution in Azure Automation. Runbooks can be defined as one of three types: a graphical runbook based on PowerShell, a PowerShell script, or Python script. PowerShell runbooks are most commonly used to manage Azure SQL resources.

**Modules** - Azure Automation defines an execution context for the PowerShell or Python code you are executing in your runbook. In order to execute your code, you need to import the supporting modules. For example, if you needed to run the Get-AzSqlDatabase PowerShell cmdlet, you would need to import the Az.SQL PowerShell module into your automation account.

**Credentials** - Credentials store sensitive information that runbooks or configurations can use at runtime.

**Schedules** - Schedules are linked to runbooks and trigger a runbook at a specific time.

## Building an Azure Automation Runbook

In order to build an automation runbook, you need to first create an automation account. Figure 16 shows this process in the Azure Portal, after selecting Azure Automation from the Azure Marketplace.

## Add Automation Account

Name \* ⓘ  
Demo-DP300 ✓

Subscription \*  
Contoso Ltd ▾

Resource group \*  
(New) DemoDP300 ▾  
[Create new](#)

Location \*  
(US) East US 2 ▾

Create Azure Run As account \* ⓘ  
[Yes](#) [No](#)

**i** This will create Azure Run As account in the Automation account which are useful for authenticating with Azure to manage Azure resources from Automation runbooks. Note that the creation of Azure Run As account may affect the security of the subscription.[Learn more](#)

Figure 18 Creating an Automation Account in the Azure Portal

You should note that this process can optionally create an Azure Run As account, which creates a service principal in your Azure Active Directory which provides authentication for Azure Automation to access Azure Resources.

In this example runbook, you are going to connect to an Azure SQL Database using PowerShell. This means you need to import modules to support those cmdlets. Before you create your runbook, you will import modules into your Azure Automation account. In order to do this, navigate to the Shared Resources section of the main blade for your automation account and click Modules Gallery. The first module you will import, is **Az.Accounts** as the **Az.SQL** module is dependent on it.

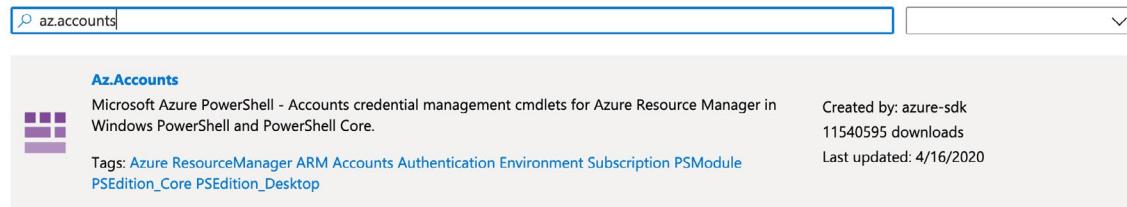


Figure 19 Modules Gallery Search for Az.Accounts Module

You will search for the module in the gallery, and search for the module you are looking for as shown in Figure 17. After you click on the module, you will have the option to import it, as shown in Figure 18.

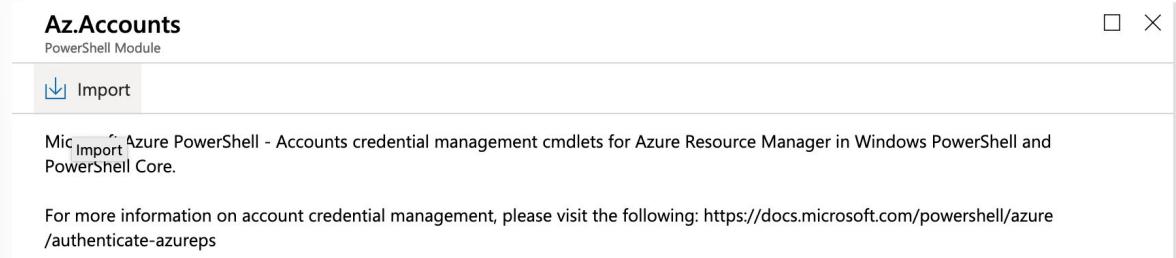
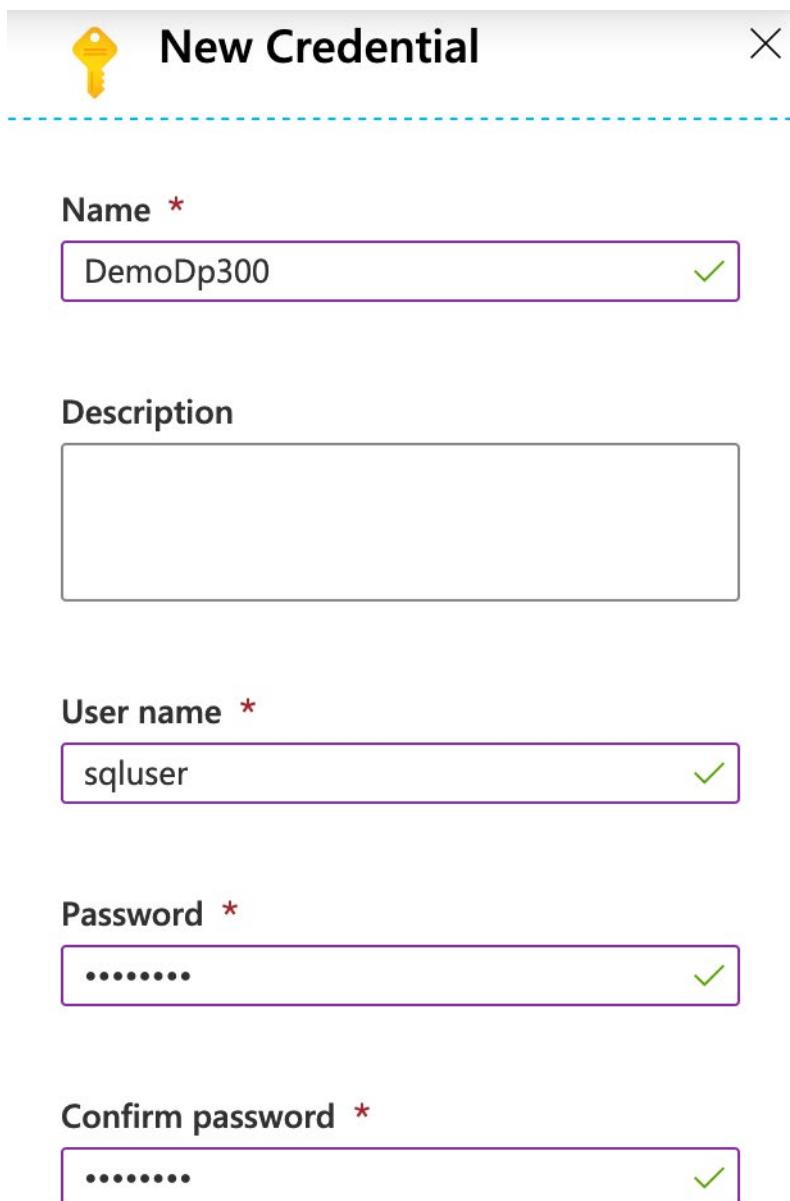


Figure 20 Import PowerShell Module into Automation Account

This will import the module into your account. In this example, the process was repeated for the Az.SQL and SqlServer PowerShell modules.

Next, you can optionally create a credential that will your runbook can utilize. You can create a credential by clicking on Credentials in the Shared Resources section of the main blade of your automation account as shown in Figure 19. You do not have to create a credential in order to use Azure Automation, but this example does refer to one.



The screenshot shows a 'New Credential' dialog box. At the top left is a yellow key icon. To its right is the title 'New Credential'. In the top right corner is a close button (an 'X'). Below the title is a dashed blue horizontal line. The first field is 'Name \*' with a red asterisk, containing the value 'DemoDp300' in a purple-bordered input field, which has a green checkmark icon to its right. The next section is 'Description', which is currently empty. The third section is 'User name \*' with a red asterisk, containing the value 'sqluser' in a purple-bordered input field, which has a green checkmark icon to its right. The fourth section is 'Password \*' with a red asterisk, containing six dots '.....' in a purple-bordered input field, which has a green checkmark icon to its right. The fifth section is 'Confirm password \*' with a red asterisk, containing six dots '.....' in a purple-bordered input field, which has a green checkmark icon to its right.

Figure 21 Create Credential in Azure Automation

Next, you will create a runbook after navigating to the Process Automation section of the automation blade and clicking Runbooks. Your account will come with three sample runbooks (one for each type of runbook).

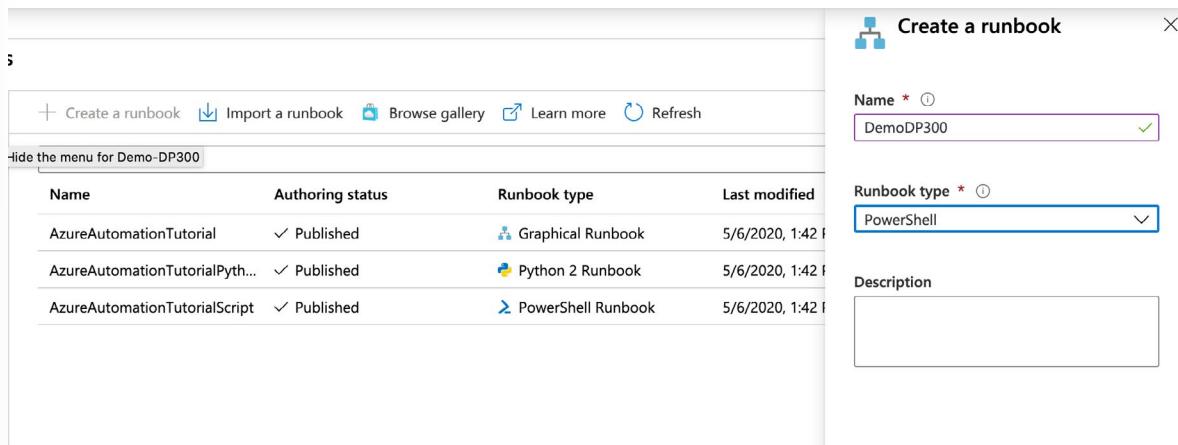


Figure 20 Runbook Creation in Azure Portal

When you are creating the runbook, you provide a name, the type of runbook, and optionally a description. Because this example specified PowerShell as the type, a PowerShell editor opens, as shown in Figure 21.

The screenshot shows the 'Edit PowerShell Runbook' page. At the top, there are buttons for 'Save', 'Publish', 'Revert to published', 'Test pane', and 'Feedback'. Below that is a tree view of available cmdlets, with 'DLETS' selected. The main area contains the following PowerShell script:

```

1 # Ensures you do not inherit an AzContext in your runbook
2 Disable-AzContextAutosave -Scope Process
3
4 $connection = Get-AutomationConnection -Name AzureRunAsConnection
5
6 # Wrap authentication in retry logic for transient network failures
7 $logonAttempt = 0
8 while(!$connectionResult) -And ($logonAttempt -le 10)
9 {
10     $logonAttempt++
11     # Logging in to Azure...
12     $connectionResult = Connect-AzAccount `-
13         -ServicePrincipal `-
14         -Tenant $connection.TenantID `-
15         -ApplicationId $connection.ApplicationID `-
16         -CertificateThumbprint $connection.CertificateThumbprint
17
18     Start-Sleep -Seconds 30
19 }
20
21 $AzureContext = Get-AzSubscription -SubscriptionId $connection.SubscriptionID
22
23 $dbname=(Get-AzSQLDatabase -ResourceGroupName 'SQLDB' -ServerName 'GSData' -Database 'GSData').DatabaseName
24
25 $AzureSQLServerName = $dbname + ".database.windows.net"
26 $Cred = Get-AutomationPSCredential -Name "SQLUser"
27 $SQLOutput = $(Invoke-Sqlcmd -ServerInstance $AzureSQLServerName -Username $Cred.UserName -Password $Cred.GetNetworkCredential().Password `-
28 -Database $DBName -Query "SELECT * FROM INFORMATION_SCHEMA.TABLES" -Verbose) 4>&1
29
30 Write-Output $SQLOutput
31
32
33
34

```

Figure 21 Sample Azure Automation Runbook

Figure 21 shows the edit runbook screen which is where you define the code you are executing. In this example, the runbook is connecting to the Azure subscription, getting information about an Azure SQL Database, running a query, and then returning the results. In this example, in lines 1-21, you are executing a series of cmdlets to connect to the Azure account. You are then getting the database name from

the `Get-AzSQLDatabase` cmdlet, and then use the `get-AutomationPSCredential` cmdlet to assign your credential to a variable. Finally, you are assigning the `Invoke-Sqlcmd` cmdlet to execute a query against the Azure SQL Database, and using the `Write-Output` cmdlet to return the results of the query.

After you have completed your code in the portal, you click on “Test Pane” in the code editor in the Azure Portal. This allows you to test your code in the context of Azure Automation. A typical development process is to create your PowerShell code locally and then test it within the automation environment. This allows you to separate any PowerShell errors from errors which might be generated from the context of automation execution. You should always test your code within automation, to ensure there are no errors in the code itself.

The screenshot shows the Azure Portal interface for testing a PowerShell runbook. At the top, the navigation bar includes Home > Demo-DP300 | Runbooks > DemoDP300 (Demo-DP300/DemoDP300) > Edit PowerShell Runbook > Test. The main area is titled "Test" and shows the status as "Completed". On the left, there are sections for "Parameters" (No input parameters), "Run Settings" (Run on Azure), and "Activity-level tracing" (using a hybrid runbook worker can increase test performance). Below these are buttons for Start, Stop, Suspend, Resume, View last test, and Refresh job streams. On the right, the command output window displays the following PowerShell session:

```
Mode          : Process
ContextDirectory :
ContextFile   :
CacheDirectory :
CacheFile     :
Settings      : {}



| TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME              | TABLE_TYPE |
|---------------|--------------|-------------------------|------------|
| gsdata        | sys          | database_firewall_rules | VIEW       |


```

Figure 22 Successfully Completed Test Runbook

Figure 22 shows the results of the completed runbook. You should note the informational bubble on the left side of the screen referring to hybrid runbooks. Hybrid runbooks are used when you need to execute cmdlets inside of a VM. This requires a configuration on the VMs and in the Azure Automation account. This concept can be a bit confusing, but the easiest way to think about it is to think of Azure resources as boxes that are managed by ARM. Without a hybrid runbook you can manage the state of those boxes, but you cannot access or manage anything within the boxes. Hybrid runbooks give you the option to control what's inside of the box.

After you have successfully tested your runbook, you can then click publish in the runbook editor screen shown in Figure 21 above. A runbook must be published in order to be executed by the service. After you have published the runbook, you can create a schedule, by clicking on schedules in the Shared Resources section of the automation account blade.

New Schedule X

Name \*  
Runbook Schedule ✓

Description

Starts \* (i)  
05/06/2020   4:30 PM

Time zone  
 ▼

Recurrence  
Once Recurring

Recur every \*  
 Day ▼

Set expiration  
Yes No

Expires  
Never

Figure 23 Create Schedule Screen in Azure Automation

Figure 23 shows the creation process for a new schedule. You should note that the default settings are for there to be no recurrence of the job. In the above example, the job has been configured to run once daily

at 4:30 PM Eastern Time. Once you have created a schedule, you can link it to a runbook, by navigating back to the runbook and clicking Link to schedule in the runbook page as shown in Figure 24.

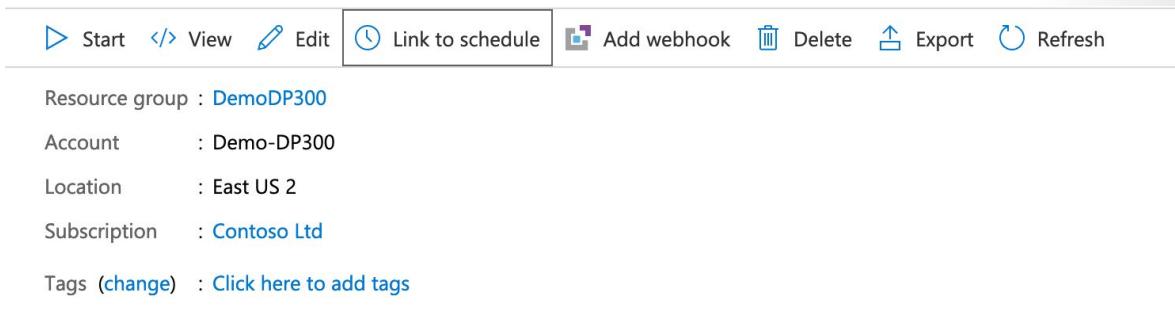


Figure 24 Link to Schedule in Azure Automation Runbook

## Configuration Management

Azure Automation also supports using PowerShell Desired State Configuration (DSC) to manage changes in configuration across VMs in your environment. DSC operates as an extension to VMs and provides a consistent configuration state across all of the VMs that the configuration is applied to. Azure Automation integrates with DSC, allowing you to automatically update configuration across physical and virtual machines, and includes reporting capabilities.

## Elastic Jobs in Azure SQL Database

One of the reasons why many DBAs became so familiar with Azure Automation is that Azure SQL Database initially lacked capabilities for scheduled jobs. The introduction of Elastic Jobs allows you to run a set of T-SQL scripts against a collection of servers or databases as a one-time job in an ad hoc manner, or by using a defined schedule. Elastic jobs work similarly to SQL Server Agent jobs, except that they are limited to executing T-SQL. The jobs work across all tiers of Azure SQL Database (excluding managed instance, which has a SQL Server Agent). To configure Elastic Jobs, you need a Job Agent and database dedicated to managing your jobs. The agent is free, but the database is required to be a paid database. The recommended service tier is S1 or higher, and this will be dependent on the number of jobs you are executing and the frequency of those jobs.

In elastic jobs you must define a target group, which can consist of a SQL Database server, an elastic pool, or one or more single databases. If a server or elastic pool is the target, a credential within the master database of the server or pool should be created so that the job agent can enumerate the databases within. For a single database, a database credential is all that is needed. It is also important to remember that T-SQL scripts being executed by Elastic Jobs be idempotent, which means if the job is run multiple times, whether accidentally or because of job failure that the job won't fail or produce unintended results. You should be able to run the same script multiple times without failure.

## Summary and Knowledge Check

In this module you learned about various ways to automate common tasks in Azure and within SQL Server. One of the benefits of cloud computing is that a framework for robust automation, monitoring, and tooling is part of the platform. In the on-premises world, a complex set of tools would need to be assembled just to match the built-in functionality available with Azure Metrics, Policy, and Automation. Additionally, SQL Server provides an automation framework using SQL Agent, and a very robust monitor-

ing solution via Extended Events. You can take advantage of both Azure Automation and Azure Elastic Jobs to automate the management of your Azure resources.

## Question 1

*What is the unit of execution for your Azure Automation Account?*

- Runbook
- Schedule
- Container

## Question 2

*What scope can Azure Policy be deployed to?*

- Tenant
- Subscription
- User

## Question 3

*What is the name for the scope of SQL Elastic Job?*

- Target Group
- Management Group
- Resource Group

## Module Summary

### Module Summary

Automation is a key skill that the DBA needs to possess. Whether it is deploying Azure resources in an automated fashion, or managing the maintenance for an Azure SQL Managed Instance, or SQL Server in an Azure VM. Extended Events provide a framework to monitor and understand the internal workings across all of the Azure SQL platform. Finally, you can use Azure Automation or Elastic Jobs to execute scheduled tasks against Azure SQL and other Azure resources.

# Answers

## Question 1

What language are ARM templates written in?

- JSON
- C#
- T-SQL

*Explanation*

## Question 2

If you want to pass in the region for a resource group deployment which option should you include in your template?

- Parameter
- Variable
- Output

*Explanation*

## Question 3

Which element of a template allows for you to build dependencies into resources?

- dependsOn
- concat
- apiVersion

*Explanation*

## Question 1

What has to be configured before the SQL Server Agent can send e-mail?

- a mail profile
- An agent job
- An alert

*Explanation*

## Question 2

Which system database stores SQL Server Agent jobs and their information?

- MSDB
- Master
- Model

*Explanation*

**Question 3**

Which operation recalculates the statistics on an index?

- Rebuild
- Reorganize
- Shrinking a file group

*Explanation*

**Question 1**

Which Extended Events target only writes to memory and is not persisted ?

- Ring Buffer
- Target File
- Event Tracing for Windows

*Explanation*

**Question 2**

Where to you implement a filter in your Extended Event Session?

- On the Event
- On the Storage Target
- On a Global Field

*Explanation*

**Question 3**

Which scenario can you troubleshoot using Extended Events?

- A server restart
- A long running query
- A failed backup

*Explanation*

**Question 1**

What is the unit of execution for your Azure Automation Account?

- Runbook
- Schedule
- Container

*Explanation*

**Question 2**

What scope can Azure Policy be deployed to?

- Tenant
- Subscription
- User

*Explanation*

**Question 3**

What is the name for the scope of SQL Elastic Job?

- Target Group
- Management Group
- Resource Group

*Explanation*

# Module 7 Planning and Implementing a High Availability and Disaster Recovery Environment

## Module Introduction

### Module Introduction

Data must be available when the business needs it. That means the solutions hosting the data must be designed with availability and recoverability in mind.

Suppose you work for a company that sells widgets both in stores and online. Your main application uses a highly transactional database for orders. What would happen if the server or platform hosting the transactional database had a problem that made it unavailable or inaccessible for some reason? What impact would it have on the business? If the right solution is put in place, the database would come online in a reasonable timeframe with minimal effort, thus allowing business to continue with little-to-no impact.

This module and its associated lab cover configuring, testing, and managing a solution for high availability and disaster recovery (HADR) in Azure, for both Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS) deployments. This module will not only cover basic requirements, but also the various options available to achieve HADR. You will get hands-on experience with some of these options through the lab.

## Learning Objectives

In this module, you will learn:

- The difference between recovery time and recovery point objectives
- The available HADR options for both IaaS and PaaS
- The considerations for planning and configuring HADR solutions including how backup and restore fits in
- The factors that comprise a HADR strategy
- How to configure a high availability solution via a hands-on lab

# High Availability and Disaster Recovery Strategies

## Introduction

Implementing HADR database platform solutions is more than just deploying a feature. You must understand why and what you are doing to implement the right strategy.

Suppose you need to implement HADR for an IaaS VM that is running SQL Server 2019. Do you know how much time you would have to bring things back online if there was a problem? Do you know what options are available in SQL Server and the reasons for choosing one option over another?

By the end of this lesson, you will have a solid foundation that will allow you to implement HADR solutions in Azure.

## Learning Objectives

After this lesson, you will understand:

- Recovery time objective (RTO)
- Recovery point objective (RPO)
- The available HADR options for both IaaS and PaaS
- How to devise a HADR strategy

## Recovery Time Objective and Recovery Point Objective

Understanding recovery time and recovery point objectives is crucial to your HADR plan as they are the foundation for any availability solution.

### Recovery Time Objective

RTO is the maximum amount of time available to bring resources online after an outage or problem. If that process takes longer than the RTO, there could be consequences such as financial penalties, work not able to be done, and so on. RTO can be specified for the whole solution which includes all resources, as well as for individual components such as SQL Server instances and databases.

### Recovery Point Objective

RPO is the point in time to which a database should be recovered and equates to the maximum amount of data loss that the business is willing to accept. For example, suppose an IaaS VM containing SQL Server experiences an outage at 10:00 AM and the databases within the SQL Server instance have an RPO of 15 minutes. No matter what feature or technology is used to bring that instance and its databases back, the expectation is that there will be at most 15 minutes worth of data lost. That means the database can be restored to 9:45 AM or later to ensure minimal to no data loss meeting that stated RPO. There may be factors that determine if that RPO is actually achievable, and those will be discussed in the next section.

## Defining Recovery Time and Recovery Point Objectives

RTOs and RPOs are driven by business requirements but are also based on various technological and other factors, such as the skills and abilities of the administrators (not just DBAs). While the business may want no downtime or zero data loss, that may not be realistic or possible for a variety of reasons. Determining your solution's RTO and RPO should be an open and honest discussion between all parties involved.

One of the aspects crucial for both RTO and RPO is knowing the cost of downtime. If you define that number and the overall effect being down or unavailable has to the business, it is easier to define solutions. For example, if the business can lose \$10,000 per hour or could be fined by a government agency if something could not be processed, that is a measurable way to help define RTO and RPO. Spending on the solution should be proportional to the amount, or the cost, of downtime. If your HADR solution costs \$X, but you wind up only being affected for a few seconds instead of hours or days when a problem occurs, it has paid for itself.

From a non-business standpoint, RTO should be defined at a component level (e.g. SQL Server) as well as for the entire application architecture. The ability to recover from an outage is only as good as its weakest link. For example, if SQL Server and its databases can be brought online in five minutes but it takes application servers 20 minutes to do the same, the overall RTO would be 20 minutes, not five. The SQL Server environment could still have an RTO of five minutes; it still will not change the overall time to recover.

RPO deals specifically with data and directly influences the design of any HADR solution as well as administrative policies and procedures. The features used must support both the RTO and RPOs that are defined. For example, if transaction log backups are scheduled every 30 minutes but there is a 15-minute RPO, a database could only be recovered to the last transaction log backup available which in a best case would be 30 minutes ago. This assumes no other issues and the backups have been tested and are known to be good. While it is hard to test every backup generated for each database in your environment, backups are just files on a file system. Without doing at least periodic restores, there is no guarantee they are good. Running checks during the backup process can give you some degree of confidence.

The specific features used, such as an Always On Availability Group (AG) or an Always On Failover Cluster Instance (FCI) will also affect your RTOs and RPOs. These features will be described in the next section. Depending on how the features are configured, IaaS or PaaS solutions may or may not automatically failover to another location, which could result in longer downtime. By defining RTO and RPO, the technical solution that supports that requirement can be designed knowing the allowances for time and data loss. If those wind up being unrealistic, RTOs and RPOs must be adjusted accordingly. For example, if there is a desired RTO of two hours but a backup will take three hours to copy to the destination server for restoring, the RTO is already missed. These types of factors must be accounted for when determining your RTOs and RPOs.

There should be RTOs and RPOs defined for both HA and DR. HA is considered a more localized event that can be recovered from more easily. One example of high availability would be an AG automatically failing over from one replica to another within an Azure region. That may take seconds, and at that point, you would need to ensure that the application can connect after the failover. SQL Server's downtime would be minimal. A local RTO or RPO may potentially be measured in minutes depending on the critical nature of the solution or system.

DR would be akin to bringing up a whole new data center. There are lots of pieces to the puzzle; SQL Server is just one component. Getting everything online may take hours or longer. This is why the RTOs and RPOs are usually separate. Even if a lot of the technologies and features used for HA and DR are the same, the level of effort and time involved may not be.

All RTOs and RPOs should be formally documented and revised periodically or as needed. Once they are documented, you can then consider what technologies and features you may use for the architecture.

## HADR Options for a Data Platform Solution

To envision a solution for VMs, you must first understand the availability options for IaaS-based deployments.

### Infrastructure-as-a-Service versus Platform-as-a-Service

When it comes to availability, the choice of IaaS or PaaS makes a difference. With IaaS, you have a virtual machine (VM), which means there is an operating system with an installation of SQL Server. The administrator or group responsible for SQL Server would have a choice of HADR solutions and a great deal of control over what how that solution was configured.

With PaaS-based deployments such as Azure SQL Database, the HADR solutions are built into the feature and often just need to be enabled. There are minimal options that can be configured.

Because of these differences, the choice of IaaS or PaaS may influence the final design of your HADR solution. For more information on IaaS and PaaS, refer back to Module 2.

### SQL Server HADR Features for Azure Virtual Machine

When using IaaS, you can use the features provided by SQL Server to increase availability. In some cases, they can be combined with Azure-level features to increase availability even further.

The features available in SQL Server are shown in the table below

Feature Name	Protects
Always On Failover Cluster Instance (FCI)	Instance
Always On Availability Group (AG)	Database
Log Shipping	Database

An instance of SQL Server is the entire installation of SQL Server (binaries, all the objects inside the instance including things like logins, SQL Server Agent jobs, and databases). Instance-level protection means that the entire instance is accounted for in the availability feature.

A database in SQL Server contains the data that end users and applications use. There are system databases that SQL Server relies on, as well as databases created for use by end users and applications. An instance of SQL Server always has its own system databases. Database-level protection means that anything that is captured in the transaction log for a user or application database is accounted for as part of the availability feature. Anything that exists outside of the database or that is not captured as part of the transaction log such as SQL Server Agent jobs and linked servers must be manually dealt with to ensure the destination server can function like the primary in the event of a planned or unplanned failover event.

Both FCIs and AGs require an underlying cluster mechanism. For SQL Server deployments running on Windows Server, it is a Windows Server Failover Cluster (WSFC) and for Linux it is Pacemaker. WSFCs will be discussed in an upcoming section after FCIs and AGs are formally introduced.

### Always On Failover Cluster Instances

An FCI is configured when SQL Server is installed. A standalone instance of SQL Server cannot be converted to an FCI. The FCI is assigned a unique name as well as an IP address that is different from the under-

lying servers, or nodes, participating in the cluster. The name and IP address must also be different from the underlying cluster mechanism. Applications and end users would use the unique name of the FCI for access. This abstraction enables applications to not have to know where the instance is running. One major difference between Azure-based FCIs versus on-premises FCIs, is that for Azure, an internal load balancer (ILB) is required. The ILB is used to help ensure applications and end users can connect to the FCI's unique name.

When an FCI fails over to another node of a cluster, whether it is initiated manually or happens due to a problem, the entire instance restarts on another node. That means the failover process is a full stop and start of SQL Server. Any applications or end users connected to the FCI will be disconnected during failover and only applications that can handle and recover from this interruption can reconnect automatically.

Upon starting up on the other node, the instance goes through the recovery process. The FCI will be consistent to the point of failure, so technically there will be no data loss but any transactions that need to be rolled back will do so as part of recovery. As noted above, because this is instance-level protection, everything necessary (logins, SQL Server Agent jobs, etc.) is already there so business can continue as usual once the databases are ready.

FCIs require one copy of a database, but that is also its single point of failure. To ensure another node can access the database, FCIs require some form of shared storage. For Windows Server-based architectures, this can be achieved via an Azure Premium File Share, iSCSI, Azure Shared Disk, Storage Spaces Direct (S2D), or a supported third-party solution like SIOS DataKeeper. FCIs using Standard Edition of SQL Server can have up to two nodes. FCIs also require the use of Active Directory Domain Services (AD DS) and Domain Name Services (DNS), so that means AD DS and DNS must be implemented somewhere in Azure for an FCI to work.

Using Windows Server 2016 or later, FCIs can use Storage Replica to create a native disaster recovery solution for FCIs without having to use another feature such as log shipping or AGs. You will learn more about AGs in the next section.

## Always On Availability Groups

AGs were introduced in SQL Server 2012 Enterprise Edition and as of SQL Server 2016, are also in Standard Edition. In Standard Edition, an AG can contain one database whereas in Enterprise Edition, an AG can have more than one database. While AGs share some similarities with FCIs, in most ways they are completely different.

The biggest difference between and FCI and an AG is that AGs provide database-level protection. The primary replica is the instance participating in an AG that contains the read/write databases. A secondary replica is where the primary sends transactions over the log transport to keep it synchronized. Data movement between a primary replica can be synchronous or asynchronous. The databases on any secondary replica are in a loading state which means they can receive transactions but cannot be a fully writeable copy until that replica becomes the primary. An AG in Standard Edition can have at most two replicas (one primary, one secondary) whereas Enterprise Edition supports up to nine (one primary, eight secondary). A secondary replica is initialized either from a backup of the database, or as of SQL Server 2016, you can use a features called 'automatic seeding'. Automatic seeding uses the log stream transport to stream the backup to the secondary replica for each database of the availability group using the configured endpoints.

An AG provides abstraction with the listener. The listener functions like the unique name assigned to an FCI and has its own name and IP address that is different from anything else (WSFC, node, etc.). The listener also requires an ILB and goes through a stop and start. Applications and end users can use the listener to connect, but unlike an FCI, if desired, the listener does not have to be used. Connections

directly to the instance can occur. With Enterprise Edition, secondary replicas in Enterprise Edition can also be configured for read-only access if desired and can be used for other functionality such as database consistency checks (DBCCs) and backups.

AGs can have a quicker failover time compared to an FCI, which is one reason they are attractive. While AGs do not require shared storage, each replica has a copy of the data which increases the total number of copies of the database and overall storage costs. The storage is local to each replica. For example, if the data footprint of the databases on the primary replica is 1TB, each replica will also have the same. If there are five replicas, that means you need 5TB of space.

Remember that any object that exists outside of the database or is not captured in the database's transaction log must manually be created and accounted for on any other SQL Server instance should that instance need to become the new primary replica. Examples of objects you would be responsible for include SQL Server Agent jobs, instance-level logins, and linked servers. If you can use Windows authentication or use contained databases with AGs, it will simplify access.

Many organizations may face challenges implementing highly available architectures, and may only need the high availability provided by the Azure platform, or using a PaaS solution like Azure SQL Managed Instance. Before we look at Azure platform solutions, there is one other SQL Server feature that you should know about: log shipping.

## Log Shipping

Log shipping has been around since the early days of SQL Server. The feature is based on backup, copy, and restore and is one of the simplest methods of achieving HADR for SQL Server. Log shipping is primarily used for disaster recovery, but it could also be used to enhance local availability.

Log shipping, like AGs, provides database-level protection which means you still need to account for SQL Server Agent jobs, linked servers, instance-level logins, etc. There is no abstraction provided natively by log shipping, so a switch to another server participating in log shipping must be able to tolerate a name change. If that is not possible, there are methods such as a DNS alias which can be configured at the network layer to try to mitigate the name change issues.

The log shipping mechanism is simple: first, take a full backup of the source database on the primary server, restore it in a loading state (STANDBY or NORECOVERY) on another instance known as a secondary server or warm standby. This new copy of the database is known as a secondary database. An automated process built into SQL Server will then automatically backup the primary database's transaction log, copy the backup to the standby server, and finally, restore the backup onto the standby.

The SQL Server HADR features are not the only options to enhance IaaS availability. There are some features in Azure that should also be considered.

## Azure HADR Features for Azure Virtual Machines

Azure provides three main options to enhance availability for IaaS deployments:

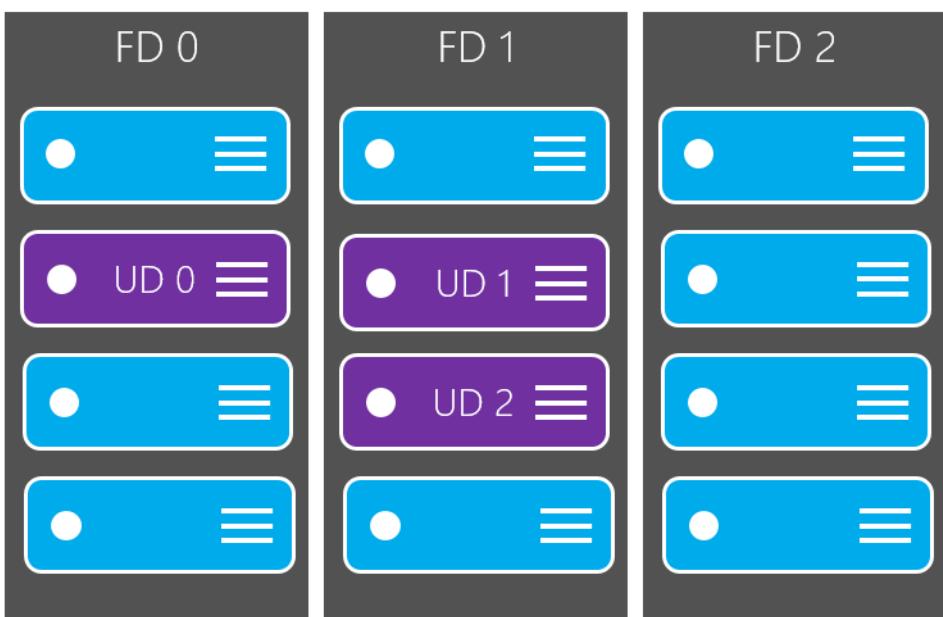
- Availability Sets
- Availability Zones
- Azure Site Recovery

All three of these options are external to the VM and do not know what kind of workload is running inside of it.

## Availability Sets

Availability Sets provide uptime against Azure related maintenance and single points of failure in a single data center. This was one of the first availability features introduced into the Azure platform, and effectively it can be thought of as anti-affinity rules for your VMs. This means if you had two SQL Server VMs in an availability set or log shipping pair, they would be guaranteed to never run on the same physical server.

Availability sets are separated into both fault domains and update domains to support both updates to the underlying Azure Infrastructure. Fault domains are sets of servers within a data center which use the same power source and network. There can be up to three fault domains in a data center as depicted in the picture below by FD 0, 1, and 2. Update domains, denoted by UD in Figure 1, indicate groups of virtual machines and underlying physical hardware that can be rebooted at the same time. Different update domains ensure separation.



*Figure 1: Fault Domains and Update Domains*

Availability sets and zones do not protect against in-guest failures, such as an OS or RDBMS crash; which is why you need to implement additional solutions such as AGs or FCIs to ensure you meet RTOs and RPOs. Both availability sets and zones are designed to limit the impact of environmental problems at the Azure level such as datacenter failure, physical hardware failure, network outages, and power interruptions.

For a multi-tier application, you should put each tier of the application into its own availability set. For example, if you were building a web application that has a SQL Server backend along with Active Directory Domain Services (AD DS), you would create an availability set for each tier (web, database, and AD DS).

Availability Sets is not the only way to separate IaaS VMs. Azure also provides Availability Zones, but the two cannot be combined. You can pick one or the other.

## Availability Zones

Availability zones account for data center-level failure in Azure. Each Azure region consists of many data centers with low latency network connections between them. When you deploy VM resources in a region

that supports Availability Zones, you have the option to deploy those resources into Zone 1,2, or 3.A zone is a unique physical location, that it, a data center, within an Azure region.

Zone numbers are logical representations. For example, if two Azure subscribers both deploy a VM into Zone 1 in their own subscriptions, that does not mean those VMs exist in the same physical Azure data center. Additionally, because of the distance there can be some additional latency introduced into zonal deployments. You should test the latency between your VMs to ensure that the latency meets performance targets. In most cases round trip latency will be less than 1 millisecond, which supports synchronous data movement in features like availability groups. You can also deploy Azure SQL Database into Availability Zones.

## Azure Site Recovery

Azure Site Recovery (ASR) provides enhanced availability for VMs at the Azure level and can work with VMs hosting SQL Server. ASR replicates a VM from one Azure region to another to create a disaster recovery solution for that VM. As noted earlier, this feature does not know that SQL Server is running in the VM and knows nothing about transactions. While ASR may meet RTO, it may not meet RPO since it is not accounting for where data is inside SQL Server. ASR has a stated monthly RTO of two hours. While most database professionals may prefer to use a database-based method for disaster recovery, ASR works well if it meets your RTO and RPO needs.

Now that you have learned about what can make IaaS deployments available, it is time to learn about the HADR options for PaaS deployments.

## HADR Options for PaaS Deployments

As noted earlier, PaaS is different when it comes to availability; you can only configure the options that Azure provides.

For the SQL Server-based options of Azure SQL Database and Azure SQL Database Managed Instance, the options are active geo-replication (Azure SQL Database only) and auto-failover groups (Azure SQL Database or Azure SQL Database Managed Instance). Both will be covered in more detail later in this module. Microsoft's documented service level agreements (SLAs) for RTO and RPO are linked in the For More Information section at the end of this module

Azure Database for MySQL has a service level agreement which guarantees an availability of 99.99, meaning nearly no downtime should be encountered. For Azure Database for MySQL, if a node-level problem happens such as hardware failure, a built-in failover mechanism will kick in. All transactional changes to the MySQL database are written synchronously to storage upon commit. If a node-level interruption occurs, the database server automatically creates a new node and attaches the data storage.

From an application standpoint, you will need to code the necessary retry logic because all connections are dropped as part of spinning up the new node and any in flight transactions are lost. This is considered a best practice for any cloud application, as they should be designed to handle transient failures.

Azure Database for PostgreSQL uses a similar model to MySQL in its standard deployment model; however, Azure PostgreSQL also offers a scale-out hyperscale solution called Citus. Citus provides both scale-out and additional high availability for a server group. If enabled, a standby replica is configured for every node of a server group which would also increase cost since it would double the number of servers in the group. In the event the original node has a problem such as becoming unresponsive or failing completely, the standby takes its place. The data is kept in sync via PostgreSQL synchronous streaming replication.

As with Azure Database for MySQL, solutions that use Azure Database for PostgreSQL must also include retry logic in the application because of dropped connections and loss of in-flight transactions.

Both Azure Database for MySQL and PostgreSQL support the option for a read replica. This means a replica can be used for activities like reporting to offload work from the primary database. A read replica also enhances availability because it exists in another region.

## Devise a HADR Strategy

Now that you understand RTOs and RPOs as well as the different features both in SQL Server as well as Azure to increase availability, you can put all of that together and design a solution to meet your HADR requirements.

## Hybrid Solutions

While an architecture can be completely deployed in one or more Azure regions, many organizations will need or want to have solutions that span both on premises and Azure, or possibly Azure to another public cloud. This type of architecture is known as a hybrid solution.

PaaS solutions by nature are not designed to allow traditional hybrid solutions. HADR is provided by the Azure infrastructure. There are some exceptions. For example, SQL Server's transactional replication feature can be configured from a publisher located on premises (or another cloud) to an Azure SQL Database Managed Instance subscriber, but not the other way.

Hybrid solutions are therefore IaaS-based since they rely on traditional infrastructure. Hybrid solutions are extremely useful. Not only can they be used to help migrate to Azure, but the most common usage of a hybrid architecture is to create a robust disaster recovery solution for an on-premises system. For example, a secondary replica for an AG can be added in Azure. That means any associated infrastructure must exist, such as AD DS and DNS.

Arguably the most important consideration for a hybrid HADR solution that extends to Azure is networking. Not having the right bandwidth could mean missing your RTO and RPO. Azure has a fast networking option called ExpressRoute. If ExpressRoute is not something your company can or will implement, configure a secure site-to-site VPN so that the Azure VMs will act as an extension of your on-premises infrastructure. Exposing IaaS VMs directly to the public internet is strongly discouraged.

Although not traditionally thought of as hybrid, Azure can also be used as the destination for a database backup and as cold, archival storage for backups. Database backup and recovery will be covered later in this module.

## Example IaaS HADR Solutions

There are many different combinations of features that could be deployed in Azure for IaaS. This section will cover five common examples of SQL Server HADR architectures in Azure. All PaaS information will be covered later.

### Single Region High Availability Example 1 – Always On Availability Groups

If you only need high availability and not disaster recovery, configuring an AG is one of the most ubiquitous methods no matter where you are using SQL Server. More detail around considerations and planning AGs in Azure is covered in the next section. Figure 2 is an example of what one possible AG in a single region could look like.

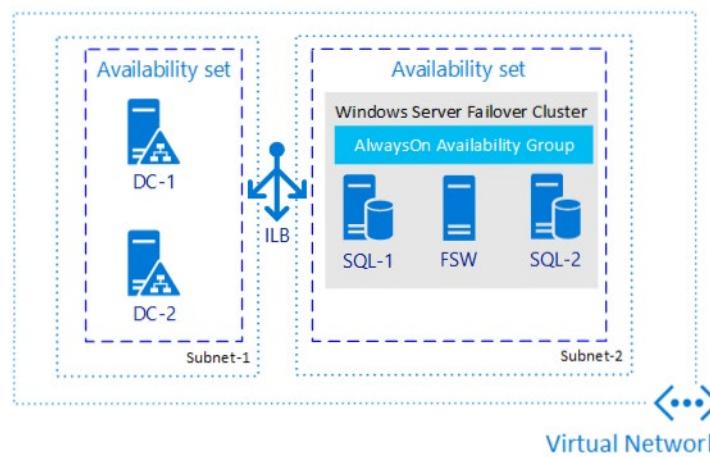


Figure 2: An Availability Group in a single region

Why is this architecture worth considering?

- This architecture protects data by having more than one copy on different VMs.
- This architecture allows you to meet RTO and RPO with minimal-to-no data loss if implemented properly.
- This architecture provides an easy, standardized method for applications to access both primary and secondary replicas (if things like read only replicas will be used).
- This architecture provides enhanced availability during patching scenarios.
- This architecture needs no shared storage, so there is less complication than when using an FCI.

## Single Region High Availability Example 2 – Always On Failover Cluster Instance

Until AGs were introduced, FCIs were the most popular way to implement SQL Server high availability. FCIs, however, were designed when physical deployments were dominant. In a virtualized world, FCIs do not provide a lot of the same protections in the way they would on physical hardware because it is extremely rare for a VM to have a problem. FCIs were designed to protect against things like network card failure or disk failure, both of which would likely not happen in Azure.

Having said that, FCIs do have a place in Azure. They work, and as long as you have the right expectations about what is and is not provided, an FCI is a perfectly acceptable solution. Figure 3 below, from the Microsoft documentation, shows a high-level view of what an FCI deployment looks like when using Storage Spaces Direct.

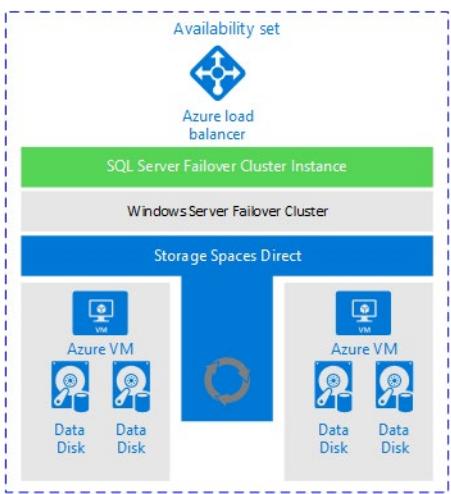


Figure 3: A FCI deployment using Storage Spaces Direct

Why is this architecture worth considering?

- FCIs are still a popular availability solution.
- The shared storage story is improving with feature like Azure Shared Disk.
- This architecture meets most RTO and RPO for HA (although DR is not handled).
- This architecture provides an easy, standardized method for applications to access the clustered instance of SQL Server.
- This architecture provides enhanced availability during patching scenarios.

## Disaster Recovery Example 1 – Multi-Region or Hybrid Always On Availability Group

If you are using AGs, one option is to configure the AG across multiple Azure regions or potentially as a hybrid architecture. This means that all nodes which contain the replicas participate in the same WSFC. This assumes good network connectivity, especially if this is a hybrid configuration. One of the biggest considerations would be the witness resource for the WSFC which will be discussed in an upcoming section. This architecture would require AD DS and DNS to be available in every region and potentially on premises as well if this is a hybrid solution. Figure 4 shows what a single AG configured over two locations looks like using Windows Server.

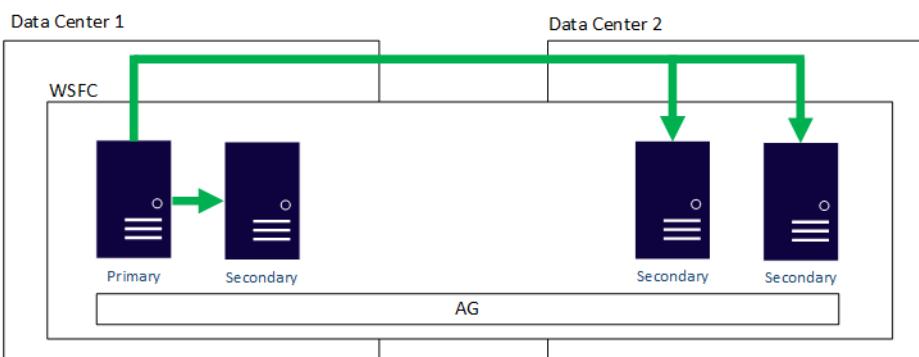


Figure 4: A single AG configured over two locations

Why is this architecture worth considering?

- This architecture is a proven solution; it is no different than having two data centers today in an AG topology.
- This architecture works with Standard and Enterprise editions of SQL Server.
- AGs naturally provide redundancy with additional copies of data.
- This architecture makes use of one feature that provides both HA and D/R

## Disaster Recovery Example 2 –Distributed Availability Group

A distributed AG is an Enterprise Edition only feature introduced in SQL Server 2016. It is different than a traditional AG. Instead of having one underlying WSFC where all of nodes contain replicas participating in one AG as described in the previous example, a distributed AG is made up of multiple AGs. The primary replica containing the read write database is known as the global primary. The primary of the second AG is known as a forwarder and keeps the secondary replica(s) of that AG in sync. In essence, this is an AG of AGs.

This architecture makes it easier to deal with things like quorum (discussed in more depth in the next section) since each cluster would maintain its own quorum, meaning it also has its own witness. A distributed AG would work whether you are using Azure for all resources, or if you are using a hybrid architecture.

Figure 5 shows an example distributed AG configuration. There are two WSFCs. Imagine each is in a different Azure region or one is on premises and the other is in Azure. Each WSFC has an AG with two replicas. The global primary in AG 1 is keeping the secondary of replica of AG 1 synchronized as well as the forwarder, which also is the primary of AG 2. That replica keeps the secondary replica of AG 2 synchronized.

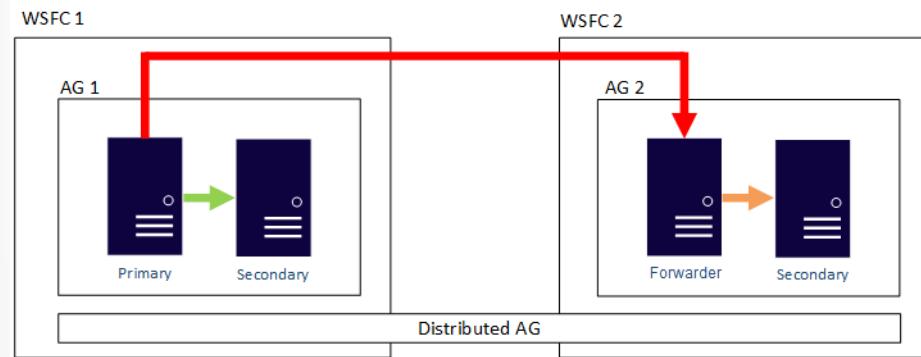


Figure 5: An example distributed AG configuration

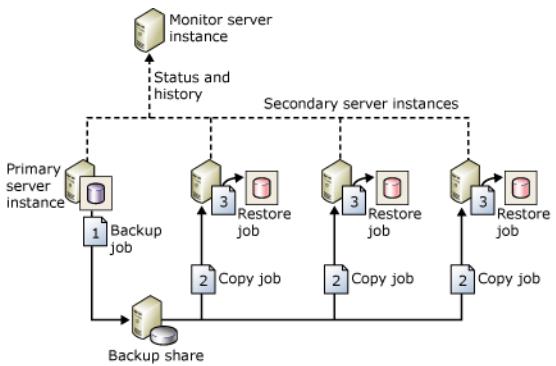
Why is this architecture worth considering?

- This architecture separates out the WSFC as a single point of failure if all nodes lose communication
- In this architecture, one primary is not synchronizing all secondary replicas.
- This architecture can provide failing back from one location to another.

## Disaster Recovery Example 3 –Log Shipping

Log shipping is one of the oldest HADR methods for configuring disaster recovery for SQL Server. As described above, the unit of measurement is the transaction log backup. Unless the switch to a warm

standby is planned to ensure no data loss, data loss will most likely occur. When it comes to disaster recovery, it is always best to assume some data loss even if minimal. Figure 6, from the Microsoft documentation, shows an example log shipping topology.



Why is this architecture worth considering?

- Log shipping is a tried-and-true feature that has been around for over 20 years
- Log shipping is easy to deploy and administer since it is based on backup and restore.
- Log shipping is tolerant of networks that are not robust.
- Log shipping meets most RTO and RPO goals for DR.
- Log shipping is a good way to protect FCIs.

## Disaster Recovery Example 4 – Azure Site Recovery

For those who do not want to implement a SQL Server-based disaster solution, ASR is a potential option. However, most data professionals prefer a database-centric approach as it will generally have a lower RPO.

Figure 7 below, from the Microsoft documentation, shows where in the Azure Portal you would configure replication for ASR.

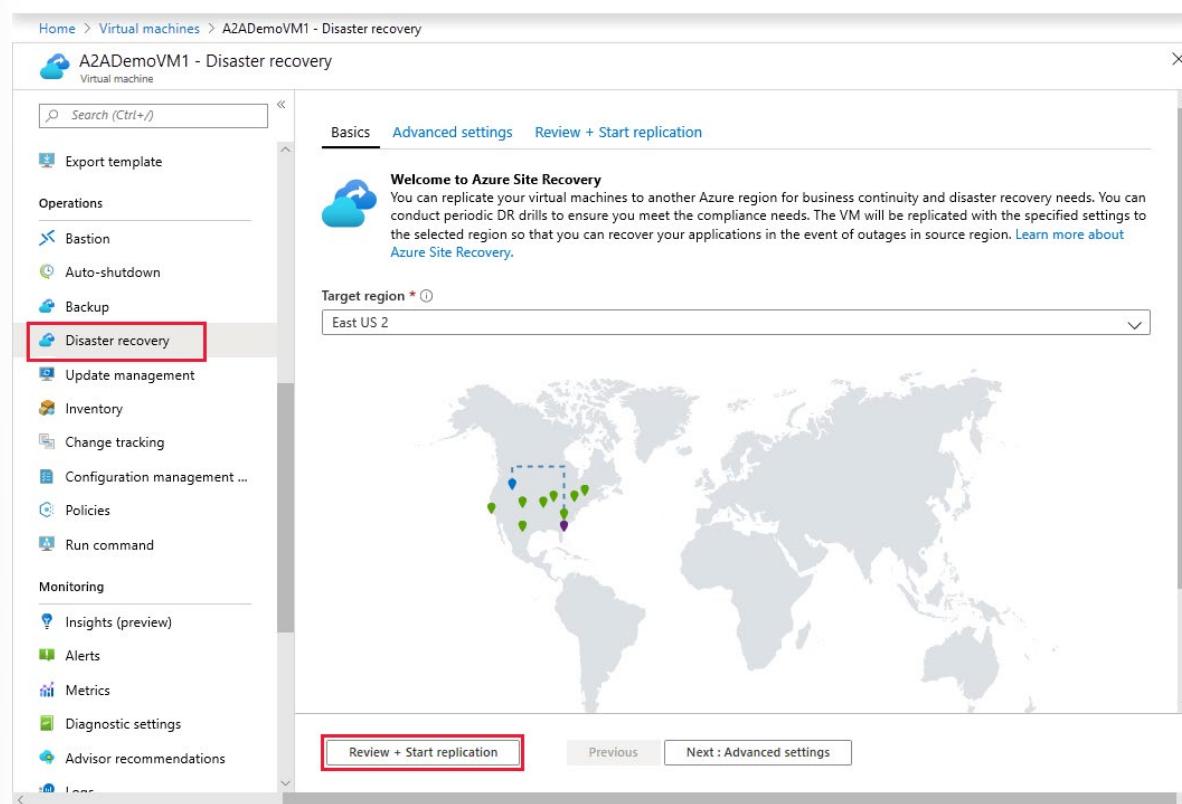


Figure 7: Configuring Azure Site Recovery

Why is this architecture worth considering?

- ASR will work with more than just SQL Server.
- ASR may meet RTO and possibly RPO.
- ASR is provided as part of the Azure platform

## Summary and Knowledge Check

Deploying the right HADR solution in Azure is more than just picking a feature. A solution must meet the requirements, and specifically, the RTO and RPO expected by the business. Depending on the path (IaaS or PaaS), there could be multiple options to choose from.

### Question 1

What is RPO?

- The number of nodes in a cluster
- The point to which data needs to be recovered after a failure
- A partial database restore

## Question 2

*What is a hybrid solution?*

- A solution that has resources both in Azure as well as on premises or in another cloud provider
- A solution that uses two different database engines e.g. MySQL and SQL Server
- A solution that spans two different versions of SQL Server

## Question 3

*What is available after failover with database-level protection in SQL Server?*

- Logins, Databases, and SQL Server Agent Job
- Databases and SQL Server Agent jobs
- Whatever is in the databases; anything outside must be dealt with manually

# IaaS Platform and Database Tools for HADR

## Introduction

If you are using IaaS for deploying your database solution, there are considerations for properly deploying HADR.

Suppose you want to use an AG. How is deploying a WSFC different? Are there any specific considerations for AGs that differ from an on-premises solution?

By the end of this lesson, you will understand what is needed to be able to deploy IaaS database platform solutions in Azure.

## Learning Objectives

In this lesson, you will learn:

- What to consider when deploying a WSFC in Azure
- What to consider when deploying an AG in Azure

## Failover Clustering in Windows Server

For all availability configurations of AGs, an underlying cluster is required. There are many aspects of setting up a cluster that you need to be aware of.

## Considerations for a Windows Server Failover Cluster in Azure

Deploying a WSFC in Azure is similar to configuring one on premises. There are some Azure-specific considerations which is what this section will focus on.

One of the most important aspects is deciding what to use for a witness resource. Witness is a core component of the quorum mechanism. Quorum is what helps ensure that everything in the WSFC stays up and running. If you lose quorum, the WSFC will go down taking an AG or FCI with it. The witness resource can be a disk, file share (SMB 2.0 or later), or cloud witness. It is recommended that you use a cloud witness as it is fully Azure-based, especially for solutions that will span multiple Azure regions or are hybrid. The cloud witness feature is available in Windows Server 2016 and later.

The next consideration is the Microsoft Distributed Transaction Coordinator (DTC or MSDTC). Some applications use it, but the majority do not. If you require DTC and are deploying an AG or FCI, you should cluster DTC. Clustering DTC requires a shared disk to work properly even though you may not require one otherwise, as in the case of an AG.

Most WSFC deployments require the use of both AD DS and DNS; FCIs always do. AGs can be configured without requiring AD DS but still needing DNS. In Windows Server 2016, there is a variant of a WSFC called a Workgroup Cluster, which can be used with AGs only. Additional considerations for this configuration will be discussed in the next section.

The WSFC itself needs a unique name in the domain (and DNS) and requires an object in AD DS called the cluster name object (CNO). Anything created in context of the WSFC that has a name will require a unique name, as well as at least one IP address. If the configuration will stay in a single region, IP addresses will be in a single subnet. If the WSFC will span multiple regions, more than one IP address will be associated with the WSFC, as well as an AG if it spans multiple regions as part of the WSFC.

A typical Azure-based WSFC will only require a single virtual network card (vNIC). Unlike the setup for an on-premises physical or virtual server, the IP address for the vNIC has to be configured in Azure, not in the VM. That means inside the VM it will show up as a dynamic (DHCP) address. This is expected, however cluster validation (covered in an upcoming section) will generate a warning that can safely be ignored.

Considerations for the WSFC's IP address is also different from on premises. There is no way to reserve the IP address at the Azure level since it is fully maintained within the guest configuration. This means that if subsequent resources that use an IP address in Azure use DHCP, you must check for conflicts.

## The Failover Clustering Feature

Before a WSFC can be configured, the underlying Windows feature must be enabled on every node that will participate in the WSFC. This can be done in one of two ways: using the Server Manager or via PowerShell.

In Server Manager, Failover Clustering must be enabled in the Add Roles and Features Wizard. The feature is highlighted in Figure 8 below.

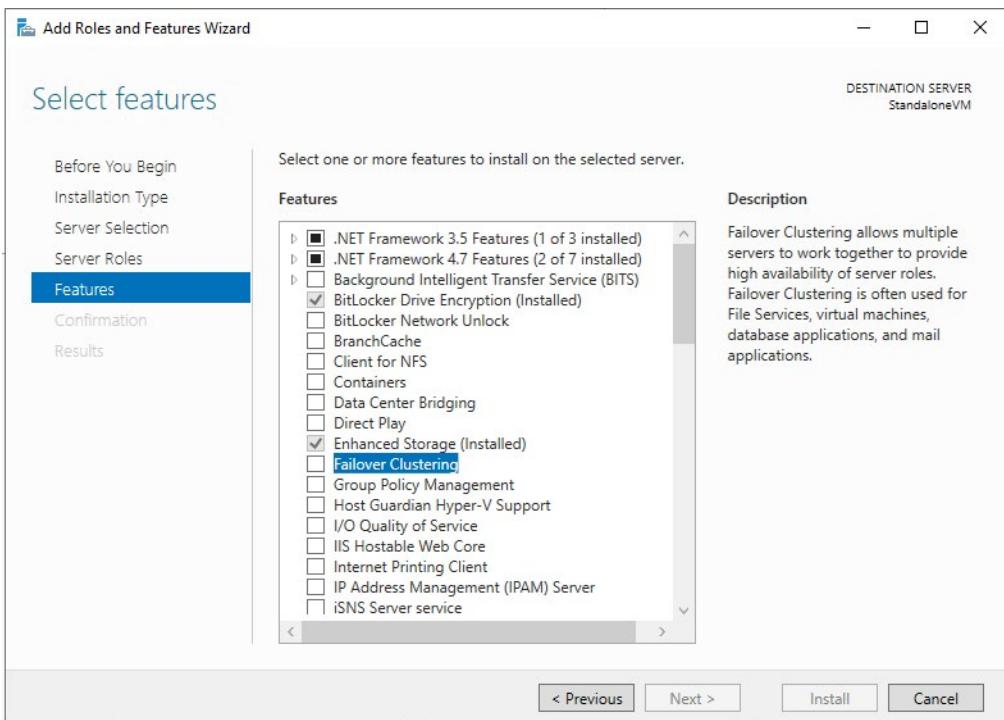


Figure 8: Using the Add Roles and Features Wizard to add the Failover Clustering feature

To enable the feature via PowerShell, use the following command which will also install the utilities which will be used to administer a WSFC:

```
Install-WindowsFeature Failover-Clustering -IncludeManagementTools
```

Once the feature is installed on the servers targeted as WSFC nodes, you must then validate the configuration.

## Cluster Validation

For a WSFC to be considered supported, it must pass cluster validation. Cluster validation is a built-in process that checks the nodes via a series of tests to ensure that the entire configuration is suitable for clustering. After running validation, each of the tests will come back with an error, a warning, a pass, or a message that the test is not applicable. Warnings are acceptable if that condition is expected in your environment. If not, it should be addressed. All errors must be resolved.

Validation can be run via Failover Cluster Manager or by using the `Test-Cluster` PowerShell cmdlet.

For FCIs, these tests also check the shared storage to ensure that the disks are configured correctly. For AGs with no shared storage, in Windows Server 2016 and later, the results will come back as not applicable. For Windows Server 2012 R2, the disk tests will show a warning when there are no shared disks. This state is expected.

Once the configuration is deemed valid by the cluster validation process, you can then create the WSFC.

## Create a Windows Server Failover Cluster

To create a WSFC properly in Azure, you cannot use the Wizard in Failover Cluster Manager for FCIs or AGs deployed using Windows Server 2016 and earlier. Due to the DHCP issue mentioned earlier, currently the only way to create the WSFC is to use PowerShell and specify the IP address. This could change in future versions of Windows Server.

For a configuration that has shared storage, use the following syntax:

```
New-Cluster -Name MyWSFC -Node Node1,Node2,...,NodeN -StaticAddress w.x.y.z
```

where `MyWSFC` is the name of the WSFC you want, `Node1,Node2,...,NodeN` are the names of the nodes that will participate in the WSFC, and `w.x.y.z` is the IP address of the WSFC (for example: 10.252.1.100). If you are creating a WSFC that spans multiple subnets, you can specify more than one IP address for `-StaticAddress` separated by commas.

For a configuration that does not have shared storage, add the `-NoStorage` option.

```
New-Cluster -Name MyWSFC -Node Node1,Node2,...,NodeN -StaticAddress w.x.y.z  
-NoStorage
```

For a Workgroup Cluster that will only use DNS, the syntax is also slightly different.

```
New-Cluster -Name MyWSFC -Node Node1,Node2,...,NodeN -StaticAddress w.x.y.z  
-NoStorage -AdministrativeAccessPoint DNS
```

Windows Server 2019 by default will use a distributed network name for IaaS. A distributed network name is one that creates just a network name, but the IP address is tied to the underlying nodes. You no longer need to specify an IP address as shown above if it is not needed or necessary. A distributed network name is for the WSFC's name only; it cannot be used with the name of an AG or FCI.

The WSFC creation mechanism in Windows Server 2019 detects if it is running in Azure or not and will create the cluster using a distributed network name unless you tell it to do something else. Currently, distributed network names are incompatible with FCIs and while they do work with AGs, if you encounter an issue, you may want to consider deploying a WSFC traditionally using PowerShell. You need to add one more option: `-ManagementPointNetwork` with a value of `Singleton`. An example would look like this:

```
New-Cluster -Name MyWSFC -Node Node1,Node2,...,NodeN -StaticAddress w.x.y.z  
-NoStorage -ManagementPointNetwork Singleton
```

For a Workgroup Cluster, you will need to ensure the name and IP address(es) are in DNS for any name or IP address created in the context of the WSFC such as the WSFC itself, an FCI name and IP address, and an AG listener name and IP address.

With Windows Server 2019, Microsoft changed how WSFCs are created by default in Azure. Instead of creating a network name and an IP address, it uses a distributed network name. This is not yet supported with either of the Always On features, so it is still required to create the WSFC using the method described above.

## Test the Windows Server Failover Cluster

After the WSFC is created but before configuring FCIs or AGs, test that the WSFC is working properly. For clusters that require shared storage, such as for those supporting a SQL Server Failover Cluster Instance, it is important that you test the ability for all of the nodes in the cluster to access any shared storage, and to take ownership of the storage as they would in a failover. You can do this by clicking on Storage, then Disks in Failover Cluster Manager, as shown in Figure 9. If you right click on each clustered disk device, you will see the option **Move Available Storage**. If you choose the **Select Node** option you can force the storage to fail over to the other nodes in the cluster to confirm this functionality.

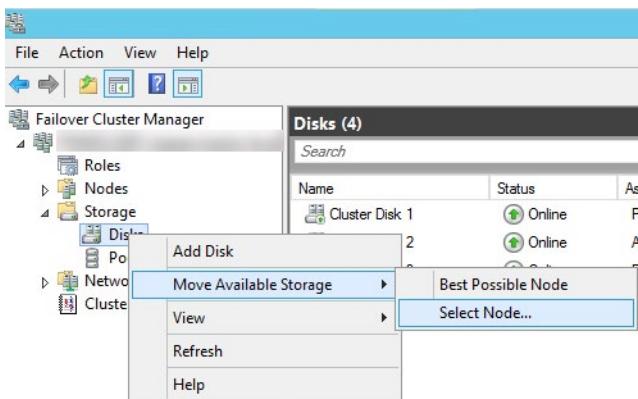


Figure 9: Testing the ability of nodes to access shared storage

Now that you understand the considerations for a WSFC in Azure, let's look at the considerations for deploying the Always On features on top of a WSFC in Azure.

## Always On Availability Groups

For all availability configurations of AGs, an underlying cluster is required, whether or not it uses AD DS. By the end of this section, you will understand the considerations for deploying an AG in Azure.

## Considerations for Always On Availability Groups in Azure

Configuring an AG is nearly the same in Azure as it is on premises as are most of the considerations, such as how to initialize secondary replicas. Most of the Azure-specific considerations were discussed earlier, such as needing an ILB. Same as the WSFC itself, you cannot reserve the listener's IP address in Azure so you need to ensure something else does not come along and grab it otherwise there could be a conflict on the network which in turn could cause availability headaches.

As mentioned earlier, do not place any permanent database on the ephemeral storage. All VMs that are participating in an AG should have the same storage configuration. You must size disks appropriately for performance depending on the application workload.

## Enable the Availability Groups Feature

Before an AG can be configured, the AG feature must be enabled. This can be done in SQL Server Configuration Manager as shown in Figure 10 or via PowerShell with the cmdlet **Enable-SqlAlwaysOn**<sup>1</sup>. Enabling the AG feature will require a stop and start of the SQL Server service.

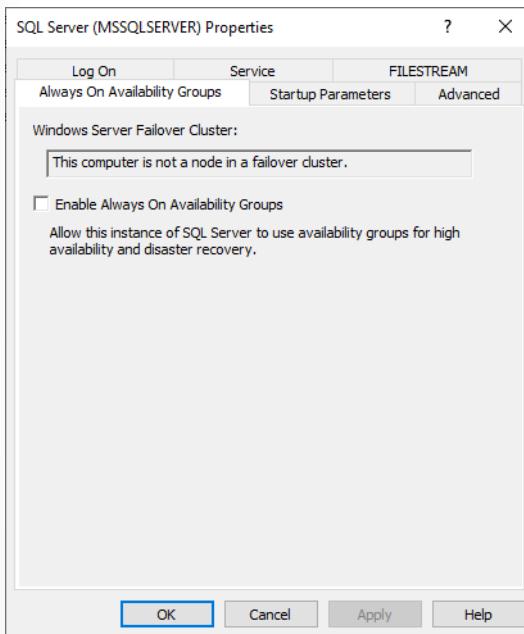


Figure 10: Enabling the Availability Groups Feature in SQL Server Configuration Manager

## Create the Availability Group

Creating an AG in Azure is the same as it is on premises. SQL Server Management Studio (SSMS), T-SQL, or PowerShell can be used. Full instructions on each of these methods can be found in the links at the end of the module.

The only difference is that whether or not you create the listener as part of the initial AG configuration, as the listener requires the creation of an Azure load balancer and has some additional configuration in the WSFC related to the load balancer.

## Create an Internal Azure Load Balancer

Once the listener is created, an internal load balancer (ILB) must be used. Without configuring an ILB, applications, end users, administrators, and others cannot use the listener unless they were actually connected to the VM that hosts an AG's primary replica. For full instructions on how to configure an ILB, see the links at the end of the module.

You can use a basic or a standard load balancer depending on your preference or configuration. Note that deployments using Availability Zones require the use of a standard load balancer. The listener IP

---

<sup>1</sup> <https://docs.microsoft.com/en-us/powershell/module/sqlps/enable-sqlalwayson?view=sqlserver-ps>

address and the port used for the listener are what is configured as part of the load balancer. A single load balancer supports more than one IP address, so depending on your standards, you may not need a different load balancer for each AG configured on those nodes.

Another consideration for the load balancer is the probe port. Without the probe port, the listener will not work properly as it is not enough just to create the load balancer. Each IP address that will use the load balancer requires a unique probe port. If there are going to be two listeners, there must be two probe ports. Probe ports are generally high numbers such as 59999.

The probe port is set on the IP address(es) associated with the listener with the following syntax:

```
Get-ClusterResource IPAddressResourceNameForListener | Set-ClusterParameter  
ProbePort PortNumber
```

Adding the probe port will require a stop and start of the IP address of the listener which will also temporarily cause the AG to be brought offline, so it is best to get this configured before deploying in production.

If you have a multi-subnet configuration, a load balancer will need to be configured in each subnet (whether or not the other subnet is deployed to different region) and the probe port for that region associated with the IP resource for that subnet in the WSFC.

Without directly connecting to the listener, the only way to ensure that the listener is configured correctly is to use the PowerShell cmdlet `Test-NetConnection` with the specific port. The syntax is as follows:

```
Test-NetConnection NameOrIPAddress -Port PortNumber
```

You should run this command from somewhere other than the VM that is hosting the primary replica.

Some environments may also require that the IP address for the WSFC and selected ports (such as 445) must be accessible for administration or other purposes, which means configuring those as part of the same or a different load balancer.

Once the load balancer is confirmed to be working, you can begin to test AG failover and connectivity to the AG via the listener.

## Distributed Availability Groups

Planning for and configuring a distributed AG is the same on premises as it is in Azure, with any Azure-specific considerations for the individual AGs. The main difference between an on-premises configuration and an Azure configuration for a distributed AG is that as part of the load balancer configuration in each region, the endpoint port for the AG needs to be added. The default port is 5022.

## Azure Site Recovery

ASR is an option that works with the Virtual Machine, whether or not SQL Server is running inside of it. It works with SQL Server but is not designed specifically to account for nuances that may be required when you have a specific RPO. The disks of a VM configured to use ASR are replicated to another region. This replication can be seen Figure 11, noted by the "Data flow" arrow.

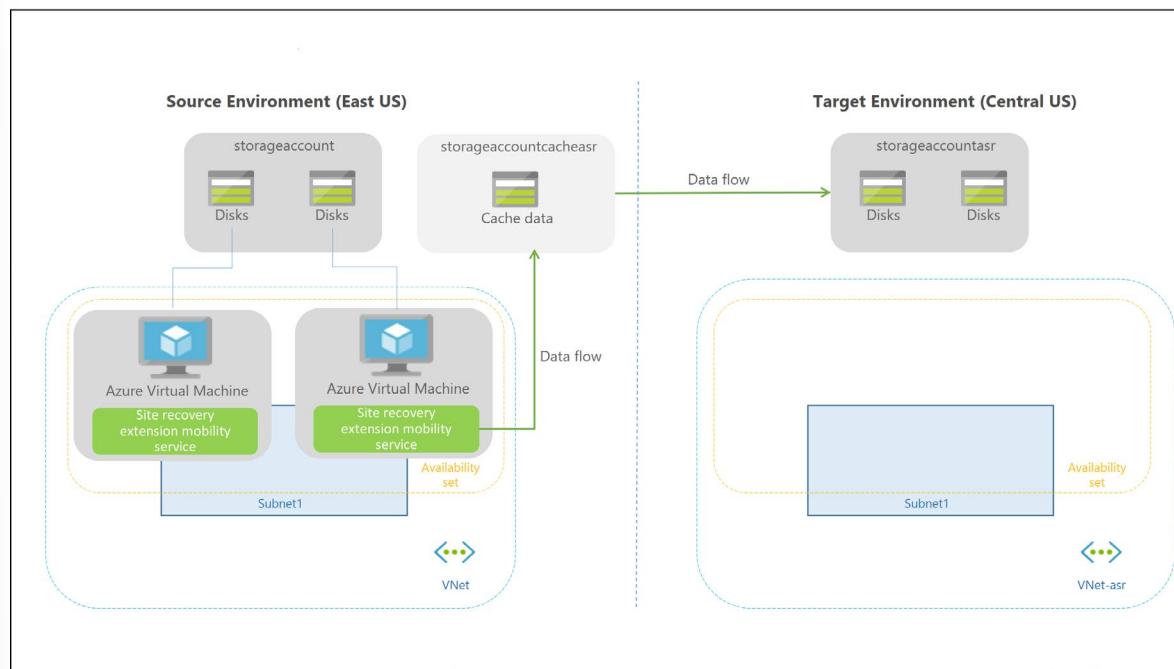


Figure 11: Replication of disks configured to use Azure Site Recovery

This means that all changes to a disk are replicated as soon as they occur, but this process knows nothing of database transactions. This is why recovering to a specific data point may not be possible with ASR in the same way it is for a SQL Server-centric solution such as when using an AG.

If it is not possible to deploy one of the in-guest options for IaaS solutions, ASR is a viable option to manage disaster recovery.

Additionally, ASR can potentially protect you against ransomware. If infected, you could roll the VM back to a point before the infection was introduced. That could also mean data loss from a SQL Server perspective, but some data loss, especially in this case, may be more than acceptable. Up and running is often better than down for hours, days, or weeks trying to remove ransomware from your network.

The key things to know when replication is enabled on a VM:

- There is a Site Recovery Mobility extension configured on the VM.
- Changes are sent continually unless ASR is unconfigured or replication is disabled
- Crash consistent recovery points are generated every five minutes, and application specific recovery points are generated according to what is configured in the replication policy. An example replication policy is shown in Figure 12.

The screenshot shows two windows side-by-side. The left window is titled 'Replication policies' and lists two entries: 'Tier1-policy-failback' and 'Tier1-policy'. The right window is titled 'Tier1-policy' and shows its configuration details under 'Replication settings' and 'Associated Configuration Servers'.

NAME	SOURCE TYPE	TARGET TYPE	STATUS
Tier1-policy-failback	Azure	VMware	Not in use
Tier1-policy	VMware / Physical ...	Azure	Not in use

**Replication settings**

- Source type: VMware / Physical machines
- Target type: Azure
- RPO threshold: 15 Minutes
- Recovery point retention: 24 Hours
- App consistent snapshot frequency: 60 Minutes

**Associated Configuration Servers**

NAME	ASSOCIATION STATUS
V2AGNK-CS	Associated

Figure 12: An example replication policy

For SQL Server, the 'App consistent snapshot frequency' value is what you may want to adjust to reduce your RPO. However, due to the nature of how ASR works – it is using Volume Shadow Service (VSS) – lowering this value could potentially cause problems for SQL Server since there is a brief freeze and thaw of I/O when the snapshots are taken. The impact of the freeze and thaw could be magnified if other options such as an AG are configured. Most will not encounter issues, but if ASR interferes with SQL Server, you may want to consider other availability options.

If multiple VMs are part of an overall solution, they can be replicated together to create a shared crash-and application-consistent recovery points. This is known as multi-VM consistency and will impact performance. Unless VMs must be restored in this way, it is recommended not to configure this option.

One major benefit of ASR is that you can test disaster recovery without needing to bring down production.

A consideration for ASR is that in the event of a failover to another region, the replica VMs are not protected when they are brought online. They will have to be re-protected.

## Summary and Knowledge Check

Deploying the right HADR solution in Azure is more than just picking a feature. A solution must meet your requirements, including the RTO and RPO expected by the business. Depending on the platform (IaaS or PaaS), there could be multiple options to choose from.

### Question 1

*What component in Azure needs to be configured for the listener in an AG to work properly?*

- The NIC
- The NSG
- A load balancer

## Question 2

*How should you create a WSFC in Azure for AGs and FCIs?*

- Wizard in Failover Cluster Manager
- PowerShell
- WMI

## Question 3

*What Azure feature lets you test disaster recovery without bringing down your production system?*

- ASR
- Azure SQL Database
- Azure Load Balancer

# PaaS Platform and Database Tools for HADR

## Introduction

Implementing a PaaS-based HADR solution for PaaS solutions is different than IaaS. As noted above, for IaaS, the configuration is mainly done at the Azure level. For PaaS, the solutions that allow your applications and data to be highly available and meet your RTOs and RPOs are configured within the database or database server.

### Learning Objectives

After completing this section, you will understand

- Temporal Tables
- Active-geo replication
- Auto-failover groups

## Temporal Tables for Azure SQL Database

Azure SQL Database allows you to track and analyze the changes to your data using a feature called Temporal Tables. This feature requires that the tables themselves be converted to be temporal, which means the table will have special properties and will also have a corresponding history table. The Temporal Table feature allows you to use the history table to recover data that may have been deleted or updated. Recovering data from the history table is a manual process involving Transact-SQL, but could be very helpful in certain scenarios such as if a user accidentally deletes important data that the business needs.

## Active Geo-Replication for Azure SQL Database

Another method to increase availability for Azure SQL Database is to use active geo-replication. Active geo-replication creates a replica of the database in another region that is synchronously kept up to date. That replica is also readable, similar to an AG in IaaS. Underneath the surface, Azure is using the same methods as an AG which is why some of the terminology and functionality is similar(primary and secondary logical servers, read only databases, etc.) as can be seen in Figure 13.

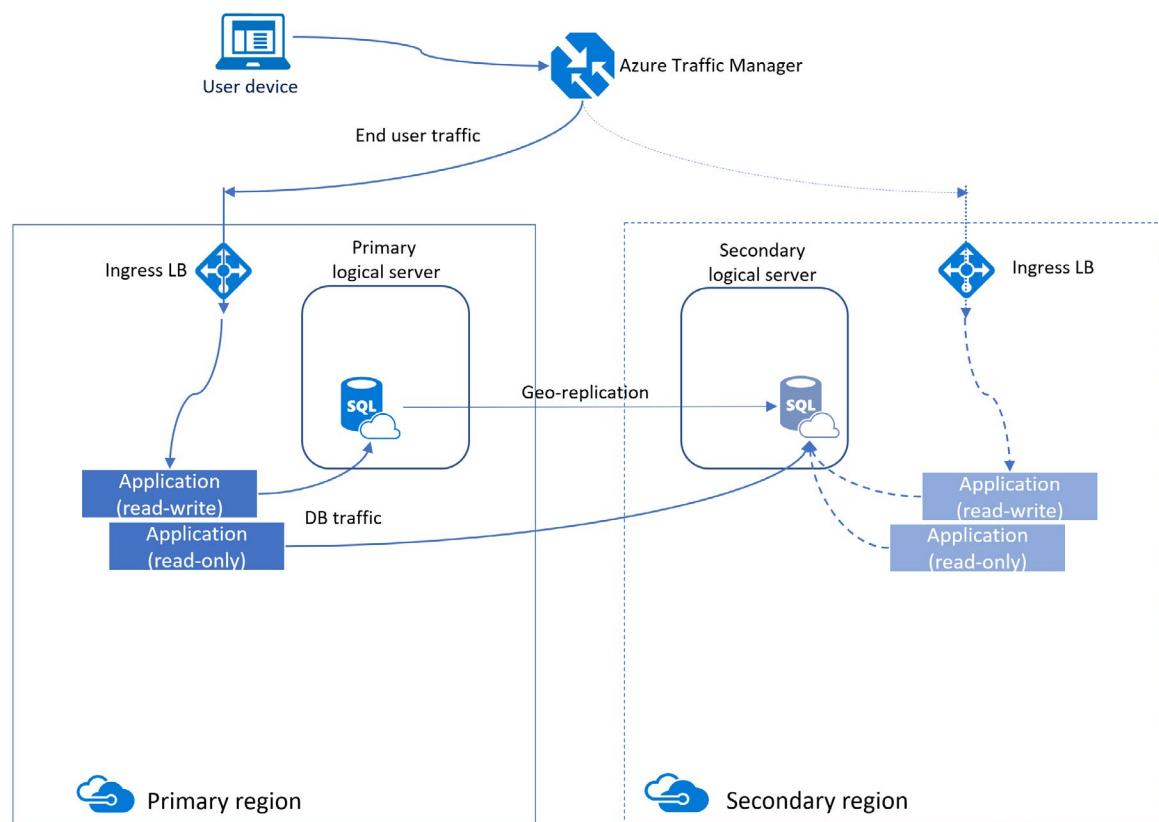


Figure 13: Active Geo-Replication for Azure SQL Database

More details can be found in the [official documentation<sup>2</sup>](#).

## Auto-Failover Groups for Azure SQL Database and Azure SQL Database Managed Instance

An auto-failover group is an availability feature that can be used with both Azure SQL Database and Azure SQL Database Managed Instance. Auto-failover groups let you manage how databases on an Azure SQL Database server or databases in Azure SQL Database Managed Instance are replicated to another region, and let you manage how failover could happen. The name assigned to the auto-failover group must be unique within the \*.database.windows.net domain. Azure SQL Database Managed Instance only supports one auto-failover group.

Auto-failover groups provide AG-like functionality called a listener, which allows both read-write and read-only activity. This functionality can be seen in Figure 14 which is slightly different than the one for active geo-replication. There are two different kinds of listeners: one for read-write and one for read-only traffic. Behind the scenes in a failover, DNS is updated so clients will be able to point to the abstracted listener name and not need to know anything else. The database server containing the read-write copies is the primary, and the server that is receiving the transactions from the primary is a secondary.

<sup>2</sup> <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-active-gc-replication>

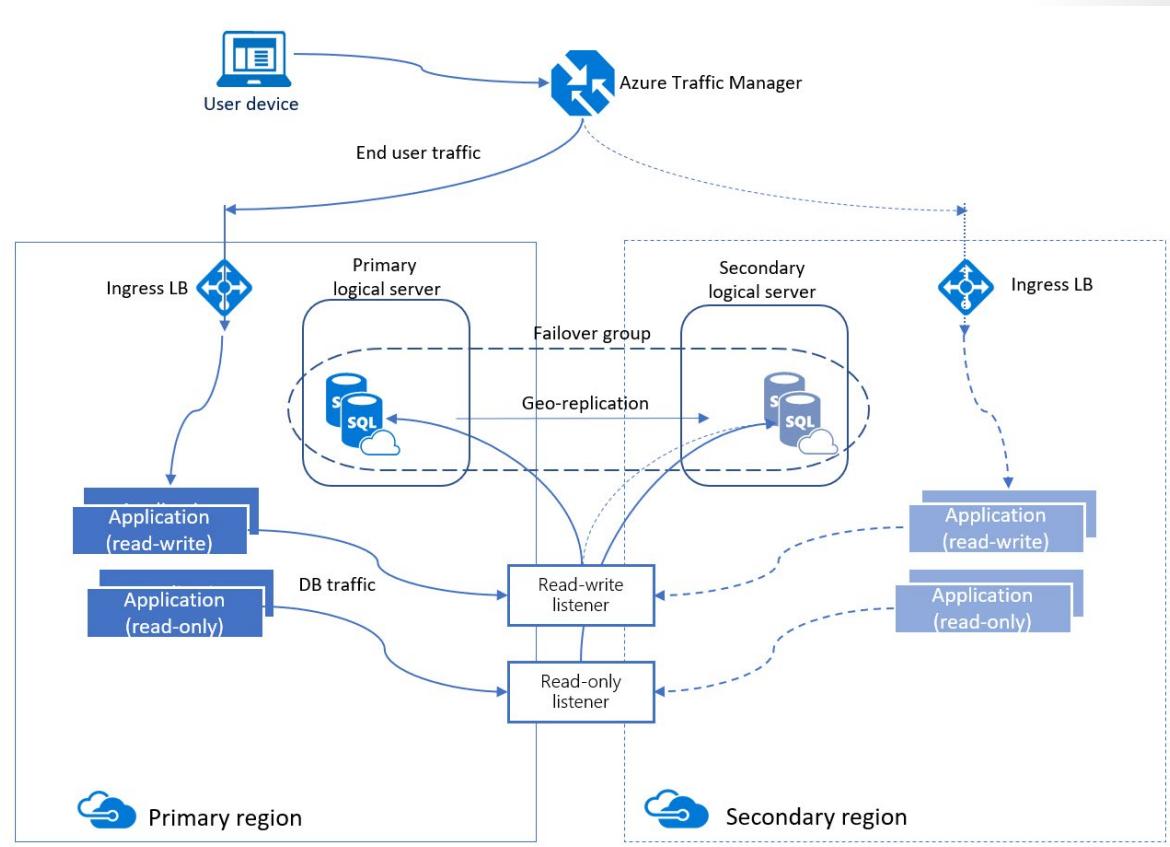


Figure 14: Auto-Failover Groups for Azure SQL Database and Azure SQL Database Managed Instance

When it comes to failover, auto-failover groups have two different policies that can be configured.

- Automatic – By default, when a failure occurs and it is determined that a failover must happen, the auto-failover group will switch regions. The ability to fail over automatically can be disabled.
- Read-Only – By default, if a failover occurs, the read-only listener is disabled to ensure performance of the new primary when the secondary is down. This behavior can be changed so that both types of traffic are enabled after a failover.

Failovers can be performed manually even if automatic failover is allowed. Depending on the type of failover, there could be data loss. Unplanned failovers could result in data loss if forced and the secondary is not fully synchronized with the primary. Configuring GracePeriodWithDataLossHours controls how long Azure waits before failing over. The default is one hour. If you have a tight RPO and cannot afford much data loss, set the value higher so Azure will wait longer before failing over, hopefully resulting in less data loss.

One auto-failover group can contain one or more databases. Note that the database size and edition will be the same on both the primary and secondary. The database is created automatically on the secondary through a process called seeding. Depending on the size of the database, this may take some time. Ensure that you plan accordingly and that you take into account things like the speed of the network.

## Summary and Knowledge Check

Depending on the PaaS option chosen for your deployment, there will be different options that you can configure to meet your availability needs, even if you do not have as many options as with an IaaS solution.

### Question 1

*What feature is not available with Azure SQL Database Managed Instance?*

- Failover group
- Secondary replica that is readable
- Active geo-replication

### Question 2

*What do Temporal Tables allow you to do with Azure SQL Database?*

- Recover deleted data
- Scale out reads
- Temporarily process additional data

### Question 3

*What setting for auto-failover groups must you change if you need to ensure a low RPO?*

- RPOZero
- AlwaysBeInSync
- GracePeriodWithDataLossHours

# Database Backup and Recovery

## Introduction

Even if you do nothing else to increase your availability with features like AGs or active geo-replication, you must have a solid backup and recovery strategy. If for some reason a feature fails, you will have to restore databases from backups.

## Learning Objectives

In this lesson, you will learn:

- Backup and restore options for IaaS
- Backup and restore options for PaaS

## Backup and Restore with SQL Server Using Azure Virtual Machines

SQL Server has two types of databases: system and user. System databases are the ones used by SQL Server such as master and msdb. User database are the ones created by users that store the data for applications. Both are important to account for when devising a backup and recovery plan. The nature of most system databases is that they are updated less frequently, although there are exceptions. As a general rule of thumb, system databases are usually not restored from one SQL Server instance to another. Your main concern should be backing up the user databases.

The most common types of backups generated for SQL Server installations are full, differential, and transaction log. Depending on the deployment method, not all of these may be available as an option.

A full database backup is a backup of a single database. When the backup is made, all the pages from the database are copied to the backup device. The backup contains enough information so that you can restore the database to the point at which the backup was made. If you want to restore to a specific point-in-time to achieve your RPO, that can happen with the use of differential and/or transaction log backups. full backup does not flush (or back up) the transaction log. Only a transaction log backup does that and is discussed below.

A differential backup contains all the database pages that have changed since the last time a full backup was made.

A transaction log backup is not only used to be able to achieve RPO and get to a more granular point in time but clears the transaction log and keeps its size manageable. Transaction log backups can be generated as frequently as every 30 seconds, although that is usually impractical. It is important to understand how the transaction log works because it impacts not only how the transaction log is backed up, but also how you can do point-in-time recovery using transaction log files. Consult the SQL Server documentation topic "SQL Server Transaction Log Architecture and Management Guide" linked at the end of this module to revisit transaction log fundamentals including how the log is flushed.

There are other backup options such as copy-only, file, filegroup, partial, and more. For a full overview of all of the backup types and other information, consult the documentation topic "Backup Overview (SQL Server)" in the For More Information section. Copy-only backups will be important to an upcoming topic.

A differential or a log backup can be restored after a full database is restored, as long as the database RESTORE command uses either the WITH NORECOVERY or the WITH STANDBY option. If neither option is used, the database RESTORE will do a recovery of the database, after which no additional backups can be

applied. To refresh your memory about these RESTORE options, read the topic [Restore Database \(Options Page\)](#) in documentation linked at the end of this module.

Every SQL Server database uses one of three recovery models: FULL, BULK\_LOGGED or SIMPLE. The recovery model is set as a database option, and governs the type of backups and restores that could be used with the database. Most databases are set to FULL or SIMPLE. FULL allows all types of backups to be generated while SIMPLE does not allow transaction log backups. This means that if you have a smaller RPO, SIMPLE may not meet your needs as you cannot restore to a granular point in time. To brush up on recovery models and how they affect the transaction log and subsequently, backup and recovery, read the documentation topic ["Recovery Models \(SQL Server\)"](#) linked at the end of this module.

## Back Up the Entire Virtual Machine

Azure Backup can back up VMs that contain SQL Server. These backups would contain not just SQL Server databases; they would everything that is in the VM so it could be restored as a whole. While this option may not be right for everyone, it can potentially protect against problems like ransomware.

VM-level backups are SQL Server-aware, also known as application aware, so they will create a application-consistent backup. This means that if you restore a VM-level backup, it will not 'break' SQL Server. If using this option, when looking in the SQL Server log you will see that the I/O has been momentarily frozen and then starts again when complete. If this causes issues with availability features like AGs, you may want to consider another backup strategy.

Combining SQL Server backups with snapshots can potentially cause issues. If snapshot delays cause backup failures, set following registry key:

[HKEY\_LOCAL\_MACHINE\SOFTWARE\MICROSOFT\BCDRAGENT]

"USEVSSCOPYBACKUP"="TRUE"

## Use Local Disks or a Network Share for Backup Files

As with on premises SQL Server instances, databases can be backed up to disks attached to the VM or to network shares (including the file share in Azure called Azure Files) that SQL Server has access to. If you are backing up to disks local to the VM, ensure that they are not written to the ephemeral storage that is erased upon shutdown or restart. You might also want to make sure that the backups are copied to a second location so as not to create a single point of failure.

## Back Up Databases to and Restore from URL

Another option is to configure backup to URL for the SQL Server instance installed in the VM. Unlike backups made on premises, backup and restore from URL for an IaaS VM is effectively a local option.

Backup to URL requires an Azure storage account and uses the Azure blob storage service. Inside the storage account there are containers, and the blobs are stored there. Unlike a path on your local disk, the path to a backup file will look something like <https://ACCOUNTNAME.blob.core.windows.net/Container-Name/MyDatabase.bak>. You can include additional folder names under your container for easier identification of backups (e.g. FULL, DIFF, LOG).

To backup to or restore from a URL, authentication must be established between the SQL Server instance and Azure. Remember that inside of an Azure VM, SQL Server does not know it is running on Azure. A SQL Server Credential can be comprised of the Azure storage account name and access key authentication or a Shared Access Signature. If the former is used, the backup will be stored as a page blob and if the latter, it will be stored as a block blob. Starting with SQL Server 2016, only block blob is available so

you should use a Shared Access Signature. From a cost perspective, block blobs are also cheaper, and Shared Access Signature tokens offer better security control.

Restoring from a URL is just as simple as restoring from disk or a network share. In the SQL Server Management Studio UI, select URL from the backup media type in the Wizard. If using Transact-SQL, instead of using FROM DISK, you would use FROM URL with the appropriate location and backup file name(s). Here are some sample statements.

The following statement would back up a transaction log.

```
BACKUP LOG contoso
```

```
TO URL = 'https://myacc.blob.core.windows.net/mycontainer/contoso202003271200.trn'
```

The following statement would restore a full database backup without recovering it, so that a differential or transaction log backups could be applied.

```
RESTORE DATABASE contoso
```

```
FROM URL = 'https://myacc.blob.core.windows.net/mycontainer/contoso20200327.bak'
```

```
WITH NORECOVERY
```

## Automated Backups Using the SQL Server Resource Provider

Any IaaS VM that has SQL Server installed can use the SQL Server resource provider. One of its options is the ability to configure automated backups so Azure takes care of backing up SQL Server databases. It requires the use of a storage account.

One benefit of implementing backups this way is that you can manage retention times for the backups. Another benefit is that you can ensure RPO due to the ability to take database and transaction log backups all in one easy-to-configure place. Figure 15 shows an example of what configuring an automated backup looks like in the Azure portal.

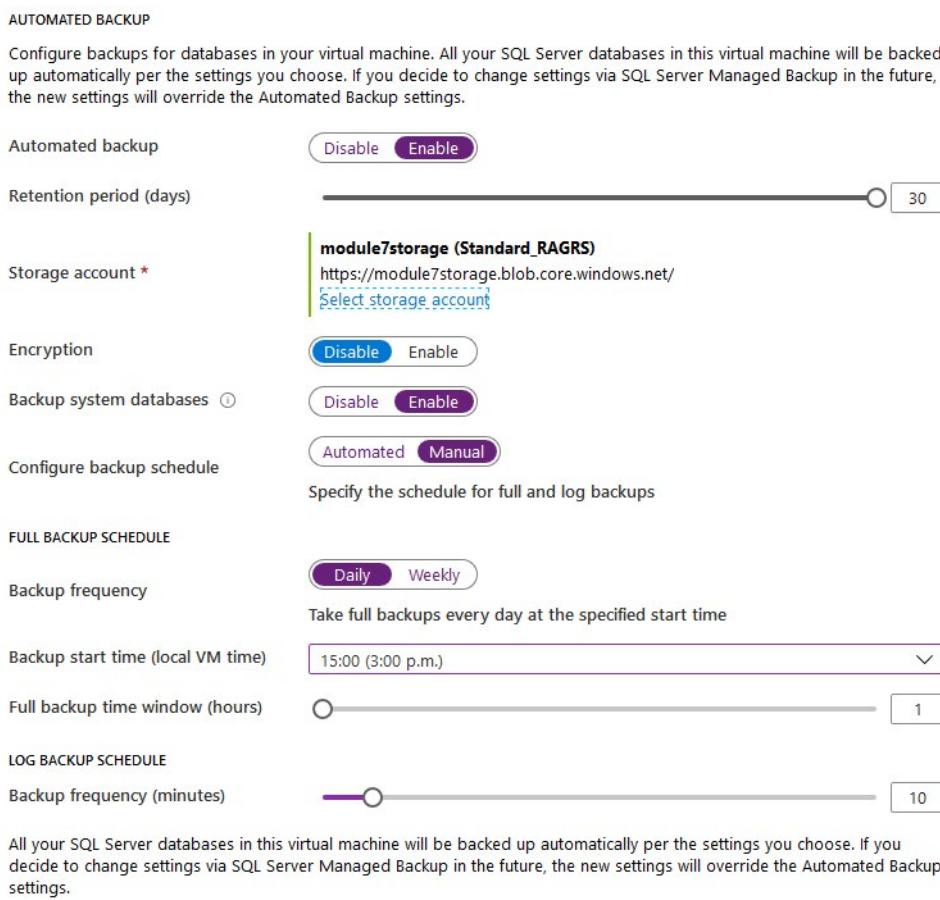


Figure 15: Configuring automated backups for SQL Server in a VM in the Azure Portal

Note that the automated backup option is currently only available for Windows Server-based SQL Server installations.

It is important that you choose one method of backing up databases with IaaS-based SQL Server deployments. For example, if you use automated backups, especially with transaction log backups, do not also configure those at the instance level inside the VM. You could cause problems with the log chain with restoring a database if things are uncoordinated, because each log backup clears the log and you must have an entire unbroken chain of log backups in order to do a log restore. For example, if transaction log backups happen inside the guest as well as at the Azure level, you may have to piece together the backups to do a restore.

While the backups can be automated, restores cannot be. You would need to configure and use the restore from URL functionality within SQL Server.

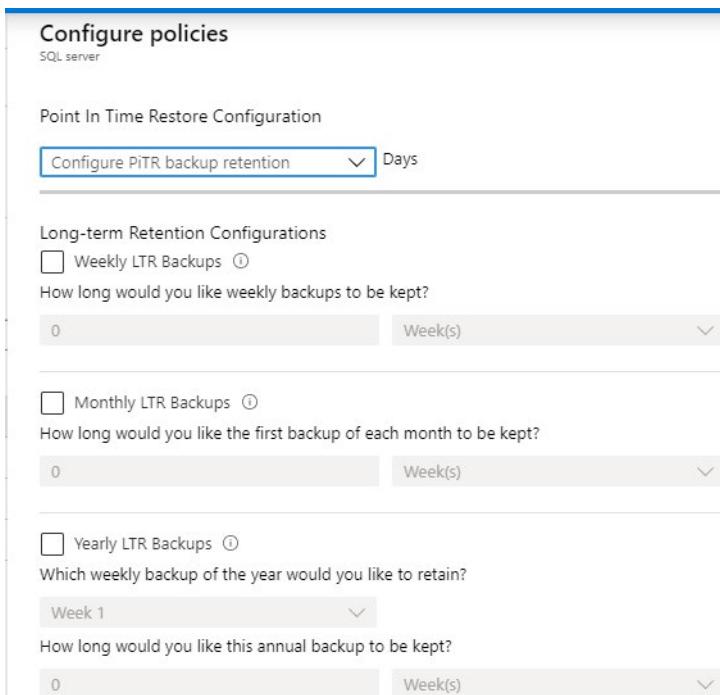
## Database Backup and Restore for Azure SQL Database

Backing up and restoring Azure SQL Database is different than IaaS.

Backups are generated automatically for Azure SQL Database. A full backup is created once a week, a differential every 12 hours, and transaction log backups every 5 – 10 minutes. All backups are located in

read-access, geo-redundant (RA-GRS) blobs replicated to a datacenter that is paired based on Azure rules. That means backups are safe from an outage in a single data center.

Backup policies can be configured per database as shown in Figure 16. Azure SQL Database can assist you to be compliant with mandatory backups for regulatory purposes with retention policies.



*Figure 16: Configuring automated backups for Azure SQL Database*

To restore a database to Azure SQL Database, follow the instructions linked in the For More Information section. One caveat to note: if the server containing the database is deleted, all backups will be deleted at the same time and there is no way to recover them. If the server is not deleted but the database is, the database can be restored normally.

Both Azure SQL Database and Azure SQL Database Managed Instance have a feature called Accelerated Database Recovery (ADR). Its purpose is to decrease the time it takes to deal with long running transactions so they do not impact the recovery time. Although ADR was developed for Azure and was originally an Azure-based feature, ADR was implemented in SQL Server 2019 as well.

Backups generated for Azure SQL Database cannot be used with Azure SQL Database Managed Instance or any version of SQL Server.

## Database Backup and Restore for Azure SQL Database Managed Instance

By default, Azure creates backups for databases in Azure SQL Database Managed Instance and functions like Azure SQL Database. To be able to restore a database to a previous point in time, follow the documentation linked at the end of the module.

You can also manually back up and restore databases with Azure SQL Database Managed Instance using the same backup to URL/restore from URL functionality found in SQL Server covered earlier. That requires the use of credentials to access Azure Blob Storage.

There is one major difference: for full backups, you can only generate a COPY\_ONLY backup since Azure SQL Database Managed Instance is maintaining the log chain. A sample backup statement would look like

```
BACKUP DATABASE contoso  
TO URL = 'https://myacc.blob.core.windows.net/testcontainer/contoso.bak'  
WITH COPY_ONLY
```

Backups generated by Azure SQL Database Managed Instance cannot be used to restore a database on SQL Server or Azure SQL Database.

## Database Backup and Restore for Azure Database for MySQL

Backups of the data files and transaction log are automatically created for Azure Database for MySQL in either locally- or geo-redundant storage. The backups are encrypted with a default retention of 7 days, and a maximum of 35. One nice thing about the backups is that there is no cost; if a server is provisioned with 100 GB of storage, you get the equivalent amount of backup space included with what you are already paying. If backup storage exceeds the 100GB, there will be a charge.

Restores can be done either to a point in time or as a geo-restore which recovers the database in another region if geo-redundant storage was configured.

Similar to Azure SQL Database and Azure SQL Database Managed Instance, these backups can only be used by Azure Database for MySQL and not a standard installation of MySQL.

## Database Backup and Restore for Azure Database for PostgreSQL

The backup and restore situation for Azure Database for PostgreSQL is similar to that of MySQL. Backups occur automatically and both data files and the transaction log are backed up. The retention and encryption are the same as are the options for local- or geo-redundancy, cost, and restore options. The only difference is that depending on the supported maximum storage size, Azure will create full and differential backups (max of 4TB) or snapshot backups (up to 16TB).

Similar to all other Azure PaaS offerings, these backups can only be used by Azure Database for PostgreSQL and not a standard installation of PostgreSQL.

## Summary and Knowledge Check

Without backups, you do not have high availability or disaster recovery. Features like AGs or active-geo replication do not supplant a proper backup and recovery strategy. Backups must be tested via a restore otherwise your backups are just files sitting on a file system that checks a box somewhere that backups were generated.

### Question 1

*How does backup to URL in SQL Server or Azure SQL Server Database Managed Instance store the backup file?*

- As a URL
- As a blob
- As a pointer

## Question 2

*Which platforms can a PaaS database backup from Azure be restored to?*

- Microsoft SQL Server
- PostgreSQL
- MySQL
- All of the above
- None of the above

## Question 3

*What is the cost for backup storage for both Azure Database for MySQL and Azure Database for PostgreSQL?*

- Free up to the size of the database
- Double the cost of the service
- Half the cost of the service

## Module Summary

### Module Summary

There are many aspects to ensuring that IaaS and PaaS are up when you need your database installations and/or databases to be accessible. There is no one right way for every scenario. You need to have documented RTOs and RPOs, and then understand any other challenges or limitations before deciding on an architecture to deploy. At a most fundamental level, ensure that your backup and recovery plan is solid and that you have backups that have been tested and verified to be good.

### Additional Resources

#### Azure Site Recovery documentation<sup>3</sup>

<https://docs.microsoft.com/en-us/azure/site-recovery/site-recovery-overview>

Azure RTO and RPO documentation

[https://azure.microsoft.com/en-us/support/legal/sla/sql-database/v1\\_4/](https://azure.microsoft.com/en-us/support/legal/sla/sql-database/v1_4/)

Configuring Hyperscale in Azure

<https://docs.microsoft.com/en-us/azure/postgresql/howto-hyperscale-high-availability>

Hyperscale SLA

[https://azure.microsoft.com/en-us/support/legal/sla/postgresql/v1\\_1/](https://azure.microsoft.com/en-us/support/legal/sla/postgresql/v1_1/)

ExpressRoute

<https://docs.microsoft.com/en-us/azure/expressroute/>

Cluster validation documentation

[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/jj134244\(v%3Dws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/jj134244(v%3Dws.11))

Creating an Always On Availability Group

<https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/creation-and-configuration-of-availability-groups-sql-server?view=sql-server-ver15>

Create an internal Azure load balancer documentation.

<https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sql/virtual-machines-windows-portal-sql-alwayson-int-listener>

Failing a VM to another region using ASR

<https://docs.microsoft.com/en-us/azure/site-recovery/site-recovery-failover>

Testing disaster recovery with ASR without bringing down production

<https://docs.microsoft.com/en-us/azure/site-recovery/site-recovery-test-failover-to-azure>

Auto-failover group documentation

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-autofailover-group?tabs=azure-powershell>

Restore Database (Options Page)

---

<sup>3</sup> <https://docs.microsoft.com/en-us/azure/site-recovery/site-recovery-overview>

<https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/restore-database-options-page?view=sql-server-ver15>

SQL Server Transaction Log Architecture and Management Guide

<https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-log-architecture-and-management-guide?view=sql-server-ver15>

Backup Overview (SQL Server)

<https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/backup-overview-sql-server?view=sql-server-ver15>

Recovery Models (SQL Server)

<https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/recovery-models-sql-server?view=sql-server-ver15>

Azure File shares using Azure Files documentation

<https://docs.microsoft.com/en-us/azure/storage/files/storage-files-introduction>

Back up to URL documentation

<https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/sql-server-backup-to-url?view=sql-server-ver15>

Restore a database to Azure SQL Database documentation

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-recovery-using-backups>

ADR documentation

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-accelerated-database-recovery>

Restore a database to a point in time with Azure SQL Database Managed Instance documentation

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-managed-instance-point-in-time-restore?tabs=azure-portal>

Back up to URL with Azure SQL Database Managed Instance

<https://techcommunity.microsoft.com/t5/azure-sql-database/native-database-backup-in-azure-sql-managed-instance/ba-p/386154>

Restoring databases to Azure SQL Database Managed Instance

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-managed-instance-get-started-restore>

Backup and restore with Azure Database for MySQL documentation

<https://docs.microsoft.com/en-us/azure/mysql/concepts-backup>

How to back up Azure Database for MySQL to blob storage using Azure CLI

<https://techcommunity.microsoft.com/t5/azure-database-for-mysql/backup-azure-database-for-mysql-to-a-blob-storage/ba-p/803830>

Backup and restore with Azure Database for PostgreSQL documentation.\

<https://docs.microsoft.com/en-us/azure/postgresql/concepts-backup>

# Answers

## Question 1

What is RPO?

- The number of nodes in a cluster
- The point to which data needs to be recovered after a failure
- A partial database restore

*Explanation*

## Question 2

What is a hybrid solution?

- A solution that has resources both in Azure as well as on premises or in another cloud provider
- A solution that uses two different database engines e.g. MySQL and SQL Server
- A solution that spans two different versions of SQL Server

*Explanation*

## Question 3

What is available after failover with database-level protection in SQL Server?

- Logins, Databases, and SQL Server Agent Job
- Databases and SQL Server Agent jobs
- Whatever is in the databases; anything outside must be dealt with manually

*Explanation*

## Question 1

What component in Azure needs to be configured for the listener in an AG to work properly?

- The NIC
- The NSG
- A load balancer

*Explanation*

## Question 2

How should you create a WSFC in Azure for AGs and FCIs?

- Wizard in Failover Cluster Manager
- PowerShell
- WMI

*Explanation*

**Question 3**

What Azure feature lets you test disaster recovery without bringing down your production system?

- ASR
- Azure SQL Database
- Azure Load Balancer

*Explanation*

**Question 1**

What feature is not available with Azure SQL Database Managed Instance?

- Failover group
- Secondary replica that is readable
- Active geo-replication

*Explanation*

**Question 2**

What do Temporal Tables allow you to do with Azure SQL Database?

- Recover deleted data
- Scale out reads
- Temporarily process additional data

*Explanation*

**Question 3**

What setting for auto-failover groups must you change if you need to ensure a low RPO?

- RPOZero
- AlwaysBeInSync
- GracePeriodWithDataLossHours

*Explanation*

**Question 1**

How does backup to URL in SQL Server or Azure SQL Server Database Managed Instance store the backup file?

- As a URL
- As a blob
- As a pointer

*Explanation*

**Question 2**

Which platforms can a PaaS database backup from Azure be restored to?

- Microsoft SQL Server
- PostgreSQL
- MySQL
- All of the above
- None of the above

*Explanation*

**Question 3**

What is the cost for backup storage for both Azure Database for MySQL and Azure Database for PostgreSQL?

- Free up to the size of the database
- Double the cost of the service
- Half the cost of the service

*Explanation*