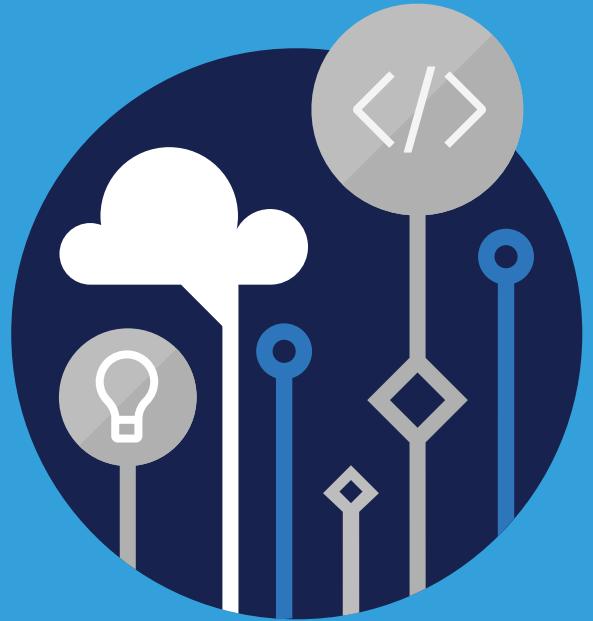


Microsoft
Official
Course



DP-200T01

Implementing an Azure
Data Solution

DP-200T01

**Implementing an Azure Data
Solution**

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>¹ are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

¹ <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.
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

| | | |
|---|---|-----|
| ■ | Module 0 Introduction | 1 |
| | Welcome to the Course | 1 |
| ■ | Module 1 Azure for the Data Engineer | 7 |
| | Module Introduction | 7 |
| | The Evolving World of Data | 8 |
| | Surveying the Azure Data Platform | 16 |
| | Data Engineering Roles and Responsibilities | 28 |
| | Course Case Study | 36 |
| | Module Summary | 40 |
| ■ | Module 2 Working with Data Storage | 45 |
| | Module Introduction | 45 |
| | Choose a data storage approach in Azure | 46 |
| | Introducing Azure Storage | 51 |
| | Introduction to Data Lake Storage | 61 |
| | Create an Azure Data Lake Storage Gen2 | 70 |
| | Module Summary | 89 |
| ■ | Module 3 Enabling Team Based Data Science with Azure Databricks | 93 |
| | Module Introduction | 93 |
| | Introduction to Azure Databricks | 94 |
| | Working with Azure Databricks | 99 |
| | Reading Data using Azure Databricks | 105 |
| | Performing Transformations with Azure Databricks | 108 |
| | Module Summary | 115 |
| ■ | Module 4 Building Globally Distributed Database with Azure Cosmos DB | 119 |
| | Module Introduction | 119 |
| | Create an Azure Cosmos DB database built to scale | 120 |
| | Insert and query data in your Azure Cosmos DB database | 132 |
| | Build a .NET Core app for Azure Cosmos DB in Visual Studio Code | 157 |
| | Distribute your Data Globally with Azure Cosmos DB | 174 |
| | Module Summary | 187 |
| ■ | Module 5 Working with Relation Data in the Cloud | 191 |
| | Module Introduction | 191 |
| | Azure SQL Database | 192 |

| | |
|--|-----|
| Azure Synapse Analytics | 205 |
| Creating and Querying an Azure Synapse Analytics | 215 |
| Using PolyBase to Load Data into Azure Synapse Analytics | 234 |
| Module Summary | 247 |
| ■ Module 6 Performing Real Time Analytics with Stream Analytics | 251 |
| Module Introduction | 251 |
| Introducing Data Streams and Event Processing | 252 |
| Data Ingestion with Event Hubs | 256 |
| Processing Data with Stream Analytics Jobs | 263 |
| Module Summary | 279 |
| ■ Module 7 Orchestrating Data Movement with Azure Data Factory | 283 |
| Module Introduction | 283 |
| Introducing Azure Data Factory | 284 |
| Azure Data Factory Components | 289 |
| Ingesting-and-Transforming-data | 302 |
| Integrate-Azure-Data-Factory-with-Databricks | 322 |
| Module-Summary | 326 |
| ■ Module 8 Securing Azure Data Platforms | 331 |
| Module Introduction | 331 |
| Introduction to Security | 332 |
| Key Security Components | 342 |
| Securing Storage Accounts and Data Lake Storage | 358 |
| Securing Data Stores | 367 |
| Securing Streaming Data | 394 |
| Module Summary | 397 |
| ■ Module 9 Monitoring and Troubleshooting Data Storage and Processing | 403 |
| Module Introduction | 403 |
| General Azure Monitoring Capabilities | 404 |
| Troubleshoot Common Data Platform Issues | 408 |
| Troubleshoot Common Data Processing Issues | 414 |
| Managing Disaster Recovery | 419 |
| Module Summary | 423 |

Module 0 Introduction

Welcome to the Course

Implementing an Azure Data Solution

This course has been created by the Microsoft World Wide Learning team to support exam DP-200: Implementing an Azure Data Solution.

Learning objectives

In this module you will gain information:

- About this course
- About the audience
- Course pre-requisites
- Understand the DP 200 certification
- About this course

Course Introduction

This module provides an overview of the following

- About this course
- Course agenda
- Course audience
- Course pre-requisites
- DP200: Implementing an Azure Data Solution certification details

About this course

In this course, the students will implement various data platform technologies into solutions that are in line with business and technical requirements including on-premises, cloud, and hybrid data scenarios incorporating both relational and No-SQL data. They will also learn how to process data using a range of technologies and languages for both streaming and batch data.

The students will also explore how to implement data security including authentication, authorization, data policies and standards. They will also define and implement data solution monitoring for both the data storage and data processing activities. Finally, they will manage and troubleshoot Azure data solutions which includes the optimization and disaster recovery of big data, batch processing and streaming data solutions.

Course Agenda

At the end of this course, the student will learn:

Module 1: Azure for the Data Engineer

This module explores how the world of data has evolved and how cloud data platform technologies are providing new opportunities for business to explore their data in different ways. The student will gain an overview of the various data platform technologies that are available, and how a Data Engineers role and responsibilities has evolved to work in this new world to an organization benefit.

Module Objectives:

At the end of this module, the students will be able to:

- Explain the evolving world of data
- Survey the services in the Azure Data Platform
- Identify the tasks that are performed by a Data Engineer
- Describe the use cases for the cloud in a Case Study

Module 2: Working with Data Storage

This module teaches the variety of ways to store data in Azure. The Student will learn the basics of storage management in Azure, how to create a Storage Account, and how to choose the right model for the data you want to store in the cloud. They will also understand how data lake storage can be created to support a wide variety of big data analytics solutions with minimal effort.

Module Objectives:

At the end of this module, the students will be able to:

- Choose a data storage approach in Azure
- Create an Azure Storage Account
- Explain Azure Data Lake Storage
- Upload data into Azure Data Lake

Module 3: Enabling Team Based Data Science with Azure Databricks

This module introduces students to Azure Databricks and how a Data Engineer works with it to enable an organization to perform Team Data Science projects. They will learn the fundamentals of Azure Databricks and Apache Spark notebooks; how to provision the service and workspaces and learn how to perform data preparation task that can contribute to the data science project.

Module Objectives:

At the end of this module, the students will be able to:

- Explain Azure Databricks
- Work with Azure Databricks
- Read data with Azure Databricks
- Perform transformations with Azure Databricks

Module 4: Building Globally Distributed Databases with Cosmos DB

In this module, students will learn how to work with NoSQL data using Azure Cosmos DB. They will learn how to provision the service, and how they can load and interrogate data in the service using Visual Studio Code extensions, and the Azure Cosmos DB .NET Core SDK. They will also learn how to configure the availability options so that users are able to access the data from anywhere in the world.

Module Objectives:

At the end of this module, the students will be able to:

- Create an Azure Cosmos DB database built to scale
- Insert and query data in your Azure Cosmos DB database
- Build a .NET Core app for Azure Cosmos DB in Visual Studio Code
- Distribute your data globally with Azure Cosmos DB

Module 5: Working with Relational Data Stores in the Cloud

In this module, students will explore the Azure relational data platform options including Azure SQL Database and the enterprise data warehouse capabilities using Azure Synapse Analytics. The student will be able explain why they would choose one service over another, and how to provision and connect to each of the services.

Module objectives

In this module, you'll learn:

- Work with Azure SQL Database
- Work with Azure Synapse Analytics

- Provision and query data in Azure Synapse Analytics
- Import data into Azure Synapse Analytics using PolyBase

Module 6: Performing Real-Time Analytics with Stream Analytics

In this module, students will learn the concepts of event processing and streaming data and how this applies to Events Hubs and Azure Stream Analytics. The students will then set up a stream analytics job to stream data and learn how to query the incoming data to perform analysis of the data. Finally, you will learn how to manage and monitor running jobs.

Module Objectives:

At the end of this module, the students will be able to:

- Explain data streams and event processing
- Data Ingestion with Event Hubs
- Processing Data with Stream Analytics Jobs

Module 7: Orchestrating Data Movement with Azure Data Factory

In this module, students will learn how Azure Data Factory can be used to orchestrate the movement and transformation of data both natively and from a wide range of data platform technologies. You will be able to explain the capabilities of the technology and set up an end to end data pipeline that ingests and transforms data with an Azure data platform technology.

Module Objectives

In this module, you will:

- Introduced to Azure Data Factory
- Understand Azure Data Factory Components
- Ingesting and Transforming Data with Azure Data Factory
- Integrate Azure Data Factory with Databricks

Module 8: Securing Azure Data Platforms

In this module, students will learn how Azure provides a multi-layered security model to protect your data. The students will explore how security can range from setting up secure networks and access keys, to defining permission through to monitoring with Advanced Threat Detection across a range of data stores.

Module Objectives:

At the end of this module, the students will be able to:

- An introduction to security

- Key security components
- Securing Storage Accounts and Data Lake Storage
- Securing Data Stores
- Securing Streaming Data

Module 9: Monitoring and Troubleshooting Data Storage and Processing

In this module, the student will get an overview of the range of monitoring capabilities that are available to provide operational support should there be issue with a data platform architecture. They will explore the common data storage and data processing issues. Finally, disaster recovery options are revealed to ensure business continuity.

Module Objectives:

At the end of this module, the students will be able to:

- Explain the monitoring capabilities that are available
- Troubleshoot common data storage issues
- Troubleshoot common data processing issues
- Manage disaster recovery

Course Audience

Primary audience

The audience for this course is data professionals, data architects, and business intelligence professionals who want to learn about the data platform technologies that exist on Microsoft Azure.

Secondary audience

The secondary audience for this course is individuals who develop applications that deliver content from the data platform technologies that exist on Microsoft Azure. |

Course Prerequisites

In addition to their professional experience, students who take this training should have technical knowledge equivalent to the following courses:

- **Azure fundamentals¹**

¹ <https://docs.microsoft.com/en-us/learn/parts/azure-fundamentals/>

Microsoft Certification

Azure Data Engineer Associate Certification

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. To gain this certification, you must pass the following two exams

- Exam DP-200: Implementing an Azure Data Solution
- Exam DP-201: Designing an Azure Data Solution

This course is used to prepare for exam DP 200.

DP-200: Implement an Azure Data Solution Exam Details

The DP200 exam is used to test the breadth of skills required to be a data engineer in Microsoft technologies today. Candidates who earn an Azure Data Engineer certification are verified by Microsoft to have the following skills and knowledge for the **DP200 exam²**.

² <https://docs.microsoft.com/en-us/learn/certifications/exams/dp-200>

Module 1 Azure for the Data Engineer

Module Introduction

Azure for the Data Engineer

This module explores how the world of data has evolved, and how cloud data platform technologies are providing new opportunities for businesses to explore their data in different ways. You will gain an overview of the various data platform technologies that are available, and how a Data Engineers role and responsibility has evolved to work in this new world for an organizations benefit.

Learning objectives

In this module, you will:

- Explain the evolving world of data
- Survey the services in the Azure Data Platform
- Identify the tasks that are performed by a Data Engineer
- Describe the use cases for the cloud in a Case Study

The Evolving World of Data

The Evolving World of Data

The amount of data generated by systems and devices has increased significantly over the last decade. To address this increase in demand, new technologies, new roles, and new approaches to working with data are impacting data professionals. Regardless of the industry, many data professionals are wanting to understand better how these changes are affecting both their careers and daily working lives.

It's essential for anyone working with data to understand how the data landscape has changed, and how the roles and technologies are evolving to generate value for both individuals and businesses. You should be able to explain this shift to any stakeholder and clearly articulate the key factors driving the change. Most important is your ability to explain the benefits brought to an organization by embracing such changes.

Learning objectives

In the following lessons, you will learn about:

- Data Abundance
- The differences between on-premises data technologies and cloud data technologies
- How the role of the data professional is changing in organizations
- The use cases impacted by these changes

Data Abundance

As the last 30 years have demonstrated, the number of devices and software generating data has increased exponentially to meet today's business and user needs. This has had an impact on the following areas:

Processes

Businesses are tasked to process data by storing it, managing and securing it, and processing and transforming it for reporting purposes to interested parties. These parties include internal management, investors, business partners, regulatory bodies, and consumers.

Consumers

Data consumers view data using web browsers, PCs, tablets, and mobile devices in either a connected or disconnected state. They can both generate and use data both in the workplace and during leisure time with social media applications. Business stakeholders use data to make business decisions, and consumers use the data to help them make decisions such as purchasing decisions as an example. In areas such as Artificial Intelligence (AI), Azure Machine Learning (Azure ML) can consume the data and make decisions instead of humans.

Variety

Many forms of data exist including text, data stream, audio, video, metadata, structured data, unstructured data, and aggregated data. With structured data, the data architect defines the structure (schema) during the creation of the data storage at design-time in the data platform technology of choice, such

as Azure SQL Database. With unstructured (NoSQL) databases, each data element can have its own schema at query time. Data can be stored as a file (Azure Blob Storage), or as NoSQL data in Azure Cosmos DB or HDInsight.

Responsibilities

Data engineers are responsible for maintaining data systems that are accurate, highly secure, and constantly available while complying with applicable regulations such as GDPR or industry standards such as PCI-DSS. Data can be located anywhere, on-premises, or in the cloud, and it can be processed either in real time or in a batch. International companies may also have special requirements to provide data to conform to regional norms such as the local language and date format.

Technologies

Microsoft Azure provides a comprehensive and rich set of data tools and technologies to provide businesses the ability to store, transform, process, analyze, and visualize a wide variety of data formats in a secure way. As data formats continue to evolve, Microsoft continually releases new technologies to the Microsoft Azure platform. Microsoft Azure customers can explore these new technologies while in preview mode. Leveraging Microsoft Azure's on-demand subscription model, customers can keep their costs to a minimum, paying only for what they consume when they need it.

On-premises systems versus the cloud

Many organizations are considering digital transformation projects as their traditional IT hardware and infrastructure comes to the end of life. Here, we'll explore features of both on-premises and cloud environments. We'll also explore the factors businesses must consider when exploring each option.

On-premises

Computing Environment

On-premises environments require an investment in physical equipment to execute applications and services. This equipment includes physical servers, network infrastructure, and storage. The equipment must have power, cooling, and requires periodic maintenance by qualified personnel. The servers must also have at least one operating system (OS) installed, and possibly more than one if the organization is utilizing virtualization technology. Each installed OS could potentially require its own licensing cost.

Licensing Model

With on-premises servers, OS and software licenses are typically sold per server or per CAL (Client Access Licensing). As companies grew, the licensing arrangements became more rigid and restrictive.

Maintainability

Maintaining on-premises systems requires maintaining the hardware, firmware, drivers, BIOS, operating system, software, and anti-virus software. These are additional considerations associated with the operational costs of such an environment. Organizations will look to reduce these costs where it makes sense.

Scalability

When scaling up a server is no longer possible, administrators look to scale out their operations. To scale an on-premises server horizontally, the server administrator would add an additional server node to a cluster. Clustering leverages either a hardware or software load balancer to distribute incoming network requests to an individual node of the cluster. A limitation of server clustering is that the hardware for each server in the cluster must be identical. Therefore, once the server cluster reaches maximum capacity, a server administrator must replace or upgrade each node in the cluster.

High Availability

High availability systems must be available most of the time. Service Level Agreements (SLAs) will dictate specific expectations. Three 9's, four 9's, or five 9's commonly refers to 99.9%, 99.99%, or 99.999% system uptime. To calculate this in terms of the number of hours, multiply these percentages by the number of hours in a single year (8760).

| Uptime Level | Uptime Hours per Year | Downtime Hours per Year |
|--------------|-----------------------|--------------------------------------|
| 99.9% | 8751.24 | (8760 – 8751.24) = 8.76 hrs per year |
| 99.99% | 8759.12 | (8760 – 8759.12) = 0.88 hrs per year |
| 99.999% | 8759.91 | (8760 - 8759.91) = 0.09 |

The higher the uptime required, the more expensive it can become for on-premises servers.

Supportability

There are hundreds of vendors for physical server hardware, requiring server administrators to learn many different platforms. This makes it more difficult for organizations to find human resources with the diverse skills required to administer, maintain and support these systems.

Multi-lingual Support

In on-premises SQL Server systems, supporting multi-lingual scenarios is difficult and expensive. One concern with multiple languages is the impact on the sort order of text data, as different languages can sort text data differently. To support this, the SQL Server Database Administrator must install and configure the collation settings for the data. To leverage these settings, SQL Database Developers needed to consider multi-lingual functionality at design time. This increases the complexity of managing and maintaining such systems.

Total Cost of Ownership (TCO)

The term Total Cost of Ownership (TCO) is used to describe the total cost of owning a given technology. In on-premises systems, TCO includes:

- Hardware Costs
- Software Licensing Costs
- Labor Costs (Installation, upgrades, maintenance)
- Data Center Overhead Costs (power, telecommunications, building, heating & cooling)

From a cost perspective, it is difficult to match the on-premises expense to actual usage. Servers are purchased with additional capacity to accommodate future growth, so when a server is purchased there will always be excess capacity that isn't used. When an on-premises server is at maximum capacity, even an incremental increase in resource demand will require the purchase of additional hardware.

From a finance perspective, due to the large expense of on-premises server systems, these costs were often capitalized, meaning that the costs were allocated across the expected lifetime of the server equipment. Capital expenditures limited IT Managers from being able to upgrade server equipment when it was required during the expected server lifetime, limiting the server system's ability to meet the increased demand. In cloud solutions, expenses are recorded on the financial statements each month as an expense instead of a capital expenditure. Therefore, there is no expected server lifetime to limit the IT manager's ability to upgrade the system to meet an increase in demand when required.

Cloud

Computing Environment

Cloud computing environments contain the physical and logical infrastructure to host services, virtual servers, intelligent applications, and containers as if they belonged to the organization. There is no capital investment required for physical servers. Instead, an organization will provision services in the cloud and only pay for what they use. In addition, operational costs are reduced as the servers and services are moved into the cloud.

An organization can provision anything from virtual servers to clusters of containerized apps to Azure services within minutes, and all the physical and logical infrastructure is automatically created and handled in the background, reducing the complexity and cost of creating the services.

Where on-premises servers used physical and virtual disks, the cloud platform refers to storage more generically as there are many different storage types such as Azure Blob Storage, File Storage, and Disk Storage. Complex systems often leverage each type of storage as part of their technical architecture. With disk storage, customers can choose to either have Microsoft manage their disk storage or the customer can pay a premium to have greater control over disk allocation.

Maintainability

In the cloud, Microsoft manages many operations to create a stable computing environment that's included as part of the investment for Microsoft Azure products. Microsoft manages key infrastructure services such as physical hardware, computer networking, firewalls and network security, data center fault tolerance, compliance, and physical security over the plant or buildings. Microsoft also spends nearly 1 billion US dollars each year in battling cybersecurity threats and applies operating system updates and firmware updates on behalf of the customer. This allows Data Engineers to focus on more of the data engineering tasks and eliminates the need to design and manage some of the system complexity.

Scalability

Achieving scalability in on-premises systems is complicated and time-consuming, achieving scalability in the cloud can be as simple as a point and click of a mouse in many cases. Scalability in the cloud is typically measured in compute units. Compute units may be defined differently for each Microsoft Azure product being evaluated.

High Availability

In Azure, Microsoft commits to duplicating customers content for redundancy and high availability. Many of the services and platforms also have specific Service Level Agreements (SLAs) to ensure customers are aware of the availability capabilities of the platform that they are using.

Supportability

Cloud systems are easy to support because the environments are standardized. When Microsoft applies an update, the update applies to all consumers of the product.

Multi-lingual support

In cloud systems, it's common to store data as a JSON file that includes the Language Code ID (LCID) to identify the language used in the data. Apps processing the data can use translation services such as the Bing Translate API to convert the data into an expected language when the data is consumed or as part of a data preparation process.

Total Cost of Ownership

In cloud systems like Microsoft Azure, costs are tracked by subscriptions, which can be based upon usage including compute units, the number of hours, or the number of transactions. The hardware, software, disk storage, and labor costs are included in this amount. Due to economies of scale, it is difficult for an on-premises system to compete with the cloud concerning the measurement of the service usage.

On-premises server systems rarely matched the cost of operating the system with the actual usage of the system. In cloud systems, the cost more closely matches the actual usage of the system. However, if a cloud administrator provisions a service that isn't used by the business, there will be costs with no usage, referred to as underutilization. Organizations can reduce underutilization by adopting a best practice to only provision production instances once an application is ready to be deployed to production. This might require developers to use emulators like the Azure Cosmos DB emulator or the Azure Storage emulator to develop cloud applications without incurring production costs during development and testing.

Understand Lift and Shift

When moving to the cloud, one strategy that many customers will use is to migrate from physical or virtualized on-premises servers to Azure Virtual Machines. This is known as "Lift and Shift." In Lift and Shift, server administrators migrate an application from a physical environment to Azure Virtual machines without rearchitecting the applications.

While Lift and Shift provides immediate benefits such as the ability to provide higher availability, reduce operational costs, and be able to transfer workloads from one data center to another, the application is unable to take advantage of the many features available within Azure.

While Lift and Shift may seem appealing in the short term, organizations are using the migration as an opportunity to transform their business practices by creating new versions of their applications and databases to take advantage of the new services in Azure such as Cognitive Services, Bots and some of the available machine learning capabilities.

Data Engineers Job Responsibilities

When moving your skills from managing on-premises database server systems such as Microsoft SQL Server to cloud-based data platform systems, the role of the SQL Server professional will evolve from being a SQL Server-specific role to a more general data-specific role, known as the Data Engineer. While SQL Server professionals were largely limited to working with relational database systems, Data Engineers also work with unstructured data and a wide variety of new data types such as streaming data.

Mastering data engineering requires learning a new set of tools, architectures, and platforms. Where the SQL Professional may have mastered T-SQL as the primary data manipulation tool, the Data Engineer may use additional technologies like HDInsight and Cosmos DB, and languages such as Hive or Python to manipulate data in big data systems.

Changing loading approaches from ETL to ELT

Data extraction is the process by which a Data Engineer retrieves raw data from a structured or unstructured data pool and migrates it to a staging data repository. Since the data source may have a different structure than the target destination, Data Engineers transform the data from the source schema to the destination schema as part of a process called transformation. Loading occurs when the transformed data is loaded into the data warehouse. A disadvantage of the Extract, Transform, and Load (ETL) approach is the transformation stage can be time-consuming, potentially causing source system resources to be tied up while waiting for the process to complete.

With the Extract, Load and Transform (ELT) approach, the data is immediately extracted and loaded into a large data repository such as Azure Cosmos DB or Azure Data Lake Storage to reduce the resource contention on source systems. Data Engineers can begin transforming the data as soon as the load is completed. This also provides a more flexible architecture for supporting multiple transformations. For example, given some raw data, how the marketing department needs to transform the data can be different than how the operations department requires that same data.

Moving from implementing to provisioning servers

SQL Server professionals would typically work with versions of on-premises SQL Server to meet the data requirement of an organization. In the past, this would entail the lengthy process of installing and configuring the servers and services to provide the infrastructure needed to support a solution. Such work would take days to complete, and in the case of high availability environments, it could also take weeks.

Microsoft Azure reduces the complexity when it comes to building and deploying servers. Data Engineers can utilize a web user interface for simple deployments or can create powerful scripts to automate more complex deployments. Within minutes, an organization can stand up a globally distributed, highly performing and available database quicker than it takes to read this module. This means Data Engineers spend less time worrying about the setup of services, and they can focus more on security and deriving more value from the data for the business.

Use Cases for the Cloud

Microsoft Azure can work for a range of industries including:

Web Retail

Using Azure Cosmos DB's multi-master replication model along with Microsoft's performance commitments, Data Engineers can implement a data architecture to support web and mobile applications that achieve less than a 10-ms response time anywhere in the world. This can increase customer satisfaction by reducing the processing time of web sites for global organizations.

Healthcare

Azure Databricks can be used to accelerate big data analytics and artificial intelligence (AI) solutions. Within the healthcare industry, it can be used to perform genome studies or pharmacy sales forecasting at petabyte scale, while using the inherent features of Azure Databricks to set up your Spark environment in minutes and auto scale quickly and easily.

Data Engineers can collaborate with Data Scientist through shared projects and workspaces in a wide range of languages including SQL, R, Scala, and Python. Native integration with Azure Active Directory (Azure AD) and other Azure services enables you to build a range of solution types including a modern data warehouse, machine learning and real-time analytics solutions.

IoT Scenarios

Over the last couple of years, hundreds of thousands of devices have been designed and sold to generate sensor data known as Internet of Things (IoT) devices. Using technologies like Azure IoT Hub, Data Engineers can easily design a data solution architecture that captures information from IoT devices to be analyzed further.

Summary

In this section, we demonstrated how the world of data is evolving and its impact on data professionals. We discussed the differences between on-premises and cloud data solutions, and we provided a few sample business cases.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

What data processing framework will be mainly used by Data Engineers to ingest data into cloud data platforms on Azure?

- Online Transactional Processing (OLTP)
- Extract, Transform and Load (ETL)
- Extract, Load and Transform (ELT)

Question 2

What type of data can have its own schema at defined at query time?

- Structured data
- Cosmos DB
- Un-structured data

Question 3

Duplicating customers content for redundancy and meeting Service Level Agreements in Azure meets which cloud technical requirement?

- Maintainability
- High Availability
- Multi-lingual support

Surveying the Azure Data Platform

Surveying the Azure Data Platform

The range of data platform technologies that are available within Microsoft Azure can be overwhelming for even the most experienced data engineer to understand. Many data engineers are faced with solving complex data problems to provide value to the business through its data, regardless of the scenario or industry. Understanding the types of data and the capabilities of the various data platform technologies will help a data engineer to pick the right tool for the job.

Imagine working for an organization that is starting to explore the capabilities of the cloud. The network infrastructure team has been asked to provide the executives of the company an overview of the benefits and weaknesses of running IT operations in Azure. It is common in these situations for the network team to approach data engineers for information relating to the data services that are available in Azure. Would you be able to answer such questions at a high level? This section helps you achieve that objective.

Learning objectives

In this section, you will understand:

- The differences between structured and non-structured data
- Azure Storage
- Azure Data Lake Storage
- Azure Databricks
- Azure Cosmos DB
- Azure SQL Database
- Azure Synapse Analytics
- Azure Stream Analytics
- Additional Azure Data Platform Services

Types of Data

Microsoft Azure provides many data platform technologies to meet the needs of the wide varieties of data that are prevalent in the world today. However, it is worth reminding ourselves of the three broad types of data.

1. **Structured data.** Structured data is data that adheres to a schema, so all of the data has the same fields or properties. Structured data can be stored in a database table with rows and columns. Structured data relies on keys to indicate how one row in a table relates to data in another row of another table. Structured data is also referred to as *relational data*, as the data's schema defines the table of data, the fields in the table, and the clear relationship between the two. Structured data is straightforward in that it's easy to enter, query, and analyze. All of the data follows the same format. Examples of structured data include retail transaction or financial data.
2. **Semi-structured data.** Semi-structured data doesn't fit neatly into tables, rows, and columns. Instead, semi-structured data uses *tags* or *keys* that organize and provide a hierarchy for the data. Semi-structured data is also referred to as *non-relational* or *NoSQL* data.

3. **Unstructured data.** Unstructured data encompasses data that has no designated structure to it. This also means that there are no restrictions on the kinds of data it can hold. For example, a blob can hold a PDF document, a JPG image, a JSON file, video content, etc. As such, unstructured data is becoming more prominent as businesses try to tap into new data sources. In the open source world, there are four types of No-SQL databases:

- **Key-Value store:** Used to store key-value pairs of data in a table structure
- **Document database:** Stores documents that are tagged with metadata to aid document searches
- **Graph database:** A structure that's composed of vertices and edges to find relationships between data points
- **Column based:** Provides column-based storage of data, rather than row based, and the columns can be defined at the runtime of a query giving flexibility in the data that is returned in a performant manner.

Now that we've reminded ourselves of the broad types of data; let's take a tour of the common data platform technologies used to facilitate the storage, processing, and querying of these different types of data.

Storing Data using Azure Storage Accounts

Azure storage accounts are the base storage type used within Microsoft Azure. Azure Storage offers a massively scalable object store for data objects and file system services for the cloud. It can also provide a messaging store for reliable messaging, or act as a NoSQL store. The four configuration options available include:

- **Azure Blobs:** A massively scalable object store for text and binary data.
- **Azure Files:** Managed file shares for cloud or on-premises deployments.
- **Azure Queues:** A messaging store for reliable messaging between application components.
- **Azure Tables:** A NoSQL store for schema-less storage of structured data.

Azure storage is used as the basis for storage when provisioning a data platform technology such as Azure Data Lake Storage and HDInsight. However, it can also be provisioned for standalone use such as an Azure BLOB Store that is either provisioned as standard storage in the form of magnetic disk storage, or premium storage in the form of solid-state drives (SSD). Azure BLOB store is what we will focus on in our definitions that follow.

When to Use It

If your business scenario is to provision a data store that will store data, but not query it, then creating a storage account configured as a BLOB store is the cheapest option for storing data and work very well with images and unstructured data. In short, if you want to store data in Azure in the cheapest way. Azure BLOB store is the cheapest way to achieve this.

Key Features

Azure storage accounts are scalable and secure, durable and highly available. Microsoft Azure handles hardware maintenance, updates, and critical issues for you and provides SDKs for Azure Storage in a variety of languages – .NET, Java, Node.js, Python, PHP, Ruby, Go, and others – as well as REST APIs. Azure Storage also supports scripting in Azure PowerShell or Azure CLI.

Ingesting Data

To ingest data, a Data Engineer can use either the Azure Data Factory, Azure Storage Explorer or the AzCopy Tool, PowerShell, or Visual Studio. To import file sizes above 2Gb using the File Upload feature, Data Engineers must use PowerShell or Visual Studio. The AzCopy Tool supports a maximum file size of 1Tb and will automatically split into multiple files if the data file exceeds 200Gb.

Querying Data

Creating a storage account as a Blob means that you cannot query the data directly. If you need to query it, then you either need to move the data to a store that can support querying or configure the Azure Storage Account for data lake storage account.

Securing Data

All data written to Azure Storage is encrypted by the service. Azure Storage also provides you with fine-grained control over who has access to your data using keys or Shared Access Signatures. There is also a permissions model using Resource Manager Role-Based Access Control (RBAC), where you can assign roles to users, groups, or applications and set permissions.

Azure Data Lake Storage

Azure Data Lake storage is a Hadoop-compatible data repository that can store any size or type of data. Azure Data Lake is available in two offerings; Generation 1 (Gen1) or Generation 2 (Gen2). Gen1 users should consider upgrading to Gen2, as Gen2 combines the storage services from Gen1 with the benefits of Azure Blob Storage and is performance tuned for processing big data analytics solutions.

Gen2 includes new features like a hierarchical file system, and developers can access the data either through the Blob API or the Azure Data Lake File API (ADLS). An additional benefit to Gen 2 is that it can act a storage layer for a wide range of compute platforms including Azure Databricks, Hadoop or Azure HDInsight without the need to load the data into those systems.

When to Use It

Azure Data Lake is designed for customers who require the ability to store massive amounts of data for big data analytics use cases. For example, Contoso Life Sciences, a cancer research center, analyzes Petabyte scale of genetic data, patient data, and billions of records of related sample data as part of their research. Azure Data Lake Gen2's ability to reduce computation times will make Contoso's Life Sciences research faster and less expensive. Additionally, the compute aspect that sits above this storage can vary and include platforms including HDInsight, Azure Synapse Analytics, Hadoop and Azure Databricks.

Key Features

Key features of Azure Data Lake include unlimited scalability, Hadoop compatibility, security support for both Access Control Lists (ACLs), POSIX-compliance and an optimized ABFS driver designed for big data analytics, zone redundant storage and geo-redundant storage.

Ingesting Data

To ingest data, a Data Engineer can use either the Azure Data Factory, Apache Sqoop, Azure Storage Explorer or the AzCopy Tool, PowerShell, or Visual Studio. To import file sizes above 2Gb using the File

Upload feature, Data Engineers must use PowerShell or Visual Studio. The AzCopy Tool supports a maximum file size of 1Tb and will automatically split into multiple files if the data file exceeds 200Gb.

Querying Data

Data Engineers can query data in Azure Data Lake Store using the Azure Blob Storage API or the Azure Data Lake System (ADLS) API. In addition, compute technologies such as Azure HDInsight, and Azure Databricks are able to query data stored in a Data Lake store

Securing Data

Since Azure Data Lake supports Azure Active Directory Access Control Lists (ACL), security administrators can use familiar Active Directory Security Groups to control data access. Role Based Access Control (RBAC) is available in Gen1. Built-in Security Groups include ReadOnlyUsers, WriteAccessUsers, and FullAccessUsers. Enabling the firewall option will limit traffic to only Azure services. Azure Data Lake automatically encrypts data at rest, protecting data privacy.

Azure Databricks

Databricks is a version of the popular open-source **Apache Spark**¹ analytics and data processing engine. Azure Databricks is the fully managed version of Databricks and is a premium offering on Azure, that brings you an enterprise-grade and secure cloud-based Big Data and Machine Learning platform.

When to Use It

The intention of this platform is to provide the ease of deploying a collaborative Machine Learning environment based on Spark that can be used between data scientists, data engineers, and business analysts.

Key Features

The key features of Azure Databricks are:

- Apache Spark-based analytics platform.
- Enterprise Security.
- Integration with other Cloud Services.

Ingesting Data

Databricks allows several ways to access your data. You can access data stored in an existing file, by uploading a data file from your local system, or by mounting an Azure blob or Data Lake store to Databricks file system. You can use notebooks to connect and ingest data from these sources.

Querying Data

You can use DataFrames to query large data files and several built-in functions of Databricks allow you to aggregate your data in different ways.

¹ <https://spark.apache.org/>

Securing Data

Azure Databricks provides enterprise-grade security, including Azure Active Directory integration, role-based controls, and SLAs that protect your data and your business.

Azure Cosmos DB

Azure Cosmos DB is a globally distributed, multi-model database. It can be deployed using several API models including:

- SQL API
- Mongo DB API
- Cassandra DB API
- Gremlin DB API
- Table API

This multi-model architecture allows the Database Engineer to leverage the inherent capabilities of each model such as MongoDB for semi-structured data, Cassandra for wide columns or Gremlin for graph databases. Using Gremlin, the Data Engineer could create graph entities and perform graph query operations to perform traversals across vertices and edges, achieving sub-second response time for complex scenarios like Natural Language Processing (NLP) or social networking associations. Additionally, applications built upon SQL, MongoDB or Cassandra will continue to operate without changes to the application despite the database server being moved from either SQL, MongoDB, or Cassandra to Azure Cosmos DB.

When to Use It

Data Engineers should deploy Azure Cosmos DB when a NoSQL database of the supported API model is required, at planet-scale, and when low latency performance is required. At the time of writing, Azure Cosmos DB supports five 9s uptime (99.999%) and can support sub 10ms response times when provisioned correctly.

For example, Contoso, an eCommerce retailer based in Manchester, UK sells children's toys. After reviewing some Power BI reports, Contoso's managers noticed a significant uptick in sales in Australia. A review of their customer service cases in Dynamics 365 demonstrated that there were a significant number of customer complaints about the site's shopping cart functionality timing out leaving Australian customers frustrated.

Contoso's network operations manager confirmed that with the company's only data center located in London, the physical distance to Australia was causing the problem. Contoso implements a new solution that provides a local version of the data to users in Australia using the Microsoft Australia East data center. Migrating their existing on-premises SQL Database to Azure Cosmos DB using the SQL API improves performance for the Australian users, as the data can be stored in the UK and replicated to Australia to improve throughput times for Australian customers.

Key Features

Cosmos DB supports 99.999% uptime plus Data Engineers can programmatically (or via the Azure Portal) invoke a regional failover. A Cosmos database will automatically failover if there is a regional disaster. Additionally, using Azure Cosmos DB's multi-master replication, it's common to be able to achieve less than one second response time from anywhere in the world. Microsoft Azure Cosmos DB is guaranteed to achieve less than 10ms response time for reads and writes.

To maintain the consistency of the data in Azure Cosmos DB, the engineering team introduces a new set of consistency levels that address the unique challenges facing planet-scale solutions. Consistency levels include Strong, Bounded Staleness, Session, Consistent Prefix, and Eventual.

Ingesting Data

To ingest data into Azure Cosmos DB, Data Engineers can use Azure Data Factory, create an application that writes data into Azure Cosmos DB through its API, upload JSON documents, or directly edit the document.

Querying Data

Data Engineers can create stored procedures, triggers, and user-defined functions (UDF's) or leverage the JavaScript query API. Additionally, there are other methods available to query the other APIs within Cosmos DB. For example, there is a Graph visualization pane in the Data Explorer component of Cosmos DB.

Securing Data

Azure Cosmos DB supports data encryption, IP Firewall configurations, and access from virtual networks. Encryption is applied automatically. User authentication is token based, and Azure Active Directory provides role-based security. Azure Cosmos DB meets many security compliance certifications including HIPAA, FedRAMP, SOCS, and HITRUST.

Azure SQL Database

Azure SQL Database is a managed relational database service in Azure that supports structures such as relational data, and unstructured formats such as spatial, and XML data. It provides Online Transaction Processing (OLTP) that can scale on demand while employing the comprehensive security and availability features that database services on Azure have.

Important

This unit is focused on **Azure SQL Database**, the *Platform as a Service* (PaaS) database offering. This unit doesn't cover Microsoft SQL Server installed locally or within an Azure virtual machine (VM). While similar in nature, the setup, configuration, and benefits of Microsoft SQL Server are different from Azure SQL Database.

When to Use It

Use Azure SQL DB when the scale up and scale down of OLTP systems needs to be performed on demand, and an organization wishes to take advantage of the security and availability features without the risk of investment in capital expenditure, and the inevitable increase in operational spending to invest in these complex on-premises system counterparts.

Azure SQL DB can prove to be a more flexible platform choice than on-premises SQL Server solution as the provisioning and configuration can be performed in minutes and backed up by the Azure Service Level Agreement (SLA).

Key Features

Azure SQL Database delivers predictable performance with multiple resource types, service tiers, and compute sizes that provides dynamic scalability with no downtime, built-in intelligent optimization,

global scalability and availability, and advanced security options — all with near-zero administration. These capabilities allow you to focus on rapid app development and accelerating your time to market, rather than allocating precious time and resources to managing virtual machines and infrastructure.

Ingesting and Processing Data

Azure SQL Database can ingest data through application integration from a wide range of developer Software Development Kits (SDKs) such as .NET, Python, Java, and Node.js. Beyond applications ingestion of data can be performed through Transact-SQL techniques and from the movement of data using Azure Data Factory.

Querying Data

Data Engineers can use Transact-SQL (T-SQL) to query the contents of a SQL Database, taking advantage of a wide range of standard SQL features to filter, order, and project the data into whatever form required.

Securing Data

Azure SQL Database provides a range of built-in security and compliance features to help your application meet various security and compliance requirements including:

- Advance Threat Protection
- SQL Database Auditing
- Data Encryption
- Azure Active Directory authentication
- Multi-factor Authentication
- Compliance certification

Azure Synapse Analytics

Azure Synapse Analytics provides a unified environment by combining the enterprise data warehouse of SQL, the Big Data analytics capabilities of Spark, and data integration technologies to ease the movement of data between both; and from external data sources. Using Azure Synapse Analytics, you are able to ingest, prepare, manage, and serve data for immediate BI and machine learning needs more easily.

When to Use It

As data loads increase, the processing time for on-premises data warehousing solutions increase. An organization may have to look at a cloud-based alternative to reduce the processing time so that business intelligence reports are made available in a timely manner. Many organizations will look to scale up on-premises servers first, but as the scale out approach reaches its physical limits, it is more pragmatic to use a solution that has Petabyte scale without the complexity of installing and configuring it. You also use Azure Synapse analytics to reduce the complexity of processing data warehouse loads between a data lake, Spark and a Data Warehousing system

Key Features

From a data warehouse perspective, Azure Synapse Analytics uses Massive Parallel Processing (MPP) to run queries across petabytes of data quickly. In tandem is a Spark engine that enables you to work with predictive analytical workloads. Specifically, it provides a unified, open source, parallel, data processing framework for Big Data Analytics using the following frameworks:

- Spark SQL for batch processing
- Spark MLlib for Machine Learning
- Spark Streaming Stream processing
- GraphX for Graph Computation

There is a data integration engine that enables you to transfer the data between both systems with ease, and it can also allow you to simplify the ingestion of data from a data lake store. Since the storage is separated from the compute nodes, the compute nodes can be scaled independently to meet any demand, at any time.

Ingesting and Processing Data

Azure Synapse Analytics uses the Extract, Loads, and Transform (ELT) approach. SQL Professionals will already be familiar with tools such as BCP and the SQL Bulk Copy API, but Data Engineers working with Azure Synapse Analytics will learn the power of PolyBase that can load data very quickly. PolyBase is a technology that removes the complexity for Data Engineers to leverage big data ingestion and processing techniques by offloading complex calculations to the cloud. Developers can use stored procedures, labels, views, and SQL to develop applications to make use of PolyBase. You also have the capability to prioritize workloads.

Querying Data

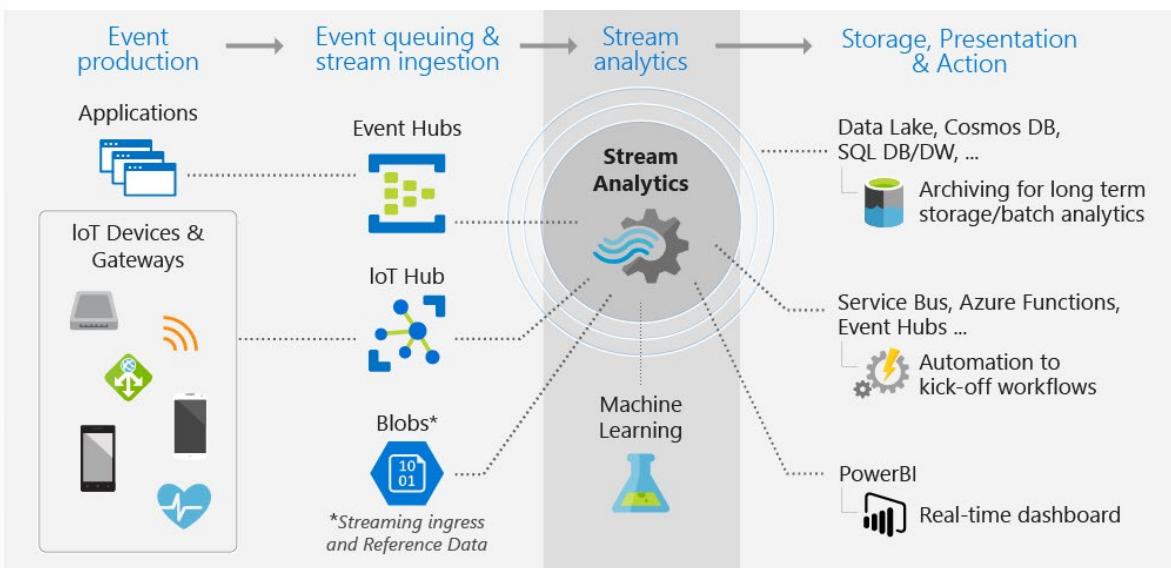
Data Engineers can use the familiar Transact-SQL to query the contents of a data warehouse in Azure Synapse Analytics, taking advantage of a wide range of features including WHERE, ORDER BY and GROUP BY clauses when querying the data. There are additional Transact SQL constructs that may be new, such as CREATE TABLE AS SELECT that is used with PolyBase to perform the fast loading of data. You also can make use of Python, Scala and R against the Spark engine. There is also support for .Net and Java.

Securing Data

Azure Synapse Analytics supports both SQL Server authentication as well as Azure Active Directory. For higher security environments, the Data Engineer can configure multi-factor authentication. From a data perspective, Azure Synapse Analytics supports both column-level and row-level security.

Stream Analytics

Applications, sensors, monitoring devices, and gateways broadcast continuous event data known as data streams. Streaming data is high volume with a lighter payload than non-streaming systems. Azure Stream Analytics allows Data Engineers to process streaming data and respond to data anomalies in real-time. Business cases include Internet of Things (IoT) monitoring, web logs, remote patient monitoring, or Point of Sale (POS) systems.



When to use it

If your organizations' challenge is to respond to data event in real-time or perform a post mortem of large batches of data in a continuous time bound stream, then Azure Stream Analytics is for your organization. The decision must be made whether your organization will be working with streaming data or batch data.

Azure Stream Analytics will ingest streaming data from applications or IoT devices and gateways into an event hub or an Internet of Things (IoT) hub in real-time. At which point the event or IoT hub will stream the data into Stream Analytics for real-time analysis.

Batch systems process groups of data stored in an Azure Blob store in a single job that executes at a pre-defined interval. Batch systems are not suitable for certain business intelligence systems where the pre-defined interval cannot be tolerated for performance reasons. For example, an autonomous vehicle cannot wait for a batch system to adjust for driving. Another example includes a fraud detection system that must prevent the financial transaction from being approved in real time.

Ingesting Data

Data Engineers configure ingestion components of Azure Stream Analytics by configuring data inputs from first-class integration sources including Azure Event Hubs, Azure IoT Hub, or Azure Blob storage.

IoT hubs are the cloud gateway that connects IoT devices to gather data to drive business insights and automation. IoT Hub includes features that enrich the relationship between your devices and your back-end systems. Bi-directional communication capabilities mean that while you receive data from devices, you can also send commands and policies back to devices, for example, to update properties or invoke device management actions. It can also authenticate access between the IoT device and the IoT hub.

Event hubs provide a big data streaming service of Azure. It is designed for high throughput data streaming scenarios where customers may send billions of requests per day. Event Hubs uses a partitioned consumer model to scale out your data stream and is integrated into the big data and analytics services of Azure including Databricks, Stream Analytics, ADLS, and HDInsight. It provides authentication through a shared key.

Azure Storage can be used to stored data before being batched processed.

Processing Data

Data Engineers configure Azure Stream Analytics jobs with input and output pipelines to process streaming data. Inputs are provided by event hubs, IoT hubs or Azure storage. Stream Analytics can route job output to many storage systems such as Azure Blob, Azure SQL Database, Azure Data Lake Stores, or Azure Cosmos DB.

After storing, you can run batch analytics with Azure HDInsight or send the output to another service such as event hubs for consumption or to Power BI for real-time visualization by using Power BI streaming API.

Querying Data

To define job transformations, you use a simple, declarative Stream Analytics query language that lets you author complex temporal queries and analytics using simple SQL constructs. Stream Analytics query language is consistent with the SQL language, familiarity with SQL language is sufficient to get started with creating jobs.

Securing Data

Security for Azure Stream Analytics is handled at the transport layer between the device and the Azure IoT Hub. Streaming data is generally discarded after the windowing operations are complete. Event hubs secure transfer of data using a shared key. If the Data Engineer chooses to store the data, security will be provided by the storage device selected.

Other Azure Data Platform Services

Azure Data Factory

Azure Data Factory (ADF) is a cloud integration service that orchestrates that movement of data between various data stores. Data Engineers can create data-driven workflows in the cloud for orchestrating and automating data movement and data transformation. Using Azure Data Factory, you can create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores.

It can process and transform the data by using compute services such as Azure HDInsight Hadoop, Spark and Azure Machine Learning. Additionally, you can publish output data to data stores such as Azure SQL Data Warehouse for business intelligence (BI) applications to consume. Ultimately, through Azure Data Factory, raw data can be organized into meaningful data stores and data lakes for better business decisions.

Azure HDInsight

Azure HDInsight provides technologies for ingesting, processing, and analyzing big data to support batch processing, data warehousing, IoT, and Data Science. Azure HDInsight is a low-cost cloud solution containing several technologies including Apache Hadoop, Apache Spark, Apache Kafka, Apache HBase, Interactive Query, and Apache Storm.

Apache Hadoop includes Apache Hive, Apache HBase, Spark, and Kafka. Hadoop stores data using a file system (HDFS) while Spark stores data in memory, making Spark approximately 100 times faster.

Apache HBase is a NoSQL database built upon Hadoop commonly used for search engines and includes automatic failover.

Apache Storm is a distributed real-time streamlining analytics solution.

Apache Kafka is an open-source platform used to compose data pipelines and to provide message queue functionality which provides publishing/subscribing of real-time data streams.

Azure Data Catalog

With Data Catalog, any user (analyst, data scientist, or developer) can discover, understand, and consume data sources. Data Catalog includes a crowdsourcing model of metadata and annotations. It is a single, central place for all of an organization's users to contribute their knowledge and build a community and culture of data sources that are owned by an organization.

It is a fully managed cloud service. Users can discover the data sources they need and understand the data sources they find, and use Data Catalog to help organizations document the information about their data sources.

Summary

Whether Data Engineers are working with small data or big data, the Microsoft Azure platform provides a rich set of technologies to analyze text, image, relational, non-relational, or streaming data. Data Engineers can choose the technologies that most appropriately meet their business requirements and scale those solutions to meet demand in a secure way.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

What data platform technology is a globally distributed, multi-model database that can offer sub second query performance?

- Azure SQL Database
- Azure Cosmos DB
- Azure Synapse Analytics

Question 2

Which of the following is the cheapest data store to use when you want to store data without the need to query it?

- Azure Stream Analytics
- Azure Databricks
- Azure Storage Account

Question 3

Which Azure Service would be used to store documentation about a data source?

- Azure Data Factory
- Azure Data Catalog
- Azure Data Lake

Data Engineering Roles and Responsibilities

Data Engineering Roles and Responsibilities

Enterprise data engineering projects require the Data Engineer to provide data to a wide range of professionals, ranging from Information Workers to Data Scientists. Many of today's organizations undertaking data transformation project will still use traditional roles such as Business Analysts, but the evolution of data types and changing methods of analytics means that new roles are appearing on these projects.

Suppose you work for an organization that wishes to undertake a digital transformation project where data plays a key role. You'll still incorporate the best of the organizations current systems such as the OLTP and Data Warehouse systems, but the organization sees the potential to exploit sales opportunities and improve operational expenditure through predictive analytics.

As the database administrator for your company, it's your job to move existing data into new systems and connect it all together. Where do you start? Is your job now obsolete? The short answer is no. You have nothing to lose as your current skills are still essential, and everything to gain in learning new skills and technologies to unlock the potential of modern data engineering techniques.

Learning Objectives

In this module you will:

- List the new roles of modern data projects
- Outline Data Engineering Practices
- Explore the high-level process for architecting a data engineering project

Roles and Responsibilities

Large data projects can be complex, often with hundreds of decisions that need to be made and executed. Multiple people will typically be involved, each playing a specific role to take the project from design to production. Some of these roles, such as business stakeholders, business analysts, and business intelligence developers are well-known and still valuable. However as data processing techniques have changed with technology improvements, new roles are starting to appear that provide specialized skills to help streamline the data engineering process.

In particular, three roles are starting to become prevalent in modern data projects.

1. Data Engineer
2. Data Scientist
3. Artificial Intelligence Engineer
NEW 1. Data Analyst
NEW 1. Data Management

The Data Engineer

Data Engineers are responsible for the provisioning and configuration of both on-premises and cloud data platform technologies. They manage and secure the flow of structured and unstructured data from multiple of sources. The data platforms they manage can include relational databases, non-relational databases, data streams, and file stores. If that wasn't enough, they also must ensure that the data

services integrate seamlessly with other data platform technologies or application services such as Cognitive Services, Azure Search, or even Bots, in a secure way.

The Azure Data Engineer focuses on data-related tasks in Azure. Primary responsibilities include ingest-ing, egressing, and transforming data from multiple sources using various services and tools. The Azure Data Engineer collaborates with business stakeholders to identify and meet data requirements while designing and implementing solutions. They also manage, monitor, and ensure the security and privacy of data to satisfy business needs.

This role is different from a Database Administrator in that their scope of work goes well beyond just looking after a database and the server it's hosted on. Data Engineers must also be able to acquire, ingest, transform, validate, and cleanup data based upon business requirements in a process referred to as *data wrangling*.

Their role can add tremendous value to both business intelligence and data science projects. It is widely regarded that data wrangling can consume a substantial amount of time on such projects. Having the Data Engineer perform these tasks can accelerate the development of these projects and enable Data Scientists to focus on other areas of their work.

Both Database Administrators and Business Intelligence professionals can easily transition to a data engineer role by learning the new tools and technology used to process these large amounts of data effectively.

The Data Scientist

Data Scientists perform advanced analytics work to help drive value from data for a business. The type of work can vary from descriptive analytics; which is an analysis of the data through a process known as Exploratory Data Analysis (EDA) through to performing predictive analytics. Predictive analytics enables a data scientist to perform modeling techniques using machine learning to create anomaly/pattern detection solutions through to predictive forecasting models. This represents one aspect of their work, and some may even go into the realms of deep learning, iteratively performing experiments to solve a complex data problem through the use of complex customized algorithms.

Anecdotal evidence supports the notion that most of the work that is spent on a Data Science project is data wrangling and features engineering. Data scientists appreciate the support they can receive from Data Engineers in this area to accelerate the experimentation process, as they recognize that Data Engineers possess the skill required to perform data wrangling successfully.

The Artificial Intelligence (AI) Engineer

Artificial Intelligence (AI) Engineers work with AI services such as Cognitive Services, Cognitive Search, and the Bot Framework. Cognitive Services includes a wide set of services including Computer Vision, Text Analytics, Bing Search, and Language Understanding (LUIS) among others. Rather than creating models, AI Engineers implement the prebuilt capabilities of Cognitive Service APIs and embed this intelligence within a new or existing application or more recently Bots. AI Engineers will rely on the expertise of Data Engineers to help them store information that is generated from AI. For example, an AI engineer may want a Data Engineer to provision a Cosmos DB instance to store metadata and tags generated by Computer Vision application that processes images.

The Data Analyst

Data Analysts enable businesses to maximize the value of their data assets by using 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. Data analysts also collaborate with key stakeholders across verticals to deliver relevant insights based on identified business requirements.

The tasks of a Data Analyst are those that focus on helping businesses understand and find meaning in their data. They work closely with the Data Engineer to understand different data sources. Equally, a data analyst may work with a Data Scientist to look at statistics in the data and identify patterns and trends from the data.

Data Engineering Practices

Let's start by looking at some of the routine tasks a Data Engineer will do when working with Azure.

- **Provision** - Data Engineers are expected to provision data platform technologies to store the data.
- **Process** - Data engineers will create data processing pipelines to move data between various data stores.
- **Secure** - Data Engineers must ensure that the data is secured from unauthorized access or malicious attacks
- **Monitor** - Data Engineers must set up proactive and reactive monitors to ensure that the solution meets Service Level Agreements and are within budget.
- **Disaster Recovery** - Data Engineers must ensure that there is the ability to recover data in the event of a disaster.

How is a Data Engineer different than a Database Administrator?

The Data Engineer role overlaps with the Database Administrator (DBA) in many ways in terms of the broad tasks they do. The difference is in scope and focus. Data Engineers work with a broader range of technologies than just databases and they focus on *cloud implementations* rather than on-premises servers.

Moving data around

When it comes to the transfer and movement of data, there are several different approaches Data Engineers will take.

Extract, Transform, Load (ETL) Process

Data Engineers may sometimes perform ETL process when processing data. The extract may come from many sources including databases, files, and streams. Each data source has unique data formats and can be structured, semi-structured, or unstructured. In Microsoft Azure, data sources include Azure Cosmos DB, Azure Data Lake, files, and Azure Blob Storage.

Extract

During the extraction process, Data Engineers will perform the following steps.

1. Define Data Source

To begin data extraction in Microsoft Azure, Data Engineers will identify source details such as the resource group, subscription, and identity information such as a key or secret.

2. Define the Data

After the connection information is established, the Data Engineer will identify the data to be extracted. This might be done with a database query, a set of files, or an Azure Blob Storage name for blob storage.

Transform

3. Define the Transformation

After extracting the data, Data Engineers define any data transformation operations, which can include splitting, combining, deriving, adding, removing, or pivoting columns. Data may also need to be aggregated or merged. Data Engineers will need to identify field mapping between the data source and data destination.

Load

4. Define the Destination

When performing a load, many of the Microsoft Azure destinations can accept data in the JavaScript Object Notation (JSON), file, or blob formats. Data Engineers may also need to write code to interact with application APIs. Azure Data Factory contains built-in support for Azure Functions that supports many programming languages such as node.js, .NET, Python, and Java. While Extensible Markup Language (XML) was common in the past, most systems have migrated to JSON due to its increased flexibility as a semi-structured data type.

5. Execute the Job

After the ETL operations have been defined, the Data Engineer will test the ETL job in a development or test environment and then migrate the job to a production environment to load the production system.

6. Monitoring execution

ETL operations can contain many complex processes and workflows. Setting up a proactive and reactive monitoring system can provide useful information when things go wrong. How logging is set up depends on the technology utilizing it.

ETL Tools

Data Engineers will use several tools to perform ETL, the most common is Azure Data Factory, which provides a robust set of tools and nearly 100 enterprise connectors. Also, there are also a wide variety of languages that can be used within Azure Data Factory to perform the transformations.

If you also require a repository for maintaining information about the data sources and dictionaries within an organization. Azure Data Catalog can prove useful to store this information centrally.

Extract, Load, and Transform Process (ELT)

The advent of Azure has meant that there are technologies that can handle unstructured data at unlimited scale. This has meant that the paradigm for loading and transforming data has shifted from ETL to ELT. The benefit of this approach is that data can be stored in its original source format, be it JSON, XML, PDFs, or images. This has provided the opportunity to make use of this source data to multiple downstream systems as the structure of this data defined during the Transform phase.

As a result, data is extracted and loaded in its native format to reduce the time it takes to load into a destination system and to limit resource contention on the data sources it is extracted from. The steps are the same as ETL, just in a different order.

Extract, Load, Transform, Load Process (ELT)

This is the same process as ELT. There is just a final load into a destination system.

Holistic Data Engineering

As the types of analysis in organizations are evolving to incorporate predictive and preemptive analytics, it is more important for Data Engineers to look at data projects holistically. In the past data, professionals would focus on ETL, but the latest developments to data platform technologies lend themselves to an ELT Approach. Therefore, the first steps of Data projects can be broken down into phases that reflect the ELT approach, including:

- **Source** – identify the source systems to extract from
- **Ingest** – identify the technology and method for loading data
- **Prepare** – identify the technology and method for transforming/preparing the data

It's also important to consider the technologies that will be used to analyze and consume the data within the project. These are the next two steps in the process:

- **Analyze** – identify the technology and method for analyzing the data
- **Consume** - identify the technology and method for consuming and presenting the data

In traditional descriptive analytics projects this may have involved transforming data in the Azure Analysis Services, and then using Power BI to consume the analyzed data. New AI technologies such as Azure Machine Learning Services and Azure Notebooks provides a wider range of technologies to automate some of the required analysis.

Moving between phases

Furthermore, it can't be assumed there's a linear flow through each of these phases. For example, due to the iterative nature of Machine Learning experimentation, it's not uncommon for the *analyze* phase to reveal issues such as missing source data or additional transformation steps needed to get the results desired.

To fully appreciate this process, let's examine it using a high-level architecture example.

Architecting Projects

The following example can help demonstrate how to perform holistic data engineering using the source, ingest, prepare, analyze, and consume approach.

Contoso Health Network recently deployed IoT devices into its Intensive Care Unit (ICU) to be able to capture patient biometric monitoring data in real time. The hospital intends on processing the streaming data from the IoT devices to help the physicians treat these patients. Contoso's research center would like to store the biometric data for further analysis in the future. Their researchers would like to understand further what treatment methods have been used to improve the quality of care and to reduce the likelihood of the patient being readmitted to the hospital through Azure Machine Learning. Contoso's Chief Medical Officer would like a historical view of this data that is visualized.

Contoso's technical architect reviewed the business case and proposed the following technologies:

- **Azure IoT Hub.** To capture real-time data from the ICU's IoT devices.
- **Azure Streaming Analytics.** To stream the IoT data, create windows, aggregations, integrate Azure Machine Learning, and to further enrich data.

- **Azure Data Lake Gen2.** To store the biometric data at speed.
- **Azure Data Factory.** To perform the extract, load, transform, and load operations from the data lake store into SQL DW.
- **Azure SQL DW.** To provide data warehousing services to support the Chief Medical Officer's needs.
- **Power BI.** To create the patient dashboard. Part of the dashboard will host real-time telemetry regarding the patient's condition while the other visual will display the patient's recent history.
- **Azure Machine Learning Services** – used to process both the raw and aggregated data to perform patient readmittance predictive analytics.

Contoso's Data Engineer creates a work plan to implement the Extract, Load, Transform, and Load operations.

Provisioning workflow

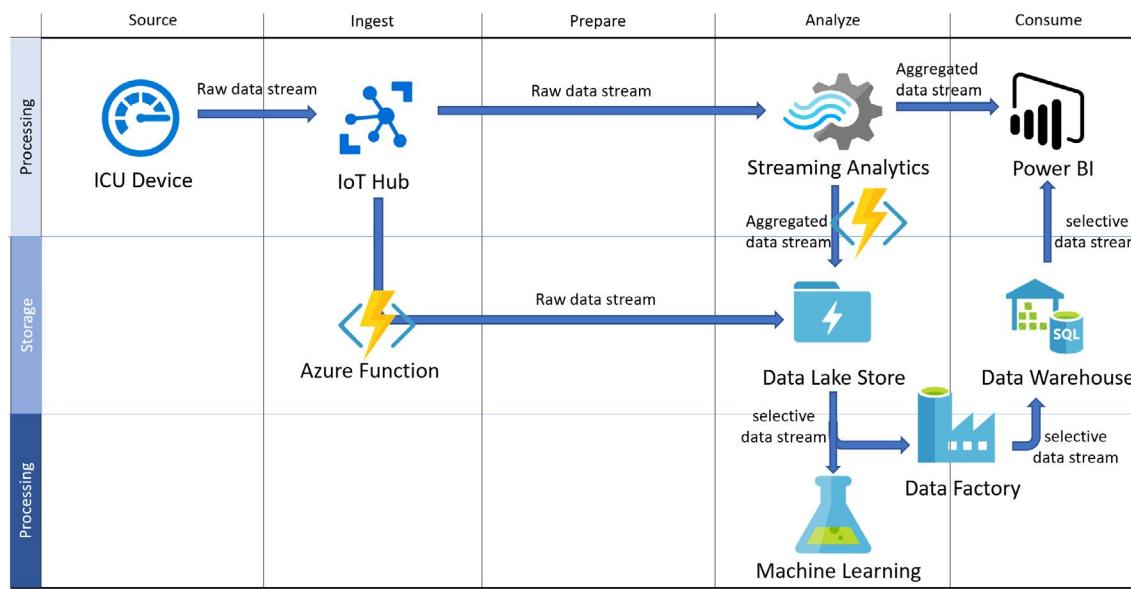
1. Provision Azure Data Lake Gen2
2. Provision Azure Synapse Analytics
3. Provision Azure IoT Hub
4. Provision Azure Streaming Analytics
5. Provision Azure Machine Learning
6. Provision Azure Data Factory
7. Provision Power BI

Holistic Workflow

1. Configure Azure IoT Hub to capture data from the ICU IoT devices.
2. Connect Azure IoT Hub to Azure Streaming Analytics. Configure the windowing functions for the ICU data that will aggregate the data for each window. At the same time, configure the IoT hub to dump the streaming data to the Azure Data Lake using Azure Functions.
3. Configure Azure functions to store the Azure Streaming Analytics aggregates to Azure Data Lake Gen2.
4. Use Azure Data Factory to load data from the Data Lake into Azure Synapse Analytics to support the Chief Medical Officer's requirements. Transformations can occur within Azure Synapse Analytics once loaded.
5. In parallel, connect Azure Machine Learning Service to the Azure Data Lake Store to perform the Predictive Analytics.
6. Connect Power BI to Streaming Analytics to pull the real-time aggregates for the Patient data and SQL Data Warehouse to pull the historical data to create a combined dashboard.

Related high-level architecture diagram

The following high-level architecture diagram provides a visualization of the solution.



Summary

In this module, we reviewed the roles and responsibilities of the Data Engineer's, and how they interact with other data and AI professionals. We have briefly explored the data engineering practices that are commonly undertaken in a role. Finally, we explored a high-level process for architecting a data engineering project.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which role works with services such as Cognitive Services, Cognitive Search, and the Bot Framework?

- Data Engineer
- Data Scientist
- AI Engineer

Question 2

Which Azure Data Platform technology is commonly used to process data in an ELT framework?

- Azure Data Factory
- Azure Databricks
- Azure Data Lake

Course Case Study

Course Case Study

In this section, you will be introduced to the case study that will be used in many of the labs on this course. You will be then given time to read the case study and then answer some questions about the case study, and have the opportunity to ask questions.

Learning objectives

In this module you will:

- Read the Course Case Study

Case Study – AdventureWorks Cycles

AdventureWorks sells bicycles and bicycle parts directly to customers and distributors. The company currently has a single office in the Netherlands, and have been selling bicycles in the United States, Germany and Spain through a chain of distributors and through online sales on its website. The fulfillment of delivery is done by local distribution centers.

The company is planning to expand by establishing new offices because the sales growth in these countries has been increasing over the last 3 years. The location are:

- Tokyo, Japan
- Seattle, USA
- Chicago, USA
- Berlin, Germany
- Barcelona, Spain
- Paris, France

In a highly competitive market, in which AdventureWorks has been in business for the last 15 years, it wants to become the most innovative bicycle company, providing both current and future bicycle owners with best in class technology and service that provides unique experiences.

The Research and Development department of AdventureWorks has successfully conceived the next wave of innovative products, and they are relying on Data Engineers, AI Engineers and Data Scientists to assist with both the design and implementation of the solution.

Given the increased level of sales and expansion at global scale, the existing data infrastructure won't meet the overall business requirements or the future growth that AdventureWorks aspires to. The Chief Information and Technology Officers have expressed the desire to abandon existing on-premises systems and move to the cloud to meet the growth expected. This is supported by the CFO as there has been a request for replacement hardware as the existing infrastructure comes to its end of life. The CFO is aware that the cloud could offer alternatives that are more cost efficient.

As a Senior Data Engineer, you will assist AdventureWorks in the solution design and implementation to meet the business, functional and technical requirements that the company has set forth to be successful for growth, expansion, and innovation strategies. You will execute this in a way that minimizes operational costs and can be monitored for effectiveness.

In a discovery workshop you ascertained the following information:

AdventureWorks Website

The web developers at AdventureWorks are transferring the existing website from an on-premises instance of IIS, to an Azure Web App. They have requested that a data store is made available that will hold the images of the products that are sold on the website.

Current Sales / Ordering system

The current software on which bicycle purchases are tracked, is a web-based application which directly stores order information into an on-premises SQL Server database named AdventureWorks2012. The current application is deployed with high-availability provided by SQL Server 2012 Always-on Availability groups. Due to global expansion and data governance requirements, AdventureWorks will transition this system to better serve their customers and will be looking for global availability of its application and data sales and ordering purposes, particularly during the months of November and December when demand for bikes grow ahead of the holiday period.

Data Analysis

The business reporting is currently being provided by a single on-premises database that is configured as a data warehouse, it holds a database named AdventureWorksDW which is used to provide historical reporting and descriptive analytics. In recent times, that server has been struggling to process the reporting data in a timely manner, as a result the organization has evaluated the data warehouse capabilities of Azure Synapse Analytics and want to migrate their on-premises data to this platform. Your team should ensure that access to the data is restricted.

In addition, AdventureWorks would like to take their data analytics further and start to utilize predictive analytics capabilities. This is currently not an activity that is undertaken. The organization understands that a recommendation or a text analytics engine could be built and would like you to direct them on what would be the best technology and approach to take in implementing such a solution that is also resilient and performant.

You are also assessing the tooling that can help with the extraction, load and transforming of data into the data warehouse, and have asked a Data Engineer within your team to show a proof of concept of Azure Data Factory to explore the transformation capabilities of the product

Customer Service / Presales

Customer service and pre-sales departments are currently experiencing scale issues due to the high call volumes. The organization wants to support the customer services staff in handling the call volumes through the implementation of chat bots in which future bicycle owners can:

- Find which bicycle is best for them:
 - Through a set of questions with the chat bot, custom recommendations are given to potential bike owners, who then can take the recommendation and place an order, or can be redirect to a sales specialist to help them with their needs
- Check status on current orders:
 - Retrieve status on current orders, and estimated delivery times

- Find bicycle parts suitable for their existing bicycle:
 - Existing bicycle owners can find recommended bicycle parts and accessories based on the serial number or model number of their bicycle
 - Existing bicycle owners, can upload a picture of their bicycle or take a picture of the serial number of their bicycle to assist with the identification of their bicycle and have recommended bicycle parts

Over the last few years the customer services departments have observed an increase in calls from fraudulent customer who are asking for support for bikes that are no longer in warranty, or bikes that have not even been purchased at AdventureWorks. The department are currently relying on the experience of customer services agents to identify this. As a result, they would like to implement a system that can help the agents track in real-time who could be making a fraudulent claim.

Finally, given its global expansion, the customer service / presales chat bot needs to respond to requests for data in near real-time regardless of where the customer is located. The chatbot should also support multiple languages such as Dutch, German, French, English, Spanish, and Japanese. This work will be handled by the AI Engineers, but they have requested a platform is provided by the Data Engineer that enables them to store conversation history.

Social Media Analysis

In recent years, the marketing department at the organization have run a wide variety of twitter campaigns at various times of the year. They are keen to measure the impact of their work by tracking social media assets such as hashtags during those campaigns. They would like to have the capability of tracking any hashtag of any name.

Connected bicycle

AdventureWorks Bicycles can be equipped with an innovative built-in bicycle computer which consist of automatic locking features of the bicycle, as well as operational status. Information captured by this bicycle computer includes:

- Bicycle model, serial number and registered owner
- Bicycle location (latitude longitude)
- Current status (stationary, in motion)
- Current speed in kilometers per hour
- Bicycle Locked / Unlocked
- Bicycle parts and components information (on electrical bicycles)

First party and 3rd party applications can have access to the information of the bicycle computer that must be secure and for the integration into mobile applications and real time display of location and bike ride sharing information.

Furthermore, daily summary data can be saved to flat files that include Bicycle model, serial number, registered owner and a summary of the total miles cycled per day and the average speed.

Bicycle Maintenance services

Existing bicycle owners can opt in to getting notifications on when their bicycle needs repair, based on:

- Telemetry from electrical bicycle based on sensor data
- Bicycle usage information coming from the built-in bicycle computers based on average mileage / wear and tear

This predictive maintenance scenario is a service in which bike owners can opt-in, offered as a paid service.

Finally, all services that are proposed should have a comprehensive business continuity that meets the corporate objective of minimizes restore times when recovering the data for a given service.

Summary

In this section, you have read the case study of AdventureWorks cycles to gain an understanding of their business requirements and digital transformation aspirations.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which requirement is likely to be the easiest to implement from a data engineering perspective?

- Social Media Analysis
- Connected Bicycle
- AdventureWorks Website

Question 2

Which data platform technology could be used to implement the predictive analytics capabilities that AdventureWorks desires?

- Azure Stream Analytics
- Azure Databricks
- Azure Storage Account

Question 3

Which data platform technology could be used to help AdventureWorks scale globally?

- Azure Data Factory
- Azure Data Catalog
- Azure Cosmos DB

Module Summary

Azure for the Data Engineer

In this module, you explored how the world of data has evolved and how cloud data platform technologies are providing new opportunities for businesses to explore their data in different ways. You gained an overview of the various data platform technologies that are available, and how a Data Engineers role and responsibility have evolved to work in this new world for an organizations benefit. You also reviewed a fictitious case study that will be used as the basis for the labs.

Learning objectives

In this module, you have learned:

- The evolving world of data.
- The services in the Azure Data Platform.
- The tasks that are performed by a Data Engineer.
- A fictitious Case Study for use in labs.

Post Course Review

After the course, consider visiting **the Microsoft Customer Case Study site²**. Use the search bar to search by an industry such as healthcare or retail, or by a technology such as Azure Cosmos DB or Stream Analytics. Read through some of the customers that are using these technologies to enhance their business models.

Important

Remember

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

² <https://customers.microsoft.com>

Answers

Question 1

What data processing framework will be mainly used by Data Engineers to ingest data into cloud data platforms on Azure?

- Online Transactional Processing (OLTP)
- Extract, Transform and Load (ETL)
- Extract, Load and Transform (ELT)

Explanation

Extract, Load and Transform (ELT) is a typical process for ingesting data from an on-premises database into the cloud. Online Transactional Processing (OLTP) describes a database model for processing transactional data. Extract, Transform and Load (ETL) is a typical process for ingesting data from an on-premises database to an on-premises data warehouse.

Question 2

What type of data can have its own schema at defined at query time?

- Structured data
- Cosmos DB
- Un-structured data

Explanation

Un-structured data will typically have its schema defined at query time. This means that data can be loaded into a data platform in its native format. Structured data typical has its schema defined at creation time and Cosmos DB is a data platform technology that exists in Microsoft Azure.

Question 3

Duplicating customers content for redundancy and meeting Service Level Agreements in Azure meets which cloud technical requirement?

- Maintainability
- High Availability
- Multi-lingual support

Explanation

High Availability duplicates customers content for redundancy and meeting Service Level Agreements in Azure. Maintainability ensure that a data platform can be easily maintained and Multi-lingual support enables a data platform technology to support multiple languages.

Question 1

What data platform technology is a globally distributed, multi-model database that can offer sub second query performance?

- Azure SQL Database
- Azure Cosmos DB
- Azure Synapse Analytics

Explanation

Azure Cosmos DB is a globally distributed, multi-model database that can offer sub second query performance. Azure SQL Database is a managed relational database service in Azure. Azure Synapse Analytics is an enterprise-class cloud-based enterprise data warehouse designed to process massive amounts of data.

Question 2

Which of the following is the cheapest data store to use when you want to store data without the need to query it?

- Azure Stream Analytics
- Azure Databricks
- Azure Storage Account

Explanation

Azure Storage offers a massively scalable object store for data objects and file system services for the cloud. Creating a storage account as a Blob means that you cannot query the data directly. Azure Stream Analytics will ingest streaming data from applications or IoT devices and gateways into an event hub or an Internet of Things (IoT) hub in real-time. At which point the event or IoT hub will stream the data into Stream Analytics for real-time analysis. Azure Data Bricks is a serverless platform optimized for Microsoft Azure which provides one-click setup, streamlined workflows and an interactive workspace for Spark-based applications.

Question 3

Which Azure Service would be used to store documentation about a data source?

- Azure Data Factory
- Azure Data Catalog
- Azure Data Lake

Explanation

Azure Data Catalog is a single, central place for all of an organization's users to contribute their knowledge and build a community and culture of data sources that are owned by an organization. Azure Data Factory (ADF) is a cloud integration service that orchestrates the movement of data between various data stores. Azure Data Lake storage is a Hadoop-compatible data repository that can store any size or type of data.

Question 1

Which role works with services such as Cognitive Services, Cognitive Search, and the Bot Framework?

- Data Engineer
- Data Scientist
- AI Engineer

Explanation

Artificial Intelligence (AI) Engineers work with AI services such as Cognitive Services, Cognitive Search, and the Bot Framework. Data Engineers are responsible for the provisioning and configuration of both on-premises and cloud data platform technologies. Data Scientists perform advanced analytics work to help drive value from data for a business.

Question 2

Which Azure Data Platform technology is commonly used to process data in an ELT framework?

- Azure Data Factory
- Azure Databricks
- Azure Data Lake

Explanation

Azure Data Factory (ADF) is a cloud integration service that orchestrates the movement of data between various data stores. Azure Databricks adds additional capabilities to Apache Spark including fully managed Spark clusters and an interactive workspace. Azure Data Lake storage is a Hadoop-compatible data repository that can store any size or type of data.

Question 1

Which requirement is likely to be the easiest to implement from a data engineering perspective?

- Social Media Analysis
- Connected Bicycle
- AdventureWorks Website

Explanation

The AdventureWorks Website requirement is the need to create a data store to host images. This can be achieved using an Azure Blob store. The Connected Bicycle is a more complex requirement as it will be dealing with Streaming Data from a bicycle. The Social Media Analysis is another complex requirement as it requires the collection of streaming data of an application.

Question 2

Which data platform technology could be used to implement the predictive analytics capabilities that AdventureWorks desires?

- Azure Stream Analytics
- Azure Databricks
- Azure Storage Account

Explanation

Azure Databricks is a server-less platform optimized for Microsoft Azure which provides one-click setup, streamlined workflows and an interactive workspace for Spark-based Machine learning applications. Azure Stream Analytics will ingest streaming data from applications or IoT devices and gateways into an event hub or an Internet of Things (IoT) hub in real-time. At which point the event or IoT hub will stream the data into Stream Analytics for real-time analysis. Azure Storage offers a massively scalable object store for data objects and file system services for the cloud. Creating a storage account as a Blob means that you cannot query the data directly.

Question 3

Which data platform technology could be used to help AdventureWorks scale globally?

- Azure Data Factory
- Azure Data Catalog
- Azure Cosmos DB

Explanation

Azure Cosmos DB is a globally distributed, multi-model database that can offer sub second query performance. Azure Data Factory (ADF) is a cloud integration service that orchestrates that movement of data between various data stores. Azure Data Catalog is a single, central place for all of an organization's users to contribute their knowledge and build a community and culture of data sources that are owned by an organization.

Module 2 Working with Data Storage

Module Introduction

Working with Data Storage

This module teaches you the variety of ways to store data in Azure. You will learn the basics of storage management in Azure, how to create a Storage Account, and how to choose the right model for the data that you want to store in the cloud. You will also understand how data lake storage can be created to support a wide variety of big data analytical solutions with minimal effort.

Learning Objectives

In this module you will:

- Choose a data storage approach in Azure
- Create an Azure Storage Account
- Explain Azure Data Lake Storage
- Upload data into Azure Data Lake Store

Choose a data storage approach in Azure

Choose a data storage approach in Azure

In exploring the topic of choosing a data storage approach in Azure, you will do this in the context of an example. Suppose you manage an online sales learning portal for your organization. The majority of your sales team are often in different geographical areas, so the online learning portal is an essential requirement. It's even more important as your organization continues to increase the skills and knowledge enhancement training for the sales staff.

Your training data includes high-quality video, detailed sales simulations, and large repositories for maintaining student data and progress. Currently, all the training content is stored in your on-premises storage. You have an aggressive plan to add new courses and would like to avoid the need to continuously increase the local storage capacity. You're looking for a storage solution that is secure, durable, scalable, and easily accessible from across the globe.

Azure provides storage features that will meet all of your business needs.

Learning objectives

In this module, you will learn:

- The Benefits of using Azure to store data
- Compare Azure data storage with on-premises storage

The Benefits of using Azure to Store Data

To address the storage problem for your online learning portal, you're considering storing your data in the cloud. But you're concerned about security, backup, and disaster recovery. On top of that, you're worried about how difficult it could be to manage cloud-hosted data. So, here's what you need to know.

The Azure data storage options are cloud-based, secure, and scalable. Its features address the key challenges of cloud storage and provide you with a reliable and durable storage solution.

Benefits of using Azure to store data

Here are some of the important benefits of Azure data storage:

- **Automated backup and recovery:** mitigates the risk of losing your data if there is any unforeseen failure or interruption.
- **Replication across the globe:** copies your data to protect it against any planned or unplanned events, such as scheduled maintenance or hardware failures. You can choose to replicate your data at multiple locations across the globe.
- **Support for data analytics:** supports performing analytics on your data consumption.
- **Encryption capabilities:** data is encrypted to make it highly secure; you also have tight control over who can access the data.
- **Multiple data types:** Azure can store almost any type of data you need. It can handle video files, text files, and even large binary files like virtual hard disks. It also has many options for your relational and NoSQL data.

- **Data storage in virtual disks:** Azure also has the capability of storing up to 8 TB of data in its virtual disks. This is a significant capability when you're storing heavy data such as videos and simulations.
- **Storage tiers:** storage tiers to prioritize access to data based on frequently used versus rarely used information.

Comparing Azure to On-Premises Storage

Now that you know about the benefits and features of Azure data storage, let's see how it differs from on-premises storage.

The term "on-premises" refers to the storage and maintenance of data on local hardware and servers. There are several factors to consider when comparing on-premises to Azure data storage.



Cost effectiveness

An on-premises storage solution requires dedicated hardware that needs to be purchased, installed, configured, and maintained. This can be a significant up-front expense (or capital cost). Change in requirements can require investment in new hardware. Your hardware needs to be capable of handling peak demand which means it may sit idle or be under-utilized in off-peak times.

Azure data storage provides a pay-as-you-go pricing model which is often appealing to businesses as an operating expense instead of an upfront capital cost. It's also scalable, allowing you to scale up or scale out as demand dictates and scale back when demand is low. You are charged for data services only as you need them.



Reliability

On-premises storage requires data backup, load balancing, and disaster recovery strategies. These can be challenging and expensive as they often each need dedicated servers requiring a significant investment in both hardware and IT resources.

Azure data storage provides data backup, load balancing, disaster recovery, and data replication as services to ensure data safety and high availability.



Storage types

Sometimes multiple different storage types are required for a solution, such as file and database storage. An on-premises approach often requires numerous servers and administrative tools for each storage type.

Azure data storage provides a variety of different storage options including distributed access and tiered storage. This makes it possible to integrate a combination of storage technologies providing the best storage choice for each part of your solution.



Agility

Requirements and technologies change. For an on-premises deployment this may mean provisioning and deploying new servers and infrastructure pieces, which is a time consuming and expensive activity.

Azure data storage gives you the flexibility to create new services in minutes. This flexibility allows you to change storage back-ends quickly without needing a significant hardware investment.

The following illustration shows differences between on-premises storage and Azure data storage.

| Needs | On-premise | Azure Data Storage |
|--|--|---|
|  Compliance and Security | 1 Dedicated servers required for privacy and security | Client side encryption and encryption at rest |
|  Store structured and unstructured data | 2 Additional IT resources with dedicated servers required | Azure Data Lake and portal analyzes and manages all types of data |
|  Replication and High Availability | 3 More resources, licensing, and servers required | Built-in replication and redundancy features available |
|  Application sharing and access to shared resources | 4 File sharing requires additional administration resources | File sharing options available without additional license |
|  Relational Data storage | 5 Needs a database server with database admin role | Offers Database-as-a-Service option |
|  Distributed storage and data access | 6 Expensive storage, networking, and compute resources needed | Azure Cosmos DB provides price-winning distributed access |
|  Messaging and load balancing | 7 Hardware redundancy impacts budget and resources | Azure Queue provides effective load balancing |
|  Tiered storage | 8 Management of tiered storage needs technology and labor skillset | Azure offers automated tiered storage of data |

Summary

In this section, you explored the benefits of using Azure to store your data. Azure provides the following features:

- Storage of both structured and unstructured data
- High security that supports global compliance standards
- Load balancing, high availability, and redundancy capabilities
- The ability to store large volumes of data directly to the browser using features such as Azure Blob storage

Ultimately, the capabilities of Azure data storage make it an ideal platform for hosting any large global application or portal.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Suppose you work at a startup with limited funding. Why might you prefer Azure data storage over an on-premises solution?

- To ensure you run on a specific brand of hardware which will let you to form a marketing partnership with that hardware vendor.
- The Azure pay-as-you-go billing model lets you avoid buying expensive hardware.
- To get exact control over the location of your data store.

Question 2

Which of the following situations would yield the most benefits from relocating an on-premises data store to Azure?

- Unpredictable storage demand that increases and decreases multiple times throughout the year.
- Long-term, steady growth in storage demand.
- Consistent, unchanging storage demand.

Introducing Azure Storage

Introducing Azure Storage

Most organizations have diverse requirements for their cloud-hosted data. For example, storing data in a specific region, or needing separate billing for different data categories. Azure storage accounts let you formalize these types of policies and apply them to your Azure data.

Suppose you work at a chocolate manufacturer that produces baking ingredients such as cocoa powder and chocolate chips. You market your products to grocery stores who then sell them to consumers.

Your formulations and manufacturing processes are trade secrets. The spreadsheets, documents, and instructional videos that capture this information are critical to your business and require geographically-redundant storage. This data is primarily accessed from your main factory, so you would like to store it in a nearby data center. The expense for this storage needs to be billed to the manufacturing department.

You also have a sales group that creates cookie recipes and baking videos to promote your products to consumers. Your priority for this data is low cost, rather than redundancy or location. This storage must be billed to the sales team.

Here, you will see how to handle these types of business requirements by creating multiple Azure storage accounts. Each storage account will have the appropriate settings for the data it holds.

Learning objectives

In this module, you will:

- Describe storage accounts
- Determine the appropriate settings for each storage account
- Choose an account creation tool
- Create a storage account using the Azure portal

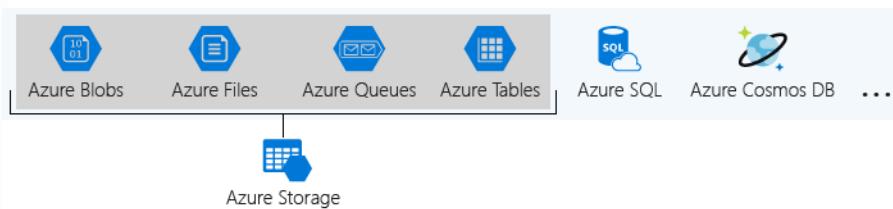
Storage accounts

Organizations often have multiple storage accounts to let them implement different sets of requirements. In the chocolate-manufacturer example, there would be one storage account for the private business data and one for the consumer-facing files. Here, you will learn the policy factors that are controlled by a storage account, which will help you decide how many accounts you need.

What is Azure Storage

Azure provides many ways to store your data. There are multiple database options like Azure SQL Server, Azure Cosmos DB, and Azure Table Storage. Azure offers multiple ways to store and send messages, such as Azure Queues and Event Hubs. You can even store loose files using services like Azure Files and Azure Blobs.

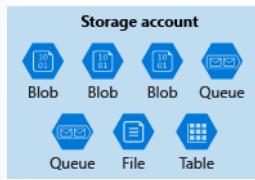
Azure selected four of these data services and placed them together under the name *Azure Storage*. The four services are Azure Blobs, Azure Files, Azure Queues, and Azure Tables. The following illustration shows the elements of Azure Storage.



These four were given special treatment because they are all primitive, cloud-based storage services and are often used together in the same application.

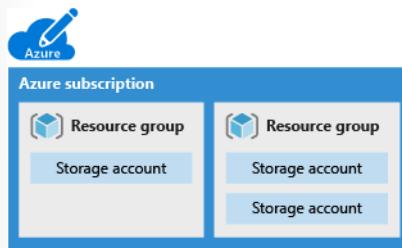
What is a storage account

A *storage account* is a container that groups a set of Azure Storage services together. Only data services from Azure Storage can be included in a storage account (Azure Blobs, Azure Files, Azure Queues, and Azure Tables). The following illustration shows a storage account containing several data services.

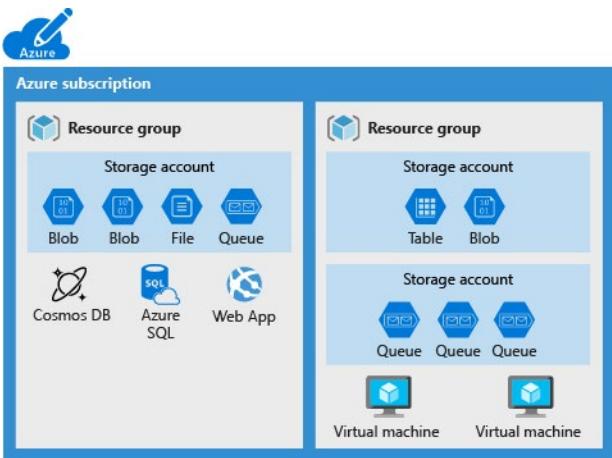


Combining data services into a storage account lets you manage them as a group. The settings you specify when you create the account, or any that you change after creation, are applied to everything in the account. Deleting the storage account deletes all of the data stored inside it.

A storage account is an Azure resource and is included in a resource group. The following illustration shows an Azure subscription containing multiple resource groups, where each group contains one or more storage accounts.



Other Azure data services like Azure SQL and Cosmos DB are managed as independent Azure resources and cannot be included in a storage account. The following illustration shows a typical arrangement: Blobs, Files, Queues, and Tables are inside storage accounts, while other services are not.



How many storage accounts do you need

A storage account represents a collection of settings like location, replication strategy, and subscription owner. You need one storage account for every group of settings that you want to apply to your data. The following illustration shows two storage accounts that differ in one setting; that one difference is enough to require separate storage accounts.

| Storage account | Storage account |
|---|--|
| Subscription: Production Location: West US Performance: Standard Replication: GRS Access tier: Hot Secure transfer: Enabled Virtual networks: Disabled | Subscription: Production Location: North Europe Performance: Standard Replication: GRS Access tier: Hot Secure transfer: Enabled Virtual networks: Disabled |

The number of storage accounts you need is typically determined by your data diversity, cost sensitivity, and tolerance for management overhead.

Data diversity

Organizations often generate data that differs in where it is consumed, how sensitive it is, which group pays the bills, etc. Diversity along any of these vectors can lead to multiple storage accounts. Let's consider two examples:

1. Do you have data that is specific to a country or region? If so, you might want to locate it in a data center in that country for performance or compliance reasons. You will need one storage account for each location.
2. Do you have some data that is proprietary and some for public consumption? If so, you could enable virtual networks for the proprietary data and not for the public data. This will also require separate storage accounts.

In general, increased diversity means an increased number of storage accounts.

Cost sensitivity

A storage account by itself has no financial cost; however, the settings you choose for the account do influence the cost of services in the account. Geo-redundant storage costs more than locally-redundant storage. Premium performance and the Hot access tier increase the cost of blobs.

You can use multiple storage accounts to reduce costs. For example, you could partition your data into critical and non-critical categories. You could place your critical data into a storage account with geo-redundant storage and put your non-critical data in a different storage account with locally-redundant storage.

Management overhead

Each storage account requires some time and attention from an administrator to create and maintain. It also increases complexity for anyone who adds data to your cloud storage; everyone in this role needs to understand the purpose of each storage account so they add new data to the correct account.

Storage accounts are a powerful tool to help you get the performance and security you need while minimizing costs. A typical strategy is to start with an analysis of your data and create partitions that share characteristics like location, billing, and replication strategy, and then create one storage account for each partition.

Storage Account Settings

A storage account defines a policy that applies to all the storage services in the account. For example, you could specify that all the contained services will be stored in the West US datacenter, accessible only over https, and billed to the sales department's subscription.

The settings that are controlled by a storage account are:

- **Subscription:** The Azure subscription that will be billed for the services in the account.
- **Location:** The datacenter that will store the services in the account.
- **Performance:** Determines the data services you can have in your storage account and the type of hardware disks used to store the data. **Standard** allows you to have any data service (Blob, File, Queue, Table) and uses magnetic disk drives. **Premium** limits you to one specific type of blob called a *page blob* and uses solid-state drives (SSD) for storage.
- **Replication:** Determines the strategy used to make copies of your data to protect against hardware failure or natural disaster. At a minimum, Azure will automatically maintain a copy of your data within the data center associated with the storage account. This is called locally-redundant storage (LRS), and guards against hardware failure but does not protect you from an event that incapacitates the entire datacenter. You can upgrade to one of the other options such as geo-redundant storage (GRS) to get replication at different datacenters across the world.
- **Access tier:** Controls how quickly you will be able to access the blobs in this storage account. Hot gives quicker access than Cool, but at increased cost. This applies only to blobs, and serves as the default value for new blobs.
- **Secure transfer required:** A security feature that determines the supported protocols for access. Enabled requires HTTPS, while disabled allows HTTP.
- **Virtual networks:** A security feature that allows inbound access requests only from the virtual network(s) you specify.

The storage account settings we've already covered apply to the data services in the account. Here, we will discuss the three settings that apply to the account itself, rather than to the data stored in the account:

- Name
- Deployment model
- Account kind

These settings impact how you manage your account and the cost of the services within it.

Name

Each storage account has a name. The name must be globally unique within Azure, use only lowercase letters and digits and be between 3 and 24 characters.

Deployment model

A *deployment model* is the system Azure uses to organize your resources. The model defines the API that you use to create, configure, and manage those resources. Azure provides two deployment models:

- **Resource Manager:** the current model that uses the Azure Resource Manager API
- **Classic:** a legacy offering that uses the Azure Service Management API

The decision on which one to choose is usually easy, because most Azure resources only work with Resource Manager. However, storage accounts, virtual machines, and virtual networks support both, so you must choose one or the other when you create your storage account.

The key feature difference between the two models is their support for grouping. The Resource Manager model adds the concept of a *resource group*, which is not available in the classic model. A resource group lets you deploy and manage a collection of resources as a single unit.

Microsoft recommends that you use **Resource Manager** for all new resources.

Account kind

Storage account *kind* is a set of policies that determine which data services you can include in the account and the pricing of those services. There are three kinds of storage accounts:

- **StorageV2 (general purpose v2):** the current offering that supports all storage types and all of the latest features
- **Storage (general purpose v1):** a legacy kind that supports all storage types but may not support all features
- **Blob storage:** a legacy kind that allows only block blobs and append blobs

Microsoft recommends that you use the **General-purpose v2** option for new storage accounts.

There are a few special cases that can be exceptions to this rule. For example, pricing for transactions is lower in general purpose v1, which would allow you to slightly reduce costs if that matches your typical workload.

The core advice here is to choose the **Resource Manager** deployment model and the **StorageV2 (general purpose v2)** account kind for all your storage accounts. The other options still exist primarily to allow existing resources to continue operation. For new resources, there are few reasons to consider the other choices.

Storage Account Creation Tool

There are several tools that create a storage account. Your choice is typically based on if you want a GUI and whether you need automation.

Available tools

The available tools are:

- Azure Portal
- Azure CLI (Command-line interface)
- Azure PowerShell
- Management client libraries

The portal provides a GUI with explanations for each setting. This makes the portal easy to use and helpful for learning about the options.

The other tools in the above list all support automation. The Azure CLI and Azure PowerShell let you write scripts, while the management libraries allow you to incorporate the creation into a client app.

How to choose a tool

Storage accounts are typically based on an analysis of your data, so they tend to be relatively stable. As a result, storage-account creation is usually a one-time operation done at the start of a project. For one-time activities, the portal is the most common choice.

In the rare cases where you need automation, the decision is between a programmatic API or a scripting solution. Scripts are typically faster to create and less work to maintain because there is no need for an IDE, NuGet packages, or build steps. If you have an existing client application, the management libraries might be an attractive choice; otherwise, scripts will likely be a better option.

Use the Azure portal to create a storage account

1. Sign into the [Azure portal¹](#) using the same account you activated the sandbox with.
2. In the top left of the Azure portal, select **Create a resource**.
3. In the selection panel that appears, select **Storage**.
4. On the right side of that pane, select **Storage account**.

¹ <https://portal.azure.com/learn.docs.microsoft.com?azure-portal=true>

The screenshot shows the Azure Marketplace interface with a search bar at the top. Below it, there are two tabs: 'Azure Marketplace' and 'Featured'. Under 'Azure Marketplace', there are several categories: 'Get started', 'Recently created', 'Compute', 'Networking', 'Storage' (which is highlighted with a blue box), 'Web', 'Mobile', 'Containers', and 'Databases'. To the right of these categories, there are corresponding icons and links. The 'Storage' section includes a 'Quickstart tutorial' link. The 'Featured' section includes 'Azure File Sync' with a 'Quickstart tutorial' link, 'Data Box Edge / Data Box Gateway (preview)' with a 'Learn more' link, and 'Data Lake Storage Gen1' with a 'Quickstart tutorial' link.

Configure the basic options

1. Select the appropriate **Subscription**.
2. Select the existing Resource Group from the drop-down list.

Under **INSTANCE DETAILS**:

1. Enter a **Storage account name**. The name will be used to generate the public URL used to access the data in the account. The name must be unique across all existing storage account names in Azure. Names must be 3 to 24 characters long and can contain only lowercase letters and numbers.
2. Select a **Location** near to you from the list above.
3. Select *Standard* for the **Performance** option. This decides the type of disk storage used to hold the data in the Storage account. Standard uses traditional hard disks, and Premium uses solid-state drives (SSD) for faster access. However, remember that Premium only supports *page blobs*. You'll need *block blobs* for your videos, and a queue for buffering - both of which are only available with the *Standard* option.
4. Select *StorageV2 (general purpose v2)* for the **Account kind**. This provides access to the latest features and pricing. In particular, Blob storage accounts have more options available with this account type. You need a mix of blobs and a queue, so the *Blob storage* option will not work. For this application, there would be no benefit to choosing a *Storage (general purpose v1)* account, since that would limit the features you could access and would be unlikely to reduce the cost of your expected workload.

5. Select *Locally-redundant storage (LRS)* for the **Replication** option. Data in Azure storage accounts are always replicated to ensure high availability - this option lets you choose how far away the replication occurs to match your durability requirements. In our case, the images and videos quickly become out-of-date and are removed from the site. As a result, there is little value to paying extra for global redundancy. If a catastrophic event results in data loss, you can restart the site with fresh content from your users.
6. Set the **Access tier** to *Hot*. This setting is only used for Blob storage. The **Hot Access Tier** is ideal for frequently accessed data, and the **Cool Access Tier** is better for infrequently accessed data. Note that this only sets the *default* value - when you create a Blob, you can set a different value for the data. In our case, we want the videos to load quickly, so you'll use the high-performance option for your blobs.

The following screenshot shows the completed settings for the **Basics** tab. Note that the resource group, subscription, and name will have different values.

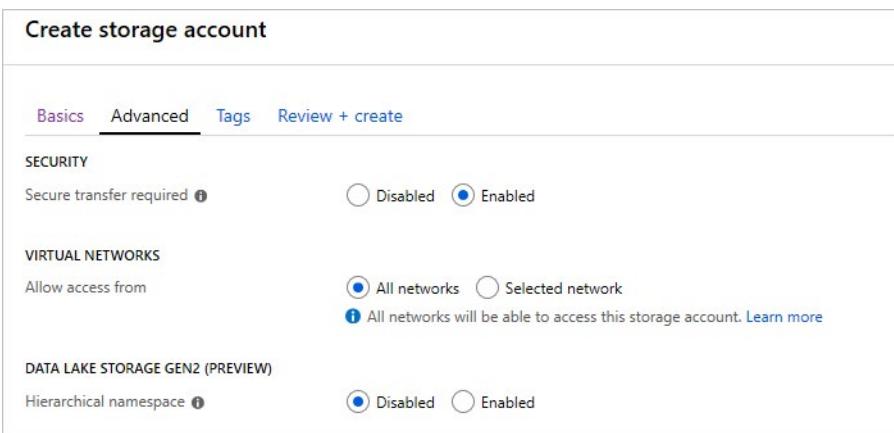
The screenshot shows the 'Basics' tab of the Azure Storage creation wizard. It includes sections for 'PROJECT DETAILS' and 'INSTANCE DETAILS'. In 'PROJECT DETAILS', the 'Subscription' is set to 'Concierge Subscription' and the 'Resource group' is set to 'ffa59036-df0b-491b-8798-23d04dd906fb'. In 'INSTANCE DETAILS', the 'Storage account name' is 'abstoacc', 'Location' is 'West US 2', 'Performance' is 'Standard', 'Account kind' is 'StorageV2 (general purpose v2)', 'Replication' is 'Locally-redundant storage (LRS)', and 'Access tier (default)' is 'Hot'.

Configure the advanced options

1. Click the **Next: Advanced >** button to move to the **Advanced** tab, or select the **Advanced** tab at the top of the screen.
2. Set **Secure transfer required** to *Enabled*. The **Secure transfer required** setting controls whether **HTTP** can be used for the REST APIs used to access data in the Storage account. Setting this option to *Enabled* will force all clients to use SSL (**HTTPS**). Most of the time you'll want to set this to *Enabled* as using HTTPS over the network is considered a best practice.

3. Set the **Virtual networks** option to *All networks*. This option allows you to isolate the storage account on an Azure virtual network. We want to use public Internet access. Our content is public facing and you need to allow access from public clients.
4. Leave the **Data Lake Storage Gen2** option as *Disabled*. This is for big-data applications that aren't relevant to this module.

The following screenshot shows the completed settings for the **Advanced** tab.



Create a storage account

1. You can explore the **Tags** settings if you like. This lets you associate key/value pairs to the account for your categorization and is a feature available to any Azure resource.
2. Click **Review + create** to review the settings. This will do a quick validation of your options to make sure all the required fields are selected. If there are issues, they'll be reported here. Once you've reviewed the settings, click **Create** to provision the storage account.

It will take a few minutes to deploy the account. While Azure is working on that, let's explore the APIs we'll use with this account.

Verify a storage account is created

1. Select the **Storage accounts** link in the left sidebar.
2. Locate the new storage account in the list to verify that creation succeeded.

NOTE

When you're working in your own subscription, you can use the following steps in the Azure portal to delete the resource group and all associated resources.

1. Select the **Resource groups** link in the left sidebar.
2. Locate the resource group you created in the list.
3. Right-click on the resource group entry and select **Delete resource group** from the context menu. You can also click the "..." menu element on the right side of the entry to get to the same context menu.
4. Type the resource group name into the confirmation field.

5. Click the **Delete** button. This may take several minutes.

You created a storage account with settings driven by your business requirements. For example, you might have selected a West US datacenter because your customers were primarily located in southern California. This is a typical flow: first analyze your data and goals, and then configure the storage account options to match.

Summary

In this section you have learned that there are different types of storage accounts including blobs, files, tables, and queues. You have learned the differences about each type and the configuration settings that needs to be defined when creating a storage account. Finally, you have also learned that the creation of the storage accounts can be defined by a range of tools.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Suppose you have two video files stored as blobs. One of the videos is business-critical and requires a replication policy that creates multiple copies across geographically diverse datacenters. The other video is non-critical, and a local replication policy is sufficient. True or false: To satisfy these constraints, the two blobs will need to be in separate storage accounts.

- True
- False

Question 2

The name of a storage account must be:

- Unique within the containing resource group.
- Unique within your Azure subscription.
- Globally unique.

Question 3

In a typical project, when would you create your storage account(s)?

- At the beginning, during project setup.
- After deployment, when the project is running.
- At the end, during resource cleanup.

Introduction to Data Lake Storage

Introduction to Data Lake Storage

Many organizations have spent the last two decades building data warehouses and business intelligence (BI) solutions based on relational database systems. Many BI solutions have lost out on opportunities to store unstructured data due to cost and complexity in these types of data and databases in the past.

Suppose that you're a data engineering consultant doing work for Contoso. They're interested in using Azure to analyze all of their business data. In this role, you'll need to provide guidance on how Azure can enhance their existing business intelligence systems. You'll also offer advice about leveraging Azure's storage capabilities to store large amounts of unstructured data to add value to their BI solution. Because of their needs, you plan to recommend **Azure Data Lake Storage**. Azure Data Lake Storage provides a repository where we can upload and store huge amounts of unstructured data with an eye toward high-performance Big Data Analytics.

Learning Objectives

In this module you will:

- Explain Azure Data Lake Storage
- Create an Azure Data Lake Store Gen 2 using the portal
- Compare Azure Blob Storage and Data Lake Store Gen 2
- Explore the stages for processing Big Data Using Azure Data Lake Store
- Describe the use cases for Data lake Storage

Data Lake Storage Generation 2 (Gen2)

A **data lake** is a repository of data stored in its natural format, usually as blobs or files. **Azure Data Lake Storage** is a comprehensive, highly scalable, and cost-effective data lake solution for big data analytics built in Azure.

Azure Data Lake Storage combines the power of a high-performance file system, with a massive storage platform to help you quickly identify insights in your data. Azure Data Lake Storage Gen2 builds on Azure Blob Storage capabilities to optimize it specifically for analytics workloads. As it's integrated with Azure Blob Storage, it enables best in class analytics performance, along with Blob Storage's tiering and data lifecycle management capabilities and the high-availability, security, and durability capabilities of Azure Storage.

The variety and volume of data generated and analyzed today is increasing. Companies have multiple sources of data - from websites to Point of Sale (POS) systems, and more recently social media sites to Internet of Things (IoT) devices. Each source provides an essential aspect of data that needs to be collected, analyzed, and potentially acted upon.

Benefits

Azure Data Lake Storage Gen2 is designed to deal with this variety and volume of data at Exabyte scale while handling hundreds of gigabytes of throughput that is secure by design. Thus, giving you the flexibility to use Data Lake Storage Gen2 as the basis for both real-time and batch solutions. Here is a list of additional benefits that Azure Data Lake Storage Gen 2 brings:

Hadoop compatible access

A key benefit of Data Lake Storage Gen2 is that you can treat the data as if it's stored in a Hadoop Distributed File System. This feature enables you to store the data in one place and access it through a wide range of compute technologies including Azure Databricks, HDInsight, and SQL Data Warehouse without moving the data between environments, enabling efficient usage while minimizing the cost.

Security

Data Lake Storage Gen2 supports Access Control Lists (ACL) and POSIX permissions. You can set a granular level of permissions at a directory or file level for the data stored within the Data Lake. This security is configurable through a variety of technologies including Hive and Spark, or utilities such as Azure Storage Explorer. Furthermore, all data stored is encrypted at rest using either Microsoft or customer-managed keys.

Performance

Azure Data Lake Storage organizes the stored data into a hierarchy of directories and subdirectories much like a file system allowing for easier navigation. As a result, data processing requires less computational resources, which in turn reduces both the time and cost.

Data redundancy

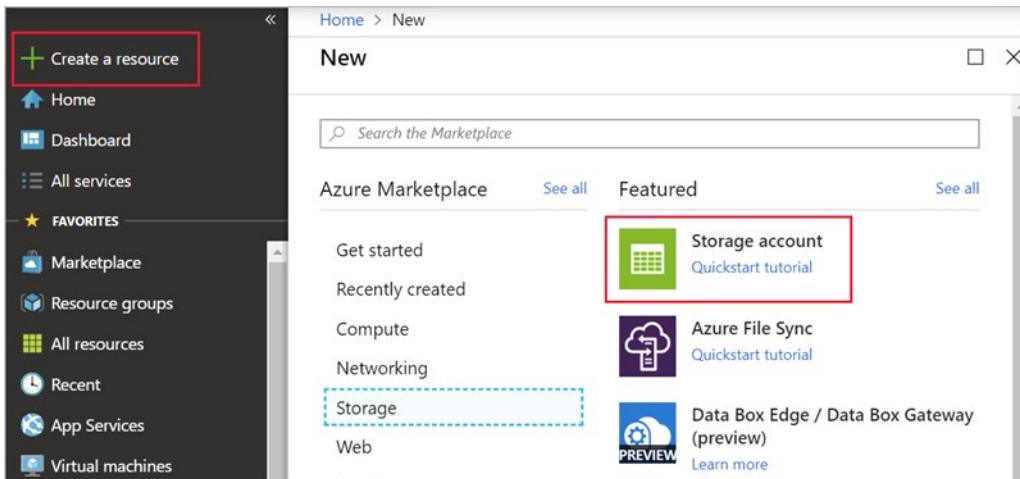
Azure Data Lake Storage Gen2 takes advantage of the Azure Blob replication models that provide data redundancy in a single data center with Locally Redundant Storage (LRS), or to a secondary region using the Geo-redundant storage option. This feature ensures your data is always available and protected if catastrophe strikes.

Create a Data Lake Storage Account (Gen2)

Azure Data Lake Storage Gen2 is easy to set up. It requires a **StorageV2 (General Purpose V2)** Azure Storage account with the Hierarchical namespace enabled. Let's walk through an example of setting up an Azure Data Lake in the Azure portal.

1. Sign in to the [Azure portal](#)²
2. Click on **Create a resource** and navigate to **Storage**.
3. In **Featured**, select **Storage account - blob, file, table, queue** as shown in the following screenshot.

² <https://portal.azure.com?azure-portal=true>



4. Next, in the **Create storage account** window, under the **Basics** tab, define a **storage account name** and **location**.
5. In the **Account kind** drop-down list, click to select **StorageV2 (general-purpose v2)** to create a Data Lake Storage Gen2 data store.

Create storage account

Basics [Advanced](#) [Tags](#) [Review + create](#)

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [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: Visual Studio Enterprise

* Resource group: Select existing... or Create new

INSTANCE DETAILS

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

* Storage account name: (input field)

* Location: East US

Performance: Standard (radio button selected) Premium (radio button)

Account kind: StorageV2 (general purpose v2) (dropdown menu, highlighted with a red box)

Replication: Read-access geo-redundant storage (RA-GRS) (dropdown menu)

Access tier (default): Cool (radio button) Hot (radio button selected)

6. Select the **Advanced** tab, there is an option of **Data lake storage gen2(preview)**, next to the option of **Hierarchical namespace**, ensure it is set to **Enabled** as shown below.

The screenshot shows the Azure portal's 'Create storage account' wizard. The 'Advanced' tab is selected. In the 'DATA LAKE STORAGE GEN2 (PREVIEW)' section, the 'Hierarchical namespace' setting is highlighted with a red box around the 'Enabled' radio button. Other settings shown include 'Secure transfer required' (Enabled), 'Allow access from' (All networks selected), and 'Virtual Networks' (All networks will be able to access this storage account).

This new Azure Storage account is now set up to host data for an Azure Data Lake. Once the account has deployed, you will find options related to Azure Data Lake in the Overview page.

Comparing Azure Blob Storage to Azure Data Lake Storage

Azure Blob Storage enables you to store large amounts of unstructured ("object") data in a single hierarchy – AKA a flat namespace – and can be accessed using HTTP or HTTPS. Azure Data Lake Storage Gen2 builds on blob storage, optimizing I/O of high-volume data using hierarchical namespaces we turned on in the previous exercise.

Hierarchical namespaces organize blob data into *directories* and store metadata about each directory and the files within it. This structure allows operations like directory renames and deletes to be done in a single atomic operation. Flat namespaces, by contrast, require several operations proportionate to the number of objects in the structure. As hierarchical namespaces keep the data organized, it yields better storage and retrieval performance for an analytical use case, which lowers the cost of analysis.

Azure Blob Storage vs. Azure Data Lake Storage

If your use case is to store data *without performing analysis on the data*, then set up the storage account as an Azure Blob Storage account by setting the **Hierarchical Namespace** option to **Disabled**. A great use case for blob storage is archiving rarely used data or storing website assets such as images and media.

If you are performing analytics on the data, then you should set the storage for Azure Data Lake Storage Gen2 by setting the **Hierarchical Namespace** to **Enabled**. Since Azure Data Lake Storage Gen2 is integrated into the Azure Storage platform, applications can use either the Blob APIs or Azure Data Lake Storage Gen2 file system APIs for accessing data.

Stages for Processing Big Data

Azure Data Lake Storage Gen2 plays a fundamental role in a wide range of big data architectures. These architectures can involve the creation of a:

- Modern data warehouse
- Advanced analytics against big data
- A real-time analytical solution

Regardless of the architecture created, there are four stages for processing big data solutions that are common to all.

1. **Ingestion** - The ingestion phase identifies the technology and processes used to acquire the source data. This data can come from files, logs, and other types of unstructured data that must be put into the Data Lake Store. The technology used will vary depending on the frequency the data is transferred. For example: for batch movement of data, Azure Data Factory may be the most appropriate technology to use. For real-time ingestion of data, Apache Kafka for HDInsight or Stream Analytics may be an appropriate technology to use.
2. **Store** - The store phase identifies where the ingested data should be placed. In this case, we're using Azure Data Lake Storage Gen 2.
3. **Prep & Train** - The Prep and train step identifies the technologies that are used to perform data preparation and model training and scoring for data science solutions. The common technologies used in this phase are Azure Databricks or Machine Learning Services.
4. **Model & Serve** - Finally, the model and serve step involves the technologies that will present the data to users. These can include visualization tools such as Power BI, or other data stores such as SQL Data Warehouse, Cosmos DB, Azure SQL, or Azure Analysis Services. Often, a combination of these technologies will be used depending on the business requirements.

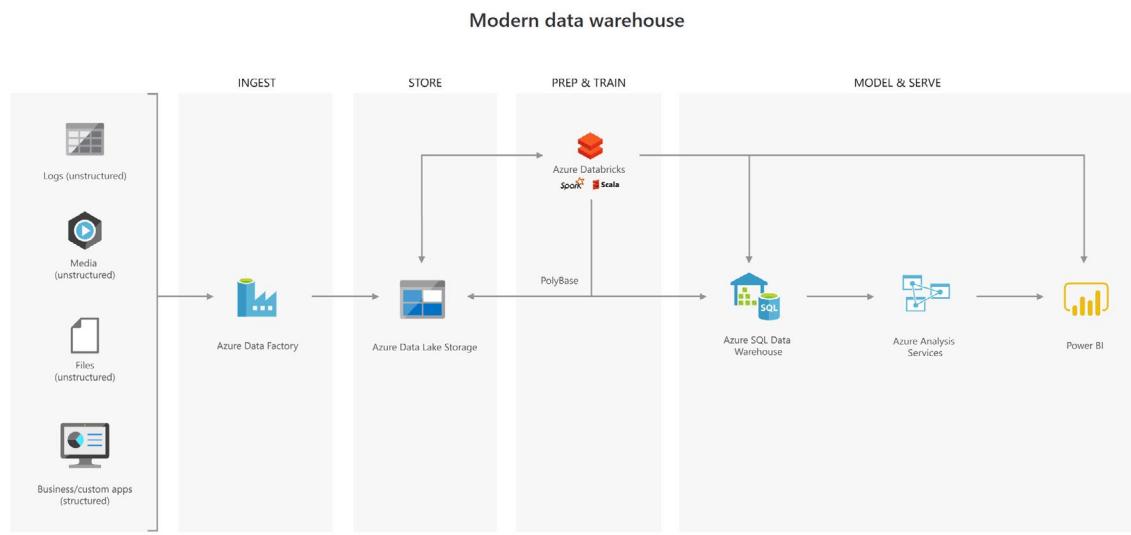
Big Data Use Cases

Let's examine three use cases for leveraging an Azure Data Lake Store.

Creating a Modern Data Warehouse

Imagine you are a Data Engineering consultant for Contoso. In the past, they've created an on-premises business intelligence solution that used a Microsoft SQL Server Database Engine, Integration Services, Analysis Services, and Reporting Services to provide historical reports. An attempt was made to use the Analysis Services Data Mining component to create a predictive analytics solution to predict the buying behavior of customers. While this approach worked well with low volumes of data, it couldn't scale once more than a gigabyte of data was collected. Furthermore, they were never able to deal with the JSON data that a third-party application generated when a customer used the feedback module of the POS application.

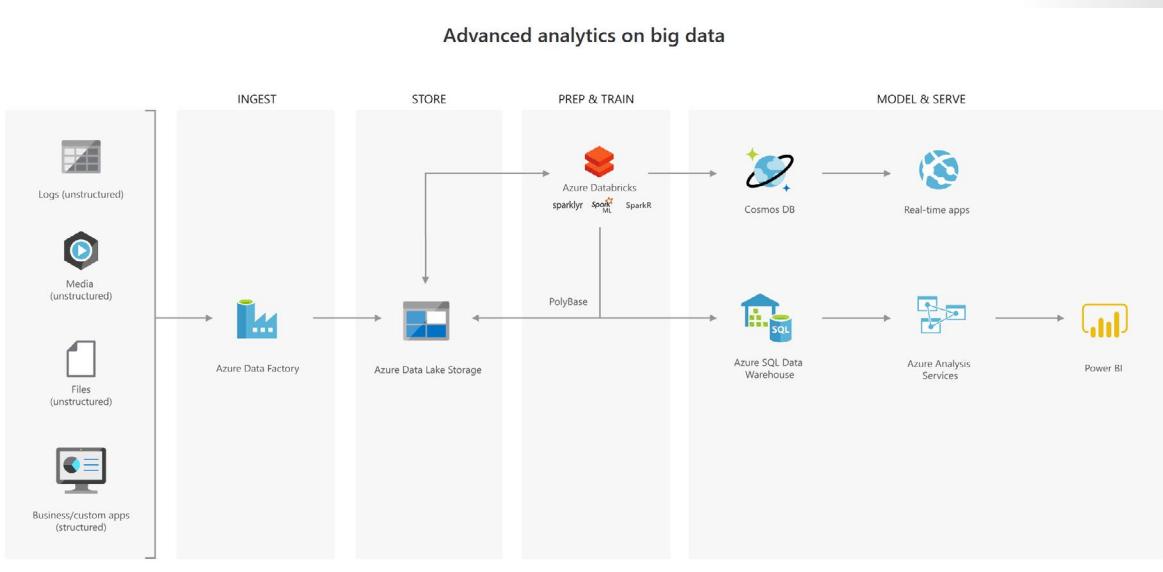
Contoso has turned to you for help with creating an architecture that can scale with the data needs that is required to create a predictive model and to handle the JSON data so that it's integrated into the BI solution. You suggest the following architecture:



The following architecture sees Azure Data Lake Storage at the heart of the solution for a modern data warehouse. Integration Services is replaced by Azure Data Factory to ingest data into the Data Lake from a business application. This serves as the source of truth for the predictive model built in Azure Databricks, and PolyBase is used to transfer the historical data into a big data relational format held in using Azure Synapse Analytics. This in turn also stores the results of the trained model from Databricks.

Advanced analytics for Big Data

In this second use case, Azure Data Lake Storage again plays an important role in providing a large-scale data store. In this case, your skills are needed by AdventureWorks, a global seller of bicycles and cycling components through a chain of resellers and on the Internet. They have a requirement to ensure the results of a recommendation engine built in Azure Databricks can scale globally to meet the needs of recommending products to customers on their websites as they browse the product catalog and add items to their basket. The basis of performing the recommendation is on the web log files that are stored on the web servers and transferred to the Azure Databricks model on an hourly basis. The response time for the recommendation should be less than 1 ms. You propose the following architecture:



In this solution, Azure Data factory is transferring terabytes of web logs from a web server to the Data Lake on an hourly basis. This data is provided as features to the predictive model in Azure Databricks, which is then trained and scored. The result of the model is then distributed globally using Azure Cosmos DB, that the real-time app – in this case the AdventureWork website – will use to provide recommendations to the customers as they add products to their online basket.

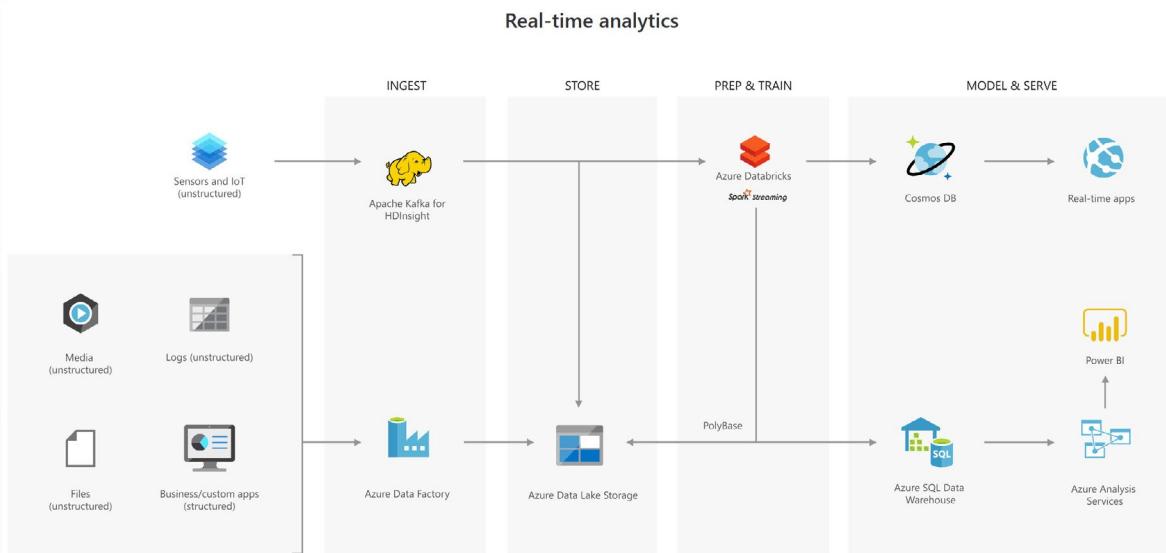
To complete this architecture, PolyBase is used against the Data Lake to transfer descriptive data to the Azure Synapse Analytics for reporting purposes. This data is presented to Analysis Services to provide the caching capability for Azure Synapse Analytics to service many users and display the data through reports using Power BI.

Real-time analytical solutions

To perform real-time analytical solutions, the major change that occurs to the architecture is to the ingestion phase for processing big data solutions. In this example architecture, you will note the introduction of Apache Kafka for HDInsight to ingest streaming data from an Internet of Things (IoT) device, although this could be replaced with IoT Hubs and Stream Analytics. The key point is that the data is persisted in Data Lake Storage to service other parts of the solution.

In this use case, you are a Data Engineer for Trey Research, an organization that is working with a transport company to monitor the fleet of Heavy Goods Vehicles (HGV) that drive around Europe. Each HGV is equipped with sensor hardware that will continuously report metric data on the temperature, the speed, and the oil and brake solution levels of an HGV. When the engine is turned off, the sensor also outputs a file with summary information about a trip, including the mileage and elevation of a trip. A trip is a period in which the HGV engine is turned on and off.

Both the real-time data and batch data is processed in a machine learning model to predict maintenance schedule for each of the HGVs, and this data is made available to the downstream application that third-party Garage companies can use should an HGV breakdown anywhere in Europe. In addition, historical reports about the HGV should be visually presented to users. As a result, the following architecture is proposed:



In this architecture, there are two ingestion streams. Azure Data Factory is used to ingest the summary files that are generated when the HGV engine is turned off. Apache Kafka provides the real-time ingestion engine for the telemetry data. Both data streams are stored in Data Lake store for use in the future, but they are also passed onto other technologies to meet the business needs. Both streaming and batch data are provided to the predictive model in Azure Databricks, of which the results are published to Azure Cosmos DB to be used by the third-party garages. PolyBase transfers data from the Data Lake Store into SQL Data Warehouse for the HGV reports created through Power BI through Azure Analysis Services.

Summary

Azure Data Lake Storage Gen2 provides a cloud storage service that is highly available, secure, durable, scalable, and redundant. It's the most comprehensive data lake solution available in market today. Azure Data Lake Storage brings new efficiencies to process big data analytics workloads and can provide data to a multitude of compute technologies including HDInsight and Azure Databricks without needing to move the data around. Creating an Azure Data Lake Storage Gen2 data store should be one of your go-to tools in building a big data analytics solution.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Mike is creating an Azure Data Lake Storage Gen 2 account. He must configure this account to be able to processes analytical data workloads for best performance. Which option should he configure when creating the storage account?

- On the Basic Tab, set the performance option to standard
- On the Basic Tab, set the performance to ON
- On the Advanced tab, set the Hierarchical Namespace to enabled

Question 2

In which phase of big data processing is Azure Data Lake Storage located?

- Ingestion
- Store
- Model & Serve

Create an Azure Data Lake Storage Gen2

Upload Data into Azure Data Lake Storage Gen2

Azure Data Lake Storage Generation 2 (Gen2) is a data lake solution explicitly designed for enterprises to run large scale analytical workloads in the cloud. It takes the core capabilities from Azure Data Lake Storage Gen1 including file system semantics and security and scale and combines it with the low-cost, highly available capabilities of Azure Blob Storage.

You want to show Contoso the different ways in which you can upload data into Azure Data Lake Storage Gen2. This will include examples of how you can perform ad-hoc data loads to integrating the upload capability into applications. You will also demonstrate how you can use Azure Data Factory to copy data into Azure Data Lake Gen 2.

Learning Objectives

In this module you will:

- Create an Azure Data Lake Gen2 Store using Powershell
- Upload data into the Data Lake Storage Gen2 using Azure Storage Explorer
- Copy data from an Azure Data Lake Store Gen1 to an Azure Data Lake Store Gen2

Create an Azure Data Lake Storage Gen2 Account

Before uploading or transferring data into a data lake, you need to create one. Using the Azure portal, you are able to provision an Azure Data Lake Storage Gen2 within minutes.

NOTE

If you don't have an Azure account, or prefer not to do the exercise in your account, you can read through the following instructions to understand the steps involved in creating an Azure Data Lake Store.

Create a new Resource Group

Open up the PowerShell console within the Azure Portal and type out the following:

```
$resourceGroup = "mslearn-datalake-test"  
$location = "westus2"  
New-AzResourceGroup -Name $resourceGroup -Location $location
```

Create a Data Lake Storage Account Gen2

In the PowerShell console within the Azure Portal and type out the following:

```
$location = "westus2"  
  
New-AzStorageAccount -ResourceGroupName $resourceGroup `  
-Name "dlakedata001" `  
-Location $location `  
-SkuName Standard_LRS `
```

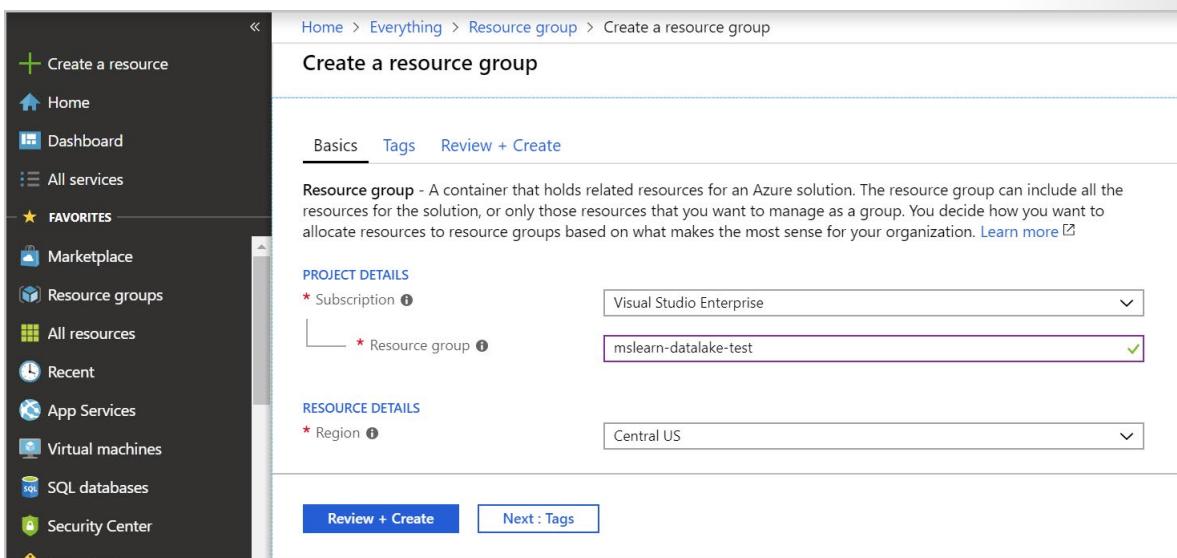
```
-Kind StorageV2
-EnableHierarchicalNamespace $True
```

If these steps fail, use the following steps in the Azure Portal:

Create a new Resource Group

First, create a new resource group to hold the data lake storage. A resource group will let you administer related services and applications together. It also makes it easier to clean up resources when you are done with this module. To create a resource group in the Azure portal, follow these steps:

1. Sign in to the [Azure portal](#)³ using your account.
2. Select **Create a resource** from the left sidebar.
3. In the search box, type “**Resource**” and select **Resource group** from the results.
4. Click the **Create** button to add a new resource group.
5. In the **Basics** tab, select the appropriate subscription you want to work in.
6. Set the name of the resource group to “**mslearn-datalake-test**” without the quotes.
7. Choose the region (location) for the resource group - you typically want to choose a location close to you or to the data you are going to work with.



8. Click the **Review + Create** button and then **Create** on the review screen.

Resource group creation is fast, you can pin the resource group to your dashboard to make it easy to find later if you like.

Create a Data Lake Storage Account Gen2

Creating an Azure Data Lake Storage Account Gen2 is the same as creating an Azure Blob Store, there's just one setting that is different. To create the data lake, perform the following steps:

1. In the Azure portal, choose **Create a resource** from the left sidebar.

³ <https://portal.azure.com?azure-portal=true>

2. Select **Storage**, and choose **Storage account**.
3. Select your **Subscription** and the **Resource group** you created earlier (**mslearn-datalake-test**).
4. Enter a unique name for your storage account. It must be unique across all of Azure, so for example use the prefix "dlakedata" with some numbers. You might have to try a few variations to find a unique name. The portal will display a green checkmark next to the name when you have a valid entry.
5. Select a location - you typically want to select a region near where the data consumption will occur. Since this is an example, just select a location near you.
6. Make sure the Account kind is **StorageV2 (general-purpose V2)**. The rest of the values can be left as their defaults.

Create storage account

PROJECT DETAILS

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

* Subscription: Visual Studio Enterprise
* Resource group: mslearn-datalake-test (selected)

INSTANCE DETAILS

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

* Storage account name: dlakedata123 (selected)

* Location: West US 2

Performance: Standard (selected)

Account kind: StorageV2 (general purpose v2) (selected)

Replication: Read-access geo-redundant storage (RA-GRS) (selected)

Access tier (default): Hot (selected)

Review + create | Previous | Next : Advanced >

7. Select **Next: Advanced >**
8. In the **Data Lake Storage Gen2 (preview)** section set **Hierarchical namespace** to **Enabled**.
9. Click **Review + Create** to create the storage account.

The screenshot shows the 'Create storage account' wizard in progress. The 'Basics' tab is selected. Under 'SECURITY', 'Secure transfer required' is set to 'Enabled'. Under 'VIRTUAL NETWORKS', 'Allow access from' is set to 'All networks'. Under 'DATA LAKE STORAGE GEN2', 'Hierarchical namespace' is set to 'Enabled'. At the bottom, there are buttons for 'Review + create', 'Previous', and 'Next : Tags >'.

10. Once the creation details have been validated, click the **Create** button to start deployment.

Wait for a few moments for the deployment to complete, once you receive the message "Your deployment is complete", click **Go to resource** to confirm the deployment.

Upload Data using Azure Storage Explorer

If you need to perform ad-hoc data transfers into an Azure Data Lake Store, you can use the **Azure Storage Explorer** to upload your files.

Azure Storage Explorer is a free application available for Windows, macOS, and Linux. The app is designed to manage unstructured data in Azure such as tables, blobs, queues, and files. It also supports data in Azure Cosmos DB and Azure Data Lake Storage, which is what we'll use it for here.

NOTE

If you don't have an Azure account, or prefer not to do the exercise in your account, you can read through the instructions to understand the steps involved to install and use the Azure Storage Explorer tool.

Download and Install Azure Storage Explorer

Start by installing the **Azure Storage Explorer**⁴.

Using Azure Storage Explorer

Once installed, you can use Azure Storage Explorer to perform several operations against data in your Azure Storage account including your data lake. Here are some features of the tool.

- You can upload files or folders from your local computer into Azure Storage.
- You can download cloud-based data to your local computer.
- You can copy or move files and folders around in the storage account.

⁴ <https://azure.microsoft.com/features/storage-explorer>

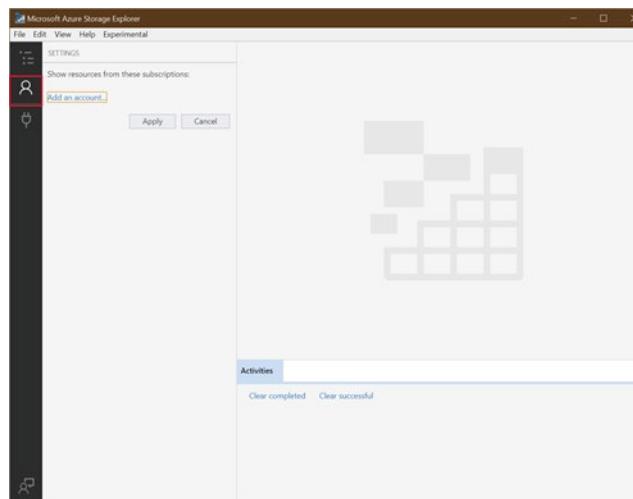
- You can delete data from the storage account.

Let's look at some of these capabilities.

Connect the Azure Storage Explorer to your Azure account

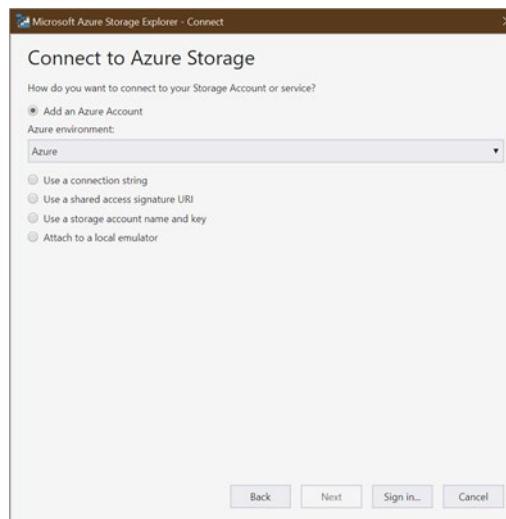
Start by adding your Azure account.

1. Click on the Account button icon in the left sidebar.



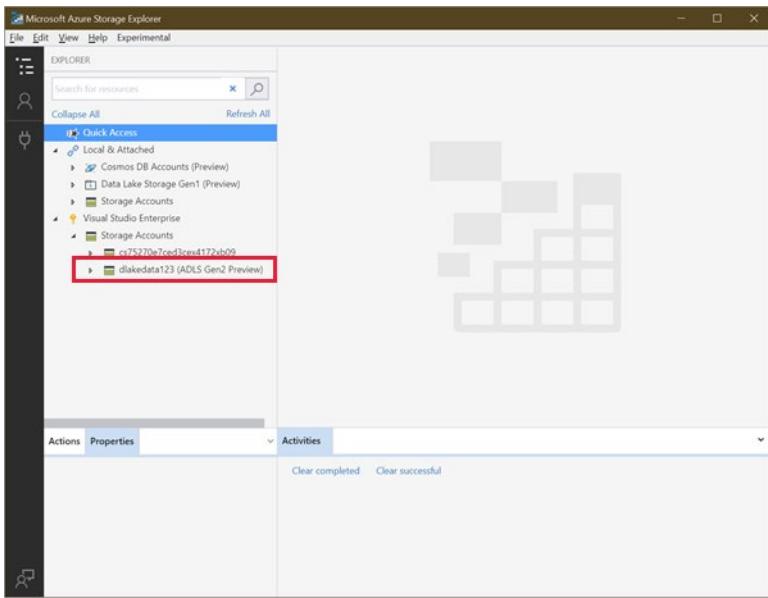
2. There are multiple options for connecting to your storage account.

- Sign in with your Azure account to provide access to all your subscriptions.
- Use a connection string to access a specific Azure Storage account.
- Use a storage account name and access key.



3. Once you sign in, you can select the subscriptions you want to work with. Make sure to select the one you created the Azure Storage account in.

The app then shows a tree of storage areas you can work with from your subscriptions. You should see your Azure Storage account in the list.



Create a filesystem using Azure Storage explorer

Blobs are always uploaded into folders. This allows you to organize groups of blobs much like you organize files on your computer.

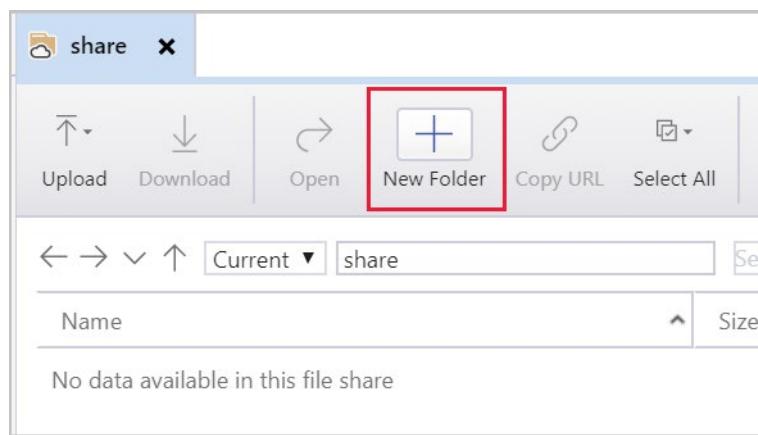
When working with Azure Data Lake, you start by creating a *filesystem*. This defines the specific container in Blob storage that will hold your data lake. You can then create folders and files within this dedicated area.

1. In Azure Storage Explorer, expand your subscription, and then expand storage accounts.
2. Expand the storage account that you have created in the previous unit, and click on **Blob Containers**.
3. Right-click **Blob Containers** and click **Create Blob Container**.
4. In the text box that appears below **Blob Containers**, type **salesdata**.
5. Once created, click on **salesdata**.

Create a folder in Storage Container using Azure Storage Explorer

Adding a folder provides a hierarchical structure for managing your data. You can create multiple levels in the account. However, you must ensure that parent folders exist before you create children.

1. Select the **New Folder** button from the menu running across the top.



2. For the folder name, enter “**sample**” without the quotes, and then select **OK** to create the directory. You may get a message box in Azure Storage Explorer that states. “Your view may be out of date. Do you want to refresh?”. If so, click **Yes**.
3. Double-click on the new folder in the UI - this will traverse into the folder, which should be empty.
4. Create another folder named “**data**”.

Create a sample text file

To provide some sample data to work with, create a local text file on your computer named “sales.txt” and paste the following text into the file.

```
#salaries Details
#Company Information
#Fields : Date company employee Salaries
01-01-2019  c1    e1 1000
01-01-2019  c2    e2 2000
01-01-2019  c1    e3 4000
01-01-2019  c2    e4 2000
01-01-2019  c1    e5 5000
01-01-2019  c3    e6 7000
```

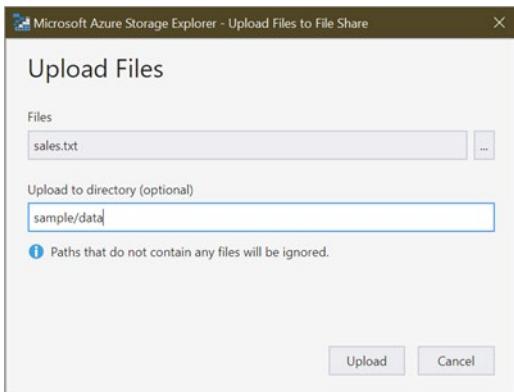
We'll upload this data file in various ways. Keep in mind that this is a *simple* example - you would typically be populating your data lake with much larger data samples from a variety of sources.

Upload a file

You can upload files and folders from your local machine to directories in your file share right from the tool.

1. In Azure Storage Explorer, double-click the folder named **data**.
2. In the top menu, select **Upload**. This gives you the option to upload a folder or a file.
3. Select **Upload Files**.
4. Select the “sales.txt” file you created earlier as the file to upload

5. In **Upload to a directory** dialog box, ensure the Destination directory states “**sample/data**”, and then select **Upload**.



When you are finished, the file appears in the list.

Download a file

To download a copy of a file from your file share, right-click the file, and then select **Download**. Choose where you want to put the file on your local machine, and then select **Save**. The progress of the download appears in the **Activities** pane at the bottom of the window.

Copy Data from Data Lake Storage Gen1 to Data Lake Storage Gen2

Azure Data Factory is a cloud-based data integration service that creates workflows in the cloud for orchestrating batch data movement and transformations. Using Azure Data Factory, you can create and schedule workflows (called *pipelines*) to ingest data from disparate data stores. The data can then be processed and transformed with services such as:

- Azure HDInsight Hadoop
- Spark
- Azure Data Lake
- Azure Machine Learning

There are many data orchestration tasks that can be conducted using Azure Data Factory. In this exercise, we'll copy data from Azure Data Lake Storage Gen1 to Azure Data Lake Storage Gen2.

NOTE

If you don't have an Azure account, or prefer not to do the exercise in your account, you can read through the following instructions to understand the steps involved using Azure Data Factory to copy data into an Azure Data Lake.

Create an Azure Data Factory instance

The first step is to provision an instance of Azure Data Factory in the Azure portal.

1. Sign into the **Azure portal**⁵.

⁵ <https://portal.azure.com?azure-portal=true>

2. On the left menu, select **New > Data + Analytics > Data Factory**:

The screenshot shows the 'New' blade in the Azure portal. On the left is a sidebar with a 'Create a resource' button highlighted by a red box. Below it are various service categories: Home, Dashboard, All services, Favorites (with All resources, Resource groups, App Services, SQL databases, SQL data warehouses, Azure Cosmos DB, Virtual machines, Load balancers, Storage accounts, Virtual networks, Azure Active Directory, Monitor, Advisor, Security Center, Cost Management + Billing, and Help + support). In the main area, there's a 'Recently created' section and a list of other services: HDInsight, Data Lake Analytics, Stream Analytics job, Analysis Services, Azure Databricks, Power BI Embedded, SQL Data Warehouse, Data Lake Storage Gen1, and Data Factory. The 'Analytics' and 'Data Factory' items are also highlighted with red boxes.

3. In the **New data factory** page, provide values for each of the required fields.

- Enter a globally unique name for your Azure data factory. Try using your initials with a suffix such as **ADFTutorialDataFactory**.
- In the Subscription drop-down list, click on your subscription
- In the resource group drop-down list, select **mslearn-datalake-test**
- Select **V2** for the version.
- Select the location for the data factory. Only supported locations are displayed in the drop-down list. The data stores that are used by the data factory can be in other locations and regions.

New data factory X

* Name i
LoadADLSDemo ✓

* Subscription
<your subscription name>

* Resource Group i
 Create new Use existing
demo

Version i
V2

* Location i
East US

Pin to dashboard

Create [Automation options](#)

4. Select **Create**.

After creation is complete, navigate to the new data factory. You should see the **Data Factory** home page.

The screenshot shows the Azure Data Factory home page. At the top, there's a header with a factory icon and the text '<your factory name> Data factory'. Below the header, there's a 'Delete' button and a 'Quick links' section. The 'Quick links' section contains two items: 'Documentation' and 'Author & Monitor'. The 'Author & Monitor' link is highlighted with a red box. To the right of the quick links, there's an 'Essentials' section displaying resource group information (Resource group: <your resource group>, Location: EastUS, Provisioning state: Succeeded), type (Data factory (V2)), and subscription details (Subscription name: <your subscription name>, Subscription id: <your subscription id>). Below the essentials section is a 'Monitoring' section titled 'PipelineRuns' with a status bar showing '8 AM' and two entries: 'SUCCEEDED PIPELINE...' and 'FAILED PIPELINE RU...'. There's also a 'All settings' button.

[!IMPORTANT]

You will need an Azure Data Lake Storage Gen1 account with data in it. If you do not have this perform the following steps.

Create a Data Lake Storage Gen 1 Account

1. In the left sidebar, click on **+** **Create a new resource**.
In the **New** blade, click **Storage** and then click **Data Lake Storage Gen1**.
2. In the Name text box, type **dlsgen1XXX** replace "XXX" with numbers of your choice. A green tick should appear confirming that the name is unique.
3. In the Subscription drop-down list, click on your subscription.
4. In the resource group drop-down list, select **mslearn-datalake-test**.
5. Select a location - you typically want to select a region near where the data consumption will occur. In this example, select a location near you.
6. Click **Create**.

Create a second sample text file

To provide some sample data to work with, create a local text file on your computer named **salesUK.txt** and paste the following text into the file.

```
#salaries Details
#Company Information
#Fields : Date company employee Salaries
01-02-2019 d1 f1 8000
01-02-2019 d2 f2 9000
01-02-2019 d1 f3 2000
01-02-2019 d2 f4 3000
01-02-2019 d1 f5 4000
01-02-2019 d3 f6 5000
```

We'll upload this data file in various ways. Keep in mind that this is a *simple* example - you would typically be populating your data lake with much larger data samples from a variety of sources.

Upload a file into data lake storage Gen 1 account

1. In the Azure portal, search for the Data Lake Storage Gen1 service you created (**dlsgen1XXX**).
2. In the overview blade, click **Data Explorer**
3. In the Data Explorer blade, click on the **Upload** icon.
4. In the Upload file blade, click on the browse icon, browse to the folder, and select your **salesUK.txt** file, click on the button **Add selected files**. Confirmation that the upload has completed is when the states column states **completed**.
5. Close the Upload files blade.

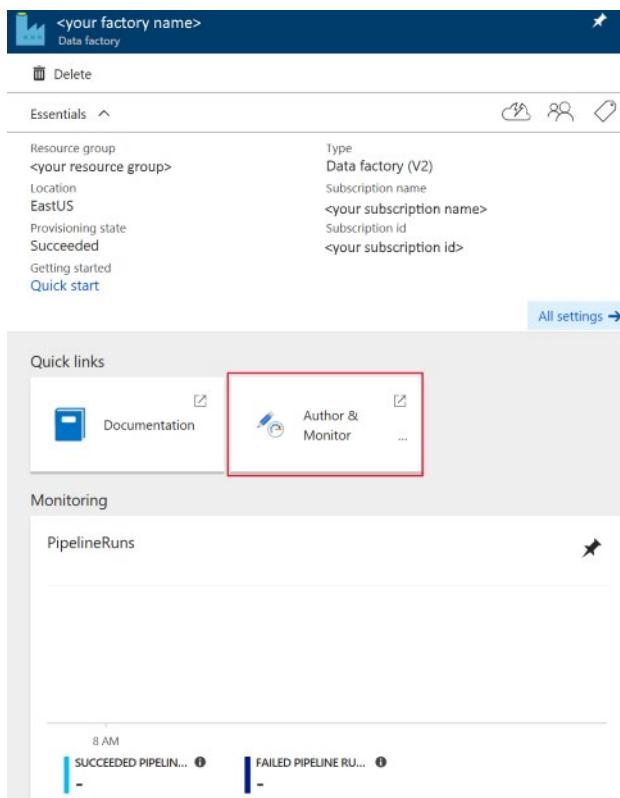
Setting permissions on the data lake storage Gen 1 account

Next, you need to set permissions to enable the Azure Data Factory instance to access the data in your Data Lake Store Gen 1.

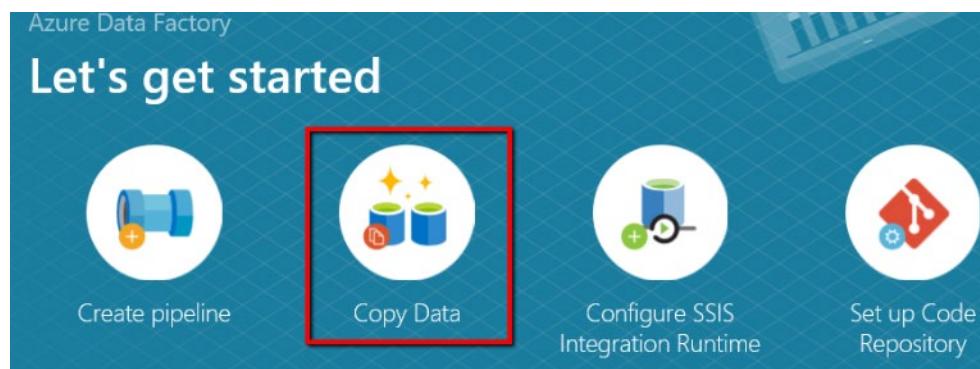
1. In the Azure portal, search for the Data Lake Storage Gen1 named **dlsgen1XXX** that you created.
2. In the overview blade, click **Access control (IAM)**
3. In the Access Control (IAM) blade, click on the **+ Add Role Assignment** button.
4. In the Add Role Assignment blade, select **Owner** for the **Role**.
5. Select the text box under **Select** and type in the Azure Data Factory instance name you created.
6. Click **Save**.
7. Close the **Access control (IAM)** blade.

Load data into Azure Data Lake Storage Gen2

1. In the Azure portal, go to your data factory. You see the **Data Factory** home page.
2. Select the **Author & Monitor** tile to launch the Data Integration Application in a separate tab.



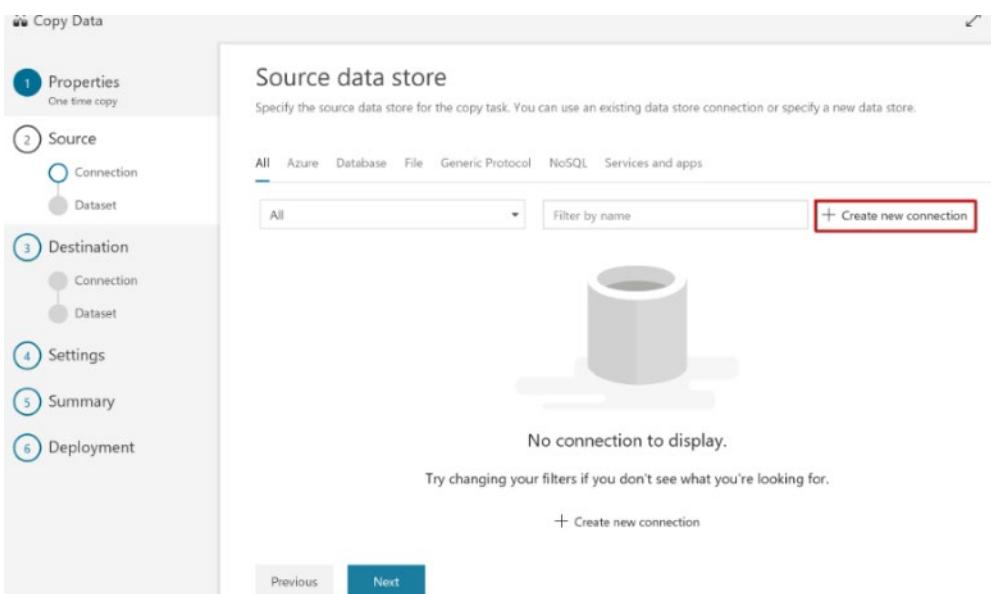
3. In the **Get started** page, select the **Copy Data** tile to launch the Copy Data tool.



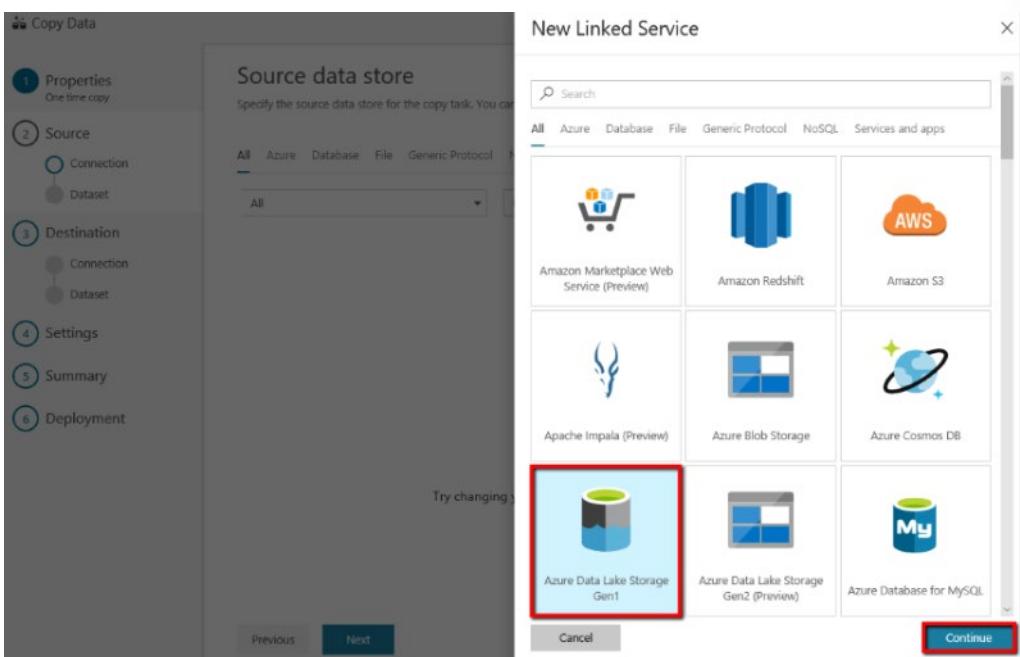
4. In the **Properties** page, specify **CopyFromADLSGen1ToGen2** for the **Task name** field, set the cadence to "once", and select **Next**:

This screenshot shows the 'Copy Data' properties page in the Azure portal. The left sidebar lists steps 1 through 6: 1. Properties (selected), 2. Source, 3. Destination, 4. Settings, 5. Summary, 6. Deployment. The main panel shows the 'Properties' configuration. The 'Task name' field is filled with 'CopyFromADLSGen1ToGen2'. The 'Task cadence or Task schedule' section has 'Run once now' selected. At the bottom are 'Previous' and 'Next' buttons, with 'Next' being the active button.

5. In the **Source data store** page, click **+ Create new connection**.

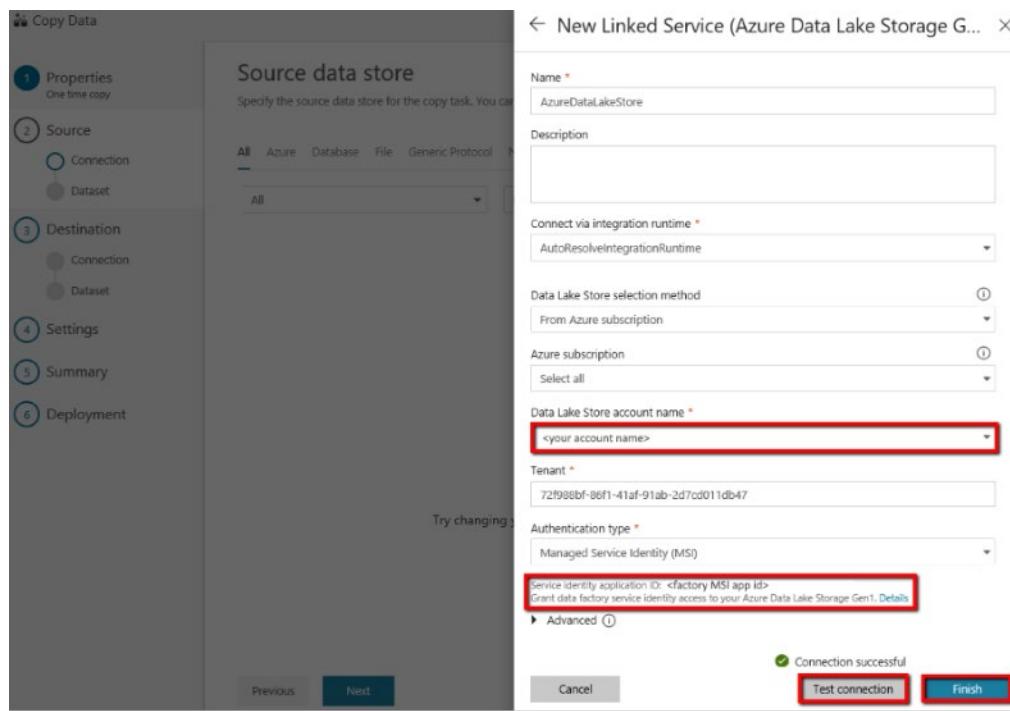


6. Select **Azure Data Lake Storage Gen1** from the connector gallery, and select **Continue**.

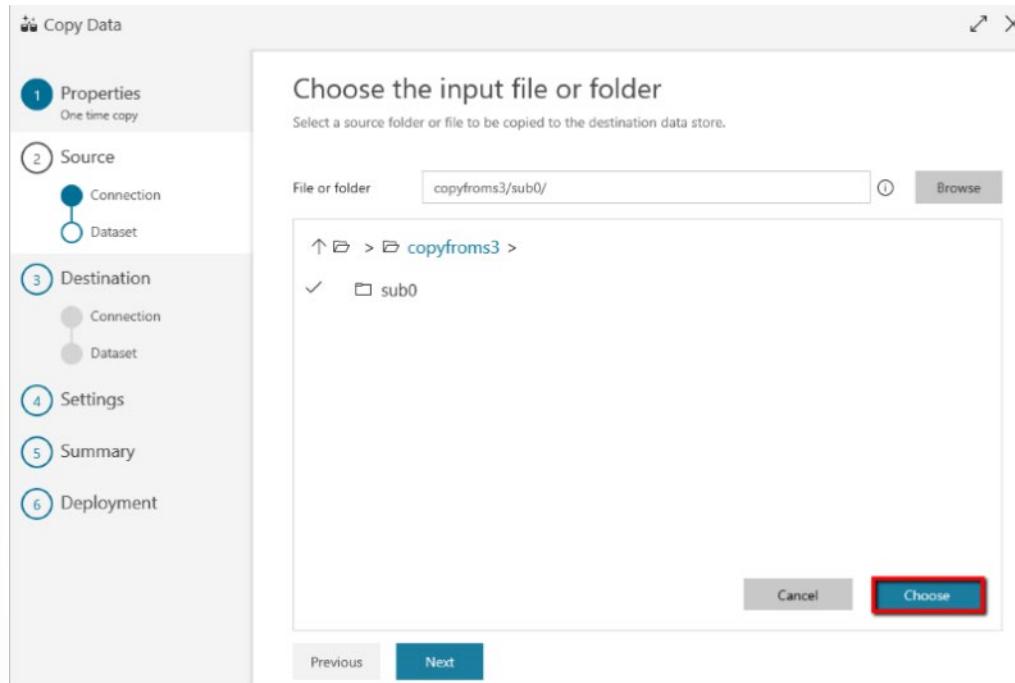


7. In the **Specify Azure Data Lake Storage Gen1 connection** page, do the following steps:

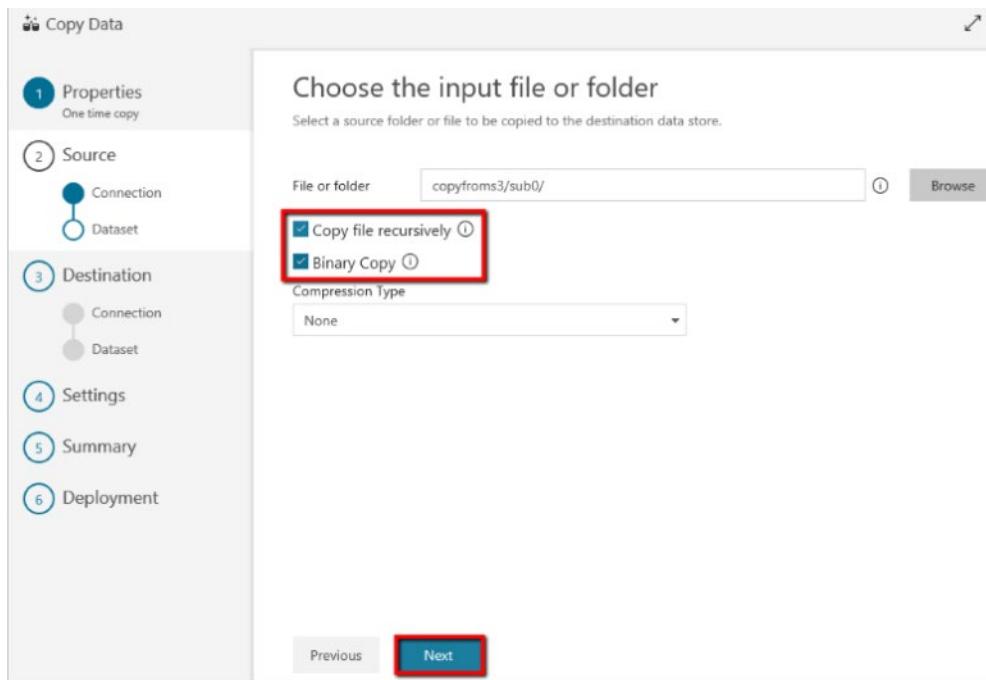
- Select your Data Lake Storage Gen1 for the name and specify or validate the Tenant.
- Click **Test connection** to validate the settings, then select **Finish**.
- You'll see a new connection gets created. Select **Next**.



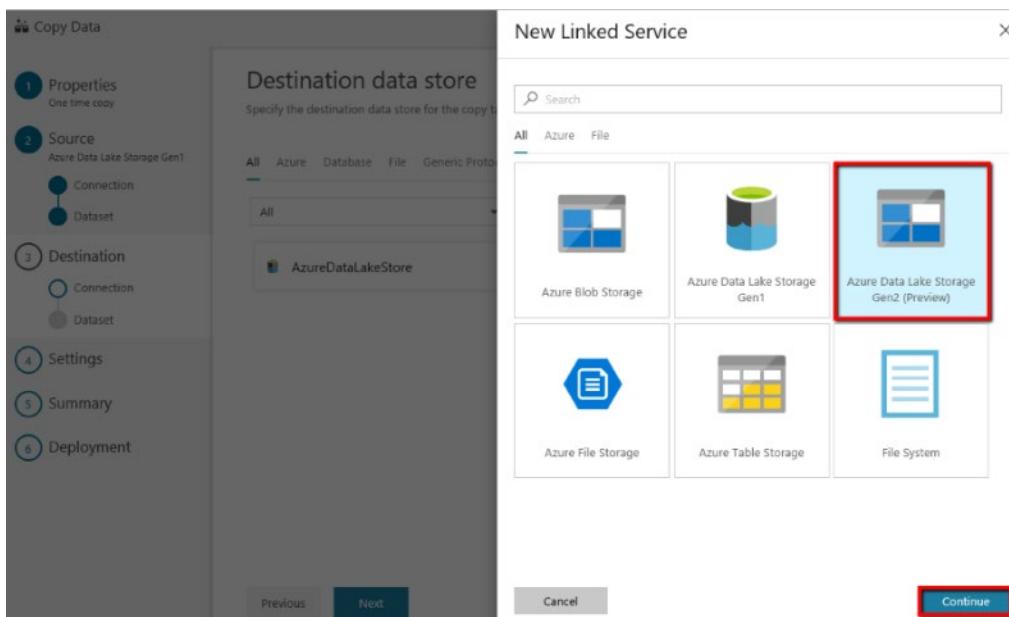
8. In the **Choose the input file or folder** page, browse to the folder and file that you want to copy over. Select the folder/file, select **Choose**.



9. Specify the copy behavior by checking the **Copy files recursively** and **Binary copy** options. Select **Next**:



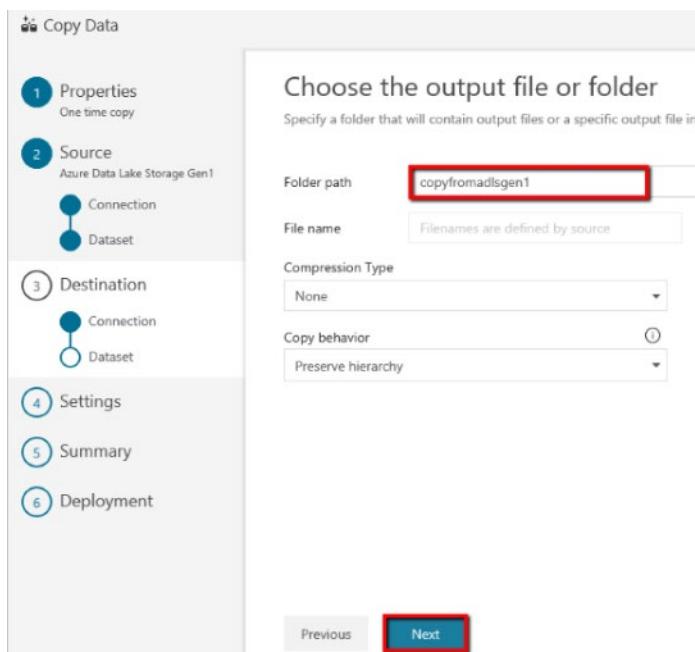
10. In the **Destination data store** page, click + **Create new connection**, select **Azure Data Lake Storage Gen2 (Preview)**, and click **Continue**.



11. In the **Specify Azure Data Lake Storage Gen2 connection** page, do the following steps:

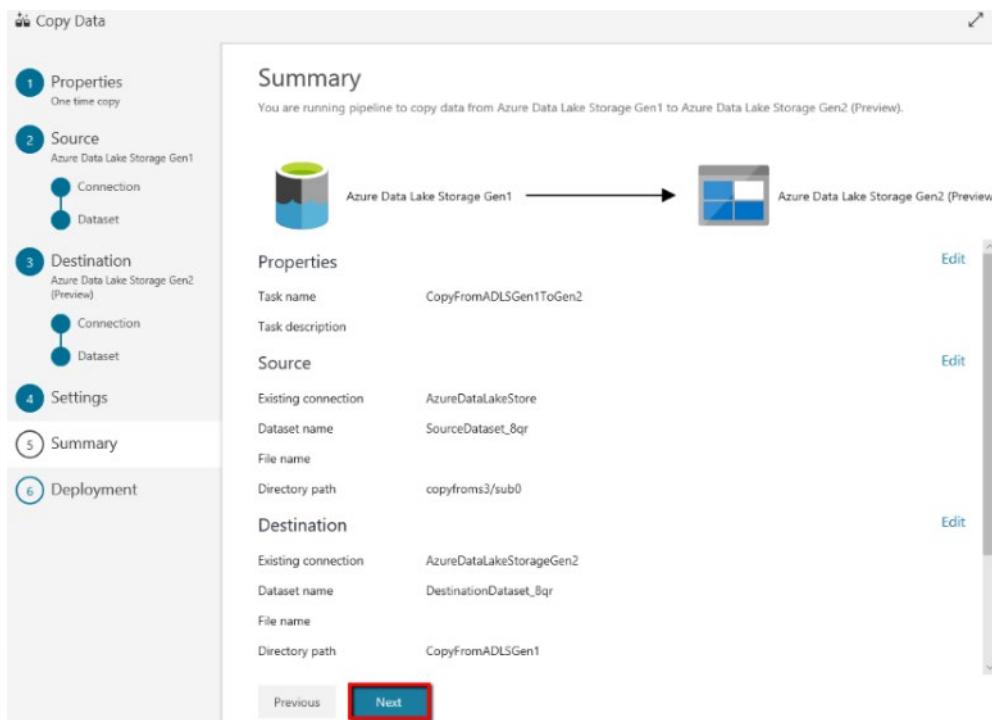
- Select your Data Lake Storage Gen2 capable account from the "Storage account name" drop-down list.
- Select **Finish** to create the connection. Then select **Next**.

12. In the **Choose the output file or folder** page, enter **copyfromadlsgen1** as the output folder name, and select **Next**.

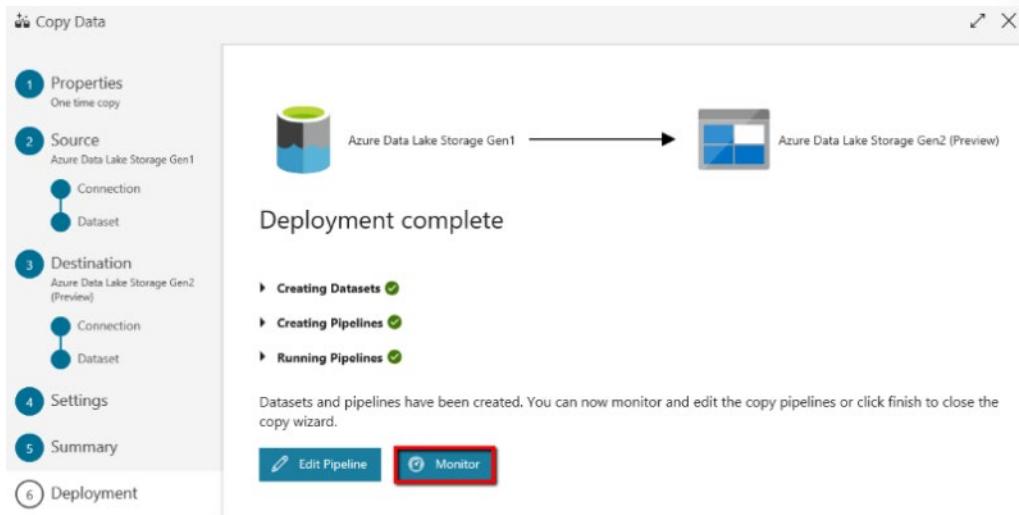


13. In the **Settings** page, select **Next** to use the default settings.

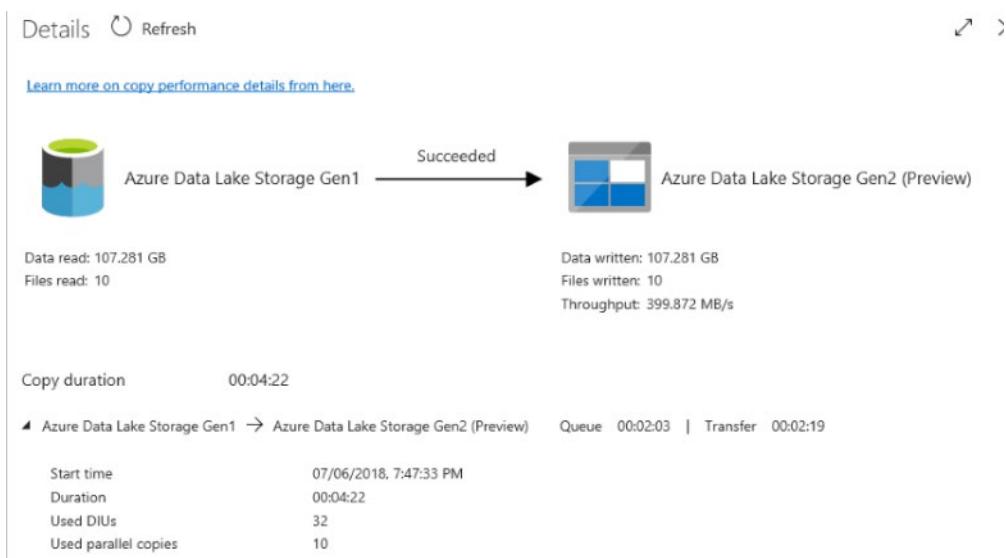
14. Review the settings, and select **Next**.



15. In the **Deployment page**, select **Monitor** to monitor the pipeline.



You can monitor details like the volume of data copied from the source, data throughput, execution steps with the corresponding duration, and used configurations.



Once the transfer is complete, you can verify the data has been copied into your Data Lake Storage Gen2 account with the Azure Storage Explorer.

Summary

Azure Data Lake Storage Gen2 provides a cloud storage service that is highly available, secure, durable, scalable, and redundant. It's the most comprehensive data lake solution available in market today. Azure Data Lake Storage brings new efficiencies to process big data analytics workloads and can provide data to a multitude of compute technologies including HDInsight and Azure Databricks without needing to move the data around. Creating an Azure Data Lake Storage Gen2 data store should be one of your go-to tools in building a big data analytics solution.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Contoso has a Data Lake Storage Account (Gen 2). Which tool would be the most appropriate tool to perform a one time without the installation or configuration of a tool to upload of a single file?

- Azure Data Factory
- Azure Storage Explorer
- Azure Portal

Question 2

Contoso has a Data Lake Storage Account (Gen 2). Which tool would be the most appropriate tool to perform a movement of hundreds of files from Amazon S3 to Azure Data Lake Storage ?

- Azure Data Factory
- Azure Data Catalog
- Azure Portal

Module Summary

Module Summary

In this module, you have learned the variety of ways to store data in Azure. You learned the basics of storage management in Azure, how to create a Storage Account, and how to choose the right model for the data you want to store in the cloud. You also understand how data lake storage can be created to support a wide variety of big data analytics solutions with minimal effort.

Learning Objectives

In this module, you have learned:

- A data storage approach in Azure
- How to create an Azure Storage Account
- How Azure Data Lake Storage Gen2 works
- How to upload data into Azure Data Lake

Post Course Review

After the course, consider watching **this video with James Baker and Lara Rubbelke that provides an overview of the enhancements of Azure Data Lake Storage Gen2⁶**.

Important

Remember

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

⁶ <http://aka.ms/DLStoreGen2>

Answers

Question 1

Suppose you work at a startup with limited funding. Why might you prefer Azure data storage over an on-premises solution?

- To ensure you run on a specific brand of hardware which will let you to form a marketing partnership with that hardware vendor.
- The Azure pay-as-you-go billing model lets you avoid buying expensive hardware.
- To get exact control over the location of your data store.

Explanation

There are no large, up-front capital expenditures (CapEx) with Azure. You pay monthly for only the services you use (OpEx).

Question 2

Which of the following situations would yield the most benefits from relocating an on-premises data store to Azure?

- Unpredictable storage demand that increases and decreases multiple times throughout the year.
- Long-term, steady growth in storage demand.
- Consistent, unchanging storage demand.

Explanation

Azure data storage is flexible. You can quickly and easily add or remove capacity. You can increase performance to handle spikes in load or decrease performance to reduce costs. In all cases, you pay for only what you use.

Question 1

Suppose you have two video files stored as blobs. One of the videos is business-critical and requires a replication policy that creates multiple copies across geographically diverse datacenters. The other video is non-critical, and a local replication policy is sufficient. True or false: To satisfy these constraints, the two blobs will need to be in separate storage accounts.

- True
- False

Explanation

Replication policy is a characteristic of a storage account. Every member in the storage account must use the same policy. If you need some data to use the geo-replication strategy and other data to use the local replication strategy, then you will need two storage accounts.

Question 2

The name of a storage account must be:

- Unique within the containing resource group.
- Unique within your Azure subscription.
- Globally unique.

Explanation

The storage account name is used as part of the URI for API access, so it must be globally unique.

Question 3

In a typical project, when would you create your storage account(s)?

- At the beginning, during project setup.
- After deployment, when the project is running.
- At the end, during resource cleanup.

Explanation

Storage accounts are usually stable for the lifetime of a project. It is common to create them at the start of a project.

Question 1

Mike is creating an Azure Data Lake Storage Gen 2 account. He must configure this account to be able to processes analytical data workloads for best performance. Which option should he configure when creating the storage account?

- On the Basic Tab, set the performance option to standard
- On the Basic Tab, set the performance to ON
- On the Advanced tab, set the Hierarchical Namespace to enabled

Explanation

If you want to enable the best performance for Analytical Workloads in Data Lake Storage Gen 2, then on the Advanced tab of the Storage Account creation set the Hierarchical Namespace to enabled. A is incorrect. Performance determines the types of physical storage that the storage account will use. Standard is a magnetic hard drive. B is incorrect. The Performance option set to ON is not a valid configuration.

Question 2

In which phase of big data processing is Azure Data Lake Storage located?

- Ingestion
- Store
- Model & Serve

Explanation

Store is the phase in which Azure Data Lake Storage resides for processing big data solution. Ingestion is the phase that typically includes technologies for ingesting data from a source such as Azure Data Factory. Model & Serve is the phase that includes technologies for presenting data for end users such as Power BI.

Question 1

Contoso has a Data Lake Storage Account (Gen 2). Which tool would be the most appropriate tool to perform a one time without the installation or configuration of a tool to upload of a single file?

- Azure Data Factory
- Azure Storage Explorer
- Azure Portal

Explanation

The Azure Portal requires no installation or configuration to work, and just requires a sign in and a click of an upload button to perform the upload. Azure Data Factory requires provisioning and configuring and Azure Storage Explorer requires installing this tool first.

Question 2

Contoso has a Data Lake Storage Account (Gen 2). Which tool would be the most appropriate tool to perform a movement of hundreds of files from Amazon S3 to Azure Data Lake Storage ?

- Azure Data Factory
- Azure Data Catalog
- Azure Portal

Explanation

Azure Data Factory would efficiently handle the movement of data from Amazon S3 to Azure Data Lake store. Azure Data Catalog is the wrong tool for the job required as it is used to document information about data stores. The Azure Portal would not support the copy of files from Amazon S3, and just requires a sign in and a click of an upload button to perform the upload.

Module 3 Enabling Team Based Data Science with Azure Databricks

Module Introduction

Enabling Team Based Data Science with Azure Databricks

This module introduces students to Azure Databricks and how a Data Engineer works with it to enable Team Data Science projects. They will learn the fundamentals of Azure Databricks and Apache Spark notebooks; how to provision the service and workspaces and learn how to perform data preparation task that can contribute to the data science project.

Learning objectives

In this module, you will:

- Describe Azure Databricks
- Provision Azure Databricks and Workspaces
- Read Data using Azure Databricks
- Perform transformations with Azure Databricks

Introduction to Azure Databricks

Introduction to Azure Databricks

Databricks is a version of the popular open-source **Apache Spark¹** analytics and data processing engine. Within Azure, Azure Databricks is a fully managed premium service offering, bringing an enterprise-grade and secure cloud-based Big Data and Machine Learning platform. This platform enables Data Scientists, Business Analysts and Data Engineers to work collaboratively to deliver production based data science solutions.

Learning objectives

In this section, you will learn:

- What is Azure Databricks
- What are Spark based analytics platform
- How Azure Databricks integrates with Enterprise Security
- How Azure Databricks integrates with other Cloud Services

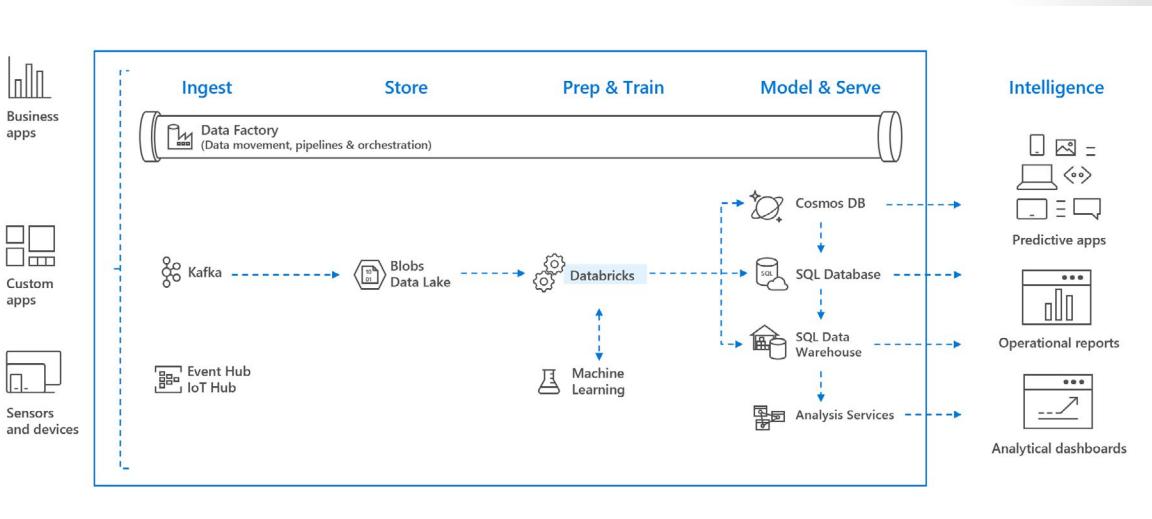
What is Azure Databricks

Databricks is a version of the popular open-source **Apache Spark²** analytics and data processing engine. Azure Databricks is the fully managed version of Databricks and is a premium offering on Azure, that brings you an enterprise-grade and secure cloud-based Big Data and Machine Learning platform. The intention of this platform is to provide the ease of deploying a collaborative Machine Learning environment based on Spark that can be used between data scientists, data engineers, and business analysts.

Data can be ingested in a variety of ways into Azure Databricks. For real-time Machine learning projects, you can ingest data through a wide range of technologies including Kafka, Event Hubs or IoT Hubs. In addition, you can ingest batches of data using Azure Data Factory from a variety of data stores including Azure Blob Storage, Azure Data Lake Storage, Azure Cosmos DB, or Azure Synapse Analytics which can then be used in the Spark based engine within Databricks.

¹ <https://spark.apache.org/>

² <https://spark.apache.org/>



Finally, Databricks provides a notebook-oriented Apache Spark as-a-service workspace environment, making it easy to manage clusters and explore data interactively.

As a result, Azure Databricks provides the following benefits

- Apache Spark-based analytics platform
- Enterprise Security
- Integration with other Cloud Services

Apache Spark-based analytics platform

What is Apache Spark

As Big Data solutions such as Hadoop began to grow in the mid 2000's, towards the end of the decade, a cluster management framework was required to handle iterative computational activities such as Machine Learning. Hadoop was too slow for this type of use case. Therefore, Apache Spark emerged to provide a parallel processing framework that supports in-memory processing to boost the performance of big-data analytic applications on massive volumes of data. This meant that it was more achievable to support the following use cases, including:

- Machine Learning
 - Apache Spark contains a number of libraries that enables a Data Scientist to perform Machine Learning activities with libraries such as MLlib. Typically, a Data Scientist would also install Anaconda that provides a variety of machine learning packages to perform different types of machine learning activities. They would typically perform such activities in an application such as Jupyter notebooks
- Streaming and realtime analytics
 - Spark natively has connectors to ingest data from technologies such as Kafka and Flume to ingest data in real-time and support the creation of real-time analytical solutions.
- Interactive data analysis of unstructured data
 - Spark can also be used by business analysts or data engineers to analyze and prepare data for use in downstream Machine Learning or reporting solutions, or to be visualized in reports using a tool

such as Power BI. Users have the ability to work with unstructured data that is stored in the Spark Cluster and define schema in a notebook that can be used to query the data, or present it in a BI tools for business stakeholders to consume.

Spark provides the capability for in-memory cluster computing. A Spark job loads data into memory and query it repeatedly. In-memory computing is much faster than disk-based applications, such as Hadoop, which shares data through Hadoop distributed file system (HDFS). Spark also integrates into the Scala programming language to let you manipulate distributed data sets like local collections, so there's no need to structure everything as map and reduce operations.

Early incarnations of the product were on-premises based which required specialism to setup. The advent of the cloud has meant that this is now available as a platform-as-a-service, and in the case of Azure, the technology is known as HDInsight

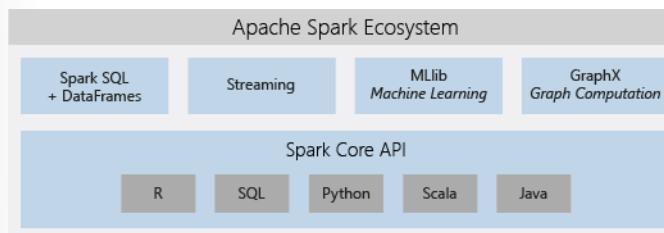
Why use Azure Databricks

Whilst it is undeniable that HDInsight simplifies the provisioning and configuration of a Spark cluster in comparison to its on premises counterpart. There are still aspects of HDInsight Spark that would be time consuming or requires specialist skills in order to configure. Databricks is essentially a wrapper around Apache Spark that simplifies the provisioning and configuration of a Spark cluster in a GUI interface.

In addition, some tasks can be performed in a click and drag interface for ease of use, although there is still the option to make use of notebooks should you prefer. Bringing Databricks into Azure makes sense as both technologies are designed to make life easier for Data Scientists, Data Engineers and Business Analysts alike.

Azure Databricks Components

Azure Databricks comprises the complete open-source Apache Spark cluster technologies and capabilities. Spark in Azure Databricks includes the following components:



- **Spark SQL and DataFrames:** Spark SQL is the Spark module for working with structured data. A DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python.
- **Streaming:** Real-time data processing and analysis for analytical and interactive applications. Integrates with HDFS, Flume, and Kafka.
- **MLlib:** Machine Learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives.
- **GraphX:** Graphs and graph computation for a broad scope of use cases from cognitive analytics to data exploration.
- **Spark Core API:** Includes support for R, SQL, Python, Scala, and Java.

With the Databricks **Serverless** option, Azure Databricks completely abstracts out the infrastructure complexity and the need for specialized expertise to set up and configure your data infrastructure. The Serverless option helps data scientists iterate quickly as a team.

For data engineers, who care about the performance of production jobs, Azure Databricks provides a Spark engine that is faster and performant through various optimizations at the I/O layer and processing layer (Databricks I/O).

Enterprise Security

As Data Science is becoming more embedded within an organizations analytical capability, there is a greater need to ensure that the platforms on which the project run are both secure and compliant. Azure Databricks provides enterprise-grade security, including Azure Active Directory integration, role-based controls, and SLAs that protect your data and your business.

- Integration with Azure Active Directory enables you to run complete Azure-based solutions using Azure Databricks.
- Azure Databricks roles-based access enables fine-grained user permissions for notebooks, clusters, jobs, and data.
- Enterprise-grade SLAs.

Note

Enterprise security is explored in depth in Module 8: Securing Azure Data Platforms.

Integration with Cloud Services

Integration with Azure services

One of the key benefits of using Azure Databricks is the ability to integrate with other products. It is able to ingest and output data to the following Azure data stores, including:

- Azure Synapse Analytics
- Cosmos DB
- Data Lake Store
- Azure Blob Storage
- Azure Data Factory

It can also work with streaming technologies such as Event Hubs or Apache Kafka, and use the AI capabilities of Azure Cognitive Services Text Analytics API to run sentiment analysis on streaming data

Integration with Analyst Tools

Azure Databricks can also act as a source of reporting for Power BI, which enables you to create and share your dashboards with others in your business. Furthermore, you can connect to databricks from Excel, R or Python to perform data analysis. There is also support for third party BI tools that can be accessed through JDBC/ODBC cluster endpoints, such as Tableau.

Summary

In this module, you learned about the purpose of using Azure Databricks, and why it would be used instead of other similar technologies. You also learned the use cases that it can support. Finally, you should know which Azure technologies it can integrate with to encourage its use by Data Scientists, Data Engineers and Business Analysts alike.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Azure Databricks encapsulates which Apache Storage technology?

- Apache HDInsight
- Apache Hadoop
- Apache Spark

Question 2

Which security features does Azure Databricks not support?

- Azure Active Directory
- Shared Access Keys
- Role-based access

Question 3

Which of the following Azure Databricks is used for support for R, SQL, Python, Scala, and Java?

- MLlib
- GraphX
- Spark Core API

Working with Azure Databricks

Provision Azure Databricks and Workspaces

In this section you will learn how to provision an Azure Databricks workspace. Once created, you will then create a Spark Cluster followed by a Notebook and then attach the notebook to the cluster so that you can start to use it.

Learning objectives

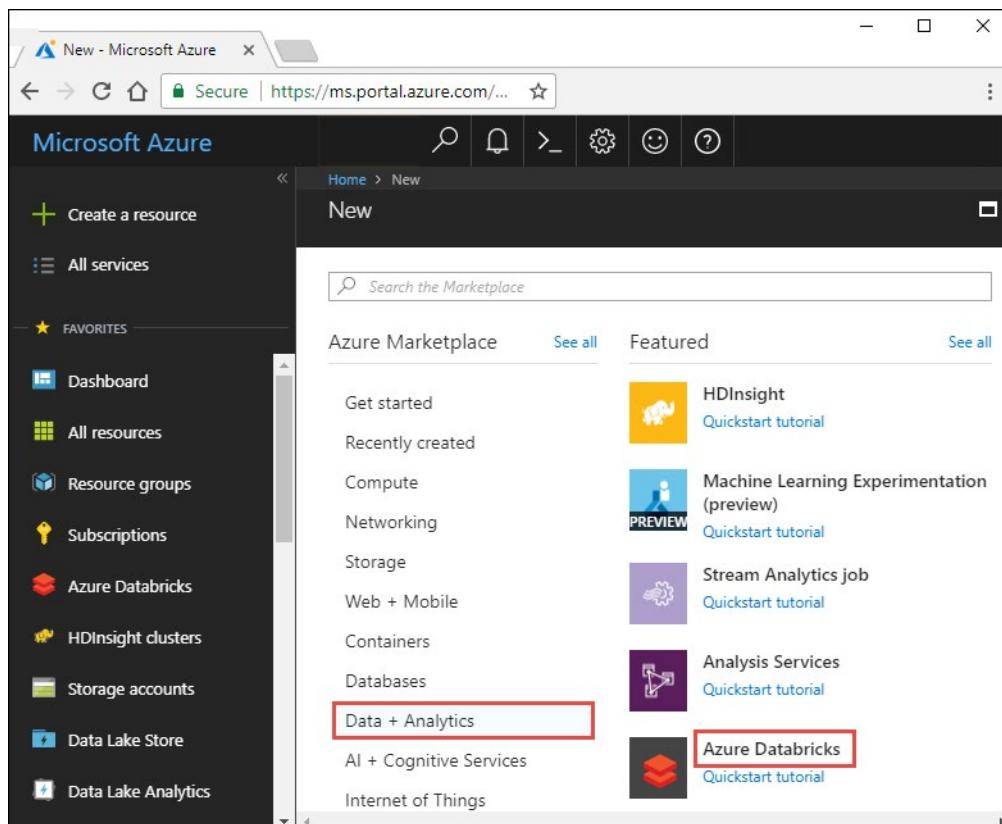
In this section, you will learn how to:

- Create your own Azure Databricks workspace
- Create a cluster and notebook in Azure Databricks

Provisioning an Azure Databricks workspace

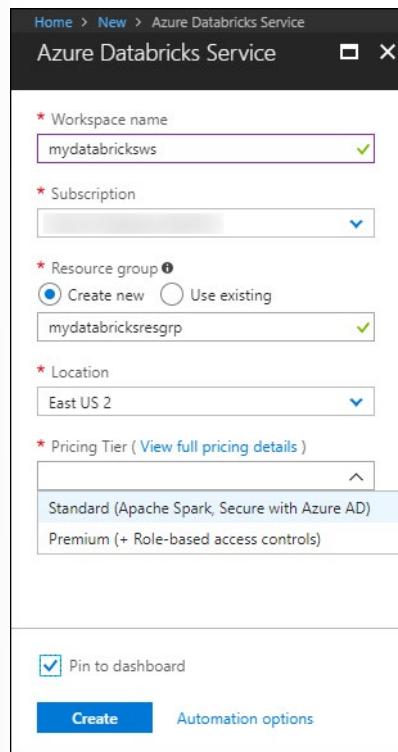
The first step to using Azure Databricks is to create and deploy a Databricks workspace. You can do this using the Azure portal.

1. In the Azure portal, click **Create a resource > Data + Analytics > Azure Databricks**.



2. Provide the required values to create your Azure Databricks workspace:
 - *Subscription*: Choose the Azure Subscription in which to deploy the workspace.
 - *Resource Group*: Leave at Create new and provide a name for the new resource group.

- *Location:* Select a location near you for deployment. For the list of regions supported by Azure Databricks, see **Azure services available by region³**.
- *Workspace Name:* Provide a name for your workspace.
- *Pricing Tier:* Ensure premium is selected.



3. Accept the terms and conditions.
4. Select **Purchase**.
5. The workspace creation takes a few minutes. During workspace creation, the portal displays the **Submitting deployment for Azure Databricks** tile on the right side. You may need to scroll right on your dashboard to see the tile. There's also a progress bar displayed near the top of the screen. You can watch either area for progress.

Creating Clusters and Notebooks

After creating your Databricks workspace, it's time to create your first notebook and Spark cluster.

What is Apache Spark notebook?

A notebook is a collection cells. These cells are run to execute code, render formatted text, or display graphical visualizations.

What is a cluster?

The notebooks are backed by clusters, or networked computers that work together to process your data. So, the first step is to create a cluster.

³ <https://azure.microsoft.com/regions/services/>

Creating a cluster

1. In the **Overview** pane of your Databricks services, click on the **Launch Workspace** button below the Databricks icon
2. On the left-hand menu of your Databricks workspace, under **Common Tasks** click **New Cluster** to add a new cluster.

Create Cluster

New Cluster

Cancel **Create Cluster** 2-8 Workers: 28.0-112.0 GB Memory, 8-32 Cores, 1 Driver: 14.0 GB Memory, 4 Cores, 0.75 DBU

Cluster Name
Test cluster

Cluster Mode ?
Standard

Databricks Runtime Version ? [Learn more](#)
Runtime: 5.0 (Scala 2.11, Spark 2.4.0)

Python Version ?
2

Autopilot Options
 Enable autoscaling
 Terminate after 120 minutes of inactivity ?

Worker Type
Standard_DS3_v2 14.0 GB Memory, 4 Cores, 0.75 DBU Min Workers 2 Max Workers 8

Driver Type
Same as worker 14.0 GB Memory, 4 Cores, 0.75 DBU

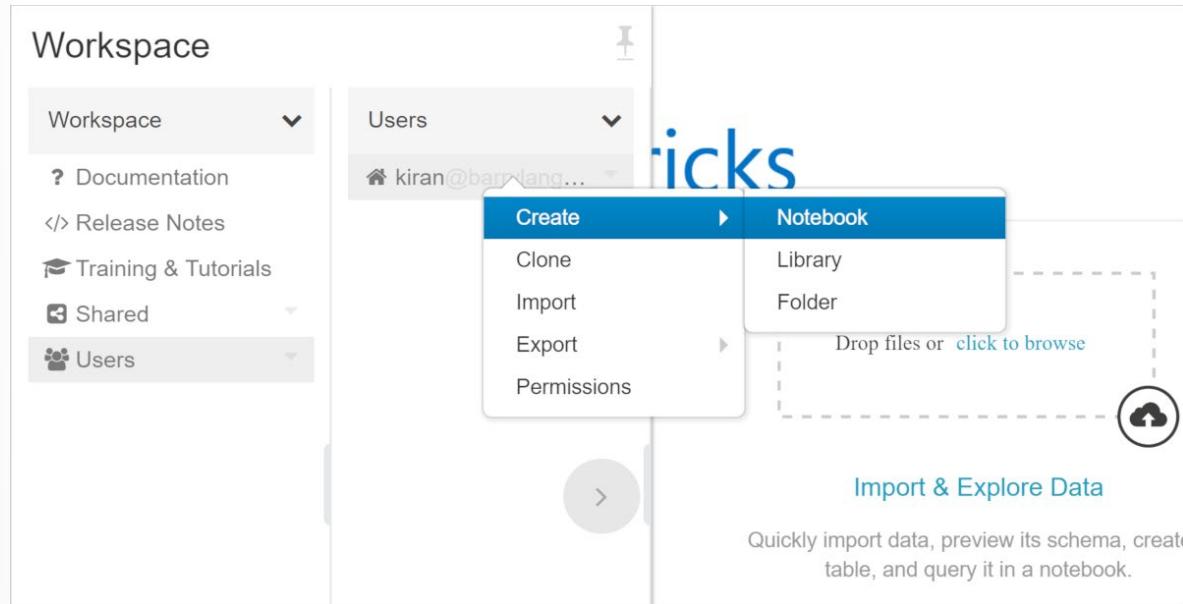
1. Enter a name for your cluster. Use your name or initials to easily differentiate your cluster from your coworkers.
2. Select the **Databricks RuntimeVersion**. We recommend the latest runtime (4.0 or newer) and Scala 2.11.
3. Specify your cluster configuration.
 - The default values are sufficient for the remaining fields.
 - For all other environments, refer to your company's policy on creating and using clusters.
4. Click **Create Cluster**.
5. As the cluster is being created, you will get status information about the cluster under the **Interactive Clusters** cell. This will take approximately 5 minutes.

NOTE

Check with your local system administrator to see if there is a recommended default cluster at your company to use for the rest of the class. This could save you some money!

Creating a notebook

1. On the left-hand menu of your Databricks workspace, click **Home**. A blade will appear named Workspace
2. In the second column, click on the **drop down arrow** next to your account name.
3. Point to **Create**.
4. Select **Notebook**.



1. Name your notebook **First Notebook**.
2. Set the language to **Python**. Note that you can choose R, Scala or SQL as well
3. Select the **cluster** that you have just created to which to attach this Notebook.

NOTE

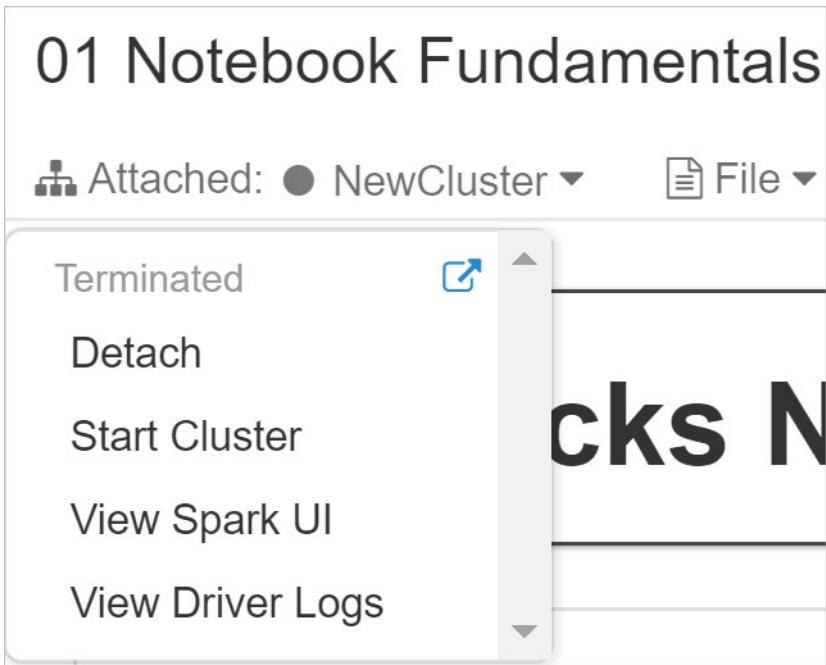
This option displays only when a cluster is currently running. You can still create your notebook and attach it to a cluster later.

1. Click **Create**.

Now that you've created your notebook, you can use it to run some code.

Attaching and detaching your notebook

To use your notebook to run a code, you must attach it to a cluster. You can also detach your notebook from a cluster and attach it to another depending upon your organization's requirements.



If your notebook is attached to a cluster you can:

- Detach your notebook from the cluster
- Restart the cluster
- Attach to another cluster
- Open the Spark UI
- View the Driver's log files

Summary

In this module, we learned about provisioning a Databricks workspace and Apache Spark notebooks. You also learned that notebooks allow you to interactively work with different types of data. You can use notebooks for processing huge data files, query, read, and write data from different sources, and processing live data streams.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which Notebook format is used in Databricks?

- DBC
- .notebook
- .spark

Question 2

Which browsers are recommended for best use with Databricks Notebook?

- Chrome and Firefox
- Microsoft Edge and IE 11
- Safari and Microsoft Edge

Reading Data using Azure Databricks

Reading Data using Azure Databricks

Databricks provides various built-in functions that allow you to query, process, and analyze a large volume of data. In this section you will explore the reading capabilities of Spark notebooks. Imagine your organization is a data analytics startup, which is now expanding with its increasing customer base. You receive customer data from multiple sources in different raw formats. The management has decided to invest in Azure Databricks to efficiently handle huge amount of customer data. Your team is responsible for analyzing this data to gain insights.

Learning objectives

In this section, you will:

- Use Azure Databricks to access data sources
- Reading Data in Azure Databricks

Accessing data

Databricks allows several ways to access your data. You can access data stored in an existing file, by uploading a data file from your local system, or by mounting an Azure blob to Databricks file system. Databricks accepts data stored in different file formats such as parquet and csv from the following types of data store

- **Data Lake** - Apache Spark and Databricks make it easy to access and work with files stored in Data lakes, such as Azure Data Lake Storage (ADLS). Data lake stores vast amount of data in its native format such as XML, JSON, CSV, and Parquet. You can use Databricks to perform exploratory data analysis (EDA) to gain insights from a Data Lake. Spark DataFrames can be used to read directly from raw files contained in a Data Lake and then execute queries to join and aggregate the data.
- **Azure Data Lake Storage Gen2** - Azure Data Lake Storage Gen2 (ADLS Gen2) is a set of capabilities dedicated to big data analytics, built on Azure Blob storage and Azure Data Lake Storage Gen1. Databricks allows you to provision an ADLS Gen2 instance and use the Azure Blob File System (ABFS) driver built into the Databricks Runtime to query data stored in ADLS Gen2.

You can also connect to the following additional data sources in Azure, including:

- Azure SQL Database
- Azure Cosmos DB
- Azure Synapse Analytics
- Azure Event Hubs

Reading Data in Azure Databricks

Querying files

You can use `DataFrames` to query large data files. `DataFrames` are derived from data structures known as Resilient Distributed Datasets (RDDs). RDDs and `DataFrames` are immutable distributed collections of data.

- *Resilient*: RDDs are fault tolerant, so if part of your operation fails, Spark quickly recovers the lost computation.
- *Distributed*: RDDs are distributed across networked machines known as a cluster.
- *DataFrame*: A data structure where data is organized into named columns, like a table in a relational database, but with richer optimizations under the hood.

Once created, a `DataFrame` object comes with methods attached to it. These methods are operations that you can perform on `DataFrames` such as filtering, counting, aggregating, and many others. Spark provides a number of built-in functions, many of which can be used directly with `DataFrames` to filter through the data and derive specific results.

Joins and aggregation

Several built-in functions of Databricks allow you to aggregate your data in different ways. You can use `DataFrames` to join different sets of data to find a correlation between them.

Working with hierarchical data

`DataFrames` makes it easier to query hierarchical data such as nested JSON records. You can query nested structured data or data containing array columns.

Key Vault-backed secret scopes

Azure Databricks has two types of secret scopes: Key Vault-backed and Databricks-backed. These secret scopes allow you to store secrets, such as database connection strings, securely. If someone tries to output a secret to a notebook, it's replaced by [REDACTED]. This helps prevent someone from viewing the secret or accidentally leaking it when displaying or sharing the notebook.

Summary

Databricks provide several built-in functions that makes your day-to-day data handling process easier. You can read, write, and query data to and from multiple sources. You can also join data from two different sources to aggregate your results.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

How do you connect your Spark cluster to the Azure Blob?

- By calling the .connect() function on the Spark Cluster.
- By mounting it
- By calling the .connect() function on the Azure Blob

Question 2

How does Spark connect to databases like MySQL, Hive and other data stores?

- JDBC
- ODBC
- Using the REST API Layer

Question 3

How do you specify parameters when reading data?

- Using .option() during your read allows you to pass key/value pairs specifying aspects of your read
- Using .parameter() during your read allows you to pass key/value pairs specifying aspects of your read
- Using .keys() during your read allows you to pass key/value pairs specifying aspects of your read

Performing Transformations with Azure Data-bricks

Performing Transformations with Azure Data-bricks

Databricks use DataFrames, where the core structure of data is immutable, to query, process, and analyze a large volume of data. Being immutable means that the data stored can't be changed once it's created. So how do you use Databricks if there's a requirement to change the original data created using DataFrame? You can change the format for the original data using transformation, and this is typically done through an ETL process.

Learning objectives

In this module, you will:

- Performing ETL to populate a data model
- Perform basic transformations
- Perform advanced transformations with user-defined functions

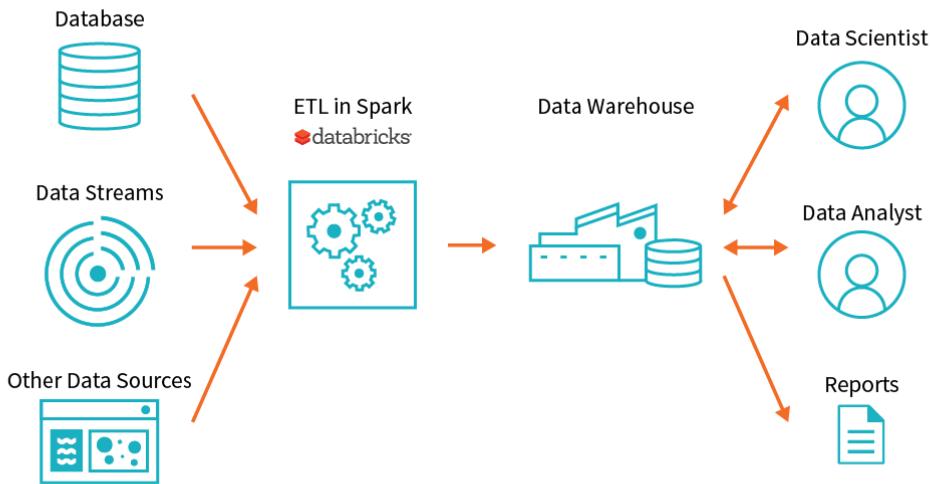
ETL Overview

The goal of transformation in Extract Transform Load (ETL) is to transform raw data to populate a data model. The most common models are relational models and snowflake (or star) schemas, though other models such as query-first modeling also exist.

The extract, transform, load (ETL) process takes data from one or more sources, transforms it, normally by adding structure, and then loads it into a target database. A common use case is an ETL job takes log files from a web server, parses out pertinent fields so it can be readily queried, and then loads it into a database.

ETL may seem simple: applying structure to data so it's in a desired form. However, the complexity of ETL is in the details. Data Engineers building ETL pipelines must understand and apply the following concepts:

- Optimizing data formats and connections
- Determining the ideal schema
- Handling corrupt records
- Automating workloads

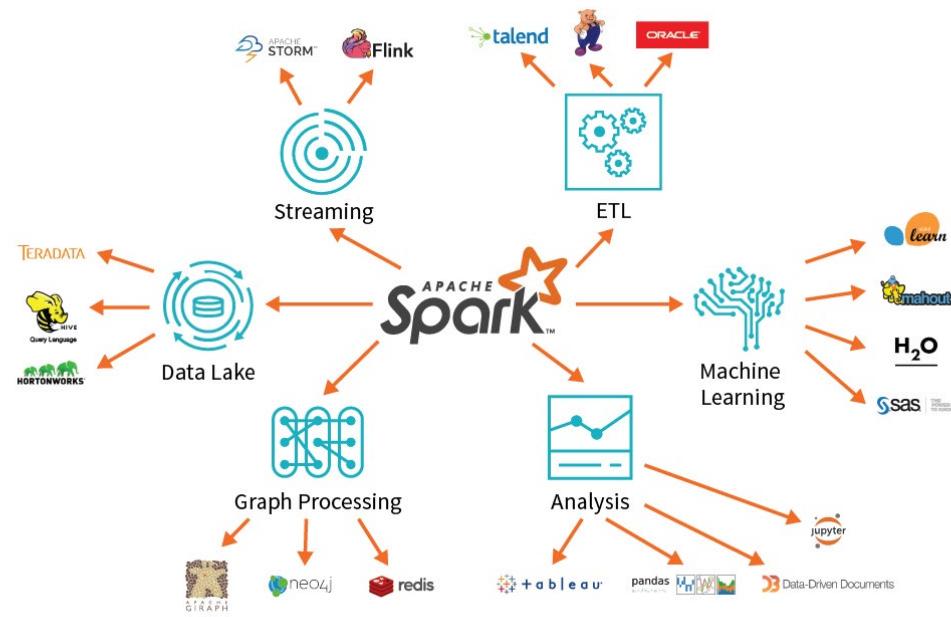


The Spark approach

Spark offers a compute engine and connectors to virtually any data source. By leveraging easily scaled infrastructure and accessing data where it lives, Spark addresses the core needs of a big data application.

These principles comprise the Spark approach to ETL, providing a unified and scalable approach to big data pipelines:

- Databricks and Spark offer a unified platform
 - Spark on Databricks combines ETL, stream processing, machine learning, and collaborative notebooks.
 - Data scientists, analysts, and engineers can write Spark code in Python, Scala, SQL, and R.
- Spark's unified platform is scalable to petabytes of data and clusters of thousands of nodes.
 - The same code written on smaller data sets scales to large workloads, often with only small changes.
- Spark on Databricks decouples data storage from the compute and query engine.
 - Spark's query engine connects to any number of data sources such as S3, Azure Blob Storage, Redshift, and Kafka.
 - It minimizes costs as there is no need to maintain a dedicated cluster and the compute cluster is easily updated to the latest version of Spark.



ETL process

A typical ETL process in Databricks includes the following steps:

1. **Extraction.** You can virtually connect to any data store including Azure Blob Storage, using Java Database Connectivity (JDBC). Databricks support connection to multiple database types, including:
 - Traditional databases like Postgres, SQL Server, and MySQL
 - Message brokers like Kafka and Kinesis
 - Distributed databases like Cassandra and Redshift
 - Data warehouses like Hive and Cosmos DB
 - File types like CSV, Parquet, and Avro
2. **Data validation.** One aspect of ETL jobs is to validate that the data is what you expect. The data validation step primarily includes finding out:
 - Approximately the expected number of records
 - The expected fields are present
 - No unexpected missing values
3. **Transformation.** This step generally includes applying structure and schema to your data to transform it into the desired format. Schemas can be applied to tabular data, such as that found in CSV files or relational databases, or to semi-structured data such as JSON data.
4. **Corrupt record handling.** Handling the bad data is also an important part of the entire ETL process. The built-in functions of Databricks allow you to handle corrupt data such as missing and incomplete information, schema mismatch, differing formats or data types, and user errors when writing data producers. They appear in a column called “`_corrupt_record`”

5. **Loading data** - A common and highly effective design pattern in the Databricks and Spark ecosystem involves loading structured data back to DBFS as a parquet file. Databricks also allows you to productionize code by scheduling notebooks for regular execution.

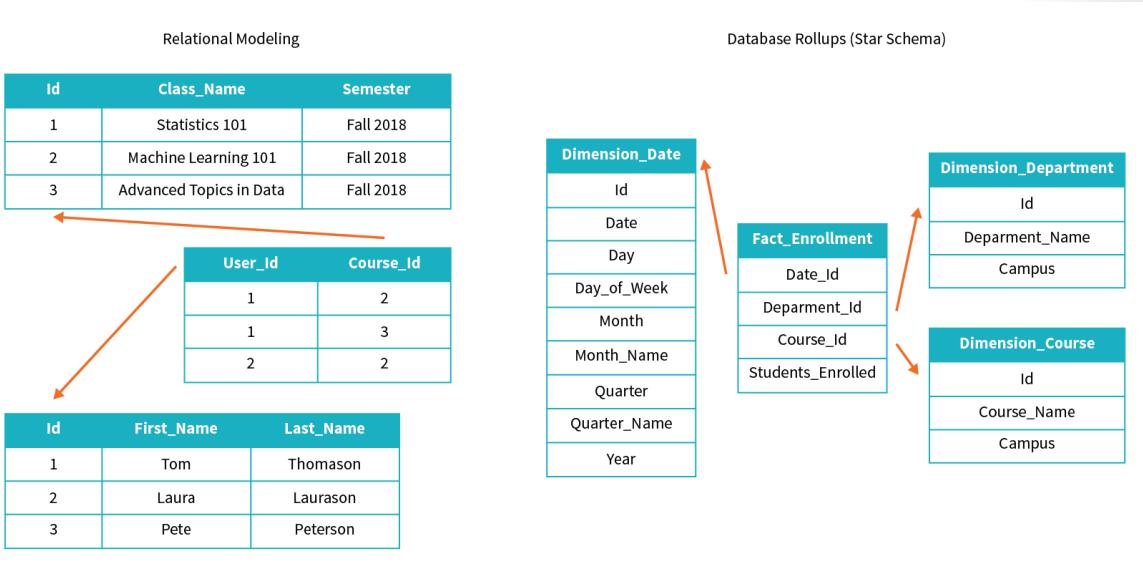
Performing Basic Transformations with Azure Databricks

When you receive data from variety of sources in different unstructured format, you need to transform and structure it into a common format before storing it. This transformation makes your data easy to use for any further analysis purposes.

There are different transformation techniques that can be selected based on the complexity of the raw data and the final output requirements. The basic transformation includes applying simple schemas to a semi-structured or tabular raw data or using built-in functions to clean corrupt data.

Transforming data can range in complexity from simply parsing relevant fields to handling null values without affecting downstream operations and applying complex conditional logic. Common transformations include:

- Normalizing values
- Inputting null or missing data
- De-duplicating data
- Pivoting DataFrames



If the structure of your raw data is of more complex nature, or you are required to prepare your data for advanced and complex analytical requirements, the basic transformation techniques may not serve the purpose. Then you need to apply some advanced data transformation methods to clean and prepare your data.

Databricks provide the capabilities to perform the end-to-end process of data extraction from the source, transformation, and finally loading into the target database. This entire process is call extract, transform, load (ETL).

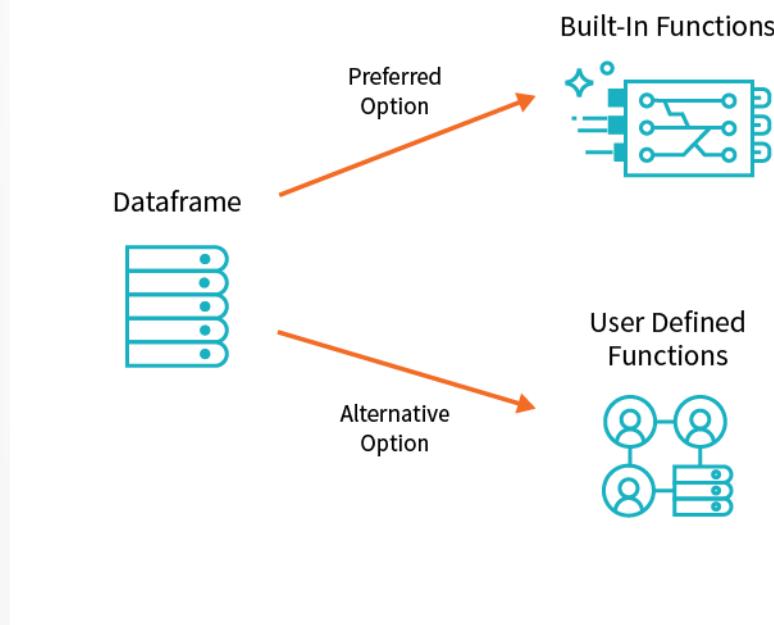
Advanced Transformational

Advanced ETL processes include data transformation using custom and advanced user-defined functions, managing complex tables and loading data into multiple databases simultaneously.

Custom and complex transformations with user-defined functions (UDF)

Spark's highly optimized built-in functions provide a wide array of functionality, covering the vast majority of data transformation use cases. However, there may be a scenario when you need to define logic specific to your use case and when you need to encapsulate that solution for reuse. User-Defined Functions (UDFs) come in handy in such cases. You should use UDFs when there's no clear way to accomplish a task using built-in functions.

UDFs provide custom, generalizable code that you can apply to ETL workloads when Spark's built-in functions won't suffice. UDFs can take multiple column inputs, and though they can't return multiple columns, they can return complex, named types that are easily accessible. This approach is especially helpful in ETL workloads that need to clean complex and challenging data structures.

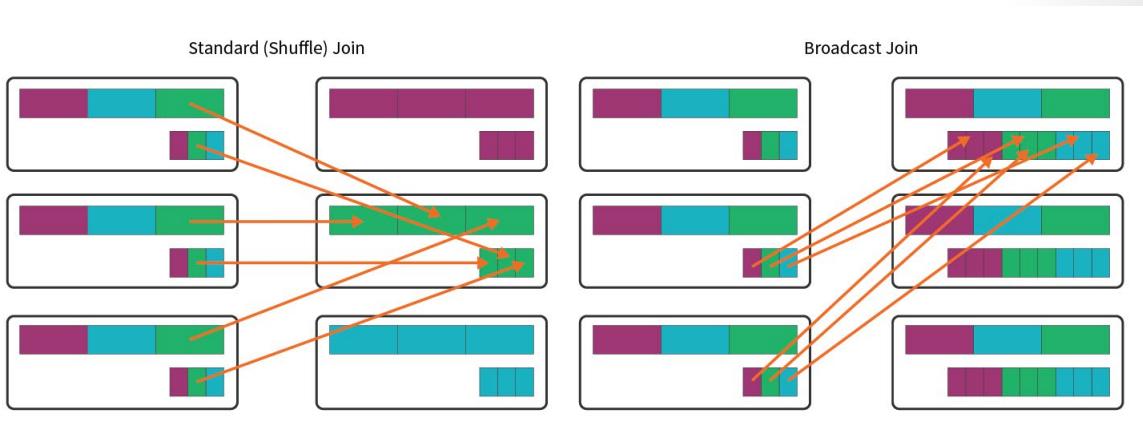


Joins and lookup tables

A common use case in ETL jobs involves joining new data to either lookup tables or historical data. Traditional databases join tables by pairing values on a given column.

A **standard** (or **shuffle**) join moves all the data on the cluster for each table to a given node on the cluster. This is expensive not only because of the computation needed to perform row-wise comparisons, but also because data transfer across a network is often the biggest performance bottleneck of distributed systems.

A **broadcast** join remedies this situation when one DataFrame is sufficiently small. A broadcast join duplicates the smaller of the two DataFrames on each node of the cluster, avoiding the cost of shuffling the bigger DataFrame.



Writing to multiple databases

Loading your transformed data to multiple target databases can be a time-consuming activity and can have an impact on your database connection. Spark makes this job easier as there're a number of variables that you can tweak to optimize performance, largely relating to how data is organized on the cluster.

Partitions are one of the options to get optimum performance from your database connections. A partition is a portion of your total data set, which is divided into many of these portions so Spark can distribute your work across a cluster.

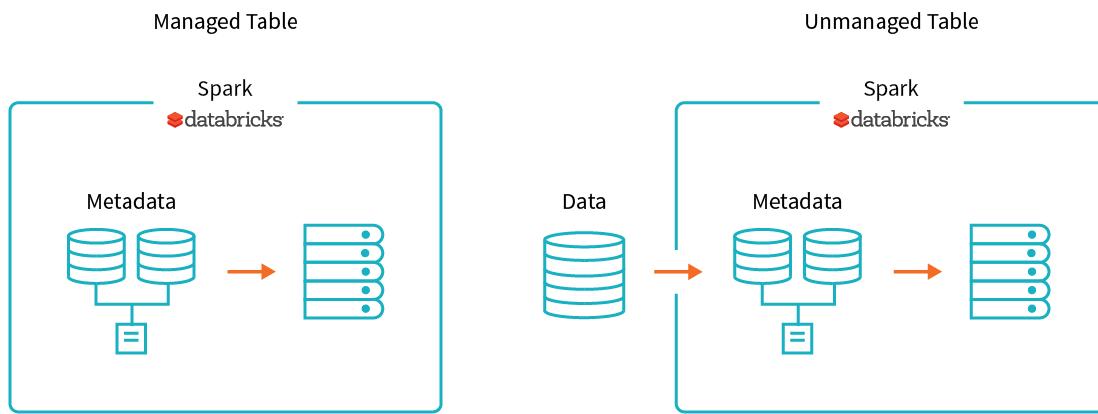
The other concept needed to understand Spark's computation is a slot (also known as a core). A slot/core is a resource available for the execution of computation in parallel. In brief, a partition refers to the distribution of data while a slot refers to the distribution of computation.

Table management

When loading data back to the target database, you might want to optimize your data storage by using managed and unmanaged tables.

A **managed table** is a table that manages both the actual data and the metadata. In this case, a 'DROP TABLE' command removes both the metadata for the table as well as the data itself.

Unmanaged tables manage the metadata from a table, while the actual data is managed separately, often backed by a blob store like the Azure Blob or S3. Dropping an unmanaged table drops only the metadata associated with the table while the data itself remains in place.



Summary

Data transformation is one of the important aspects of the ETL process. It can be as simple as applying schema to incoming data to restructure it, or of a more complex nature where you perform customized transformation. The complexity of the transformation is based on the type of raw data and the requirements of specific scenario.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

By default, how are corrupt records dealt with using spark.read.json()

- They appear in a column called "_corrupt_record"
- They get deleted automatically
- They throw an exception and exit the read operation

Question 2

What is the recommended storage format to use with Spark?

- JSON
- XML
- Apache Parquet

Module Summary

Module Summary

In this module you have been introduced to Azure Databricks and how a Data Engineer works with it to enable Team Data Science projects. You learned the fundamentals of Azure Databricks and Apache Spark notebooks; how to provision the service and workspaces and learned how to perform data preparation task that can contribute to the data science project.

Learning objectives

In this module, you have learned:

- Azure Databricks
- How to provision Azure Databricks and Workspaces
- How to read Data using Azure Databricks
- How to perform transformations with Azure Databricks

Post Course Review

After the course, consider watching **this video with Yatharth Gupta that provides a deep dive into Azure Databricks deployment, networking and security⁴**.

Important

Remember

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

⁴ <https://youtu.be/lbtVZaUvESQ>

Answers

Question 1

Azure Databricks encapsulates which Apache Storage technology?

- Apache HDInsight
- Apache Hadoop
- Apache Spark

Explanation

Azure Databricks is an Apache Spark-based analytics platform optimized for the Microsoft Azure. Apache HDInsight does not exist, Azure HDInsight is a fully managed, full-spectrum, open-source analytics service for enterprises. HDInsight is a cloud service that makes it easy, fast, and cost-effective to process massive amounts of data. Apache Hadoop is the original open-source framework for distributed processing and analysis of big data sets on clusters.

Question 2

Which security features does Azure Databricks not support?

- Azure Active Directory
- Shared Access Keys
- Role-based access

Explanation

Shared Access Keys are a security feature used within Azure storage accounts. Azure Active Directory and Role-based access are supported security features in Azure Databricks.

Question 3

Which of the following Azure Databricks is used for support for R, SQL, Python, Scala, and Java?

- MLlib
- GraphX
- Spark Core API

Explanation

Spark Core API support for R, SQL, Python, Scala, and Java in Azure Databricks. MLlib is the Machine Learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives. GraphX provides graphs and graph computation for a broad scope of use cases from cognitive analytics to data exploration.

Question 1

Which Notebook format is used in Databricks?

- DBC
- .notebook
- .spark

Explanation

dbc file types are the supported Databricks notebook format. There is no no .notebook or .spark file format available.

Question 2

Which browsers are recommended for best use with Databricks Notebook?

- Chrome and Firefox
- Microsoft Edge and IE 11
- Safari and Microsoft Edge

Explanation

Chrome and Firefox are the recommended browsers by Databricks. Microsoft Edge and IE 11 are not recommended because of faulty rendering of iFrames, but Safari is also an acceptable browser.

Question 1

How do you connect your Spark cluster to the Azure Blob?

- By calling the .connect() function on the Spark Cluster.
- By mounting it
- By calling the .connect() function on the Azure Blob

Explanation

By mounting it. Mounts require Azure credentials such as SAS keys and give access to a virtually infinite store for your data. The .connect() function is not a valid method.

Question 2

How does Spark connect to databases like MySQL, Hive and other data stores?

- JDBC
- ODBC
- Using the REST API Layer

Explanation

JDBC. JDBC stands for Java Database Connectivity, and is a Java API for connecting to databases such as MySQL, Hive, and other data stores. ODBC is not an option and the REST API Layer is not available

Question 3

How do you specify parameters when reading data?

- Using .option() during your read allows you to pass key/value pairs specifying aspects of your read
- Using .parameter() during your read allows you to pass key/value pairs specifying aspects of your read
- Using .keys() during your read allows you to pass key/value pairs specifying aspects of your read

Explanation

Using .option() during your read allows you to pass key/value pairs specifying aspects of your read. For instance, options for reading CSV data include header, delimiter, and inferSchema.

Question 1

By default, how are corrupt records dealt with using spark.read.json()

- They appear in a column called "_corrupt_record"
- They get deleted automatically
- They throw an exception and exit the read operation

Explanation

They appear in a column called "_corrupt_record". They do not get deleted automatically or throw an exception and exit the read operation

Question 2

What is the recommended storage format to use with Spark?

- JSON
- XML
- Apache Parquet

Explanation

Apache Parquet. Apache Parquet is a highly optimized solution for data storage and is the recommended option for storage.

Module 4 Building Globally Distributed Database with Azure Cosmos DB

Module Introduction

Building Globally Distributed Databases with Cosmos DB

In this module, students will learn how to work with NoSQL data using Azure Cosmos DB. They will learn how to provision the service, and how they can load and interrogate data in the service using Visual Studio Code extensions, and the Azure Cosmos DB .NET Core SDK. They will also learn how to configure the availability options so that users are able to access the data from anywhere in the world.

Learning objectives

In this module, you will learn to:

- Create an Azure Cosmos DB database built to scale
- Insert and query data in your Azure Cosmos DB database
- Build a .NET Core app for Azure Cosmos DB in Visual Studio Code
- Distribute your data globally with Azure Cosmos DB

Create an Azure Cosmos DB database built to scale

Create an Azure Cosmos DB database built to scale

Databases need to scale to meet both the data velocity and volume demands created by global audiences.

Suppose you work for an online retailer and your database needs to scale to accommodate an expanding product catalog. You also have a growing global customer base, and your customers expect quick responses to their queries and immediate updates to product availability.

Your database should scale automatically to account for new products, and it should have high throughput and low latency to keep pace with your customers' expectations.

Learning objectives

In this module, you will:

- What is Cosmos DB
- Create an Azure Cosmos DB account
- What is a request units
- Choose a partition key
- Create a database and container for NoSQL data in Azure Cosmos DB

What is Cosmos DB

Azure Cosmos DB is Microsoft's globally distributed, multi-model database service. With the click of a button, Cosmos DB enables you to elastically and independently scale throughput and storage across any number of Azure's geographic regions. You can elastically scale throughput and storage, and take advantage of fast, single-digit-millisecond data access using your favorite API among SQL, MongoDB, Cassandra, Tables, or Gremlin.

Scalability

In traditional relational models, implementing distributed storage is non-trivial. In contrast, Cosmos DB offers global distribution capabilities out of the box. Specifically, Cosmos DB automatically replicates data to other regions and as such guarantees data consistency across various regions where the data is replicated. This is known as Turnkey global distribution.

Performance

Since Cosmos DB internally implements local caching of data and automatically indexes incoming data, the response times of read/write operations are typically in the order of 10s of milliseconds. Automatic indexing also reduces the operational burden of maintaining indexes.

Avalability

Azure Cosmos DB transparently replicates your data across all the Azure regions associated with your Cosmos account. The guarantees for high availability provided by Cosmos DB are 99.99% availability for a reads and writes of data in a single region and 99.999% availability for a reads and writes of data for multi-region accounts with multi region writes.

Programming models

Cosmos DB can be integrated into distributed applications and services using one of the various available programming models, thus reducing the ramp-up and onboarding time. These APIs provide programmatic access to create, query, and delete databases, collections, and documents.

- SQL API: Cosmos DB SQL API allows data access like relational models.
- Mongo API: Cosmos DB exposes Mongo API in order to facilitate Mongo DB users to easily ramp up on the offering.
- Gremlin API: Gremlin is a graph traversal language used to interact with graph databases. Cosmos DB exposes Gremlin API to store and interact with graph data stored in Cosmos DB.
- Cassandra API: Cosmos DB's Cassandra API enables interaction with data stored in Azure Cosmos DB using the Cassandra Query Language (CQL), Cassandra-based tools, and Cassandra client drivers. In addition, Spark connector is used to connect to Azure Cosmos DB Cassandra API.
- Table API: Applications that use Azure Tables can be migrated seamlessly to Cosmos DB by using Cosmos DB Table API.

Further, these APIs are exposed via .NET, Java, Node.js, Python, and Xamarin SDKs.

Create an Azure Cosmos DB account in the Azure portal

Your company has chosen Azure Cosmos DB to meet the demands of their expanding customer and product base. You have been tasked with creating the database.

The first step is to create an Azure Cosmos DB account.

What is an Azure Cosmos DB account

An Azure Cosmos DB account is an Azure resource that acts as an organizational entity for your databases. It connects your usage to your Azure subscription for billing purposes.

Each Azure Cosmos DB account is associated with one of the several data models Azure Cosmos DB supports, and you can create as many accounts as you need.

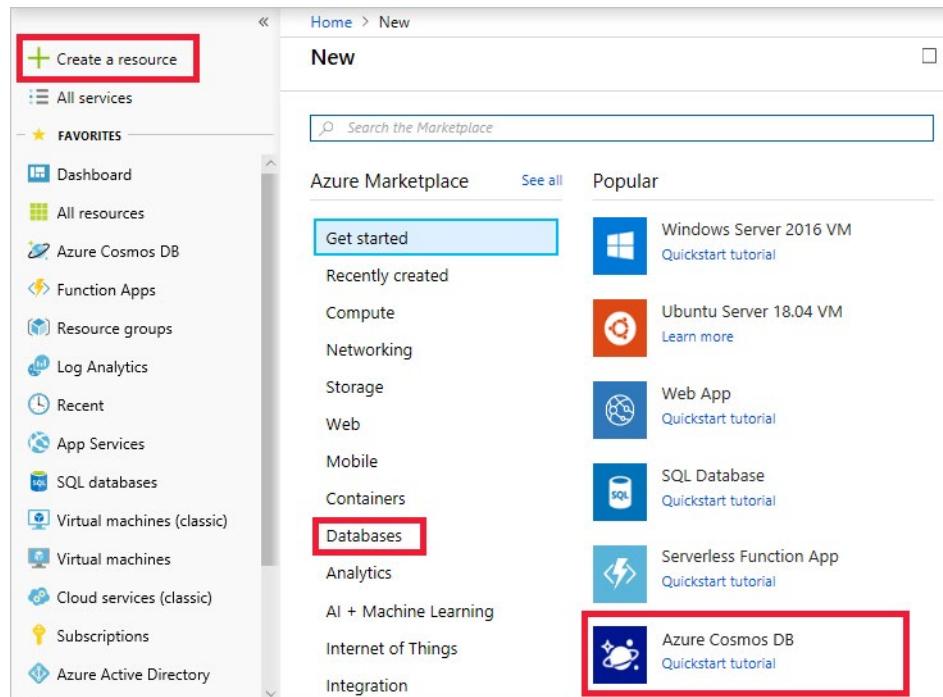
SQL API is the preferred data model if you are creating a new application. If you're working with graphs or tables, or migrating your MongoDB or Cassandra data to Azure, create additional accounts and select relevant data models.

When creating an account, choose an ID that is meaningful to you; it is how you identify your account. Further, create the account in the Azure region that's closest to your users to minimize latency between the datacenter and your users.

You can optionally set up virtual networks and geo-redundancy during account creation, but this can also be done later. In this module we will not enable those settings.

Creating an Azure Cosmos DB account in the portal

1. Sign into the **Azure portal for your azure subscription¹** using the same account you activated your subscription with.
2. Click **Create a resource > Databases > Azure Cosmos DB**.



3. On the **Create Azure Cosmos DB Account** page, enter the settings for the new Azure Cosmos DB account, including the location.

| Setting | Value | Description |
|----------------|---|---|
| Subscription | <i>Concierge Subscription</i> | Select the Concierge Subscription. If you do not see the Concierge Subscription listed, you have multiple tenants enabled on your subscription, and you need to change tenants. To do so, login again using the following portal link: Azure portal for sandbox (https://portal.azure.com/learn.docs.microsoft.com?azure-portal=true). |
| Resource Group | Use existing <rgn<[sandbox resource group name]</rgn> | Here you would either create a new resource group, or select an existing one in your subscription. |

¹ <https://portal.azure.com>

| Setting | Value | Description |
|---------------------|---|---|
| Account Name | <i>Enter a unique name</i> | <p>Enter a unique name to identify this Azure Cosmos DB account. Because <code>documents.azure.com</code> is appended to the ID that you provide to create your URI, use a unique but identifiable ID.</p> <p>The ID can contain only lower-case letters, numbers, and the hyphen (-) character, and it must contain 3 to 31 characters.</p> |
| API | SQL | <p>The API determines the type of account to create. Azure Cosmos DB provides five APIs to suit the needs of your application: SQL (document database), Gremlin (graph database), MongoDB (document database), Azure Table, and Cassandra, each of which currently requires a separate account.</p> <p>Select Core (SQL) because in this module you are creating a document database that is queryable using SQL syntax and accessible with the SQL API.</p> |
| Location | <i>Select the region closest to you from the list above</i> | Select the location where the database should be located. |
| Geo-Redundancy | Disable | This setting creates a replicated version of your database in a second (paired) region. Leave this set to disabled for now, as the database can be replicated later. |
| Multi-region Writes | Enable | This setting enables you to write to multiple regions at the same time. |

4. Click **Review + Create**.

The screenshot shows the Microsoft Azure portal interface for creating a new Azure Cosmos DB account. The top navigation bar includes icons for search, back, forward, notifications, and help. The main title is "Create Azure Cosmos DB Account". On the left, there's a sidebar with various icons representing different Azure services. The "Basics" tab is currently active. The "PROJECT DETAILS" section asks to select a subscription and resource group. The "INSTANCE DETAILS" section includes fields for account name, API (set to SQL), location, geo-redundancy (set to Enable), and multi-region writes (set to Enable). At the bottom, there are "Review + create" and "Next: Network" buttons.

5. After the settings are validated, click **Create** to create the account.
6. The account creation takes a few minutes. Wait for the portal to display the notification that the deployment succeeded and click the notification.

This screenshot shows a deployment notification window from the Azure portal. It displays a green checkmark icon, the text "Deployment succeeded" and "1:07 AM", and the message "Deployment 'Microsoft.DocumentDB' to resource group 'learning-module' was successful." At the bottom, there are two buttons: "Go to resource" and "Pin to dashboard".

7. In the notification window, click **Go to resource**.

This screenshot shows the confirmation page after creating an Azure Cosmos DB account. The title is "Congratulations! Your Azure Cosmos DB account was created". It includes a summary of the account details and a "View resource" button.

The portal displays the **Congratulations! Your Azure Cosmos DB account was created** page.

The screenshot shows the Azure Cosmos DB Quick Start page for the 'learning-module' account. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start (which is selected and highlighted in blue), Data Explorer, Settings, Replicate data globally, Default consistency, and Firewall and virtual networks. The main content area displays a message: 'Congratulations! Your Azure Cosmos DB account was created.' It also provides instructions to connect using a sample app and lists supported platforms: .NET, .NET Core, Xamarin, Java, Node.js, and Python. Two numbered steps are shown: Step 1, 'Add a collection', which explains that data is stored in collections and provides a 'Create 'Items' collection' button; Step 2, 'Download and run your .NET app', which instructs users to download a sample app once a collection is created.

Summary

You have created an Azure Cosmos DB account, which is the first step in creating an Azure Cosmos DB database. You selected appropriate settings for your data types and set the account location to minimize latency for your users.

What is a Request Unit

Next, let's consider the data for your database. Adequate throughput is important to ensure you can handle the volume of transactions for your business needs. Throughput requirements aren't always consistent. For example, you may be building a shopping website that needs to scale during sales or holidays. We'll start by learning about request units and how to estimate throughput requirements.

What is database throughput

Database throughput is the number of reads and writes that your database can perform in a single second.

To scale throughput strategically, you need to estimate your throughput needs by estimating the number of reads and writes you'll have to support at different times and for different document sizes. If you estimate correctly, you'll keep your users happy when demand spikes. If you estimate incorrectly, your requests can get rate-limited and operations will have to wait and retry, likely causing high latency and unhappy customers.

What is a request unit

Azure Cosmos DB measures throughput using something called a **request unit (RU)**. Request unit usage is measured per second, so the unit of measure is **request units per second (RU/s)**. You must reserve the number of RU/s you want Azure Cosmos DB to provision in advance, so it can handle the load you've estimated, and you can scale your RU/s up or down at any time to meet current demand.

Request unit basics

A single request unit, 1 RU, is equal to the approximate cost of performing a single GET request on a 1-KB document using a document's ID. Performing a GET by using a document's ID is an efficient means for retrieving a document, and thus the cost is small. Creating, replacing, or deleting the same item requires additional processing by the service, and therefore requires more request units.

The number of request units used for an operation changes depending on the document size, the number of properties in the document, the operation being performed, and some additional complex concepts such as consistency and indexing policy.

The following table shows the number of request units required for items of three different sizes (1 KB, 4 KB, and 64 KB) and at two different performance levels (500 reads/second + 100 writes/second and 500 reads/second + 500 writes/second). In this example, the data consistency is set to **Session**, and the indexing policy is set to **None**.

| Item size | Reads/second | Writes/second | Request units |
|-----------|--------------|---------------|---|
| 1 KB | 500 | 100 | $(500 * 1) + (100 * 5) = 1,000 \text{ RU/s}$ |
| 1 KB | 500 | 500 | $(500 * 1) + (500 * 5) = 3,000 \text{ RU/s}$ |
| 4 KB | 500 | 100 | $(500 * 1.3) + (100 * 7) = 1,350 \text{ RU/s}$ |
| 4 KB | 500 | 500 | $(500 * 1.3) + (500 * 7) = 4,150 \text{ RU/s}$ |
| 64 KB | 500 | 100 | $(500 * 10) + (100 * 48) = 9,800 \text{ RU/s}$ |
| 64 KB | 500 | 500 | $(500 * 10) + (500 * 48) = 29,000 \text{ RU/s}$ |

As you can see, the larger the item is, and the more reads and writes are required, the more request units you need to reserve. If you need to estimate the throughput needs of an application, the Azure Cosmos DB **Capacity Planner**² is an online tool that enables you to upload a sample JSON document and set the number of operations you need to complete per second. It then provides an estimated total to reserve.

Exceeding throughput limits

If you don't reserve enough request units, and you attempt to read or write more data than your provisioned throughput allows, your request will be rate-limited. When a request is rate-limited, the request has to be retried again after a specified interval. If you use the .NET SDK, the request will be retried automatically after waiting the amount of time specified in the retry-after header.

Creating an account built to scale

You can change the number of request units provisioned to a database at any time. So, during heavy volume periods, you can scale up to accommodate those high demands, and then reduce provisioned throughput during off peak times to reduce costs.

When you create an account, you can provision a minimum of 400 RU/s, or a maximum of 250,000 RU/s in the portal. If you need even more throughput, fill out a ticket in the Azure portal. Setting the initial throughput to 1000 RU/s is recommended for almost all accounts, as it is the minimum value in which

² <https://www.documentdb.com/capacityplanner>

your database will autoscale should you need more than 10 GB of storage. If you set the initial throughput to any value less than 1000 RU/s, your database will not be able to scale to larger than 10 GB unless you reprovision the database and provide a partition key. Partition keys enable quick lookup of data in Azure Cosmos DB and enable it to autoscale your database when needed. Partition keys are discussed a bit later in the module.

Summary

You now understand how to estimate and scope throughput for an Azure Cosmos DB using request units, and can make an appropriate selection when creating a new Azure Cosmos DB collection. Request units can be modified at any time, but setting them to 1000 RU/s when you create an account helps ensure your database is ready to scale later.

How to Choose a Partition Key

The online retailer that you work for plans to expand to a new geographical area soon. This move will increase your customer base and transaction volume. You must ensure that your database is equipped to handle expansion whenever required.

Having a partition strategy ensures that when your database needs to grow, it can do so easily and continue to perform efficient queries and transactions.

What is a partition strategy?

If you continue to add new data to a single server or a single partition, it will eventually run out of space. To prepare for this, you need a partitioning strategy to **scale out** instead of up. Scaling out is also called horizontal scaling, and it enables you to add more partitions to your database as your application needs them.

The partition and scale-out strategy in Azure Cosmos DB is driven by the partition key, which is a value set when you create a collection. Once the partition key is set, it cannot be changed without recreating the collection, so selecting the right partition key is an important decision to make early in your development process.

In this unit, you will learn how to choose a partition key that's right for your scenario and will take advantage of the autoscaling that Azure Cosmos DB can do for you.

What is a partition key?

A partition key is the value by which Azure organizes your data into logical divisions. In our online retail scenario, using the `userID` or `productId` value as the partition key is a good choice because it will be unique and likely used to lookup records. `userID` is a good choice, as your application frequently needs to retrieve the personalization settings, shopping cart, order history, and profile information for the user, just to name a few. `productId` is also a good choice, as your application needs to query inventory levels, shipping costs, color options, warehouse locations, and more.

A partition key should aim to distribute operations across the database. You want to distribute requests to avoid hot partitions. A hot partition is a single partition that receives many more requests than the others, which can create a throughput bottleneck. For example, for your e-commerce application, the current time would be a poor choice of partition key, because all the incoming data would go to a single partition key. `userID` or `productId` would be better, as all the users on your site would likely be adding and updating their shopping cart or profile information at about the same frequency, which distributes the reads and writes across all the user and product partitions.

The storage space for the data associated with each partition key cannot exceed 10 GB, which is the size of one physical partition in Azure Cosmos DB. So, if your single `userID` or `productId` record is going to be larger than 10 GB, think about using a composite key instead so that each record is smaller. An example of a composite key would be `userID-date`, which would look like **CustomerName-08072018**. This composite key approach would enable you to create a new partition for each day a user visited the site.

Best practices

When you're trying to determine the right partition key and the solution isn't obvious, here are a few tips to keep in mind.

- Don't be afraid of choosing a partition key that has a large range of values. The more values your partition key has, the more scalability you have.
- To determine the best partition key for a read-heavy workload, review the top three to five queries you plan on using. The value most frequently included in the WHERE clause is a good candidate for the partition key.
- For write-heavy workloads, you'll need to understand the transactional needs of your workload, because the partition key is the scope of multi-document transactions.

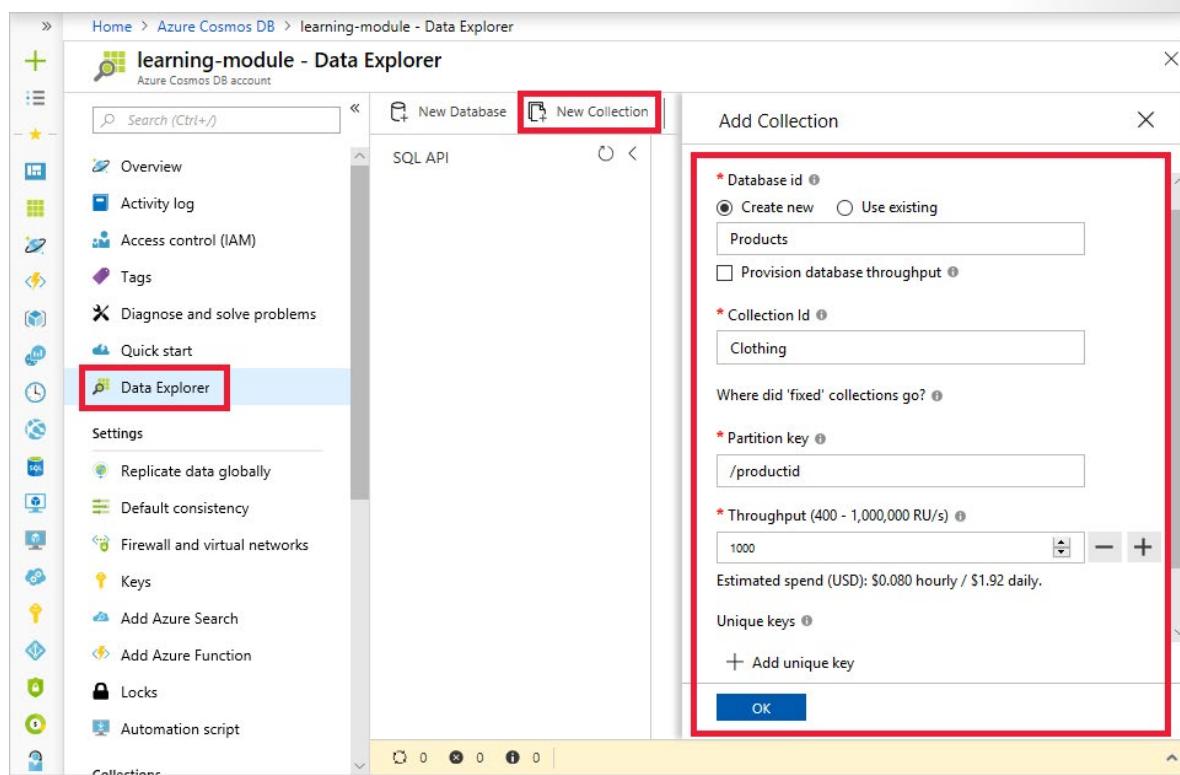
Creating a Database and a Collection in Cosmos DB

Now that you understand how request units are used to determine database throughput, and how the partition key creates the scale-out strategy for your database, you're ready to create your database and collection. You'll set your partition key and throughput values when you create your collection, so it's recommended that you understand those concepts before you create a database.

Creating your database and collection

1. In the Azure portal, select **Data Explorer** from your Cosmos DB resource and then click the **New Collection** button in the toolbar.

The **Add Collection** area is displayed on the far right. You may need to scroll right to see it.

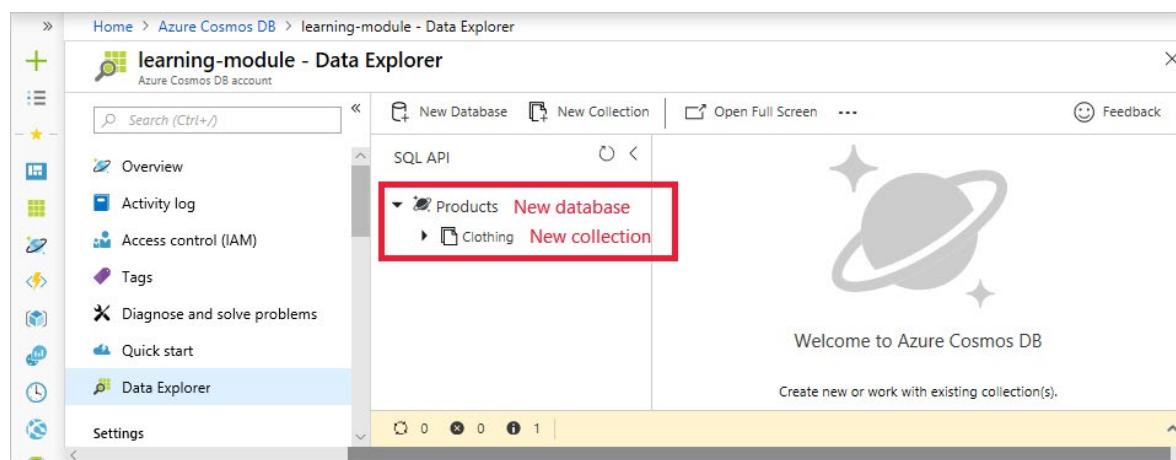


2. In the **Add collection** page, enter the settings for the new collection.

| Setting | Suggested value | Description |
|---------------|-----------------|---|
| Database id | Products | Enter <i>Products</i> as the name for the new database. Database names must be 1 to 255 characters in length, and must not contain /, \, #, ?, or a trailing space. |
| Collection id | Clothing | Enter <i>Clothing</i> as the name for your new collection. Collection ids have the same character requirements as database names. |
| Partition key | productId | productId is a good partition key for an online retail scenario, as so many queries are based around the product ID. |
| Throughput | 1000 RU | Change the throughput to 1000 request units per second (RU/s). 1000 is the minimum RU/s value you can set to enable automatic scaling. |

For now, don't check the **Provision database throughput** option, and don't add any unique keys to the collection.

3. Click **OK**. The Data Explorer displays the new database and collection.



Summary

In this unit, you used your knowledge of partition keys and request units to create a database and collection with throughput and scaling settings appropriate for your business needs.

Summary

This module has shown you how to create an Azure Cosmos DB account that you can use for real-world scenarios like online retail apps. By creating a database with a smart partition key, you'll be able to scale horizontally as your need for data storage grows. You've also learned about request unit needs for your application, and how to set them during account creation so that you can scale up your throughput later, when user demand increases.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

You want to ensure that there is 99.999% availability for the reading and writing of all your data. How can this be achieved?

- By configuring reads and writes of data in a single region.
- By configuring reads and writes of data for multi-region accounts with multi region writes.
- By configuring reads and writes of data for multi-region accounts with a single region writes.

Question 2

What are the three main advantages to using Cosmos DB?

- Cosmos DB offers global distribution capabilities out of the box.
- Cosmos DB provides a minimum of 99.99% availability.
- Cosmos DB response times of read/write operations are typically in the order of 10s of milliseconds.
- All of the above.

Insert and query data in your Azure Cosmos DB database

Insert and query data in your Azure Cosmos DB database

Adding data and querying data in your database should be straightforward.

Imagine you work for an online clothing retailer and you need to add inventory data to your e-commerce database in Azure Cosmos DB. Once the data is in the database, you can query it by using familiar SQL queries (yes, just like the ones you use in SQL Server) and perform complex operations by using stored procedures and user-defined functions (UDFs).

Azure Cosmos DB provides a Data Explorer in the Azure portal that you can use to perform all these operations: adding data, modifying data, and creating and running stored procedures. The Data Explorer can be used in the Azure portal or it can be undocked and used in a standalone web browser, providing additional space to work with your data.

Learning objectives

In this module, you will:

- Create product catalog documents in the Data Explorer
- Perform Azure Cosmos DB queries
- Create and run operations on your documents by using stored procedures
- Working with graph data

Setting up Cosmos DB Programmatically

The first thing we need to do is create an empty Azure Cosmos DB database and collection to work with. We want them to match the ones you created in the last module in this Learning Path: a database named **"Products"** and a collection named **"Clothing"**. Use the following instructions and the Azure Cloud Shell on the right side of the screen to recreate the database.

Create an Azure Cosmos DB account + database with the Azure CLI

Select a subscription

If you've been using Azure for a while, you might have multiple subscriptions available to you. This is often the case for developers who might have a subscription for Visual Studio, and another for corporate resources.

The Azure sandbox has already selected the Concierge Subscription for you in the Cloud Shell, and you can validate the subscription setting using these steps. Or, when you are working with your own subscription, you can use the following steps to switch subscriptions with the Azure CLI.

1. Start by listing the available subscriptions.

```
az account list --output table
```

If you're working with a Concierge Subscription, it should be the only one listed.

1. Next, if the default subscription isn't the one you want to use, you can change it with the `account set` command:

```
az account set --subscription "<subscription name>"
```

1. Get the Resource Group that has been created for you by the sandbox. If you are using your own subscription, skip this step and just supply a unique name you want to use in the `RESOURCE_GROUP` environment variable below. Take note of the Resource Group name. This is where we will create our database.

```
az group list --out table
```

Setup environment variables

1. Set a few environment variables so you don't have to type the common values each time. Start by setting a name for the Azure Cosmos DB account, for example `export NAME="mycosmosdbaccount"`. The field can contain only lowercase letters, numbers and the '-' character, and must be between 3 and 31 characters.

```
export NAME="<>Azure Cosmos DB account name>"
```

1. Set the resource group to use the existing sandbox resource group.

```
export RESOURCE_GROUP="<>rgn>[sandbox resource group name]</rgn>"
```

1. Select the region closest to you, and set the environment variable, such as `export LOCATION="EastUS"`.

```
export LOCATION="<>location>"
```

1. Set a variable for the database name. Name it "Products" so it matches the database we created in the last module.

```
export DB_NAME="Products"
```

Create a resource group in your subscription

When you are creating a Cosmos DB on your own subscription you will want to create a new resource group to hold all the related resources.

In your own subscription you would use the following command to create the Resource Group.

```
az group create --name <name> --location <location>
```

Create the Azure Cosmos DB account

1. Create the Azure Cosmos DB account with the `cosmosdb create` command. The command uses the following parameters and can be run with no modifications if you set the environment variables as recommended.

- `--name`: Unique name for the resource.
- `--kind`: Kind of database, use *GlobalDocumentDB*.
- `--resource-group`: The resource group.

```
az cosmosdb create --name $NAME --kind GlobalDocumentDB --resource-group  
$RESOURCE_GROUP
```

The command takes a few minutes to complete and the cloud shell displays the settings for the new account once it's deployed.

1. Create the `Products` database in the account.

```
az cosmosdb database create --name $NAME --db-name $DB_NAME --re-  
source-group $RESOURCE_GROUP
```

1. Finally, create the `Clothing` collection.

```
az cosmosdb collection create --collection-name "Clothing" --parti-  
tion-key-path "/productId" --throughput 1000 --name $NAME --db-name $DB_  
NAME --resource-group $RESOURCE_GROUP
```

Now that you have your Azure Cosmos DB account, database, and collection, let's go add some data!

Adding Data to Cosmos DB

Adding data to your Azure Cosmos DB database is simple. You open the Azure portal, navigate to your database, and use the Data Explorer to add JSON documents to the database. There are more advanced ways to add data, but we'll start here because the Data Explorer is a great tool to get you acquainted with the inner workings and functionality provided by Azure Cosmos DB.

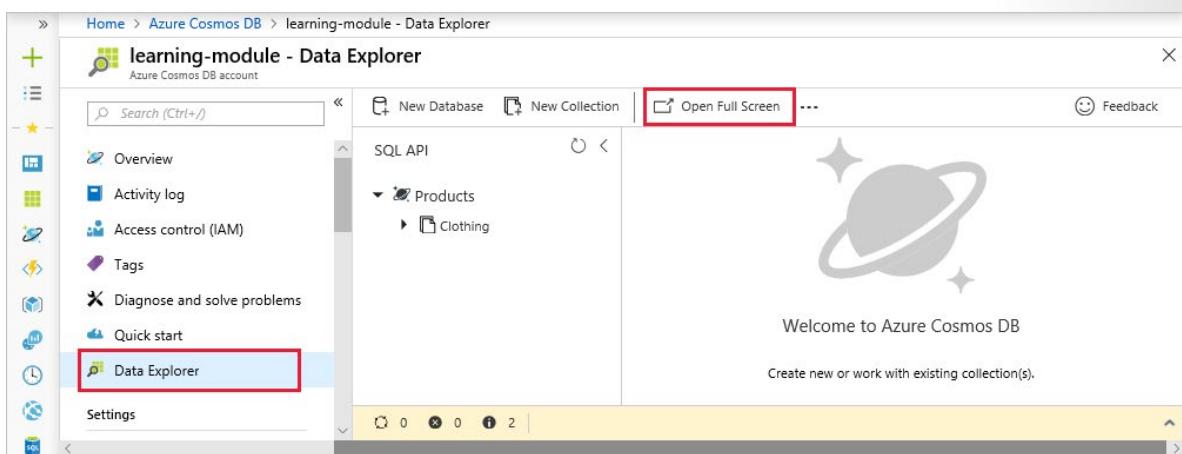
What is the Data Explorer

The Azure Cosmos DB Data Explorer is a tool included in the Azure portal that is used to manage data stored in an Azure Cosmos DB. It provides a UI for viewing and navigating data collections, as well as for editing documents within the database, querying data, and creating and running stored procedures.

Add data using the Data Explorer

1. Sign into the **Azure portal for sandbox**³ using the same account you activated the Azure Pass with.
2. Click **All services > Databases > Azure Cosmos DB**. Then select your account, click **Data Explorer**, and then click **Open Full Screen**.

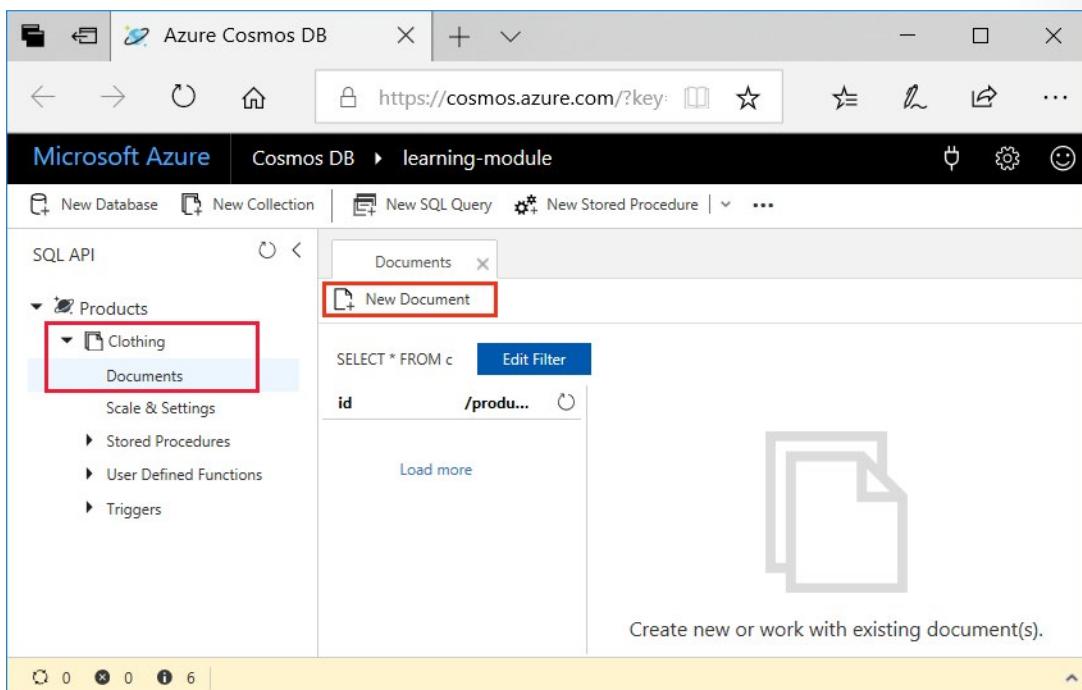
³ <https://portal.azure.com/learn.docs.microsoft.com?azure-portal=true>



3. In the **Open Full Screen** box, click **Open**.

The web browser displays the new full-screen Data Explorer, which gives you more space and a dedicated environment for working with your database.

3. To create a new JSON document, in the SQL API pane, expand **Clothing**, click **Documents**, then click **New Document**.

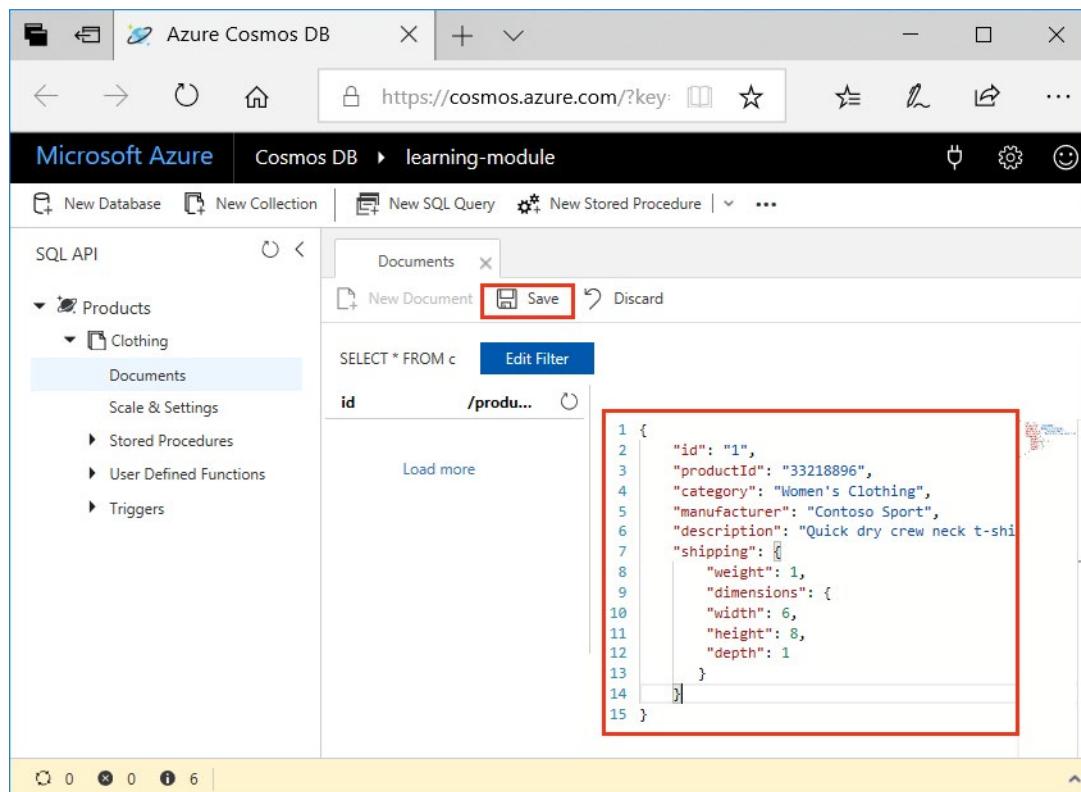


4. Now, add a document to the collection with the following structure. Just copy and paste the following code into the **Documents** tab, overwriting the current content:

```
{  
    "id": "1",  
    "productId": "33218896",  
    "category": "Women's Clothing",  
    "manufacturer": "Contoso Sport",  
    "description": "Quick dry crew neck t-shirt",  
}
```

```
        "price": "14.99",
        "shipping": {
            "weight": 1,
            "dimensions": {
                "width": 6,
                "height": 8,
                "depth": 1
            }
        }
    }
```

5. Once you've added the JSON to the **Documents** tab, click **Save**.



6. Create and save one more document clicking **New Document** again, and copying the following JSON object into Data Explorer and clicking **Save**.

```
{  
    "id": "2",  
    "productId": "33218897",  
    "category": "Women's Outerwear",  
    "manufacturer": "Contoso",  
    "description": "Black wool pea-coat",  
    "price": "49.99",  
    "shipping": {  
        "weight": 2,  
        "dimensions": {  
            "width": 8,  
            "height": 11,
```

```
        "depth": 3
    }
}
}
```

7. Confirm the documents have been saved by clicking **Documents** on the left-hand menu.

Data Explorer displays the two documents in the **Documents** tab.

In this unit, you added two documents, each representing a product in your product catalog, to your database by using the Data Explorer. The Data Explorer is a good way to create documents, modify documents, and get started with Azure Cosmos DB.

Querying SQL API

Using the two documents you added to the database as the target of our queries, let's walk through some query basics. Azure Cosmos DB uses SQL queries, just like SQL Server, to perform query operations. All properties are automatically indexed by default, so all data in the database is instantly available to query.

SQL query basics

Every SQL query consists of a SELECT clause and optional FROM and WHERE clauses. In addition, you can add other clauses like ORDER BY and JOIN to get the information you need.

A SQL query has the following format:

```
SELECT <select_list>
[FROM <optional_from_specification>]
[WHERE <optional_filter_condition>]
[ORDER BY <optional_sort_specification>]
[JOIN <optional_join_specification>]
```

SELECT clause

The SELECT clause determines the type of values that will be produced when the query is executed. A value of `SELECT *` indicates that the entire JSON document is returned.

Query

```
SELECT *
FROM Products p
WHERE p.id ="1"
```

Returns

```
[
{
    "id": "1",
    "productId": "33218896",
    "category": "Women's Clothing",
    "manufacturer": "Contoso Sport",
    "description": "Quick dry crew neck t-shirt",
```

```
        "price": "14.99",
        "shipping": {
            "weight": 1,
            "dimensions": {
                "width": 6,
                "height": 8,
                "depth": 1
            }
        },
        "_rid": "iAEeANrzNAAJAAAAAAA==",
        "_self": "dbs/iAEeAA==/colls/iAEeANrzNAA=/docs/iAEeANrzNAA-
JAAAAAAA==/",
        "_etag": "\"00003a02-0000-0000-0000-5b9208440000\"",
        "_attachments": "attachments/",
        "_ts": 1536297028
    }
]
```

Or, you can limit the output to include only certain properties by including a list of properties in the SELECT clause. In the following query, only the ID, manufacturer, and product description are returned.

Query

```
SELECT
    p.id,
    p.manufacturer,
    p.description
FROM Products p
WHERE p.id = "1"
```

Returns

```
[
    {
        "id": "1",
        "manufacturer": "Contoso Sport",
        "description": "Quick dry crew neck t-shirt"
    }
]
```

FROM clause

The FROM clause specifies the data source upon which the query operates. You can make the whole collection the source of the query or you can specify a subset of the collection instead. The FROM clause is optional unless the source is filtered or projected later in the query.

A query such as `SELECT * FROM Products` indicates that the entire Products collection is the source over which to enumerate the query.

A collection can be aliased, such as `SELECT p.id FROM Products AS p` or simply `SELECT p.id FROM Products p`, where `p` is the equivalent of `Products`. `AS` is an optional keyword to alias the identifier.

Once aliased, the original source cannot be bound. For example, `SELECT Products.id FROM Products p` is syntactically invalid because the identifier "Products" cannot be resolved anymore.

All properties that need to be referenced must be fully qualified. In the absence of strict schema adherence, this is enforced to avoid any ambiguous bindings. Therefore, `SELECT id FROM Products p` is syntactically invalid because the property `id` is not bound.

Subdocuments in a FROM clause

The source can also be reduced to a smaller subset. For instance, to enumerate only a subtree in each document, the subroot could then become the source, as shown in the following example:

Query

```
SELECT *
FROM Products.shipping
```

Results

```
[  
  {  
    "weight": 1,  
    "dimensions": {  
      "width": 6,  
      "height": 8,  
      "depth": 1  
    }  
  },  
  {  
    "weight": 2,  
    "dimensions": {  
      "width": 8,  
      "height": 11,  
      "depth": 3  
    }  
  }  
]
```

Although the above example used an array as the source, an object could also be used as the source, which is what's shown in the following example. Any valid JSON value (that's not undefined) that can be found in the source is considered for inclusion in the result of the query. If some products don't have a `shipping.weight` value, they are excluded in the query result.

Query

```
SELECT *
FROM Products.shipping.weight
```

Results

```
[  
  1,  
  2
```

```
]
```

WHERE clause

The WHERE clause specifies the conditions that the JSON documents provided by the source must satisfy in order to be included as part of the result. Any JSON document must evaluate the specified conditions to **true** to be considered for the result. The WHERE clause is optional.

The following query requests documents that contain an ID whose value is 1:

Query

```
SELECT p.description  
FROM Products p  
WHERE p.id = "1"
```

Results

```
[  
 {  
   "description": "Quick dry crew neck t-shirt"  
 }  
]
```

ORDER BY clause

The ORDER BY clause enables you to order the results in ascending or descending order.

The following ORDER BY query returns the price, description, and product ID for all products, ordered by price, in ascending order:

Query

```
SELECT p.price, p.description, p.productId  
FROM Products p  
ORDER BY p.price ASC
```

Results

```
[  
 {  
   "price": "14.99",  
   "description": "Quick dry crew neck t-shirt",  
   "productId": "33218896"  
 },  
 {  
   "price": "49.99",  
   "description": "Black wool pea-coat",  
   "productId": "33218897"  
 }  
]
```

JOIN clause

The JOIN clause enables you to perform inner joins with the document and the document subroots. So in the product database, for example, you can join the documents with the shipping data.

In the following query, the product IDs are returned for each product that has a shipping method. If you added a third product that didn't have a shipping property, the result would be the same because the third item would be excluded for not having a shipping property.

Query

```
SELECT p.productId  
FROM Products p  
JOIN p.shipping
```

Results

```
[  
  {  
    "productId": "33218896"  
  },  
  {  
    "productId": "33218897"  
  }  
]
```

Geospatial queries

Geospatial queries enable you to perform spatial queries using GeoJSON Points. Using the coordinates in the database, you can calculate the distance between two points and determine whether a Point, Polygon, or LineString is within another Point, Polygon, or LineString.

For product catalog data, this would enable your users to enter their location information and determine whether there were any stores within a 50-mile radius that have the item they're looking for.

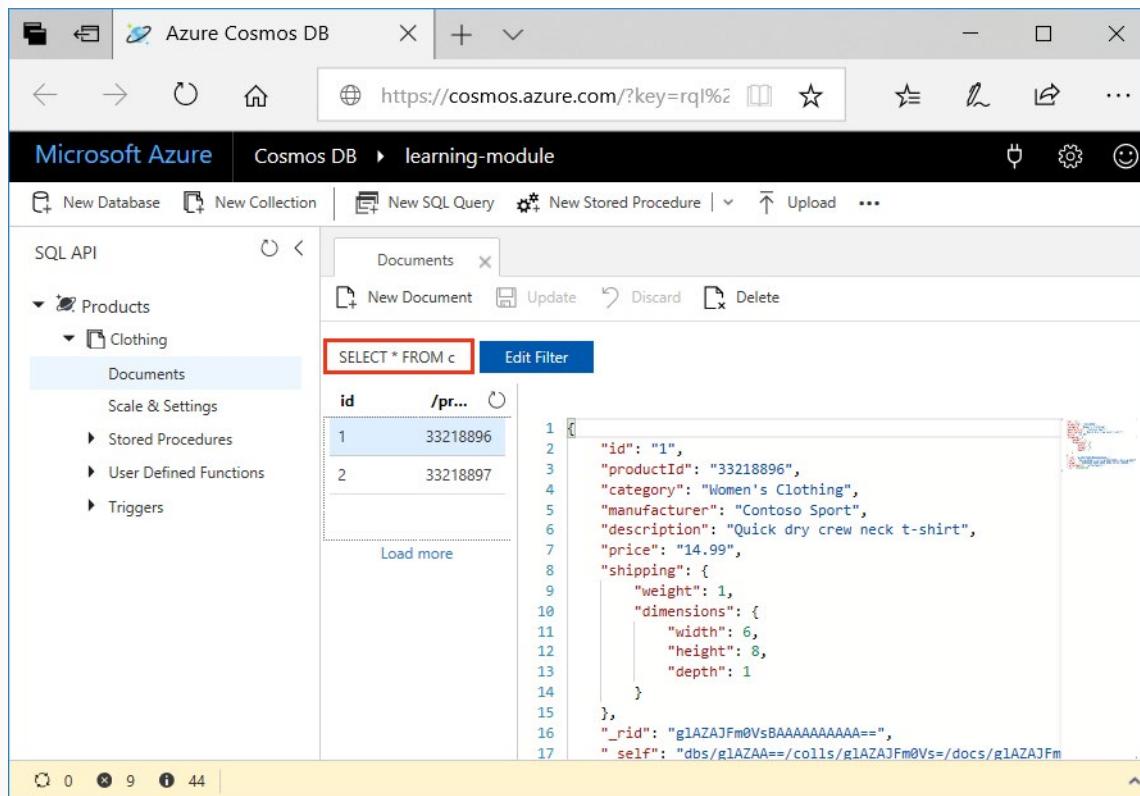
Summary

Being able to quickly query all your data after adding it to Azure Cosmos DB and using familiar SQL query techniques can help you and your customers to explore and gain insights into your stored data.

Run Queries in the Azure Portal

Now that you've learned about what kinds of queries you can create, let's use the Data Explorer in the Azure portal to retrieve and filter your product data.

In your Data Explorer window, note that by default, the query on the **Document** tab is set to `SELECT * FROM c` as shown in the following image. This default query retrieves and displays all documents in the collection.



Create a new query

1. In Data Explorer, click **New SQL Query**. Note that the default query on the new **Query 1** tab is again `SELECT * from c`, which will return all documents in the collection.
 2. Click **Execute Query**. This query returns all results in the database.

The screenshot shows the Azure Cosmos DB SQL API interface. The left sidebar shows a tree structure for a 'Products' collection under 'Clothing'. The main area has tabs for 'Documents' and 'Query 1'. The 'Query 1' tab is active, with a red box highlighting it. Below the tab is a button 'Execute Query' with a red box around it. The query editor contains the following SQL:

```
1 SELECT * FROM c
```

The results pane shows the output of the query:

```
{
  "id": "1",
  "productId": "33218896",
  "category": "Women's Clothing",
  "manufacturer": "Contoso Sport",
  "description": "Quick dry crew neck t-shirt",
  "price": "14.99",
  "shipping": {
    "weight": 1,
    "dimensions": {
      "width": 6,
      "height": 8,
      "depth": 1
    }
  },
  "_rid": "g1AZAJFm0VsBAAAAAAA==",
  "_self": "dbs/g1AZAA=/colls/g1AZAJFm0Vs=/docs/g1AZAJFm0VsBAAAAAAA==/",
  "_etag": "\"00008b00-0000-0000-5b734fc80000\"",
  "_attachments": "attachments/",
  "_ts": 1534283720
},
{
  "id": "2",
}
```

3. Now, let's run some of the queries discussed in the previous unit. On the query tab, delete `SELECT *` from `c`, copy and paste the following query, and then click **Execute Query**:

```
SELECT *
FROM Products p
WHERE p.id ="1"
```

The results return the product whose `productId` is 1.

The screenshot shows the Azure Cosmos DB SQL API interface. On the left, there's a navigation pane for the 'Products' collection under the 'Clothing' category. In the center, a query editor window titled 'Query 1' contains the following SQL code:

```
1 SELECT *
2 FROM Products p
3 WHERE p.id = "1"
```

A red box highlights the 'Execute Query' button. Below the query editor, the results pane shows the output of the query:

```
{
  "id": "1",
  "productId": "33218896",
  "category": "Women's Clothing",
  "manufacturer": "Contoso Sport",
  "description": "Quick dry crew neck t-shirt",
  "price": "14.99",
  "shipping": {
    "weight": 1,
    "dimensions": {
      "width": 6,
      "height": 8,
      "depth": 1
    }
  },
  "_rid": "g1AZAJFm0VsBAAAAAAA==",
  "_self": "dbs/g1AZAA==/colls/g1AZAJFm0Vs=/docs/g1AZAJFm0VsBAAAAAAA==",
  "_etag": "\\"0000b00-0000-0000-0000-5b734fc80000\\",
  "_attachments": "attachments/",
  "_ts": 1534283720
}
```

3. Delete the previous query, copy and paste the following query, and click **Execute Query**. This query returns the price, description, and product ID for all products, ordered by price, in ascending order.

```
SELECT p.price, p.description, p.productId
FROM Products p
ORDER BY p.price ASC
```

Summary

You have now completed some basic queries on your data in Azure Cosmos DB.

Running Complex Operations on your Data

Multiple documents in your database frequently need to be updated at the same time.

For your online retail application, when a user places an order and wants to use a coupon code, a credit, or a dividend (or all three at once), you need to query their account for those options, make updates to their account indicating they used them, update the order total, and process the order.

All of these actions need to happen at the same time, within a single transaction. If the user chooses to cancel the order, you want to roll back the changes and not modify their account information, so that their coupon codes, credits, and dividends are available for their next purchase.

The way to perform these transactions in Azure Cosmos DB is by using stored procedures and user-defined functions (UDFs). Stored procedures are the only way to ensure ACID (Atomicity, Consistency, Isolation, Durability) transactions because they are run on the server, and are thus referred to as server-side programming. UDFs are also stored on the server and are used during queries to perform computational logic on values or documents within the query.

In this module, you'll learn about stored procedures and UDFs, and then run some in the portal.

Stored procedure basics

Stored procedures perform complex transactions on documents and properties. Stored procedures are written in JavaScript and are stored in a collection on Azure Cosmos DB. By performing the stored procedures on the database engine and close to the data, you can improve performance over client-side programming.

Stored procedures are the only way to achieve atomic transactions within Azure Cosmos DB; the client-side SDKs do not support transactions.

Performing batch operations in stored procedures is also recommended because of the reduced need to create separate transactions.

Stored procedure example

The following sample is a simple HelloWorld stored procedure that gets the current context and sends a response that displays "Hello, World".

```
function helloWorld() {  
    var context = getContext();  
    var response = context.getResponse();  
  
    response.setBody("Hello, World");  
}
```

User-defined function basics

UDFs are used to extend the Azure Cosmos DB SQL query language grammar and implement custom business logic, such as calculations on properties and documents. UDFs can be called only from inside queries and, unlike stored procedures, they do not have access to the context object, so they cannot read or write documents.

In an online commerce scenario, a UDF could be used to determine the sales tax to apply to an order total or a percentage discount to apply to products or orders.

User-defined function example

The following sample creates a UDF to calculate tax on a product in a fictitious company based the product cost:

```
function producttax(price) {  
    if (price == undefined)
```

```
        throw 'no input';

var amount = parseFloat(price);

if (amount < 1000)
    return amount * 0.1;
else if (amount < 10000)
    return amount * 0.2;
else
    return amount * 0.4;
}
```

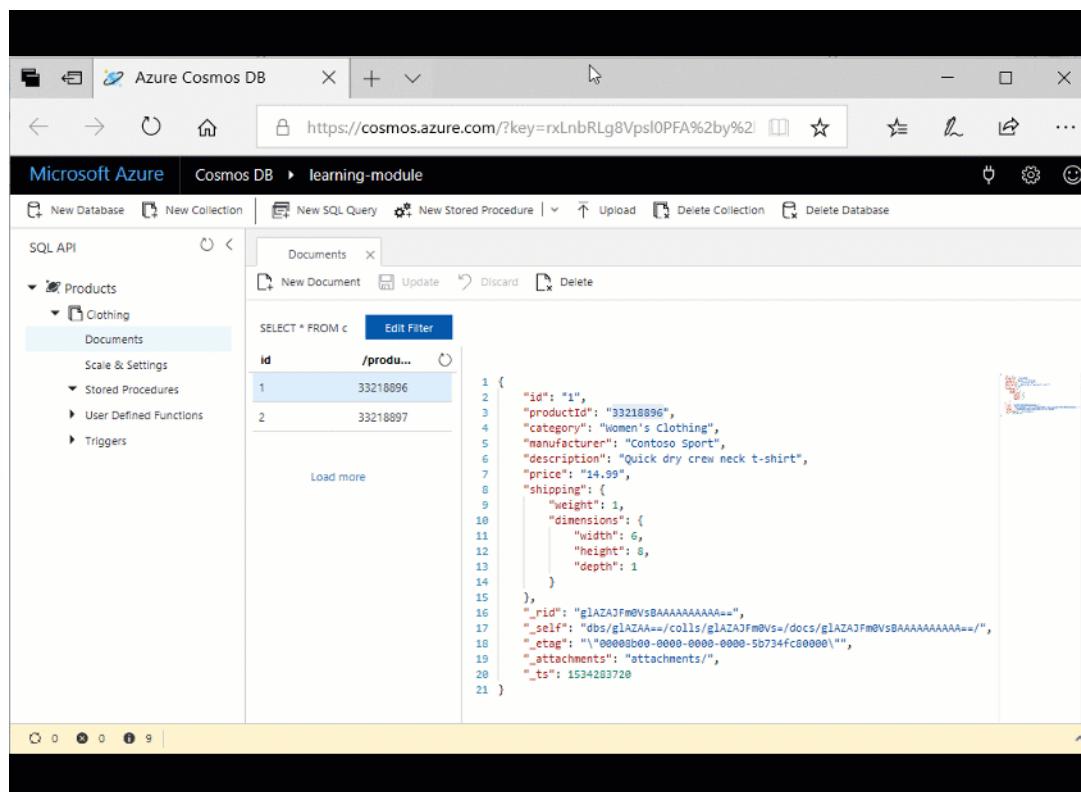
Create a stored procedure in the portal

Let's create a new stored procedure in the portal. The portal automatically populates a simple stored procedure that retrieves the first item in the collection, so we'll run this stored procedure first.

1. In the Data Explorer, click **New Stored Procedure**.

Data Explorer displays a new tab with a sample stored procedure.

2. In the **Stored Procedure Id** box, enter the name *sample*, click **Save**, and then click **Execute**.
3. In the **Input parameters** box, type the name of a partition key, *33218896*, and then click **Execute**. Note that stored procedures work within a single partition.



The **Result** pane displays the feed from the first document in the collection.

Create a stored procedure that creates documents

Now, let's create a stored procedure that creates documents.

1. In the Data Explorer, click **New Stored Procedure**. Name this stored procedure *createMyDocument*, copy and paste the following code into the **Stored Procedure Body** box, click **Save**, and then click **Execute**.

```
function createMyDocument() {
    var context = getContext();
    var collection = context.getCollection();

    var doc = {
        "id": "3",
        "productId": "33218898",
        "description": "Contoso microfleece zip-up jacket",
        "price": "44.99"
    };

    var accepted = collection.createDocument(collection.getSelfLink(),
        doc,
        function (err, documentCreated) {
            if (err) throw new Error('Error' + err.message);
            context.getResponse().setBody(documentCreated)
        });
    if (!accepted) return;
}
```

2. In the Input parameters box, enter a Partition Key Value of *33218898*, and then click **Execute**.

Data Explorer displays the newly created document in the Result area.

You can go back to the Documents tab, click the **Refresh** button and see the new document.

Create a user-defined function

Now, let's create a UDF in Data Explorer.

In the Data Explorer, click **New UDF**. You may need to click the down arrow next to **New Stored Procedure** to see **New UDF**. Copy the following code into the window, name the UDF *producttax*, and then click **Save**.

```
function producttax(price) {
    if (price == undefined)
        throw 'no input';

    var amount = parseFloat(price);

    if (amount < 1000)
        return amount * 0.1;
    else if (amount < 10000)
        return amount * 0.2;
    else
        return amount * 0.4;
```

```
}
```

Once you have defined the UDF, go to the **Query 1** tab and copy and paste the following query into the query area to run the UDF.

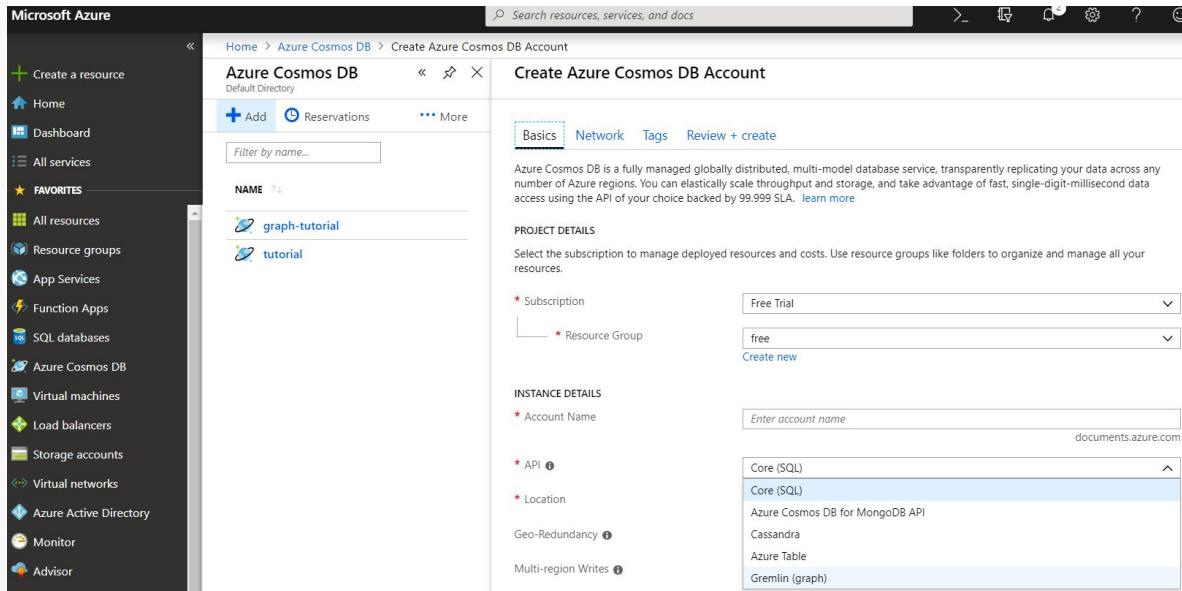
```
SELECT c.id, c.productId, c.price, udf.producttax(c.price) AS producttax  
FROM c
```

Working with Graph Data

Gremlin API for Azure Cosmos DB enables modeling and traversal of graph data. Gremlin is a graph traversal language that enables traversals across connected data such as social network data. Gremlin API for Cosmos DB is available for various language stacks, namely .NET, Java, Node.js, and Python. Gremlin API is different enough compared to other APIs, so it deserves some additional detail in order to demonstrate Graph API support in Cosmos DB. Follow the steps below to import graph data into Cosmos DB and query it.

Creating a graph database

The first step to import graph data into Cosmos DB is to create a database with Graph API support. Following the steps in the preceding unit called "Creating and configuring Azure Cosmos DB", create a Cosmos DB account/database with Gremlin API as the choice of API and then a container (called Graph in this case.) Note that we are creating a graph with no partitioning for this demo purposes. For production purposes, the graph should be created using partitioning keys.



and define the Graphid and size as follows:

Add Graph X

* Graph Id i

* Storage capacity i

* Throughput (400 - 10,000 RU/s) i
 - +

Estimated spend (USD): \$0.080 hourly / \$1.92 daily.
Choose unlimited storage capacity for more than 10,000 RU/s.

Adding Graph Data

We will use an example of two national parks in the state of California to demonstrate the graph API capabilities of Cosmos DB. Both Yosemite National Park and Joshua Tree National Park are in the state of California. The former has an alpine ecosystem and the latter has a desert ecosystem. Both are in high-altitude regions. The relationships in this scenario can be modeled by a graph.

We will use the Azure portal to create the graph and visualize it into Cosmos DB using the following steps.

- Download and install Python from [here](#)⁴
- Open command shell in elevated mode and install Azure Python SDK by typing the following command into the command shell: **pip install azure-cosmos**
- The installation output will look similar to the following:

⁴ <https://www.python.org/downloads/>

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python37-32\Scripts>pip install azure-cosmos
Collecting azure-cosmos
  Downloading https://files.pythonhosted.org/packages/f8/af/d1e22dcc61ba299f0786518e68f2287b73be027b16ee0120548e652a1aa8/azure-cosmos-3.0.2.tar.gz (119kB)
    100% |██████████| 122kB 1.9MB/s
Collecting six<1.6 (from azure-cosmos)
  Downloading https://files.pythonhosted.org/packages/7d/fb/00a976f728d0dfecfe898238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl
Collecting requests<2.10.0 (from azure-cosmos)
  Downloading https://files.pythonhosted.org/packages/14/e3/20f3d364d6c8e5d2353c72a67778eb189176f08e873c9900e10c0287b84b/requests-2.21.0-py2.py3-none-any.whl (57kB)
    100% |██████████| 61kB 3.2MB/s
Collecting idna<2.9,>=2.5 (from requests<2.10.0->azure-cosmos)
  Downloading https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl (58kB)
    100% |██████████| 61kB 1.5MB/s
Collecting urllib3<1.25,>=1.21.1 (from requests<2.10.0->azure-cosmos)
  Downloading https://files.pythonhosted.org/packages/62/00/ee1d7de624db8ba7090d1226aeefab96a2c71cd5cfa7629d6ad3f61b79e/urllib3-1.24.1-py2.py3-none-any.whl (118kB)
    100% |██████████| 122kB 2.5MB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests<2.10.0->azure-cosmos)
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl (133kB)
    100% |██████████| 143kB 3.5MB/s
Collecting certifi>=2017.4.17 (from requests<2.10.0->azure-cosmos)
  Downloading https://files.pythonhosted.org/packages/9f/e0/accfc1b56b57e9750eba272e24c4dddeac86852c2beb1236674d7887e8a/certifi-2018.11.29-py2.py3-none-any.whl (154kB)
    100% |██████████| 163kB 3.3MB/s
Installing collected packages: six, idna, urllib3, chardet, certifi, requests, azure-cosmos
  The script chardetect.exe is installed in 'c:\users\administrator\appdata\local\programs\python\python37-32\scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  Running setup.py install for azure-cosmos ... done
Successfully installed azure-cosmos-3.0.2 certifi-2018.11.29 chardet-3.0.4 idna-2.8 requests-2.21.0 six-1.12.0 urllib3-1.24.1
```

- Install the Gremlin Python package using the following command line: **pip install gremlinpython**
- Assuming that you have created a Cosmos DB database with graph API and created a collection, copy the following code into your favorite Python code editor, save it, and run it

```
from gremlin_python.driver import client, serializer
import sys, traceback

CLEANUP_GRAPH = "g.V().drop()"

INSERT_NATIONAL_PARK_VERTICES = [
    "g.addV('Park').property('id', 'p1').property('name', 'Yosemite').
    property('Feature', 'El Capitan')",
    "g.addV('Park').property('id', 'p2').property('name', 'Joshua Tree').
    property('Feature', 'Yucca Brevifolia')",
    "g.addV('State').property('id', 's1').property('name', 'California').
    property('Location', 'USA')",
    "g.addV('Ecosystem').property('id', 'e1').property('name', 'Alpine')",
    "g.addV('Ecosystem').property('id', 'e2').property('name', 'Desert')",
    "g.addV('Ecosystem').property('id', 'e3').property('name', 'High Altitude')"
]

INSERT_NATIONAL_PARK_EDGES = [
    "g.V('p1').addE('is in').to(g.V('s1'))",
    "g.V('p2').addE('is in').to(g.V('s1'))",
    "g.V('p1').addE('has ecosystem of').to(g.V('e1'))",
    "g.V('p2').addE('has ecosystem of').to(g.V('e2'))",
    "g.V('p1').addE('has ecosystem of').to(g.V('e3'))",
    "g.V('p2').addE('has ecosystem of').to(g.V('e3'))"
]

COUNT_VERTICES = "g.V().count()"

def cleanup_graph(client):
    print("\tExecuting Gremlin query:\n\t{0}".format(CLEANUP_GRAPH))
    callback = client.submitAsync(CLEANUP_GRAPH)
    if callback.result() is not None:
        print("\tCleaned up the graph!")
```

```
print("\n")

def create_national_park_vertices(client):
    for query in INSERT_NATIONAL_PARK_VERTICES:
        print("\tExecuting Gremlin query:\n\t{0}\n".format(query))
        callback = client.submitAsync(query)
        if callback.result() is not None:
            print("\tInserted this vertex:\n\t{0}\n".format(callback.result().one()))
        else:
            print("This query failed: {0}".format(query))
    print("\n")

def create_national_park_edges(client):
    for query in INSERT_NATIONAL_PARK_EDGES:
        print("\tExecuting Gremlin query:\n\t{0}\n".format(query))
        callback = client.submitAsync(query)
        if callback.result() is not None:
            print("\tInserted this edge:\n\t{0}\n".format(callback.result().one()))
        else:
            print("This query failed:\n\t{0}\n".format(query))
    print("\n")

def count_national_park_vertices(client):
    print("\tExecuting Gremlin query:\n\t{0}\n".format(COUNT_VERTICES))
    callback = client.submitAsync(COUNT_VERTICES)
    if callback.result() is not None:
        print("\tCount of vertices: {0}\n".format(callback.result().one()))
    else:
        print("This query failed: {0}\n".format(COUNT_VERTICES))
    print("\n")

try:
    client = Client('wss://GRAPH_DATABASE_NAME.gremlin.cosmosdb.azure.com:443/', 'g',
                    username="/dbs/graphdb/colls/national_parks",
                    password="COSMOS_DB_KEY",
                    message_serializer=serializer.GraphSONSerializerV2d0())
    )

    # Drop the entire Graph
    print("Cleaning up the National Parks graph...")
    cleanup_graph(client)

    # Insert all vertices
    print("Inserting National Parks vertices...")
```

```
create_national_park_vertices(client)

# Create edges between vertices
print("Inserting National Parks edges...")
create_national_park_edges(client)

# Count all vertices
print("Counting graph vertices...")
count_national_park_vertices(client)

except Exception as e:
    print('An exception was thrown: {}'.format(e))
    traceback.print_exc(file=sys.stdout)
    sys.exit(1)

print("\nDONE")
```

The output will look similar to the following:

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python37-32\python.exe
C:/Users/Administrator/PycharmProjects/graph_demo/graph_demo.py
Cleaning up the National Parks graph...
Executing Gremlin query:
g.V().drop()
Cleaned up the graph!

Inserting National Parks vertices...
Executing Gremlin query:
g.addV('Park').property('id', 'p1').property('name', 'Yosemite').proper-
ty('Feature', 'El Capitan')

Inserted this vertex:
[{"id": "p1", "label": "Park", "type": "vertex", "properties": {"name": [
        {"id": "03ccaf85-a7e7-4c4b-a464-58425db8e207", "value": "Yosemite"}],
        "Feature": [{"id": "21907904-1a2a-45a5-9c07-ba8a1585c3b7", "value": "El
        Capitan"}]}]

Executing Gremlin query:
g.addV('Park').property('id', 'p2').property('name', 'Joshua Tree').proper-
ty('Feature', 'Yucca Brevifolia')

Inserted this vertex:
[{"id": "p2", "label": "Park", "type": "vertex", "properties": {"name": [
        {"id": "fe8113d0-412e-4df3-af75-8c77f2d2d826", "value": "Joshua Tree"}],
        "Feature": [{"id": "1c89b021-246d-4737-a52a-55f8737309e4", "value": "Yucca
        Brevifolia"}]}]

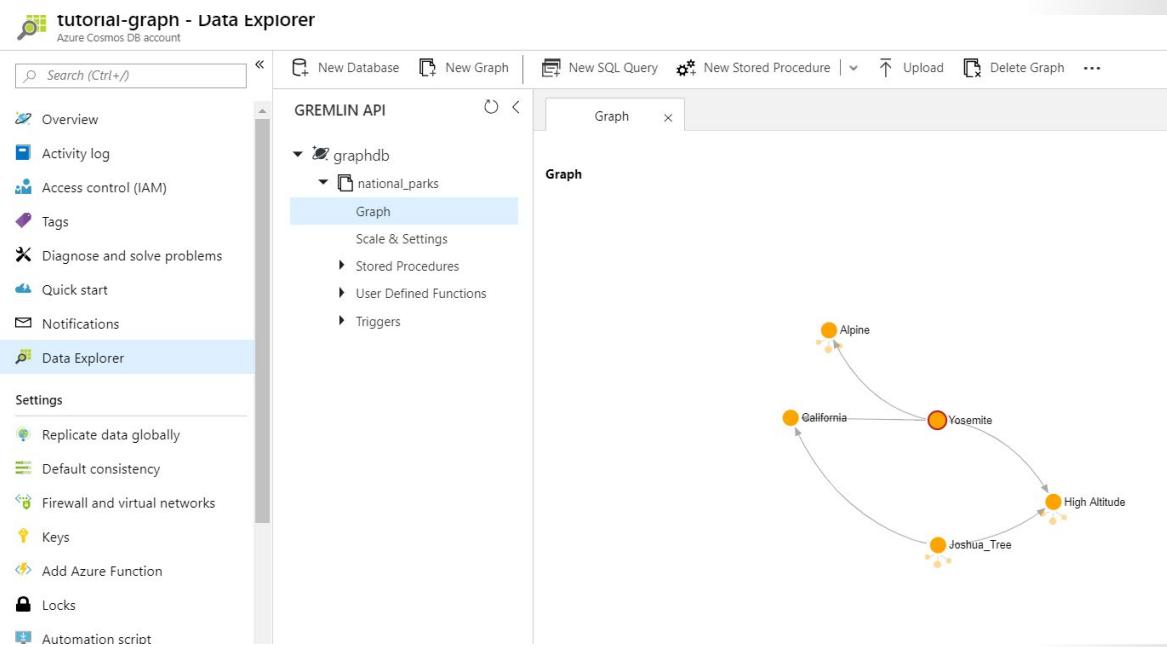
Executing Gremlin query:
g.addV('State').property('id', 's1').property('name', 'California').proper-
ty('Location', 'USA')
```

```
Inserted this vertex:  
[{"id": "s1", "label": "State", "type": "vertex", "properties": {"name": [{"id": "b250d636-a9d9-4886-8073-67b80c15d996", "value": "California"}], "Location": [{"id": "b9876b15-6fb1-4e1c-adb7-2bca4e8a962e", "value": "USA"}]}]  
  
Executing Gremlin query:  
g.addV('Ecosystem').property('id', 'e1').property('name', 'Alpine')  
  
Inserted this vertex:  
[{"id": "e1", "label": "Ecosystem", "type": "vertex", "properties": {"name": [{"id": "a59436b4-fc74-4038-83a6-398c29261018", "value": "Alpine"}]}]  
  
Executing Gremlin query:  
g.addV('Ecosystem').property('id', 'e2').property('name', 'Desert')  
  
Inserted this vertex:  
[{"id": "e2", "label": "Ecosystem", "type": "vertex", "properties": {"name": [{"id": "ad0047e1-14c2-49d8-9221-816585019ela", "value": "Desert"}]}]  
  
Executing Gremlin query:  
g.addV('Ecosystem').property('id', 'e3').property('name', 'High Altitude')  
  
Inserted this vertex:  
[{"id": "e3", "label": "Ecosystem", "type": "vertex", "properties": {"name": [{"id": "9612893c-ce9d-4408-bb4f-9efb105fd04d", "value": "High Altitude"}]}]  
  
  
Inserting National Parks edges...  
Executing Gremlin query:  
g.V('p1').addE('is in').to(g.V('s1'))  
  
Inserted this edge:  
[{"id": "891924f6-48e5-43d0-8b31-487f18d98786", "label": "is in", "type": "edge", "inVLabel": "State", "outVLabel": "Park", "inV": "s1", "outV": "p1"}]  
  
Executing Gremlin query:  
g.V('p2').addE('is in').to(g.V('s1'))  
  
Inserted this edge:  
[{"id": "df5413c3-3d37-44ab-bc77-fa8db04448cc", "label": "is in", "type": "edge", "inVLabel": "State", "outVLabel": "Park", "inV": "s1", "outV": "p2"}]  
  
Executing Gremlin query:  
g.V('p1').addE('has ecosystem of').to(g.V('e1'))
```

```
Inserted this edge:  
[{"id": "900edba5-4764-413e-a512-58c951de1a04", "label": "has ecosystem  
of", "type": "edge", "inVLabel": "Ecosystem", "outVLabel": "Park", "inV":  
'e1', "outV": 'p1'}]  
  
Executing Gremlin query:  
g.V('p2').addE('has ecosystem of').to(g.V('e2'))  
  
Inserted this edge:  
[{"id": "27023a87-034b-4b01-ab69-11c9535a3e35", "label": "has ecosystem  
of", "type": "edge", "inVLabel": "Ecosystem", "outVLabel": "Park", "inV":  
'e2', "outV": 'p2'}]  
  
Executing Gremlin query:  
g.V('p1').addE('has ecosystem of').to(g.V('e3'))  
  
Inserted this edge:  
[{"id": "72953c01-c9d7-4588-9823-3c74ed3baaf6", "label": "has ecosystem  
of", "type": "edge", "inVLabel": "Ecosystem", "outVLabel": "Park", "inV":  
'e3', "outV": 'p1'}]  
  
Executing Gremlin query:  
g.V('p2').addE('has ecosystem of').to(g.V('e3'))  
  
Inserted this edge:  
[{"id": "40a0f76f-3a76-4285-954b-0de990cbde7", "label": "has ecosystem  
of", "type": "edge", "inVLabel": "Ecosystem", "outVLabel": "Park", "inV":  
'e3', "outV": 'p2'}]  
  
Counting graph vertices...  
Executing Gremlin query:  
g.V().count()  
Count of vertices: [6]  
  
DONE  
Process finished with exit code 0
```

Visualizing the graph data

Now that we have the graph data in Cosmos DB, go to **Azure portal** and view it in the **Graph pane** of the **Cosmos DB database**.



Summary

In this module, you've created multiple documents, each representing a product in your product catalog. You've learned about the types of queries you can perform on your data and performed those queries, and you've learned how to perform complex operations and store them on the database engine by using stored procedures and UDFs.

These fundamentals give you insight into how Azure Cosmos DB works and how you can use it as the foundation for large database projects like e-commerce applications, which require the ability to quickly query and display data in a catalog, and perform operations and transactions on data in the database.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

You are a data engineer wanting to make the data that is currently stored in a Table Storage account located in the West US region available globally. Which Cosmos DB model should you migrate to?

- Gremlin API
- Cassandra API
- Table API
- Mongo DB API

Question 2

What type of data model provides a traversal language that enables connections and traversals across connected data?

- Gremlin API
- Cassandra API
- Table API
- Mongo DB API

Build a .NET Core app for Azure Cosmos DB in Visual Studio Code

Build a .NET Core app for Azure Cosmos DB in Visual Studio Code

Imagine you're managing storage for an online retailer. You need tools to create, update, and delete your user and product data.

In this module, you will build a .NET Core console application in Visual Studio Code to create, update, and delete user records, query your data, and perform stored procedures using C#.

Your data will be stored in Azure Cosmos DB, which has a convenient Visual Studio Code extension so you can easily create an Azure Cosmos DB account, database, and collection, and copy your connection string into the app without having to open the Azure portal.

Learning objectives

In this module, you will:

- Create an Azure Cosmos DB account, database, and collection in Visual Studio Code using the Azure Cosmos DB extension
- Create an application to store and query data in Azure Cosmos DB
- Use the Terminal in Visual Studio Code to quickly create a console application
- Add Azure Cosmos DB functionality with the help of the Azure Cosmos DB extension for Visual Studio Code

Creating Azure Cosmos DB in Visual Studio Code

The Azure Cosmos DB extension for Visual Studio Code simplifies account, database, and collection creation by enabling you to create resources using the command window.

In this unit you will install the Azure Cosmos DB extension for Visual Studio, and then use it to create an account, database, and collection.

Install the Azure Cosmos DB extension for Visual Studio

1. Go to the [Visual Studio Marketplace](#)⁵ and install the **Azure Cosmos DB** extension for Visual Studio Code.
2. When the extension tab loads in Visual Studio Code, click **Install**.
3. After installation is complete, click **Reload**.



Visual Studio Code displays the Azure icon on the left side of the screen after the extension is installed and reloaded.

⁵ <https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-cosmosdb&azure-portal=true>

Create an Azure Cosmos DB account in Visual Studio Code

1. In Visual Studio Code, sign in to Azure by clicking **View > Command Palette** and typing **Azure: Sign In**. You must have the **Azure Account⁶** extension installed to use Azure: Sign In.

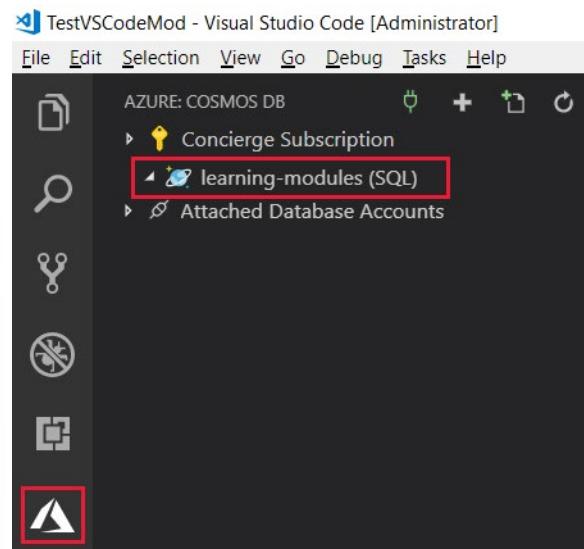
Follow the prompts to copy and paste the code provided in the web browser, which authenticates your Visual Studio Code session.



2. Click the **Azure** icon on the left menu, and then right-click **Subscription**, and click **Create Account**.
3. Click the **+** button to start creating a Cosmos DB account. You will be asked to select the subscription if you have more than one.
4. In the text box at the top of the screen, enter a unique name for your Azure Cosmos DB account, and then press enter. The account name can contain only lowercase letters, numbers and the '-' character, and must be between 3 and 31 characters.
5. Next, select **SQL (DocumentDB)**, then select **<rgn>[resource group name]</rgn>**, and then select a location.

The output tab in Visual Studio Code displays the progress of the account creation. It takes a few minutes to complete.

6. After the account is created, expand your Azure subscription in the **Azure: Cosmos DB** pane and the extension displays the new Azure Cosmos DB account. In the following image, the new account is named **learning-modules**.



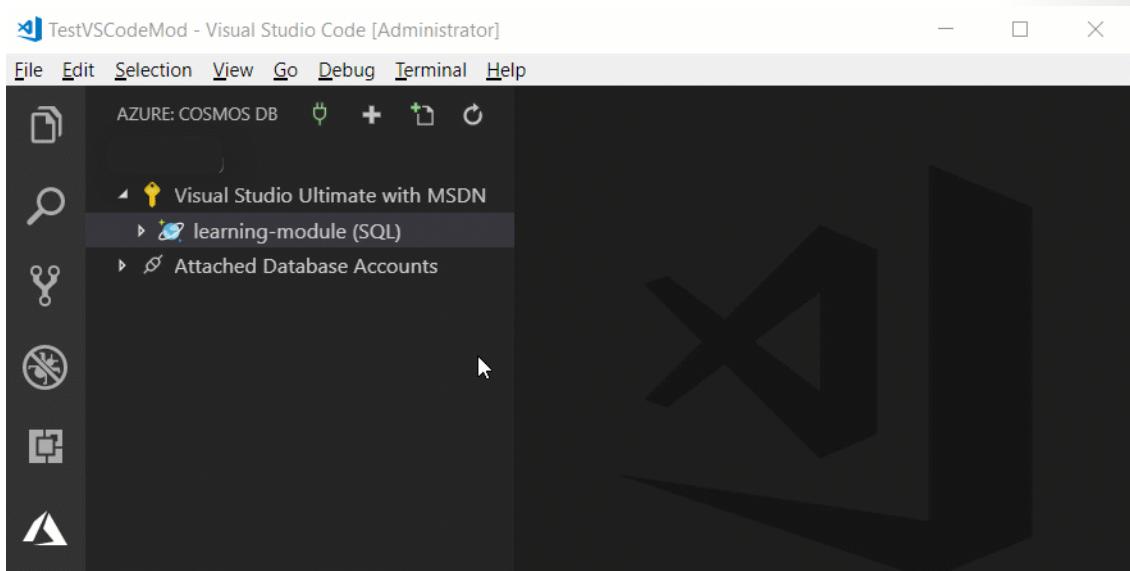
Create an Azure Cosmos DB database and collection in Visual Studio Code

Now let's create a new database and collection for your customers.

1. In the Azure: Cosmos DB pane, right-click your new account, and then click **Create Database**.
2. In the input palette at the top of the screen, type **Users** for the database name and press Enter.

⁶ <https://marketplace.visualstudio.com/items?itemName=ms-vscode.azure-account&azure-portal=true>

3. Enter `WebCustomers` for the collection name and press Enter.
4. Enter `userId` for the partition key and press Enter.
5. Finally, confirm `1000` for the initial throughput capacity and press Enter.
6. Expand the account in the **Azure: Cosmos DB** pane, and the new **Users** database and **WebCustomers** collection are displayed.



Now that you have your Azure Cosmos DB account, lets get to work in Visual Studio Code!

Create a Console App to connect to Cosmos DB

Visual Studio Code enables you to create a console application by using the integrated terminal and a few short commands.

In this unit, you will create a basic console app using the integrated terminal, retrieve your Azure Cosmos DB connection string from the extension, and then configure the connection from your application to Azure Cosmos DB.

Create a console app

1. In Visual Studio Code, select **File > Open Folder**.
2. Create a new folder named `learning-module` in the location of your choice, and then click **Select Folder**.
3. Ensure that file auto-save is enabled by clicking on the File menu and checking **Auto Save** if it is blank. You will be copying in several blocks of code, and this will ensure you are always operating against the latest edits of your files.
4. Open the integrated terminal from Visual Studio Code by selecting **View > Terminal** from the main menu.
5. In the terminal window, copy and paste the following command.

```
dotnet new console
```

This command creates a **Program.cs** file in your folder with a basic “Hello World” program already written, along with a C# project file named **learning-module.csproj**.

1. In the terminal window, copy and paste the following command to run the “Hello World” program.

```
dotnet run
```

The terminal window displays “Hello world!” as output.

Connect the app to Azure Cosmos DB

1. At the terminal prompt, copy and paste the following command block to install the required NuGet packages.

```
dotnet add package System.Net.Http  
dotnet add package System.Configuration  
dotnet add package System.Configuration.ConfigurationManager  
dotnet add package Microsoft.Azure.DocumentDB.Core  
dotnet add package Newtonsoft.Json  
dotnet add package System.Threading.Tasks  
dotnet add package System.Linq  
dotnet restore
```

1. At the top of the Explorer pane, click **Program.cs** to open the file.
2. Add the following using statements after `using System;`.

```
using System.Configuration;  
using System.Linq;  
using System.Threading.Tasks;  
using System.Net;  
using Microsoft.Azure.Documents;  
using Microsoft.Azure.Documents.Client;  
using Newtonsoft.Json;
```

If you get a message about adding required missing assets, click **Yes**.

1. Create a new file named App.config in the learning-module folder, and add the following code.

```
<?xml version="1.0" encoding="utf-8"?>  
  <configuration>  
    <appSettings>  
      <add key="accountEndpoint" value="<replace with your Endpoint URL>" />  
      <add key="accountKey" value="<replace with your Primary Key>" />  
    </appSettings>  
  </configuration>
```



1. Copy your connection string by clicking the Azure icon on the left, expanding your Concierge Subscription, right-clicking your new Azure Cosmos DB account, and then clicking **Copy Connection String**.

- Paste the connection string into the end of the App.config file, and then copy the **AccountEndpoint** portion from the connection string into the **accountEndpoint** value in App.config.

The **accountEndpoint** should look like the following:

```
<add key="accountEndpoint" value="https://<account-name>.documents.azure.com:443/" />
```

- Now copy the **AccountKey** value from the connection string into the **accountKey** value, and then delete the original connection string you copied in.

Your final App.config file looks similar to this.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <appSettings>
        <add key="accountEndpoint" value="https://my-account.documents.azure.com:443/" />
        <add key="accountKey" value="<account-key>" />
    </appSettings>
</configuration>
```

- At the terminal prompt, copy and paste the following command to run the program.

```
dotnet run
```

The program displays Hello World! in the terminal.

Create the DocumentClient

Now it's time to create an instance of the `DocumentClient`, which is the client-side representation of the Azure Cosmos DB service. This client is used to configure and execute requests against the service.

- In `Program.cs`, add the following to the beginning of the `Program` class.

```
private DocumentClient client;
```

- Add a new asynchronous task to create a new client, and check whether the **Users** database exists by adding the following method after the `Main` method.

```
private async Task BasicOperations()
{
    this.client = new DocumentClient(new Uri(ConfigurationManager.AppSettings["accountEndpoint"]),
        ConfigurationManager.AppSettings["accountKey"]);

    await this.client.CreateDatabaseIfNotExistsAsync(new Database { Id = "Users" });

    await this.client.CreateDocumentCollectionIfNotExistsAsync(UriFactory.CreateDatabaseUri("Users"),
        new DocumentCollection { Id = "WebCustomers" });

    Console.WriteLine("Database and collection validation complete");
```

```
}
```

1. In the integrated terminal, again, copy and paste the following command to run the program to ensure it runs.

```
dotnet run
```

1. Copy and paste the following code into the **Main** method, overwriting the current `Console.WriteLine("Hello World!");` line.

```
try
{
    Program p = new Program();
    p.BasicOperations().Wait();
}
catch (DocumentClientException de)
{
    Exception baseException = de.GetBaseException();
    Console.WriteLine("{0} error occurred: {1}, Message: {2}", de.StatusCode, de.Message, baseException.Message);
}
catch (Exception e)
{
    Exception baseException = e.GetBaseException();
    Console.WriteLine("Error: {0}, Message: {1}", e.Message, baseException.Message);
}
finally
{
    Console.WriteLine("End of demo, press any key to exit.");
    Console.ReadKey();
}
```

1. In the integrated terminal, again, type the following command to run the program to ensure it runs.

```
dotnet run
```

The console displays the following output.

```
Database and collection validation complete
End of demo, press any key to exit.
```

In this unit, you set up the groundwork for your Azure Cosmos DB application. You set up your development environment in Visual Studio Code, created a basic “Hello World” project, connected the project to the Azure Cosmos DB endpoint, and ensured your database and collection exist.

Working with documents programmatically

Once the connection to Azure Cosmos DB has been made, the next step is to create, read, replace, and delete the documents that are stored in the database. In this unit, you will create User documents in your `WebCustomer` collection. Then, you'll retrieve them by ID, replace them, and delete them.

Working with documents

Data is stored in JSON documents in Azure Cosmos DB. **Documents**⁷ can be created, retrieved, replaced, or deleted in the portal, as shown in the previous module, or programmatically, as described in this module. Azure Cosmos DB provides client-side SDKs for .NET, .NET Core, Java, Node.js, and Python, each of which supports these operations. In this module, we'll be using the .NET Core SDK to perform CRUD (create, retrieve, update, and delete) operations on the NoSQL data stored in Azure Cosmos DB.

The main operations for Azure Cosmos DB documents are part of the **DocumentClient**⁸ class:

- **CreateDocumentAsync**⁹
- **ReadDocumentAsync**¹⁰
- **ReplaceDocumentAsync**¹¹
- **UpsertDocumentAsync**¹²
- **DeleteDocumentAsync**¹³

Upsert performs a create or replace operation depending on whether the document already exists.

To perform any of these operations, you need to create a class that represents the object stored in the database. Because we're working with a database of users, you'll want to create a **User** class to store primary data such as their first name, last name, and user id (which is required, as that's the partition key to enable horizontal scaling) and classes for shipping preferences and order history.

Once you have those classes created to represent your users, you'll create new user documents for each instance, and then we'll perform some basic CRUD operations on the documents.

Create documents

1. First, create a **User** class that represents the objects to store in Azure Cosmos DB. We will also create **OrderHistory** and **ShippingPreference** classes that are used within **User**. Note that documents must have an **Id** property serialized as **id** in JSON.

To create these classes, copy and paste the following **User**, **OrderHistory**, and **ShippingPreference** classes underneath the **BasicOperations** method.

```
public class User
{
    [JsonProperty("id")]
    public string Id { get; set; }
    [JsonProperty("userId")]
    public string UserId { get; set; }
    [JsonProperty("lastName")]
    public string LastName { get; set; }
    [JsonProperty("firstName")]
    public string FirstName { get; set; }
    [JsonProperty("email")]
    public string Email { get; set; }
```

⁷ <https://docs.microsoft.com/azure/cosmos-db/sql-api-resources#documents>

⁸ <https://docs.microsoft.com/dotnet/api/microsoft.azure.documents.client.documentclient?view=azure-dotnet>

⁹ <https://docs.microsoft.com/dotnet/api/microsoft.azure.documents.client.documentclient.createdocumentasync?view=azure-dotnet>

¹⁰ <https://docs.microsoft.com/dotnet/api/microsoft.azure.documents.client.documentclient.readdocumentasync?view=azure-dotnet>

¹¹ <https://docs.microsoft.com/dotnet/api/microsoft.azure.documents.client.documentclient.replacedocumentasync?view=azure-dotnet>

¹² <https://docs.microsoft.com/dotnet/api/microsoft.azure.documents.client.documentclient.upsertdocumentasync?view=azure-dotnet>

¹³ <https://docs.microsoft.com/dotnet/api/microsoft.azure.documents.client.documentclient.deletedocumentasync?view=azure-dotnet>

```
[JsonProperty("dividend")]
public string Dividend { get; set; }
[JsonProperty("OrderHistory")]
public OrderHistory[] OrderHistory { get; set; }
[JsonProperty("ShippingPreference")]
public ShippingPreference[] ShippingPreference { get; set; }
[JsonProperty("CouponsUsed")]
public CouponsUsed[] Coupons { get; set; }
public override string ToString()
{
    return JsonConvert.SerializeObject(this);
}
}

public class OrderHistory
{
    public string OrderId { get; set; }
    public string DateShipped { get; set; }
    public string Total { get; set; }
}

public class ShippingPreference
{
    public int Priority { get; set; }
    public string AddressLine1 { get; set; }
    public string AddressLine2 { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public string ZipCode { get; set; }
    public string Country { get; set; }
}

public class CouponsUsed
{
    public string CouponCode { get; set; }
}

private void WriteToConsoleAndPromptToContinue(string format, params
object[] args)
{
    Console.WriteLine(format, args);
    Console.WriteLine("Press any key to continue ...");
    Console.ReadKey();
}
```

1. In the integrated terminal, type the following command to run the program to ensure it runs.

```
dotnet run
```

1. Now copy and paste the **CreateUserDocumentIfNotExists** task under the **WriteToConsoleAndPromptToContinue** method at the end of the Program.cs file.

```
private async Task CreateUserDocumentIfNotExists(string databaseName,
string collectionName, User user)
{
    try
    {
        await this.client.ReadDocumentAsync(UriFactory.CreateDocument-
tUri(databaseName, collectionName, user.Id), new RequestOptions { Parti-
tionKey = new PartitionKey(user.UserId) });
        this.WriteLineAndPrompt("User {0} already exists
in the database", user.Id);
    }
    catch (DocumentClientException de)
    {
        if (de.StatusCode == HttpStatusCode.NotFound)
        {
            await this.client.CreateDocumentAsync(UriFactory.CreateDoc-
umentCollectionUri(databaseName, collectionName), user);
            this.WriteLineAndPrompt("Created User {0}",
user.Id);
        }
        else
        {
            throw;
        }
    }
}
```

1. Then, return to the **BasicOperations** method and add the following to the end of that method.

```
User yanhe = new User
{
    Id = "1",
    UserId = "yanhe",
    LastName = "He",
    FirstName = "Yan",
    Email = "yanhe@contoso.com",
    OrderHistory = new OrderHistory[]
    {
        new OrderHistory {
            OrderId = "1000",
            DateShipped = "08/17/2018",
            Total = "52.49"
        }
    },
    ShippingPreference = new ShippingPreference[]
    {
        new ShippingPreference {
            Priority = 1,
            AddressLine1 = "90 W 8th St",
            AddressLine2 = null,
            City = "Austin",
            State = "TX",
            ZipCode = "78701"
        }
    }
}
```

```
        City = "New York",
        State = "NY",
        ZipCode = "10001",
        Country = "USA"
    }
},
};

await this.CreateUserDocumentIfNotExists("Users", "WebCustomers",
yanhe);

User nelapin = new User
{
    Id = "2",
    UserId = "nelapin",
    LastName = "Pindakova",
    FirstName = "Nela",
    Email = "nelapin@contoso.com",
    Dividend = "8.50",
    OrderHistory = new OrderHistory[]
    {
        new OrderHistory {
            OrderId = "1001",
            DateShipped = "08/17/2018",
            Total = "105.89"
        }
    },
    ShippingPreference = new ShippingPreference[]
    {
        new ShippingPreference {
            Priority = 1,
            AddressLine1 = "505 NW 5th St",
            City = "New York",
            State = "NY",
            ZipCode = "10001",
            Country = "USA"
        },
        new ShippingPreference {
            Priority = 2,
            AddressLine1 = "505 NW 5th St",
            City = "New York",
            State = "NY",
            ZipCode = "10001",
            Country = "USA"
        }
    },
    Coupons = new CouponsUsed[]
    {
        new CouponsUsed{
            CouponCode = "Fall2018"
        }
    }
}
```

```
        }

    await this.CreateUserDocumentIfNotExists("Users", "WebCustomers",
nelapin);
```

1. In the integrated terminal, again, type the following command to run the program.

```
dotnet run
```

The terminal will display output as the application creates each new user document. Press any key to complete the program.

```
Database and collection validation complete
Created User 1
Press any key to continue ...
Created User 2
Press any key to continue ...
End of demo, press any key to exit.
```

Read documents

1. To read documents from the database, copy in the following code and place after the **WriteToConsoleAndPromptToContinue** method in the Program.cs file.

```
private async Task ReadUserDocument(string databaseName, string collectionName, User user)
{
    try
    {
        await this.client.ReadDocumentAsync(UriFactory.CreateDocumentUri(databaseName, collectionName, user.Id), new RequestOptions { PartitionKey = new PartitionKey(user.UserId) });
        this.WriteLine("Read user {0}", user.Id);
    }
    catch (DocumentClientException de)
    {
        if (de.StatusCode == HttpStatusCode.NotFound)
        {
            this.WriteLine("User {0} not found", user.Id);
        }
        else
        {
            throw;
        }
    }
}
```

1. Copy and paste the following code to the end of the **BasicOperations** method, after the `await this.CreateUserDocumentIfNotExists("Users", "WebCustomers", nelapin);` line.
`await this.ReadUserDocument("Users", "WebCustomers", yanhe);`

1. In the integrated terminal, type the following command to run the program.

```
dotnet run
```

The terminal displays the following output, where the output "Read user 1" indicates the document was retrieved.

```
Database and collection validation complete
User 1 already exists in the database
Press any key to continue ...
User 2 already exists in the database
Press any key to continue ...
Read user 1
Press any key to continue ...
End of demo, press any key to exit.
```

Replace documents

Azure Cosmos DB supports replacing JSON documents. In this case, we'll update a user record to account for a change to their last name.

1. Copy and paste the **ReplaceUserDocument** method after the **ReadUserDocument** method in the Program.cs file.

```
private async Task ReplaceUserDocument(string databaseName, string
collectionName, User updatedUser)
{
    try
    {
        await this.client.ReplaceDocumentAsync(UriFactory.CreateDocu-
mentUri(databaseName, collectionName, updatedUser.Id), updatedUser, new
RequestOptions { PartitionKey = new PartitionKey(updatedUser.UserId) });
        this.WriteLineAndPromptToContinue("Replaced last name for
{0}", updatedUser.LastName);
    }
    catch (DocumentClientException de)
    {
        if (de.StatusCode == HttpStatusCode.NotFound)
        {
            this.WriteLineAndPromptToContinue("User {0} not found
for replacement", updatedUser.Id);
        }
        else
        {
            throw;
        }
    }
}
```

```
}
```

1. Copy and paste the following code to the end of the **BasicOperations** method, after the `await this.CreateUserDocumentIfNotExists("Users", "WebCustomers", nelapin);` line.

```
yanhe.LastName = "Suh";
await this.ReplaceUserDocument("Users", "WebCustomers", yanhe);
```

1. In the integrated terminal, run the following command.

```
dotnet run
```

The terminal displays the following output, where the output "Replaced last name for Suh" indicates the document was replaced.

```
Database and collection validation complete
User 1 already exists in the database
Press any key to continue ...
Replaced last name for Suh
Press any key to continue ...
User 2 already exists in the database
Press any key to continue ...
Read user 1
Press any key to continue ...
End of demo, press any key to exit.
```

Delete documents

1. Copy and paste the **DeleteUserDocument** method underneath your **ReplaceUserDocument** method.

```
private async Task DeleteUserDocument(string databaseName, string
collectionName, User deletedUser)
{
    try
    {
        await this.client.DeleteDocumentAsync(UriFactory.CreateDocument-
tUri(databaseName, collectionName, deletedUser.Id), new RequestOptions {
PartitionKey = new PartitionKey(deletedUser.UserId) });
        Console.WriteLine("Deleted user {0}", deletedUser.Id);
    }
    catch (DocumentClientException de)
    {
        if (de.StatusCode == HttpStatusCode.NotFound)
        {
            this.WriteToConsoleAndPromptToContinue("User {0} not found
for deletion", deletedUser.Id);
        }
        else
        {
            throw;
        }
    }
}
```

```
        }  
    }  
}
```

1. Copy and paste the following code in the end of the **BasicOperations** method.

```
await this.DeleteUserDocument("Users", "WebCustomers", yanhe);
```

1. In the integrated terminal, run the following command.

```
dotnet run
```

The terminal displays the following output, where the output "Deleted user 1" indicates the document was deleted.

```
Database and collection validation complete  
User 1 already exists in the database  
Press any key to continue ...  
Replaced last name for Suh  
Press any key to continue ...  
User 2 already exists in the database  
Press any key to continue ...  
Read user 1  
Press any key to continue ...  
Deleted user 1  
End of demo, press any key to exit.
```

In this unit you created, replaced, and deleted documents in your Azure Cosmos DB database.

Querying documents programmatically

Now that you've created documents in your application, let's query them from your application. Azure Cosmos DB uses SQL queries and LINQ queries. This unit focuses on running SQL queries and LINQ queries from your application, as opposed to the portal.

We'll use the user documents you've created for your online retailer application to test these queries.

LINQ query basics

LINQ is a .NET programming model that expresses computations as queries on streams of objects. You can create an **IQueryable** object that directly queries Azure Cosmos DB, which translates the LINQ query into a Cosmos DB query. The query is then passed to the Azure Cosmos DB server to retrieve a set of results in JSON format. The returned results are deserialized into a stream of .NET objects on the client side. Many developers prefer LINQ queries, as they provide a single consistent programming model across how they work with objects in application code and how they express query logic running in the database.

The following table shows how LINQ queries are translated into SQL.

| LINQ expression | SQL translation |
|---|---|
| input.Select(family => family.parents[0].familyName); | SELECT VALUE f.parents[0].familyName FROM Families f |

| | |
|--|---|
| input.Select(family => family.children[0].grade + c); // c is an int variable | SELECT VALUE f.children[0].grade + c FROM Families f |
| input.Select(family => new { name = family.children[0].familyName, grade = family.children[0].grade + 3}); | SELECT VALUE {"name":f.children[0].familyName, "grade": f.children[0].grade + 3 } FROM Families f |
| input.Where(family=> family.parents[0].familyName == "Smith"); | SELECT * FROM Families f WHERE f.parents[0].familyName = "Smith" |

Run SQL and LINQ queries

1. The following sample shows how a query could be performed in SQL, LINQ, or LINQ lambda from your .NET code. Copy the code and add it to the end of the Program.cs file.

```
private void ExecuteSimpleQuery(string databaseName, string collectionName)
{
    // Set some common query options
    FeedOptions queryOptions = new FeedOptions { MaxItemCount = -1,
        EnableCrossPartitionQuery = true };

    // Here we find nelapin via their LastName
    IQueryables<User> userQuery = this.client.CreateDocumentQuery<User>(
        UriFactory.CreateDocumentCollectionUri(databaseName, collectionName),
        queryOptions)
        .Where(u => u.LastName == "Pindakova");

    // The query is executed synchronously here, but can also be executed asynchronously via the IDocumentQuery<T> interface
    Console.WriteLine("Running LINQ query...");
    foreach (User user in userQuery)
    {
        Console.WriteLine("\tRead {0}", user);
    }

    // Now execute the same query via direct SQL
    IQueryables<User> userQueryInSql = this.client.CreateDocumentQuery<User>(
        UriFactory.CreateDocumentCollectionUri(databaseName, collectionName),
        "SELECT * FROM User WHERE User.lastName = 'Pindakova'", queryOptions);

    Console.WriteLine("Running direct SQL query...");
    foreach (User user in userQueryInSql)
    {
        Console.WriteLine("\tRead {0}", user);
    }

    Console.WriteLine("Press any key to continue ...");
    Console.ReadKey();
}
```

```
}
```

1. Copy and paste the following code to your **BasicOperations** method, before the `await this.`

```
    DeleteUserDocument("Users", "WebCustomers", yanhe); line.  
  
    this.ExecuteSimpleQuery("Users", "WebCustomers");
```

1. In the integrated terminal, run the following command.

```
dotnet run
```

The console displays the output of the LINQ and SQL queries.

In this unit you learned about LINQ queries, and then added a LINQ and SQL query to your application to retrieve user records.

Create a stored procedure in your application

Multiple documents in your database frequently need to be updated at the same time. This unit discusses how to create, register, and run stored procedures from your .NET console application.

Create a stored procedure

In this stored procedure, the **OrderId**, which contains a list of all the items in the order, is used to calculate an order total. The order total is calculated from the sum of the items in the order, less any dividends (credits) the customer has, and takes any coupon codes into account.

1. In Visual Studio Code, in the **Azure: Cosmos DB** tab, expand your Azure Cosmos DB account > **Users** > **WebCustomers** and then right-click **Stored Procedures** and then click **Create Stored Procedure**.
2. In the text box at the top of the screen, type `UpdateOrderTotal` and press Enter to give the stored procedure a name.
3. In the **Azure: Cosmos DB** tab, expand **Stored Procedures** and click **UpdateOrderTotal**.

By default, a stored procedure that retrieves the first item is provided.

1. To run this default stored procedure from your application, add the following code to the **Program.cs** file.

```
public async Task RunStoredProcedure(string databaseName, string collectionName, User user)  
{  
    await client.ExecuteStoredProcedureAsync<string>(UriFactory.Create-  
    StoredProcedureUri(databaseName, collectionName, "UpdateOrderTotal"), new  
    RequestOptions { PartitionKey = new PartitionKey(user.UserId) });  
    Console.WriteLine("Stored procedure complete");  
}
```

1. Now copy the following code and paste it before the `this.DeleteUserDocument("Users", "WebCustomers", yanhe);` line in the **BasicOperations** method.

```
await this.RunStoredProcedure("Users", "WebCustomers", yanhe);
```

1. In the integrated terminal, run the following command to run the sample with the stored procedure.

```
dotnet run
```

The console displays output indicating that the stored procedure was completed.

Summary

In this module you've created a .NET Core console application that creates, updates, and deletes user records, queries the users by using SQL and LINQ, and runs a stored procedure to query items in the database.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Suppose you are using Visual Studio Code to develop a .NET Core application that accesses Azure Cosmos DB. You need to include the connection string for your database in your application configuration. What is the most convenient way to get this information into your project?

- Directly from Visual Studio Code
- From the Azure portal
- Using the Azure CLI

Question 2

When working with Azure Cosmos DB's SQL API, which of these can be used to perform CRUD operations?

- LINQ
- Apache Cassandra client libraries
- Azure Table Storage libraries

Question 3

When working with the Azure Cosmos DB Client SDK's DocumentClient class, you use a NOSQL model. How would you use this class to change the FirstName field of a Person Document from 'Ann' to 'Fran'?

- Call UpdateDocumentAsync with FirstName=Fran
- Call UpsertDocumentAsync with an updated Person object
- Call ReplaceDocumentAsync with an updated Person object

Distribute your Data Globally with Azure Cosmos DB

Distribute your Data Globally with Azure Cosmos DB

Sending a packet across the world under ideal network conditions takes hundreds of milliseconds. The only way to cheat the speed of light is to place data locally, so that it has less distance to travel.

Imagine you work at an online clothing retailer, with large customer populations in Los Angeles, New York, and Tokyo. By replicating your data in multiple data-centers around the world, closest to where your users are, you ensure they all have the fastest access to your products, so you don't lose out to competitors.

Learning objective

In this module, you will:

- Learn about the benefits of writing and replicating data to multiple regions around the world
- Cosmos DB multi-master replication
- Cosmos DB failover management
- Change the consistency setting for your database

Cosmos DB Global Distribution

Providing your customers the fastest access to the products on your online clothing site is paramount to your customers, and your businesses success. By decreasing the distance data has to travel to your users, you can deliver more content faster. If your data is stored in Azure Cosmos DB, replicating your site's data to multiple regions around the world is a point and click operation.

In this unit you'll learn the benefits of global distribution and a natively multi-mastered database service, and then replicate your account into three additional regions.

Global distribution basics

Global distribution enables you to replicate data from one region into multiple Azure regions. You can add or remove regions in which your database is replicated at any time, and Azure Cosmos DB ensures that when you add an additional region, your data is available for operations within 30 minutes, assuming your data is 100 TBs or less.

There are two common scenarios for replicating data in two or more regions:

1. Delivering low-latency data access to end users no matter where they are located around the globe
2. Adding regional resiliency for business continuity and disaster recovery (BCDR)

To deliver low-latency access to customers, it is recommended that you replicate the data to regions closest to where your users are. For your online clothing company, you have customers in Los Angeles, New York, and Tokyo. Take a look at the [Azure regions¹⁴](#) page, and determine the closest regions to those sets of customers, as those are the locations you'll replicate users to.

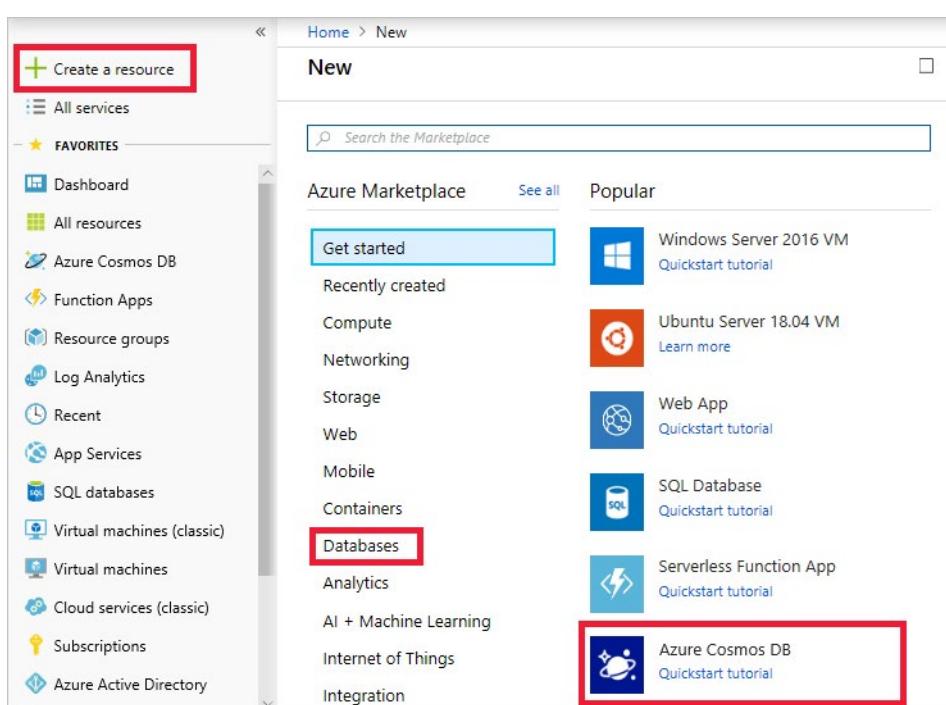
¹⁴ <https://azure.microsoft.com/global-infrastructure/regions/>

To provide a BCDR solution, it is recommended to add regions based on the region pairs described in the [Business continuity and disaster recovery \(BCDR\): Azure Paired Regions](#)¹⁵ article.

When a database is replicated, the throughput and storage are replicated equally as well. So if your original database had 10GB of storage, and throughput of 1,000 RU/s, and if you replicated that to three additional regions, each region would have 10GB of data and 1,000 RU/s of throughput. Because the storage and throughput is replicated in each region, the cost of replicating to a region is the same as the original region, so replicating to 3 additional regions, would cost approximately four times the original non-replicated database.

Creating an Azure Cosmos DB account in the portal

1. Sign in to the [Azure portal](#)¹⁶ using the same account you used to activate the sandbox.
2. Click **Create a resource > Databases > Azure Cosmos DB**.



3. On the **Create Azure Cosmos DB Account** page, enter the settings for the new Azure Cosmos DB account, including the location.

¹⁵ <https://azure.microsoft.com/documentation/articles/best-practices-availability-paired-regions/>

¹⁶ <https://portal.azure.com/learn.docs.microsoft.com?azure-portal=true>

| Setting | Value | Description |
|----------------|--|---|
| Subscription | <i>Concierge Subscription</i> | Select your Concierge Subscription. If you do not see the Concierge Subscription listed, you have multiple tenants enabled on your subscription, and you need to change tenants. To do so, login again using the following portal link: Azure portal for sandbox (https://portal.azure.com/learn/docs.microsoft.com?azure-portal=true). |
| Resource Group | Use existing <rgn>[sandbox resource group name]</rgn> | Select Use existing , and then enter <rgn>[sandbox resource group name]</rgn>. |
| Account Name | <i>Enter a unique name</i> | Enter a unique name to identify this Azure Cosmos DB account. Because <i>documents.azure.com</i> is appended to the ID that you provide to create your URL, use a unique but identifiable ID. The ID can contain only lowercase letters, numbers, and the hyphen (-) character, and it must contain 3 to 31 characters. |
| API | SQL | The API determines the type of account to create. Azure Cosmos DB provides five APIs to suit the needs of your application: SQL (document database), Gremlin (graph database), MongoDB (document database), Azure Table, and Cassandra, each of which currently requires a separate account. Select SQL because in this module you are creating a document database that is queryable using SQL syntax and accessible with the SQL API. |
| Location | <i>Select the region closest to you</i> | Select the region closest to you from the list of regions above. |

| Setting | Value | Description |
|---------------------|---------|---|
| Geo-Redundancy | Disable | This setting creates a replicated version of your database in a second (paired) region. Leave this set to disabled for now, as you will replicate the database later. |
| Multi-region Writes | Enable | This setting enables you to write to multiple regions at the same time. This setting can only be configured during account creation. |

4. Click **Review + Create**.

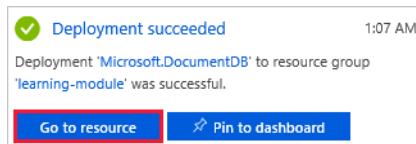
The screenshot shows the 'Create Azure Cosmos DB Account' wizard in the Microsoft Azure portal. The 'Basics' tab is selected. The 'PROJECT DETAILS' section shows a 'Subscription' dropdown set to 'Concierge Subscription' and a 'Resource Group' dropdown with a placeholder 'GUID for your resource group in your Concierge Subscription'. The 'INSTANCE DETAILS' section includes fields for 'Account Name' (placeholder 'enter a unique name'), 'API' (selected 'SQL'), 'Location' (placeholder 'choose the location closest to you'), 'Geo-Redundancy' (button 'Disable'), and 'Multi-region Writes' (button 'Disable'). At the bottom, there are 'Review + create', 'Previous', and 'Next: Network' buttons.

5. After the settings are validated, click **Create** to create the account.

6. The account creation takes a few minutes. Wait for the portal to display the notification that the deployment succeeded and click the notification.



7. In the notification window, click **Go to resource**.



The portal displays the **Congratulations! Your Azure Cosmos DB account was created** page.

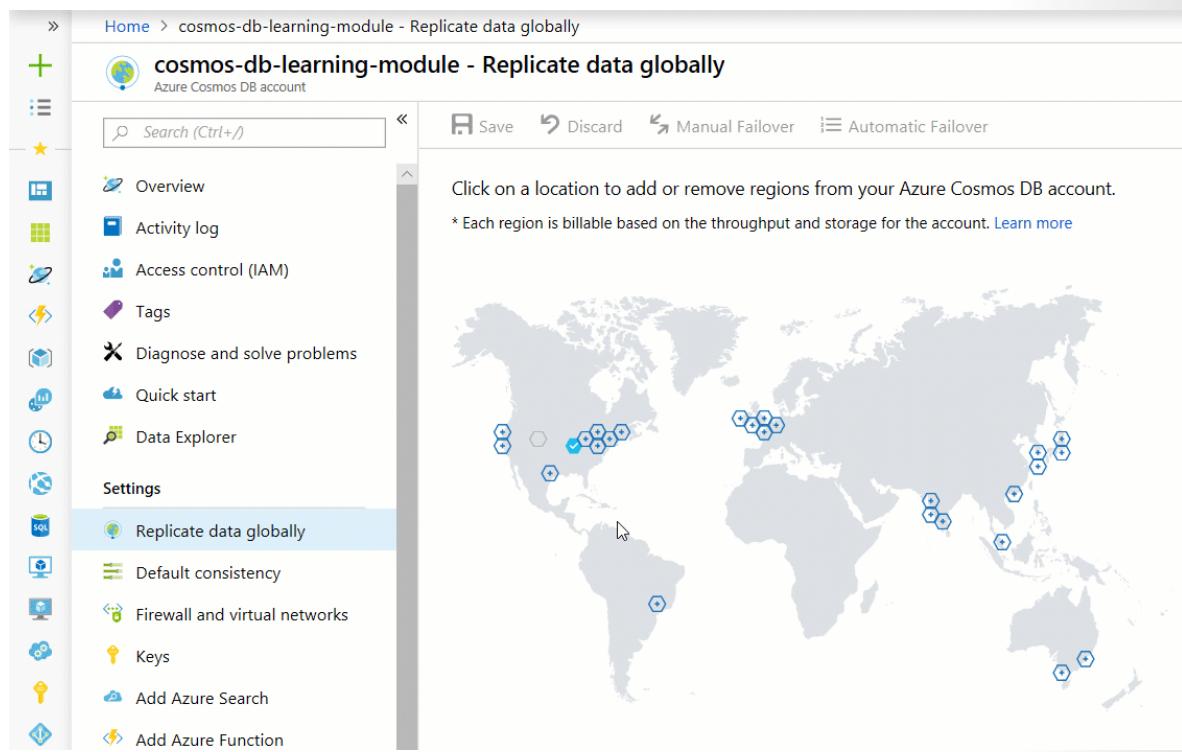
Replicate data in multiple regions

Let's now replicate your database closest to your global users in Los Angeles, New York, and Tokyo.

1. In the account page, click **Replicate data globally** from the menu.
2. In the **Replicate data globally** page, select the West US 2, East US, and Japan East regions, and then click **Save**.

If you don't see the map in the Azure portal, minimize the menus of the left side of the screen to display it.

The page will display an **Updating** message while the data is written to the new regions. Data in the new regions will be available within 30 minutes.



Summary

In this unit, you replicated your database to the regions of the world in which your users are most concentrated, providing them lower-latency access to the data on your site.

Cosmos DB Multi-Master Replication

Assuring a great user experience in an e-commerce scenario such as yours requires high availability and low latency.

By enabling multi-master support on the Azure Cosmos DB account you created, there are multiple benefits to both of these performance aspects, and all of the benefits are guaranteed by the financially backed service level agreements (SLAs) provided by Azure Cosmos DB.

What is multi-master support?

Multi-master support is an option that can be enabled on new Azure Cosmos DB accounts. Once the account is replicated in multiple regions, each region is a master region that equally participates in a write-anywhere model, also known as an active-active pattern.

Azure Cosmos DB regions operating as master regions in a multi-master configuration automatically work to converge data written to all replicas and ensure global consistency and data integrity.

With Azure Cosmos DB multi-master support, you can perform writes on any container in a write-enabled region world-wide. Written data is propagated to all other regions immediately.

What are the benefits of multi-master support?

The benefits of multi-master support are:

- Single-digit write latency – Multi-master accounts have an improved write latency of <10 ms for 99% of writes, up from <15 ms for non-multi-master accounts.
- 99.999% read-write availability - The write availability multi-master accounts increases to 99.999%, up from the 99.99% for non-multi-master accounts.
- Unlimited write scalability and throughput – With multi-master accounts, you can write to every region, providing unlimited write scalability and throughput to support billions of devices.
- Built-in conflict resolution – Multi-master accounts have three methods for resolving conflicts to ensure global data integrity and consistency.

Conflict resolution

With the addition of multi-master support comes the possibility of encountering conflicts for writes to different regions. Conflicts are rare in Azure Cosmos DB and can only occur when an item is simultaneously changed in multiple regions, before propagation between the regions has happened. Given the speed with which replication happens globally, you should not experience conflicts often, if at all. However, Azure Cosmos DB does provide conflict resolution modes that allow users to decide how to handle scenarios where the same record is updated simultaneously by different writers in two or more regions.

There are three conflict resolution modes offered by Azure Cosmos DB.

- **Last-Writer-Wins (LWW)**, in which conflicts are resolved based on the value of a user-defined integer property in the document. By default _ts is used to determine the last written document. Last-Writer-Wins is the default conflict handling mechanism.
- **Custom - User-defined function**, in which you can fully control conflict resolution by registering a User-defined function to the collection. A User-defined function is a special type of stored procedure with a specific signature. If the User-defined function fails or does not exist, Azure Cosmos DB will add all conflicts into the read-only conflicts feed they can be processed asynchronously.
- **Custom - Async**, in which Azure Cosmos DB excludes all conflicts from being committed and registers them in the read-only conflicts feed for deferred resolution by the user's application. The application can perform conflict resolution asynchronously and use any logic or refer to any external source, application, or service to resolve the conflict.

Cosmos DB Fail-Over Management

Once you've replicated your data in multiple regions, you can take advantage of the automated fail-over solutions Azure Cosmos DB provides. Automated fail-over is a feature that comes into play when there's a disaster or other event that takes one of your read or write regions offline, and it redirects requests from the offline region to the next most prioritized region.

For your online clothing site, which you just replicated into West US 2, East US, and Japan East, you can prioritize that if the East US goes offline, you can redirect reads to West US 2 instead of Japan East, to limit latency.

In this unit you'll learn about how fail-over works and set the priority for the regions in which your company's data has been replicated.

Failover basics

In the rare event of an Azure regional outage or data center outage, Azure Cosmos DB automatically triggers fail-overs of all Azure Cosmos DB accounts with a presence in the affected region.

What happens if a read region has an outage?

Azure Cosmos DB accounts with a read region in one of the affected regions are automatically disconnected from their write region and marked offline. The Cosmos DB SDKs implement a regional discovery protocol that allows them to automatically detect when a region is available and redirect read calls to the next available region in the preferred region list. If none of the regions in the preferred region list is available, calls automatically fall back to the current write region. No changes are required in your application code to handle regional fail-overs. During this entire process, consistency guarantees continue to be honored by Azure Cosmos DB.

Once the affected region recovers from the outage, all the affected Azure Cosmos DB accounts in the region are automatically recovered by the service. Azure Cosmos DB accounts that had a read region in the affected region will then automatically sync with current write region and turn online. The Azure Cosmos DB SDKs discover the availability of the new region and evaluate whether the region should be selected as the current read region based on the preferred region list configured by the application. Subsequent reads are redirected to the recovered region without requiring any changes to your application code.

What happens if a write region has an outage?

If the affected region is the current write region and automatic fail-over is enabled for the Azure Cosmos DB account, then the region is automatically marked as offline. Then, an alternative region is promoted as the write region for the affected Azure Cosmos DB account.

During automatic fail-overs, Azure Cosmos DB automatically chooses the next write region for a given Azure Cosmos DB account based on the specified priority order. Applications can use the `WriteEndpoint` property of `DocumentClient` class to detect the change in write region.

Once the affected region recovers from the outage, all the affected Cosmos DB accounts in the region are automatically recovered by the service.

Now let's modify the read region for your database.

Set read region priorities

1. In the Azure portal, on the **Replicate data globally** screen, click **Automatic Fail-over**. Automatic fail-over is only enabled if the database has already been replicated to more than one region.
2. On the **Automatic Fail-over** screen, change **Enable Automatic Fail-over** to **ON**.
3. In the **Read regions** section, click the left portion of the **East US** row, and then drag and drop at the top position.
4. Click the left portion of the **Japan East** row, and then drag and drop to the second position.
5. Click **OK**.

The screenshot shows the Azure portal interface for managing an Azure Cosmos DB account named "cosmos-db-learning-module". The left sidebar contains a navigation menu with icons for Home, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Data Explorer, and several settings-related items. The "Replicate data globally" option under Settings is currently selected and highlighted in blue. The main content area features a world map where various regions are marked with blue hexagonal icons, indicating where data is replicated. A tooltip above the map provides instructions on how to manage regions. A note below the map specifies that each region is billable based on throughput and storage.

Summary

In this unit, you've learned what automatic fail-over provides, how you can use it to protect against unforeseen outages, and how to modify read region priorities.

Cosmos DB Data Consistency Levels

Azure Cosmos DB allows developers to choose between five well-defined consistency models along the consistency spectrum – strong, bounded staleness, session, consistent prefix, and eventual. These consistency levels enable you to maximize the availability and performance of your database, depending on your needs. For cases when data must be processed in a specific order, strong consistency might be the right choice. Or for cases when data doesn't need to be immediately consistent, eventual consistency might be the right choice.

In this unit, you'll familiarize yourself with the available consistency levels in Azure Cosmos DB and determine the correct consistency level for your online clothing site.

Consistency basics

Each database account has a default consistency level, which determines the consistency of data within the account. At one end of the spectrum is Strong consistency, which offers a linearizability guarantee with the reads guaranteed to return the most recent version of an item. At the other end of the spectrum is Eventual consistency, which guarantees that in absence of any further writes, the replicas within the

group eventually converge. In the middle is Session consistency, which is the most popular because it guarantees monotonic reads, monotonic writes, and read your own writes (RYW) guarantees.



Guarantees about each consistency level are listed in the following table.

Consistency levels and guarantees

| Consistency Level | Guarantees |
|-------------------|--|
| Strong | Linearizability. Reads are guaranteed to return the most recent version of an item. |
| Bounded Staleness | Consistent Prefix. Reads lag behind writes by at most k prefixes or t interval. |
| Session | Consistent Prefix. Monotonic reads, monotonic writes, read-your-writes, write-follows-reads. |
| Consistent Prefix | Updates returned are some prefix of all the updates, with no gaps. |
| Eventual | Out of order reads. |

You can configure the default consistency level on your Azure Cosmos DB account (and later override the consistency on a specific read request) in the Azure portal. Internally, the default consistency level applies to data within the partition sets, which may span regions.

In Azure Cosmos DB, reads served at session, consistent prefix and eventual consistency are twice as cheap, in terms of request unit consumption, as reads with strong or bounded staleness consistency.

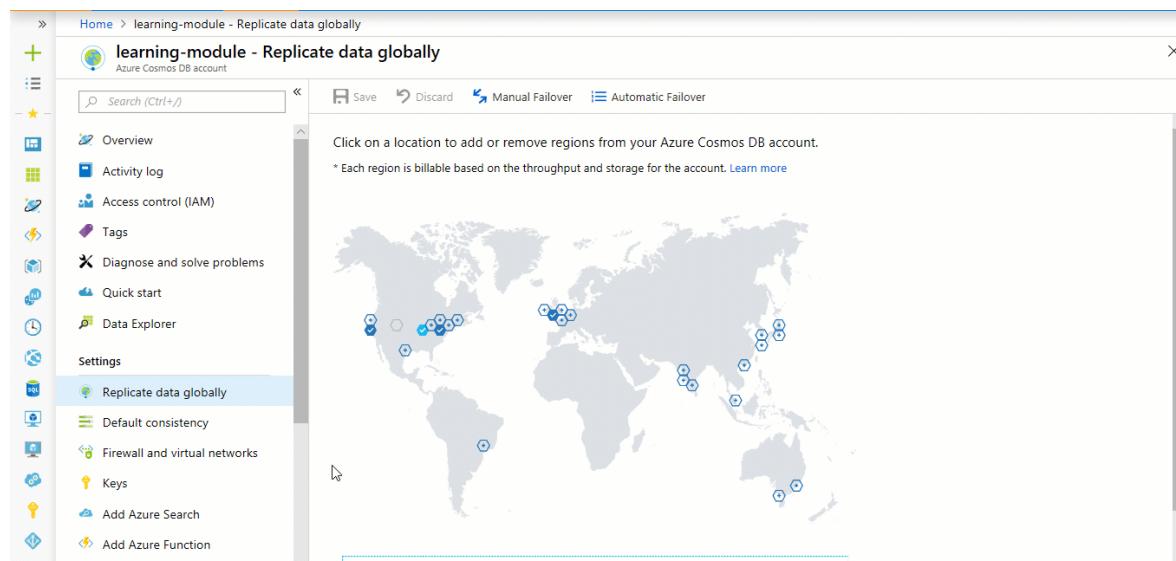
Use of consistency levels

About 73% of Azure Cosmos DB tenants use session consistency and 20% prefer bounded staleness. Approximately 3% of Azure Cosmos DB customers experiment with various consistency levels initially before settling on a specific consistency choice for their application. Only 2% of Azure Cosmos DB tenants override consistency levels on a per request basis.

Consistency levels in detail

To learn more about the consistency levels, review the music-note based consistency examples provided in the Azure portal, then review the information below about each level.

1. In the Azure portal, click **Default consistency**.
2. Click through each of the different consistency models and watch the musical examples. See how data is written to the different regions and how the choice of consistency impacts how the note data is written. Note that Strong is grayed out as it is only available for data written to a single region.



Let's learn more about the consistency levels. Think about how each of these consistency levels could work for your product and user data for your clothing retail site.

Strong consistency

- Strong consistency offers a **linearizability**¹⁷ guarantee with the reads guaranteed to return the most recent version of an item.
- Strong consistency guarantees that a write is only visible after it is committed durably by the majority quorum of replicas. A write is either synchronously committed durably by both the primary and the quorum of secondaries, or it is aborted. A read is always acknowledged by the majority read quorum, a client can never see an uncommitted or partial write and is always guaranteed to read the latest acknowledged write.
- Azure Cosmos DB accounts that are configured to use strong consistency cannot associate more than one Azure region with their Azure Cosmos DB account.
- The cost of a read operation (in terms of request units consumed) with strong consistency is higher than session and eventual, but the same as bounded staleness.

Bounded staleness consistency

- Bounded staleness consistency guarantees that the reads may lag behind writes by at most K versions or prefixes of an item or t time-interval.
- Therefore, when choosing bounded staleness, the "staleness" can be configured in two ways: number of versions K of the item by which the reads lag behind the writes, and the time interval t .
- Bounded staleness offers total global order except within the "staleness window." The monotonic read guarantees exist within a region both inside and outside the "staleness window."
- Bounded staleness provides a stronger consistency guarantee than session, consistent-prefix, or eventual consistency. For globally distributed applications, we recommend you use bounded staleness for scenarios where you would like to have strong consistency but also want 99.99% availability and low latency.

¹⁷ <https://aphyr.com/posts/313-strong-consistency-models>

- Azure Cosmos DB accounts that are configured with bounded staleness consistency can associate any number of Azure regions with their Azure Cosmos DB account.
- The cost of a read operation (in terms of RUs consumed) with bounded staleness is higher than session and eventual consistency, but the same as strong consistency.

Session consistency

- Unlike the global consistency models offered by strong and bounded staleness consistency levels, session consistency is scoped to a client session.
- Session consistency is ideal for all scenarios where a device or user session is involved since it guarantees monotonic reads, monotonic writes, and read your own writes (RYW) guarantees.
- Session consistency provides predictable consistency for a session, and maximum read throughput while offering the lowest latency writes and reads.
- Azure Cosmos DB accounts that are configured with session consistency can associate any number of Azure regions with their Azure Cosmos DB account.
- The cost of a read operation (in terms of RUs consumed) with session consistency level is less than strong and bounded staleness, but more than eventual consistency.

Consistent prefix consistency

- Consistent prefix guarantees that in absence of any further writes, the replicas within the group eventually converge.
- Consistent prefix guarantees that reads never see out of order writes. If writes were performed in the order A, B, C, then a client sees either A, A, B, or A, B, C, but never out of order like A, C or B, A, C.
- Azure Cosmos DB accounts that are configured with consistent prefix consistency can associate any number of Azure regions with their Azure Cosmos DB account.

Eventual consistency

- Eventual consistency guarantees that in absence of any further writes, the replicas within the group eventually converge.
- Eventual consistency is the weakest form of consistency where a client may get the values that are older than the ones it had seen before.
- Eventual consistency provides the weakest read consistency but offers the lowest latency for both reads and writes.
- Azure Cosmos DB accounts that are configured with eventual consistency can associate any number of Azure regions with their Azure Cosmos DB account.
- The cost of a read operation (in terms of RUs consumed) with the eventual consistency level is the lowest of all the Azure Cosmos DB consistency levels.

Summary

In this unit, you've learned how consistency levels can be used to maximize high-availability and minimize latency.

Summary

This module has shown you how to globally distribute data, modify the read priorities for your database, set up multiple regions to write to, and determine the right consistency level for your online clothing site.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

You want to maximize the data integrity of data that is stored in a Cosmos DB. Which consistency level should you choose?

- Session
- Strong
- Eventual

Module Summary

Module Summary

In this module, you have learned how to work with NoSQL data using Azure Cosmos DB. You learned how to provision the service, and how you can load and interrogate data in the service using Visual Studio Code extensions, and the Azure Cosmos DB .NET Core SDK. You also learned how to configure the availability options so that users are able to access the data from anywhere in the world.

Learning objectives

In this module, you have learned how to:

- Create an Azure Cosmos DB database built to scale.
- Insert and query data in your Azure Cosmos DB database.
- Build a .NET Core app for Azure Cosmos DB in Visual Studio Code.
- Distribute your data globally with Azure Cosmos DB.

Post Course Review

After the course, consider visiting [the Microsoft Cosmos DB Whitepapers site¹⁸](#) to explore Azure Cosmos DB concepts at a deeper level.

Important

Remember

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

¹⁸ <https://docs.microsoft.com/en-us/azure/cosmos-db/whitepapers>

Answers

Question 1

You want to ensure that there is 99.999% availability for the reading and writing of all your data. How can this be achieved?

- By configuring reads and writes of data in a single region.
- By configuring reads and writes of data for multi-region accounts with multi region writes.
- By configuring reads and writes of data for multi-region accounts with a single region writes.

Explanation

By configuring reads and writes of data for multi-region accounts with multi region writes, you can achieve 99.999% availability

Question 2

What are the three main advantages to using Cosmos DB?

- Cosmos DB offers global distribution capabilities out of the box.
- Cosmos DB provides a minimum of 99.99% availability.
- Cosmos DB response times of read/write operations are typically in the order of 10s of milliseconds.
- All of the above.

Explanation

All of the above. Cosmos DB offers global distribution capabilities out of the box, provides a minimum of 99.99% availability and has response times of read/write operations are typically in the order of 10s of milliseconds.

Question 1

You are a data engineer wanting to make the data that is currently stored in a Table Storage account located in the West US region available globally. Which Cosmos DB model should you migrate to?

- Gremlin API
- Cassandra API
- Table API
- Mongo DB API

Explanation

The Table API Cosmos DB model will enable you to provide global availability of your table storage account data. Gremlin API is used to store Graph databases. Cassandra API is used to store data from Cassandra databases and Mongo DB API is used to store Mongo DB databases.

Question 2

What type of data model provides a traversal language that enables connections and traversals across connected data?

- Gremlin API
- Cassandra API
- Table API
- Mongo DB API

Explanation

Gremlin API is used to store Graph databases and provides a traversal language that enables connections and traversals across connected data. Cassandra API is used to store data from Cassandra databases. The Table API is used to store key-value pair of data and Mongo DB API is used to store Mongo DB databases.

Question 1

Suppose you are using Visual Studio Code to develop a .NET Core application that accesses Azure Cosmos DB. You need to include the connection string for your database in your application configuration. What is the most convenient way to get this information into your project?

- Directly from Visual Studio Code
- From the Azure portal
- Using the Azure CLI

Explanation

The Azure Cosmos DB extension lets you administer and create Azure Cosmos DB accounts and databases from within Visual Studio Code. The Azure Portal does not aid the development of a .Net Core Application, and Azure CLI helps more with automation.

Question 2

When working with Azure Cosmos DB's SQL API, which of these can be used to perform CRUD operations?

- LINQ
- Apache Cassandra client libraries
- Azure Table Storage libraries

Explanation

LINQ and SQL are two of the valid methods for querying the SQL API. Apache Cassandra client libraries are used for the Cassandra API and Azure Table Storage libraries for the Table API and not the SQL API.

Question 3

When working with the Azure Cosmos DB Client SDK's DocumentClient class, you use a NOSQL model. How would you use this class to change the FirstName field of a Person Document from 'Ann' to 'Fran'?

- Call UpdateDocumentAsync with FirstName=Fran
- Call UpsertDocumentAsync with an updated Person object
- Call ReplaceDocumentAsync with an updated Person object

Explanation

ReplaceDocumentAsync will replace the existing document with the new one. In this case we'd intend the old and new to be the same other than FirstName. The DocumentClient class doesn't have an UpdateDocumentAsync method. Updating a single field is not consistent with the document-style NOSQL approach.

While calling UpsertDocumentAsync with an updated Person object would work, it isn't the minimum necessary access to meet our requirements. Upsert operations will replace a document if its key already exists or add a new document if not. We don't want to add a new one, so using this method risks introducing subtle, hard to track bugs.

Question 1

You want to maximize the data integrity of data that is stored in a Cosmos DB. Which consistency level should you choose?

- Session
- Strong
- Eventual

Explanation

Strong ensures that the data integrity is maintained. Eventual maximizes concurrency to the cost of data integrity. Session is a consistency level between Strong and Eventual.

Module 5 Working with Relation Data in the Cloud

Module Introduction

Working with Relational Data Stores in the Cloud

In this module, students will explore the Azure relational data platform options including Azure SQL Database and the enterprise data warehouse capabilities using Azure Synapse Analytics. The student will be able explain why they would choose one service over another, and how to provision and connect to each of the services.

Learning objectives

In this module, you'll learn:

- Work with Azure SQL Database
- Work with Azure Synapse Analytics
- Provision and query data in Azure Synapse Analytics
- Import data into Azure Synapse Analytics using PolyBase

Important This module will only focus on the enterprise data warehouse capabilities of Azure Synapse Analytics.

Azure SQL Database

Azure SQL Database

Managing data is critical for any business. Relational databases, and specifically Microsoft SQL Server, have been among the most common tools for handling data for decades.

If we want to manage our data using the cloud, we *can* just use Azure virtual machines to host our own Microsoft SQL Server instances. Sometimes that's the right solution, but Azure offers another way that is often much easier and more cost effective. Azure SQL Database are a Platform-as-a-Service (PaaS) offering, meaning much less infrastructure and maintenance to manage yourself.

To understand better, let's consider a scenario: You're a software development lead at a transportation logistics company, Contoso Transport.

The transportation industry requires tight coordination among everyone involved: schedulers, dispatchers, drivers, and even customers.

Your current process involves piles of paper forms and hours on the phone to coordinate shipments. You find that paperwork is often missing signatures and dispatchers are frequently unavailable. These holdups leave drivers sitting idle, which causes important shipments to arrive late. Customer satisfaction and repeat business are crucial to your bottom line.

Your team decides to move from paper forms and phone calls to digital documents and online communication. Going digital will enable everyone to coordinate and track shipment times through their web browser or mobile app.

You want to quickly prototype something to share with your team. Your prototype will include a database to hold driver, customer, and order information. Your prototype will be the basis for your production app. So the technology choices you make now should carry to what your team delivers.

Learning objectives

In this module, you'll learn:

- Why Azure SQL Database is a good choice for running your relational database
- What configuration and pricing options are available for your Azure SQL database
- How to create an Azure SQL database from the portal
- How to use Azure Cloud Shell to connect to your Azure SQL database, add a table, and work with data

Planning an Azure SQL Database

Azure provides PaaS services to help you manage all kinds of data, from highly structured relational data to unstructured data.

Here you'll learn why Azure SQL Database is a convenient, cost-effective, and secure way to host your relational databases.

Why choose Azure SQL Database?

Your transportation logistics application requires stored procedures that run basic CRUD (**create**, **read**, **update**, and **delete**) operations. You have experience working with SQL Server and other relational databases, and you want to take advantage of that existing skillset.

You consider two choices for your database:

1. Host SQL Server on-premises. Your IT team runs a small in-house data center to support the finance department and a few other teams. You can work with IT to host a SQL Server deployment in their data center.
2. Host Azure SQL Database in the cloud. Azure SQL Database is based on SQL Server and provides the relational database functionality you need.

You've decided to build the web and application tiers for your logistics app on Azure. So it makes sense to also host your database there. But there are some other reasons why Azure SQL Database is a smart choice, and why it's even easier than using virtual machines.

- **Convenience**

Setting up SQL Server on a VM or on physical hardware requires you to know about hardware and software requirements. You'll need to understand the latest security best practices and manage operating system and SQL Server patches on a routine basis. You also need to manage backup and data retention issues yourself.

With Azure SQL Database, Microsoft manage the hardware, software updates, and OS patches for you. All you specify is the name of your database and a few options. You'll have a running SQL database in minutes.

You can bring up and tear down Azure SQL Database instances at your convenience. Azure SQL Database comes up fast and is easy to configure. You can focus less on configuring software and more on making your app great.

- **Cost**

Because we manage things for you, there are no systems for you to buy, provide power for, or otherwise maintain.

Azure SQL Database has several pricing options. These pricing options enable you to balance performance versus cost. You can start for just a few dollars a month.

- **Scale**

You find that the amount of transportation logistics data you must store doubles every year. When running on-premises, how much excess capacity should you plan for?

With Azure SQL Database, you can adjust the performance and size of your database on the fly when your needs change.

- **Security**

Azure SQL Database comes with a firewall that's automatically configured to restrict connections from the Internet.

You can add IP addresses you trust to a Safe recipient list. The Safe recipient list lets you use Visual Studio, SQL Server Management Studio, or other tools to manage your Azure SQL database.

With Azure SQL Database, Microsoft provide buying options to help you get the performance you need at a predictable cost. Azure SQL Database also comes with a firewall so that you can control access to your data.

Although you don't need to be a DBA to use Azure SQL Database, there are a few concepts you should understand before you start. We'll cover these concepts next.

Creating an Azure SQL Database

Your transportation company wants to set themselves apart from other companies but without breaking the bank. You must have a good handle on how to set up the database to provide the best service while controlling costs.

Here, you'll learn:

- What considerations you need to make when creating an Azure SQL database, including:
 - How a logical server acts as an administrative container for your databases.
 - The differences between purchasing models.
 - How elastic pools enable you to share processing power among databases.
 - How collation rules affect how data is compared and sorted.
- How to bring up Azure SQL Database from the portal.
- How to add firewall rules so that your database is accessible from only trusted sources.

Let's take a quick look at some things you need to consider when you create an Azure SQL database.

One server, many databases

When you create your first Azure SQL database, you also create an *Azure SQL logical server*. Think of a logical server as an administrative container for your databases. This can include both Azure SQL Databases and Azure SQL Data Warehouse databases. You can control logins, firewall rules, and security policies through the logical server. You can also override these policies on each database within the logical server.

For now, you need just one database. But a logical server enables you to add more later and tune performance among all your databases.

Choose performance: DTUs versus vCores

Azure SQL Database has two purchasing models: DTU and vCore.

What are DTUs

DTU stands for Database Transaction Unit and is a combined measure of compute, storage, and IO resources. Think of the DTU model as a simple, preconfigured purchase option.

Because your logical server can hold more than one database, there's also the idea of eDTUs, or elastic Database Transaction Units. This option enables you to choose one price, but allow each database in the pool to consume fewer or greater resources depending on current load.

What are vCores

vCore gives you greater control over what compute and storage resources you create and pay for.

While the DTU model provides fixed combinations of compute, storage, and IO resources, the vCore model enables you to configure resources independently. For example, with the vCore model you can increase storage capacity but keep the existing amount of compute and IO throughput.

Your transportation and logistics prototype only needs one Azure SQL Database instance. You decide on the DTU option because it provides a good balance of compute, storage, and IO performance and is less expensive to get started.

What are SQL elastic pools

When you create your Azure SQL database, you can create a *SQL elastic pool*.

SQL elastic pools relate to eDTUs. They enable you to buy a set of compute and storage resources that are shared among all the databases in the pool. Each database can use the resources they need, within the limits you set, depending on current load.

For your prototype, you won't need a SQL elastic pool because you need only one SQL database.

What are SQL Managed Instances

The managed instance deployment option for Azure SQL Database creates a database with near 100% compatibility with the latest SQL Server on-premises Enterprise Edition database engine, favorable for on-premises SQL Server customers who would like to migrate their on-premises instance in a "lift and shift" manner. SQL Managed Instance provides a way to migrate huge databases into Azure cloud without changing any code and stay compliant with all the applications so you can make a one-to-one transfer.

What is collation

Collation refers to the rules that sort and compare data. Collation helps you define sorting rules when case sensitivity, accent marks, and other language characteristics are important.

Let's take a moment to consider what the default collation, **SQL_Latin1_General_CI_AS**, means.

- **Latin1_General** refers to the family of Western European languages.
- **CP1** refers to code page 1252, a popular character encoding of the Latin alphabet.
- **CI** means that comparisons are case insensitive. For example, "HELLO" compares equally to "hello".
- **AS** means that comparisons are accent sensitive. For example, "résumé" doesn't compare equally to "resume".

Because you don't have specific requirements around how data is sorted and compared, you choose the default collation.

Resource limits

It is important to bear in mind that Azure SQL Database has a number of different offering that can be purchased including basic, standard and premium tiers. The tier selected differs in the maximum resource limits, which therefore has an impact on cost. For example, the basic tier; if selected, can only utilize a maximum of 5 DTU's, with a 2GB maximum on disk space and a maximum of 300 concurrent connections. At the time of writing, the highest premium tier of P15 enables the use of a maximum of 4000 DTU's, with 4TB of disk space and a maximum of 30000 concurrent connections.

NOTE

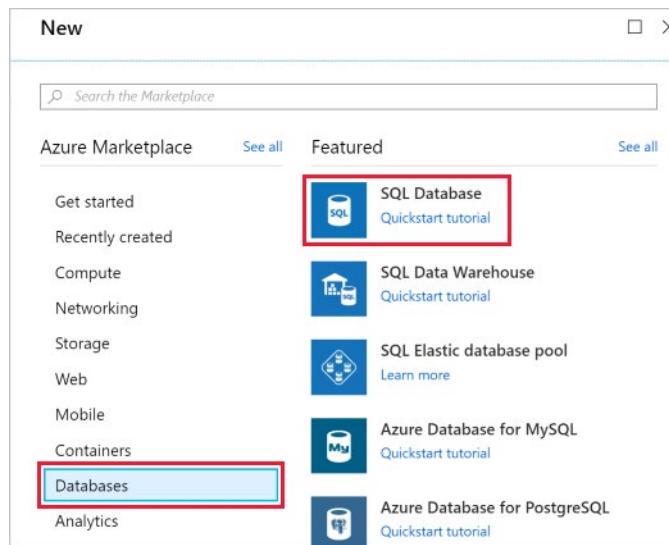
A full list of limits can be found [here¹](#), and it is important to note that these are limits based on an Azure SQL Database DTU limits. The limits can also vary based on where the Azure SQL Databases is configured as a DTU, elastic pool or managed instance.

Create your Azure SQL database

Here you'll set up your database, which includes creating your logical server. You'll choose settings that support your transportation logistics application. In practice, you would choose settings that support the kind of app you're building.

Over time if you realize you need additional compute power to keep up with demand, you can adjust performance options or even switch between the DTU and vCore performance models.

1. Sign into the [Azure portal²](#) using the same account you activated the sandbox with.
2. From the portal, click **Create a resource** from the upper left-hand corner. Select **Databases**, then select **SQL Database**.



3. Under **Server**, click **Configure required settings**, fill out the form, then click **Select**. Here's more information on how to fill out the form:

| Setting | Value |
|---------------------------|--|
| Server name | A globally unique server name (https://docs.microsoft.com/azure/architecture/best-practices/naming-conventions). |
| Server admin login | A database identifier (https://docs.microsoft.com/sql/relational-databases/databases/database-identifiers) that serves as your primary administrator login name. |

¹ <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-dtu-resource-limits-single-databases>

² <https://portal.azure.com/learn/docs.microsoft.com?azure-portal=true>

| Setting | Value |
|-----------------|---|
| Password | Any valid password that has at least eight characters and contains characters from three of these categories: uppercase characters, lowercase characters, numbers, and non-alphanumeric characters. |
| Location | Any valid location from the available list below. |

4. Click **Pricing tier** to specify the service tier. Select the **Basic** service tier, then click **Apply**.
5. Use these values to fill out the rest of the form.

| Setting | Value |
|--------------------------------------|---|
| Database name | Logistics |
| Subscription | Your subscription |
| Resource group | Use the existing group <rgn>[sandbox resource group name]</rgn> |
| Select source | Blank database |
| Want to use SQL elastic pool? | Not now |
| Collation | SQL_Latin1_General_CI_AS |

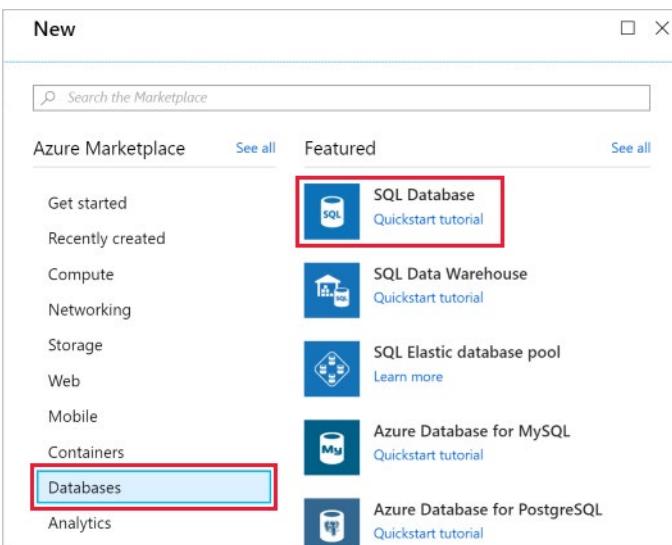
6. Click **Create** to create your Azure SQL database.

IMPORTANT

Remember your server name, admin login, and password for later.

7. On the toolbar, click **Notifications** to monitor the deployment process.

When the process completes, click **Pin to dashboard** to pin your database server to the dashboard so that you have quick access when you need it later.



Set the server firewall

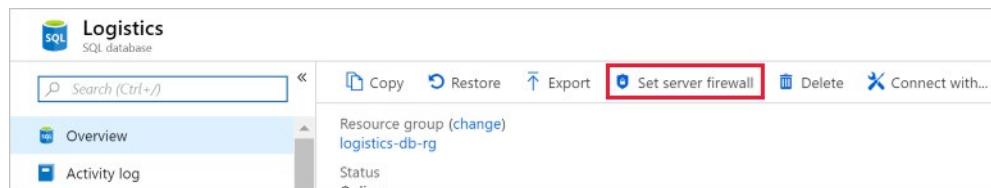
Your Azure SQL database is now up and running. You have many options to further configure, secure, monitor, and troubleshoot your new database.

You can also specify which systems can access your database through the firewall. Initially, the firewall prevents all access to your database server from outside of Azure.

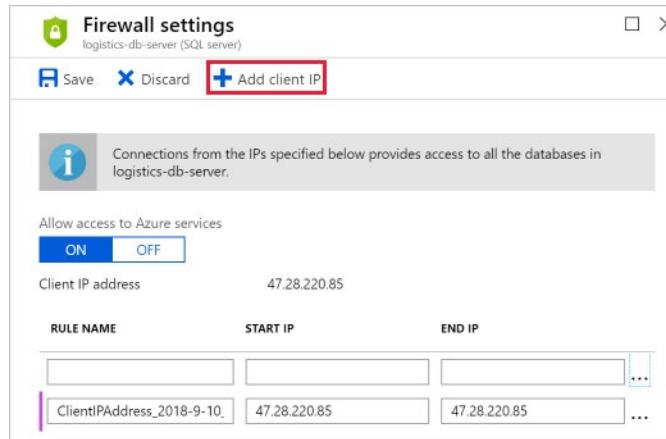
For your prototype, you only need to access the database from your laptop. Later, you can add additional systems, such as your mobile app.

Let's enable your development computer to access the database through the firewall now.

1. Go to the overview blade of the Logistics database. If you pinned the database earlier, you can click the **Logistics** tile on the dashboard to get there.
2. Click **Set server firewall**.



3. Click **Add client IP**.



4. Click **Save**.

In the next part, you'll get some hands-on practice with your new database and with Azure Cloud Shell. You'll connect to the database, create a table, add some sample data, and execute a few SQL statements.

Adding Data to Azure SQL Database

Before you connect the database to your app, you want to check that you can connect to it, add a basic table, and work with sample data.

Microsoft maintain the infrastructure, software updates, and patches for your Azure SQL database. Beyond that, you can treat your Azure SQL database like you would any other SQL Server installation. For example, you can use Visual Studio, SQL Server Management Studio, Azure Data Studio, or other tools to manage your Azure SQL database.

How you access your database and connect it to your app is up to you. But to get some experience working with your database, here you'll connect to it directly from the portal, create a table, and run a few basic CRUD operations. You'll learn:

- What Cloud Shell is and how to access it from the portal.

- How to access information about your database from the Azure CLI, including connection strings.
- How to connect to your database using `sqlcmd`.
- How to initialize your database with a basic table and some sample data.

What is Azure Cloud Shell?

Azure Cloud Shell is a browser-based shell experience to manage and develop Azure resources. Think of Cloud Shell as an interactive console that runs in the cloud.

Behind the scenes, Cloud Shell runs on Linux. But depending on whether you prefer a Linux or Windows environment, you have two experiences to choose from: Bash and PowerShell.

Cloud Shell is accessible from anywhere. Besides the portal, you can also access Cloud Shell from **shell.azure.com**³, the Azure mobile app, or from Visual Studio Code. The panel on the right is a Cloud Shell terminal for you to use during this exercise.

Cloud Shell includes popular tools and text editors. Here's a brief look at `az`, `jq`, and `sqlcmd`, three tools you'll use for our current task.

- `az` is also known as the Azure CLI. It's the command-line interface for working with Azure resources. You'll use this to get information about your database, including the connection string.
- `jq` is a command-line JSON parser. You'll pipe output from `az` commands to this tool to extract important fields from JSON output.
- `sqlcmd` enables you to execute statements on SQL Server. You'll use `sqlcmd` to create an interactive session with your Azure SQL database.

Get information about your Azure SQL database

Before you connect to your database, it's a good idea to verify it exists and is online.

Here, you use the `az` utility to list your databases and show some information about the **Logistics** database, including its maximum size and status.

1. The `az` commands you'll run require the name of your resource group and the name of your Azure SQL logical server. To save typing, run this `azure configure` command to specify them as default values.
Replace `<server-name>` with the name of your Azure SQL logical server. Note that depending on the blade you are on in the portal this may show as a FQDN (`servername.database.windows.net`), but you only need the logical name without the `.database.windows.net` suffix.

```
az configure --defaults group=<rgn>[sandbox resource group name]</rgn>
sql-server=<server-name>
```

1. Run `az sql db list` to list all databases on your Azure SQL logical server.

```
az sql db list
```

You see a large block of JSON as output.

1. Since we want just the database names, run the command a second time. This time, pipe the output to `jq` to print out only the name fields.

³ <https://shell.azure.com/>

```
az sql db list | jq '[.[] | {name: .name}]'
```

You should see this output.

```
[  
  {  
    "name": "Logistics"  
  },  
  {  
    "name": "master"  
  }  
]
```

Logistics is your database. Like SQL Server, **master** includes server metadata, such as sign-in accounts and system configuration settings.

1. Run this `az sql db show` command to get details about the **Logistics** database.

```
az sql db show --name Logistics
```

As before, you see a large block of JSON as output.

1. Run the command a second time. This time, pipe the output to `jq` to limit output to only the name, maximum size, and status of the **Logistics** database.

```
az sql db show --name Logistics | jq '{name: .name, maxSizeBytes: .maxSizeBytes, status: .status}'
```

You see that the database is online and can hold around 2 GB of data.

```
{  
  "name": "Logistics",  
  "maxSizeBytes": 2147483648,  
  "status": "Online"  
}
```

Connect to your database

Now that you understand a bit about your database, let's connect to it using `sqlcmd`, create a table that holds information about transportation drivers, and perform a few basic CRUD operations.

Remember that CRUD stands for **create**, **read**, **update**, and **delete**. These terms refer to operations you perform on table data and are the four basic operations you need for your app. Now's a good time to verify you can perform each of them.

1. Run this `az sql db show-connection-string` command to get the connection string to the **Logistics** database in a format that `sqlcmd` can use.

```
az sql db show-connection-string --client sqlcmd --name Logistics
```

Your output resembles this.

```
"sqlcmd -S tcp:contoso-1.database.windows.net,1433 -d Logistics -U <username> -P <password> -N -l 30"
```

1. Run the `sqlcmd` statement from the output of the previous step to create an interactive session. Remove the surrounding quotes and replace `<username>` and `<password>` with the username and password you specified when you created your database. Here's an example.

```
sqlcmd -S tcp:contoso-1.database.windows.net,1433 -d Logistics -U martina  
-P "password1234$" -N -l 30
```

[!TIP]

Place your password in quotes so that "&" and other special characters aren't interpreted as processing instructions.

1. From your `sqlcmd` session, create a table named `Drivers`.

```
CREATE TABLE Drivers (DriverID int, LastName varchar(255), FirstName var-  
char(255), OriginCity varchar(255));  
GO
```

The table contains four columns: a unique identifier, the driver's last and first name, and the driver's city of origin.

[!NOTE]

The language you see here is Transact-SQL, or T-SQL.

1. Verify that the `Drivers` table exists.

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

You should see this output.

```
name  
-----  
-----  
Drivers  
  
(1 rows affected)
```

1. Run this `INSERT` T-SQL statement to add a sample row to the table. This is the **create** operation.

```
INSERT INTO Drivers (DriverID, LastName, FirstName, OriginCity) VALUES  
(123, 'Zirne', 'Laura', 'Springfield');  
GO
```

You see this to indicate the operation succeeded.

```
(1 rows affected)
```

1. Run this `SELECT` T-SQL statement to list the `DriverID` and `OriginCity` columns from all rows in the table. This is the **read** operation.

```
SELECT DriverID, OriginCity FROM Drivers;
GO
```

You see one result with the `DriverID` and `OriginCity` for the row you created in the previous step.

```
DriverID      OriginCity
-----
123          Springfield
(1 rows affected)
```

1. Run this UPDATE T-SQL statement to change the city of origin from "Springfield" to "Boston" for the driver with a `DriverID` of 123. This is the **update** operation.

```
UPDATE Drivers SET OriginCity='Boston' WHERE DriverID=123;
GO
SELECT DriverID, OriginCity FROM Drivers;
GO
```

You should see the following output. Notice how the `OriginCity` reflects the update to Boston.

```
DriverID      OriginCity
-----
123          Boston
(1 rows affected)
```

1. Run this DELETE T-SQL statement to delete the record. This is the **delete** operation.

```
DELETE FROM Drivers WHERE DriverID=123;
GO

(1 rows affected)
```

1. Run this SELECT T-SQL statement to verify the `Drivers` table is empty.

```
SELECT COUNT(*) FROM Drivers;
GO
```

You see that the table contains no rows.

```
-----
0

(1 rows affected)
```

Now that you understand how to work with Azure SQL Database from Cloud Shell, you can get the connection string for your favorite SQL management tool – whether that's from SQL Server Management Studio, Visual Studio, or something else.

Cloud Shell makes it easy to access and work with your Azure resources. Because Cloud Shell is browser-based, you can access it from Windows, macOS, or Linux – essentially any system with a web browser.

You gained some hands-on experience running Azure CLI commands to get information about your Azure SQL database. As a bonus, you practiced your T-SQL skills.

Summary

Now that your Azure SQL database is up and running, you can connect it to your favorite SQL Server management tool to populate it with real data.

You initially considered whether to run your database on-premises or in the cloud. With Azure SQL Database, you configure a few basic options and you have a fully functional SQL database that you can connect to your apps.

There's no infrastructure or software patches to maintain. You're now free to focus more on getting your transportation logistics app prototype up and running and less on database administration. Your prototype won't be a throw-away demo, either. Azure SQL Database provides production-level security and performance features.

Remember that each Azure SQL logical server contains one or more databases. Azure SQL Database provides two pricing models, DTU and vCore, to help you balance cost versus performance across all your databases.

Choose DTU if you're just getting started or want a simple, preconfigured buying option. Choose vCore when you want greater control over what compute and storage resources you create and pay for.

Azure Cloud Shell makes it easy to start working with your databases. From Cloud Shell, you have access to the Azure CLI, which enables you to get information about your Azure resources. Cloud Shell also provides many other common utilities, such as `sqlcmd`, to help you start working right away with your new database.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Who's responsible for performing software updates on your Azure SQL databases and the underlying OS?

- You are. It's up to you to periodically log in and install the latest security patches and updates.
- Microsoft Azure. Azure manages the hardware, software updates, and OS patches for you.
- No one. Your database stays with its original OS and software configuration.

Question 2

What is an Azure SQL logical server?

- An administrative container for your databases.
- Another name for an Azure SQL database instance.
- A server that defines the logical rules that sort and compare data

Question 3

Your Azure SQL database provides adequate storage and compute power. But you find that you need additional IO throughput. Which performance model might you use?

- DTU
- vCore
- SQL elastic pool

Azure Synapse Analytics

Azure Synapse Analytics

A data warehouse is a centralized relational database that integrates data from one or more disparate sources. Data warehouses store current and historical data and are used for reporting and analysis of the data. Many organizations need to store and process data more quickly and easily at an increasing scale to meet business demand and respond to market shifts.

Creating and managing data warehouses has involved complex processes, using different technologies including Big Data technologies and data integration tools to ingest and prepare the data before presenting it to applications.

Azure Synapse Analytics is designed to simplify this process with limitless scale through one experience.

Assume you are a Data Engineer for Contoso. Your on-premises business intelligence solution consistently completes around 11:00 AM every morning. The Database Administrator has tried everything to improve performance including scaling up and tuning the servers as far as they can go, but the impact has been minimal. Report generation will complete by 10:00 AM at the earliest. As a Data Engineer, you have been tasked to determine how Azure Synapse Analytics can improve the situation and help the database administrator set it up for your company.

Learning Objectives

In this module you will:

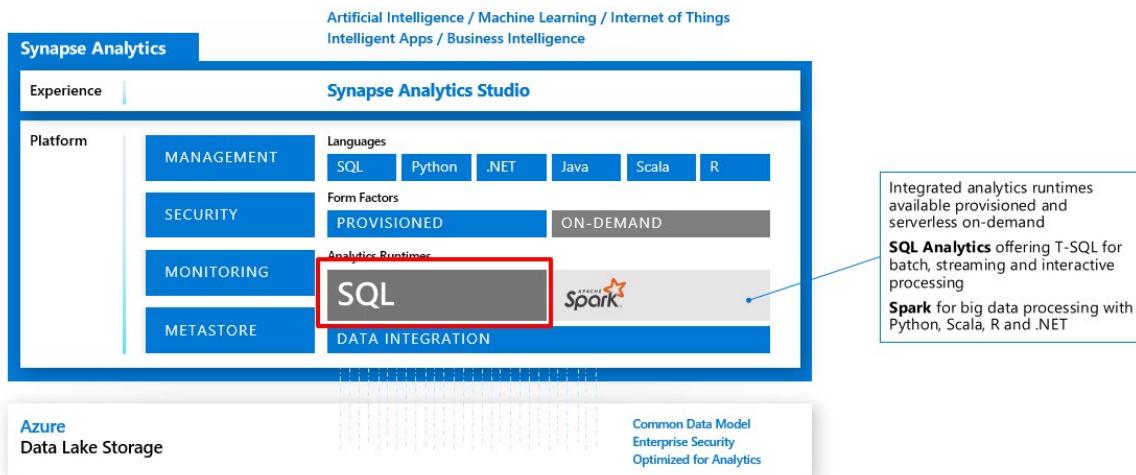
- Explain Azure Synapse Analytics
- List the types of solution workloads
- Explain Massively Parallel Processing Concepts
- Compare Table Geometries
- Create an Azure Synapse Analytics Service

What is Azure Synapse Analytics

Azure Synapse Analytics provides a unified environment by combining the enterprise data warehouse of SQL, the Big Data analytics capabilities of Spark, and data integration technologies to ease the movement of data between both, and from external data sources. Using Azure Synapse Analytics, you are able to ingest, prepare, manage, and serve data for immediate BI and machine learning needs more easily.

Using SQL Analytics and SQL Pools in Azure Synapse Analytics, you will be able to work with modern data warehouse use cases to create database and tables, load data using PolyBase and query a data warehouse using the Massively Parallel Processing architecture. In fact, you can use any existing code that you have created for Azure SQL Data Warehouse with Azure Synapse Analytics seamlessly.

Azure Synapse Analytics stores the incoming data into relational tables with columnar storage. This format significantly reduces the data storage costs and improves the query performance. Once information is stored in Azure Synapse Analytics, you can run analytics at massive scale. Compared to the traditional database systems, analysis queries on Azure Synapse Analytics finishes in a fraction of the time. Within this, SQL Analytics offers T-SQL for batch, streaming and interactive processing of data



Data Warehousing with Azure Synapse Analytics

SQL Pools using Data Warehouse Units

With Azure Synapse Analytics, CPU, memory, and IO are bundled into units of compute scale called SQL pool. It is a normalized measure of compute resources and performance, and the size of SQL pool is determined by Data Warehousing Units (DWU). The size of SQL pool is determined by Data Warehousing Units (DWU). By changing your service level you alter the number of DWUs that are allocated to the system, which in turn adjusts the performance, and the cost, of your system. To pay for higher performance, you can increase the number of data warehouse units. To pay for less performance, reduce data warehouse units. Storage and compute costs are billed separately, so changing data warehouse units does not affect storage costs.

Benefits of data warehousing with Azure Synapse Analytics

Azure Synapse Analytics is a key component required for creating end-to-end relational big data solutions in the cloud today. It allows data to be ingested from a variety of data sources and leverages a scale-out architecture to distribute computational processing of data across a large cluster of nodes, which can:

- Independently size compute power irrespective of the storage needs.
- Grow or shrink compute power without moving data.
- Pause compute capacity while leaving data intact, so you only pay for storage.
- Resume compute capacity during operational hours.

This is made possible due to the decoupling of computation and storage using the Massively Parallel Processing architecture.

In future releases, Azure Synapse Analytics will be able to provide the following additional services, including:

- Tight integration with **Apache Spark**
- **Data integration** capabilities to make it easier to ingest and process data

- **Azure Synapse Analytics Studio** to provide a unified experience for management

Azure Synapse Analytics features

As well as the Massively Parallel Processing architecture at limitless scale, there are additional key features of SQL Analytics that can be beneficial in Modern Data Warehousing use cases, including:

Workload Management

Azure Synapse Analytics provides the capability to prioritize the query workloads that take place on the server using Workload Management. This is managed by three related areas:

Workload Groups

Workload Groups enable you to define the resources to isolate and reserve resources for its use. It brings about the following benefits:

- Reserve resources for a group of requests
- Limit the amount of resources a group of requests can consume
- Shared resources accessed based on importance level
- Set Query timeout value. Get DBAs out of the business of killing runaway queries

A workload group is defined using T-SQL as follows:

```
CREATE WORKLOAD GROUP group_name
WITH
(
    MIN_PERCENTAGE_RESOURCE = value
    , CAP_PERCENTAGE_RESOURCE = value
    , REQUEST_MIN_RESOURCE_GRANT_PERCENT = value
    [ [ , ] REQUEST_MAX_RESOURCE_GRANT_PERCENT = value ]
    [ [ , ] IMPORTANCE = {LOW | BELOW_NORMAL | NORMAL | ABOVE_NORMAL | HIGH}
]
    [ [ , ] QUERY_EXECUTION_TIMEOUT_SEC = value ]
) [ ; ]
```

Workload Classification

Using T-SQL, you can create a workload classifier to map queries to a specific classifier. A classifier can define the level of importance of the request, so that it can be mapped to a specific workload group that has an allocation of specific resources during the execution of the query.

```
CREATE WORKLOAD CLASSIFIER classifier_name
WITH
(
    WORKLOAD_GROUP = 'name'
    , MEMBERNAME = 'security_account'
    [ [ , ] IMPORTANCE = {LOW|BELOW_NORMAL|NORMAL|ABOVE_NORMAL|HIGH} ]
    [ [ , ] WLM_LABEL = 'label' ]
    [ [ , ] WLM_CONTEXT = 'name' ]
    [ [ , ] START_TIME = 'start_time' ]
```

```
[ [ , ] END_TIME      = 'end_time' ]
) [ ; ]  
  
WORKLOAD_GROUP: maps to an existing resource class  
IMPORTANCE: specifies relative importance of request  
MEMBERNAME: database user, role, AAD login or AAD group
```

Workload Importance

Workload importance is defined in the CREATE WORKLOAD CLASSIFIER command and allows higher priority queries to receive resources ahead of lower priority queries that are in the queue. By default, queries are released from the queue on a first-in, first-out basis as resources become available, but workload importance overrides this.

Result-set Cache

In scenarios where the same results are requested on a regular basis, result-set caching can be used to improve the performance of the queries that retrieve these results. When result-set caching is enabled, the results of the query are cached in the SQL pool storage. This enables interactive response times for repetitive queries against tables with infrequent data changes.

The result-set cache persists even if SQL pool is paused and resumed later. Although the query cache is invalidated and refreshed when the underlying table data or query code changes. To ensure that the cache is fresh, the result cache is evicted on a regular basis on a time-aware least recently used algorithm (TLRU). You can set result-set caching on at the database level or at a session level using the following code:

```
-- Turn on/off result-set caching for a database
-- Must be run on the MASTER database
ALTER DATABASE {database_name}
SET RESULT_SET_CACHING { ON | OFF }

-- Turn on/off result-set caching for a client session
-- Run on target Azure Synapse Analytics
SET RESULT_SET_CACHING {ON | OFF}
```

Materialized Views

A materialized view pre-computes, stores, and maintains its data like a table. They are automatically updated when data in underlying tables are changed. This is a synchronous operation that occurs as soon as the data is changed. This auto caching functionality allows Azure Synapse Analytics Query Optimizer to consider using an indexed view even if the view is not referenced in the query. They also support the following aggregations: **MAX, MIN, AVG, COUNT, COUNT_BIG, SUM, VAR, STDEV**

CI/CD support through SSDT

Database project support in SQL Server Data Tools (SSDT) allows teams of developers to collaborate over a version-controlled Azure Synapse Analytics, and track, deploy and test schema changes.

Other supported features include

- Ordered Columnstore
- JSON support
- Dynamic Data Masking
- Row-Level Security

Types of Solution Scenarios.

There are three common types of solution workloads that can be supported by Azure Synapse Analytics.

Modern Data Warehouse workloads:

A Modern Data Warehouse is a centralized data store that provides analytics and decision support services across the whole enterprise using structured, unstructured, or streaming data sources. Data flows into the warehouse from multiple transactional systems, relational databases, and other data sources on a periodic basis. The stored data is used for historical and trend analysis reporting. The data warehouse acts as a central repository for many subject areas and contains the "single source of truth."

SQL Analytics stores data in relational tables with columnar storage. This format significantly reduces the data storage costs, and improves query performance. Once data is stored, you can run analytics at massive scale. Compared to traditional database systems, analysis queries finish in seconds instead of minutes, or hours instead of days.

TIP

A data warehouse is useful to reduce stress on production systems and to reorganize, index, or filter multiple data sources into a single storage area. To accomplish this goal, deliberate thought and design needs to be put into how you organize and structure the data. It's not a dumping ground for tables from various sources without any design put into it.

Advanced Analytical Workloads

You can perform advanced analytics in the form of predictive or preemptive analytics using Azure Synapse Analytics. Using the tight integration with Spark, and the hybrid data integration engine with Data Integration. In a cloud data solution, data is ingested into big data stores from a variety of sources. Once in a big data store, Hadoop, Spark, and machine learning algorithms prepare and train the data. When the data is ready for complex analysis, SQL Analytics uses PolyBase to query the big data stores.

PolyBase uses standard T-SQL queries to bring the data into SQL Analytics tables. As a result, the worlds of the Modern Data Warehouse and the Advanced Analytical Workloads are brought together.

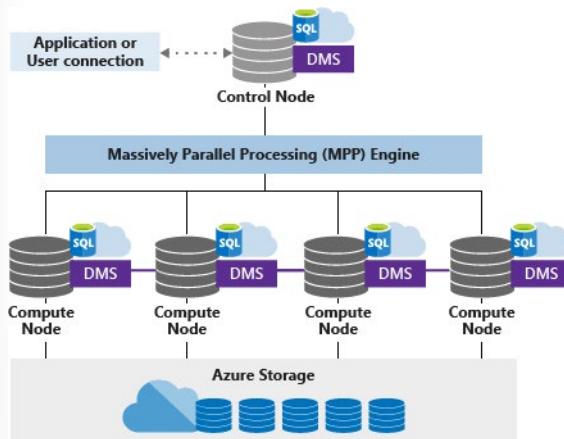
Real-time analytics

Real time analytics enables you to capture data continuously from devices or applications and to process it in near real time. This enables you to get live insights and can be setup with ease from a range of devices.

Massively Parallel Processing(MPP) Concepts

Azure Synapse Analytics separates the computation from the underlying storage which enables one to scale the computing power independently of the data storage. This is done by abstracting the CPU, memory, and IO, and bundling them into units of compute scale called SQL pool.

A SQL pool represents an abstract, normalized measure of compute resources and performance. By changing the service level, you alter the number of Data Warehouse Units (DWUs) that are allocated to the system, which in turn adjusts the performance, and thus the cost, of the system. To achieve higher performance, one can increase the number of data warehouse units, which also increases the associated costs. To achieve a lower cost and thereby less performance, one can reduce the data warehouse units. Storage and compute costs are billed separately, so changing the data warehouse units does not affect the storage costs.



Azure Synapse Analytics uses a node-based architecture. Applications connect and issue T-SQL commands to a Control node, which is the single point of entry for the data warehouse. The Control node runs the Massively Parallel Processing (MPP) engine which optimizes queries for parallel processing and then passes operations to Compute nodes to do their work in parallel. The Compute nodes store all user data in Azure Storage and run the parallel queries. The Data Movement Service (DMS) is a system-level internal service that moves data across the nodes as necessary to run queries in parallel and return accurate results.

Control node

The Control node is the brain of the data warehouse. It is the front end that interacts with all applications and connections. The MPP engine runs on the Control node to optimize and coordinate parallel queries. When one submits a T-SQL query to the Azure Synapse Analytics, the Control node transforms it into queries that run against each distribution in parallel.

Compute nodes

The Compute nodes provide the computational power. Distributions map to compute nodes for processing. As one pays for more compute resources, SQL Data Warehouse re-maps the distributions to the available compute nodes. The number of compute nodes ranges from 1 to 60 and is determined by the service level for the data warehouse.

Data Movement Service

Data Movement Service (DMS) is the data transport technology that coordinates data movement between the compute nodes. When SQL Data Warehouse runs a query, the work is divided into 60 smaller queries that run in parallel. Each of the 60 smaller queries runs on one of the underlying data distributions. A distribution is the basic unit of storage and processing for parallel queries that run on distributed data. Some queries require data movement across nodes to ensure the parallel queries return accurate results. When data movement is required, DMS ensures the right data gets to the correct location.

Azure Synapse Analytics Table Geometries

Azure Synapse Analytics uses Azure Storage to store data. Since the data is stored and managed by Azure Storage, Azure Synapse Analytics charges separately for the storage consumption. The data itself is sharded into distributions to optimize the performance of the system. One can choose which sharding pattern to use to distribute the data when one defines the table. Azure Synapse Analytics supports these sharding patterns:

- Hash
- Round Robin
- Replicate

The following strategies can be used to determine which pattern is most suitable for which scenario:

| Type | Great fit for... | Watch out if... |
|-----------------------|--|---|
| Replicated | <ul style="list-style-type: none"> • Small dimension tables in a star schema with less than 2 GB of storage after compression (~5x compression) | <ul style="list-style-type: none"> • Many write transactions are on table (such as insert, insert, delete, update) • You change Data Warehouse Units (DWU) provisioning frequently • You only use 2-3 columns, but your table has many columns • You index a replicated table |
| Round Robin (default) | <ul style="list-style-type: none"> • Temporary/staging table • No obvious joining key or good candidate column | <ul style="list-style-type: none"> • Performance is slow due to data movement |
| Hash | <ul style="list-style-type: none"> • Fact tables • Large dimension tables | <ul style="list-style-type: none"> • The distribution key cannot be updated |

Some tips that can help developers in this regard are:

- Start with Round Robin, but aspire to a Hash distribution strategy to take advantage of a massively parallel architecture.
- Make sure that common hash keys have the same data format.
- Do not distribute on *varchar* format.
- Dimension tables with a common hash key to a fact table with many join operations can be hash distributed.
- Use *sys.dm_pdw_request_steps* to analyze data movements behind queries, monitor the time broadcast and shuffle operations take. This is helpful to review your distribution strategy.

Hash-distributed tables

A hash distributed table can deliver the highest query performance for joins and aggregations on large tables.

To shard data into a hash-distributed table, Azure Synapse Analytics uses a hash function to assign each row to one distribution deterministically. In the table definition, one of the columns is designated as the distribution column. The hash function uses the values in the distribution column to assign each row to a distribution.

The following is an example of a create table statement that defines a hash distribution.

```
CREATE TABLE [dbo].<a href="#">[Date] [varchar](30) ,  
[BookId] [decimal](38, 0) ,  
[P&L] [decimal](31, 7) ,  
[VaRLower] [decimal](31, 7)" title="" target="_blank" data-generated='''>EquityTimeSeriesData</a>  
WITH  
(  
    CLUSTERED COLUMNSTORE INDEX  
,  
    DISTRIBUTION = HASH([P&L])  
)  
;
```

Round-robin distributed tables

A round-robin table is the most straightforward table to create and delivers fast performance when used as a staging table for loads.

A round-robin distributed table distributes data evenly across the table but without any further optimization. A distribution is first chosen at random, and then buffers of rows are assigned to distributions sequentially. It is quick to load data into a round-robin table, but query performance can often be better with hash distributed tables. Joins on round-robin tables require reshuffling data, and this takes additional time.

The following is an example of a create table statement that defines a round robin distribution.

```
CREATE TABLE [dbo].<a href="#">[Date] [datetime2](3) ,  
[DateKey] [decimal](38, 0) ,  
..  
..  
[WeekDay] [nvarchar](100) ,  
[Day Of Month] [decimal](38, 0)" title="" target="_blank" data-generated='''>Dates</a>  
WITH  
(  
    CLUSTERED COLUMNSTORE INDEX  
,  
    DISTRIBUTION = ROUND_ROBIN  
)  
;
```

Replicated Tables

A replicated table provides the fastest query performance for small tables.

A table that is replicated caches a full copy on each compute node. Consequently, replicating a table removes the need to transfer data among compute nodes before a join or aggregation. Replicated tables are best utilized with small tables. Extra storage is required, and there are additional overheads that are incurred when writing data which make large tables impractical.

The following is an example of a create table statement that defines a replicate distribution.

```
CREATE TABLE [dbo].<a href="#">BusinessHierarchies</a>
(
    [BookId] [nvarchar] (250) ,
    [Division] [nvarchar] (100) ,
    [Cluster] [nvarchar] (100) ,
    [Desk] [nvarchar] (100) ,
    [Book] [nvarchar] (100) ,
    [Volcker] [nvarchar] (100) ,
    [Region] [nvarchar] (100) " title="" target="_blank" data-generated="1">BusinessHierarchies</a>
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX
    , DISTRIBUTION = REPLICATE
)
;
```

Summary

In this module, you've learned that Azure Synapse Analytics provides a unified environment by combining an enterprise data warehouse and Big Data analytics capabilities. You've also learned how to optimize the storage of the data warehouse tablea using Round Robin, Hash Distribution by using replicated tables, and finally, you created a data warehouse.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which of the following terms refer to the scale of compute that is being used in an Azure SQL Synapse Analytics server?

- RTU
- DWU
- DTU

Question 2

You have an Azure Synapse Analytics database, within this, you have a dimension table named *Stores* that contains store information. There is a total of 263 stores nationwide. Store information is retrieved in more than half of the queries that are issued against this database. These queries include staff information per store, sales information per store and finance information. You want to improve the query performance of these queries by configuring the table geometry of the *stores* table. Which is the appropriate table geometry to select for the *stores* table?

- Round Robin
- Non Clustered
- Replicated table

Creating and Querying an Azure Synapse Analytics

Creating and Querying an Azure SQL Data Warehouse

In this module, you are going to learn the most fundamental operation to the Azure Synapse Analytics – querying the database. You are going to focus on how to read the data from a sample database with the SELECT statement and its clauses.

After being convinced that Azure Synapse can solve the issue of loading the business intelligence solution promptly, report writers within Contoso are nervous and are asking you as the data engineer to ease their concerns at writing reports against a cloud database. You intend to show the report writers that they can continue to use Transact SQL statements with the standard clauses including SELECT, FROM, WHERE, GROUP BY and HAVING. As you know that the major difference in their work is changing a connection string, you will create a new Azure Synapse Analytics server with a sample database to highlight the connection information with clarity.

Learning Objectives

In this module you will:

- Create an Azure Synapse Analytics sample database
- Query the sample database with the SELECT statement and its clauses
- Use the queries in different client applications such as SQL Server Management Studio, and PowerBI

Create an Azure Synapse Analytics

Below are the steps to create a sample data warehouse in Azure with sample data. You will need an Azure account to go through this exercise.

1. Sign in to the **Azure portal**⁴.
2. Select **Create a resource** from the left sidebar.
3. Select **Databases > Azure Synapse Analytics (formerly SQL DW)** to start the creation process.

⁴ <https://portal.azure.com?azure-port=true>

The screenshot shows the Azure Marketplace 'New' page. At the top, there's a breadcrumb navigation 'Home > New' and a search bar 'Search the Marketplace'. Below that, there are two tabs: 'Azure Marketplace' with 'See all' and 'Featured' with 'See all'. The 'Featured' tab is active. On the left, there's a sidebar with categories: Get started, Recently created, AI + Machine Learning, Analytics, Blockchain, Compute, Containers, Databases (which is highlighted with a red box), Developer Tools, DevOps, Identity, Integration, Internet of Things, and Media. On the right, there are several service cards: Azure SQL Managed Instance (Quickstart tutorial), SQL Database (Quickstart tutorial), Azure Synapse Analytics (formerly SQL DW) (which is highlighted with a red box and has a tooltip 'Azure Synapse Analytics (formerly SQL DW)'), Azure Database for MariaDB (Learn more), Azure Database for MySQL (Quickstart tutorial), Azure Database for PostgreSQL (Quickstart tutorial), and Azure Cosmos DB (Quickstart tutorial).

4. From the **SQL Data Warehouse** blade, create an Azure Synapse Analytics with the following settings:
- Subscription: the name of the subscription you are using in this lab
 - In **Additional setting** tab, under data source, click **Sample**. Click on the **Basics** tab
 - Resource group name: **mslearn-demodw**.
 - Database warehouse name: **sampleDataWH**.
 - Server: Create a new server by clicking **Create new** with the following settings and click on **OK**:
 - Server name: **sampledatawhxx**, where **xx** are your initials
 - Server admin login: **dwdbadmin**.
 - Password: **Pa55w.rd**
 - Confirm Password: **Pa55w.rd**
 - Location: choose a **location** near to you.
 - Select the checkbox to Allow Azure services to access server

The screenshot shows the Azure portal interface for creating a new SQL Data Warehouse. On the left, the main page has tabs for 'Basics', 'Additional settings*', 'Tags', and 'Review + create'. It includes sections for 'Project details' (subscription to 'chtestao', resource group '(New) mslearn-demodw'), 'Data warehouse details' (name 'demodW', server dropdown which is currently empty and highlighted with a red border), and a note about performance level. On the right, a separate modal window titled 'New server' is open, prompting for 'Server name' (set to 'demo-dw-server-ix'), 'Server admin login' (set to 'ctestoneill'), 'Password' (set to '*****'), 'Confirm password' (set to '*****'), 'Location' (set to '(Europe) West Europe'), and a checked checkbox for 'Allow Azure services to access server'.

5. Click the **OK** button to continue.
6. Click the *Performance level* entry to expand the **Configure performance** blade. The default performance level is DW1000c, Under Gen 2. Draw the slider to the left end. Then the performance level of the server will be downgraded to **DW100c**; this will reduce the cost per hour. Click **Apply**.

The screenshot shows the 'Configure performance' blade. It displays the 'Gen2' performance level, which is described as offering the highest performance and storage scalability options for intensive workloads, starting at 1.51 USD / hour. Below this, there's a 'Scale your system' section with a slider. The slider is currently positioned at the far left, corresponding to the 'DW100c' performance level, which is highlighted with a red border. The main configuration pane on the left lists database name ('demodw'), subscription ('Visual Studio Enterprise'), resource group ('(New) mslearn-demodw'), select source ('Sample'), server ('demo-dw-server (Central US)'), performance level ('Gen2: DW1000c'), and collation ('Specified by sample').

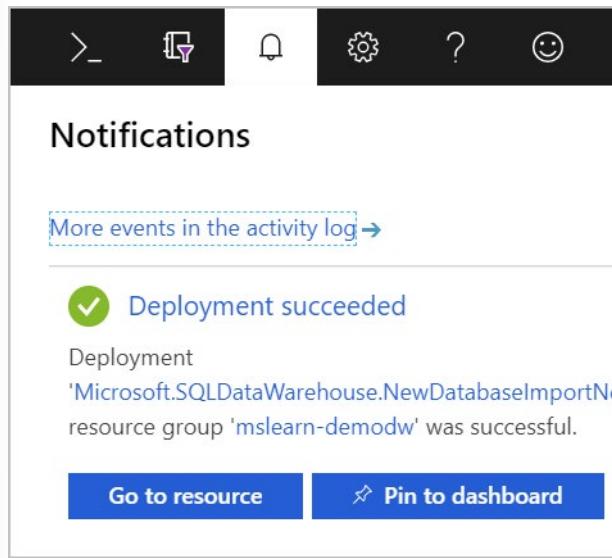
7. Click the **Review and Create** button
8. Click the **Create** button to start the deployment process.

This deployment will take a few minutes. Azure is doing several things:

1. Provisioning a server in a data center
2. Creating a new database

3. Loading the AdventureWorks data into the database

You can click the **Notification icon** on the toolbar to observe the progress of the creation. Once it's deployed, you will get a popup notification allowing you to navigate to the new resource or pin the resource to your dashboard.



Click **Go to the resource** to open the **Overview** page for your new database.

View the new database details

You can explore the database details on the **Overview** page. In the future, you can get to this through the global search feature or your Azure dashboard.

Here, you can get a view of the traffic being processed, access various features and configuration options, and perform common tasks such as backup/restore on the DB.

As an example, the "Server name" and "Connection strings" are two useful bits of information you can access on this view.

A screenshot of the Azure Synapse Analytics Overview page for the database "demodW". The page has a navigation menu on the left with items like "Overview", "Activity log", "Tags", "Diagnose and solve problems", "Settings", "Maintenance schedule", "Quick start", "Geo-backup policy", "Connection strings", "Properties", "Locks", "Export template", and "Security". The main content area shows a summary card with "Welcome to Azure Synapse Analytics (formerly known as Azure SQL Data Warehouse). Learn more." Below this, there are sections for "Resource group (change)" (mslearn-demodw), "Server name" (demo-dw-server-xx.database.windows.net, highlighted with a red box), "Status" (Online), "Location" (West Europe), "Subscription (change)" (chtestao), "Subscription ID" (96632c2c-b45e-4ad2-846b-359d2372565c), "Tags (change)" (Click here to add tags), and "DWU Usage" (1 hour, 24 hours, 7 days, Aggregation type: Max). A purple banner at the top right says "SQL Data Warehouse brings new workload management capabilities. Click here to learn more. →".

Connecting to Azure Synapse Analytics with SQL Server Management Studio (SSMS)

There are many applications you can use as a client for the Azure Synapse Analytics server. Data engineers often use SQL Server Management Studio (SSMS) to access Microsoft database products including Azure SQL Database, Azure Synapse Analytics, and on-premises SQL Server instances. Data analysts usually use Excel or Power BI as their client application to query the database. Developers use Visual Studio to write all kinds of third-party applications to query the database.

Since SMSS is a common tool, let's look at how to use it to query our data warehouse. To connect to our database, we need to create a *connection string*.

Note: Azure Synapse Analytics currently has a tool named Azure Synapse Analytics Studio in preview mode to interact with Azure Synapse Analytics.

Configuring SMSS to communicate with a database

A database connection string identifies the protocol, URL, port, and security options used to communicate with the database. The client uses this information to establish a network connection to the database. You can click on the **Show database connection** field on the **Overview** screen to get the connection string in a variety of framework formats:

- ADO.NET
- JDBC
- ODBC
- PHP

For example, here's the connection string for ADO.NET.

The screenshot shows the Azure portal interface for managing database connection strings. At the top, the navigation path is: Home > Resource groups > mslearn-demodw > demodw (demo-dw-server/demodw) > Data. Below this, the title is "Database connection strings" with a "demodw" tag. There are tabs for ADO.NET, JDBC, ODBC, and PHP. The ADO.NET tab is selected. Below the tabs, it says "ADO.NET (SQL authentication)". A large text input field contains the connection string:

```
Server=tcp:demo-dw-server.database.windows.net,1433;Initial Catalog=demodw;Persist Security Info=False;User ID={your_username};Password={your_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False
```

Below the text field is a link "Download ADO.NET driver for SQL server".

Notice that the connection string identifies the protocol as TCP/IP, includes the URL, and uses the port **1433** (the default in this case).

Since TCP:1433 is a well-known and public port, once your Azure Synapse Analytics server's name is made public, it may invite DoS (denial-of-service) attacks. To protect the server from such attacks, you can configure the Azure firewall to restrict network access to specific IP addresses.

Configure the firewall

To configure the firewall rules, navigate back to the **Overview** page of the Azure Synapse Analytics Server page

1. Click the **Show firewall settings** link.

Resource group (change) : mslearn-demodw
Status : Available
Location : Central US
Server admin : dwadmin
Firewalls and virtual net... : **Show firewall settings**
Active Directory admin : Not configured

2. From here, you can add a single IP address, an address segment, or a virtual network configuration to connect it securely to specific Azure regions or your on-premises networks.
 - For convenience, the portal lists the IP address of your computer.
 - You can click the **Add client IP** button in the top menu to add it as a rule, or add a set of known IP addresses/segments through the rules section.

demo-dw-server-xx - Firewalls and virtual networks

Search (Ctrl+ /)

Save Discard + Add client IP

Connections from the IPs specified below provides access to all the databases in demo-dw-server-xx.

Allow Azure services and resources to access this server

ON OFF

Client IP address 31.51.197.131

| Rule name | Start IP | End IP |
|-----------|----------|--------|
| | | |

No firewall rules configured.

Connections from the VNET/Subnet specified below provides access to all databases in demo-dw-server-xx.

Virtual networks + Add existing virtual network + Create new virtual network

| Rule name | Virtual network | Subnet | Address Range | Endpoint |
|-----------|-----------------|--------|---------------|----------|
| | | | | |

No vnet rules for this server.

Download SQL Server Management Studio (SSMS)

Once you have the connection string information, you can connect to the database with a client application. We're going to use SQL Server Management Studio. This is a free tool that runs on Windows. If you don't have it installed on your computer, you can use these **instructions**⁵ to download and install it.

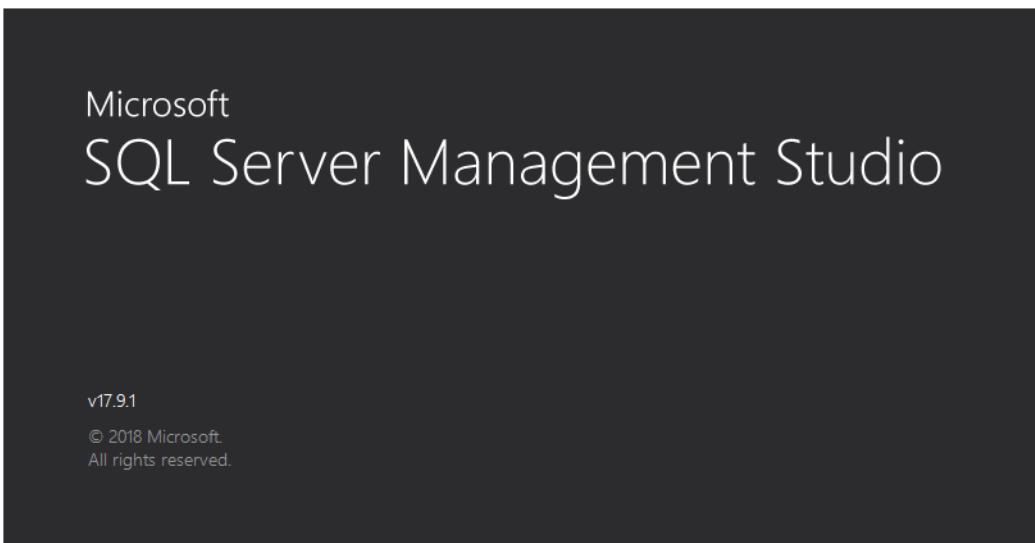
[!TIP]

While SSMS is Microsoft's preferred tool, you can also use other 3rd-party tools on other platforms. As long as it can connect to a SQL Server database and perform T-SQL queries, it will likely work for the following steps.

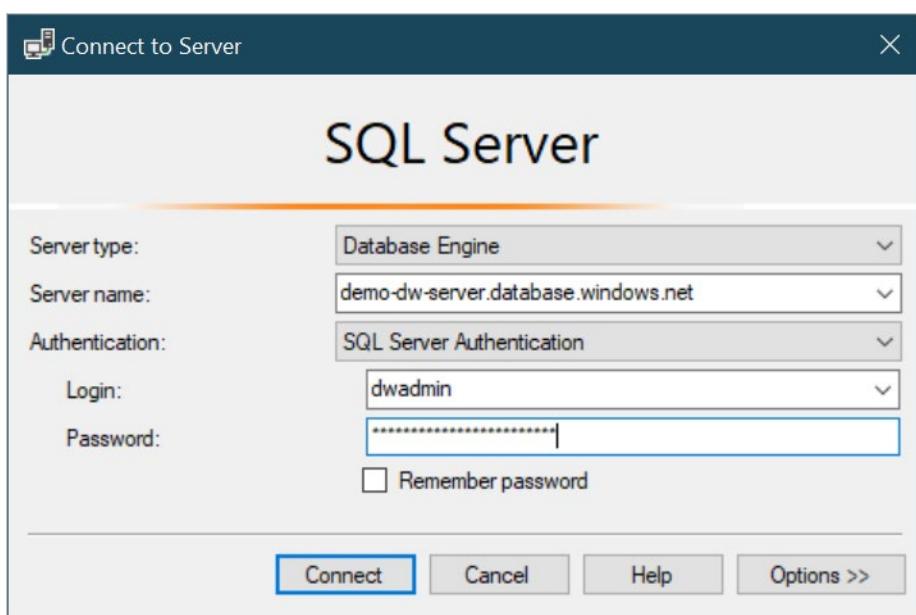
Connect to the database with SMSS

1. Launch SQL Server Management Studio.

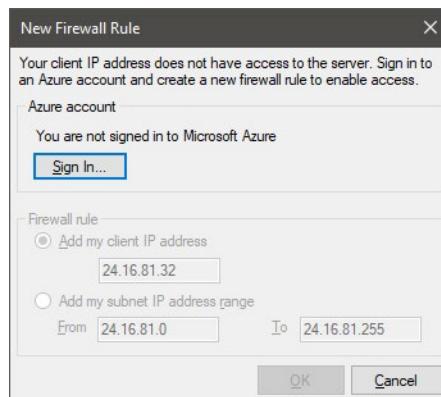
⁵ <https://docs.microsoft.com/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017>



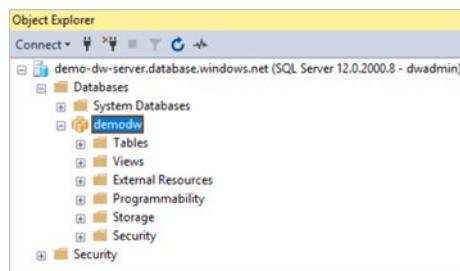
1. In the Connect to Server dialog:
 - Fill the server name as **demo-dw-server-xx.database.windows.net**
 - Select the Authentication type as **SQL Server Authentication**
 - Fill the login credentials. If you ever forget the admin credentials, you can get the admin ID from the database overview, as well as reset the password.
2. Click the Connect button to establish the network connection.



3. Since this is an Azure Synapse Analytics, you will get a prompt to sign into Azure. Go ahead and sign in using the same account you used to create the data warehouse.



4. Since the IP address of your computer has already been added to the firewall, your connection should be successful. And, you should see the ASDW server node and its database in the Object Explorer panel of the SSMS.



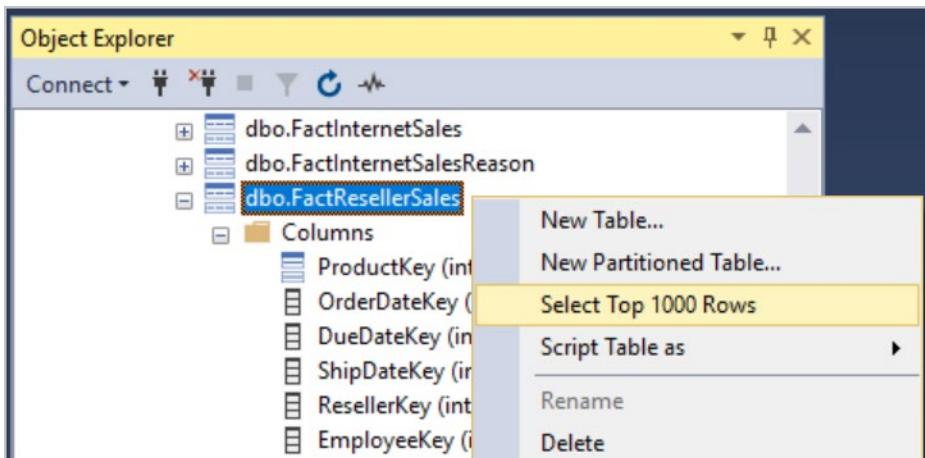
Querying Azure Synapse Analytics with SQL Server Management Studio (SSMS)

To extract data from the database, we need to form a *database query*. This is a textual command expressed in one or more SQL statements. Microsoft's variant of SQL is referred to as T-SQL and, although T-SQL is a sophisticated programming language, the most commonly used data queries are very straightforward. The pattern of the query to read data is:

```
SELECT <column(s)> FROM <fact_table>
    JOIN <dim_table> ON <condition(s)>
    WHERE <source_filter_condition>
    GROUP BY <dim_table_column(s)>
        HAVING <result_filter_condition>
    ORDER BY <source_or_result_column(s)>
```

Let's compose a T-SQL query step by step against an existing fact table **FactResellerSales**.

1. Expand the **demodw** database node in the SMSS Object Explorer. This will list all the tables available to you.
2. Find the fact table **FactResellerSales** and the dimension table **DimReseller**.
3. Expand these two tables to see their columns.
4. Right-click each of the the tables and select **Select Top 1000 Rows** in the context menu to view the data of the table.

**TIP**

The column view in Object Explorer provides complete details about the table structure. For example, from here you can see that the column **ResellerKey** in **FactResellerSales** and the column **ResellerKey** of **DimReseller** are the links on which the two tables can join together.

Create a new query

Let's try a custom query. Let's start by querying the fact table, which holds the central data.

1. With the **FactResellerSales** table selected in the tree, click the **New Query** toolbar button to open a new query editor.
 - You may get a warning about unsupported features such as transactions. That's ok - just dismiss the warning.
2. Type the following SQL into the editor to retrieve the sales information.

```
SELECT S.[SalesAmount] AS [Sales]
FROM [FactResellerSales] AS S
```

Notice the use of aliases in this query. The table **FactResellerSales** is aliased as **S** so that it is easier to reference in other parts of the query. The column **SalesAmount** is aliased as **Sales** to change its name in the output.

1. Click the Execute button in the toolbar (or press **<kbd>F5</kbd>**); the query will be sent to the ASDW server, executed, and results will be sent back and displayed in the bottom pane of the query window.

```
SELECT S.[SalesAmount] AS [Sales]
FROM [FactResellerSales] AS S
```

| | Sales |
|----|-----------|
| 1 | 28.8404 |
| 2 | 404.664 |
| 3 | 201.8828 |
| 4 | 551.8146 |
| 5 | 183.9382 |
| 6 | 59.97 |
| 7 | 6074.982 |
| 8 | 1445.1898 |
| 9 | 3728.5554 |
| 10 | 2097.2945 |
| 11 | 1260.2767 |

- Next, let's bring the dimension table in using the shared column. Type the following query into the editor.

```
SELECT S.[SalesAmount] AS [Sales], R.[BusinessType], R.[ResellerName]
FROM [FactResellerSales] AS S
JOIN [DimReseller] AS R ON S.[ResellerKey] = R.[ResellerKey]
```

The fact table and dimension table are joined together by the ResellerKey columns which provides the relationship between them.

- Execute the query. Here are the first few rows ..

| Sales | BusinessType | ResellerName |
|----------|----------------------|---|
| 90.8393 | Warehouse | Bike Dealers Association |
| 173.0424 | Warehouse | Retail Mall |
| 20.1865 | Specialty Bike Shop | Professional Containers and Packaging Co. |
| 151.3989 | Warehouse | Real Sporting Goods |
| 144.202 | Value Added Reseller | Commerce Bicycle Specialists |
| 288.404 | Warehouse | Original Bicycle Supply Company |
| 86.5212 | Specialty Bike Shop | Roving Sports |
| 20.1865 | Value Added Reseller | Frugal Bike Shop |
| 60.5596 | Value Added Reseller | Great Bikes |
| 230.7232 | Warehouse | Go-cart and Bike Specialists |

```
... remainder omitted for length
```

By observing the values in the `BusinessType` column, we find out the resellers fall into three types of businesses: **Specialty Bike Shop**, **Warehouse**, and **Value Added Reseller**. If you want to focus on *warehouse resellers* you can add a `WHERE` condition to the source data.

1. Add a `WHERE` clause to the query, it should look something like this:

```
SELECT S.[SalesAmount] AS [Sales], R.[BusinessType], R.[ResellerName]
FROM [FactResellerSales] AS S
JOIN [DimReseller] AS R ON S.[ResellerKey] = R.[ResellerKey]
WHERE R.[BusinessType] = 'Warehouse'
```

1. Execute the query to limit the results.
2. Since there are only *warehouse resellers* left, we don't need the `BusinessType` column in the query result, go ahead and remove it.

```
SELECT S.[SalesAmount] AS [Sales], R.[ResellerName]
FROM [FactResellerSales] AS S
JOIN [DimReseller] AS R ON S.[ResellerKey] = R.[ResellerKey]
WHERE R.[BusinessType] = 'Warehouse'
```

Performing aggregate calculations

Most of the data analysis tasks involve aggregation calculations, including min/max values, totals, summations, and averaging data.

For example, if you want to know the sales performance of each reseller, you would calculate the total sales amount for each of the resellers. In T-SQL, this can be done by the `GROUP BY` clause and the `SUM` aggregation function.

1. Type the following query into the editor.

```
SELECT SUM(S.[SalesAmount]) AS [Sales], R.[ResellerName]
FROM [FactResellerSales] AS S
JOIN [DimReseller] AS R ON S.[ResellerKey] = R.[ResellerKey]
WHERE R.[BusinessType] = 'Warehouse'
GROUP BY R.[ResellerName]
```

This returns the following results:

| Sales | ResellerName |
|-------------|----------------------------|
| 284876.0711 | Bike Boutique |
| 186217.0681 | Excellent Bikes |
| 8406.4692 | Unusual Bicycle Company |
| 373658.4657 | Catalog Store |
| 17623.2972 | Grown-up Bike Store |
| 1538.172 | Good Bicycle Store |
| 3385.3678 | Lots of Bikes Storehouse |
| 268209.286 | Commercial Sporting Goods |
| 147887.682 | Utilitarian Sporting Goods |
| 2917.9404 | Expert Sports Store |

```
... remainder omitted for length
```

Filtering aggregated results

Based on the aggregated data, you may want only to keep the resellers whose total sales amount are greater than \$700,000. In this case, the WHERE clause won't help since it can only be used to filter the *source data*. While the aggregated total sales amount does not exist in the source data; it *is* part of the query result. To filter the query result, we can add a HAVING clause to the query:

1. Adjust the query to include a HAVING clause. It should look like this:

```
SELECT SUM(S.[SalesAmount]) AS [Sales], R.[ResellerName]
FROM [FactResellerSales] AS S
JOIN [DimReseller] AS R ON S.[ResellerKey] = R.[ResellerKey]
WHERE R.[BusinessType] = 'Warehouse'
GROUP BY R.[ResellerName]
HAVING SUM(S.[SalesAmount]) > 700000
```

TIP

Notice that the `SUM(S.[SalesAmount])` aggregation appears twice - both in `SELECT` and `HAVING`. So far, T-SQL does not support using a column alias in the `HAVING` clause.

Sorting results

Notice that the results are not sorted - we need to fix that.

To sort the query result, use the `ORDER BY` clause. With an optional `ASC` argument, the `ORDER BY` clause will sort the query result in ascending order. In contrast, using `ORDER BY` with the `DESC` argument will sort the query result descending. To highlight the high-performance resellers, let's sort the query result in descending order.

1. Change the query to look like the following.

```
SELECT SUM(S.[SalesAmount]) AS [Sales], R.[ResellerName]
FROM [FactResellerSales] AS S
JOIN [DimReseller] AS R ON S.[ResellerKey] = R.[ResellerKey]
WHERE R.[BusinessType] = 'Warehouse'
GROUP BY R.[ResellerName]
HAVING SUM(S.[SalesAmount]) > 700000
ORDER BY [Sales] DESC
```

1. Run the query; you will see the descended sorted result.

Selecting the top section of data

What if you need to find the "Top 10" best resellers? In SQL, we can use the `TOP` argument of `SELECT`.

1. Remove the `HAVING` clause, and add `TOP 10` just between the `SELECT` keyword and `SUM` function call. The query should look something like:

```
SELECT TOP 10 SUM(S.[SalesAmount]) AS [Sales], R.[ResellerName]
FROM [FactResellerSales] AS S
JOIN [DimReseller] AS R ON S.[ResellerKey] = R.[ResellerKey]
WHERE R.[BusinessType] = 'Warehouse'
GROUP BY R.[ResellerName]
ORDER BY [Sales] DESC
```

1. Execute the query, and you should see the top 10 best warehouse resellers are listed. Note, the `TOP` filtering occurs later than `ORDER BY`. This is why the result is the top 10 best warehouse resellers.

| Sales | ResellerName |
|-------------|---------------------------|
| 841908.7708 | Vigorous Exercise Company |
| 799277.8953 | Retail Mall |
| 746317.5293 | Outdoor Equipment Store |
| 730798.7139 | Health Spa, Limited |
| 727272.6494 | Fitness Toy Store |
| 643745.896 | Metropolitan Equipment |
| 636226.4703 | Eastside Department Store |
| 602559.8926 | Top Sports Supply |
| 585516.433 | Golf and Cycle Store |
| 580222.3286 | Registered Cycle Store |

Exporting results

You can export results right from the query window.

- Copy as text
 - Copy as text with headers
 - Save results to a file (typically comma-delimited)
1. Right-click the data grid of the query result, and click the **Select All** menu item.
 2. Then, right-click the data grid again, and click the **Copy with Headers** menu item.

You can then paste these results wherever necessary. For example, you could paste results to an Excel sheet. Then you can visualize the data either with a pie chart or a bar chart to get some business insight. You can even use PivotTable of Excel to go further. However, this is beyond the scope of this course.

Querying Azure Synapse Analytics with Power BI.

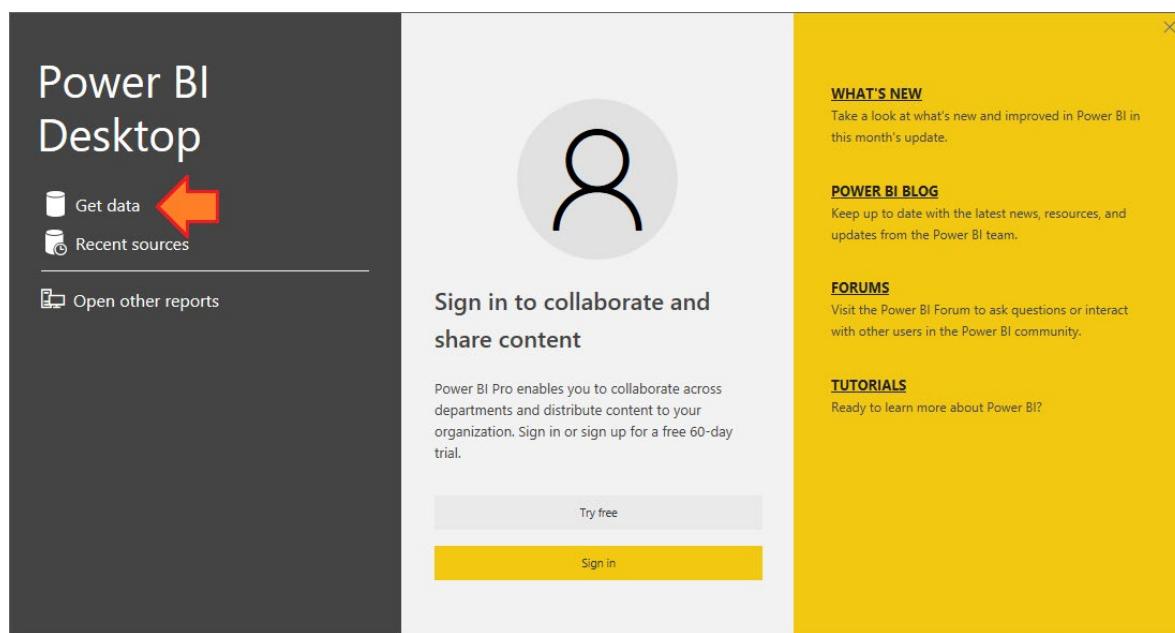
Power BI is a powerful tool dedicated to data modeling, data analysis, and data visualization. As a final exercise, let's take a look at how to connect to the ASDW database from Power BI, then query the database to generate a modern dashboard with a few clicks.

NOTE

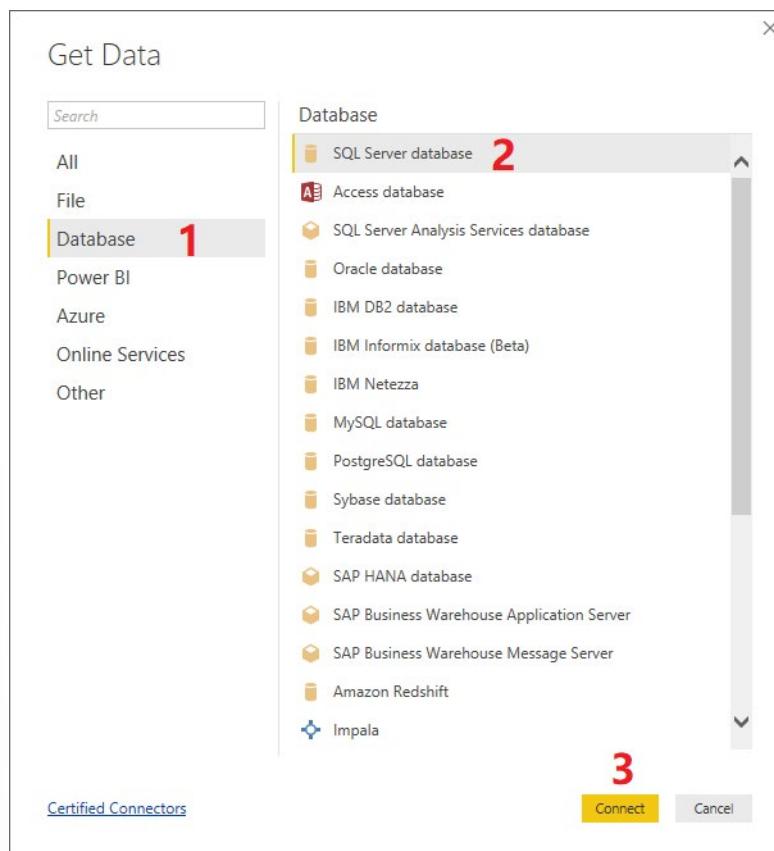
If you don't have the Power BI tools for Windows installed, please go to **Power BI**⁶ to download and install.

1. Start the Power BI tool. Click the **Get data** button on the left.

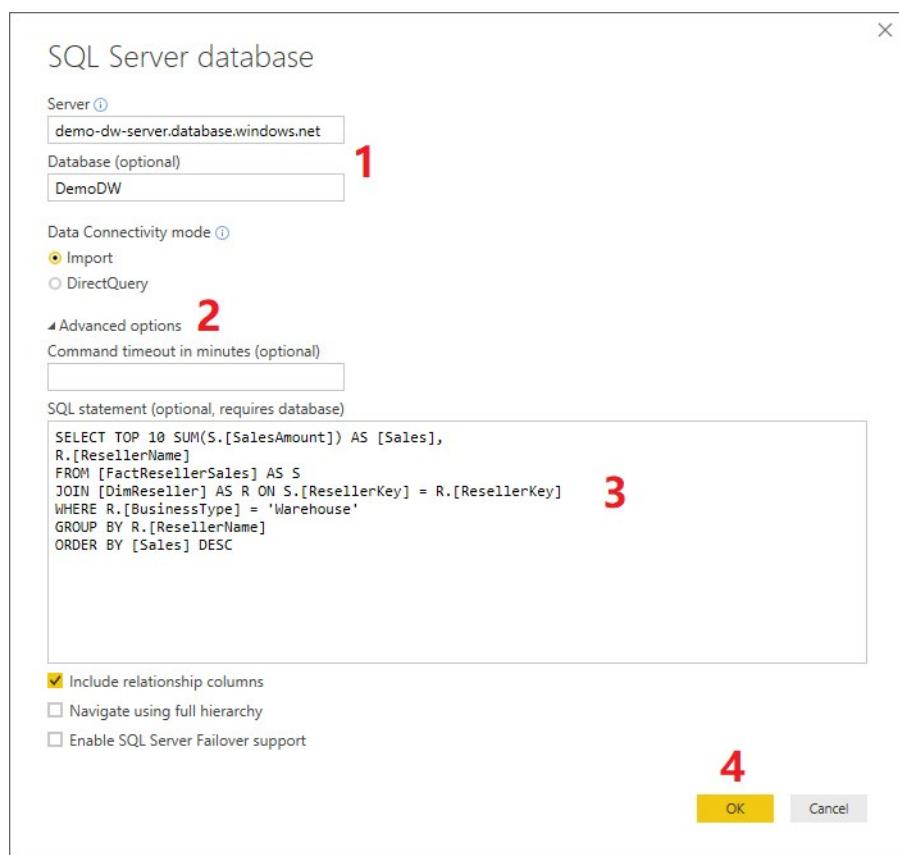
⁶ <https://powerbi.microsoft.com/>



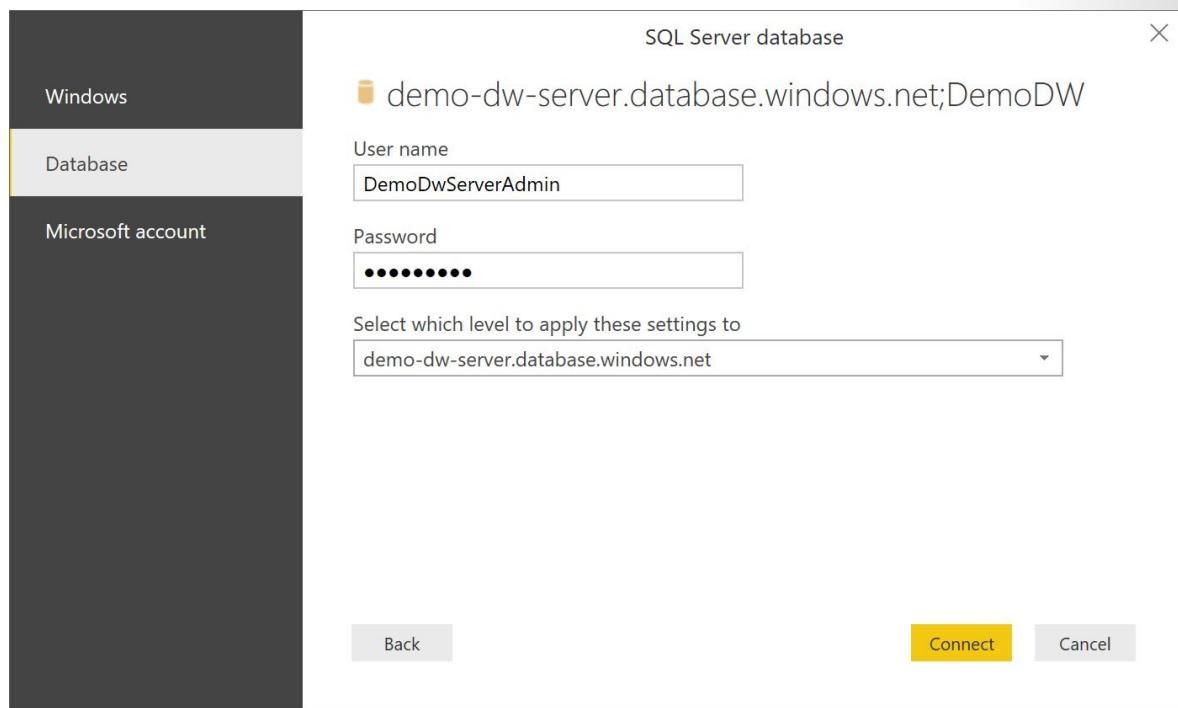
2. In the Get Data dialog, select the **Database** category, then select the **SQL Server database**, then click the **Connect** button.



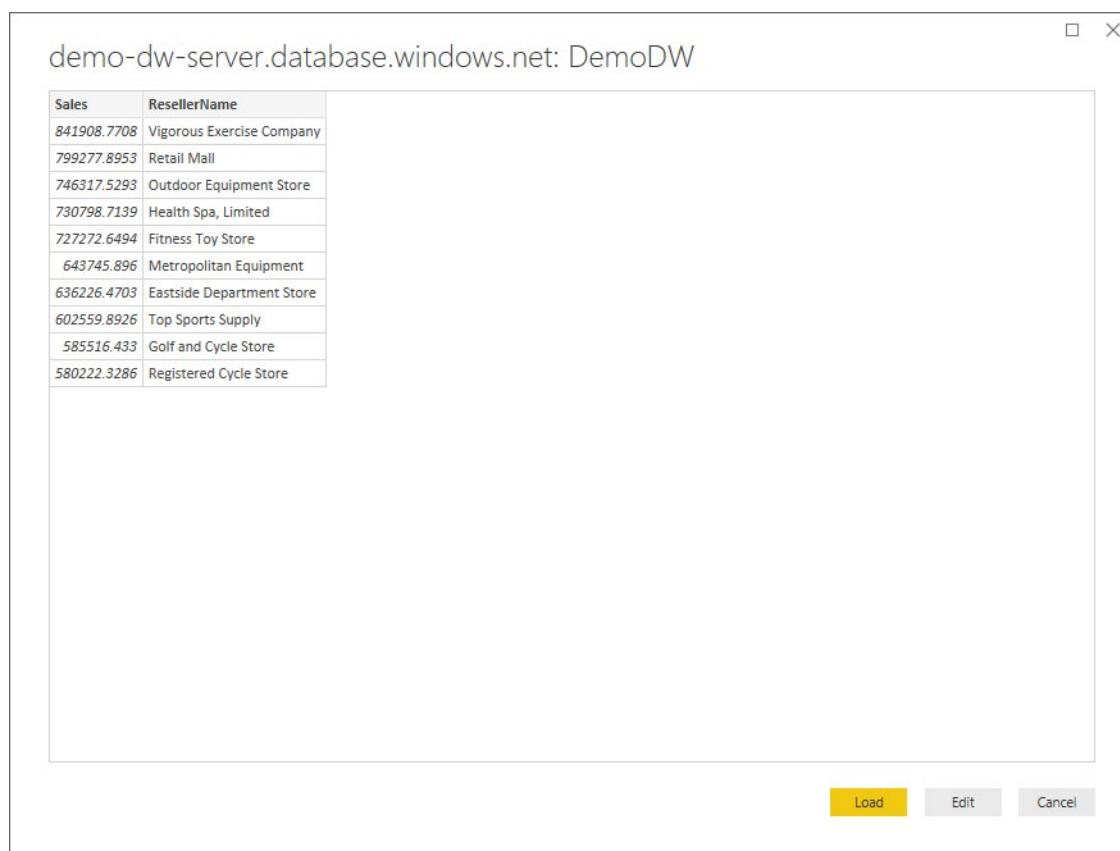
3. In the SQL Server database dialog, fill in the ASDW server name and the sample database name. Then expand the **Advanced options** node, input the last T-SQL query into the **SQL statement** textbox. Then click the **OK** button.



4. You will be asked for the ASDW username and password. After filling in, click the **Connect** button.



Since we are trying to connect the ASDW with a SQL query, once the connection is established, the SQL query will be executed immediately. Then, you will see the query result as below:



5. Click **Load** button, the data of the query result will be loaded to the data model maintained by Power BI.
6. After the query result is loaded to the Power BI model, the Power BI data model is generated from the schema of the query result. To review the data model, click the **Data** button on the left side.

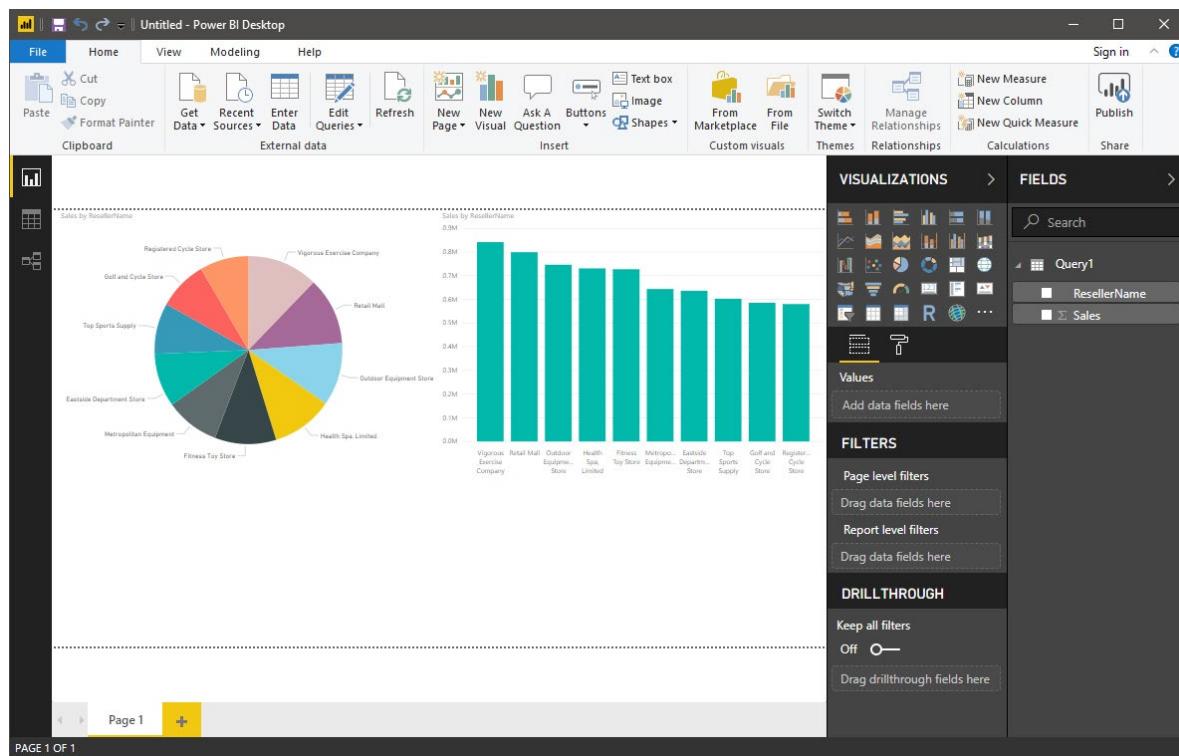
The screenshot shows the Power BI Desktop interface with a table visualization titled "Query1" containing 10 rows of data. The Fields pane on the right shows the columns "ResellerName" and "Sales".

| | ResellerName |
|-------------|---------------------------|
| 841908.7708 | Vigorous Exercise Company |
| 799277.8953 | Retail Mall |
| 746317.5293 | Outdoor Equipment Store |
| 730798.7139 | Health Spa, Limited |
| 727272.6494 | Fitness Toy Store |
| 643745.896 | Metropolitan Equipment |
| 636226.4703 | Eastside Department Store |
| 602559.8926 | Top Sport Supply |
| 585516.433 | Golf and Cycle Store |
| 580222.3286 | Registered Cycle Store |

- Click the **Report** button and switch back to the dashboard view. Click the pie chart icon of and vertical bar chart. This will insert two empty charts into the dashboard. Adjust the size and position of the empty charts until it looks good.

The screenshot shows the Power BI Desktop interface in report mode, displaying a dashboard with two charts: a pie chart and a bar chart. The Fields pane on the right shows the columns "ResellerName" and "Sales".

8. Then, select each empty chart, and check ResellerName and Sales checkbox in the **FIELDS** panel for the chart.



Congratulations! You now have a modern dashboard which is visualizing the query result from an Azure Synapse Analytics database!

Summary

In this module, you have explored how to create a sample database with Azure Synapse Analytics and how to connect to it using SQL Server Management Studio. We then explored simple queries that can be issued against the data warehouse and examined how the data can be visualized in Power BI.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

What is the default port for connecting to an enterprise data warehouse in Azure Synapse Analytics?

- TCP port 1344
- UDP port 1433
- TCP port 1433

Question 2

The following query is to retrieve the sales by business reseller, but the performance of the query is slow. The query is as follows:

```
SELECT
S.[SalesAmount] AS [Sales],
R.[BusinessType],
R.[ResellerName]
FROM [FactResellerSales] AS S
JOIN [DimReseller] AS R
ON S.[ResellerKey] = R.[ResellerKey].
```

The tables referenced within the query are configured with a distribution of Round_Robin with a clustered columnstore index. The Data Engineer wants to improve the performance of the query. What operation can be used to improve the performance of the query?

- Remove the CLUSTERED COLUMNSTORE INDEX for both tables.
- Change the Distribution to HASH(GeographyKey) for both tables.
- Change the Distribution to HASH(ResellerKey) for both tables.

Using PolyBase to Load Data into Azure Synapse Analytics

Using PolyBase to Load Data into Azure Synapse Analytics

One of the key benefits of using Azure Synapse Analytics is the fast loading of data. Importing data can be either a manual process that requires an understanding of PolyBase and the necessary steps to transfer data. More commonly however, you will use a data ingestion tool such as Azure Data Factory to schedule this data movement on a regular basis. Once understood, the loading of Terabytes or even Petabytes of data can take minutes instead of hours.

Suppose you want to resolve a long-running data load that is delaying the daily population of an on-premises data warehouse within Contoso. You've decided that the best approach to solve this problem is to create an Azure Synapse Analytics and use PolyBase to perform data loads. Your goal is to explore and understand the steps required so that you can explain it to the rest of the data team at Contoso.

Learning Objectives

In this module you will:

- Explore how PolyBase works
- Upload text data to Azure Blob store
- Collect the security keys for Azure Blob store
- Create an Azure Synapse Analytics server
- Import data from Blob Storage to the Azure Synapse Analytics

Introduction to PolyBase

The data warehouse component of Azure Synapse Analytics is a relational big data store that uses a massively parallel processing (MPP) architecture. It takes advantage of the on-demand elastic scale of Azure compute and storage resources to load and process Petabytes of data - giving you quicker access to the critical information needed to make good business decisions.

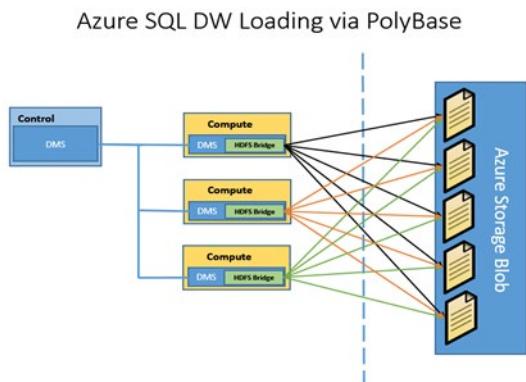
NOTE

There are two versions of compute nodes that are available at the time of writing. Both Generation 1 (Gen1) and Generation 2 (Gen2) can have **60 compute nodes** assigned to process data when the maximum Data Warehouse Unit (DWU) is selected. Gen2 is a newer architecture that has five times the compute capacity and four times the concurrent queries of Gen1.

A key feature of Azure Synapse Analytics is that you only pay for the processing you need. You can decide how much parallelism is needed for your work and you can pause the compute nodes when it's not in use, so you only pay for the CPU time you use.

Azure Synapse Analytics supports many loading methods including non-PolyBase options such as BCP and the SQL Bulk Copy API. However, the fastest and most scalable way to load date is through PolyBase. PolyBase is a technology that accesses external data stored in Azure Blob storage, Hadoop, or Azure Data Lake Store via the Transact-SQL language.

The following architecture diagram shows how this is achieved with each HDFS bridge of the Data Movement Service (DMS) service on every Compute node connecting to an external resource such as Azure Blob Storage. PolyBase then bidirectionally transfers data between SQL Data Warehouse and the external resource providing the fast load performance.



Using PolyBase to extract, load and transform data

The steps for implementing a PolyBase ELT for SQL Data Warehouse are:

1. Extract the source data into text files.
2. Load the data into Azure Blob storage, Hadoop, or Azure Data Lake Store.
3. Import the data into SQL Data Warehouse staging tables using PolyBase.
4. Transform the data (optional).
5. Insert the data into production tables.

Let's look more closely at the import process defined by steps 1-3.

Create an Azure Blob Storage account to host the files

Next, let's create an Azure Storage account and Blob storage container to hold the data we want to import into our data warehouse.

NOTE

This exercise is optional. If you don't have an Azure account, or prefer not to do the exercise in your account, you can read through the instructions to understand the steps involved in creating the Blob container and associated storage account.

Create the Azure Storage account

1. Sign into the [Azure portal⁷](https://portal.azure.com) with your Azure account.
2. Click **Create a Resource** in the left sidebar.
3. Select **Storage > Storage account**.

⁷ <https://portal.azure.com>

The screenshot shows the Azure Storage account 'demodwstorage - Blobs'. On the left, there's a sidebar with links: Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The main area has a search bar 'Search (Ctrl+ /)' and buttons for Container, Refresh, Delete, and Change access level. It displays the storage account information: 'Storage account: demodwstorage'. Below that is another search bar 'Search containers by prefix'. A table lists the container 'data-files': NAME (data-files), LAST MODIFIED (2/1/2019, 8:54:00 AM), PUBLIC ACCESS L... (Blob), and LEASE STATE (Available). There's also a '...' button.

4. On the **Basics** tab, select your subscription and use the **mslearn-demodw** Resource group.
5. Name your storage account **demodwstorage**.
6. Select the same *Location* you placed your database in.
7. Click **Review + create** to validate the account details.
8. Once validation passes, click **Create** to start the deployment process.

Wait for the deployment to complete. You should get a popup notification from the browser, or you can click the **Notification** icon in the toolbar to monitor the deployment progress.

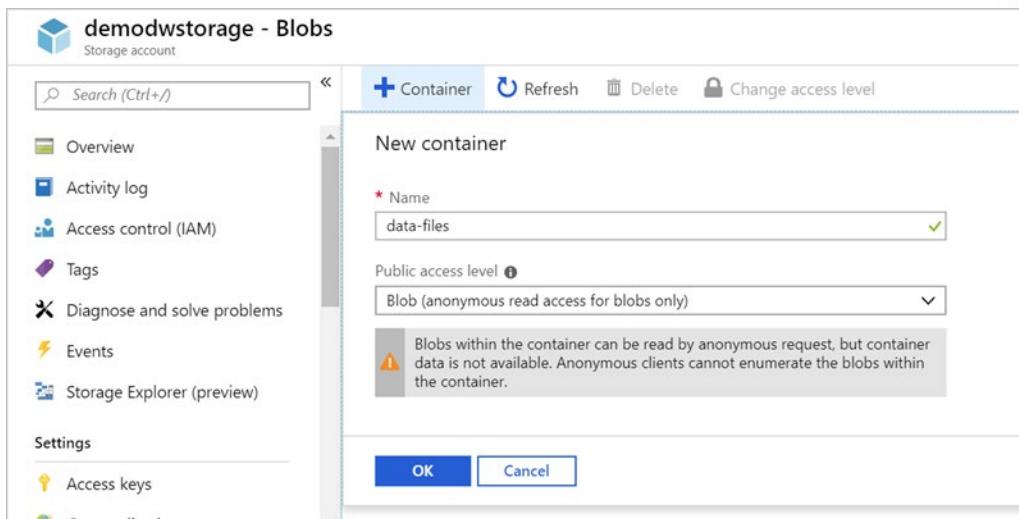
Create the Blob container for our import data

Next, we need to create a blob container that will hold our source data.

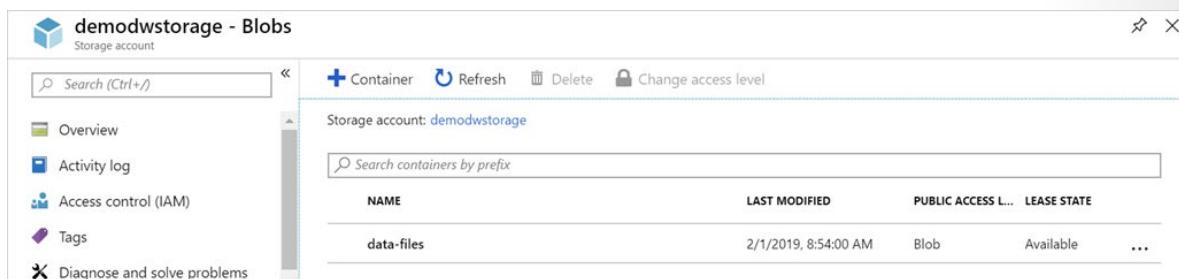
1. Select **Go to resource** from the deployment page, or locate your new storage account using the search bar at the top of the window.
2. Select **Blobs** in the **Services** section of the **Overview** page.

The screenshot shows the 'Services' section of the Azure portal. It lists two services: 'Blobs' and 'Files'. The 'Blobs' service is highlighted with a red box. The 'Blobs' card contains the text: 'REST-based object storage for unstructured data', 'Explore data using Azure AD preview', and 'Learn more'. The 'Files' service is shown next, with its own card.

3. You need a container to store the files, click **+ Container** from the top menu.
4. Name the container **data-files**.
5. Set the *Public access* to **Blob (anonymous read access for blobs only)**.



6. Click **OK** to create the container.



Upload text Data into Azure Blob Store

PolyBase can read data from several file formats and data sources. Before uploading your data into Azure Synapse Analytics, you must prepare the source data into an acceptable format for Polybase. These include:

- Comma-delimited text files (UTF-8 and UTF-16).
- Hadoop file formats including RC files, Optimized Row Columnar (ORC) files, and Parquet files.
- Gzip and Snappy compressed files.

If the data is coming from a relational database such as Microsoft SQL Server, you'll need to retrieve the data and store it in text files and then load it into an acceptable data store such as an Azure Blob storage account. Tools such as SQL Server Integration Services can ease this transfer process.

Let's take some sample data and upload it to our Blob storage container.

Obtain the source data

Our data processing department has exported the data we need into a comma-delimited file. Let's import that data into the Blob container we created earlier.

Start by downloading the **sample data file**⁸ file onto your local computer. The file contains time and data information with details about each entry.

The text file is named `DimDate2.txt` and has 1188 rows/lines of data. Each line of data is separated into 12-column values using commas. Here's a sample of the data file:

```
2011-01-01 00:00:00.000,20110102,1,Jan,Q1,2011,2011-Q1,2011-Jan,2011-01,1,Sun,2
2011-01-02 00:00:00.000,20110103,1,Jan,Q1,2011,2011-Q1,2011-Jan,2011-01,2,Mon,3
2011-01-03 00:00:00.000,20110104,1,Jan,Q1,2011,2011-Q1,2011-Jan,2011-01,3,Tue,4
2011-01-04 00:00:00.000,20110105,1,Jan,Q1,2011,2011-Q1,2011-Jan,2011-01,4,Wed,5
...
...
```

Import data into Blob storage

Let's upload it into the blob container.

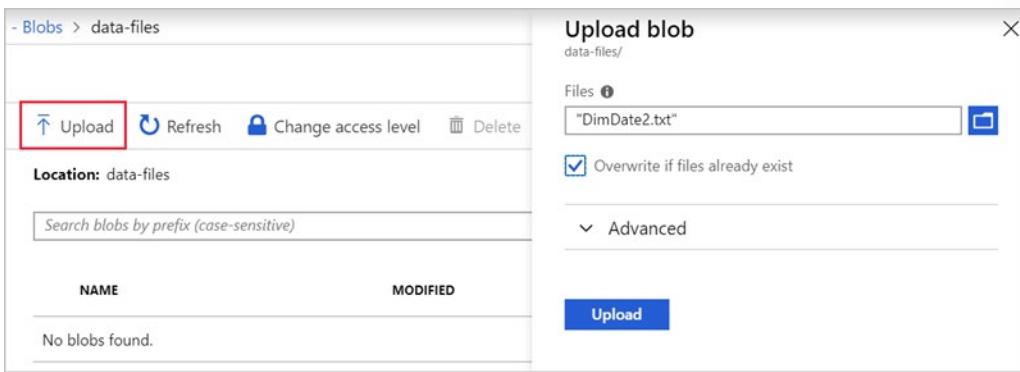
1. Sign into the **Azure portal**⁹ with the Azure account you created the storage account in.
2. Click **All resources** and in the *Search* box type **demodwstorage**. Select your storage account from the results.
3. Select **Blobs** from the **Blob service** section in the storage account.

The screenshot shows the Azure Storage Account blobs blade. At the top, there is a search bar labeled "Search (Ctrl+/" and several action buttons: "Container" (with a plus icon), "Refresh" (with a circular arrow icon), "Delete" (with a trash bin icon), and "Change access level" (with a lock icon). Below these is a section titled "Storage account: demodwstorage". A search bar here is labeled "Search containers by prefix". The main area displays a table with two columns: "NAME" and "LAST MODIFIED". There is one row in the table, which is highlighted with a blue background: "data-files" (with a checked checkbox) and "2/1/2018". On the left side, there is a sidebar with various options: "Advanced Threat Protection ...", "Static website", "Properties", "Locks", "Automation script", "Blob service" (which is selected and highlighted in blue), and "Custom domain".

4. Click on the container named **data-files** to open it.
5. Click the **Upload** icon. In the Upload blob blade on the right, browse and select the **DimDate2.txt** file you downloaded.

⁸ <https://raw.githubusercontent.com/MicrosoftDocs/mslearn-implement-azure-sql-data-warehouse/master/import-data-into-asdw-with-polybase/DimDate2.txt>

⁹ <https://portal.azure.com?azure-portal=true>



6. Once it's uploaded, you can close the upload blade and you should see the file in your blob container.

TIP

While this example uses a single text file to upload data, it's a best practice to split up the data between data files of equal size that match the number of compute nodes in your Data Warehouse. That way you gain full parallelism of all the text files against each available compute node. For example, if you use a **Gen1 - DWU6000** or **Gen2 - DW30000c** configuration, you can import 60 text files in parallel as there are 60 nodes.

Obtain the Azure Storage URL and Key

Clients and applications must be authenticated to connect to an Azure Blob storage account. We have several ways to do this, but the easiest approach for trusted applications is to use the *storage key*. Since we're managing the import process, let's use this approach.

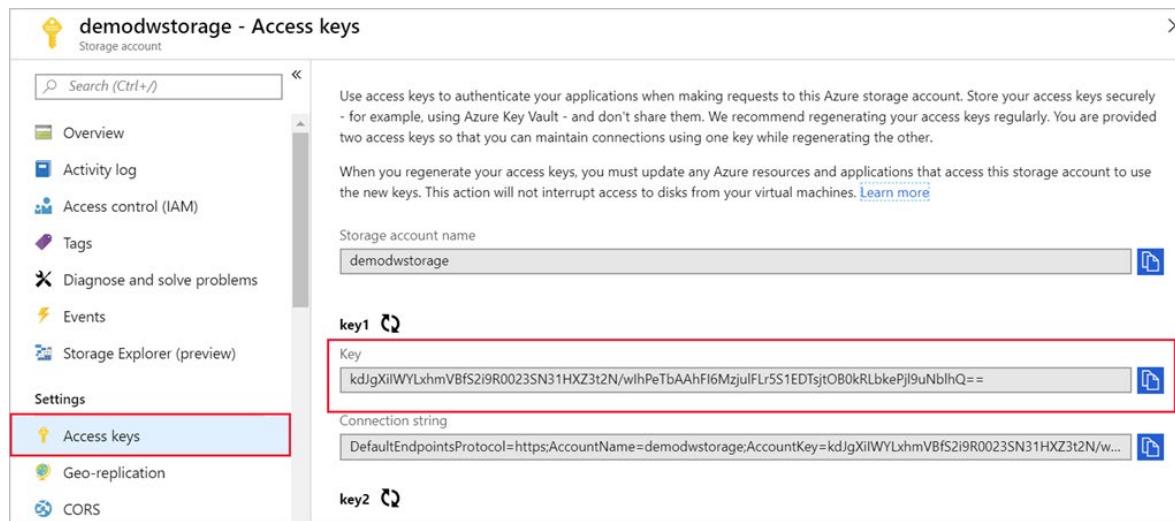
We'll need two pieces of information to connect PolyBase to our Azure Storage account:

1. URL of the storage account
2. Private storage key

We can get both of these values from the Azure portal.

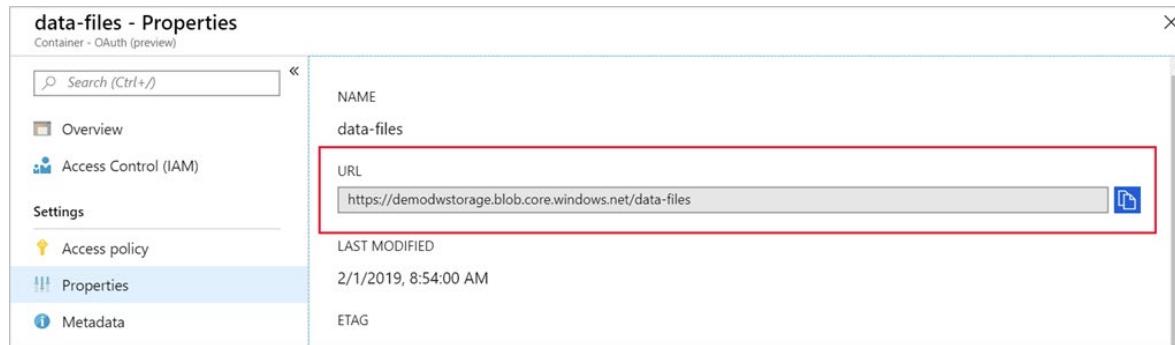
1. Sign in to the **Azure portal**¹⁰.
2. In the Storage account, under **Settings**, select **Access keys**.
3. In the Access keys blade, copy the value of **key1** and save it to a temporary file - we'll use it in the next unit.

¹⁰ <https://portal.azure.com?azure-portal=true>



The screenshot shows the 'Access keys' page for an Azure storage account named 'demodwstorage'. The left sidebar lists various account settings like Overview, Activity log, and Tags. The 'Access keys' option is selected and highlighted with a red box. On the right, there's a brief description of access keys, a note about regenerating them, and a text input field for the storage account name. Below this, two key entries are shown: 'key1' with its value 'kdJgXiiWYLxhmVBfs2i9R0023SN31HZ3t2N/wlhPeTbAAhFl6MzjuFLr5S1EDTsjtOB0kRLbkePjI9uNbLhQ==' highlighted with a red box, and 'key2'.

4. Next, navigate to your blob container (use the **Blobs** section of the storage account and then select the **data-files** container).
5. Select the **Properties** option under **Settings**.
6. Copy the **URL** to the container - it should be something like `https://demodwstorage.blob.core.windows.net/data-files`.



The screenshot shows the 'Properties' page for a blob container named 'data-files'. The left sidebar lists options like Overview, Access Control (IAM), Settings, Access policy, Properties (which is selected and highlighted with a red box), and Metadata. The main area displays container properties: NAME (data-files), URL (highlighted with a red box and containing the value 'https://demodwstorage.blob.core.windows.net/data-files'), LAST MODIFIED (2/1/2019, 8:54:00 AM), and ETAG.

7. Save the URL into the same text file.

IMPORTANT

The Storage account access keys allow *full-access* to the contents in the storage account. Never disclose these keys to untrusted parties or third-party applications. Check out the **Secure your Azure Storage account¹¹** module for more information on managing the security of your storage account.

Importing Data from Azure Blob Store into Azure Synapse Analytics

You can now import the data from the blob storage to the Azure Synapse Analytics database. Let's connect to the database and execute the appropriate SQL queries to create a staging table with our data.

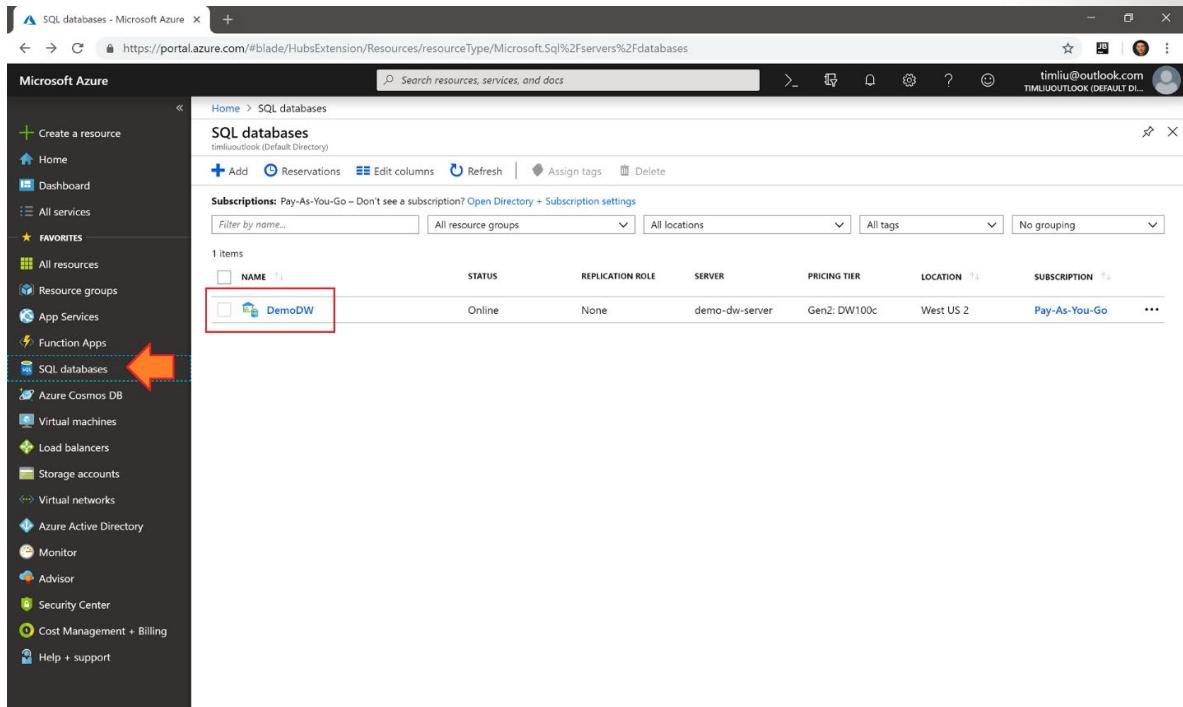
Note For this section, we have assumed that you now know how to provision an Azure Synapse Analytics as shown in a previous section should one be required

¹¹ <https://docs.microsoft.com/learn/modules/secure-azure-storage-account/>

Open the query editor in the Azure portal

We're going to use a built-in query editor in the Azure portal to execute the necessary queries. However any query tool that supports connecting to an Azure Synapse Analytics instance can be used. Some common free tools you can use on Windows are Visual Studio and SQL Server Management Studio (SMSS). On Linux and macOS, you can use Visual Studio Code with the **mssql** extension.

1. Sign into the **Azure portal**¹².
2. Select **SQL database** in the left sidebar. If it's not present, you can search for the database by name with the search box at the top.

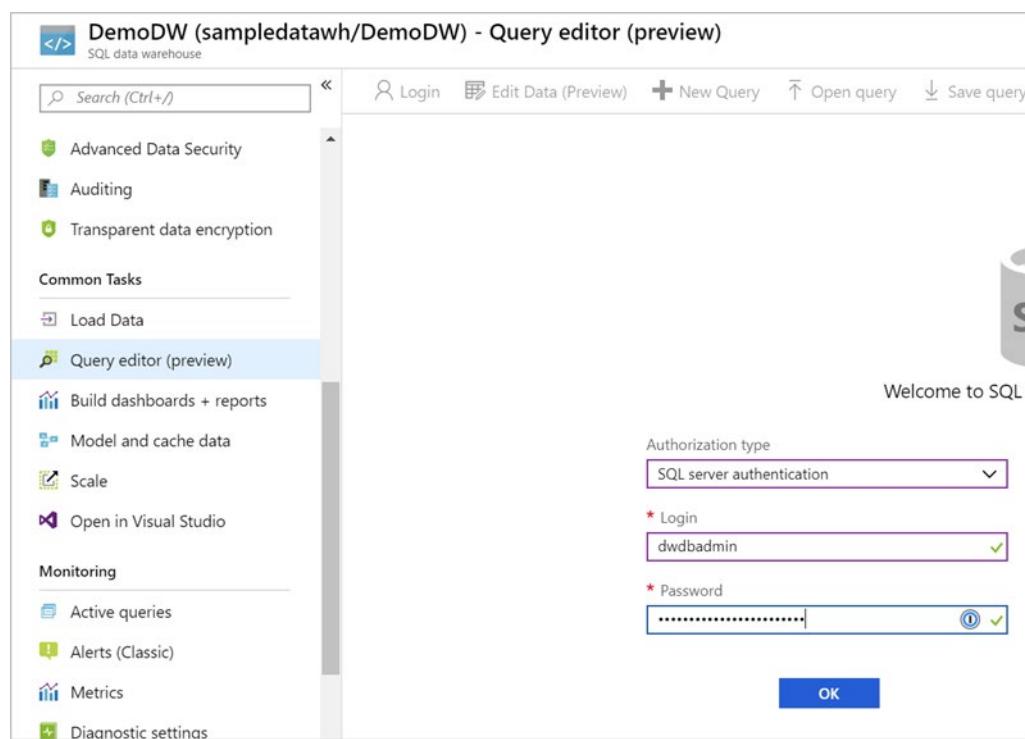


The screenshot shows the Microsoft Azure portal interface. The left sidebar has a dark theme with various service icons. An orange arrow points to the 'SQL databases' icon. The main content area is titled 'SQL databases' and shows a table with one item:

| NAME | STATUS | REPLICATION ROLE | SERVER | PRICING TIER | LOCATION | SUBSCRIPTION |
|--------|--------|------------------|----------------|--------------|-----------|---------------|
| DemoDW | Online | None | demo-dw-server | Gen2: DW100c | West US 2 | Pay-As-You-Go |

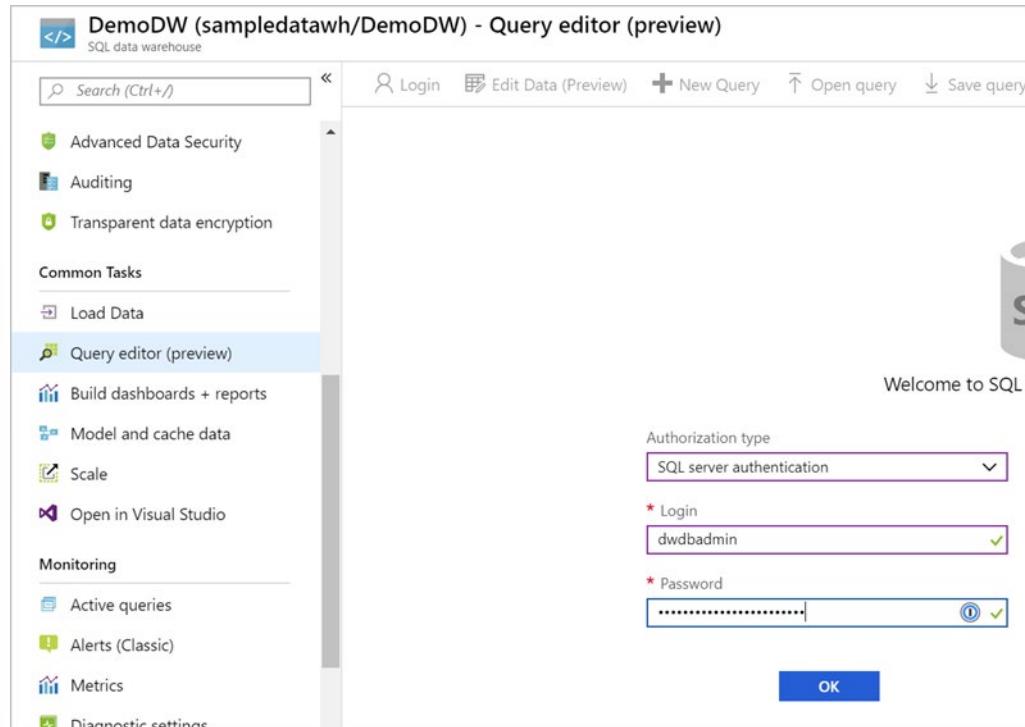
3. Select the name of the target database where you want to import the data (**DemoDW**).
4. Select **Query editor (preview)** from the **Common tools** section. This is a built-in SQL query editor.

¹² <https://portal.azure.com?azure-portal>

**TIP**

You can launch Visual Studio from here as well if you prefer to work with a desktop-based tool.

5. Enter the admin/password credentials you used to create the database with earlier and click **OK** to authenticate. You should see the query explorer.



Create an import database

The first step in using PolyBase is to create a database scoped credential that secures the credentials to the Azure Blob store. This involves creating a master key first, and then using this key to encrypt the database scoped credential named **AzureStorageCredential**.

1. Paste the code below into the query window. Make sure to replace the **SECRET** value with the access key you retrieved in the previous exercise.

```
CREATE MASTER KEY;

CREATE DATABASE SCOPED CREDENTIAL AzureStorageCredential
WITH
    IDENTITY = 'DemoDwStorage',
    SECRET = 'THE-VALUE-OF-THE-ACCESS-KEY' -- put key1's value here
;
```

1. Click the **Run** button to execute the query. It should report **Query succeeded: Affected rows: 0.**

Create external data source connection

The next step is to take the database scoped credential and use it to create an external data source named **AzureStorage**. Note the location URL point to the container you created (**data-files**) in the Azure Blob store. The Type **Hadoop** is used for both Hadoop-based and Azure blob-based access.

1. Paste the code below into the query window. Make sure to replace the **LOCATION** value with your correct value from the previous exercise.

```
CREATE EXTERNAL DATA SOURCE AzureStorage
WITH (
    TYPE = HADOOP,
    LOCATION = 'wasbs://data-files@demodwstorage.blob.core.windows.net',
    CREDENTIAL = AzureStorageCredential
);
```

1. Click the **Run** button to execute the query. It should report **Query succeeded: Affected rows: 0..**

Define the import file format

The next step is to define the external file format named **TextFile**. This indicates to PolyBase that the format of the text file is **DelimitedText** and the field terminator is a comma.

1. Paste the following code into the query window.

```
CREATE EXTERNAL FILE FORMAT TextFile
WITH (
    FORMAT_TYPE = DelimitedText,
    FORMAT_OPTIONS (FIELD_TERMINATOR = ',')
);
```

1. Click the **Run** button to execute the query. It should report `Query succeeded: Affected rows: 0..`

Create temporary table

Next, we need to create an external table named `dbo.Temp` with the column definition for your table. At the bottom of the query, you will use a `WITH` clause to call the data source definition named **AzureStorage** (defined above) and the file format named **TextFile** (defined above). The location denotes that the files for the load are in the root folder of the data source.

[!NOTE]

External tables are in-memory tables that do not persist onto the physical disk. These can be queried like any other table.

The table definition needs to match the fields defined in the input file. So we have 12 defined columns, with data types that match the input file data.

1. Add the code below into the Visual Studio window below the previous code.

```
-- Create a temp table to hold the imported data
CREATE EXTERNAL TABLE dbo.Temp (
    [Date] datetime2(3) NULL,
    [DateKey] decimal(38, 0) NULL,
    [MonthKey] decimal(38, 0) NULL,
    [Month] nvarchar(100) NULL,
    [Quarter] nvarchar(100) NULL,
    [Year] decimal(38, 0) NULL,
    [Year-Quarter] nvarchar(100) NULL,
    [Year-Month] nvarchar(100) NULL,
    [Year-MonthKey] nvarchar(100) NULL,
    [WeekDayKey] decimal(38, 0) NULL,
    [WeekDay] nvarchar(100) NULL,
    [Day Of Month] decimal(38, 0) NULL
)
WITH (
    LOCATION='/',
    DATA_SOURCE=AzureStorage,
    FILE_FORMAT=TextFile
);
```

1. Click the **Run** button to execute the query. It will take a few seconds to complete, and should report `Query succeeded: Affected rows: 0..`

Create destination table

The next step is to create a physical table in the Data Warehouse database. In the following example, you will create a table named `dbo.DimDate` that has a clustered column store index define on all the columns and use a table geometry of `round_robin`. This is by design as `round_robin` is the best table geometry to use for loading data.

1. Paste the following code into the query window.

```
-- Load the data from Azure blob storage to SQL Data Warehouse
CREATE TABLE [dbo].[StageDate]
```

```
WITH (
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = ROUND_ROBIN
)
AS
SELECT * FROM [dbo].[Temp];
```

1. Click the **Run** button to execute the query. It will take a few seconds to complete, and should report Query succeeded: Affected rows: 0..

Add statistics onto columns to improve query performance

Finally, as an optional step, you can create statistics on columns that feature in queries to improve the query performance against the table.

1. Paste the following code into the query window.

```
-- Create statistics on the new data
CREATE STATISTICS [DateKey] on [StageDate] ([DateKey]);
CREATE STATISTICS [Quarter] on [StageDate] ([Quarter]);
CREATE STATISTICS [Month] on [StageDate] ([Month]);
```

1. Click the **Run** button to execute the query. It should report Query succeeded: Affected rows: 0..

Congratulations, you've loaded your first staging table in Azure Synapse Analytics. From here, you can write further Transact-SQL queries to perform transformations into dimension and fact tables. Try it out by querying the `StageDate` table in the query explorer, or in another query tool. You can refresh the left-hand view and see your new table(s) created. In addition, you can reuse the steps above in a persistent SQL script to load additional data as necessary.

Summary

PolyBase lets you import data from a variety of formats and sources to populate your data warehouse with all the data you need to analyze. In this module you've learned how to take data from an external source, format it, import it into blob storage and then use the built-in PolyBase tools to pull it into your Azure Synapse Analytics.

You can use the techniques explored in this module to populate your Azure Synapse Analytics with whatever data you need.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Mike is the data engineer for Contoso and has a Data Warehouse created with a database named Crystal. Within the database is a table named DimSuppliers. The suppliers' information is stored in a single text file named Suppliers.txt and is 1200MB in size. It is currently stored in a container with an Azure Blob store. Your Azure Synapse Analytics is configured as Gen 2 DW30000c. How can Mike maximize the performance of the data load?

- Increase the Gen 2 DWU units.
- Split the text file into 60 files of 20MB each.
- Use Gen 1 DW6000.

Question 2

Mike is the data engineer for Contoso and has a Data Warehouse created with a database name Crystal. He has created a master key, followed by a database scoped credential. What should he create next?

- An external data source.
- An external table.
- A physical table.

Module Summary

Module Summary

In this module, you have explored the Azure relational data platform options including SQL Database and SQL Data Warehouse. You are also able explain why they would choose one service over another, and how to provision, connect and manage each of the services.

Learning objectives

In this module, you have learned how to:

- Work with Azure SQL Database.
- Work with SQL Data Warehouse.
- Provision and query data in Azure Synapse Analytics.
- Import data into Azure Synapse Analytics using PolyBase.

Post Course Review

After the course, **learn about the service, use cases, and customer stories in this data sheet.¹³**

Important

Remember

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

¹³ <https://azure.microsoft.com/en-us/resources/azure-sql-data-warehouse-datasheet/>

Answers

Question 1

Who's responsible for performing software updates on your Azure SQL databases and the underlying OS?

- You are. It's up to you to periodically log in and install the latest security patches and updates.
- Microsoft Azure. Azure manages the hardware, software updates, and OS patches for you.
- No one. Your database stays with its original OS and software configuration.

Explanation

Azure SQL databases are a Platform-as-a-Service (PaaS) offering, meaning much less maintenance to manage yourself.

Question 2

What is an Azure SQL logical server?

- An administrative container for your databases.
- Another name for an Azure SQL database instance.
- A server that defines the logical rules that sort and compare data

Explanation

vCore gives you greater control over what compute and storage resources you create and pay for. You can increase IO throughput but keep the existing amount of compute and storage. DTU, or Database Transaction Unit, provides a simple, preconfigured purchase option. To increase IO throughput, you would need to move to a higher tier that also increases your storage and compute power, things you don't need. SQL elastic pools enable you to buy a set of compute and storage resources that are shared among all the databases in the pool. This option won't help with IO performance because you're working with just one database.

Question 3

Your Azure SQL database provides adequate storage and compute power. But you find that you need additional IO throughput. Which performance model might you use?

- DTU
- vCore
- SQL elastic pool

Explanation

Azure SQL logical server is an administrative container for your databases. You can control logins, firewall rules, and security policies through the logical server. It is not another name for an Azure SQL Database instance and Collations are rules that sort and compare data.

Question 1

Which of the following terms refer to the scale of compute that is being used in an Azure SQL Synapse Analytics server?

- RTU
- DWU
- DTU

Explanation

DWU refers to Data Warehouse Units. It is the measure of compute scale that is assigned to an Azure SQL Data Warehouse. RTU is a compute scale unit of Cosmos DB. DTU is a compute scale unit of Azure SQL Database.

Question 2

You have an Azure Synapse Analytics database, within this, you have a dimension table named Stores that contains store information. There is a total of 263 stores nationwide. Store information is retrieved in more than half of the queries that are issued against this database. These queries include staff information per store, sales information per store and finance information. You want to improve the query performance of these queries by configuring the table geometry of the stores table. Which is the appropriate table geometry to select for the stores table?

- Round Robin
- Non Clustered
- Replicated table

Explanation

A replicated table is an appropriate table geometry choice as the size of the data in the table is less than 200m and the table will be replicated to every distribution node of an Azure Synapse Analytics to improve the performance. A Round Robin distribution is a table geometry that is useful to perform initial data loads. Non Clustered is not a valid table geometry in Azure Synapse Analytics.

Question 1

What is the default port for connecting to an enterprise data warehouse in Azure Synapse Analytics?

- TCP port 1344
- UDP port 1433
- TCP port 1433

Explanation

The default port for connecting to an Azure Synapse Analytics is TCP port 1433.

Question 2

The following query is to retrieve the sales by business reseller, but the performance of the query is slow. The query is as follows:

```
SELECT
    S.[SalesAmount] AS [Sales],
    R.[BusinessType],
    R.[ResellerName]
FROM [FactResellerSales] AS S
JOIN [DimReseller] AS R
ON S.[ResellerKey] = R.[ResellerKey].
```

The tables referenced within the query are configured with a distribution of Round_Robin with a clustered columnstore index. The Data Engineer wants to improve the performance of the query. What operation can be used to improve the performance of the query?

- Remove the CLUSTERED COLUMNSTORE INDEX for both tables.
- Change the Distribution to HASH(GeographyKey) for both tables.
- Change the Distribution to HASH(ResellerKey) for both tables.

Explanation

Placing Hash distribution on the ResellerKey on both the FactResellerSales and DimReseller will improve the performance of the query. Placing Hash distribution on the GeographyKey on both the FactResellerSales and DimReseller will not help the the performance of this query. Removing the CLUSTERED COLUMNSTORE INDEX for both tables would reduce the performance of this query.

Question 1

Mike is the data engineer for Contoso and has a Data Warehouse created with a database named Crystal. Within the database is a table named DimSuppliers. The suppliers' information is stored in a single text file named Suppliers.txt and is 1200MB in size. It is currently stored in a container with an Azure Blob store. Your Azure Synapse Analytics is configured as Gen 2 DW30000c. How can Mike maximize the performance of the data load?

- Increase the Gen 2 DWU units.
- Split the text file into 60 files of 20MB each.
- Use Gen 1 DW6000.

Explanation

Split the text file into 60 files of 20MB each. eparating the single text file of Suppliers.txt into 60 files can take advantage of the fact that Gen 2 DW30000c uses 60 compute nodes and the parallelism of the data load can be evenly spread for quicker performance. Increasing the Gen 2 DWU units will not work as Gen 2 DW30000c is the highest limit and cannot be scaled beyond this. Gen 1 compute nodes has less power than Gen 2 compute nodes and will not improve the performance.

Question 2

Mike is the data engineer for Contoso and has a Data Warehouse created with a database name Crystal. He has created a master key, followed by a database scoped credential. What should he create next?

- An external data source.
- An external table.
- A physical table.

Explanation

An external data source. Once the master key and the database scoped credential is created, Mike should create an external data source that contains a url to the Blob location and the name of the database scoped credential. Mike must have an external data source before he creates an external table and he must have an external data source and an external table before he can create a physical table.

Module 6 Performing Real Time Analytics with Stream Analytics

Moudle Introductions

Performing Real-Time Analytics with Stream Analytics

In this module, you will learn the concepts of event processing and streaming data and how this applies to Events Hubs and Azure Stream Analytics. The students will then set up a stream analytics job to stream data and learn how to query the incoming data to perform analysis of the data. Finally, you will learn how to manage and monitor running jobs.

Note

Internet of Things (IoT) Hubs is not covered in this course, but information can be viewed [here¹](#)

Learning Objectives

In this module, you will learn about:

- Data streams and event processing
- Data Ingestion with Event Hubs
- Processing Data with Stream Analytics Jobs

¹ https://azure.microsoft.com/en-gb/services/iot-hub/?&OCID=AID719823_SEM_kKOEH3WE&lnkd=Bing_Azure_Brand&dclid=CjkKEQiAwojKBRDxj7KPjtXT1qABEiQAmr1ulVSM3TdkMM1QgPo7DKqjBxDvsNZaw-y03oDA3O_H3JTw_wcB

Introducing Data Streams and Event Processing

Introducing Data Streams and Event Processing

As more data is generated from a variety of connected devices, sensors and applications, transforming this data into actionable insights in near real time is now an operational imperative. Azure Stream Analytics integrates your streaming data with a streaming analytics engine to gain insights with streaming data in real time.

Learning Objectives

In this module you will:

- Explain data streams
- Explain event processing
- Learn about processing events with Azure Stream Analytics

Introducing Data Streams

In the context of analytics, data streams are event data generated by sensors or other sources that can be analyzed by another technology. Analyzing a data stream is typically done to measure the state change of a component or to capture information on an area of interest. The intent being to:

- Continuously analyze data to detect issues and understand or respond to them.
- Understand component or system behavior under various conditions to fuel further enhancements of said component or system.
- Trigger specific actions when certain thresholds are identified.

In today's world, data streams are ubiquitous. Companies can harness the latent knowledge in data streams to improve efficiencies and further innovation. Examples of use cases that analyze data streams include:

- Stock market trends
- Monitoring data of water pipelines, and electrical transmission and distribution systems by utility companies
- Mechanical component health monitoring data in automotive and automobile industries
- Monitoring data from industrial and manufacturing equipment
- Sensor data in transportation such as traffic management and highway toll lanes
- Patient health monitoring data in the healthcare industry
- Satellite data in the space industry
- Fraud detection in the banking and finance industries
- Sentiment analysis of social media posts

Approaches to data stream processing

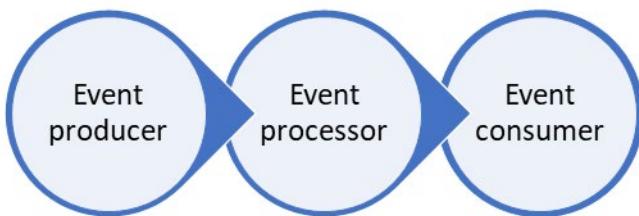
There are two approaches to processing data streams: streaming and reference data.

Reference data is streaming data that can be collected over time and persisted in storage as static data. The data can then be processed when convenient, or during times when compute costs are lower. The downside to this approach is the cost of storing the data. In contrast, streaming data have relatively low storage requirements. They also require more processing power to run computations in sliding windows over continuously incoming data to generate the insights.

Event Processing

The process of consuming data streams, analyzing them, and deriving actionable insights out of them is called Event Processing. An event processing pipeline has three distinct components:

- **Event producer:** Examples include sensors or processes that generate data continuously such as a heart rate monitor or a highway toll lane sensor.
- **Event processor:** An engine to consume event data streams and deriving insights from them. Depending on the problem space, event processors either process one incoming event at a time (such as a heart rate monitor) or process multiple events at a time (such as Azure Stream Analytics processing the highway toll lane sensor).
- **Event consumer:** An application which consumes the data and takes specific action based on the insights. Examples of event consumers include alert generation, dashboards, or even sending data to another event processing engine.



Event Processing with Azure Stream Analytics

Microsoft Azure Stream Analytics is an event processing engine. It enables the consumption and analysis of high volumes of streaming data generated by sensors, devices, or applications and processes the data in real time. A typical event processing pipeline built on top of Stream Analytics consists of four components: An event producer, an event ingestion system, the stream analytics engine, and finally the consumer.

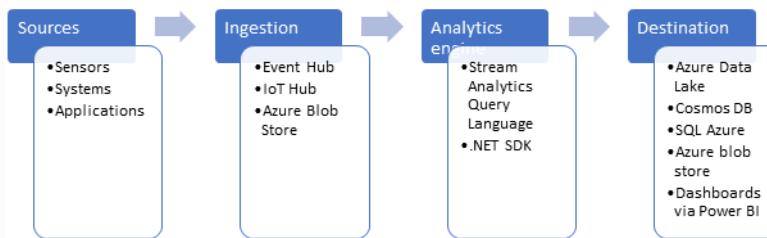
- **Event Producer:**
 - An event producer is any application, system, or sensor that continuously produces event data of interest. Examples can include a sensor that tracks the flow of water in a utility pipe to an application such as a Twitter that generates tweets against a single hashtag.
- **Event Ingestion**
 - An event ingestion system takes the data from the source system or application to pass onto an analytics engine. Azure Event Hub, Azure IoT Hub, or Azure Blobs can all serve as the ingestion system.

- **Stream Analytics Engine**

- The stream analytics engine is where compute is run over the incoming streams of data and insights extracted. Azure Stream Analytics exposes Stream Analytics Query Language (SAQL), a subset of Transact-SQL tailored to perform computations over streaming data. The engine supports Windowing functions that are fundamental to stream processing and are implemented using SAQL.

- **event consumer**

- An event consumer is a destination of the output from stream analytics engine. The target can be storage (Azure Data Lake, Cosmos DB, SQL Azure, or Azure blob store) or dashboards powered by Power BI.



Operational aspects

Stream Analytics guarantees **exactly-once** event processing and **at-least-once** event delivery, so events are never lost. It has built-in recovery capabilities in case the delivery of an event fails. Also, Stream Analytics provides built-in checkpointing to maintain the state of your job and produces repeatable results.

Because Azure Stream Analytics is a PaaS service, it is fully managed and highly reliable. With built-in integration with various sources and destinations coupled with a flexible programmability model, it enhances programmer productivity. Furthermore, since the Stream Analytics engine enables in-memory compute, it offers superior performance. All these factors contribute to low total cost of ownership (TCO) of Azure Stream Analytics.

Summary

The process of consuming data streams, analyzing them, and deriving actionable insights out of them is called Event Processing. It requires an Event producer, Event Processor, and Event Consumer. Azure Stream Analytics provides the event processing aspect to streaming the is fully managed and highly reliable.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which of the following technologies typically provide an ingestion point for data streaming in an event processing solution that uses static data as a source?

- IoT Hubs
- Event Hubs
- Azure Blob storage

Data Ingestion with Event Hubs

Data Ingestion with Event Hubs

Azure Event Hubs provides applications the capability to process large volume of streaming data and scale out during exceptionally high-demand periods as and when required. Azure Event Hubs decouple the sending and receiving messages to manage the data processing. This helps eliminate the risk of overwhelming consumer applications and prevent data loss due to any unplanned interruptions.

Learning Objectives

In this module, you will learn:

- Describe Azure Event Hubs
- Create an Event Hub
- Evaluate the performance of an Event Hub
- Configure applications to use an Event Hub

Introducing Azure Event Hubs

What is an Azure Event Hub?

Azure Event Hubs is a cloud-based, event-processing service that's capable of receiving and processing millions of events per second. Event Hubs acts as a front door for an event pipeline, where it receives incoming data and stores it until processing resources are available.

An entity that sends data to the Event Hubs is called a *publisher* and an entity that reads data from the Event Hubs is called a *consumer* or a *subscriber*. Azure Event Hubs sits between these two entities to divide the production (from publisher) and consumption (to subscriber) of an event stream. This decoupling helps to manage scenarios where the rate of event production is much higher than the consumption. The following illustration shows the role of an Event Hub.



Events

An **event** is a small packet of information (a *datagram*) that contains a notification. Events can be published individually, or in batches, but a single publication (individual or batch) can't exceed 256 KB.

Publishers and subscribers

Event publishers are any application or device that can send out events using either HTTPS or Advanced Message Queuing Protocol (AMQP) 1.0.

For publishers that send data frequently, AMQP has the better performance. However, it has a higher initial session overhead, because a persistent bidirectional socket and transport level security (TLS) or SSL/TLS has to be set up first.

For more intermittent publishing, HTTPS is the better option. Though HTTPS requires additional overhead for each request, there isn't the session initialization overhead.

NOTE

Existing Kafka-based clients, using Apache Kafka 1.0 and newer client versions, can also act as Event Hubs publishers.

Event subscribers are applications that use one of two supported programmatic methods to receive and process events from an Event Hub.

- **EventHubReceiver** - A simple method that provides limited management options.
- **EventProcessorHost** - An efficient method that we'll use later in this module.

Consumer groups

An Event Hub **consumer group** represents a specific view of an Event Hub data stream. By using separate consumer groups, multiple subscriber applications can process an event stream independently, and without affecting other applications. However, the use of multiple consumer groups is not a requirement, and for many applications, the single default consumer group is sufficient.

Pricing

There are three pricing tiers for Azure Event Hubs: Basic, Standard, and Dedicated. The tiers differ in terms of supported connections, number of available Consumer groups, and throughput. When using Azure CLI to create an Event Hubs namespace, if you don't specify a pricing tier, the default of **Standard** (20 Consumer groups, 1000 Brokered connections) is assigned.

Creating and configuring a new Azure Event Hubs

An Event Hub is an Azure resource, so your first step is to create a new hub in Azure and configure it to meet the specific requirements of your applications. There are two main steps when creating and configuring a new Azure Event Hubs. The first step is to define an Event Hubs **namespace**. The second step is to create an Event Hub in that namespace.

Defining an Event Hubs namespace

An Event Hubs namespace is a containing entity for managing one or more Event Hubs. Creating an Event Hubs namespace typically involves the following:

1. Defining namespace-level settings. Certain settings such as namespace capacity (configured using **throughput units**), pricing tier, and performance metrics are defined at the namespace level. These are applicable for all the Event Hubs within that namespace. If you don't define these settings, a default value is used: 1 for capacity and *Standard* for pricing tier.

You cannot change the throughput unit once you set it. You must balance your configuration against your Azure budget expectations. You might consider configuring different Event Hubs for different throughput requirements. For example, if you have a sales data application and you are planning for two Event Hubs, one for high throughput collection of real-time sales data telemetry and one for

infrequent event log collection, it would make sense to use a separate namespace for each hub. This way you only need to configure (and pay for) high throughput capacity on the telemetry hub.

2. Selecting a unique name for the namespace. The namespace is accessible through this url: *namespace.servicebus.windows.net*
3. Defining the following optional properties:
 - Enable Kafka. This option enables Kafka applications to publish events to the Event Hub.
 - Make this namespace zone redundant. Zone-redundancy replicates data across separate data centers with their own independent power, networking, and cooling infrastructures.
 - Enable Auto-Inflate, and Auto-Inflate Maximum Throughput Units. Auto-Inflate provides an automatic scale-up option, by increasing the number of throughput units up to a maximum value. This is useful to avoid throttling in situations when incoming or outgoing data rates exceed the currently set number of throughput units.

Azure CLI commands for creating an Event Hubs namespace

To create a new Event Hubs namespace, you will use the `az eventhubs namespace` commands. Here's a brief description of the sub-commands we will use in the exercise.

| Command | Description |
|---------------------------------|--|
| <code>create</code> | Create the Event Hubs namespace. |
| <code>authorization-rule</code> | All Event Hubs within the same Event Hubs namespace share common connection credentials. You will need these credentials when you configure applications to send and receive messages using the Event Hub. This command returns the connection string for your Event Hubs namespace. |

Configuring a new Event Hub

After the Event Hubs namespace has been created, you can create an Event Hub. When creating a new Event Hub, there are several mandatory parameters.

The following parameters are required to create an Event Hub:

- **Event Hub name** - Event Hub name that is unique within your subscription and:
 - Is between 1 and 50 characters long
 - Contains only letters, numbers, periods, hyphens, and underscores
 - Starts and ends with a letter or number
- **Partition Count** - The number of partitions required in an Event Hub (between 2 and 32). This should be directly related to the expected number of concurrent consumers. This cannot be changed after the hub has been created. The partition separates the message stream, so that consumer or receiver applications only need to read a specific subset of the data stream. If not defined, this defaults to 4.
- **Message Retention** - The number of days (between 1 and 7) that messages will remain available, if the data stream needs to be replayed for any reason. If not defined, this defaults to 7.

You can also optionally configure an Event Hub to stream data to an Azure Blob storage or Azure Data Lake Store account.

Azure CLI commands for creating an Event Hub

To create a new Event Hub with the Azure CLI, you will use the `az eventhubs eventhub` command set. Here's a brief description of the sub-commands we will be using:

| Command | Description |
|---------------------|--|
| <code>create</code> | This creates the Event Hub in a specified namespace. |
| <code>show</code> | This displays the details of your Event Hub. |

Summary

To deploy Azure Event Hubs, you must configure an Event Hubs namespace and then configure the Event Hub itself. In the next section, you will go through the detailed configuration steps to create a new namespace and Event Hub.

Configuring Applications to use Event Hubs

After you have created and configured your Event Hub, you'll need to configure applications to send and receive event data streams.

For example, a payment processing solution will use some form of sender application to collect customer's credit card data and a receiver application to verify that the credit card is valid.

Although there are differences in how a Java application is configured, compared to a .NET application, there are general principles for enabling applications to connect to an Event Hub, and to successfully send or receive messages. So, although the process of editing Java configuration text files is different to preparing a .NET application using Visual Studio, the principles are the same.

What are the minimum Event Hub application requirements?

To configure an application to send messages to an Event Hub, you must provide the following information, so that the application can create connection credentials:

- Event Hub namespace name
- Event Hub name
- Shared access policy name
- Primary shared access key

To configure an application to receive messages from an Event Hub, provide the following information, so that the application can create connection credentials:

- Event Hub namespace name
- Event Hub name
- Shared access policy name
- Primary shared access key

- Storage account name
- Storage account connection string
- Storage account container name

If you have a receiver application that stores messages in Azure Blob Storage, you'll also need to first configure a storage account.

Azure CLI commands for creating a general-purpose standard storage account

The Azure CLI provides a set of commands you can use to create and manage a storage account. We'll work with them in the next unit, but here's a basic synopsis of the commands.

[!TIP]

There are several MS Learn modules that cover storage accounts, starting in the module **Introduction to Azure Storage**.

| Command | Description |
|--|--|
| storage account create | Create a general-purpose V2 Storage account. |
| storage account key list | Retrieve the storage account key. |
| storage account show-connection-string | Retrieve the connection string for an Azure Storage account. |
| storage container create | Creates a new container in a storage account. |

Shell command for cloning an application GitHub repository

Git is a collaboration tool that uses a distributed version control model, and is designed for collaborative working on software and documentation projects. Git clients are available for multiple platforms, including Windows, and the Git command line is included in the Azure Bash cloud shell. GitHub is a web-based hosting service for Git repositories.

If you have an application that is hosted as a project in GitHub, you can make a local copy of the project, by cloning its repository using the **git clone** command. We'll do this in the next unit.

Editing files in the Cloud SHell

You can use one of the built-in editors in the Cloud Shell to modify all the files that make up the application and add your Event Hub namespace, Event Hub name, shared access policy name, and primary key.

The Cloud Shell supports **nano**, **vim** and **emacs** as well as a Visual Studio Code-like editor named **code**. Just type the name of the editor you want and it will launch in the environment. We'll use the **code** editor in the next unit.

Summary

Sender and receiver applications must be configured with specific information about the Event Hub environment. You create a storage account if your receiver application stores messages in Blob Storage. If your application is hosted on GitHub, you have to clone it to your local directory. Text editors, such as **nano** are used to add your namespace to the application.

Evaluating the Performance of Event Hubs

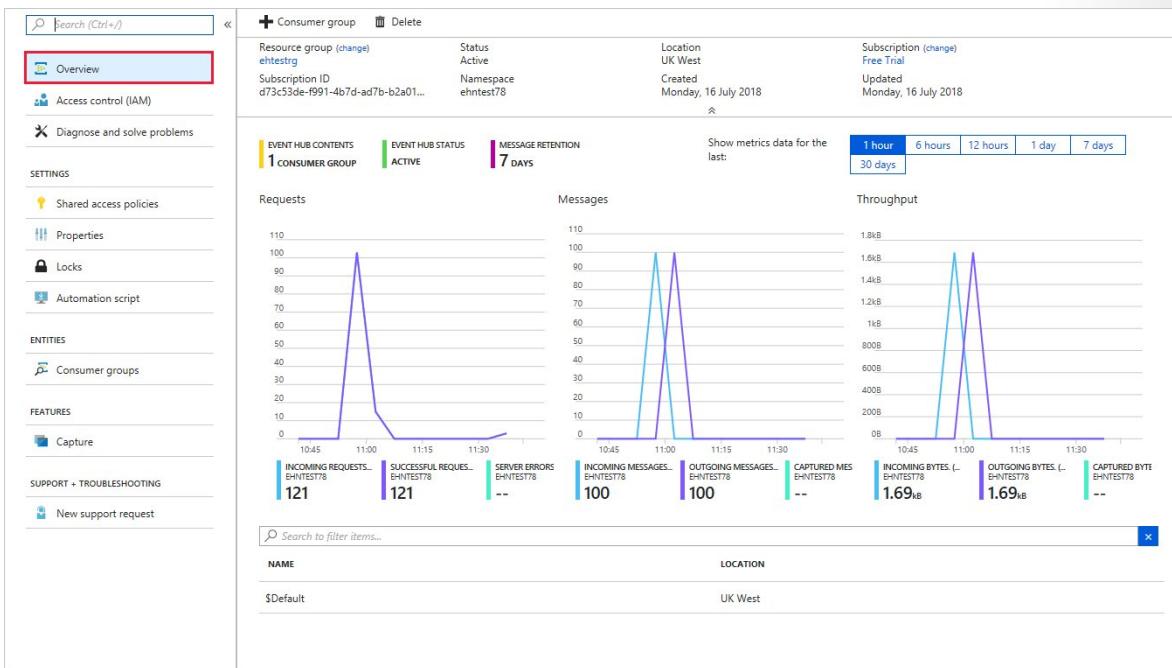
When using Event Hubs, it's crucial for you to monitor your hub to ensure that it's working and performing as expected.

Continuing with the banking example, suppose that you've deployed Azure Event Hubs and configured sender and receiver applications. Your applications are ready for testing the payment processing solution. The sender application collects customer's credit card data and the receiver application verifies that the credit card is valid. Due to the sensitive nature of your employer's business, it's essential that your payment processing is robust and reliable, even when it's temporarily unavailable.

You must evaluate your Event Hub by testing that your Event Hub is processing data as expected. The metrics available in the Event Hubs allow you to ensure that it's working fine.

How do you use the Azure portal to view your Event Hub activity?

The Azure portal > Overview page for your Event Hub shows message counts. These message counts represent the data (events) received and sent by the Event Hub. You can choose the timescale for viewing these events.



How can you test Event Hub resilience?

Azure Event Hubs keeps receiving messages from the sender application even when it's unavailable. The messages received during this period are transmitted successfully as soon as the hub becomes available.

To test this functionality, you can use the Azure portal to disable your Event Hub.

When you re-enable your Event Hub, you can rerun your receiver application and use Event Hubs metrics for your namespace to check whether all sender messages have been successfully transmitted and received.

Other useful metrics available in the Event Hubs include:

- Throttled Requests: The number of requests that were throttled because the throughput unit usage was exceeded.
- ActiveConnections: The number of active connections on a namespace or Event Hub.
- Incoming/Outgoing Bytes: The number of bytes sent to/received from the Event Hubs service over a specified period.

Summary

Azure Event Hubs provides applications the capability to process large volume of streaming data and scale out during exceptionally high-demand periods as and when required. Azure Event Hubs decouple the sending and receiving messages to manage the data processing. This helps eliminate the risk of overwhelming consumer application and data loss due to any unplanned interruptions.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Applications that publish messages to Azure Event Hub very frequently will get the best performance using Advanced Message Queuing Protocol (AMQP) because it establishes a persistent socket. True or False?

- True
- False

Question 2

By default, how many partitions will a new Event Hub have?

- 1
- 2
- 3
- 4

Question 3

If an Event Hub goes offline before a consumer group can process the events it holds, those events will be lost. True or False?

- True
- False

Processing Data with Stream Analytics Jobs

Processing Data with Stream Analytics Jobs

Jobs are the foundation of performing real-time analytics in Azure Stream Analytics, and the good news is that it's straightforward to set them up. You'll set the job inputs to ingest the data, define how the data is to be processed and configure a job output to store or display the results.

Let's say that you're demonstrating to the Contoso team how to set up Azure Stream Analytics. You first provide a high level of the workflow before showing the team how to create a job. Once completed you'll show them how to create a job input followed by a job output. Finally, you'll provide example queries that can be used to perform the analytics before showing the team how to start the job and verify the result.

Learning Objectives

In this module you will:

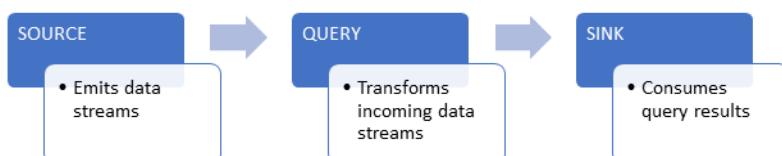
- Explore the Streaming Analytics workflow
- Create a Stream Analytics Job
- Configure a Stream Analytics job input
- Configure a Stream Analytics job output
- Write a transformation query
- Start a Stream Analytics job

What is a Stream Analytics Job

In order to examine the data streams that are passed onto Stream Analytics from an Event Hub or a Blob store a Stream Analytics Job must be defined. A job is a data pipeline that composes of three phases:

1. An **Input** provides the source of the data stream. This can either be a streaming data input, or a reference data input.
2. A **Transformation Query** A SQL-like query language for performing transformations and computations over streams of events such as the filtering and aggregation of the data. It takes the input data, applies the query, and then the result is passed to the output.
3. An **Output** which identifies the destination of the transformed data.

The Stream Analytics pipeline provides a transformed data flow going from input to output as shown in the following diagram.



Create and Stream Analytics Job

You can create a new Azure Stream Analytics job in the Azure portal.

1. Open the [Azure portal](#)² and sign into your account.
2. Select **All services** in the left pane.
3. In the search box, type in “**Stream Analytics**” and select **Stream Analytics jobs** from the results.
4. On the Stream Analytics jobs page, click **Add** to add a new job.

The screenshot shows the Microsoft Azure Stream Analytics jobs interface. The left sidebar includes options like 'Create a resource', 'Home', 'Dashboard', 'All services', 'Favorites' (with 'All resources' selected), 'Resource groups', 'App Services', 'Function Apps', 'SQL databases', 'Azure Cosmos DB', 'Virtual machines', 'Load balancers', 'Storage accounts', 'Virtual networks', and 'Azure Active Directory'. The main content area has a header 'Stream Analytics jobs' with a 'Default Directory' dropdown, 'Add' button, 'Edit columns' button, 'Refresh' button, and 'Assign tags' button. It displays 'Subscriptions: Free Trial' and filtering options for 'Filter by name...', 'All resource groups', and 'All locations'. Below this, it says '0 items' and lists columns for 'NAME' (sorted descending), 'STATUS', 'CREATED', and 'SUBSCR'. A large gear icon with waves is centered, and below it, a message says 'No stream analytics jobs to display. Try changing your filters if you don't see what you're looking for.' A blue 'Create stream analytics job' button is at the bottom.

5. Enter a Job name such as “SimpleTransformer”.
6. Create a new Resource group - name it “mslearn-streamanalytics”.
7. Take note of the Location setting. Ideally we'll create our job in the same location as any storage account(s) used for as a source or destination.
8. Ensure that the Hosting environment is **Cloud**.
9. Set the Streaming units to “1” to minimize the cost since we're just doing a simple test.
10. Select **Create** to create the new job.

² <https://portal.azure.com?azure-portal=true>

The screenshot shows the 'New Stream Analytics job' configuration page. The 'Job name' field is filled with 'SimpleTransformer'. The 'Subscription' dropdown is set to 'Visual Studio Enterprise'. The 'Resource group' dropdown shows '(New) mslearn-streamanalytics' with a 'Create new' link. The 'Location' dropdown is set to 'Central US'. Under 'Hosting environment', the 'Cloud' button is selected. A 'Streaming units (1 to 120)' slider is set to 1. At the bottom right are 'Create' and 'Automation options' buttons.

11. After a few moments, click **Refresh** to see your new Stream Analytics job.

| Stream Analytics jobs | | | |
|--|-------------------|---------|---------|
| Subscriptions: 4 of 11 selected – Don't see a subscription? Open Directory + Subscription settings | | | |
| <input type="checkbox"/> | NAME | STATUS | CREATED |
| <input type="checkbox"/> | SimpleTransformer | Created | 201 |

Now that we have a Stream Analytics job, we're ready to configure the job to serve a streaming workload. We'll start with the input.

Configuring a Job Input

An Azure Stream Analytics job supports three input types:

| Input Type | Use case |
|------------------------|--|
| Azure Event Hub | Azure Event Hub consumes live streaming data from applications with low latency and high throughput. |

| Input Type | Use case |
|---------------------------|--|
| Azure IoT Hub | Azure IoT Hub consumes live streaming events from IoT devices. This service enables bi-directional communication scenarios where commands can be sent back to IoT devices to trigger specific actions based on analyzing streams they send to the service. |
| Azure Blob Storage | Azure Blob Storage is used as the input source to consume files persisted in blob storage. |

Create the input source

Let's use an Azure Blob store as the input. Recall that Azure Blob Storage has three aspects to it: the Storage account to provide the globally unique namespace in Azure, the container which acts like a folder, and the blob itself which is similar to a file in a file system.

Let's start by creating an Azure Blob Storage account.

1. Switch back to the **Azure portal**³.
2. Select **All services** in the left sidebar.
3. Type “**storage**” in the search field and select **Storage accounts** from the results.
4. Click **Add** to create a new Azure Storage account.
5. On the **Basics** tab, select the new **mslearn-streamanalytics** resource group.
6. Set the Storage account name to a unique name - try using the prefix “streams” with your initials or a numeric value. This value has to be unique across all Azure storage accounts, so you might have to try a few combinations to find one that works for you. The portal will place a green checkmark next to the name if it's valid.
7. Check the Location - you can set it to the same location as the job to avoid having to pay to transfer data between regions.

TIP

This isn't absolutely necessary, but can be a cost-savings measure if you plan to have a lot of data flowing in or out of storage accounts.

8. Leave the rest of the fields as default values.
9. Select **Review + create**.
10. Once the request has been validated, select **Create** to submit the deployment request.

Wait for a few moments for the deployment to complete, once you receive the message “Your deployment is complete” go to the next step.

Connect the input source to the Stream Analytics job

Next, let's connect our Stream Analytics job to our new Blob Storage account.

1. In the Azure portal, Select **All services** in the left sidebar.
2. In the search box, type in **Stream Analytics**. Select the **Stream Analytics jobs** from the results.

³ <https://portal.azure.com?azure-portal=true>

3. In the list of jobs, select the Stream Analytics job you created earlier (**SimpleTransformer**). This will display the overview page for your job.

```

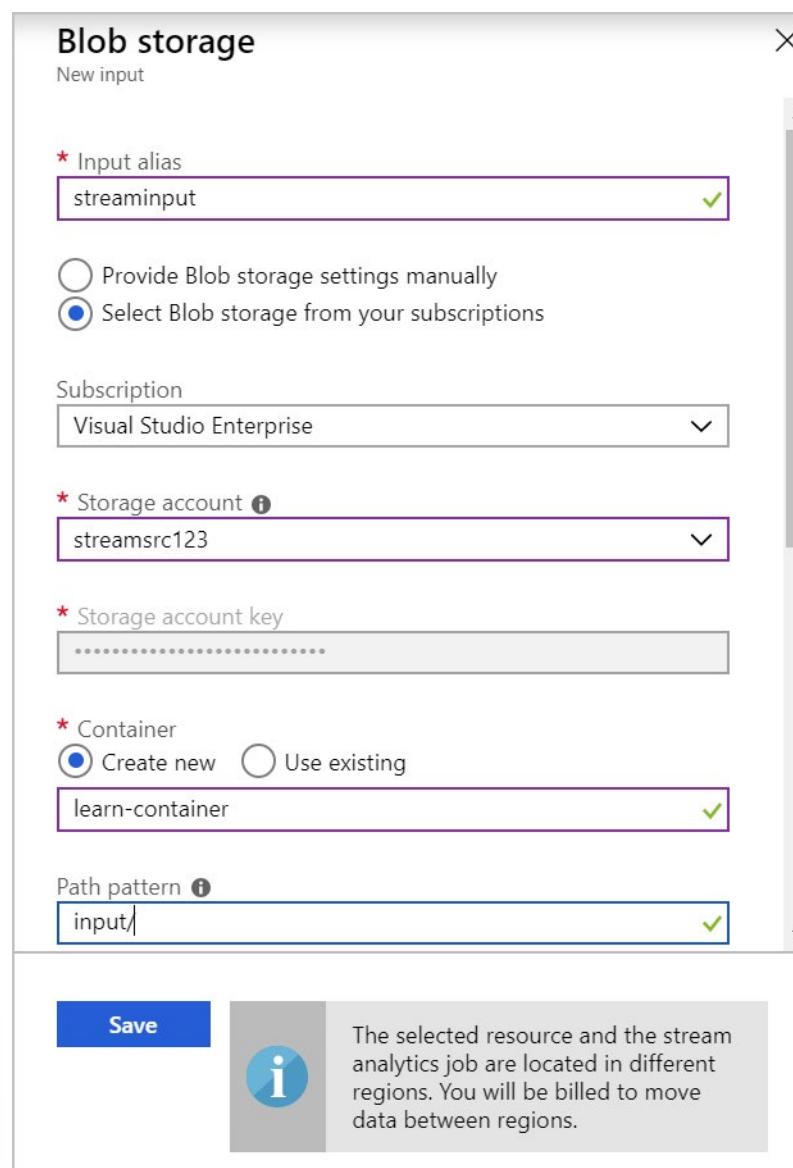
1 | SELECT *
2 | INTO [YourOutputAlias]
3 | FROM [YourInputAlias]

```

4. Under **Job topology**, select **Inputs**.
 5. Select **Add stream input** and select **Blob storage** from the drop-down list.
 6. Type **streaminput** in **Input alias** field - this is a friendly name you use to identify the input.
 7. Select the storage account you created earlier - recall it starts with **streamsdc**.
 8. Select **Create New** for Container field and give it a unique name such as **learn-container**.
 9. Enter **input/** for Path pattern.
 10. Leave the rest of the fields as the current default values.

NOTE

If the job and storage account are in different locations, you'll get a warning as shown in the following image.



11. Select **Save** to associate the input.

Configuring a Job Output

Stream Analytics jobs support various output sinks such as Azure Blob storage, Azure SQL Database, and Event Hub. There's a full list of output types in the documentation. In this exercise, we will use Blob storage as the output sink for our Stream Analytics job.

The steps will be very similar to what we just did to create the input. Let's start by creating a second Blob Storage account to hold the output.

1. In the **Azure portal**⁴, create a new storage account - just like you did in the prior exercise.
2. In the **Basics** tab, select your new **mslearn-streamanalytics** Resource group.

⁴ <https://portal.azure.com?azure-portal=true>

3. Use the prefix **streamsink** for the account name with a numeric suffix. Remember you might need to try a few combinations to find a unique name in Azure.
4. Use default values for all other fields.

The screenshot shows the 'Create storage account' wizard in the Azure portal. The 'Basics' tab is selected. In the 'PROJECT DETAILS' section, the subscription is set to 'Visual Studio Enterprise' and the resource group is 'mslearn-streamanalytics'. Under 'INSTANCE DETAILS', the storage account name is 'streamsink123' and the location is 'East US'. At the bottom, there are 'Review + create', 'Previous', and 'Next : Advanced >' buttons.

5. Select **Review + create**.
 6. Once the request has been validated, select **Create** to run the deployment step.
- Wait a few moments for the deployment to complete, once you receive the message "Your deployment is complete", move onto the next step.

Connect an output sink to a Stream Analytics job

Next, let's connect the storage account as the destination for our Azure Stream Analytics job.

1. Select **All services** in the left pane.
2. In the search box, type in **Stream Analytics** and select **Stream Analytics job** from the results.
3. Select the Stream Analytics job you created.
4. Under **Job topology**, select **Outputs**.
5. Click **Add** and select **Blob storage** from the drop-down list.
6. Type **streamoutput** in **Output alias** field - remember this is your own name for the output.
7. Select the storage account (**streamsink**) you created in the previous section.

NOTE

If your account doesn't appear in the dropdown, it likely just needs to be refreshed - just quit the Azure portal by closing the browser and then re-open the portal and try again.

1. Select **Create New** for Container field and give it a unique name such as **learn-container**.
2. Enter **output/** for Path pattern.
3. Leave the rest of the fields as default.
4. Select **Save**.

Using the Stream Analytics Query Language

An Azure Stream Analytics query transforms an input data stream and produces an output. Queries are written in a SQL-like language which is a subset of the Transact-SQL (T-SQL) language. We'll use a simple transformation of the input data to demonstrate the transformation query capabilities exposed by Stream Analytics.

NOTE

This content does not cover the full capability of the Stream Analytics Query Language, please refer to this [documentation](#)⁵ for full coverage of the capabilities of the language.

Let's assume we have some census data for specific cities and we need to pull specific elements out of the data - in this case, the coordinates of each city we got data for. We'll use a simple JSON file as our input, run a transformation query to pull coordinates out of the data, and then write the results out to a new file in our Blob storage.

Create the sample input file

Let's start by creating a simple input file. Create a file named **input.json** with the following contents on your local computer.

```
{
    "City" : "Reykjavík",
    "Coordinates" :
    {
        "Latitude": 64,
        "Longitude": 21
    },
    "Census" :
    {
        "Population" : 125000,
        "Municipality" : "Reykjavík",
        "Region" : "Höfuðborgarsvæðið"
    }
}
```

Upload the input file to blob store container

Next, let's upload that file to Azure Blob Storage.

1. Open the [Azure portal](#)⁶.
2. Navigate to your source Blob storage account (**streamsrc**).

TIP

You can use the Search field at the top of the Azure portal window to locate created resources by name, or use your Resource Group to find related resources.

1. Select the **streamsource** blob storage account you created earlier.
2. Select **Blobs** under **Blob service**.

⁵ <https://docs.microsoft.com/en-us/stream-analytics-query/stream-analytics-query-language-reference>

⁶ <https://portal.azure.com?azure-portal=true>

The screenshot shows the Azure Storage Blobs interface for the account 'streamsrc123'. On the left, there's a sidebar with options like Advanced Threat Protection, Static website, Properties, Locks, Automation script, Blob service, Blobs (which is selected), and Custom domain. The main area shows a table with one row for 'learn-container'. The table has columns for NAME, LAST MODIFIED, PUBLIC ACCESS L..., and LEASE STATE. The 'learn-container' row shows 'learn-container' in the NAME column, '1/29/2019, 3:42:37 PM' in the LAST MODIFIED column, 'Private' in the PUBLIC ACCESS L... column, and 'Available' in the LEASE STATE column.

3. Select the **learn-container** container you created. It should be empty.
4. Select **Upload**, click the folder icon next to the Files input and select the JSON file with the file dialog.
5. Expand the **Advanced** options if it's not expanded already.
6. Use **input/[YYYY-MM-DD]** for **Upload to folder** field where **YYYY-MM-DD** is the current date.
7. Leave defaults for all other options.
8. Select **Upload** to upload the file.

The screenshot shows the Azure Storage Blobs interface for the 'learn-container'. The left sidebar shows 'Overview', 'Access Control (IAM)', 'Settings' (with 'Access policy', 'Properties', 'Metadata', and 'Editor (preview)'), and 'Search blobs by prefix (case-sensitive)'. The main area shows a table with 'NAME' and 'MODIFIED' columns, both currently empty. A modal window titled 'Upload blob' is open, showing 'learn-container/' as the location. The 'Files' input field contains 'input.json'. There's a checkbox for 'Overwrite if files already exist'. The 'Advanced' section includes fields for 'Authentication type' (set to 'OAuth (preview)' with 'SAS' selected), 'Blob type' (set to 'Block blob'), a checked checkbox for 'Upload .vhdx files as page blobs (recommended)', 'Block size' (set to '4 MB'), and 'Upload to folder' (set to 'input/2019-01-31'). At the bottom of the modal is a blue 'Upload' button.

9. Once it's uploaded, you should see the "input" folder in the container. If you click on it, you can explore the blob hierarchy and see the data.

Configure the output blob store container

Next, let's configure the destination for the transformed data.

1. Navigate to your destination Blob storage account (**streamsink**).
2. Select **Storage Explorer (preview)** from the set of choices on the Overview page.

The screenshot shows the Azure Storage Explorer (preview) interface. On the left, there's a sidebar with icons for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, and Storage Explorer (preview). The Storage Explorer (preview) icon is highlighted with a red border. The main pane displays details for a storage account named 'streamsink123'. It includes a search bar at the top, followed by resource group information ('mslearn-streamanalytics'), status ('Primary: Available, Secondary: Available'), location ('East US, West US'), subscription details ('Visual Studio Enterprise'), subscription ID ('5270e7ce-d3ce-4172-b096-d0e9bb9e93ed'), and tags ('Click here to add tags').

3. In the right pane, navigate to **BLOB CONTAINERS**.
4. Select the container previously created.
5. Select **New Folder** from the menu above the container details. If it's not visible, click the **More...** drop-down to find it.
6. Type **output** for the folder name and select **Create**. This will create a placeholder, but Azure won't show it until a file is put into the folder.

Write the transformation query

Now we are ready to write our transformation query. We need to pull the coordinates out of the input data and write it to the output. We'll do that using a `SELECT` statement. The full query options are documented online and there's a link in the summary of this module.

1. Locate your Stream Analytics job in the Azure portal using the search field - recall its name is **Simple-Transformer**.
2. Under **Job topology**, select **Query**.
3. In the Query pane, add the following query.

```
SELECT City,
       Coordinates.Latitude,
       Coordinates.Longitude
  INTO streamoutput
  FROM streaminput
```


1. Select **Save** to save your changes.

Testing your query

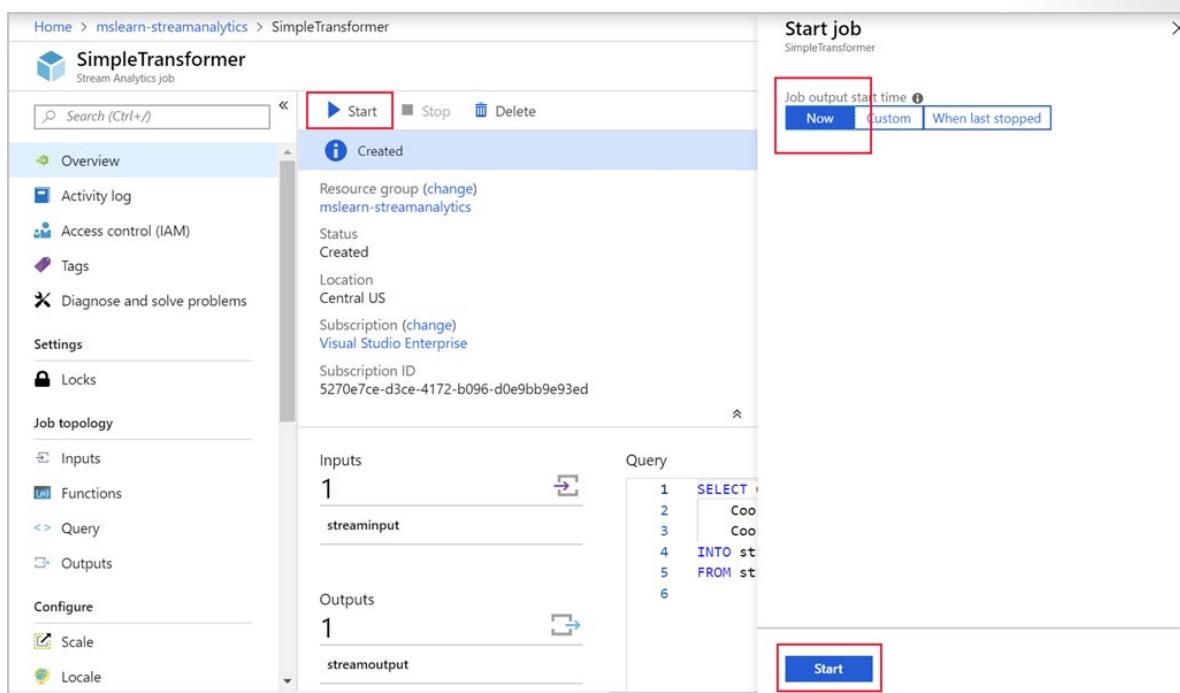
A useful step before you run a job is to test the query to ensure it does what you want. Navigate to your Stream Analytics job in the Azure portal, select **Query** under **Job topology**, and then select **Test**. You can then upload sample data to use with the query in order to perform the test.

Start a Stream Analytics Job

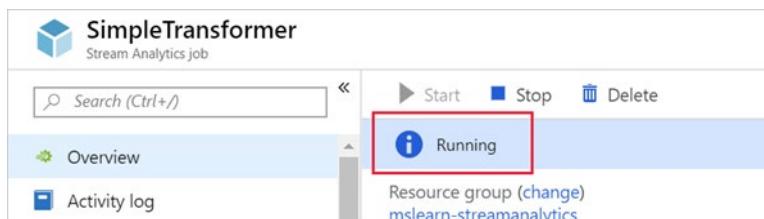
Now that you have created a job and configured it, it needs to be started to produce an output.

Use the following steps in the Azure portal.

1. In your job, select **Overview** in the menu to get back to the main overview page.
2. Select **Start** from the menu bar across the top of the overview page.
3. Select **Now**.
4. Select **Start**.



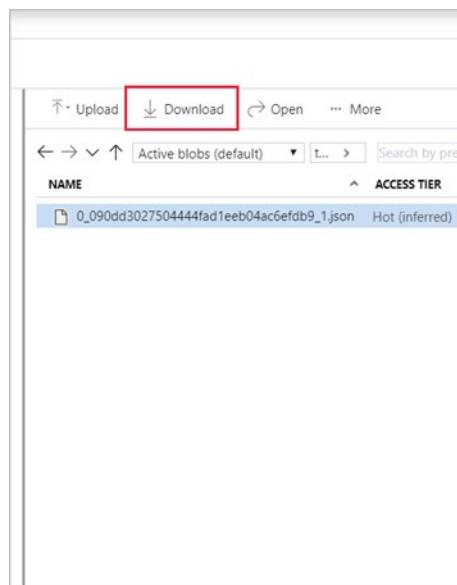
The job should transition to the "Starting" and then after a few seconds, the "Running" state. It will run for a few minutes and then finish in the "Completed" state. You should be able to watch this transition on the **Overview** page for your job.



View the Stream Analytics Job Results

Once a Stream Analytics job has completed, you can view the results right in the Azure portal. Within the Overview pane for the job, not only can you get information about the status, but you can confirm the location and resource group in which the service is provisioned, as well as the subscription details. You can also confirm dates when the service was created and started.

1. In the Azure portal, navigate to your output Storage account (**streamsink**).
2. Select Storage Explorer (preview).
3. In the right pane, open your container (**learn-container**) under **BLOB CONTAINERS**.
4. Navigate to **output** folder and download the blob to view the query output results.



If you open the file, you should see something like the following:

```
{  
    "city" : "Reykjavik",  
    "latitude" : 64,  
    "longitude" : 21  
}
```

Monitoring Stream Analytics Job

Monitoring is a key part of any mission or business-critical workload. It helps to proactively detect and prevent issues that might otherwise cause application or service downtime. There are several ways to monitor Azure Stream Analytics jobs:

1. Activity log specific to each running job
2. Real-time dashboards to show service and application health trends
3. Alerts on application or service issues
4. Diagnostic logs to investigate issues

Activity log

The Azure Stream Analytics activity log provides details about each job you run. It's a low-level troubleshooting capability that can help identify issues with data sources, outputs, or transformation queries. There's an Activity Log blade available with each job you create. You can expand to see each job and then select each event to see the details in JSON as shown here.

The screenshot shows the 'Activity log' blade for a Stream Analytics job named 'tutorial'. The left sidebar includes options like 'Create a resource', 'Home', 'Dashboard', and 'All services'. The main area has a search bar and filters for 'Subscription', 'Timespan', 'Event severity', 'Resource group', and 'Resource'. A table lists 10 items under 'OPERATION NAME', showing various 'Start Stream Analytics Job' events with different statuses (Succeeded, Started, Accepted) and times. The table also includes columns for 'STATUS', 'TIME', 'TIME STAMP', and 'SUBSCRIPTION'.

| OPERATION NAME | STATUS | TIME | TIME STAMP | SUBSCRIPTION |
|----------------------------|-----------|-----------|-----------------|--------------|
| Start Stream Analytics Job | Succeeded | 6 min ago | Tue Jan 15 2... | Free Trial |
| Start Stream Analytics Job | Started | 7 min ago | Tue Jan 15 2... | Free Trial |
| Start Stream Analytics Job | Started | 7 min ago | Tue Jan 15 2... | Free Trial |
| Start Stream Analytics Job | Accepted | 7 min ago | Tue Jan 15 2... | Free Trial |
| Start Stream Analytics Job | Accepted | 7 min ago | Tue Jan 15 2... | Free Trial |
| Start Stream Analytics Job | Succeeded | 6 min ago | Tue Jan 15 2... | Free Trial |
| Start Stream Analytics Job | Accepted | 7 min ago | Tue Jan 15 2... | Free Trial |
| Start Stream Analytics Job | Accepted | 7 min ago | Tue Jan 15 2... | Free Trial |
| Start Stream Analytics Job | Accepted | 7 min ago | Tue Jan 15 2... | Free Trial |

Dashboards

To view live dashboards for key health metrics of your Stream Analytics jobs, go to the Azure portal, select your Stream Analytics job, and select Metrics under Monitoring.

The screenshot shows the 'Metrics' blade for a Stream Analytics job named 'tutorial'. The left sidebar includes options like 'Create a resource', 'Home', 'Dashboard', and 'All services'. The main area has a search bar and a 'Configure' section with tabs for 'Scale', 'Locale', 'Event ordering', 'Error policy', 'Compatibility level', 'General', 'Tools', 'Properties', and 'Monitoring'. Under 'Monitoring', there is a 'Metrics' tab. On the right, there is a 'New chart' interface with options to 'Add metric', 'Add filter', 'Apply splitting', and 'Line chart'. A dropdown menu for 'RESOURCE' shows 'tutorial'. A dropdown for 'METRIC NAMESPACES' shows 'Stream Analytics job'. A dropdown for 'METRIC' shows 'Select metric'. A dropdown for 'AGGREGATION' shows 'Select value(s)'. A list of metrics is displayed, including 'Backlogged Input Events', 'Data Conversion Errors', 'Early Input Events', 'Failed Function Requests', 'Function Events', 'Function Requests', 'Input Deserialization Errors', and 'Input Event Bytes'.

Alerts

To enable proactive detection of issues, Azure Stream Analytics can be configured to fire alerts based on various metrics and thresholds. To configure alerts in the Azure portal, navigate to your Stream Analytics job, and select **Alert rules** under **Monitoring**. Then select **+ New alert rule**.

Alerts can be configured to be sent as emails, SMS, or voicemail. Also, alerts can also be used to trigger other workflows.

Diagnostics

Diagnostic logging is a key part of the operational infrastructure to help root-cause issues in production deployments. Stream Analytics diagnostics, off by default, can be turned on in Azure portal as needed by selecting the Stream Analytics job and selecting **Diagnostic logs** under **Monitoring**. Diagnostic logs can be conveniently delivered to various sinks or destinations for root-cause analysis.

Diagnostics settings can be persisted in an Azure Storage account, sent to Azure Event Hub, or sent to Azure Log Analytics. Diagnostics logs can be generated for job execution or job authoring.

The screenshot shows the 'Diagnostics settings' page for a Stream Analytics job named 'tutorial - Diagnostics logs'. The left sidebar lists various Azure services, and the top navigation bar shows the full URL: Home > Stream Analytics jobs > tutorial - Diagnostics logs > Diagnostics settings. The main area contains fields for 'Name' (with a placeholder 'tutorial'), and checkboxes for 'Archive to a storage account', 'Stream to an event hub', and 'Send to Log Analytics'. Below these are sections for 'LOG' (Execution, Authoring) and 'METRIC' (AllMetrics), each with its own checkbox.

Summary

In this unit, we learned how to create an Azure Stream Analytics job, configure an input, write a transformation query, and configure output. We used Azure Storage as our source and destination and created a transformation query to produce some basic results. In addition, we learned how to start a Stream Analytics job and view the job results.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which job input consumes data streams from applications at low latencies and high throughput?

- Azure Blob
- Event Hubs
- IoT hubs

Question 2

Streaming Analytics Query Language is a subset of which query language

- WQL
- TSQL
- JSON

Question 3

You are a Data Engineer for Contoso. You want to view key health metrics of your Stream Analytics jobs. Which tool in Streaming Analytics should you use?

- Diagnostics
- Alerts
- Dashboards

Module Summary

Module Summary

In this module, you have learned the concepts of event processing and streaming data and how this applies to Events Hubs and Azure Stream Analytics. You will then set up a stream analytics job to stream data and learn how to query the incoming data to perform analysis of the data. Finally, you will learn how to manage and monitor running jobs.

Learning Objectives

In this module, you have learned:

- Data streams and event processing.
- Data Ingestion with Event Hubs.
- Processing Data with Stream Analytics Jobs.

Post Course Review

After the course, consider reading the **Reference architecture for real-time event processing with Microsoft Azure Stream Analytics**⁷. The reference architecture for real-time event processing with Azure Stream Analytics is intended to provide a generic blueprint for deploying a real-time platform as a service (PaaS) stream-processing solution with Microsoft Azure.

Important

Remember

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

⁷ <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-real-time-event-processing-reference-architecture>

Answers

Question 1

Which of the following technologies typically provide an ingestion point for data streaming in an event processing solution that uses static data as a source?

- IoT Hubs
- Event Hubs
- Azure Blob storage

Explanation

Azure Blob storage provide an ingestion point for data streaming in an event processing solution that uses static data as a source. IoT hubs provide an ingestion point for data streaming in an event processing solution that uses streaming data from an IoT device as a source. Event hubs provide an ingestion point for data streaming in an event processing solution that uses streaming data from an application as a source.

Question 1

Applications that publish messages to Azure Event Hub very frequently will get the best performance using Advanced Message Queuing Protocol (AMQP) because it establishes a persistent socket. True or False?

- True
- False

Explanation

True. Publishers can use either HTTPS or AMQP. AMQP opens a socket and can send multiple messages over that socket.

Question 2

By default, how many partitions will a new Event Hub have?

- 1
- 2
- 3
- 4

Explanation

Event Hubs default to 4 partitions. Partitions are the buckets within an Event Hub. Each publication will go into only one partition. Each consumer group may read from one or more than one partition.

Question 3

If an Event Hub goes offline before a consumer group can process the events it holds, those events will be lost. True or False?

- True
- False

Explanation

False. Events are persistent. Each consumer group has its own cursor maintaining its position within the partition. The consumer groups can resume processing at their cursor position when the Event Hub is again available.

Question 1

Which job input consumes data streams from applications at low latencies and high throughput?

- Azure Blob
- Event Hubs
- IoT hubs

Explanation

Event hubs consumes data streams from applications at low latencies and high throughput. Azure Blob stores static data that can be rerun through a streaming job. Azure IoT Hub is a data stream ingestion service that consumes events from IoT devices and also provides bi-directional communication between Azure and IoT devices.

Question 2

Streaming Analytics Query Language is a subset of which query language

- WQL
- TSQL
- JSON

Explanation

Streaming Analytics Query Language is a subset of Transact-SQL. WQL is Windows Management Instrumentation query language. JSON is an open standard file format

Question 3

You are a Data Engineer for Contoso. You want to view key health metrics of your Stream Analytics jobs. Which tool in Streaming Analytics should you use?

- Diagnostics
- Alerts
- Dashboards

Explanation

Dashboard are used to view the key health metrics of your Stream Analytics jobs. Diagnostic logging is turned off by default and can help with root-cause analysis in production deployments. Alerts enable proactive detection of issues in Stream Analytics.

Module 7 Orchestrating Data Movement with Azure Data Factory

Module Introduction

Orchestrating Data Movement with Azure Data Factory

In this module, students will learn how Azure Data Factory can be used to orchestrate the movement and transformation of data both natively and from a wide range of data platform technologies. You will be able to explain the capabilities of the technology and set up an end to end data pipeline that ingests and transforms data with an Azure data platform technology.

Learning Objectives

In this module, you will:

- Introduced to Azure Data Factory
- Understand Azure Data Factory Components
- Ingesting and Transforming Data with Azure Data Factory
- Integrate Azure Data Factory with Databricks

Introducing Azure Data Factory

Introducing Azure Data Factory

With the wide range of data stores available in Azure, there is the need to manage and orchestrate the movement data between them. In fact, you may want to automate a regular process of data movement as part of a wider enterprise analytical solution. Azure Data Factory meets that need, and in this section, you will be introduced to this technology, its component parts, the Azure Data Factory process and the security required to provision and manage the service.

Learning Objectives

In this section, you will learn:

- What is Azure Data Factory
- The Data Factory Process
- Azure Data Factory components
- Azure Data Factory Security

Introduction to Azure Data Factory

The need to trigger the batch movement of data, or to set up a regular schedule is a requirement for most analytics solutions. Azure Data Factory (ADF) is the service that can be used to fulfil such a requirement. ADF provides a cloud-based data integration service that orchestrates the movement and transformation of data between various data stores and compute resources.

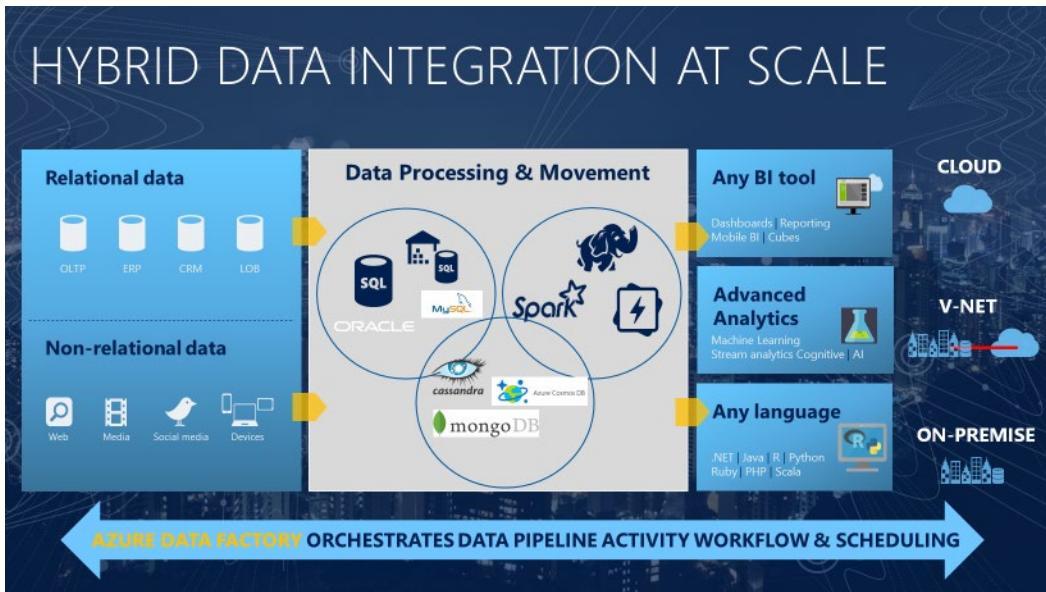
Azure Data Factory is the cloud-based ETL and data integration service that allows you to create data-driven workflows for orchestrating data movement and transforming data at scale. Using Azure Data Factory, you can create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores. You can build complex ETL processes that transform data visually with data flows or by using compute services such as Azure HDInsight Hadoop, Azure Databricks, and Azure Synapse Analytics.

What is meant by orchestration

To use an analogy, think about a symphony orchestra. The central member of the orchestra is the conductor. The conductor does not play the instruments, they simply lead the symphony members through the entire piece of music that they perform. The musicians use their own skills to produce particular sounds at various stages of the symphony, so they may only learn certain parts of the music. The conductor orchestrates the entire piece of music, and therefore is aware of the entire score that is being performed. They will also use specific arm movements that provide instructions to the musicians how a piece of music should be played.

ADF can use a similar approach, whilst it has native functionality to ingest and transform data, sometimes it will instruct another service to perform the actual work required on its behalf, such as a Databricks to execute a transformation query. So, in this case, it would be Databricks that performs the work, not ADF. ADF merely orchestrates the execution of the query, and then provides the pipelines to move the data onto the next step or destination.

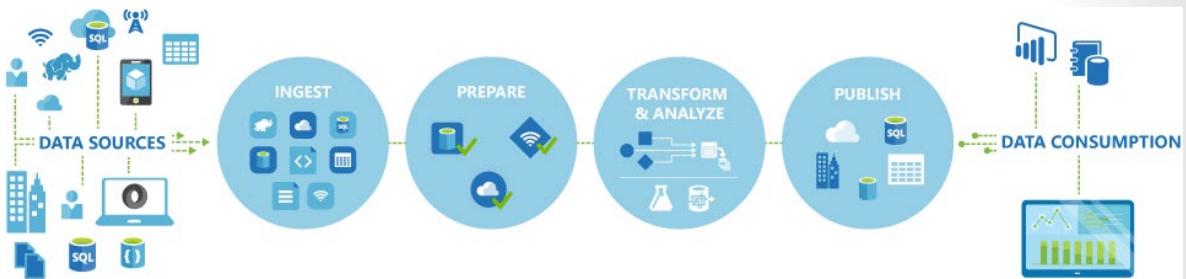
It also provides rich visualizations to display the lineage and dependencies between your data pipelines, and monitor all your data pipelines from a single unified view to easily pinpoint issues and setup monitoring alerts.



The Azure Data Factory Process

Data-driven workflows

The pipelines (data-driven workflows) in Azure Data Factory typically perform the following four steps:



Connect and collect

The first step in building an orchestration system is to define and connect all the required sources of data together, such as databases, file shares, and FTP web services. The next step is to move the data as needed to a centralized location for subsequent processing.

Transform and enrich

Compute services such as Databricks and Machine Learning can be used to produce transformed data on a maintainable and controlled schedule to feed production environments with cleansed and transformed data. In some instances, you may even augment the source data with additional data to aid analysis, or

consolidate it through a normalization process to be used in a Machine Learning experiment as an example.

Publish

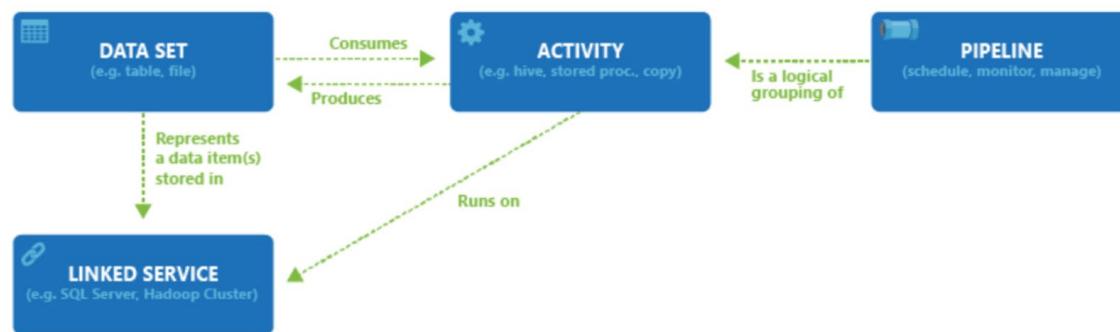
After the raw data has been refined into a business-ready consumable form from the transform and enrich phase, you can load the data into Azure Data Warehouse, Azure SQL Database, Azure Cosmos DB, or whichever analytics engine your business users can point to from their business intelligence tools

Monitor

Azure Data Factory has built-in support for pipeline monitoring via Azure Monitor, API, PowerShell, Azure Monitor logs, and health panels on the Azure portal, to monitor the scheduled activities and pipelines for success and failure rates.

Data Factory Components

An Azure subscription might have one or more Azure Data Factory instances. Azure Data Factory is composed of four core components. These components work together to provide the platform on which you can compose data-driven workflows with steps to move and transform data.



Data Factory supports a wide variety of data sources that you can connect to through the creation of an object known as a **Linked Service**. This enables you to ingest the data from a data source in readiness to prepare the data for transformation and/or analysis. In addition, Linked Services can fire up compute services on demand. For example, you may have a requirement to start an on-demand HDInsight cluster for the purpose of just processing data through a Hive query. So Linked Services enables you to define data sources, or compute resource that are required to ingest and prepare data.

With the linked service defined, Azure Data Factory is made aware of the datasets that it should use through the creation of a **Datasets** object. Datasets represent data structures within the data store that is being referenced by the Linked Service object. Datasets can also be used by an ADF object known as an Activity.

Activities typically contain the transformation logic or the analysis commands of the Azure Data Factory's work. This could include the execution of a stored procedure, Hive Query or Pig script to transform the data. You can push data into a Machine Learning model to perform analysis. It is not uncommon for multiple activities to take place that may include transforming data using a SQL stored procedure and then perform analytics with Databricks. In this case, multiple activities can be logically grouped together with an object referred to as a **Pipeline**, and these can be *scheduled* to execute, or a *trigger* can be

defined that determines when a pipeline execution needs to be kicked off. There are different types of triggers for different types of events.

Control flow is an orchestration of pipeline activities that includes chaining activities in a sequence, branching, defining parameters at the pipeline level, and passing arguments while invoking the pipeline on-demand or from a trigger. It also includes custom-state passing and looping containers, that is, For-each iterators.

Parameters are key-value pairs of read-only configuration. Parameters are defined in the pipeline. The arguments for the defined parameters are passed during execution from the run context that was created by a trigger or a pipeline that was executed manually. Activities within the pipeline consume the parameter values.

Azure Data Factory has an *integration runtime* that enables it to bridge between the activity and linked Services objects. It is referenced by the linked service, and provides the compute environment where the activity either runs on or gets dispatched from. This way, the activity can be performed in the region closest possible. There are three types of Integration Runtime, including Azure, Self-hosted and Azure-SSIS.

Once all the work is complete you can then use Data Factory to publish the final dataset to another linked service that can then be consumed by technologies such as Power BI or Machine Learning.

Azure Data Factory Security

To create Data Factory instances, the user account that you use to sign in to Azure must be a member of the *contributor* or *owner* role, or an *administrator* of the Azure subscription.

To create and manage Data Factory objects including datasets, linked services, pipelines, triggers, and integration runtimes, the following requirements must be met:

- To create and manage child resources in the Azure portal, you must belong to the *Data Factory Contributor* role at the resource group level or above.
- To create and manage resources with PowerShell or the SDK, the *contributor* role at the resource level or above is sufficient.

Data Factory Contributor role

When you are added as a member of this role, you have the following permissions:

- Create, edit, and delete data factories and child resources including datasets, linked services, pipelines, triggers, and integration runtimes.
- Deploy Resource Manager templates. Resource Manager deployment is the deployment method used by Data Factory in the Azure portal.
- Manage App Insights alerts for a data factory.
- At the resource group level or above, lets users deploy Resource Manager template.
- Create support tickets.

If the Data Factory Contributor role does not meet your requirement, you can create your own **custom role**¹.

¹ <https://docs.microsoft.com/en-us/azure/role-based-access-control/custom-roles>

Summary

In this section, you have learned how Azure Data Factory can be used to manage a regular process of data movement as part of a wider enterprise analytical solution. In this introductory section you have learned how Azure Data Factory meets that need through its component parts and the Azure Data Factory process. You also now understand the security groups required to provision and manage the service.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which Azure Data Factory process involves using compute services to produce data to feed production environments with cleansed data?

- Connect and collect
- Transform and enrich
- Publish
- Monitor

Question 2

Which Azure Data Factory component contains the transformation logic or the analysis commands of the Azure Data Factory's work?

- Linked Services
- Datasets
- Activities
- Pipelines

Azure Data Factory Components

Azure Data Factory Components

In this section, we will explore in more depth how the various core Azure Data Factory components can be developed and configured. This can be done either using the graphical user interface, or programmatically using JSON notation and you will explore examples of how they can be configured.

Learning Objectives

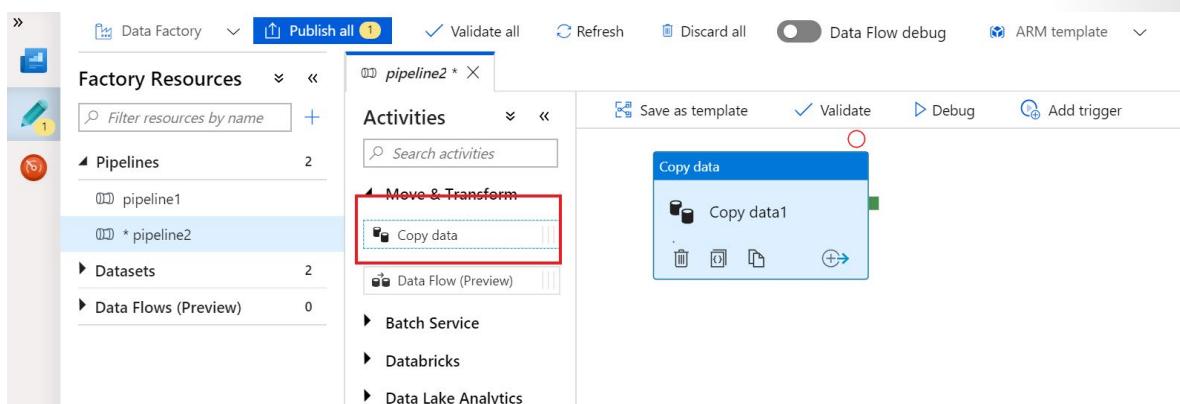
In this module, you will learn:

- Linked Services
- Datasets
- Data Factory Activities
- Pipelines
- Integration runtimes

Creating Linked Services

Before you create a dataset, you must create a **linked service** to link your data store to the data factory. Linked services are much like connection strings, which define the connection information needed for Data Factory to connect to external resources. There are over 80 connectors that can be used to define a linked service.

A linked service in Data Factory is defined using the Copy Data Activity in the ADF designer. The Copy Activity copies data between the source and destination, and when you run this activity you are asked to define a linked service as part of the copy activity definition



Alternatively you can programmatically define a linked service in the JSON format to be used via REST APIs or the SDK, using the following notation:

```
{  
    "name": "<Name of the linked service>",  
    "properties": {  
        "type": "<Type of the linked service>",  
        "typeProperties": {  
            "<data store or compute-specific type properties>"  
        }  
    }  
}
```

```
        }  
    },  
    "connectVia": {  
        "referenceName": "<name of Integration Runtime>",  
        "type": "IntegrationRuntimeReference"  
    }  
}  
}  
}
```

The following table describes properties in the above JSON:

| Property | Description | Required |
|----------------|--|----------|
| name | Name of the linked service. | Yes |
| type | Type of the linked service. For example: AzureStorage (data store) or AzureBatch (compute). See the description for typeProperties. | Yes |
| typeProperties | The type properties are different for each data store or compute. For the supported data store types and their type properties, see the dataset type table (https://docs.microsoft.com/en-us/azure/data-factory/concepts-datasets-linked-services#dataset-type). Navigate to the data store connector article to learn about type properties specific to a data store. | Yes |
| connectVia | The Integration Runtime (https://docs.microsoft.com/en-us/azure/data-factory/concepts-integration-runtime) to be used to connect to the data store. You can use Azure Integration Runtime or Self-hosted Integration Runtime (if your data store is located in a private network). If not specified, it uses the default Azure Integration Runtime. | No |

Example of a Linked Service

Azure SQL Database

The following example creates a linked service named "AzureSqlLinkedService" that connects to an Azure SQL Database named "ctosqlldb" with the userid of "ctesta-oneill" and the password of "P@ssw0rd".

```
{  
    "name": "AzureSqlLinkedService",  
    "properties": {  
        "type": "AzureSqlDatabase",  
        "typeProperties": {  
            "connectionString": "Server=tcp:<server-name>.database.windows.  
net,1433;Database=EquityDB;User ID=<user-name>;Password=P@ssw0rd;Trusted_  
Connection=False;Encrypt=True;Connection Timeout=30"  
        }  
    }  
}
```

Azure Blob Storage

The following example creates a linked service named "StorageLinkedService" that connects to an Azure Blob Store named "ctostorageaccount" with the storage account key used to connect to the data store

```
{  
    "name": "StorageLinkedService",  
    "properties": {  
        "type": "AzureStorage",  
        "typeProperties": {  
            "connectionString": "DefaultEndpointsProtocol=https;AccountName=ctos-  
storageaccount;AccountKey=<account-key>"  
        }  
    }  
}
```

Creating Datasets

A dataset is a named view of data that simply points or references the data you want to use in your activities as inputs and outputs. Datasets identify data within different data stores, such as tables, files, folders, and documents. For example, an Azure Blob dataset specifies the blob container and folder in Blob storage from which the activity should read the data.

A dataset in Data Factory can be defined as an object within the Copy Data Activity or in JSON format as follows:

```
{  
    "name": "<name of dataset>",  
    "properties": {  
        "type": "<type of dataset: AzureBlob, AzureSql etc...>",  
        "linkedServiceName": {  
            "referenceName": "<name of linked service>",  
            "type": "LinkedServiceReference",  
        },  
        "structure": [  
            {  
                "name": "<Name of the column>",  
                "type": "<Name of the type>"  
            }  
        ]  
    }  
}
```

```
        ],
    "typeProperties": {
        "<type specific property>": "<value>",
        "<type specific property 2>": "<value 2>",
    }
}
}
```

The following table describes properties in the above JSON:

| Property | Description | Required |
|----------------|--|----------|
| name | Name of the dataset. | Yes |
| type | Type of the dataset. Specify one of the types supported by Data Factory (for example: AzureBlob, AzureSqlTable). | Yes |
| structure | Schema of the dataset. | No |
| typeProperties | The type properties are different for each type (for example: Azure Blob, Azure SQL table). | Yes |

Example of a Dataset

Azure SQL Database

The following example creates a dataset named "AzureBlobOutput" that defines a dataset in Azure Blob Store, in a container named datacontainer and a folder named partitioneddata which will contain a text document.

```
{
    "name": "AzureBlobOutput",
    "properties": {
        "published": false,
        "type": "AzureBlob",
        "linkedServiceName": "AzureStorageLinkedService",
        "typeProperties": {
            "folderPath": "datacontainer/partitioneddata",
            "format": {
                "type": "TextFormat",
                "columnDelimiter": ","
            }
        }
    }
}
```

Data Factory Activities

Activities within Azure Data Factory defines the actions that will be performed on the data and there are three categories including:

- Data movement activities
- Data transformation activities
- Control activities

Data movement activities

Data movement activities simply move data from one data store to another. You can use the Copy Activity to perform data movement activities, or by using JSON. There are a wide range of data stores that are supported as a source and as a sink. This list is ever increasing, and you can find the [latest information here²](#).

Data transformation activities

Data transformation activities can be performed natively within the authoring tool of Azure Data Factory using the Mapping Data Flow. Alternatively, you can call a compute resource to change or enhance data through transformation, or perform analysis of the data. These include compute technologies such as Azure Databricks, Azure Batch, SQL Database and Azure Synapse Analytics, Machine Learning Services, Azure Virtual machines and HDInsight. You can make use of any existing SQL Server Integration Services (SSIS) Packages stored in a Catalog to execute in Azure.

As this list is always evolving, you can get the [latest information here³](#).

Control activities

When graphically authoring ADF solutions, you can use the control flow within the designed to orchestrate pipeline activities that includes chaining activities in a sequence, branching, defining parameters at the pipeline level, and passing arguments while invoking the pipeline on-demand or from a trigger. The current capabilities include:

| Control Activity | Description |
|---------------------------|--|
| Execute Pipeline Activity | Execute Pipeline activity allows a Data Factory pipeline to invoke another pipeline. |
| ForEachActivity | ForEach Activity defines a repeating control flow in your pipeline. This activity is used to iterate over a collection and executes specified activities in a loop. The loop implementation of this activity is similar to Foreach looping structure in programming languages. |
| WebActivity | Web Activity can be used to call a custom REST endpoint from a Data Factory pipeline. You can pass datasets and linked services to be consumed and accessed by the activity. |

² <https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities#data-movement-activities>

³ <https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities#data-transformation-activities>

| Control Activity | Description |
|-----------------------|---|
| Lookup Activity | Lookup Activity can be used to read or look up a record/ table name/ value from any external source. This output can further be referenced by succeeding activities. |
| Get Metadata Activity | GetMetadata activity can be used to retrieve metadata of any data in Azure Data Factory. |
| Until Activity | Implements Do-Until loop that is similar to Do-Until looping structure in programming languages. It executes a set of activities in a loop until the condition associated with the activity evaluates to true. You can specify a timeout value for the until activity in Data Factory. |
| If Condition Activity | The If Condition can be used to branch based on condition that evaluates to true or false. The If Condition activity provides the same functionality that an if statement provides in programming languages. It evaluates a set of activities when the condition evaluates to true and another set of activities when the condition evaluates to false. |
| Wait Activity | When you use a Wait activity in a pipeline, the pipeline waits for the specified period of time before continuing with execution of subsequent activities. |

You can get the [latest information here⁴](#).

Activities and Pipelines

Defining Activities

When using JSON notation, the activities section can have one or more activities defined within it. There are two main types of activities: Execution and Control Activities. Execution (AKA Compute) activities include data movement and data transformation activities. They have the following top-level structure:

```
{  
    "name": "Execution Activity Name",  
    "description": "description",  
    "type": "<ActivityType>",  
    "typeProperties":  
    {  
    },  
    "linkedServiceName": "MyLinkedService",  
    "policy":  
    {  
    },  
    "dependsOn":  
    {  
    }  
}
```

⁴ <https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities#control-activities>

```
}
```

The following table describes properties in the above JSON:

| Property | Description | Required |
|-------------------|---|---|
| name | Name of the activity. | Yes |
| description | Text describing what the activity or is used for. | Yes |
| type | Defines the type of the activity. | Yes |
| linkedServiceName | Name of the linked service used by the activity. | Yes for HDInsight , Machine Learning Batch Scoring Activity and Stored Procedure Activity |
| typeProperties | Properties in the typeProperties section depend on each type of activity. | No |
| policy | Policies that affect the run-time behavior of the activity. This property includes timeout and retry behavior. | No |
| dependsOn | This property is used to define activity dependencies, and how subsequent activities depend on previous activities. | No |

Defining Control Activities

A Control Activity in Data Factory is defined in JSON format as follows:

```
{
  "name": "Control Activity Name",
  "description": "description",
  "type": "<ActivityType>",
  "typeProperties": {
    {
    },
    "dependsOn": {
      {
      }
    }
}
```

The following table describes properties in the above JSON:

| Property | Description | Required |
|-------------|---|----------|
| name | Name of the activity. | Yes |
| description | Text describing what the activity or is used for. | Yes |
| type | Defines the type of the activity. | Yes |

| Property | Description | Required |
|----------------|---|----------|
| typeProperties | Properties in the typeProperties section depend on each type of activity. | No |
| dependsOn | This property is used to define activity dependencies, and how subsequent activities depend on previous activities. | No |

Defining Pipelines

Here is how a pipeline is defined in JSON format:

```
{
    "name": "PipelineName",
    "properties":
    {
        "description": "pipeline description",
        "activities":
        [
        ],
        "parameters": {}
    }
}
```

The following table describes properties in the above JSON:

| Property | Description | Required |
|-------------|--|----------|
| name | Name of the pipeline. | Yes |
| description | Text describing what the pipeline is used for. | No |
| activities | The activities section can have one or more activities defined within it. | Yes |
| parameters | The parameters section can have one or more parameters defined within the pipeline, making your pipeline flexible for reuse. | No |

Example

The following JSON defines pipeline named "MyFirstPipeline" that contains one activity type of HDInsightHive that will call a query from a script name "partitionweblogs.hql" that is stored in the linked service named "StorageLinkedService", with an input named "AzureBlobInput" and an output named "AzureBlobOutput". It execute this against the compute resource defined in the linked service named "HDInsightOnDemandLinkedService"

The pipeline is scheduled to execute on a monthly basis, and will attempt to execute 3 times should it fail.

```
{  
    "name": "MyFirstPipeline",  
    "properties": {  
        "description": "My first Azure Data Factory pipeline",  
        "activities": [  
            {  
                "type": "HDInsightHive",  
                "typeProperties": {  
                    "scriptPath": "adfgetstarted/script/partitionweblogs.  
hql",  
                    "scriptLinkedService": "StorageLinkedService",  
                    "defines": {  
                        "inputtable": "wasb://adfgetstarted@ctostorageac-  
count.blob.core.windows.net/inputdata",  
                        "partitionedtable": "wasb://adfgetstarted@ctostor-  
ageaccount.blob.core.windows.net/partitioneddata"  
                    }  
                },  
                "inputs": [  
                    {  
                        "name": "AzureBlobInput"  
                    }  
                ],  
                "outputs": [  
                    {  
                        "name": "AzureBlobOutput"  
                    }  
                ],  
                "policy": {  
                    "concurrency": 1,  
                    "retry": 3  
                },  
                "scheduler": {  
                    "frequency": "Month",  
                    "interval": 1  
                },  
                "name": "RunSampleHiveActivity",  
                "linkedServiceName": "HDInsightOnDemandLinkedService"  
            }  
        ],  
        "start": "2017-04-01T00:00:00Z",  
        "end": "2017-04-02T00:00:00Z",  
        "isPaused": false,  
        "hubName": "ctogetstarteddf_hub",  
        "pipelineMode": "Scheduled"  
    }  
}
```

Integration runtime

In Data Factory, an activity defines the action to be performed. A linked service defines a target data store or a compute service. An integration runtime provides the bridge between the activity and linked services.

It is referenced by the linked service or activity, and provides the compute environment where the activity either runs on or gets dispatched from. This way, the activity can be performed in the region closest possible to the target data store or compute service in the most performant way while meeting security and compliance needs.

In short, the Integration Runtime (IR) is the compute infrastructure used by Azure Data Factory. It provides the following data integration capabilities across different network environments, including:

- **Data Flow:** Execute a Data Flow in managed Azure compute environment.
- **Data movement:** Copy data across data stores in public network and data stores in private network (on-premises or virtual private network). It provides support for built-in connectors, format conversion, column mapping, and performant and scalable data transfer.
- **Activity dispatch:** Dispatch and monitor transformation activities running on a variety of compute services such as Azure Databricks, Azure HDInsight, Azure Machine Learning, Azure SQL Database, SQL Server, and more.
- **SSIS package execution:** Natively execute SQL Server Integration Services (SSIS) packages in a managed Azure compute environment.

Integration runtime types

Data Factory offers three types of Integration Runtime, and you should choose the type that best serve the data integration capabilities and network environment needs you are looking for. These three types are:

- Azure
- Self-hosted
- Azure-SSIS

The following table describes the capabilities and network support for each of the integration runtime types:

| IR type | Public network | Private network |
|-------------|--|---------------------------------|
| Azure | Data Flow Data movement Activity dispatch | |
| Self-hosted | Data movement Activity dispatch | Data movement Activity dispatch |
| Azure-SSIS | SSIS package execution | SSIS package execution |

Azure integration runtime

An Azure integration runtime is capable of:

- Running Data Flows in **Azure**
- Running copy activity **between cloud data stores**
- Dispatching the following transform activities in **public network**: Databricks Notebook/ Jar/ Python activity, HDInsight Hive activity, HDInsight Pig activity, HDInsight MapReduce activity, HDInsight Spark

activity, HDInsight Streaming activity, Machine Learning Batch Execution activity, Machine Learning Update Resource activities, Stored Procedure activity, Data Lake Analytics U-SQL activity, .NET custom activity, Web activity, Lookup activity, and Get Metadata activity.

You can set a certain location of an Azure IR, in which case the data movement or activity dispatch will happen in that specific region. If you choose to use the auto-resolve Azure IR which is the default, ADF will make a best effort to automatically detect your sink and source data store to choose the best location either in the same region if available or the closest one in the same geography for the copy activity. For anything else, it will use the IR in the data factory region.

Self-hosted integration runtime

A self-hosted IR is capable of:

- Running copy activity between a cloud data stores and a data store in private network.
- Dispatching the following transform activities against compute resources in on-premises or Azure Virtual Network: HDInsight Hive activity (BYOC-Bring Your Own Cluster), HDInsight Pig activity (BYOC), HDInsight MapReduce activity (BYOC), HDInsight Spark activity (BYOC), HDInsight Streaming activity (BYOC), Machine Learning Batch Execution activity, Machine Learning Update Resource activities, Stored Procedure activity, Data Lake Analytics U-SQL activity, Custom activity (runs on Azure Batch), Lookup activity, and Get Metadata activity.

The self-hosted IR is logically registered to the Data Factory and the compute used to support its functionalities is provided by you. Therefore there is no explicit location property for self-hosted IR. When used to perform data movement, the self-hosted IR extracts data from the source and writes into the destination.

Azure-SSIS Integration Runtime

To lift and shift existing SSIS workload, you can create an Azure-SSIS IR to natively execute SSIS packages. Selecting the right location for your Azure-SSIS IR is essential to achieve high performance in your extract-transform-load (ETL) workflows.

- The location of your Azure-SSIS IR does not need be the same as the location of your data factory, but it should be the same as the location of your own Azure SQL Database/Managed Instance server where SSISDB is to be hosted. This way, your Azure-SSIS Integration Runtime can easily access SSISDB without incurring excessive traffics between different locations.
- If you do not have an existing Azure SQL Database/Managed Instance server to host SSISDB, but you have on-premises data sources/destinations, you should create a new Azure SQL Database/Managed Instance server in the same location of a virtual network connected to your on-premises network. This way, you can create your Azure-SSIS IR using the new Azure SQL Database/Managed Instance server and joining that virtual network, all in the same location, effectively minimizing data movements across different locations.
- If the location of your existing Azure SQL Database/Managed Instance server where SSISDB is hosted is not the same as the location of a virtual network connected to your on-premises network, first create your Azure-SSIS IR using an existing Azure SQL Database/Managed Instance server and joining another virtual network in the same location, and then configure a virtual network to virtual network connection between different locations.

Determining which IR to use

Copy activity

For the Copy activity, it requires source and sink linked services to define the direction of data flow. The following logic is used to determine which integration runtime instance is used to perform the copy:

- Copying between two cloud data sources: when both source and sink linked services are using Azure IR, ADF will use the regional Azure IR if you specified, or auto determine a location of Azure IR if you choose the auto-resolve IR (default) as described in Integration runtime location section.
- Copying between a cloud data source and a data source in private network: if either source or sink linked service points to a self-hosted IR, the copy activity is executed on that self-hosted Integration Runtime.
- Copying between two data sources in private network: both the source and sink Linked Service must point to the same instance of integration runtime, and that integration runtime is used to execute the copy Activity.

Lookup and GetMetadata activity

The Lookup and GetMetadata activity is executed on the integration runtime associated to the data store linked service.

Transformation activity

Each transformation activity has a target compute Linked Service, which points to an integration runtime. This integration runtime instance is where the transformation activity is dispatched from.

Data Flow activity

Data Flow activity is executed on the integration runtime associated to it.

Summary

Azure Event Hubs provides applications the capability to process large volume of streaming data and scale out during exceptionally high-demand periods as and when required. Azure Event Hubs decouple the sending and receiving messages to manage the data processing. This helps eliminate the risk of overwhelming consumer application and data loss due to any unplanned interruptions.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

A pipeline JSON definition is embedded into an Activity JSON definition. True or False?

- True
- False

Question 2

You are moving data from an Azure Data Lake Gen2 store to Azure Synapse Analytics. Which Azure Data Factory integration runtime would be used in a data copy activity?

- Azure-SSIS
- Azure
- Self-hosted
- Pipelines

Ingesting-and-Transforming-data

Ingesting and Transforming data in Azure Data Factory

Azure Data Factory now has the ability to ingest and transform data natively using a number of graphical tools to make this happen. This evolution of the product makes it easier than before to set up data movement and transformation activities without the need to become involved in the JSON notation that was required in the past.

Learning Objectives

In this section, you will learn:

- How to setup Azure Data Factory
- Ingest data using the Copy Activity
- Transforming data with the Mapping Data Flow

Setting up Azure Data Factory

It is very easy to setup Azure Data Factory from within the Azure Portal, you only require the following information:

- **Name:** The name of the Azure Data Factory instance
- **Subscription:** The subscription in whch the ADF instance is created
- **Resource group:** The resource group where the ADF instance will reside
- **Version:** select V2 for the latest features
- **Location:** The datacenter location in which the instance is stored

Enable Git provides the capability to integrate the code that you create with a Git repository enabling you to source control the code that you would create. Simply define the GIT url, repository name, branch name and the root folder.

The screenshot shows the 'New data factory' creation interface. It includes the following fields:

- Name ***: An empty input field.
- Version ⓘ**: A dropdown menu showing 'V2'.
- Subscription ***: A dropdown menu showing 'chtestao'.
- Resource Group ***: A dropdown menu with options 'Select existing...' and 'Create new'. 'Select existing...' is currently selected.
- Location * ⓘ**: A dropdown menu showing 'South Central US'.
- Enable GIT ⓘ**: A checked checkbox.
- GIT URL ***: An empty input field.
- Repo name ***: An empty input field.
- Branch Name ***: An empty input field.
- Root folder ***: An empty input field.

Create

Alternatively, there are a number of different ways that you can provision the service programmatically. In this example you can see PowerShell at work to setup the environment.

```
#####
##          PART I: Creating an Azure Data Factory      ##
#####

# Sign in to Azure and set the WINDOWS AZURE subscription to work with
$SubscriptionId = "add your subscription in the quotes"

Add-AzureRmAccount
Set-AzureRmContext -SubscriptionId $SubscriptionId

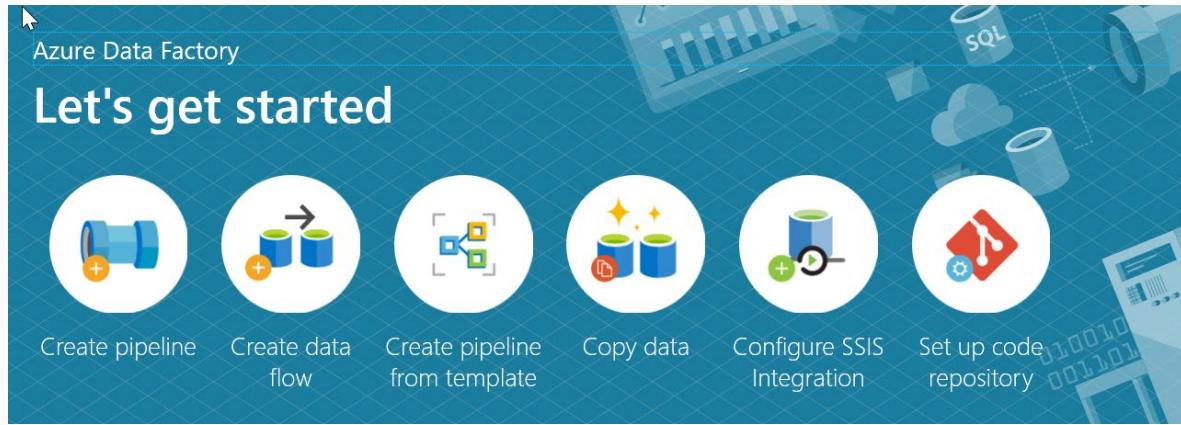
# register the Microsoft Azure Data Factory resource provider
Register-AzureRmResourceProvider -ProviderNamespace Microsoft.DataFactory
```

```
# DEFINE RESOURCE GROUP NAME AND LOCATION PARAMETERS
$resourceGroupName = "cto_ignite"
$rglocation = "West US 2"

# CREATE AZURE DATA FACTORY
New-AzureRmDataFactoryV2 -ResourceGroupName $resourceGroupName -Name
"ctoigniteADF" -Location $rglocation
```

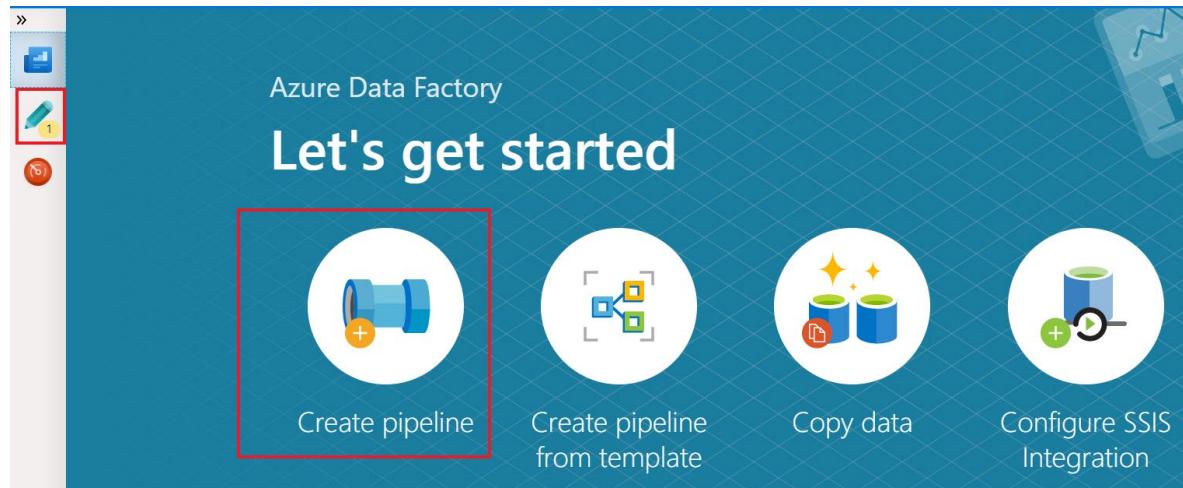
Ingesting data with the Copy Activity

Once the creation of the Data Factory instance is complete, you can go to the resource where you can begin to create your data pipelines by clicking on the **Author & Monitor** button. This will open up the following screen:

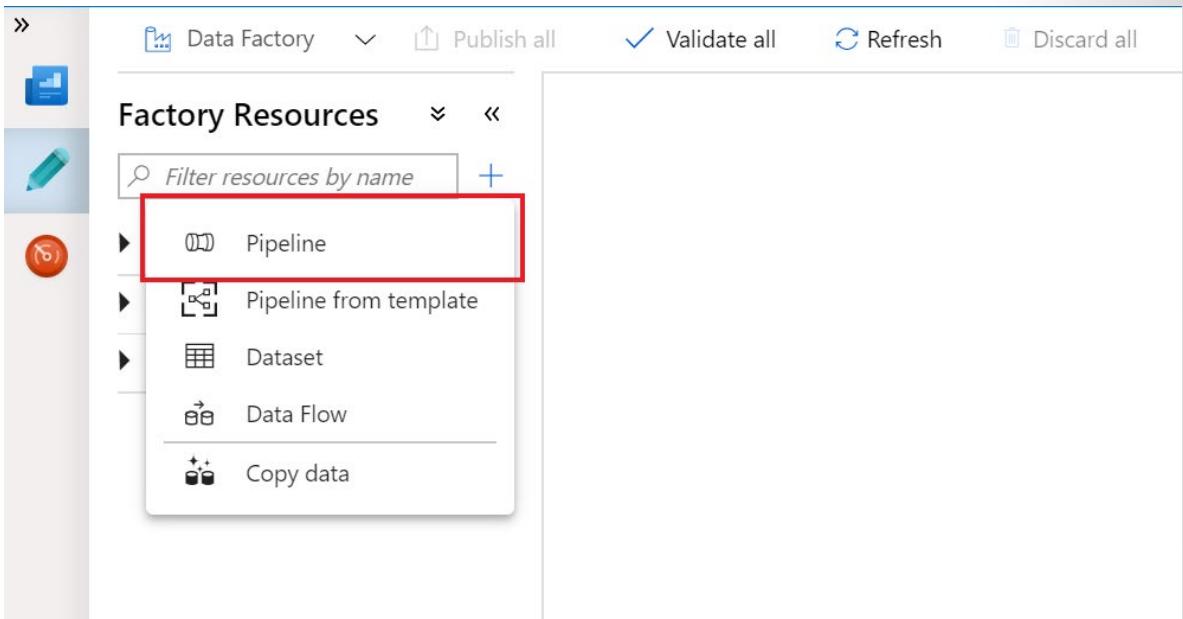


The first step in your pipeline is creating a Copy Activity that copies data between the source and destination using the following steps.

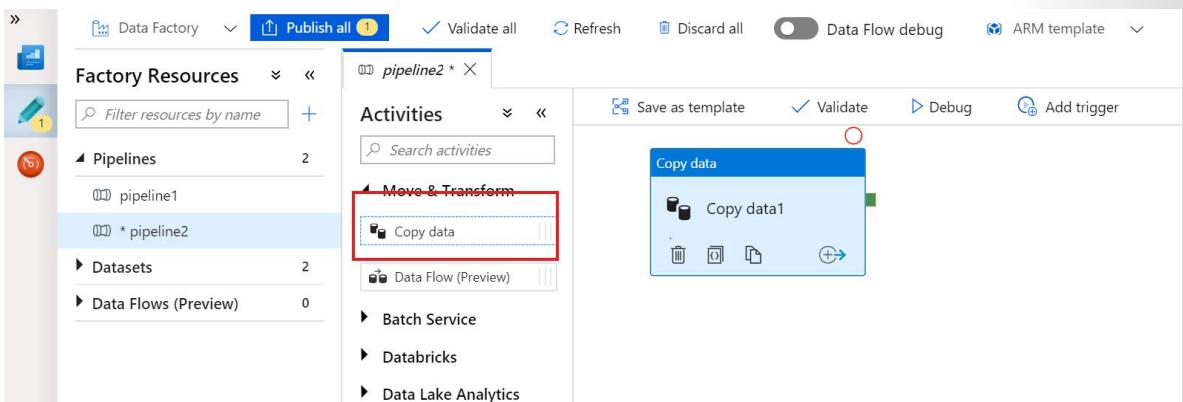
1. Open the **authoring canvas** by clicking on the **pencil icon** on the left sidebar or the create pipeline button to open the authoring canvas.



2. Create the **pipeline**. Click on the + button in the Factory Resources pane and select **Pipeline**.

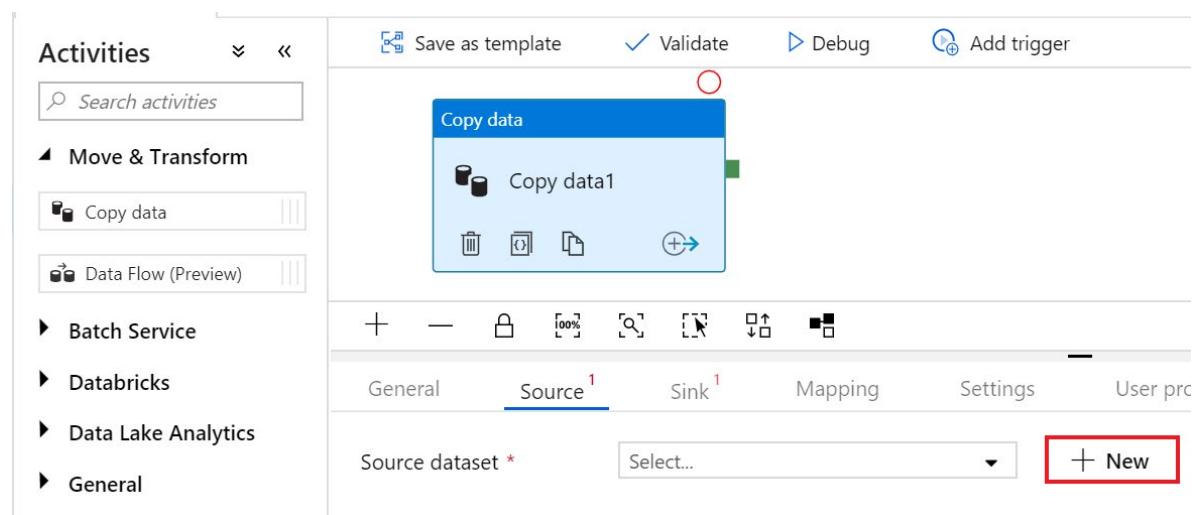


3. Add a **copy activity**. In the **Activities** pane, open the **Move and Transform** accordion and drag the **Copy Data** activity onto the pipeline canvas.



With the Copy Activity added, you then start to define the source data

1. In the **Source** tab of the Copy activity settings, click **+ New** to select a data source.



2. For example, In the data store list, select the **Amazon S3** tile and click **continue**

New Dataset

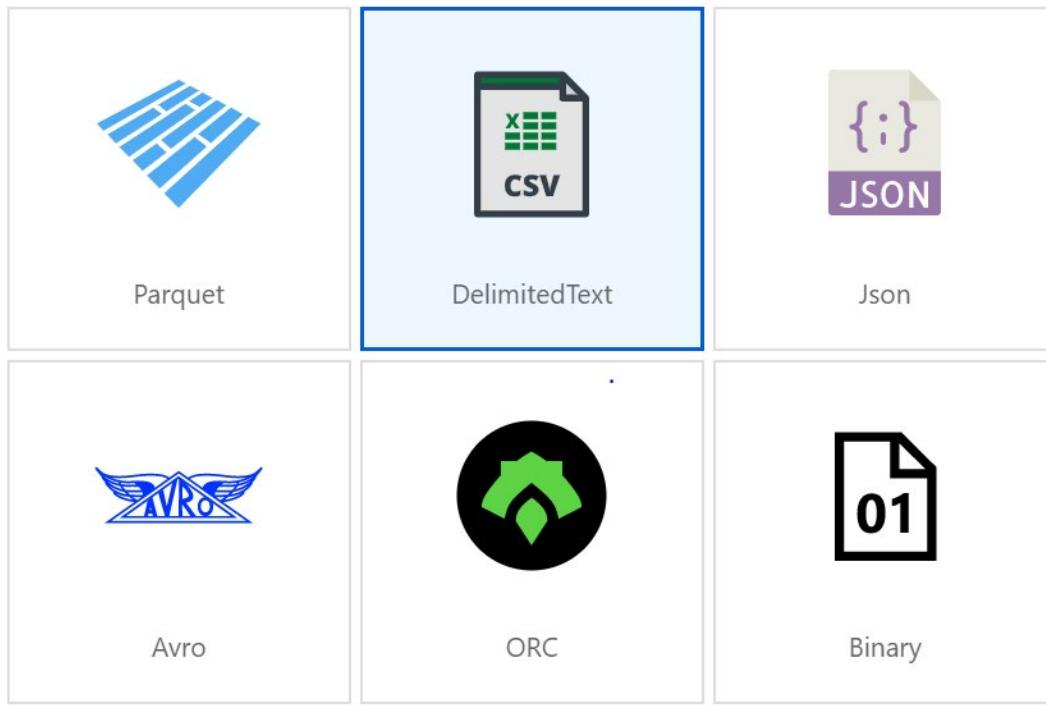
Select a Data Store

The screenshot shows a 'Select a Data Store' dialog with a search bar and a list of categories: All, Azure, Database, File, Generic protocol, NoSQL, Services and apps. Below the categories are three tiles: 'Amazon Marketplace Web Service' (with a shopping cart icon), 'Amazon Redshift' (with a blue server icon), and 'Amazon S3' (with an orange cloud icon). The 'Amazon S3' tile is highlighted with a blue border, indicating it has been selected.

3. In the **file format list**, select the **DelimitedText** format tile and click **continue**

Select format

Choose the format type of your data



4. In **Set Properties** window, give your dataset an understandable **name** and click on the **Linked Service dropdown**. If you have not created your S3 Linked Service, select **New**.

Set properties

Name

MoviesS3

Linked service *

Select...

Filter...

Select...

+ New

5. Specific to the S3 Linked Service configuration pane, specify your S3 **access key** and **secret key**. The data factory service encrypts credentials with certificates managed by Microsoft. For more information, see Data Movement Security Considerations. To verify your credentials are valid, click **Test Connection**. Click **Create** when finished.

New Linked Service (Amazon S3)

X

Name *

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Access Key ID *

Secret Access Key

Azure Key Vault

Secret Access Key *

Service URL

Annotations

+ New

► Advanced ⓘ

Create

Test connection

Cancel

6. Once you have created and selected the linked service, specify the rest of your dataset settings. These settings specify how and where in your connection you want to pull the data. Click **Finish** once completed.

Set properties X

Name
MoviesS3

Linked service *
AmazonS31 ▼

[Edit connection](#)

File path
daperlov-test / Directory / moviesDB.csv Browse ▾

First row as header

Import schema
 From connection/store From sample file None

OK Next->Advanced Back Cancel

7. To verify your dataset is configured correctly, click **Preview Data** in the Source tab of the copy activity to get a small snapshot of your data.

The screenshot shows the 'Data Preview' section of the Azure Data Factory pipeline editor. The pipeline is named 'pipeline2'. The preview is for the 'MoviesHTTP' activity, which uses the 'AmazonS31' linked service. The data is presented in a table format:

| movie | title | genres | year | Rating | Rotten Tomato |
|--------|---|---------------------------------|-------|--------|---------------|
| 108583 | Fawlty Towers (1975) | Comedy | -1980 | 1 | 54 |
| 32898 | Trip to the Moon, A (Voyage dans la lune, Le) | Action Adventure Fantasy Sci-Fi | 1902 | 7 | 80 |
| 7065 | Birth of a Nation, The | Drama War | 1915 | 6 | 92 |
| 7243 | Intolerance: Love's Struggle Throughout the Ages | Drama | 1915 | 4 | 82 |
| 62383 | 20,000 Leagues Under the Sea | Action Adventure Sci-Fi | 1915 | 9 | 92 |
| 8511 | Immigrant, The | Comedy | 1917 | 4 | 59 |
| 3309 | Dog's Life, A | Comedy | 1917 | 3 | 83 |
| 72626 | Billy Blazes, Esq. | Comedy Western | 1918 | 2 | 63 |
| 6987 | Cabinet of Dr. Caligari, The (Cabinet des Dr. Caligari., Das) | Crime Fantasy Horror | 1919 | 4 | 63 |

A blue button labeled 'Preview data' is visible on the right side of the preview area.

With the source data defined, then you will define the sink into which the data will be loaded. In this example the sink will be Azure Data Lake Storage Gen2 by performing the following steps:

1. In the **Sink** tab, click **+ New**

The screenshot shows the configuration of a 'Copy data' activity in the Azure Data Factory pipeline. The activity is named 'Copy data1'. The 'Sink' tab is selected. Below it, a red box highlights the '+ New' button next to the 'Sink dataset' dropdown, which contains a placeholder 'Select...'. The pipeline toolbar at the top includes icons for adding, deleting, locking, and saving.

2. Select the **Azure Data lake Storage Gen2** tile and click **continue**

New Dataset

Select a Data Store

All Azure Database File Generic protocol Services and apps

| | | |
|--|---|---|
|  Azure Blob Storage |  Azure Cosmos DB (MongoDB API) |  Azure Cosmos DB (SQL API) |
|  Azure Data Explorer (Kusto) |  Azure Data Lake Storage Gen1 |  Azure Data Lake Storage Gen2 |

3. In **Set Properties** sidenav, give your dataset an understandable **name** and click on the **Linked Service dropdown**. If you have not created your ADLS Linked Service, select **New**.

Set properties

Name
MoviesS3

Linked service *

Select...

4. In the ADLS linked service configuration pane, select your **authentication method** and **enter your credentials**. In the example below, an account key and selected my storage account from the drop down.

New Linked Service (Azure Data Lake Storage Gen2) X

Name *
ADLS

Description

Connect via integration runtime * i
AutoResolveIntegrationRuntime

Authentication method
Account key

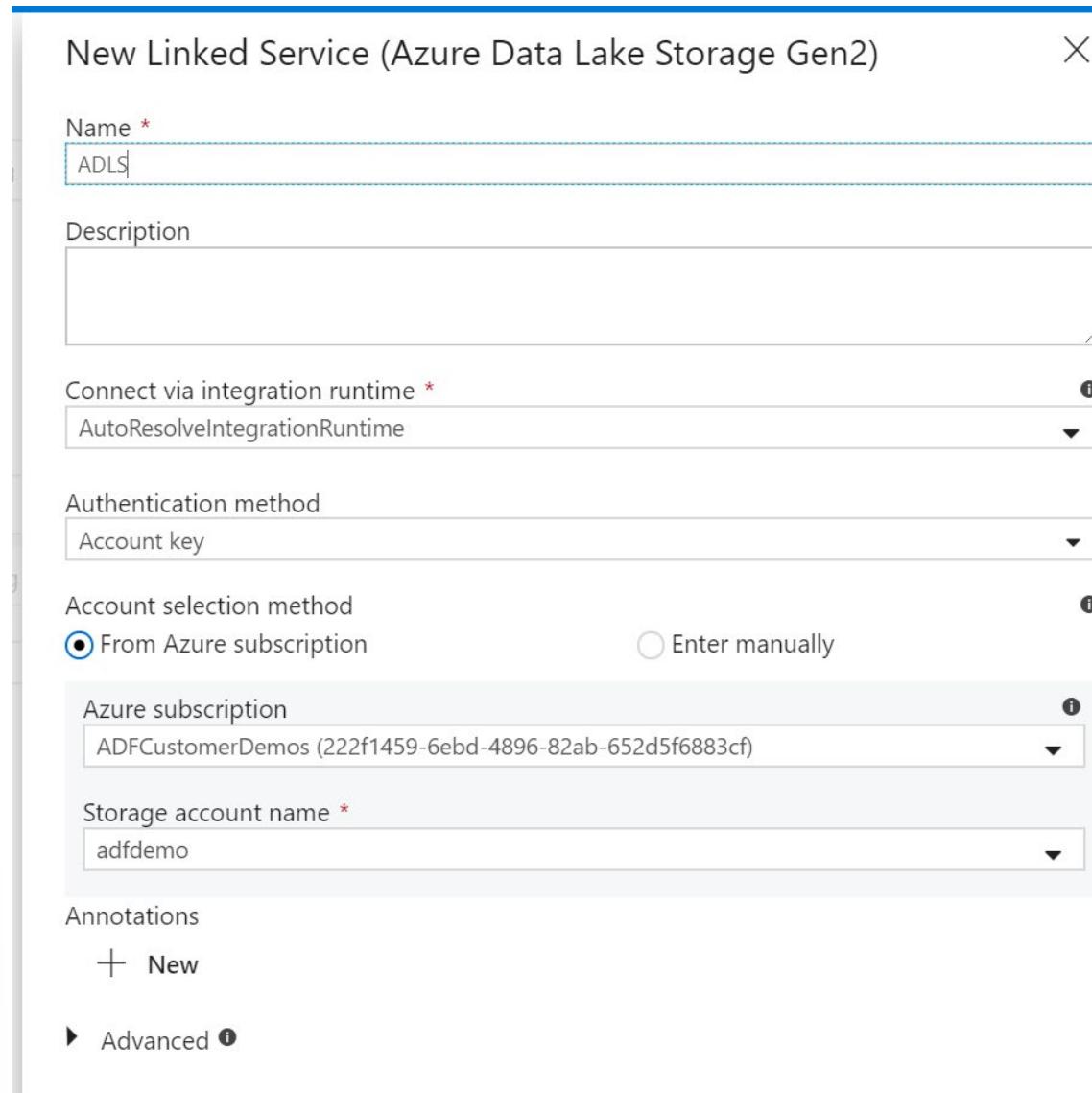
Account selection method
 From Azure subscription i Enter manually

Azure subscription i
ADFCustomerDemos (222f1459-6ebd-4896-82ab-652d5f6883cf)

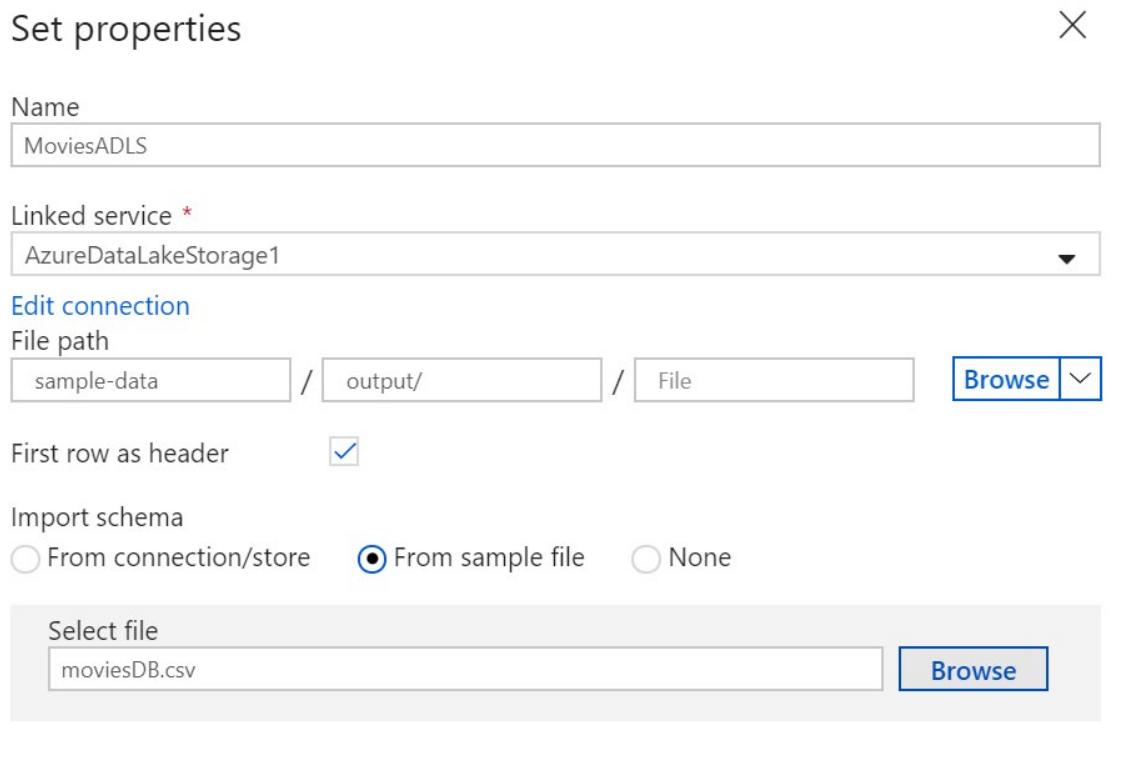
Storage account name *
adfdemo

Annotations
+ New i

► Advanced i

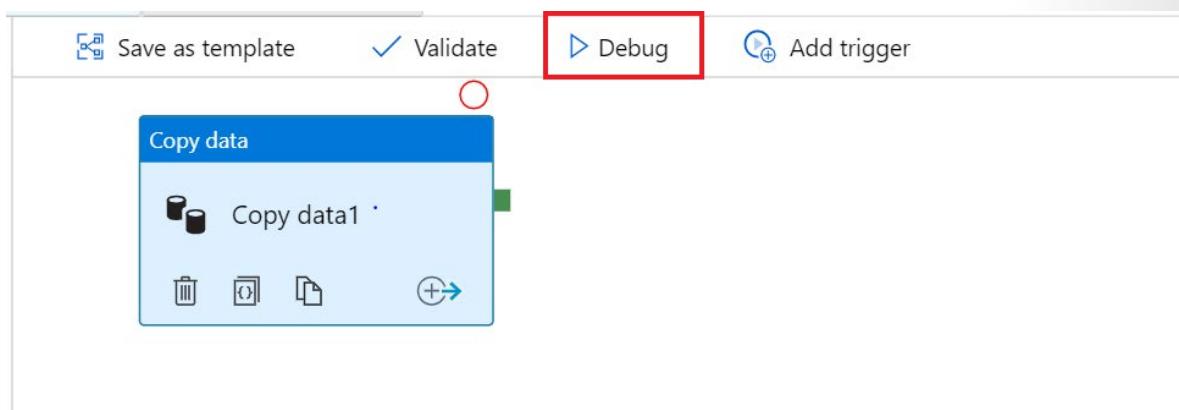


5. Once you have configured your linked service, enter in the **ADLS dataset configuration**. Click **finish** once completed.



At this point, you have fully configured your copy activity.

1. To test it out, click on the **Debug** button at the top of the pipeline canvas. This will start a pipeline debug run.

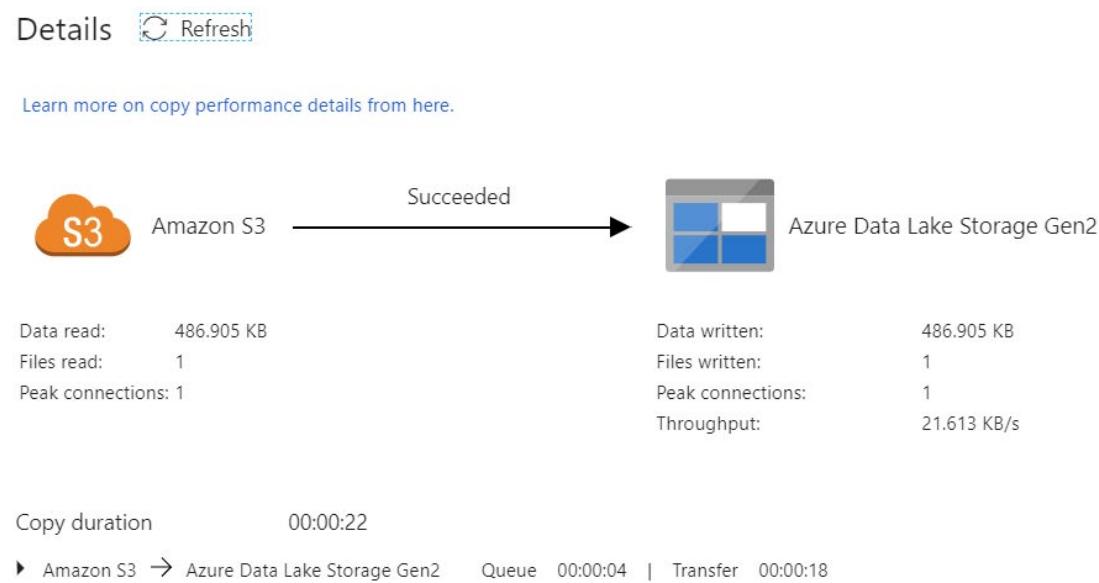


2. To monitor the progress of a pipeline debug run, click on the Output tab of the pipeline

| General | Parameters | Variables | Output |
|---|------------|-----------|----------|
| Pipeline Run ID: ee9431fe-c67c-420d-9b9e-2ece8fd990f1 | [@] | ● | |
| NAME | TYPE | RUN START | DURATION |

The table shows the pipeline run details. The 'NAME' column lists 'MoveFromS3ToA...', the 'TYPE' column lists 'Copy', the 'RUN START' column lists '07/10/2019 3:26 PM', and the 'DURATION' column lists '00:00:24'. The 'STATUS' column shows a green checkmark and 'Succeeded'. The 'ACTIONS' column contains a trash can icon and a refresh icon. The 'RUNID' column lists '18cb6b00-0407-48df-a928-19e7073449df'. A red box highlights the refresh icon in the 'ACTIONS' column.

- To view a more detailed description of the activity output, click on the eyeglasses icon. This will open up the copy monitoring screen which provides useful metrics such as Data read/written, throughput and in-depth duration statistics.

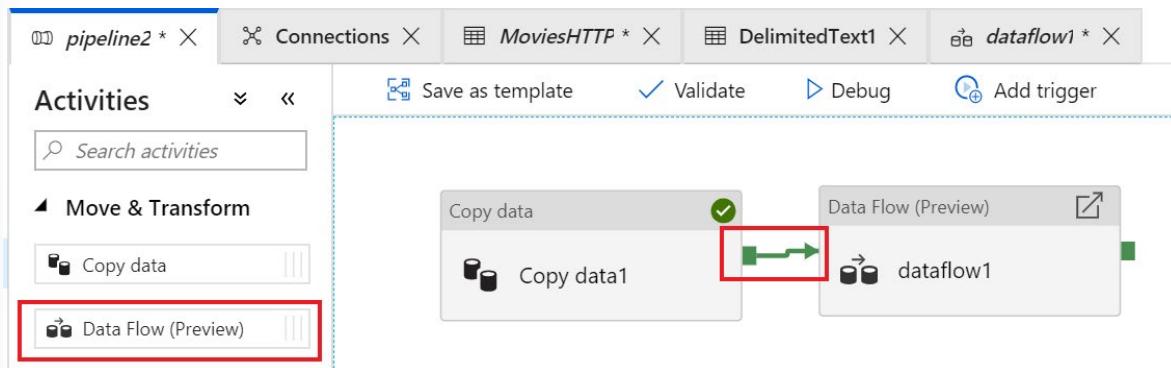


To verify the copy worked as expected, open up your ADLS gen2 storage account and check to see your file was written as expected

Transforming data with the Mapping Data Flow

You can natively perform data transformations with Azure Data Factory code free using the Mapping Data Flow task. Mapping Data Flows provide a fully visual experience with no coding required. Your data flows will run on your own execution cluster for scaled-out data processing. Data flow activities can be operationalized via existing Data Factory scheduling, control, flow, and monitoring capabilities.

A Mapping Data Flow task will typically appear in a Control Flow interface of a Pipeline designer after a Copy Data Activity.



However, it is very important that before you add the Mapping Data Flow, you should enable the Spark cluster that will run the Mapping Data Flow. This is achieved by a single click on the Data Flow Debug

slider located at the top of the authoring module. Data Flow clusters take 5-7 minutes to warm up and users are recommended to turn on debug first if they plan to do Data Flow development.



With the Mapping Data Flow added, and the Spark cluster running, this will enable you to perform the transformation, and run and preview the data. No coding is required as Azure Data Factory handles all the code translation, path optimization, and execution of your data flow jobs.

Adding source data to the Mapping Data Flow

Open the Mapping Data Flow canvas. Click on the Add Source button in the Data Flow canvas. In the source dataset dropdown, select your data source, in this case the ADLS Gen2 dataset is used in this example

| Source Settings | Source Options | Projection | Optimize | Inspect | Data Preview |
|----------------------|---|--|----------|---|------------------------------------|
| Output stream name * | MoviesADLS | | | <input checked="" type="checkbox"/> Documentation | |
| Source dataset * | DelimitedText1 | | | <input type="button" value="Edit"/> | <input type="button" value="New"/> |
| Options | <input checked="" type="checkbox"/> Allow schema drift <input type="checkbox"/> Infer drifted column types <input type="checkbox"/> Validate schema | | | | |
| Skip line count | | | | | |
| Sampling * | <input type="radio"/> Enable | <input checked="" type="radio"/> Disable | | | |

There are a copy of points to note:

- If your dataset is pointing at a folder with other files and you only want to use one file, you may need to create another dataset or utilize parameterization to make sure only a specific file is read
- If you have not imported your schema in your ADLS, but have already ingested your data, go to the dataset's 'Schema' tab and click 'Import schema' so that your data flow knows the schema projection.

Mapping Data Flow follows an extract, load, transform (ELT) approach and works with staging datasets that are all in Azure. Currently the following datasets can be used in a source transformation:

- Azure Blob Storage (JSON, Avro, Text, Parquet)
- Azure Data Lake Storage Gen1 (JSON, Avro, Text, Parquet)
- Azure Data Lake Storage Gen2 (JSON, Avro, Text, Parquet)
- Azure Synapse Analytics
- Azure SQL Database
- Azure CosmosDB

Azure Data Factory has access to over 80 native connectors. To include data from those other sources in your data flow, use the Copy Activity to load that data into one of the supported staging areas.

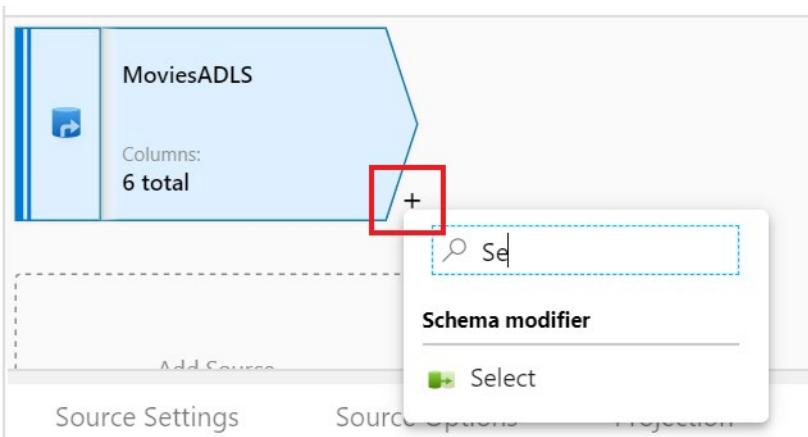
Once your debug cluster is warmed up, verify your data is loaded correctly via the Data Preview tab. Once you click the refresh button, Mapping Data Flow will show calculate a snapshot of what your data looks like when it is at each transformation.

The screenshot shows the Data Preview tab for a source named 'MoviesADLS'. The preview pane displays a table with the following data:

| movie | abc | title | abc | genres | abc | year | abc | Rating | abc | Rotten Tomat |
|--------|-----|----------------------------------|-----|---------------------------------|-----|-------|-----|--------|-----|--------------|
| 108583 | | Fawlty Towers (1975 | | Comedy | | -1980 | | 1 | | 54 |
| 32898 | | Trip to the Moon, A (Voyage ... | | Action Adventure Fantasy Sci... | | 1902 | | 7 | | 80 |
| 7065 | | Birth of a Nation, The | | Drama War | | 1915 | | 6 | | 92 |
| 7243 | | Intolerance: Love's Struggle ... | | Drama | | 1915 | | 4 | | 82 |
| 62383 | | 20,000 Leagues Under the Sea | | Action Adventure Sci-Fi | | 1915 | | 9 | | 92 |
| 8511 | | Immigrant, The | | Comedy | | 1917 | | 4 | | 59 |
| 3309 | | Dog's Life, A | | Comedy | | 1917 | | 3 | | 83 |

Using transformations in the Mapping Data Flow

With the source added, the Mapping Data Flow contains an array of transformations that you can use to modify the data. To add a transformation, click on the **+** icon next to your data source node and choosing the transformation under Schema modifier. In this example the Select transformation is being added.



To add additional transformations, a **+** icon will appear in each transformation that you add.

Azure Data Factory supports the following code-free transformations

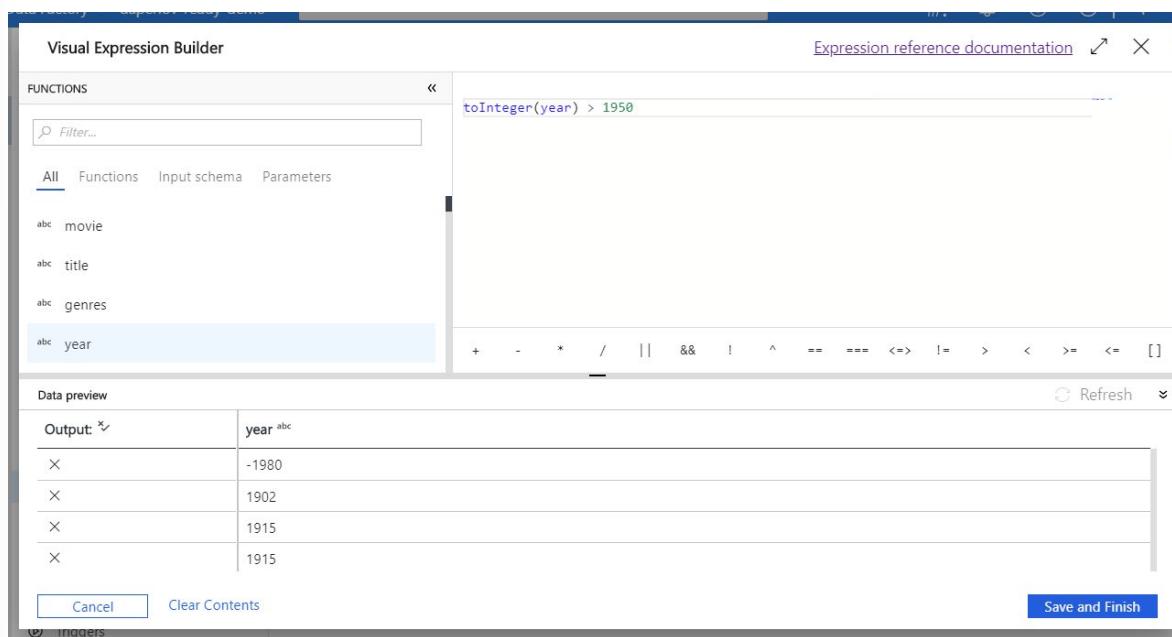
| Transformation type | Description |
|---------------------|---|
| Aggregate | The Aggregate transformation defines aggregations of columns in your data streams. Using the Expression Builder, you can define different types of aggregations such as SUM, MIN, MAX, and COUNT grouped by existing or computed columns. |
| Alter row | Use the Alter Row transformation to set insert, delete, update, and upsert policies on rows. You can add one-to-many conditions as expressions. These conditions should be specified in order of priority, as each row will be marked with the policy corresponding to the first-matching expression. Each of those conditions can result in a row (or rows) being inserted, updated, deleted, or upserted. Alter Row can produce both DDL & DML actions against your database. |
| Conditional split | The conditional split transformation routes data rows to different streams based on matching conditions. The conditional split transformation is similar to a CASE decision structure in a programming language. The transformation evaluates expressions, and based on the results, directs the data row to the specified stream. |
| Derived column | Use the derived column transformation to generate new columns in your data flow or to modify existing fields. |

| Transformation type | Description |
|---------------------|---|
| Exists | The exists transformation is a row filtering transformation that checks whether your data exists in another source or stream. The output stream includes all rows in the left stream that either exist or don't exist in the right stream. The exists transformation is similar to SQL WHERE EXISTS and SQL WHERE NOT EXISTS. |
| Filter | The Filter transforms allows row filtering based upon a condition. The output stream includes all rows that matching the filtering condition. The filter transformation is similar to a WHERE clause in SQL. |
| Join | Use the join transformation to combine data from two sources or streams in a mapping data flow. The output stream will include all columns from both sources matched based on a join condition. |
| Lookup | Use Lookup to add reference data from another source to your Data Flow. The Lookup transform requires a defined source that points to your reference table and matches on key fields. |
| New branch | Branching will take the current data stream in your data flow and replicate it to another stream. Use New Branch to perform multiple sets of operations and transformations against the same data stream. |
| Pivot | Use Pivot in ADF Data Flow as an aggregation where one or more grouping columns has its distinct row values transformed into individual columns. Essentially, you can Pivot row values into new columns (turn data into metadata). |
| Select | Use this transformation for column selectivity (reducing number of columns), alias columns and stream names, and reorder columns. |
| Sink | After you transform your data flow, you can sink the data into a destination dataset. In the sink transformation, choose a dataset definition for the destination output data. You can have as many sink transformations as your data flow requires. |
| Sort | The Sort transformation allows you to sort the incoming rows on the current data stream. The outgoing rows from the Sort Transformation will subsequently follow the ordering rules that you set. |
| Source | A source transformation configures your data source for the data flow. When designing data flows, your first step will always be configuring a source transformation. To add a source, click on the Add Source box in the data flow canvas. |

| Transformation type | Description |
|---------------------|---|
| Surrogate key | Use the Surrogate Key Transformation to add an incrementing non-business arbitrary key value to your data flow rowset. This is useful when designing dimension tables in a star schema analytical data model where each member in your dimension tables needs to have a unique key that is a non-business key, part of the Kimball DW methodology. |
| Union | Union will combine multiple data streams into one, with the SQL Union of those streams as the new output from the Union transformation. All of the schema from each input stream will be combined inside of your data flow, without needing to have a join key. |
| Unpivot | Use Unpivot in ADF mapping data flow as a way to turn an unnormalized dataset into a more normalized version by expanding values from multiple columns in a single record into multiple records with the same values in a single column. |
| Window | The Window transformation is where you will define window-based aggregations of columns in your data streams. In the Expression Builder, you can define different types of aggregations that are based on data or time windows (SQL OVER clause) such as LEAD, LAG, NTILE, CUMEDIST, RANK, etc.). A new field will be generated in your output that includes these aggregations. You can also include optional group-by fields. |

Some of the transformations that you can define have an **Data Flow Expression Builder** that will enable you to customize the functionality of a transformation using columns, fields, variables, parameters, functions from your data flow in these boxes.

To build the expression, use the Expression Builder, which is launched by clicking in the expression text box inside the transformation. You'll also sometimes see "Computed Column" options when selecting columns for transformation. When you click that, you'll also see the Expression Builder launched.



The Expression Builder tool defaults to the text editor option. the auto-complete feature reads from the entire Azure Data Factory Data Flow object model with syntax checking and highlighting.

Loading data using the Mapping Data Flow

In order to load data into a destination, you use the **Sink** transformation. You add a sink by clicking on the + icon next to your last transformation and clicking Sink under Destination. From there you can select the data set you want to define

Output stream name * SQLSink

Incoming stream * MoviesADLS

Sink dataset * Select... + New

Options

Allow schema drift !

Validate schema !

There may be additional settings to configure based on the dataset chosen

Summary

In this section, you have learned how Azure Data Factory can be used to create code free data ingestion and transformation activities natively. You have learned how the Copy Activity can be used to ingest data and place it in a sink. You then learned how the Mapping Data Flow can be used to perform a wide range of transformations

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which transformation in the Mapping Data Flow is used to routes data rows to different streams based on matching conditions?

- Lookup
- Conditional Split
- Select

Question 2

Which transformation is used to load data into a data store or compute resource?

- Window
- Source
- Sink

Integrate-Azure-Data-Factory-with-Databricks

Integrate Azure Data Factory with Databricks

Imagine you're part of an analytics team that has recently received a huge assignment of analyzing crime data of several metropolitan cities. The dataset that you received has detailed crime information for major cities. However, each dataset is formatted and structured differently and stored in different data stores. Each city uses a different category and terms for similar type of data. Your team is responsible to analyze all the datasets and report aggregated number of crimes per month in each city.

Your team has decided to leverage the capabilities of Azure Data Factory and Azure Databricks to ingest, transform, and aggregate the required data.

Learning objectives

In this module, you will:

- Use Azure Data Factory (ADF) to ingest data and create an ADF pipeline.
- Use Copy activities within ADF to copy data from one location to another.
- Use ADF to orchestrate data transformations using a Databricks Notebook activity.

ADF and Azure Databricks

You can use Azure Data Factory to ingest raw data collected from different sources and work with Azure Databricks to restructure it as per your requirements.

The integration of Azure Databricks with ADF allows you to add Databricks notebooks within an ADF pipeline to leverage the analytical and data transformation capabilities of Databricks. You can add a notebook within your data workflow to structure and transform raw data loaded into ADF from different sources. Once the data is transformed using Databricks, you can then load it to any data warehouse source.

Data ingestion and transformation using the collective capabilities of ADF and Azure Databricks essentially involves the following steps:

1. **Create Azure storage account** - The first step is to create an Azure storage account to store your ingested and transformed data.
2. **Create an Azure Data Factory** - Once you have your storage account setup, you need to create your Azure Data Factory using Azure portal.
3. **Create data workflow pipeline** - After your storage and ADF is up and running, you start by creating a pipeline, where the first step is to copy data from your source using ADF's Copy activity. Copy Activity allows you to copy data from different on-premises and cloud sources.
4. **Add Databricks notebook to pipeline** - Once your data is copied to ADF, you add your Databricks notebook to the pipeline, after copy activity. This notebook may contain syntax and code to transform and clean raw data as required.
5. **Perform analysis on data** - Now that your data is cleaned up and structured into the required format, you can use Databricks notebooks to further train or analyze it to output required results.

You have learned what Azure Data Factory is and how its integration with Azure Databricks helps you to load and transform your data. Now let's create an end-to-end sample data workflow.

Create Azure Storage account and the Azure Data Factory Instance

Let's start by creating an Azure storage account and an Azure Data Factory.

Create Azure Storage account

1. In the Azure portal, select + Create a resource, enter "storage account" into the Search the Marketplace box select Storage account - blob, file, table, queue from the results, and select Create.
2. In the Create storage account blade, enter the following:
 - Subscription: Select the subscription you are using for this module.
 - *Resource group*: Choose your module resource group.
 - *Storage account name*: Enter a unique name (make sure you see a green checkbox).
 - *Location*: Select the location you are using for resources in this module.
 - *Performance*: Select Standard.
 - *Account kind*: Select Storage (general purpose v1).
 - *Replication*: Select Locally-redundant storage (LRS).
3. Select Next: **Advanced >**.
4. In the Advanced tab, select the following:
 - *Secure transfer required*: Select Disabled
 - *Virtual network*: Select None
5. Select **Review + create**.
6. In the Review tab, select **Create**.

Acquire account name and key

1. Once provisioned, navigate to your storage account.
2. Select **Access keys** from the left-hand menu, and copy the Storage account name and key1 Key value into a text editor, such as Notepad, for later use.

Create the dwtemp container

1. Select Blobs from the left-hand menu, then select + **container** to create a new container.
2. Enter *dwtemp* for the container name.
3. Leave the public access level selected as *Private*
4. Select **OK**.

Create Azure Data Factory

1. Navigate to **Azure portal**⁵.

⁵ <https://portal.azure.com>

2. Select + **Create a resource**, type “data factory” into the search bar, select Data Factory in the results, and then select **Create**.
3. Set the following configuration on the Data Factory creation form:
 - *Name*: Enter a globally unique name, as indicated by a green checkmark.
 - *Subscription*: Select the subscription you are using for this workshop.
 - *Resource Group*: Choose Use existing, and select the resource group for this workshop.
 - *Version*: V2
 - *Location*: Select a region.
4. Select **Create** to provision ADF v2.

Using Azure Databricks with Azure Data Factory

You have your storage account and Azure Data Factory up and running, now it's time to switch to your Databricks workspace to complete rest of the workflow. We'll use a sample dataset to create an ADF pipeline and use sample notebooks to transform and analyze the data.

NOTE

To complete the following procedures you must already have deployed your Azure Databricks workspace in your Azure portal.

Clone the Databricks archive

1. From the Azure portal, navigate to your Azure Databricks workspace and select **Launch Workspace**.
2. Within the Workspace, using the command bar on the left, select **Workspace, Users**, and select your username (the entry with house icon).
3. In the blade that appears, select the downwards pointing chevron next to your name, and select **Import**.
4. On the Import Notebooks dialog, select URL and paste in the following URL:

```
https://github.com/MicrosoftDocs/mslearn-data-ingestion-with-azure-data-factory/blob/master/DBC/03-Data-Ingestion-Via-ADFdbc?raw=true
```

1. Select **Import**.
2. A folder named after the archive should appear. Select that folder.
3. The folder will contain one or more notebooks that you'll use in completing this lab.

Complete the following notebooks

1. **01 Getting Started** - This notebook contains instructions for setting up your storage account and Azure Data Factory (ADF). If you've already set up your storage account in the previous unit, you can skip this notebook.
2. **02 Data Ingestion** - In this notebook you create an ADF v2 pipeline to ingest data from a public dataset into your Azure Storage account. Once the data is ingested, you use Databricks notebook function to examine the data.

3. **03 Data Transformation** - This notebook contains instructions to create connectivity between your Azure Data Factory and Databricks workspace. You use a sample notebook to add to your ADF pipeline that will transform and restructure your data. You'll also perform some basic aggregation on the sample dataset to generate required reports.

Summary

Azure Data Factory allows you to ingest raw unstructured data from different sources. The integration between ADF and Azure Databricks helps you to create end-to-end data workflow to ingest, prepare, transform, and load your data into any destination storage in a required format and structure.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

What is the DataFrame method call to create temporary view?

- createOrReplaceTempView()
- ViewTemp()
- createTempViewDF()

Question 2

How do you create a DataFrame object?

- Using the createDataFrame() function
- content: create.DF
- An object is created by introducing a variable name and equating it to something like myDataFrameDF =

Question 3

Why do you chain methods (operations) myDataFrameDF.select().filter().groupBy()?

- To avoid the creation of temporary DataFrames as local variables
- it is the only way to do it
- to get accurate results

Module-Summary

Module Summary

In this module, you have learned how Azure Data Factory can be used to orchestrate the movement and transformation of data from a wide range of data platform technologies. You can also explain the capabilities of the technology and set up an end to end data pipeline that ingests and transforms data with Azure Databricks.

Learning Objectives

In this module, you have:

- Learned Azure Data Factory
- Understood Azure Data Factory Components
- Integrate Azure Data Factory with Databricks

Post Course Review

After the course, read **the white paper on data migration from on-premise relational data warehouse to Azure using Azure Data Factory⁶**.

Important

Remember

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

⁶ <https://azure.microsoft.com/en-us/resources/data-migration-from-on-premise-relational-data-warehouse-to-azure-data-lake-using-azure-data-factory/>

Answers

Question 1

Which Azure Data Factory process involves using compute services to produce data to feed production environments with cleansed data?

- Connect and collect
- Transform and enrich
- Publish
- Monitor

Explanation

Transform and enrich. This is used to invoke compute services to produce date to feed production environments with cleansed data. Connect and collect defines and connect all the required sources of data together. Publish load the data into a data store or analytical engine for your business users to point their business intelligence tools. Monitor is used to monitor the scheduled activities and pipelines for success and failure rates.

Question 2

Which Azure Data Factory component contains the transformation logic or the analysis commands of the Azure Data Factory's work?

- Linked Services
- Datasets
- Activities
- Pipelines

Explanation

Activities contains the transformation logic or the analysis commands of the Azure Data Factory's work. Linked Services are objects that are used to define the connection to data stores or compute resources in Azure. Datasets represent data structures within the data store that is being referenced by the Linked Service object. Pipelines are a logical grouping of activities.

Question 1

A pipeline JSON definition is embedded into an Activity JSON definition. True or False?

- True
- False

Explanation

False. an Activity JSON definition is embedded into a pipeline JSON definition.

Question 2

You are moving data from an Azure Data Lake Gen2 store to Azure Synapse Analytics. Which Azure Data Factory integration runtime would be used in a data copy activity?

- Azure-SSIS
- Azure
- Self-hosted
- Pipelines

Explanation

When moving data between Azure data platform technologies, the Azure Integration runtime is used when copying data between two Azure data platforms. Azure-SSIS IR is used when you lift and shift existing SSIS workload, while Self-hosted IR is used when working with data movement from private networks to the cloud and vice versa

Question 1

Which transformation in the Mapping Data Flow is used to routes data rows to different streams based on matching conditions?

- Lookup
- Conditional Split
- Select

Explanation

A Conditional Split transformation routes data rows to different streams based on matching conditions. The conditional split transformation is similar to a CASE decision structure in a programming language. A Lookup transformation is used to add reference data from another source to your Data Flow. The Lookup transform requires a defined source that points to your reference table and matches on key fields. A Select transformation is used for selecting and/or defining your columns

Question 2

Which transformation is used to load data into a data store or compute resource?

- Window
- Source
- Sink

Explanation

A Sink transformation allows you to choose a dataset definition for the destination output data. You can have as many sink transformations as your data flow requires. A Window transformation is where you will define window-based aggregations of columns in your data streams. A Source transformation configures your data source for the data flow. When designing data flows, your first step will always be configuring a source transformation.

Question 1

What is the DataFrame method call to create temporary view?

- createOrReplaceTempView()
- ViewTemp()
- createTempViewDF()

Explanation

createOrReplaceTempView(). This is the correct method name for creating temporary views in DataFrames and createTempViewDF() are incorrect method name.

Question 2

How do you create a DataFrame object?

- Using the createDataFrame() function
- content: create.DF
- An object is created by introducing a variable name and equating it to something like myDataFrameDF =

Explanation

An object is created by introducing a variable name and equating it to something like myDataFrameDF =. There is no such method as createDataFrame() for creating a DataFrame and DF.create() is incorrect syntax and wrong way to create DataFrame Objects.

Question 3

Why do you chain methods (operations) myDataFrameDF.select().filter().groupBy()?

- To avoid the creation of temporary DataFrames as local variables
- it is the only way to do it
- to get accurate results

Explanation

To avoid the creation of temporary DataFrames as local variables. You could have written the above as - tempDF1 = myDataFrameDF.select(), tempDF2 = tempDF1.filter() and then tempDF2.groupBy(). This is syntactically equivalent, but, notice how you now have extra local variables.

Module 8 Securing Azure Data Platforms

Moudle Introduction Securing Azure Data Platforms

In this module, students will learn how Azure provides a multi-layered security model to protect your data. The students will explore how security can range from setting up secure networks and access keys, to defining permission through to monitoring with Advanced Threat Detection across a range of data stores.

Learning objectives

In this module, you will learn how:

- An introduction to security
- Key security components
- Securing Storage Accounts and Data Lake Storage
- Securing Data Stores
- Securing Streaming Data

Introduction to Security

Introduction to Security

Every system, architecture, and application needs to be designed with security in mind. There's just too much at risk. For instance, a denial of service attack could prevent your customer from reaching your web site or services and block you from doing business. Defacement of your website damages your reputation. And a data breach is perhaps worst of all — as it can ruin hard-earned trust, while causing significant personal and financial harm. As administrators, developers, and IT management, we all must work to guarantee the security of our systems.

Imagine you work at a company called Contoso Shipping, and you're spearheading the development of drone deliveries in rural areas—while having truck drivers leverage mobile apps to deliver to urban areas. You're in the process of moving a lot of Contoso Shipping's infrastructure to the cloud to maximize efficiency, as well as moving several physical servers in the company's data center to Azure virtual machines. Your team plans on creating a hybrid solution, with some of the servers remaining on-premises, so you'll need a secure, high-quality connection between the new virtual machines and the existing network.



Additionally, Contoso Shipping has some out-of-network devices that are part of your operations. You are using network-enabled sensors in your drones that send data to Azure Event Hubs, while delivery drivers use mobile apps to get route maps and record signatures for receipt of shipments. These devices and apps must be securely authenticated before data can be sent to or from them.

So how do you keep your data secure? and what about Government organizations?

Learning objectives

In this module, you will learn about:

- Shared Security responsibility
- A layered approach to security
- The Azure Security Center
- Azure Government

Security as a Shared Responsibility

As computing environments move from customer-controlled data centers to cloud data centers, the responsibility of security also shifts. Security is now a concern shared both by cloud providers, organizations and individuals. For every application and solution, it's important to understand what's your responsibility and what's Azure's responsibility.

Share security responsibility with Azure

The first shift that organizations typically make is from on-premises data centers to infrastructure as a service (IaaS). With IaaS, you are leveraging the lowest-level service and asking Azure to create virtual machines (VMs) and virtual networks. At this level, it's still your responsibility to patch and secure your operating systems and software, as well as configure your network to be secure. At Contoso Shipping, you are taking advantage of IaaS when you start using Azure VMs instead of your on-premises physical servers. In addition to the operational advantages, you receive the security advantage of having outsourced concern over protecting the physical parts of the network.

Moving to platform as a service (PaaS) outsources a lot of security concerns. At this level, Azure is taking care of the operating system and of most foundational software like database management systems. Everything is updated with the latest security patches and can be integrated with Azure Active Directory for access controls. PaaS also comes with a lot of operational advantages. Rather than building whole infrastructures and subnets for your environments by hand, you can "point and click" within the Azure portal or run automated scripts to bring complex, secured systems up and down, and scale them as needed. Contoso Shipping uses Azure Event Hubs for ingesting telemetry data from drones and trucks — as well as a web app with an Azure Cosmos DB back end with its mobile apps — which are all examples of PaaS.

With software as a service (SaaS), you outsource almost everything. SaaS is software that runs with an internet infrastructure. The code is controlled by the vendor but configured to be used by the customer. Like so many companies, Contoso Shipping uses Office 365, which is a great example of SaaS!

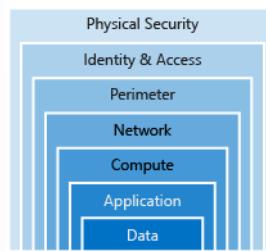
| Responsibility | On-prem | IaaS | PaaS | SaaS |
|--|---------|------|------|------|
| Data governance & rights management | ■ | ■ | ■ | ■ |
| Client endpoints | ■ | ■ | ■ | ■ |
| Account & access management | ■ | ■ | ■ | ■ |
| Identity & directory infrastructure | ■ | ■ | ■ | ■ |
| Application | ■ | ■ | ■ | ■ |
| Network controls | ■ | ■ | ■ | ■ |
| Operating system | ■ | ■ | ■ | ■ |
| Physical hosts | ■ | ■ | ■ | ■ |
| Physical network | ■ | ■ | ■ | ■ |
| Physical datacenter | ■ | ■ | ■ | ■ |
| ■ Microsoft ■ Customer | | | | |

A layered approach to security

Defense in depth is a strategy that employs a series of mechanisms to slow the advance of an attack aimed at acquiring unauthorized access to information. Each layer provides protection so that if one layer is breached, a subsequent layer is already in place to prevent further exposure. Microsoft applies a layered approach to security, both in physical data centers and across Azure services. The objective of defense in depth is to protect and prevent information from being stolen by individuals who are not authorized to access it.

Defense in depth can be visualized as a set of concentric rings, with the data to be secured at the center. Each ring adds an additional layer of security around the data. This approach removes reliance on any

single layer of protection and acts to slow down an attack and provide alert telemetry that can be acted upon, either automatically or manually. Let's take a look at each of the layers.



Data

In almost all cases, attackers are after data:

- Stored in a database
- Stored on disk inside virtual machines
- Stored on a SaaS application such as Office 365
- Stored in cloud storage

It's the responsibility of those storing and controlling access to data to ensure that it's properly secured. Often, there are regulatory requirements that dictate the controls and processes that must be in place to ensure the confidentiality, integrity, and availability of the data.



Application

- Ensure applications are secure and free of vulnerabilities.
- Store sensitive application secrets in a secure storage medium.

- Make security a design requirement for all application development.

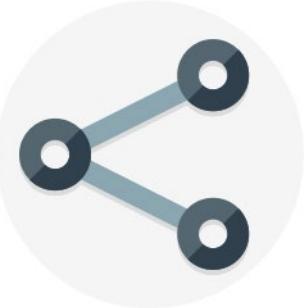
Integrating security into the application development life cycle will help reduce the number of vulnerabilities introduced in code. We encourage all development teams to ensure their applications are secure by default, and that they're making security requirements non-negotiable.



Compute

- Secure access to virtual machines.
- Implement endpoint protection and keep systems patched and current.

Malware, unpatched systems, and improperly secured systems open your environment to attacks. The focus in this layer is on making sure your compute resources are secure, and that you have the proper controls in place to minimize security issues.



Networking

- Limit communication between resources.
- Deny by default.
- Restrict inbound internet access and limit outbound, where appropriate.
- Implement secure connectivity to on-premises networks.

At this layer, the focus is on limiting the network connectivity across all your resources to allow only what is required. By limiting this communication, you reduce the risk of lateral movement throughout your network.



Perimeter

- Use distributed denial of service (DDoS) protection to filter large-scale attacks before they can cause a denial of service for end users.
- Use perimeter firewalls to identify and alert on malicious attacks against your network.

At the network perimeter, it's about protecting from network-based attacks against your resources. Identifying these attacks, eliminating their impact, and alerting you when they happen are important ways to keep your network secure.



Identity and access

- Control access to infrastructure and change control.
- Use single sign-on and multi-factor authentication.
- Audit events and changes.

The identity and access layer is all about ensuring identities are secure, access granted is only what is needed, and changes are logged.



Physical security

- Physical building security and controlling access to computing hardware within the data center is the first line of defense.

With physical security, the intent is to provide physical safeguards against access to assets. This ensures that other layers can't be bypassed, and loss or theft is handled appropriately.

Summary

We've seen here that Azure helps a lot with your security concerns. But security is still a **shared responsibility**. How much of that responsibility falls on us depends on which model we use with Azure.

We use the *defense in depth* rings as a guideline for considering what protections are adequate for our data and environments.

The Azure Security Center

A great place to start when examining the security of your Azure-based solutions is **Azure Security Center**. Security Center is a monitoring service that provides threat protection across all of your services both in Azure, and on-premises. Security Center can:

- Provide security recommendations based on your configurations, resources, and networks.
- Monitor security settings across on-premises and cloud workloads, and automatically apply required security to new services as they come online.
- Continuously monitor all your services, and perform automatic security assessments to identify potential vulnerabilities before they can be exploited.
- Use machine learning to detect and block malware from being installed on your virtual machines and services. You can also define a list of allowed applications to ensure that only the apps you validate are allowed to execute.
- Analyze and identify potential inbound attacks, and help to investigate threats and any post-breach activity that might have occurred.
- Provide just-in-time access control for ports, reducing your attack surface by ensuring the network only allows traffic that you require.

Azure Security Center is part of the **Center for Internet Security (CIS) recommendations¹**.

¹ <https://www.cisecurity.org/cis-benchmarks/>



Available tiers

Azure Security Center is available in two tiers:

1. *Free*. Available as part of your Azure subscription, this tier is limited to assessments and recommendations of Azure resources only.
2. *Standard*. This tier provides a full suite of security-related services including continuous monitoring, threat detection, just-in-time access control for ports, and more.

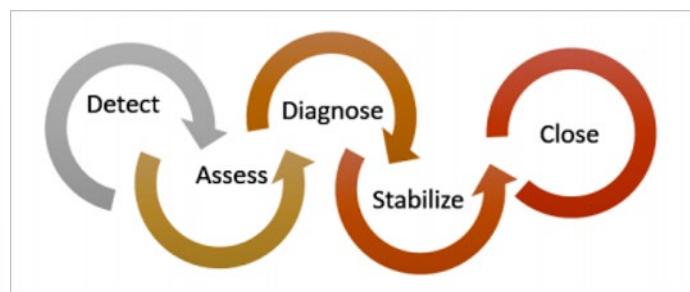
To access the full suite of Azure Security Center services, you will need to upgrade to a Standard tier subscription. You can access the 60-day free trial from within the Azure Security Center dashboard in the Azure portal. After the 60-day trial period is over, Azure Security Center is \$15 per node per month.

Usage scenarios

You can integrate Security Center into your workflows and use it in many ways. Here are two examples.

1. Use Security Center for incident response.

Many organizations learn how to respond to security incidents only after suffering an attack. To reduce costs and damage, it's important to have an incident response plan in place before an attack occurs. You can use Azure Security Center in different stages of an incident response.



You can use Security Center during the detect, assess, and diagnose stages. Here are examples of how Security Center can be useful during the three initial incident response stages:

- *Detect*. Review the first indication of an event investigation. For example, you can use the Security Center dashboard to review the initial verification that a high-priority security alert was raised.
 - *Assess*. Perform the initial assessment to obtain more information about the suspicious activity. For example, obtain more information about the security alert.
 - *Diagnose*. Conduct a technical investigation and identify containment, mitigation, and workaround strategies. For example, follow the remediation steps described by Security Center in that particular security alert.
2. Use Security Center recommendations to enhance security.

You can reduce the chances of a significant security event by configuring a security policy, and then implementing the recommendations provided by Azure Security Center.

- A *security policy* defines the set of controls that are recommended for resources within that specified subscription or resource group. In Security Center, you define policies according to your company's security requirements.
- Security Center analyzes the security state of your Azure resources. When Security Center identifies potential security vulnerabilities, it creates recommendations based on the controls set in the security policy. The recommendations guide you through the process of configuring the needed security controls. For example, if you have workloads that do not require the *Azure SQL Database Transparent Data Encryption* (TDE) policy, turn off the policy at the subscription level and enable it only in the resources groups where SQL TDE is required.

[!IMPORTANT]

To upgrade a subscription to the Standard tier, you must be assigned the role of *Subscription Owner*, *Subscription Contributor*, or *Security Admin*.

Azure Government

Azure Government is a cloud environment specifically built to meet compliance and security requirements for US government. This mission-critical cloud delivers breakthrough innovation to U.S. government customers and their partners. Azure Government applies to government at any level — from state and local governments to federal agencies including Department of Defense agencies.

While there are multiple cloud providers in the public sector, not many can offer the unique capabilities required by state and local and federal government agencies. Azure Government provides hybrid flexibility, deep security, and broad compliance coverage across regulatory standards.

The key difference between Microsoft Azure and Microsoft Azure Government is that Azure Government is a sovereign cloud. It's a physically separated instance of Azure, dedicated to U.S. government workloads only. It's built exclusively for government agencies and their solution providers.

Azure Government is designed for highly sensitive data, enabling government customers to safely transfer mission-critical workloads to the cloud.

With Azure Government, agencies can:

- **Modernize their services with the trusted cloud.** Leverage state-of-the-art security to protect citizen data, applications, and hardware in an environment built to exceed the highest compliance requirements.
- **Embrace the new era of agile government.** With flexible hybrid solutions, effectively engage with citizens through consistent interactions and scalable solutions, in the cloud or on-premises. Environments can be tailored through options across platforms, tools, and services, allowing agencies to architect their cloud at their own pace.
- **Advance their mission.** Harness the power of Internet of Things (IoT) devices, analytics and deep learning, to enable actionable insights. Rapidly innovate government processes with tailored solutions that deliver enhanced citizen experiences.
- **Expand the possibilities.** Advanced computing capabilities powered by an intelligent government cloud empower agencies as never before. Leverage advanced analysis and storage tools in an intelligent and powerful government cloud to uncover new insights in data.

Summary

In this module, we discussed the basic concepts for protecting your infrastructure and data when you work in the cloud.

Defense in depth is the overriding theme - think about security as a multi-layer, multi-vector concern. Threats come from places we don't expect, and they can come with strength that will surprise us.



Azure has out-of-the-box help for a great deal of the security issues we face. One of the first steps we should take is assessing how much help from Azure we can use based on whether we're leveraging IaaS, PaaS, or SaaS.

Azure Security Center centralizes much of the help Azure has to offer. It provides a single dashboard, with a view into many of your services, and helps make sure you are following best practices. Continuously updated machine learning algorithms help identify whether the latest threats are aimed at your resources. And it helps your organization mitigate threats.

In addition, you learned about Azure Government, the sovereign cloud built specifically for the U.S. Government. Whether you are creating solutions for Federal, state or local government, you can take advantage of the world-class security, protection, and compliance of Azure Government.

Of course, this module is introductory. Security is a deep and complex topic, so whatever your cloud approach, an ongoing security education is necessary. But this module should get you started in the right direction, so you know what you need to learn next.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Cloud security is a shared responsibility between you and your cloud provider. Which category of cloud services requires the greatest security effort on your part?

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as a service (SaaS)

Question 2

Which one of the following is a key difference between global Azure and Azure Government?

- Azure Government has a marketplace from which you can deploy pre-built images from Microsoft and partners.
- Azure Government has a portal from which you can manage your resources.
- Azure Government is a physically separate instance of Azure.

Key Security Components

Key Security Components

There are key security components that all IT professionals should be aware of when building solutions for an organization. Data Engineers play an important roles in helping to design and implement a range of security features that meets the defense in depth approach that this module focuses on.

Learning objectives

In this module, you will learn about:

- Network security
- Identity and access management
- Encryption capabilities built into Azure
- Advanced Threat Protection

Network Security

Securing your network from attacks and unauthorized access is an important part of any architecture. Here, we'll take a look at what network security looks like, how to integrate a layered approach into your architecture, and how Azure can help you provide network security for your environment. It is important that you work with your Azure Administrator and Azure Solution Architect as you may need to rely on their specialist knowledge in this area.

A layered approach to network security

You've probably noticed that a common theme throughout this module is the emphasis of a layered approach to security, and this is no different at the network layer. It's not enough to just focus on securing the network perimeter, or focusing on the network security between services inside a network. A layered approach provides multiple levels of protection, so that if an attacker gets through one layer, there are further protections in place to limit further attack.

Let's take a look at how Azure can provide the tools for a layered approach to securing your network footprint.



Internet protection**

If we start on the perimeter of the network, we're focused on limiting and eliminating attacks from the internet. We suggest first assessing the resources that are internet-facing, and to only allow inbound and outbound communication where necessary. Make sure you identify all resources that are allowing inbound network traffic of any type, and then ensure they are restricted to only the ports and protocols required. Azure Security Center is a great place to look for this information, because it will identify internet-facing resources that don't have network security groups associated with them, as well as resources that are not secured behind a *firewall*.

What is a Firewall?

A firewall is a service that grants server access based on the originating IP address of each request. You create firewall rules that specify ranges of IP addresses. Only clients from these granted IP addresses will be allowed to access the server. Firewall rules, generally speaking, also include specific network protocol and port information.

To provide inbound protection at the perimeter, you have several choices.

- **Azure Firewall** is a managed, cloud-based, network security service that protects your Azure Virtual Network resources. It is a fully stateful firewall as a service with built-in high availability and unrestricted cloud scalability. Azure Firewall provides inbound protection for non-HTTP/S protocols. Examples of non-HTTP/S protocols include: Remote Desktop Protocol (RDP), Secure Shell (SSH), and File Transfer Protocol (FTP). It also provides outbound, network-level protection for all ports and protocols, and application-level protection for outbound HTTP/S.
- **Azure Application Gateway** is a load balancer that includes a Web Application Firewall (WAF) that provides protection from common, known vulnerabilities in websites. It is specifically designed to protect HTTP traffic.
- **Network virtual appliances (NVAs)** are ideal options for non-HTTP services or advanced configurations, and are similar to hardware firewall appliances.
- **Azure Storage Firewalls** can be used to provide protection at a service level, giving you the ability to enable a firewall on services such as Azure Storage Accounts, SQL Database and SQL Data Warehouse as well as Cosmos DB.

Stopping Distributed Denial of Service (DDoS) attacks

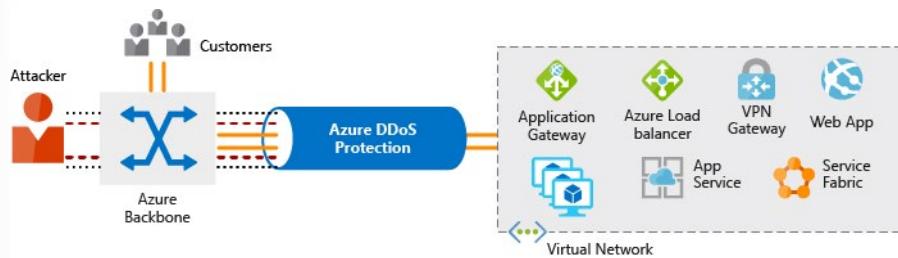
The following services would not likely be configured by a Data Engineer, but it is worth being mindful that these options are available.

Any resource exposed on the internet is at risk of being attacked by a denial of service attack. These types of attacks attempt to overwhelm a network resource by sending so many requests that the resource becomes slow or unresponsive.

When you combine **Azure DDoS Protection** with application design best practices, you help provide defense against DDoS attacks. DDoS Protection leverages the scale and elasticity of Microsoft's global network to bring DDoS mitigation capacity to every Azure region. The Azure DDoS Protection service protects your Azure applications by scrubbing traffic at the Azure network edge before it can impact your service's availability. Within a few minutes of attack detection, you are notified using Azure Monitor metrics.

This diagram shows network traffic flowing into Azure from both customers and an attacker. Azure DDoS protection identifies the attacker's attempt to overwhelm the network and blocks further traffic from

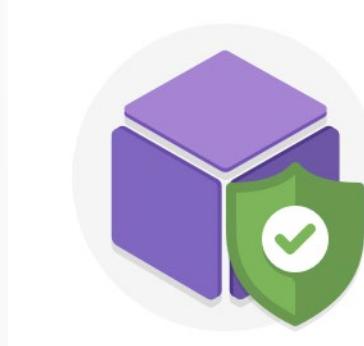
reaching Azure services. Legitimate traffic from customers still flows into Azure without any interruption of service.



Azure DDoS Protection provides the following service tiers:

- **Basic.** The Basic service tier is automatically enabled as part of the Azure platform. Always-on traffic monitoring and real-time mitigation of common network-level attacks provide the same defenses that Microsoft's online services use. Azure's global network is used to distribute and mitigate attack traffic across regions.
- **Standard.** The Standard service tier provides additional mitigation capabilities that are tuned specifically to Microsoft Azure Virtual Network resources. DDoS Protection Standard is simple to enable and requires no application changes. Protection policies are tuned through dedicated traffic monitoring and machine learning algorithms. Policies are applied to public IP addresses which are associated with resources deployed in virtual networks, such as Azure Load Balancer and Application Gateway. DDoS standard protection can mitigate the following types of attacks:
 - Volumetric attacks. The attacker's goal is to flood the network layer with a substantial amount of seemingly legitimate traffic.
 - Protocol attacks. These attacks render a target inaccessible, by exploiting a weakness in the layer 3 and layer 4 protocol stack.
 - Resource (application) layer attacks. These attacks target web application packets to disrupt the transmission of data between hosts.

Controlling the traffic inside your virtual network



Virtual network security

Once inside a virtual network (VNet), it's crucial that you limit communication between resources to only what is required.

For communication between virtual machines, *Network Security Groups* (NSGs) are a critical piece to restrict unnecessary communication.

Network Security Groups allow you to filter network traffic to and from Azure resources in an Azure virtual network. An NSG can contain multiple inbound and outbound security rules that enable you to filter traffic to and from resources by source and destination IP address, port, and protocol. They provide a list of allowed and denied communication to and from network interfaces and subnets, and are fully customizable.

You can completely remove public internet access to your services by restricting access to service endpoints. With service endpoints, Azure service access can be limited to your virtual network.



Network integration

It's common to have existing network infrastructure that needs to be integrated to provide communication from on-premises networks or to provide improved communication between services in Azure. There are a few key ways to handle this integration and improve the security of your network.

Virtual private network (VPN) connections are a common way of establishing secure communication channels between networks. Connection between Azure Virtual Network and an on-premises VPN device is a great way to provide secure communication between your network and your VNet on Azure.

To provide a dedicated, private connection between your network and Azure, you can use Azure ExpressRoute. ExpressRoute lets you extend your on-premises networks into the Microsoft cloud over a private connection facilitated by a connectivity provider. With ExpressRoute, you can establish connections to Microsoft cloud services, such as Microsoft Azure, Office 365, and Dynamics 365. This improves the security of your on-premises communication by sending this traffic over the private circuit instead of over the public internet. You don't need to allow access to these services for your end users over the public internet, and you can send this traffic through appliances for further traffic inspection.

Summary

A layered approach to network security helps reduce your risk of exposure through network-based attacks. Azure provides several services and capabilities to secure your internet-facing resource, internal resources, and communication between on-premises networks. These features make it possible to create secure solutions on Azure.

You can also combine multiple Azure networking and security services to manage your network security and provide increased layered protection. For example, you can use Azure Firewall to protect inbound and outbound traffic to the Internet, and Network Security Groups to limit traffic to resources inside your virtual networks.

Identity and Access

Network perimeters, firewalls, and physical access controls used to be the primary protection for corporate data. But network perimeters have become increasingly porous with the explosion of bring your own device (BYOD), mobile apps, and cloud applications.

Identity has become the new primary security boundary. Therefore, proper authentication and assignment of privileges is critical to maintaining control of your data.

Your company, Contoso Shipping, is focused on addressing these concerns right away. Your team's new hybrid cloud solution needs to account for mobile apps that have access to secret data when an authorized user is signed in — in addition to having shipping vehicles constantly send a stream of telemetry data that is critical to optimizing the company's business.

Authentication and authorization

Two fundamental concepts that need to be understood when talking about identity and access control are authentication and authorization. They underpin everything else that happens and occur sequentially in any identity and access process:

- *Authentication* is the process of establishing the identity of a person or service looking to access a resource. It involves the act of challenging a party for legitimate credentials, and provides the basis for creating a security principal for identity and access control use. It establishes if they are who they say they are.
- *Authorization* is the process of establishing what level of access an authenticated person or service has. It specifies what data they're allowed to access and what they can do with it.

NOTE

Authentication is sometimes shortened to *AuthN*, and authorization is sometimes shortened to *AuthZ*.

Azure provides services to manage both authentication and authorization through Azure Active Directory (Azure AD).

What is Azure Active Directory?

Azure AD is a cloud-based identity service. It has built in support for synchronizing with your existing on-premises Active Directory or can be used stand-alone. This means that all your applications, whether on-premises, in the cloud (including Office 365), or even mobile can share the same credentials. Administrators and developers can control access to internal and external data and applications using centralized rules and policies configured in Azure AD.

Azure AD provides services such as:

- **Authentication.** This includes verifying identity to access applications and resources, and providing functionality such as self-service password reset, multi-factor authentication (MFA), a custom banned password list, and smart lockout services.
- **Single-Sign-On (SSO).** SSO enables users to remember only one ID and one password to access multiple applications. A single identity is tied to a user, simplifying the security model. As users change roles or leave an organization, access modifications are tied to that identity, greatly reducing the effort needed to change or disable accounts.
- **Application management.** You can manage your cloud and on-premises apps using Azure AD Application Proxy, SSO, the My apps portal (also referred to as Access panel), and SaaS apps.

- **Business to business (B2B) identity services.** Manage your guest users and external partners while maintaining control over your own corporate data
- Business-to-Customer (B2C) identity services. Customize and control how users sign up, sign in, and manage their profiles when using your apps with services.
- **Device Management.** Manage how your cloud or on-premises devices access your corporate data.

Let's explore of a few of these in more detail.

Single sign-on

The more identities a user has to manage, the greater the risk of a credential-related security incident. More identities mean more passwords to remember and change. Password policies can vary between applications and, as complexity requirements increase, it becomes increasingly difficult for users to remember them.

Now, consider the logistics of managing all those identities. Additional strain is placed on help desks as they deal with account lockouts and password reset requests. If a user leaves an organization, tracking down all those identities and ensuring they are disabled can be challenging. If an identity is overlooked, this could allow access when it should have been eliminated.

With single sign-on (SSO), users need to remember only one ID and one password. Access across applications is granted to a single identity tied to a user, simplifying the security model. As users change roles or leave an organization, access modifications are tied to the single identity, greatly reducing the effort needed to change or disable accounts. Using single sign-on for accounts will make it easier for users to manage their identities and will increase the security capabilities in your environment.



SSO with Azure Active Directory

By leveraging Azure AD for SSO you'll also have the ability to combine multiple data sources into an intelligent security graph. This security graph enables the ability to provide threat analysis and real-time identity protection to all accounts in Azure AD, including accounts that are synchronized from your on-premises AD. By using a centralized identity provider, you'll have centralized the security controls, reporting, alerting, and administration of your identity infrastructure.

As Contoso Shipping integrates its existing Active Directory instance with Azure AD, you will make controlling access consistent across the organization. Doing so will also greatly simplify the ability to sign into email and Office 365 documents without having to reauthenticate.

:::column-end:::

:::row-end:::

Multi-factor authentication

Multi-factor authentication (MFA) provides additional security for your identities by requiring two or more elements for full authentication. These elements fall into three categories:

- *Something you know*
- *Something you possess*
- *Something you are*

Something you know would be a password or the answer to a security question. **Something you possess** could be a mobile app that receives a notification or a token-generating device. **Something you are** is typically some sort of biometric property, such as a fingerprint or face scan used on many mobile devices.

Using MFA increases security of your identity by limiting the impact of credential exposure. An attacker who has a user's password would also need to have possession of their phone or their face in order to fully authenticate. Authentication with only a single factor verified is insufficient, and the attacker would be unable to use those credentials to authenticate. The benefits this brings to security are huge, and we can't emphasize enough the importance of enabling MFA wherever possible.

Azure AD has MFA capabilities built in and will integrate with other third-party MFA providers. It's provided free of charge to any user who has the Global Administrator role in Azure AD, because these are highly sensitive accounts. All other accounts can have MFA enabled by purchasing licenses with this capability — as well as assigning a license to the account.

For Contoso Shipping, you decide to enable MFA any time a user is signing in from a non-domain-connected computer — which includes the mobile apps your drivers use.

Providing identities to services

It's usually valuable for services to have identities. Often, and against best practices, credential information is embedded in configuration files. With no security around these configuration files, anyone with access to the systems or repositories can access these credentials and risk exposure.

Azure AD addresses this problem through two methods: service principals and managed identities for Azure services.



Service principals

To understand service principals, it's useful to first understand the words **identity** and **principal**, because of how they are used in the identity management world.

An **identity** is just a thing that can be authenticated. Obviously, this includes users with a user name and password, but it can also include applications or other servers, which might authenticate with secret keys or certificates. As a bonus definition, an **account** is data associated with an identity.

A **principal** is an identity acting with certain roles or claims. Usually, it is not useful to consider identity and principal separately, but think of using `sudo` on a Bash prompt in Linux or on Windows using "run as Administrator." In both those cases, you are still logged in as the same identity as before, but you've changed the role under which you are executing. Groups are often also considered principals because they can have rights assigned.

A **service principal** is an identity that is used by a service or application. And like other identities, it can be assigned roles.



Managed identities for Azure services

The creation of service principals can be a tedious process, and there are a lot of touch points that can make maintaining them difficult. Managed identities for Azure services are much easier and will do most of the work for you.

A managed identity can be instantly created for any Azure service that supports it—and the list is constantly growing. When you create a managed identity for a service, you are creating an account on the Azure AD tenant. The Azure infrastructure will automatically take care of authenticating the service and managing the account. You can then use that account like any other Azure AD account, including securely letting the authenticated service access other Azure resources.

:::column-end:::

:::row-end:::

Role-based access control

Roles are sets of permissions, like "Read-only" or "Contributor", that users can be granted to access an Azure service instance.

Identities are mapped to roles directly or through group membership. Separating security principals, access permissions, and resources provides simple access management and fine-grained control. Administrators are able to ensure the minimum necessary permissions are granted.

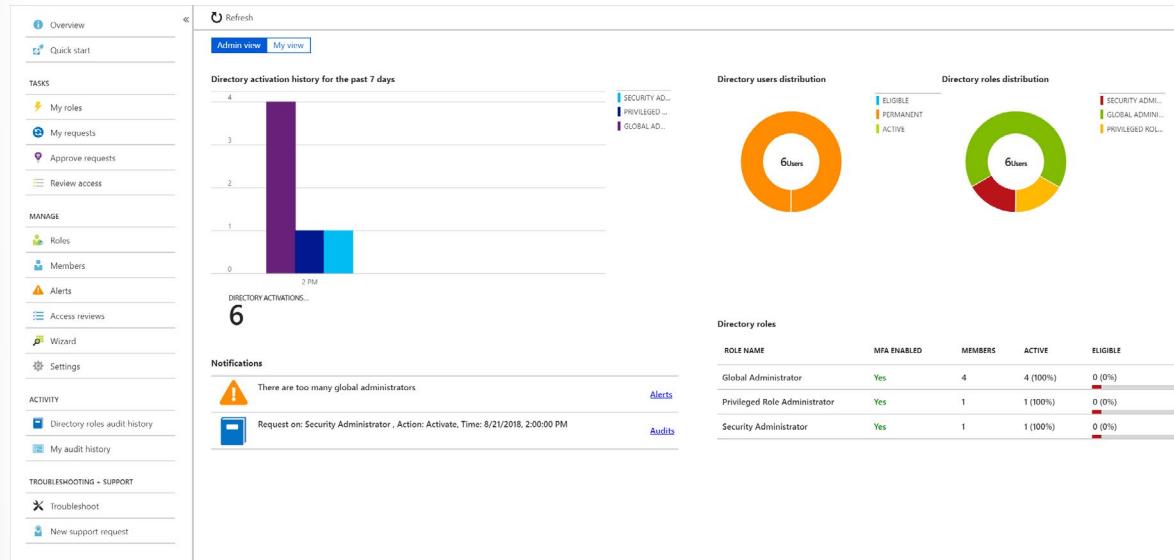
Roles can be granted at the individual service instance level, but they also flow down the Azure Resource Manager hierarchy.

Here's a diagram that shows this relationship. Roles assigned at a higher scope, like an entire subscription, are inherited by child scopes, like service instances.



Privileged Identity Management

In addition to managing Azure resource access with role-based access control (RBAC), a comprehensive approach to infrastructure protection should consider including the ongoing auditing of role members as their organization changes and evolves. Azure AD Privileged Identity Management (PIM) is an additional, paid-for offering that provides oversight of role assignments, self-service, and just-in-time role activation and Azure AD and Azure resource access reviews.



Summary

Identity allows us to maintain a security perimeter, even outside our physical control. With single sign-on and appropriate role-based access configuration, we can always be sure who has the ability to see and manipulate our data and infrastructure.

Encryption

For most organizations, data is the most valuable and irreplaceable asset. Encryption serves as the last and strongest line of defense in a layered security strategy.

Contoso Shipping knows that encryption is the only protection its data has once it leaves the data center and is stored on mobile devices that could potentially be hacked or stolen.

What is encryption?

Encryption is the process of making data unreadable and unusable to unauthorized viewers. To use or read the encrypted data, it must be *decrypted*, which requires the use of a secret key. There are two top-level types of encryption: **symmetric** and **asymmetric**.

Symmetric encryption uses the same key to encrypt and decrypt the data. Consider a desktop password manager application. You enter your passwords and they are encrypted with your own personal key (your key is often derived from your master password). When the data needs to be retrieved, the same key is used, and the data is decrypted.

Asymmetric encryption uses a public key and private key pair. Either key can encrypt but a single key can't decrypt its own encrypted data. To decrypt, you need the paired key. Asymmetric encryption is used for things like Transport Layer Security (TLS) (used in HTTPS) and data signing.

Both symmetric and asymmetric encryption play a role in properly securing your data. Encryption is typically approached in two ways:

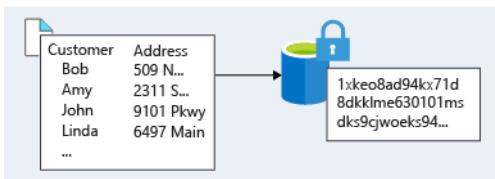
1. Encryption at rest
2. Encryption in transit

Encryption at rest

Data at rest is the data that has been stored on a physical medium. This could be data stored on the disk of a server, data stored in a database, or data stored in a storage account. Regardless of the storage mechanism, encryption of data at rest ensures that the stored data is unreadable without the keys and secrets needed to decrypt it. If an attacker was to obtain a hard drive with encrypted data and did not have access to the encryption keys, the attacker would not compromise the data without great difficulty.

The actual data that is encrypted could vary in its content, usage, and importance to the organization. This could be financial information critical to the business, intellectual property that has been developed by the business, personal data about customers or employees that the business stores, and even the keys and secrets used for the encryption of the data itself.

Here's a diagram that shows what encrypted customer data might look like as it sits in a database.



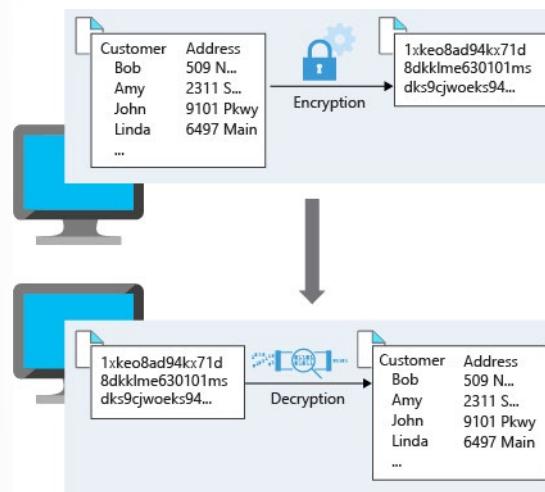
Encryption in transit

Data in transit is the data actively moving from one location to another, such as across the internet or through a private network. Secure transfer can be handled by several different layers. It could be done by encrypting the data at the application layer prior to sending it over a network. HTTPS is an example of application layer in transit encryption.

You can also set up a secure channel, like a virtual private network (VPN), at a network layer, to transmit data between two systems.

Encrypting data in transit protects the data from outside observers and provides a mechanism to transmit data while limiting risk of exposure.

This diagram shows the process. Here, customer data is encrypted as it's sent over the network. Only the receiver has the secret key that can decrypt the data to a usable form.



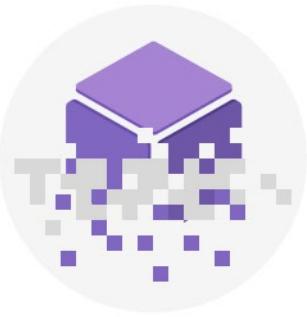
Encryption on Azure

Let's take a look at some ways that Azure enables you to encrypt data across services.

Encrypt Raw



Azure Storage Service Encryption for data at rest helps you protect your data to meet your organizational security and compliance commitments. With this feature, the Azure storage platform automatically encrypts your data before persisting it to Azure Managed Disks, Azure Blob storage, Azure Files, or Azure Queue storage, and decrypts the data before retrieval. The handling of encryption, encryption at rest, decryption, and key management in Storage Service Encryption is transparent to applications using the services.



Encrypt virtual machine disks

Storage Service Encryption provides low-level encryption protection for data written to physical disk, but how do you protect the virtual hard disks (VHDs) of virtual machines? If malicious attackers gained access to your Azure subscription and got the VHDs of your virtual machines, how would you ensure they would be unable to access the stored data?

Azure Disk Encryption is a capability that helps you encrypt your Windows and Linux IaaS virtual machine disks. Azure Disk Encryption leverages the industry-standard BitLocker feature of Windows and the dm-crypt feature of Linux to provide volume encryption for the OS and data disks. The solution is integrated with Azure Key Vault to help you control and manage the disk encryption keys and secrets (and you can use managed service identities for accessing Key Vault).

For Contoso Shipping, using VMs was one of the first moves toward the cloud. Having all the VHDs encrypted is a very easy, low-impact way to ensure that you are doing all you can to secure your company's data.



Encrypt Databases

Transparent data encryption (TDE) helps protect Azure SQL Database and Azure Data Warehouse against the threat of malicious activity. It performs real-time encryption and decryption of the database, associated backups, and transaction log files at rest without requiring changes to the application. By default, TDE is enabled for all newly deployed Azure SQL Database instances.

TDE encrypts the storage of an entire database by using a symmetric key called the database encryption key. By default, Azure provides a unique encryption key per logical SQL Server instance and handles all the details. Bring your own key (BYOK) is also supported with keys stored in Azure Key Vault (see below).

Because TDE is enabled by default, you are confident that Contoso Shipping has the proper protections in place for data stored in the company's databases.



Encrypt secrets

We've seen that the encryption services all use keys to encrypt and decrypt data, so how do we ensure that the keys themselves are secure? Corporations may also have passwords, connection strings, or other sensitive pieces of information that they need to securely store. In Azure, we can use **Azure Key Vault** to protect our secrets.

Azure Key Vault is a centralized cloud service for storing your application secrets. Key Vault helps you control your applications' secrets by keeping them in a single, central location and by providing secure access, permissions control, and access logging capabilities. It is useful for a variety of scenarios:

- *Secrets management.* You can use Key Vault to securely store and tightly control access to tokens, passwords, certificates, *Application Programming Interface* (API) keys, and other secrets.
- *Key management.* You also can use Key Vault as a key management solution. Key Vault makes it easier to create and control the encryption keys used to encrypt your data.
- *Certificate management.* Key Vault lets you provision, manage, and deploy your public and private *Secure Sockets Layer/ Transport Layer Security* (SSL/ TLS) certificates for your Azure, and internally connected, resources more easily.
- *Store secrets backed by hardware security modules* (HSMs). The secrets and keys can be protected either by software, or by FIPS 140-2 Level 2 validated HSMs.

The benefits of using Key Vault include:

- *Centralized application secrets.* Centralizing storage for application secrets allows you to control their distribution, and reduces the chances that secrets may be accidentally leaked.
- *Securely stored secrets and keys.* Azure uses industry-standard algorithms, key lengths, and HSMs, and access requires proper authentication and authorization.
- *Monitor access and use.* Using Key Vault, you can monitor and control access to company secrets.
- *Simplified administration of application secrets.* Key Vault makes it easier to enroll and renew certificates from public Certificate Authorities (CAs). You can also scale up and replicate content within regions, and use standard certificate management tools.
- *Integrate with other Azure services.* You can integrate Key Vault with storage accounts, container registries, event hubs and many more Azure services.

Because Azure AD identities can be granted access to use Azure Key Vault secrets, applications with managed service identities enabled can automatically and seamlessly acquire the secrets they need.

Summary

As you may know, encryption is often the last layer of defense from attackers and is an important piece of a layered approach to securing your systems. Azure provides built-in capabilities and services to encrypt and protect data from unintended exposure. Protection of customer data stored within Azure services is of paramount importance to Microsoft and should be included in any design. Foundational services such as Azure Storage, Azure Virtual Machines, Azure SQL Database, and Azure Key Vault can help secure your environment through encryption.

Auditing and Azure Advanced Threat Protection

Even if your organization has made concerted efforts to secure its infrastructure and data, it still pragmatic to keep an eye on the activities that are taking place on your estate. Sometimes you will want to monitor unauthorized access to resources, or detect unusual patterns such as staff logging on when they are on sabbatical, there are a number of ways this can be achieved

Auditing

Auditing can be enabled on a wide range of Azure resources and can help you monitor and identify any gaps in your security. The level of auditing can be specific to a service and a user and to the level of activity that is being undertaken in a resource.

Azure Advanced Threat Protection

Azure Advanced Threat Protection (Azure ATP) is a cloud-based security solution that identifies, detects, and helps you investigate advanced threats, compromised identities, and malicious insider actions directed at your organization.

Azure ATP is capable of detecting known malicious attacks and techniques, security issues, and risks against your network.

Azure ATP components

Azure ATP consists of several components.

Azure ATP portal

Azure ATP has its own portal, through which you can monitor and respond to suspicious activity. The Azure ATP portal allows you to create your Azure ATP instance, and view the data received from Azure ATP sensors. You can also use the portal to monitor, manage, and investigate threats in your network environment. You can sign in to the Azure ATP portal at <https://portal.atp.azure.com>. You must sign in with a user account that is assigned to an Azure AD security group that has access to the Azure ATP portal.

Azure ATP sensor

Azure ATP sensors are installed directly on your domain controllers. The sensor monitors domain controller traffic without requiring a dedicated server or configuring port mirroring.

Azure ATP cloud service

Azure ATP cloud service runs on Azure infrastructure and is currently deployed in the United States, Europe, and Asia. Azure ATP cloud service is connected to Microsoft's intelligent security graph.

The screenshot shows the Azure Advanced Threat Protection (ATP) Timeline page for the tenant 'contoso-corp'. The timeline lists several events:

- 4:04 PM Today**: Honeytoken activity (Updated). The following activities were performed by Bob Minion:
 - Logged in to 2 computers via Contoso-DC.
 - Authenticated from 2 computers using Kerberos when accessing 5 resources against Contoso-DC.
 - Authenticated from ITARGOET-T470S using NTLM against corporate resources via Contoso-DC.Started at 3:08 PM Jan 22, 2018.
- 3:23 PM Jan 22, 2018**: Remote execution attempt detected. The following remote execution attempts were performed on Contoso-DC from ALICE-DESKTOP:
 - Attempted remote execution of one or more WMI methods by AdminUser.
- 3:06 PM Jan 22, 2018**: Suspicious service creation. AdminUser created 10 services in order to execute potentially malicious commands on Contoso-DC.
- 3:03 PM Jan 22, 2018**: Brute force attack using LDAP simple bind. 200 password guess attempts were made on 2 accounts from ALICE-DESKTOP. 2 account passwords were successfully guessed.
- 2:59 PM Jan 22, 2018**: Reconnaissance using account enumeration. Suspicious account enumeration activity using Kerberos protocol, originating from ALICE-DESKTOP, was detected. The attacker performed a total of 101 guess attempts for account names, 2 guess attempts matched existing account names in Active Directory.
- 12:38 PM Jan 21, 2018**: Malicious replication of directory services. Malicious replication requests were attempted by Alice Liddel, from ALICE-DESKTOP against Contoso-DC.
- 11:59 AM Jan 21, 2018**: Reconnaissance using DNS. Suspicious DNS activity was observed, originating from ALICE-DESKTOP (which is not a DNS server) against Contoso-DC.

Purchasing Azure Advanced Threat Protection

Azure ATP is available as part of the Enterprise Mobility + Security E5 suite (EMS E5) and as a standalone license. You can acquire a license directly from the **Enterprise Mobility + Security Pricing Options²** page or through the Cloud Solution Provider (CSP) licensing model. It is not available to purchase via the Azure portal.

² <https://www.microsoft.com/en-us/enterprise-mobility-security/compare-plans-and-pricing>

Summary

In this module, you learned the key security components for protecting your infrastructure and data when you work in the cloud. You began with focussing on the wide range of network security measures that can provide the first line of defense. You then learned how you can also control access to data with the various identity and access management features that are available. You then learned how encryption can be used as the last line of defense and how Advanced Threat Protection is a cloud-based security solution that identifies weaknesses and threats.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which of these is the strongest way to protect sensitive customer data?

- Encrypt data as it sits in your database
- Encrypt data as it travels over the network
- Encrypt data both as it sits in your database and as it travels over the network

Question 2

You want to store certificates in Azure to centrally manage them for your services. Which Azure service should you use?

- Azure Key Vault
- Azure AD

Securing Storage Accounts and Data Lake Storage

Securing Storage Accounts and Data Lake Storage

Azure Storage accounts provide a wealth of security options that protect your cloud-based data. Because many services such as Blob storage, File shares, Table storage, and Azure Data Lake Storage all build on top of Azure Storage, they all benefit from the fine-grained security controls provided.

Imagine that the network administrator at Contoso is performing a security audit of the assets within the domain. When it comes to cloud-based data, it's essential that she's satisfied that all the data stored in Azure follows the corporate policy of defense in depth. As their primary data consultant, your job is to take her through the available options in Azure Storage to ensure the security requirements meet the needs of the organization.

Learning Objectives

In this module you will:

- Investigate the Azure Storage account security features
- Explore the authentication options available to access data
 - Storage Account Keys
 - Shared Access Signatures
- Control network access to the data
- Managing encryption
- Advanced Threat Detection
- Explore the Azure Data Lake enterprise-class security features

Storage Account Security Features

Contoso relies heavily on massive amounts of data stored in Azure Storage. In their many applications, they rely on blobs, unstructured table storage, Azure Data Lake, and SMB-based file shares. One of their competitors had a data breach, and the network admin was tasked with ensuring the same thing won't happen to them. As their consulting, you've assured them that Azure Storage accounts provide several high-level security benefits that ensure the data secured in the cloud.

- Protecting the data at "rest"
- Protecting the data in "transit"
- Controlling who can access data
- Auditing requests

Encryption at rest

First, all data written to Azure Storage is automatically encrypted using Storage Service Encryption (SSE) with a 256-bit AES cipher. SSE automatically encrypts your data when writing it to Azure Storage. When you read data from Azure Storage, it is decrypted by Azure Storage before being returned. There are no additional charges, nor any performance degradation and the feature cannot be disabled.

For VMs, Azure provides the ability to encrypt virtual hard disks (VHDs) with Azure Disk Encryption. This uses BitLocker for Windows images, and DM-Crypt for Linux. The keys are stored in an Azure Key Vault automatically to help you control and manage the disk-encryption keys and secrets. This ensures that even if someone obtains access to the VHD image and downloads it, they still won't be able to access the data contained in it.

Encryption in transit

You can ensure your data remains secured by enabling *transport-level security* between Azure and the client. The first recommendation is to always use **HTTPS** to ensure secure communication over the public Internet. You can enforce the use of HTTPS when calling the REST APIs to access objects in storage accounts by enabling **Secure transfer required**³ for the storage account. Connections using HTTP will be refused once this is enabled. This flag will also enforce secure transfer over SMB by enforcing SMB 3.0 be used for all file share mounts.

Role-based access control

To access data in a storage account, the client makes a request over HTTP/HTTPS. Every request to a secure resource must be authorized, so that the service ensures that the client has the permissions required to access the data. There are several options available, the first one (and arguably the most flexible) is *role-based access*.

Azure Storage supports Azure Active Directory (Azure AD) and Role-Based Access Control (RBAC) for both resource management and data operations. You can assign RBAC roles scoped to the storage account to security principals and use Azure AD to authorize *resource management operations* such as configuration. In addition, Azure AD is supported for *data operations* on Blob and Queue services.

You can assign RBAC roles scoped to a subscription, resource group, storage account, or an individual container or queue to a security principal or a managed identity for Azure resources.

Auditing access

Audit is the other half of controlling access. Azure Storage access can be audited using a built-in service: Storage Analytics. Every operation is logged in real-time and you can search the storage analytics logs for specific requests. You can filter based on the authentication mechanism used, whether the operation was successful, or by the resource being accessed.

Storage Account Keys

Much of the data Contoso works with is generated or consumed by custom applications written in a variety of languages. As mentioned, Azure Storage accounts can create authorized apps in Active Directory to control access to the data in blobs and queues. This authentication approach is the best solution if the app is using Blob or Queue storage.

³ <https://docs.microsoft.com/azure/storage/storage-require-secure-transfer>

For other storage models, clients can use a different approach: a *shared key* or shared secret. This authentication option supports blobs, files, queues, and tables. It's one of the more straightforward approaches to use: the client embeds the shared key in the HTTP `Authorization` header of every request, and the Storage account validates it.

As an example, an application could issue a `GET` request against a blob resource:

```
GET http://myaccount.blob.core.windows.net/?restype=service&comp=stats
```

with the following HTTP headers that control the version of the REST API, the date, and the encoded shared key.

```
x-ms-version: 2018-03-28  
Date: Wed, 23 Oct 2018 21:00:44 GMT  
Authorization: SharedKey myaccount:CY1OP3O3jGFpYFbTCBimLn0Xov0vt0khH/  
E5Gy0fXvg=
```

Account Keys

Shared keys in Azure Storage accounts are called "Storage Account Keys". Azure creates two of these keys (primary and secondary) for each storage account you create and they give access to *everything* in the account. You can view the created storage keys in the Azure portal view of the storage account under **Settings > Access keys** as shown below.

The screenshot shows the Microsoft Azure portal interface. The left sidebar contains various navigation options like 'Create a resource', 'Home', 'Dashboard', etc. The main content area is titled 'Storage accounts > cs75270e7ced3cex4172xb09 - Access keys'. A sidebar on the left lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Storage Explorer (preview)', 'Settings', 'Access keys' (which is selected and highlighted in blue), 'Geo-replication', 'CORS', 'Configuration', 'Encryption', 'Shared access signature', 'Firewalls and virtual networks', 'Advanced Threat Protection ...', and 'Properties'. The 'Access keys' section displays two sets of keys: 'key1' and 'key2'. Each key has a 'Key' field containing a long hex-encoded string and a 'Connection string' field containing a URL. There are download icons next to each key and connection string.

Protecting shared keys

Because there are only two keys, and they provide full access to the account, it's recommended to only use these keys with *trusted in-house applications* that you have complete control over. If the keys are

compromised, you can change the key values in the Azure portal. There are several other reasons to regenerate your storage account keys.

- You might regenerate them on a regular basis for security reasons.
- You must regenerate your storage account keys if someone managed to hack into an application and retrieve the key that was hardcoded or saved in a configuration file, giving them full access to your storage account.
- Another case for key regeneration is if your team is using a Storage Explorer application that retains the storage account key, and one of the team members leaves. The application would continue to work, giving them access to your storage account after they're gone.

The process to refresh keys is simple:

1. Change each trusted app to use the *secondary* key.
2. Refresh the primary key in the Azure portal. You can consider it the new "secondary" key value.

IMPORTANT

Any client attempting to use the old key value will be refused. You must make sure to identify all clients using the shared key and update them to keep them operational.

Shared Access Signatures

The best practice is to not share storage account keys to external third-party applications, another approach must be used.

For untrusted clients, you can use a **Shared Access Signature** (SAS) if you require these applications to have access to your data. A Shared Access Signature is a string containing a security token that can be attached to a URI that allows you to delegate access to storage objects and specify constraints such as the permissions and the date/time range of access.

As an example, you could give one customer a SAS token they can use to upload pictures to a filesystem in Blob Storage and provide a web application permission to read those pictures. In both cases, there is a separation of duty concerns – each application can be granted just the access that they require to perform their task. This is possible using Shared Access Signatures.

Types of Shared Access Signatures

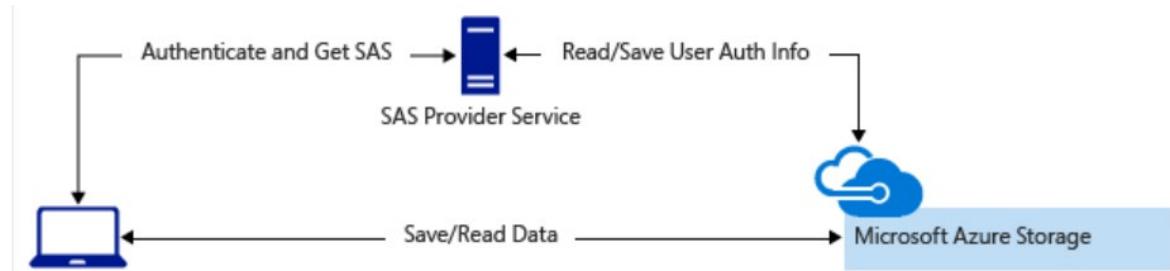
- A service-level SAS can be used to access specific resources in a storage account. Some examples of this are retrieving a list of files in a filesystem or downloading a file.
- An account-level SAS can be used to access anything that a service-level SAS can be used for. Additionally, it can give options to resources that are not permitted with a service-level SAS, such as the ability to create filesystems.

A typical scenario where a SAS is useful is a service where users read and write their data to your storage account. In a scenario where a storage account stores user data, there are two typical design patterns:

1. Clients upload and download data via a front-end proxy service, which performs authentication. This front-end proxy service has the advantage of allowing validation of business rules, but for large amounts of data or high-volume transactions, creating a service that can scale to match demand may be expensive or complicated.



2. A lightweight service authenticates the client as needed and then generates a SAS. Once the client receives the SAS, they can access storage account resources directly with the permissions defined by the SAS and for the interval allowed by the SAS. The SAS mitigates the need for routing all data through the front-end proxy service.



Default Network Access Rules

By default, storage accounts accept connections from clients on any network. To limit access to selected networks, you must first change the default action. You can restrict access to specific IP addresses, ranges, or virtual networks.

IMPORTANT

Making changes to network rules can impact your applications' ability to connect to Azure Storage. Setting the default network rule to **deny** blocks all access to the data unless specific network rules to **grant** access are also applied. Be sure to grant access to any allowed networks using network rules before you change the default rule to deny access.

Managing default network access rules

You can manage default network access rules for storage accounts through the Azure portal, PowerShell, or Azure CLI.

Azure portal

To perform this action in the Azure portal.

1. Navigate to the storage account you want to secure.
2. Select **Firewalls and virtual networks**.
3. Choose to allow access from **Selected networks** to restrict traffic. To allow traffic from all networks, choose to allow access from **All networks**.
4. Click **Save** to apply your changes.

Managing Encryption

Databases stores information that is sensitive, such as physical addresses, email addresses, and phone numbers. If exposed, this could be used by malicious attackers to harm our business or our customers. Let's look at how we can use encryption and data masking to enhance the security of our database.

TLS network encryption

Azure SQL Database and Data Warehouse enforces Transport Layer Security (TLS) encryption at all times for all connections, which ensures all data is encrypted "in transit" between the database and the client. By using TLS encryption, you can ensure that anyone who may have intercepted the traffic between the app server and database would not be able to read the data. TLS encryption is a standard of securing traffic over the internet, and in this case ensures your network traffic to and from your database is secure by default.

Transparent data encryption

Both Azure Data Warehouse and SQL Database protects your data at rest using transparent data encryption (TDE). TDE performs real-time encryption and decryption of the database, associated backups, and transaction log files at rest without requiring changes to the application. Using a database encryption key, transparent data encryption performs real-time I/O encryption and decryption of the data at the page level. Each page is decrypted when it's read into memory and then encrypted before being written to disk.

By default, TDE is enabled for all newly deployed Azure SQL databases. It's important to check that data encryption hasn't been turned off, and older Azure SQL Server databases may not have TDE enabled.

As an example, you can take a look in the portal at where TDE is configured for a database.

1. Sign into the **Azure portal**⁴ using the same account you activated the sandbox with.
2. In the search bar at the top of the portal, search for **marketplaceDb**, then select the database in the portal.
3. In the left menu, in the **Security** section, select the **Transparent data encryption** option.
4. In the data encryption option, verify that **Data encryption** is set to **On**. You should also see an encryption status of **Encrypted**.

Since new databases are encrypted by default, we can be sure that our data is encrypted on disk from as soon as we create the database.

NOTE

Azure includes a built in service called Azure Security Center that gives you visibility into the security of your environment, including Azure SQL databases. Azure Security Center will flag any databases that don't have TDE enabled on them, giving you the ability to report and take action to secure your data.

Encrypt data in transit in your application

Data in transit is a method to prevent man-in-the-middle attacks. To encrypt data in transit, specify **Encrypt=true** in the connection string in your client applications as follows. This ensures that all data sent between your client application and SQL data warehouse is encrypted with SSL.

```
String connectionURL =  
    "jdbc:sqlserver://<your_azure_sql_datawarehouse_fqdn>:1433;" +  
    "databaseName=DemoDW;username=your_username;password=your_password " +  
    "encrypt=true ";
```

Advanced Thread Protection

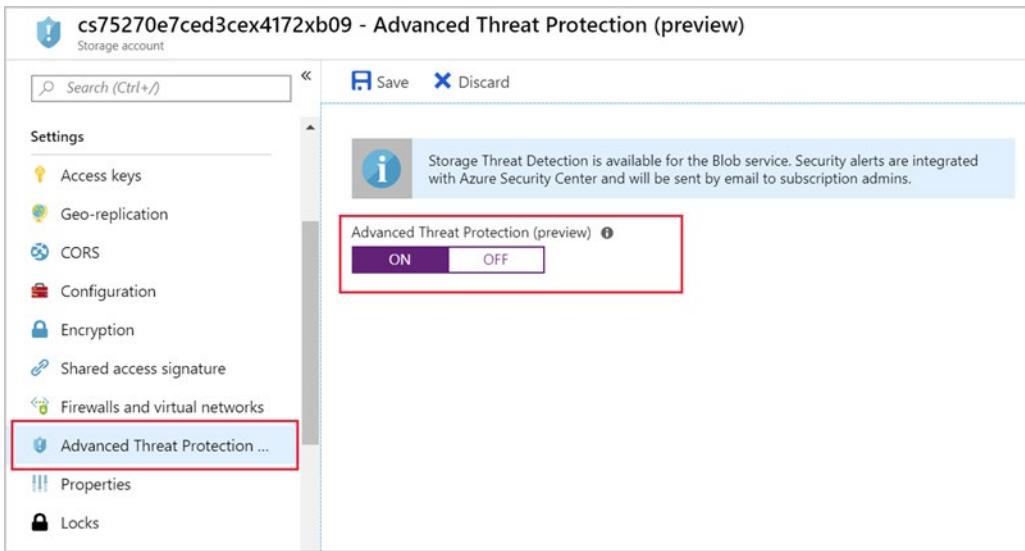
Detecting threats to your data is an important part of security. As mentioned earlier, you can check an audit trail for all activity against a storage account. However, that's often only going to show you when an intrusion *has already occurred*. What we really want is a way to be notified when suspicious activity is actively occurring. Enter **Azure Storage Advanced Thread Protection**.

Azure Storage Advanced Threat Protection (preview) detects anomalies in account activity and notifies you of potentially harmful attempts to access your account. This layer of protection allows you to address threats without the need to be a security expert or manage security monitoring systems. Today, Storage Threat Detection is available for the Blob service. Security alerts are integrated with Azure Security Center and are sent by email to subscription admins.

You can turn on threat protection in the Azure portal through the configuration page of the Azure Storage account.

1. In the **Settings** page, select **Advanced Threat Protection**.
2. In the **Advanced Threat Protection** configuration blade
 - Turn **ON** Advanced Threat Protection
 - Click **Save** to save the new or updated Advanced Threat Protection policy.

⁴ <https://portal.azure.com/learn.docs.microsoft.com?azure-portal=true>



Azure Data Lake Storage Gen2 Security Features

Azure Data Lake Storage Gen2 provides a first-class data lake solution for enterprises to pull together their data. It's built on top of Azure Blob storage and inherits all the security features we've seen in this module. Along with Role Based Access Control (RBAC), Azure Data Lake Storage Gen2 provides **POSIX-compliant Access Control Lists (ACLs)** that restrict access to only authorized users, groups, or service principals in a flexible, fine-grained, and manageable manner. Authentication is via Azure Active Directory OAuth 2.0 bearer tokens, which allows for flexible authentication schemes including federation with **AAD Connect** and multi-factor authentication for stronger protection than just passwords.

More significantly, these authentication schemes are integrated into the main analytics services that utilize the data including Azure Databricks, HDInsight, and SQL Data Warehouse as well as management tools such as Azure Storage Explorer. Once authenticated, permissions are applied at the finest granularity to ensure the right level of authorization for protecting an enterprise's big data assets.

The Azure Storage end-to-end encryption of data and transport layer protections complete the security shield for an enterprise data lake. The same set of analytics engines and tools can take advantage of these additional layers of protection, resulting in a complete end to end protection of your analytics pipelines.

Summary

Azure Storage provides a layered security model. This model enables you to secure your storage accounts to a specific set of supported networks. When network rules are configured, only applications requesting data from over the specified set of networks can access a storage account. Authorization is supported with Azure Active Directory (AD) credentials (for blobs and queues) (preview), a valid account access key, or a SAS token. Encryption of the data is enabled by default, and you can perform proactive monitoring with Advanced Threat Detection.

Review Questions

Question 1

Mike is working as a consultant developing an application for a national Realtor company. They store thousands of images of houses in an Azure BLOB storage account. The web application Mike is developing needs to have access these images. How can Mike provide secure access for the third-party web application?

- Use Anonymous access to give the web application access
- Use a storage account key to give the web application access
- Use a Shared Access Signature to give the web application access.

Question 2

Mike wants to gain insights should any unusual activity be occurring with his storage account with minimal configuration. What can Mike use to achieve this?

- Encryption
- Storage account signature
- Automatic Threat Detection

Securing Data Stores

Securing Data Stores

Data security is critical when protecting your customer's privacy and your organization's reputation. Businesses have closed due to financial damage or ruined reputations. Customers have had their data accessed unlawfully because of security breaches exposing their personal details, and again, possibly causing financial harm.

Suppose you work at an online retailer. To support the online marketplace your company provides, you store an assortment of data from your customers, such as phone numbers, addresses, and credit cards. You also store data that is critical to the health of your business, such as financial account balances and intellectual property. If exposed or maliciously accessed it could jeopardize the health of your business and the trust your customers place in you. It's your responsibility to make sure this data stored in your databases is as secure as possible, to protect both your customer data and your business data.

Put yourself in the shoes of an attacker. If you were trying to maliciously attack a system, would a single layer of protection or multiple layers of protection make it more difficult to gain access to the data? Defense in depth is a strategy that employs a layered approach to slow the advance of an attack aimed at acquiring unauthorized access to information. Each layer provides protection so that if one layer is breached, a subsequent layer is already in place to prevent further exposure.

The Data stores available on Azure have a number of built-in capabilities you can use to ensure your data is secure and practice defense in depth. Azure Data warehouse is no different, with security hardening the data warehouse and its related assets to protect privacy, integrity, and availability of the warehouse.

In this section, you will look at ways to secure databases by configuring the database firewall, securing access, encrypting communication, and other techniques for database security. With this layered approach, you can help ensure your data is secure.

Learning objectives

In this module, you will:

- Control network access to your data stores using firewall rules
- Control user access to your data stores using authentication and authorization
- Protect your data in transit and at rest
- Audit and monitor your Azure SQL Database for access violations

Restricting Network Access to Data Stores

Users will connect to our app server to enter orders, update their account, and perform similar activities, which will in turn update the database with these changes. Because personal data is stored in the database it's critical to ensure that we only allow access from trusted and necessary resources. Let's take a look at a number of ways you can control access to your Azure SQL Database or Data Warehouse over the network.

Firewall rules

Data Stores such as Azure SQL Database and Data Warehouse has a built-in firewall that is used to allow and deny network access to both the database server itself, as well as individual databases. By default,

Azure Cosmos DB is accessible from internet, with a valid authorization token. To configure IP policy-based access control, the user must provide the set of IP addresses or IP address ranges

Firewall rules are configured at the server and/or database level, and will specifically state which network resources are allowed to establish a connection to the database. Depending on the level, the rules you can apply will be as follows:

- **Server-level firewall rules**

- Allow access to Azure services
- IP address rules
- Virtual network rules

- **Database-level firewall rules**

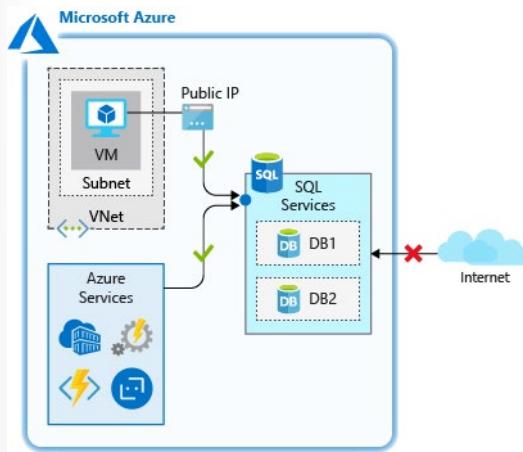
- IP address rules

Let's take a closer look at how these rules work.

Server-level firewall rules

These rules enable clients to access your entire Azure SQL server, that is, all the databases within the same logical server. There are three types of rules that can be applied at the server level.

The **Allow access to Azure services** rule allows services within Azure to connect to your Azure SQL Database or Data Warehouse. When enabled, this setting allows communications from all Azure public IP addresses. This includes all Azure Platform as a Service (PaaS) services, such as Azure App Service and Azure Container Service, as well as Azure VMs that have outbound internet access. This rule can be configured through the **ON/OFF** option in the firewall pane in the portal, or by an IP rule that has 0.0.0.0 as the start and end IP addresses.

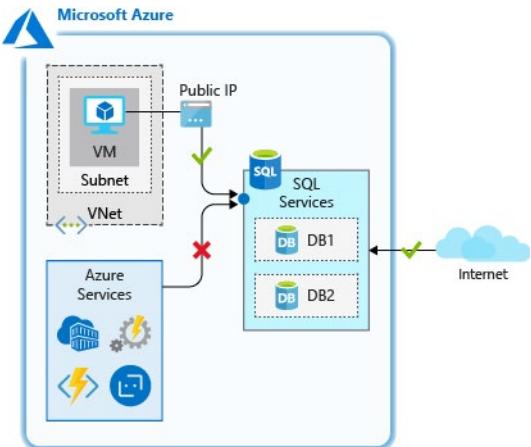


This rule is used when you have applications running on PaaS services in Azure, such as Azure Logic Apps or Azure Functions, that need to access. Many of these services don't have a static IP address, so this rule is needed to ensure they are able to connect to the database.

IMPORTANT

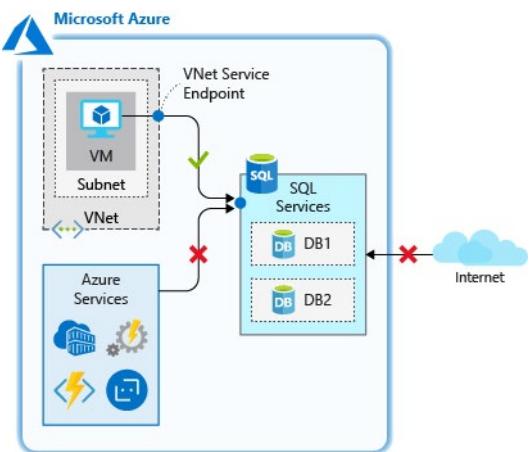
This option configures the firewall to allow all connections from Azure including connections from the subscriptions of other customers. When selecting this option, make sure your login and user permissions limit access to only authorized users.

IP address rules are rules that are based on specific public IP address ranges. IP addresses connecting from an allowed public IP range will be permitted to connect to the database.



These rules can be used when you have a static public IP address that needs to access your database.

Virtual network rules allow you to explicitly allow connection from specified subnets inside one or more Azure virtual networks (VNets). Virtual network rules can provide greater access control to your databases and can be a preferred option depending on your scenario. Since Azure VNet address spaces are private, you can effectively eliminate exposure to public IP addresses and secure connectivity to those addresses you control.

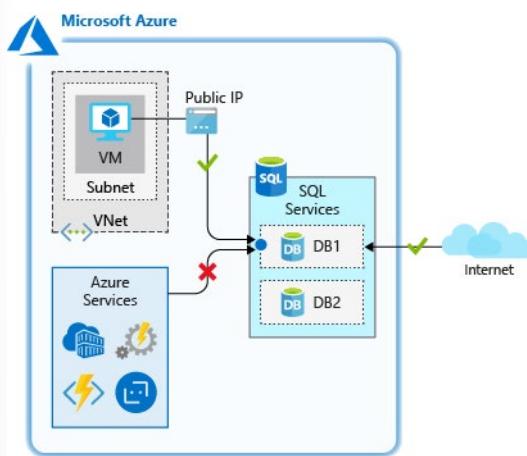


Virtual network rules are used when you have Azure VMs that need to access your database.

For server-level rules, all of the above can be created and manipulated through the portal, PowerShell, the CLI and through Transact-SQL (T-SQL).

Database-level firewall rules

These rules allow access to an individual database on a logical server and are stored in the database itself. For database-level rules, only **IP address rules** can be configured. They function the same as when applied at the server-level, but are scoped to the database only.



The benefits of database-level rules are their portability. When replicating a database to another server, the database-level rules will be replicated, since they are stored in the database itself.

The downside to database-level rules is that you can only use IP address rules. This may limit the flexibility you have and can increase administrative overhead.

Lastly, database-level firewall rules can be created and manipulated only through T-SQL.

Restricting network access in practice

Let's take a look at how these work in practice, and how you can secure network access to only allow what is necessary. Recall that we created an Azure SQL Database logical server, a database, and the *appServer* Linux VM acting as an application server. This scenario is often seen when a database has been migrated to Azure SQL Database and resources inside of a virtual network need to access it. The firewall feature of Azure SQL Database can be used in many scenarios, but this is an example that has practical applicability and demonstrates how each of the rules functions.

Let's go through the firewall settings and see how they work. We'll use both the cloud shell and the portal for these exercises.

The database we created currently does not allow access from any connections. This is by design based on the commands that we ran to create the logical server and database. Let's confirm this.

1. In the cloud shell, SSH into your Linux VM if you aren't already connected.

```
ssh <X.X.X.X>
```

1. Recall the `sqlcmd` command we retrieved earlier. Go ahead and run it to attempt to connect to the database. Make sure you replace `<username>` and `<password>` with the `ADMINUSER` credentials you specified in the previous unit. Make sure to keep the single quotes around the username and password so that any special characters aren't misinterpreted by the shell.

```
sqlcmd -S tcp:server<12345>.database.windows.net,1433 -d marketplaceDb -U '<username>' -P '<password>' -N -l 30
```

You should receive an error when trying to connect. This is expected since we've not allowed any access to the database.

```
Sqlcmd: Error: Microsoft ODBC Driver 17 for SQL Server : Cannot open server 'securedb' requested by the login. Client with IP address '40.112.128.214'
```

is not allowed to access the server. To enable access, use the Windows Azure Management Portal or run `sp_set_firewall_rule` on the master database to create a firewall rule for this IP address or address range. It may take up to five minutes for this change to take effect..

Let's grant access so we can connect.

Use the server-level allow access to Azure services rule

Since our VM has outbound internet access, we can use the **Allow access to Azure services** rule to allow access from our VM.

1. Sign into the **Azure portal**⁵ using the same account you activated the sandbox with.
2. In the **Search resources, services, and docs** box at the top, search for your database server name, `server<12345>`. Select the SQL server.
3. In the SQL server panel, in the **Security** section in the left menu, select **Firewalls and virtual networks**.
4. Set **Allow access to Azure services** to **ON** and click **Save**.
5. Back in your SSH session, let's try to connect to your database again.

```
sqlcmd -S tcp:server<12345>.database.windows.net,1433 -d marketplaceDb -U
'<username>' -P '<password>' -N -l 30
```

At this point, you should be able to connect. If it's successful, you should see a `sqlcmd` prompt.

```
1>
```

So we've opened up connectivity, but this setting currently allows access from *any* Azure resource, including resources outside of our subscription. Let's restrict this further to limit network access to only resources that are within our control.

Use a database-level IP address rule

Recall that database-level IP address rules allow only access to an individual database on a logical server. We'll use one here to grant access to the static IP of our `appServer` VM.

To create a database-level IP rule, we'll need to run some T-SQL commands. You'll create a database rule using the following convention, where you pass in the rule name, the starting IP address, and the ending IP address. By specifying the start and end IP to be the same, we're limiting access to a single IP, though we could expand the range if we had a larger block of addresses that required access.

```
EXECUTE sp_set_database_firewall_rule N'My Firewall Rule', '40.112.128.214',
'40.112.128.214'
```

1. While still at the `sqlcmd` prompt, run the following command, replacing the public IP address of your `appServer` VM in both locations below.

TIP

When running T-SQL commands such as the following, the `GO` on the second line may not copy through

⁵ <https://portal.azure.com/learn.docs.microsoft.com?azure-portal=true>

to the `sqlcmd` prompt, so you will likely need to type this out. The T-SQL command won't execute without it, so make sure to run the `GO` command.

```
EXECUTE sp_set_database_firewall_rule N'Allow appServer database level
rule', '<From IP Address>', '<To IP Address>';
GO
```

Once the command completes, type `exit` to exit `sqlcmd`. Remain connected via SSH.

1. In the portal, on the **Firewalls and virtual networks** panel for your SQL server, set **Allow access to Azure services** to **OFF** and click **Save**. This will disable access from all Azure services, but we'll still be able to connect since we have a database-level IP rule for our server.
2. Back in cloud shell, in the VM you are connected via SSH to, try connecting to your database again.

```
sqlcmd -S tcp:server<12345>.database.windows.net,1433 -d marketplaceDb -U
'<username>' -P '<password>' -N -l 30
```

At this point, you should be able to connect. If it's successful, you should see a `sqlcmd` prompt.

```
1>
```

Using a database-level rule allows access to be isolated specifically to the database. This can be useful if you'd like to keep your network access configured per database. If multiple databases share the same level of network access, you can simplify administration by using a server-level rule to apply the same access to all databases on the server.

Use a server-level IP address rule

Database-level rules are a great option, but what if we had multiple databases on the same server that our *appServer* VM needed to connect to? We could add a database-level rule to each database, this can take more work as we add more databases. It would reduce our administration efforts to allow access with a server-level rule, which would apply to all databases on the server.

Let's now use a server-level IP rule to restrict the systems that can connect.

1. While still at the `sqlcmd` prompt, run the following command to delete the database-level IP address rule.

```
EXECUTE sp_delete_database_firewall_rule N'Allow appServer database level
rule';
GO
```

Once the command completes, type `exit` to exit `sqlcmd`. Remain connected via SSH.

1. Back in the portal, on the **Firewalls and virtual networks** panel for your SQL server, add a new rule with a **RULE NAME** of **Allow appServer** and with the **START IP** and **END IP** set to the public IP address of the *appServer* VM.

Click **Save**

1. Back in cloud shell, on your *appServer* VM, try connecting to your database again.

```
sqlcmd -S tcp:server<12345>.database.windows.net,1433 -d marketplaceDb -U
'<username>' -P '<password>' -N -l 30
```

At this point you should be able to connect, since the server-level rule is allowing access based on the public IP address of the *appServer* VM. If it's successful, you should see a sqlcmd prompt.

1>

Type `exit` to exit sqlcmd. Remain connected via SSH.

So we've isolated connectivity to only the IP address we specified in the rule. This works great, but can still be an administrative challenge as you add more systems that need to connect. It also requires a static IP or an IP from a defined IP address range; if the IP is dynamic and changes, we'd have to update the rule to ensure connectivity. The *appServer* VM is currently configured with a dynamic IP address, so this IP address is likely to change at some point, breaking our access as soon as that happens. Let's now look at how virtual network rules can be beneficial in our configuration.

Use a server-level virtual network rule

In this case, since our VM is running in Azure, we can use a server-level virtual network rule to isolate access and make it easy to enable future services to gain access to the database.

1. Back in the portal and still on the **Firewalls and virtual networks** panel, in the **Virtual networks** section click the **+ Add existing virtual network** option.
2. The Create/Update virtual network rule dialog will show. Set the following values:

| Setting | Value |
|-------------------------------------|-------------------------------|
| Name | Leave the default value |
| Subscription | Concierge Subscription |
| Virtual network | appServerVNET |
| Subnet name / Address prefix | appServerSubnet / 10.0.0.0/24 |

Click **Enable** to enable the service endpoint on the subnet, then **OK** once the endpoint is enabled to create the rule.

3. Now, let's remove the IP address rule. Click the ... next to your **Allow appServer** rule and click **Delete**, then click **Save**.
4. Back in cloud shell, on your *appServer* VM, try connecting to your database again.

```
sqlcmd -S tcp:server<12345>.database.windows.net,1433 -d marketplaceDb -U
'<username>' -P '<password>' -N -l 30
```

At this point, you should be able to connect. If it's successful, you should see a sqlcmd prompt.

1>

What we've done here effectively removes any public access to the SQL server, and only permits access from the specific subnet in the Azure VNet we defined. If we were to add additional app servers in that subnet, no additional configuration would be necessary, as any server in that subnet would have the ability to connect to the SQL server. This limits our exposure to services outside of our scope of control, and eases administration if we were to add additional servers. This is an effective method of securing network access to an Azure SQL Database.

Managing Authentication and Authorization

Even though we may be able to connect to the database over the network, that doesn't mean we can actually gain access to the data itself. Following a layered approach, we'll want to ensure that only users who need access to the data can actually access it. This is where authentication and authorization come in to play.

Authentication

Authentication is the process of verifying an identity. This identity could be a user, a service running on a system, or a system itself (such as a virtual machine). Through the process of authentication, we ensure that the person or system is who they claim to be. SQL Database and Data Warehouse supports two types of authentication: SQL authentication and Azure Active Directory authentication.

SQL authentication

SQL authentication method uses a username and password. User accounts can be created in the master database and can be granted permissions in all databases on the server, or they can be created in the database itself (called contained users) and given access to only that database. When you created the logical server for your database, you specified a "server admin" login with a username and password. Using these credentials, you can authenticate to any database on that server as the database owner, or "dbo".

Azure Active Directory authentication

This authentication method uses identities managed by Azure Active Directory (AD) and is supported for managed and integrated domains. Use Azure AD authentication (integrated security) whenever possible. With Azure AD authentication, you can centrally manage the identities of database users and other Microsoft services in one central location. Central ID management provides a single place to manage database users and simplifies permission management. If you want to use Azure AD authentication, you must create another server admin called the "Azure AD admin," which is allowed to administer Azure AD users and groups. This admin can also perform all operations that a regular server admin can.

Authorization

Authorization refers to what an identity can do within a database. This is controlled by permissions granted directly to the user account and/or database role memberships. A database role is used to group permissions together to ease administration, and a user is added to a role to be granted the permissions the role has. These permissions can grant things such as the ability to log in to the database, the ability to read a table, and the ability to add and remove columns from a database. As a best practice, you should grant users the least privileges necessary. The process of granting authorization to both SQL and Azure AD users is the same.

In our example here, the server admin account you are connecting with is a member of the db_owner role, which has authority to do anything within the database.

Authentication and authorization in practice

Let's now take a look at how to set up a user and grant them access to a database. In this case we'll use SQL authentication for our user, but the process would be essentially the same if we were using Azure AD authentication.

Create a database user

Let's go ahead and create a new user that we can use to grant access to.

1. In cloud shell, on your *appServer* VM, connect to your database again as your `ADMINUSER`.

```
sqlcmd -S tcp:server<12345>.database.windows.net,1433 -d marketplaceDb -U '<username>' -P '<password>' -N -I 30
```

2. Run the following command to create a new user. This will be a *contained user* and will only allow access to the *marketplace* database. Feel free to adjust the password as necessary, but be sure and note it as we'll need it for a future step.

```
CREATE USER ApplicationUser WITH PASSWORD = 'YourStrongPassword1';  
GO
```

With these credentials, the user will be able to authenticate to the database, but they aren't authorized to access any data. Let's grant this user access.

Grant permissions to a user

Let's make the user a member of the `db_datareader` and `db_datawriter` roles, granting access to read and write to the database, respectively. We also want to prevent this user from accessing a table with addresses.

1. While still connected to `sqlcmd` on *appServer*, run the following T-SQL to grant the `db_datareader` and `db_datawriter` roles to the user we just created.

```
ALTER ROLE db_datareader ADD MEMBER ApplicationUser;  
ALTER ROLE db_datawriter ADD MEMBER ApplicationUser;  
GO
```

2. We can narrow the scope of access further. We could deny a user's access to other elements within the database using the `DENY` operator. Run the following T-SQL to deny the user *ApplicationUser* the ability to select data from the *SalesLT.Address* table.

```
DENY SELECT ON SalesLT.Address TO ApplicationUser;  
GO
```

Let's now log in as that user and take a look at this in action.

1. While still at the T-SQL prompt, type `exit` to exit your session.
2. Now let's log back in to the database, but as the user we just created.

```
sqlcmd -S tcp:server<12345>.database.windows.net,1433 -d marketplaceDb -U 'ApplicationUser' -P '<password>' -N -I 30
```

3. Run the following query. This is pulling data from a table that the user is authorized to access.

```
SELECT FirstName, LastName, EmailAddress, Phone FROM SalesLT.Customer;  
GO
```

You should get back a listing of customers.

| FirstName | LastName | EmailAddress | Phone |
|-----------|----------|------------------------------|--------------|
| <hr/> | | | |
| Orlando | Gee | orlando0@adventure-works.com | 245-555-0173 |
| Keith | Harris | keith0@adventure-works.com | 170-555-0127 |
| Donna | Carreras | donna0@adventure-works.com | 279-555-0130 |
| Janet | Gates | janet1@adventure-works.com | 710-555-0173 |
| ... | | | |

- Now let's see what happens when we try to query a table that we don't have access to.

```
SELECT * FROM SalesLT.Address;  
GO
```

You should get a message that you don't have access to this table.

```
Msg 229, Level 14, State 5, Server server-22942, Line 1  
The SELECT permission was denied on the object 'Address', database 'marketplace', schema 'SalesLT'.
```

As you can see here, even though we've granted read/write access to the database, we can further secure access to data by explicitly denying access to tables. If you had multiple users who shared similar access, you could create custom roles with the proper permissions and simplify your administration.

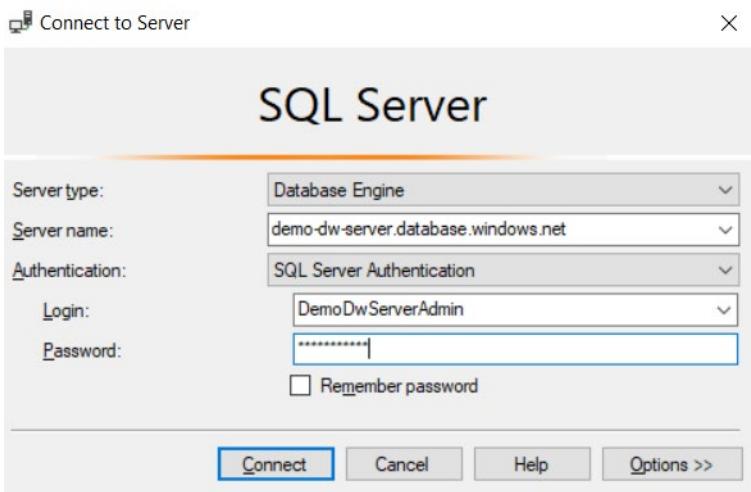
Restrict Azure SQL DW data access with granular security controls using SQL Server Management Studio

In specific scenarios, granular security controls are essential for privacy and compliance reasons. For instance, in an employee data warehouse that hosts data for all employees in a company, you would want to provide access to restrict access to individual columns or rows in data warehouse tables to specific groups of management, depending on their role and function. Azure Synapse Analytics provides fine-grained security controls via column-level security and row-level security. In this section, we will walk through how to enforce these controls on your data stored in Azure Synapse Analytics.

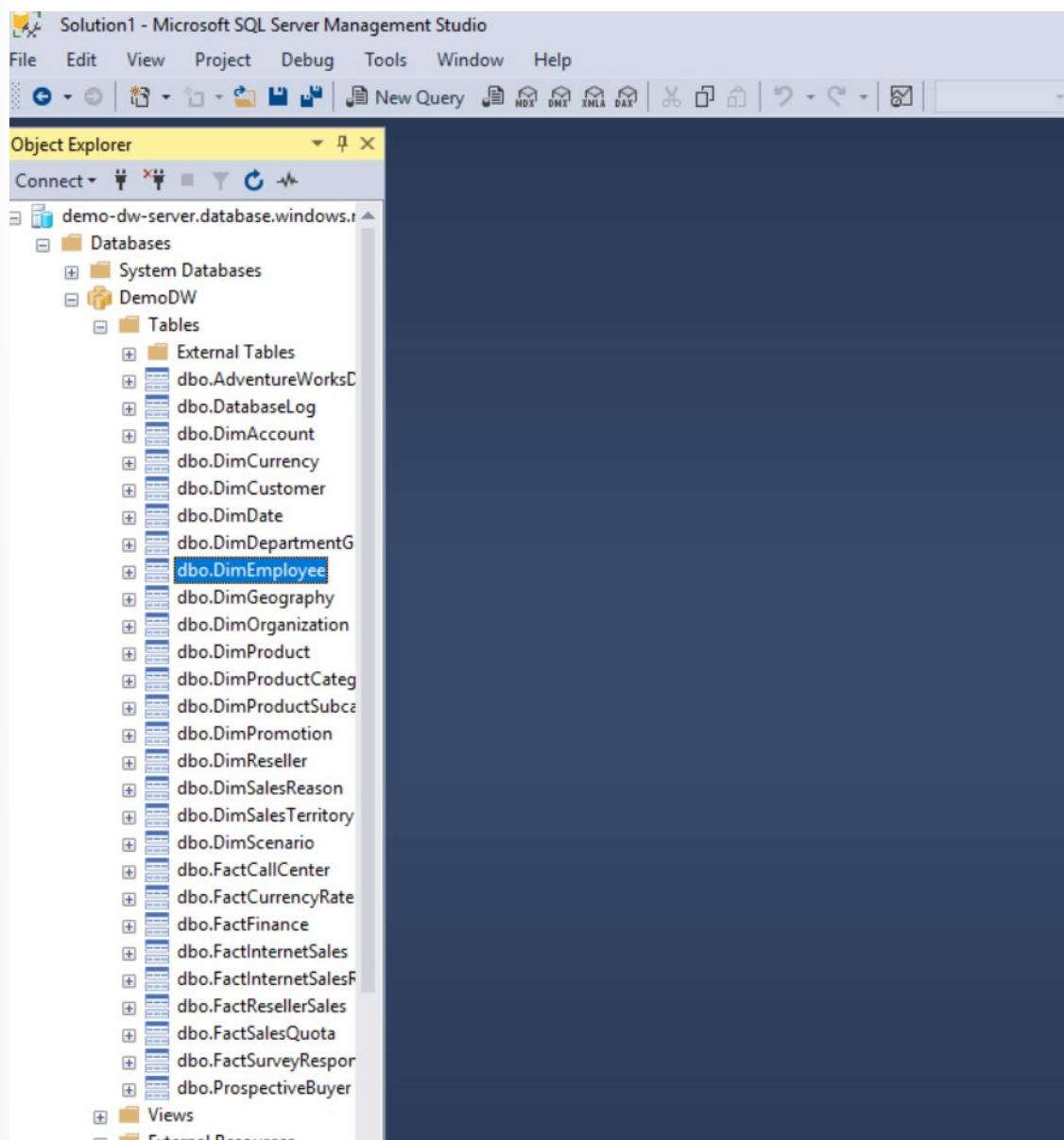
Grant read access to data using column-level security

Use the following steps to grant access to specific users to only certain columns of a table.

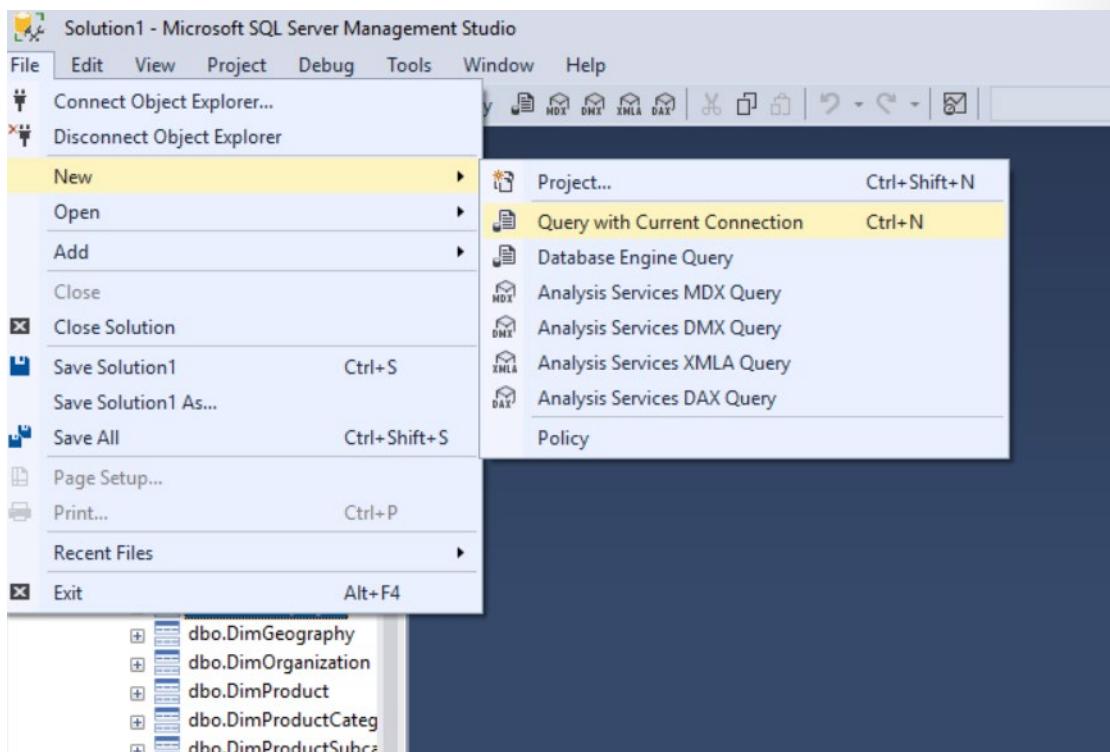
- Open SQL Server Management Studio that you installed as a prerequisite and connect to the data warehouse. Get the server name from Azure portal by navigating to the data warehouse you previously created and use the username and password you specified when you created the data warehouse.



- Select **Connect**, which logs you into the SQL data warehouse
- Navigate to the DemoDW database and expand Tables node. You will see dbo.DimEmployee table, which is the table we will be limiting access to the user we previously created, namely **demodw_user**.

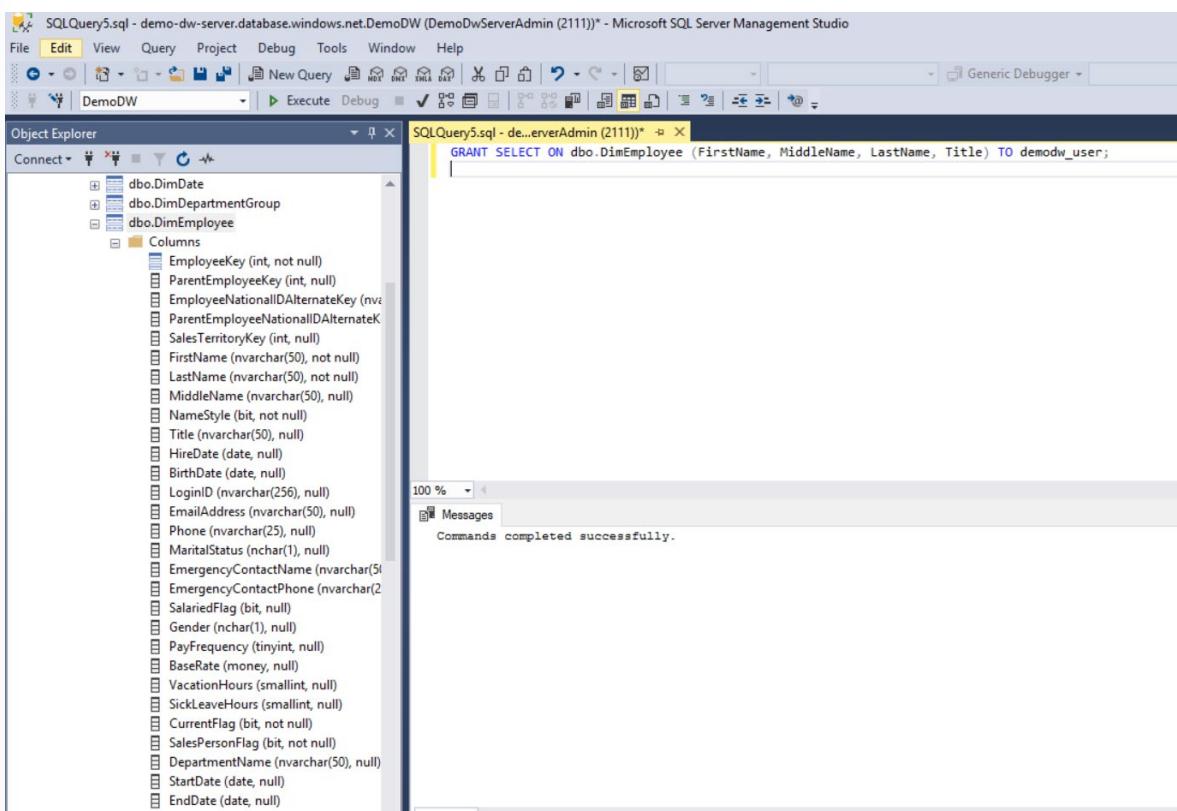


- From **File** menu, select **New** and then **Query with Current Connection** menu option



- In the query pane, type the following command and execute the query by clicking on the Execute button

```
GRANT SELECT ON dbo.DimEmployee (FirstName, MiddleName, LastName, Title) TO  
demodw_user;
```



- The user **demodw_user** can now only access the four columns in the table for which it has been granted access, and no other columns are visible to that user.

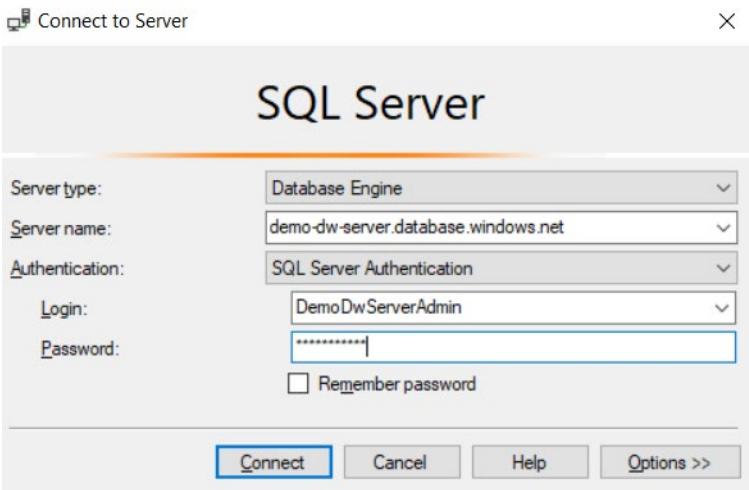
Restrict access to data using row-level security

Azure Synapse Analytics provides predicate-based row-level security. The rows on a table that don't satisfy the predicates are silently excluded from queries. Enforcing row-level security in Azure Synapse Analytics is a two-step process, namely:

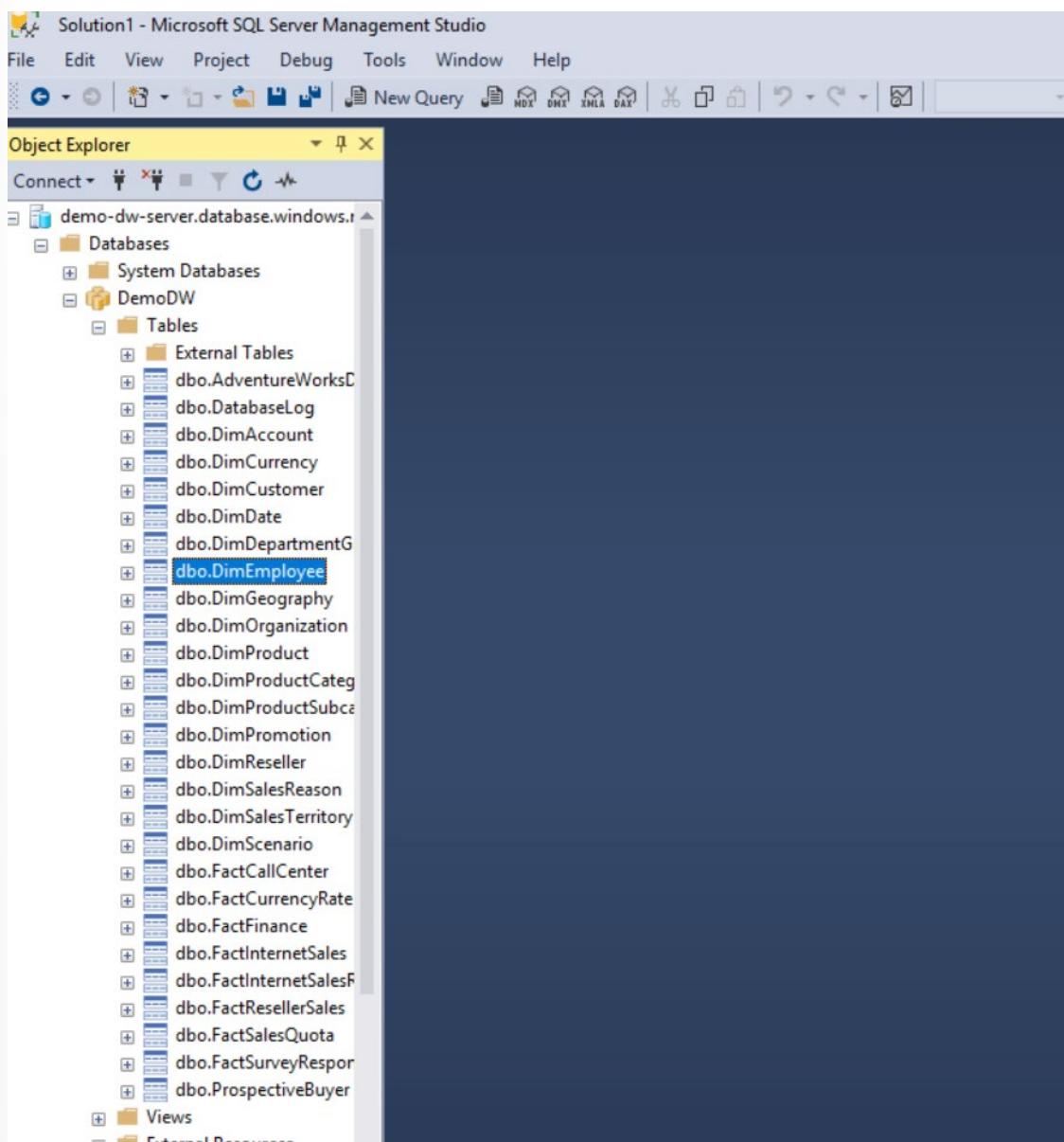
1. creation of a table-valued function to define the access criteria, and
2. the creation of a security policy that adds filter predicates created in the previous step to any tables for which row-level access needs to be restricted

Follow the steps below to enforce row-level security for the table `dbo.DimEmployee` in the `DemoDW` database.

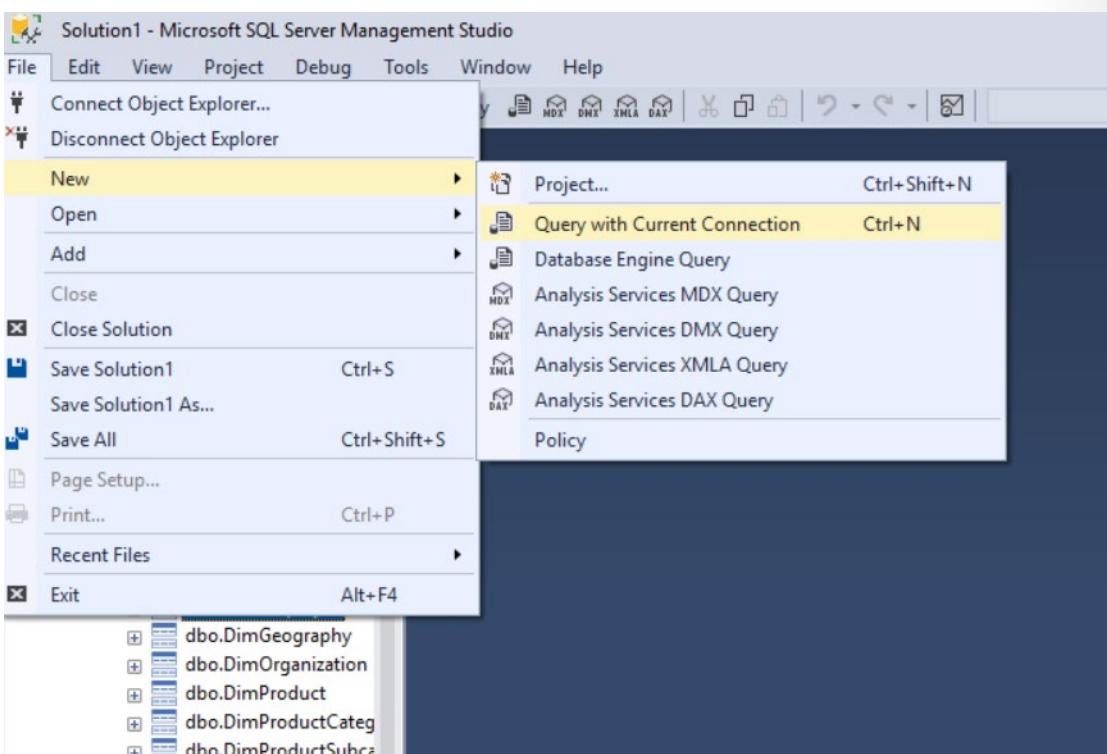
- Open SQL Server Management Studio that you installed as a prerequisite and connect to the data warehouse. Get the server name from Azure portal by navigating to the data warehouse you previously created and use the username and password you specified when you created the data warehouse.



- Select **Connect**, which logs you into the SQL data warehouse
- Navigate to DemoDW database and expand Tables node. You will see `dbo.DimEmployee` table, which is the table we will be limiting access to the user we previously created, namely **demodw_user**.



- From **File** menu, select **New** and then **Query with Current Connection** menu option



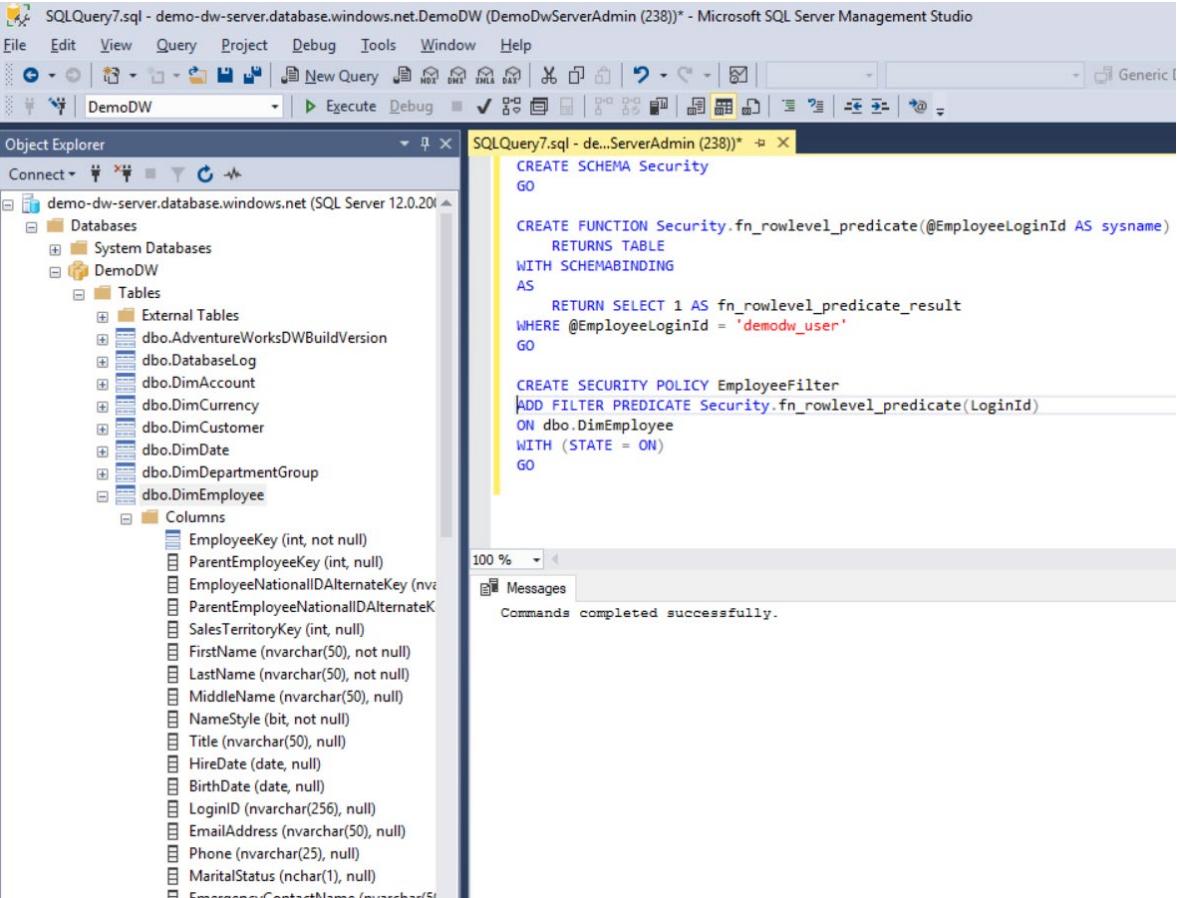
- In the query pane, type the following command and execute the query by clicking on the Execute button

```
CREATE SCHEMA Security
GO
```

```
CREATE FUNCTION Security.fn_rowlevel_predicate(@EmployeeLoginId AS sysname)
RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN SELECT 1 AS fn_rowlevel_predicate_result
WHERE @EmployeeLoginId = 'demodw_user'
GO
```

```
CREATE SECURITY POLICY EmployeeFilter
ADD FILTER PREDICATE Security.fn_rowlevel_predicate(LoginId)
ON dbo.DimEmployee
WITH (STATE = ON)
GO
```

- The rows in table DimEmployee can now only be accessed by **demodw_user**.



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'demo-dw-server.database.windows.net' under 'DemoDW'. The 'Tables' node is expanded, showing various dimensions like DimAccount, DimCurrency, DimCustomer, DimDate, DimDepartmentGroup, DimEmployee, and DimProduct. The 'Columns' node under DimEmployee lists numerous columns such as EmployeeKey, ParentEmployeeKey, EmployeeNationalIDAlternateKey, SalesTerritoryKey, FirstName, LastName, MiddleName, NameStyle, Title, HireDate, BirthDate, LoginID, EmailAddress, Phone, MaritalStatus, and EmergencyContactName. The central pane shows T-SQL code being executed:

```
CREATE SCHEMA Security
GO

CREATE FUNCTION Security.fn_rowlevel_predicate(@EmployeeLoginId AS sysname)
    RETURNS TABLE
    WITH SCHEMABINDING
AS
    RETURN SELECT 1 AS fn_rowlevel_predicate_result
    WHERE @EmployeeLoginId = 'demodw_user'
GO

CREATE SECURITY POLICY EmployeeFilter
ADD FILTER PREDICATE Security.fn_rowlevel_predicate(LoginId)
ON dbo.DimEmployee
WITH (STATE = ON)
GO
```

The 'Messages' tab at the bottom right shows the message "Commands completed successfully."

It's important to properly secure your database, and only grant access where necessary. Azure SQL Database provides the built-in ability to fully control the ability to authenticate and authorize identities to access the data in your database.

Dynamic data masking

Note

This feature is currently only available in Azure SQL Database

When running queries some of the information returned in the database is sensitive; there are phone numbers, email addresses, and other information that we may not want to fully display to everyone with access to the data.

Maybe we don't want our users to be able to see the full phone number or email address, but we'd still like to make a portion of the data available for customer service representatives to identify a customer. By using the dynamic data masking feature of Azure SQL Database, we can limit the data that is displayed to the user. Dynamic data masking is a policy-based security feature that hides the sensitive data in the result set of a query over designated database fields, while the data in the database is not changed.

Data masking rules consist of the column to apply the mask to, and how the data should be masked. You can create your own masking format, or use one of the standard masks such as:

- Default value, which displays the default value for that data type instead.

- Credit card value, which only shows the last four digits of the number, converting all other numbers to lower case x's.
- Email, which hides the domain name and all but the first character of the email account name.
- Number, which specifies a random number between a range of values. For example, on the credit card expiry month and year, you could select random months from 1 to 12 and set the year range from 2018 to 3000.
- Custom string. This allows you to set the number of characters exposed from the start of the data, the number of characters exposed from the end of the data, and the characters to repeat for the remainder of the data.

When querying the columns, database administrators will still see the original values, but non-administrators will see the masked values. You can allow other users to see the non-masked versions by adding them to the SQL users excluded from masking list.

Let's take a look at how this would work in a database.

1. While still in the portal on the database panel, in the **Security** section in the left menu select **Dynamic Data Masking**.
2. The Masking rules screen shows a list of existing dynamic data masks, and recommendations for columns that should potentially have a dynamic data mask applied.

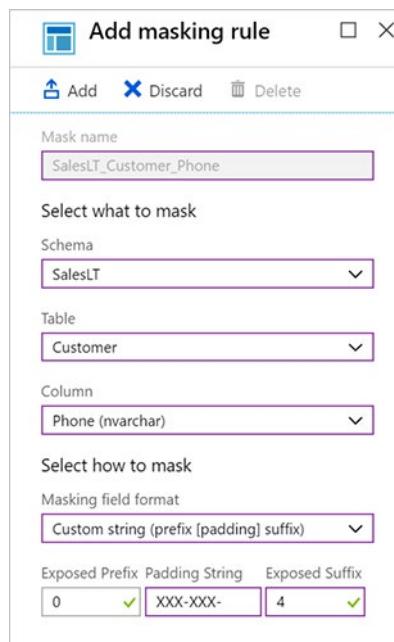
| Masking rules | | | |
|--|---------------|--------------|---|
| MASK NAME | MASK FUNCTION | | |
| You haven't created any masking rules. | | | |
| SQL users excluded from masking (administrators are always excluded) ⓘ | | | |
| SQL users excluded from masking (administrators are always excluded) | | | <input checked="" type="checkbox"/> |
| Recommended fields to mask | | | |
| SCHEMA | TABLE | COLUMN | Add mask |
| SalesLT | Address | AddressID | <input type="button" value="Add mask"/> |
| SalesLT | Address | AddressLine1 | <input type="button" value="Add mask"/> |
| SalesLT | Address | AddressLine2 | <input type="button" value="Add mask"/> |
| SalesLT | Customer | FirstName | <input type="button" value="Add mask"/> |
| SalesLT | Customer | LastName | <input type="button" value="Add mask"/> |
| Load more | | | |

3. Let's add a mask for the phone number that only displays the last four digits. Click the **+ Add mask** button at the top to open the **Add masking rule** dialog.
4. Select the following values.

| Setting | Value |
|-----------------------------|---|
| Schema | SalesLT |
| Table | Customer |
| Column | Phone (nvarchar) |
| Masking field format | Custom string (prefix [padding] suffix) |
| Exposed Prefix | 0 |
| Padding String | XXX-XXX- |

| Setting | Value |
|----------------|-------|
| Exposed Suffix | 4 |

5. Click **Add** to add the masking rule.



6. Let's add one more for the email address. Click the **+ Add mask** button at the top again to open up the **Add masking rule** dialog.

| Setting | Value |
|-----------------------------|-------------------------|
| Schema | SalesLT |
| Table | Customer |
| Column | EmailAddress (nvarchar) |
| Masking field format | Email (aXXX@XXX.com) |

7. Click **Add** to add the masking rule.
8. Each new mask will be added to the masking rules list. Click the **Save** button to apply the masks.

Let's take a look at how this changes our query.

1. Now let's log back in to the database, but as the *ApplicationUser* user.

```
sqlcmd -S tcp:server<12345>.database.windows.net,1433 -d marketplaceDb -U 'ApplicationUser' -P '<password>' -N -l 30
```

2. Run the following query.

```
SELECT FirstName, LastName, EmailAddress, Phone FROM SalesLT.Customer;  
GO
```

Look at how the output has been masked.

| FirstName | LastName | EmailAddress | Phone |
|-----------|----------|----------------|--------------|
| Orlando | Gee | oXXXX@XXXX.com | XXX-XXX-0173 |

| | | | |
|-------|----------|---------------|--------------|
| Keith | Harris | kXXX@XXXX.com | XXX-XXX-0127 |
| Donna | Carreras | dXXX@XXXX.com | XXX-XXX-0130 |
| Janet | Gates | jXXX@XXXX.com | XXX-XXX-0173 |
| ... | | | |

With the masking rules we created, our data is masked with format that we've specified. This allows our customer service reps to verify a customer with the last four digits of their phone number, but hides the full number and the customer's email address from reps view.

Monitoring your Databases

Imagine you receive an alert from your company's security administrator that a potential security breach has been detected on your network. It's suspected that an unauthorized individual may have accessed your database through malicious activity. How would you track this down? You know you need to be actively monitoring your database for suspicious activity, but what can you do to not only gain visibility into what's happening in your database, but to also prevent malicious activity from occurring?

Azure SQL Database has built-in features that can help you track what's happening in your database, and will monitor and alert you if malicious activity is identified.

Azure SQL Database and SQL Data Warehouse Auditing

By enabling auditing, operations that occur on the database are stored for later inspection or to have automated tools analyze them. Auditing is also used for compliance management or understanding how your database is used. Auditing is also required if you wish to use Azure threat detection on your Azure SQL database.

Audit logs are written to Append Blobs in an Azure Blob storage account that you designate. Audit policies can be applied at the server-level or database-level. Once enabled, you can use the Azure portal to view the logs, or send them to Log Analytics or Event Hub for further processing and analysis.

Let's look at the steps you take to set up auditing on your system.

1. Sign into the **Azure portal**⁶ using the same account you activated the sandbox with.
2. In the search bar at the top of the portal, search for **server<12345>**, then select the server in the portal.
3. In the left menu, in the **Security** section, select the **Auditing** option.
4. Auditing is turned off by default. To enable it on your database server, tap the **ON** button.
5. Once the ON button is selected, select the **Storage** checkbox, then click **Storage details** to define the storage account.
6. In the **Storage settings** dialog, you can select an existing storage account or create a new storage account to store your audits. The storage account must be configured to use the same region as your server. In this case, we'll define a new storage account. Click **Storage account**, which will then open up the **Create storage account** dialog. Name the storage account **server<12345>auditing** replacing the <12345> with the number from your logical server name. Leave the rest of the options at their defaults and select **OK**. Back in the **Storage settings** dialog, leave the defaults and click **OK**.
7. Click the **Save** button in the toolbar to save your changes and enable auditing on your database server.

⁶ <https://portal.azure.com/learn.docs.microsoft.com?azure-portal=true>

Now let's generate some audit records and take a look at what you can expect.

1. Let's log back in to the database as the *ApplicationUser* user.

```
sqlcmd -S tcp:server<12345>.database.windows.net,1433 -d marketplaceDb -U 'ApplicationUser' -P '<password>' -N -l 30
```

1. Run the following query.

```
SELECT FirstName, LastName, EmailAddress, Phone FROM SalesLT.Customer;  
GO
```

1. Back in the portal on your SQL server, select **SQL databases** in the left menu and select the *marketplace* database.
2. In the left menu on your *marketplace* database, in the **Security** section select **Auditing**.
3. Since we enabled auditing at the server-level, you should see that it's enabled here. Select **View audit logs** in the top menu bar to view the logs.
4. You should see one or more audit records with **PRINCIPAL NAME** of *ApplicationUser* and **EVENT TYPE** of **BATCH COMPLETED**. One of them should contain the details of the query you just executed. You might also see other events such as authentication failures and success. Select any record to see the full details of the event.

The screenshot shows two windows side-by-side. The left window is titled 'Audit records' and displays a list of audit events for the 'marketplace' database. It includes columns for Event Time (UTC), Principal Name, Event Type, and Action Status. Most events are for 'ApplicationUser' with 'BATCH COMPLETED' type and 'Succeeded' status. One event for 'sqladmin' is listed with 'DATABASE AUTHENTICATION SUCCEEDED'. The right window is titled 'Audit record' and shows a detailed view of a specific event from 10/16/2018 4:34:40 PM. It lists various properties: Event Time (UTC), Event Type (BATCH COMPLETED), Server Name (server-22942), Database Name (marketplace), Application Name (SQLCMD), Principal Name (ApplicationUser), Client IP (10.0.0.4), and Status (Succeeded). Below this, a 'STATEMENT' section shows the SQL query: 'SELECT FirstName, LastName, EmailAddress, Phone FROM SalesLT.Customer;'. There is also a link 'Expand statement'.

These actions configure the audits at the database server level and will apply to all databases on the server. You can also configure auditing at a database level.

Let's take a look at another feature that leverages these logs to increase the security of your database.

Auditing in Azure Cosmos DB

Azure diagnostics logs for Azure Cosmos DB will enable users to see logs for all requests made to their respective database account at the individual request level. The diagnostics logs help track how and when your databases are accessed. This feature will also provide a convenient method for configuring the

destination of the logs for the customer. Users will be able to choose the destination to either Storage Account, Event Hub or Operation Management Suite Log Analytics.

Advanced Threat Protection for Azure SQL Database

The Advanced Threat Protection system analyzes audit logs to look for potential problems and threats against your database. It includes functionality for discovering and classifying sensitive data, surfacing and mitigating potential database vulnerabilities, and detecting anomalous activities that could indicate a threat to your database. It provides a single go-to location for enabling and managing these capabilities. With one click, you can enable ATP on your entire database server, applying to all databases on the server.

Setup and configuration

Let's enable Advanced Threat Protection on our database. Advanced Threat Protection is a server-level setting, so we'll start there.

1. Back in the portal, navigate to your SQL server. In the search bar at the top of the portal, search for **server<12345>**, then select the server.
2. In the left menu, in the **Security** section, select the **Advanced Threat Protection** option.
3. Click the **ON** button to enable Advanced Threat Protection.
4. You can optionally define where notification emails will be delivered as a list of semicolon separated email addresses. **Email service and co-administrators** are enabled by default to send the threats to the service administrators.
5. You can optionally specify a storage account for historical logging of suspicious queries that have been alerted on. It is not a requirement for enabling Advanced Threat Protection, but can be useful for historical tracking. Click **Storage details** to open up the **Storage settings** dialog.
6. In the **Choose storage account** dialog, select the **server<12345>auditing** storage account you created in the previous section. Leave the other settings at the default and click **OK**.
7. Finally, select **Threat Detection types** to take a quick look at those. The preferred option is All, which is the default setting.

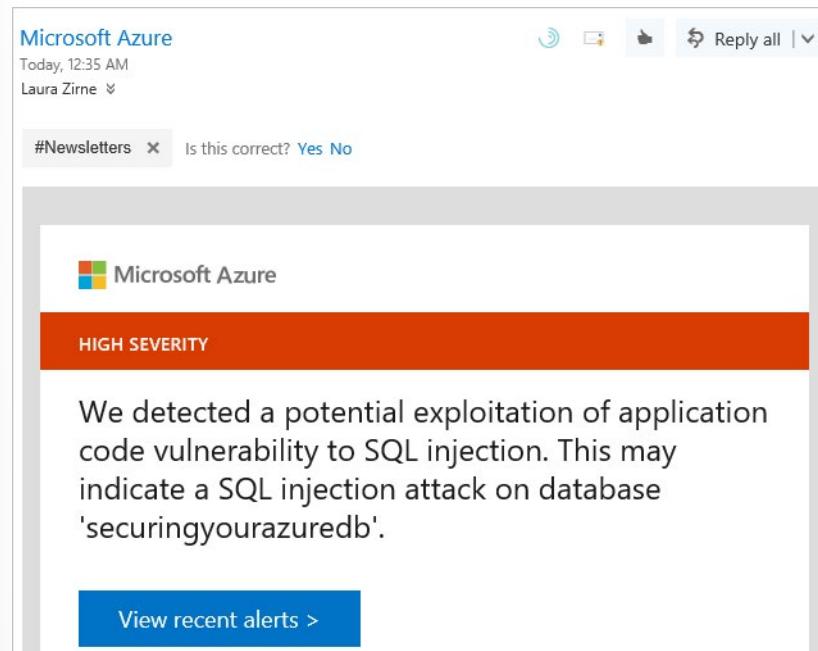
All represents the following values:

- SQL injection reports where SQL injections attacks have occurred;
 - SQL injection vulnerability reports where the possibility of a SQL injection is likely; and
 - Anomalous client login looks at logins that are irregular and could be cause for concern, such as a potential attacker gaining access.
8. Click the **Save** button to apply the changes and enable Advanced Threat Protection on your server.
- There is one database-level setting that we'll want to enable as well, and that's the Vulnerability Assessment.
1. Navigate to your *marketplace* database. In the search bar at the top of the portal, search for **market-place**, then select the database in the portal.
 2. In the left menu, in the **Security** section, select the **Advanced Threat Protection** option.
 3. In the **Vulnerability Assessment** box, you should see a message that says "Click to configure a storage account for storing scan results". You'll need to do this to enable this feature, so go ahead and click to configure.

4. In the **Choose storage account** dialog, select the **server<12345>auditing** storage account you created in the previous section.
5. You can also turn on **periodic recurring scans** to configure Vulnerability Assessment to run automatic scans once per week. A scan result summary is sent to the email address(es) you provide. In this case, we'll leave this **OFF**. Go ahead and click **Save** at the top to save your settings and enable Vulnerability Assessment.
6. Back in the main Vulnerability Assessment panel, go ahead and click **Scan** to initiate a scan of your database. When it's complete, you'll see recommendations to enhance the security of your database.

Once Advanced Threat Protection is enabled, you'll view details and results at a database level.

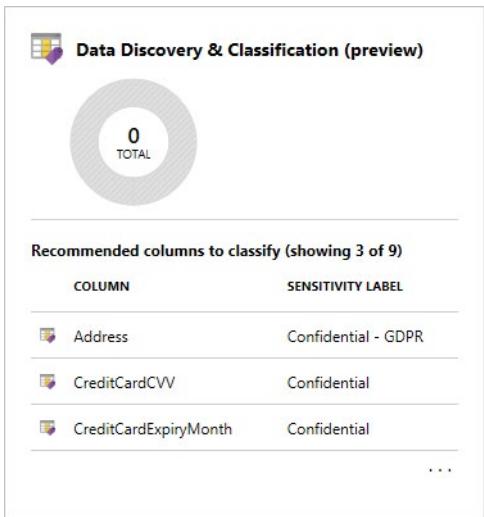
You'll receive email notifications as vulnerabilities are detected. The email will outline what occurred and the actions to take.



Data Discovery & Classification

Click on the **Data Discovery & Classification** panel.

The Data Discovery & Classification panel shows columns within your tables that need to be protected. Some of the columns may have sensitive information or may be considered classified in different countries or regions.



A message will be displayed if any columns need protection configured. This message will be formatted like “*We have found 10 columns with classification recommendations*”. You can click on the text to view the recommendations.

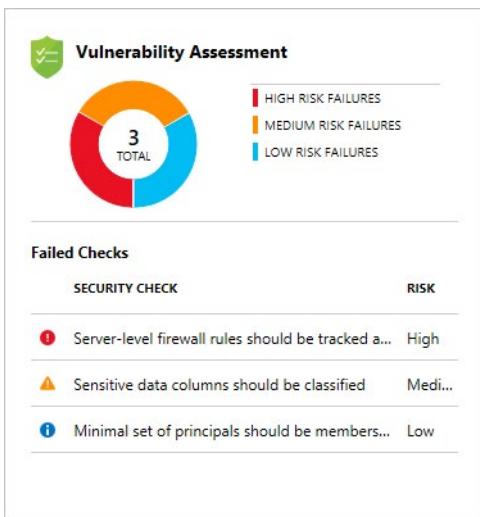
Select the columns that you want to classify by clicking the checkmark next to the column, or select the checkbox to the left of the schema header. Select the Accept selected recommendations options to apply the classification recommendations.

Next, you'll edit the columns and then define the information type and the sensitivity label for the database. Click on the Save button to save the changes.

No active recommendations should be listed once you've managed the recommendations successfully.

Vulnerability Assessment

Click on the **Vulnerability Assessment** panel.



The Vulnerability Assessment lists configuration issues on your database and the associated risk. For example, in the image above, you can see the server-level firewall needs to be set up.

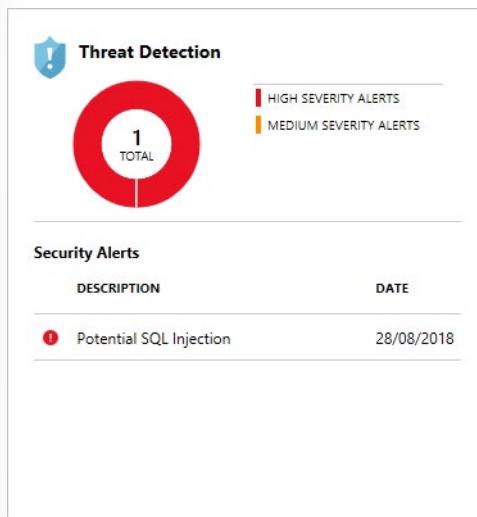
Click on the Vulnerability Assessment panel to review a full list of vulnerabilities. From here, you can click on each individual vulnerability.

On the vulnerability page you will see the details such as the risk level, which database it applies to, a description of the vulnerability, and the recommended remediation to fix the issue. Apply the remediation to fix the issue or issues. Make sure to address all the vulnerabilities.

Threat Detection

Click on the **Threat Detection** panel.

This displays a list of detected threats. For example, in this list you one potential SQL injection attacks.



Address any issues by following the recommendations. For issues such as the SQL injection warnings, you'll be able to look at the query and work backward to where the query is being executed in code. Once found, you should rewrite the code so it will no longer have the issue.

Summary

In this section, you've seen how Azure SQL Database and Azure Synapse Analytics provides a wide array of tools and features to secure your data. By using these tools and features, you can put multiple layers of defense in place to thwart attacks and keep your customer and business data secure.

You've learned how to:

- Control network access to your Database using firewall rules
- Control user access to your Database using authentication and authorization
- Protect your data in transit and at rest
- Audit and monitor your Database for access violations

All of the aspects work together to secure your database. Your customers are important, your reputation is important, and that makes your database security important.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left

running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

Which of the following is the most efficient way to secure a database to allow only access from a VNet while restricting access from the internet?

- An allow access to Azure services rule
- A server-level IP address rule
- A server-level virtual network rule
- A database-level IP address rule

Question 2

A mask has been applied to a column in the database that holds a user's email address, laura@contoso.com. From the list of options, what would the mask display for a database administrator account?

- lxxx@xxxx.com
- laura@contoso.com
- laura@xxxxxxxx.com
- Data not available

Question 3

Encrypted communication is turned on automatically when connecting to an Azure SQL Database or Azure Synapse Analytics. True or False?

- True
- False

Securing Streaming Data

Securing Streaming Data

In this section, you will learn what security can be applied to both Stream Analytics and Event Hubs to ensure that only expected data streams will be ingested into Stream Analytics for analysis.

Learning objectives

In this section, you will:

- Understand Stream Analytic Security
- Understand Event Hub Security

Stream Analytics Security

In terms of security, Azure Stream Analytics encrypts all incoming and outgoing communications and supports Transport Layer Security v 1.2. Built-in checkpoints are also encrypted.

It's important to note that Stream Analytics doesn't store the incoming data since all processing is done in-memory. As a result, you have to consider the overall architecture when considering the security of streaming data, and that may mean setting security for services such as Event Hubs or Internet of Things Hubs.

Event Hub Security

Authentication

Event Hub authentication makes use of Shared Access Signatures and Event Publishers to ensure that only applications or devices with valid credentials are only allowed to send data to an Event Hub. An Event Publisher defines a virtual endpoint for an event hub and there is only one way communication that takes place as the publisher can only be used to send messages to an event hub.

Each Event Hubs client is assigned a unique token, which is uploaded to the client. The tokens are produced such that each unique token grants access to a different unique publisher. A client that possesses a token can only send to one publisher, but no other publisher. If multiple clients share the same token, then each of them shares a publisher.

As shown earlier in the module, Shared Access Signature (SAS) is a string containing a security token that can be attached to a URI that allows you to delegate access to storage objects and specify constraints such as the permissions. In the context of Event Hubs, SAS signs an Event Hubs client token to secure it. The SAS key is generated automatically when the Event Hub namespace is created and it is a 256 bit key named RootManageSharedAccessKey.

The following example creates a send-only key when creating the event hub:

```
// Create namespace manager.  
string serviceNamespace = "YOUR_NAMESPACE";  
string namespaceManageKeyName = "RootManageSharedAccessKey";  
string namespaceManageKey = "YOUR_ROOT_MANAGE_SHARED_ACCESS_KEY";  
Uri uri = ServiceBusEnvironment.CreateServiceUri("sb", serviceNamespace,  
string.Empty);  
TokenProvider td = TokenProvider.CreateSharedAccessSignatureTokenProvid-
```

```
er(namespaceManageKeyName, namespaceManageKey);
NamespaceManager nm = new NamespaceManager(namespaceUri, namespaceManageTo-
kenProvider);

// Create event hub with a SAS rule that enables sending to that event hub
EventHubDescription ed = new EventHubDescription("MY_EVENT_HUB") { Parti-
tionCount = 32 };
string eventHubSendKeyName = "EventHubSendKey";
string eventHubSendKey = SharedAccessAuthorizationRule.GenerateRandomKey();
SharedAccessAuthorizationRule eventHubSendRule = new SharedAccessAuthoriza-
tionRule(eventHubSendKeyName, eventHubSendKey, new[] { AccessRights.Send
});
ed.Authorization.Add(eventHubSendRule);
nm.CreateEventHub(ed);
```

Sending data

Once the tokens have been created, each client is provisioned with its own unique token. When the client sends data into an event hub, it tags its send request with the token. To prevent an attacker from eavesdropping and stealing the token, the communication between the client and the event hub must occur over an encrypted channel.

Unsafe clients

If a token is stolen by an attacker, the attacker can impersonate the client whose token has been stolen. Adding a client to a Blocked client list renders that client unusable until it receives a new token that uses a different publisher.

Summary

In this section, you have learned that securing Event Hubs is the main task to ensure that only expected data streams will be ingested into Stream Analytics for analysis.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

You need to set the encryption for the data stored in Stream Analytics. What should you do?

- Use Transport Layer Security v1.2
- Set server-level IP address rule
- It cannot be done

Question 2

Authentication for an Event hub is defined with a combination of an Event Publisher and which other component?

- Shared Access Signature
- Storage Account Key
- Transport Layer Security v1.2

Module Summary

Securing Azure Data Platforms

In this module, you have learned how Azure provides a multi-layered security model to protect your data. You explored how security can range from setting up secure networks and access keys, to defining permission through to monitoring with Advanced Threat Detection across a range of data stores.

Learning objectives

In this module, you have learned:

- An introduction to security
- key security components
- Securing Storage Accounts and Data Lake Storage
- Securing Data Stores
- Securing Streaming Data

Post Course Review

After the course, consider visiting [the Microsoft Learn website⁷](#) to learn more about Role Based Access Permissions.

Important

Remember

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

⁷ <https://docs.microsoft.com/en-us/learn/modules/secure-azure-resources-with-rbac/>

Answers

Question 1

Cloud security is a shared responsibility between you and your cloud provider. Which category of cloud services requires the greatest security effort on your part?

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as a service (SaaS)

Explanation

Infrastructure as a service (IaaS). At this level, the cloud provider provides physical security to compute resources. However, it's your responsibility to patch and secure your operating systems and software, as well as configure your network to be secure. With Platform as a service (PaaS) the cloud provider handles many security concerns, more than with other categories. This includes providing physical security and keeping operating systems patched and up to date. With Software as a service (SaaS) the cloud provider handles most security concerns for you. Your main responsibility is to provide your organization's users with proper access to the software.

Question 2

Which one of the following is a key difference between global Azure and Azure Government?

- Azure Government has a marketplace from which you can deploy pre-built images from Microsoft and partners.
- Azure Government has a portal from which you can manage your resources.
- Azure Government is a physically separate instance of Azure.

Explanation

Azure Government is a sovereign cloud and is physically separate in its own set of data centers. The other answers are incorrect as Both global Azure and Azure Government offer a marketplace and a portal.

Question 1

Which of these is the strongest way to protect sensitive customer data?

- Encrypt data as it sits in your database
- Encrypt data as it travels over the network
- Encrypt data both as it sits in your database and as it travels over the network

Explanation

Encrypting your data at all times, both as it sits in your database and as it travels over the network, minimizes the opportunity for an attacker to access your data in plain text.

Question 2

You want to store certificates in Azure to centrally manage them for your services. Which Azure service should you use?

- Azure Key Vault
- Azure AD

Explanation

Azure Key Vault, because it is a centralized cloud service for storing application secrets, referred to as a secret store. MSIP is a cloud-based solution that helps an organization classify, and optionally, protect its documents and emails by applying labels. Azure AD is Microsoft's cloud-based identity and access management service that helps employees of an organization sign in and access resources.

Question 1

Mike is working as a consultant developing an application for a national Realtor company. They store thousands of images of houses in an Azure BLOB storage account. The web application Mike is developing needs to have access these images. How can Mike provide secure access for the third-party web application?

- Use Anonymous access to give the web application access
- Use a storage account key to give the web application access
- Use a Shared Access Signature to give the web application access.

Explanation

The shared access signature is the best approach to use should you require a third party application to have access to data in a blob storage account. This can provide access without sharing the storage account key. Anonymous access is not appropriate as you require secure access to the data and the storage account key is the key that provides access. Sharing this with third party increases the risk of the data being compromised

Question 2

Mike wants to gain insights should any unusual activity be occurring with his storage account with minimal configuration. What can Mike use to achieve this?

- Encryption
- Storage account signature
- Automatic Threat Detection

Explanation

Automatic Threat detection is used to proactively advise if there is any unusual activity with a storage account. Encryption is used to protect data at rest or when in transit. It is not used to give access to data in a storage account. Storage account signature does not exist.

Question 1

Which of the following is the most efficient way to secure a database to allow only access from a VNet while restricting access from the internet?

- An allow access to Azure services rule
- A server-level IP address rule
- A server-level virtual network rule
- A database-level IP address rule

Explanation

A server-level virtual network rule will allow you to allow connectivity from specific Azure VNet subnets, and will block access from the internet. This is the most efficient manner to secure this configuration.

Question 2

A mask has been applied to a column in the database that holds a user's email address, laura@contoso.com. From the list of options, what would the mask display for a database administrator account?

- lxxx@xxxx.com
- laura@contoso.com
- laura@xxxxxxxx.com
- Data not available

Explanation

laura@contoso.com. When database administrator accounts access data that have a mask applied, the mask is removed, and the original data is visible.

Question 3

Encrypted communication is turned on automatically when connecting to an Azure SQL Database or Azure Synapse Analytics. True or False?

- True
- False

Explanation

True. Azure SQL Database enforces encryption (SSL/TLS) at all times for all connections.

Question 1

You need to set the encryption for the data stored in Stream Analytics. What should you do?

- Use Transport Layer Security v1.2
- Set server-level IP address rule
- It cannot be done

Explanation

As Stream Analytics does not store data, you will be unable to set the encryption for the data stored in Stream Analytics. As a result, Using Transport Layer Security v1.2 or a server-level IP address rule is not correct in the context of this question

Question 2

Authentication for an Event hub is defined with a combination of an Event Publisher and which other component?

- Shared Access Signature
- Storage Account Key
- Transport Layer Security v1.2

Explanation

A Shared Access Signature in combination with an Event Publisher is used to define authentication for an event hub.

Module 9 Monitoring and Troubleshooting Data Storage and Processing

Module Introduction

Monitoring and Troubleshooting Data Storage and Processing

In this module, the student will get an overview of the range of monitoring capabilities that are available to provide operational support should there be issue with a data platform technology. They will explore the common data storage and data processing issues. Finally, disaster recovery options are revealed to ensure business continuity.

Learning Objectives

In this module, you will learn:

- General Azure monitoring capabilities
- Troubleshoot common data storage issues
- Troubleshoot common data processing issues
- Manage disaster recovery

General Azure Monitoring Capabilities

General Azure monitoring capabilities

Even though services are being hosted and monitored by Microsoft in Azure, you also have the capability to perform general monitoring to help you proactively baseline and measure the performance or operations of your data estate. This may facilitate the need to understand if you have the right size of storage, that performance is meeting service level agreements, or even use the monitoring to help you reduce costs of a service.

Learning Objectives

In this section, you will learn:

- Azure Monitor
- Monitoring the network
- Diagnose and Solve Problems

Azure Monitor

When setting up monitoring, it is important to get an holistic view of the operations of an organizations application and data platform estate. Azure Monitor can help in this by collecting, analyzing, and acting on telemetry from both cloud and on-premises environments. It helps you understand how your applications are performing and proactively identifies issues affecting them and the resources they depend on such as the data store on which the application resides. Azure Monitor collects both metric and log data, and enables you to set up alerts

Metric data

Provides quantifiable information about a system over time that enables you to observe the behavior of a system.

Log data

Logs can be queried and even analysed using Azure Monitor logs. In addition, this information is typically presented in the overview page of an Azure Resource in the Azure portal. There are different types of logs that contain a different kind of data organized into records with a different sets of properties.

Alerts

Alerts notify you of critical conditions and potentially take corrective automated actions based on triggers from metrics or logs.

Azure Monitor collects data from each of the following tiers:

- **Application monitoring data:** Data about the performance and functionality of the code you have written, regardless of its platform.
- **Guest OS monitoring data:** Data about the operating system on which your application is running. This could be running in Azure, another cloud, or on-premises.
- **Azure resource monitoring data:** Data about the operation of an Azure resource.
- **Azure subscription monitoring data:** Data about the operation and management of an Azure subscription, as well as data about the health and operation of Azure itself.

- **Azure tenant monitoring data:** Data about the operation of tenant-level Azure services, such as Azure Active Directory.

There are several features and tools that can help you to gain insights into the areas that you are monitoring, including Application Insights to provide deep insights into different aspects of your application and specific Azure services.

Monitoring solutions

Azure Monitor includes *monitoring solutions* which are packaged sets of logic that provide insights for a particular application or service. They include logic for collecting monitoring data for the application or service, queries to analyze that data, and views for visualization. Monitoring solutions are available from Microsoft and partners to provide monitoring for various Azure services and other applications.

Using Azure Monitor

Azure Monitor is available when you log into the Azure Portal. By clicking on the Monitor icon on the portal blade in the right, you are presented with the following screen.

The screenshot shows the Azure Monitor - Overview page. On the left, there's a navigation sidebar with links like Overview, Activity log, Alerts, Metrics, Logs, Service Health, Applications, Virtual Machines (preview), Containers, Network, More, Settings, Diagnostics settings, Autoscale, Usage and estimated costs, Advisor recommendations, and New support request. The main content area has a heading "Monitor your applications and infrastructure" with a subtext: "Get full stack visibility, find and fix problems, optimize your performance, and understand customer behavior all in one place. [Learn more](#)". Below this are three sections: "Monitor & Visualize Metrics" (with a button "Explore Metrics"), "Query & Analyze Logs" (with a button "Search Logs"), and "Setup Alert & Actions" (with a button "Create Alert"). At the bottom, there's a "Quick Starts" section with links for "Azure VMs", "Azure Web Apps", and "ASP.NET Apps from Visual Studio".

From here, you can click on the **Explore Metrics** button to view information about the specific service you want to monitor by adding counter and measures. Alternatively, the **Search Logs** enables you to write queries to retrieve specific information, or to view a set of logs, or to view the errors and warnings. Finally, the **Create Alert** button in Azure Monitor enables you to send email or text messages when a certain condition is met for a given application or service.

Note

You will want to liaise with the Azure administrator to setup Azure monitor. They will likely have a monitoring strategy in place that you will want to align to

Monitoring the network

As part of the holistic approach to monitoring, it also makes sense to make sure that you monitor the network activity. Azure Monitor logs within Azure monitor has the capability to monitor and measure network activity. As an example, you may receive complaints that users are unable to access reports; however, this complaint may be a symptom of an underlying root cause. In this example, the fact that users are unable to access reports may be due to a network issues.

Within Azure, there are two main activities to monitor your network activity:

- **Network Performance Monitor**

- Network performance monitor measure the performance and reachability of the networks that you have configured. It is used to monitor connectivity between:
 - Public cloud and on-premises
 - Data centers and user locations (branch offices)
 - Subnets hosting various tiers of a multi-tiered application.

You can use this to monitor network latency, dropped packets and connection issues so that you can establish that the underlying communications are working fine

- **Application Gateway Analytics**

- Networks can also contain additional devices and applications to manage the networks. Examples of application gateways can include Power BI and Azure Data Factory gateways that manage network traffic from on-premises data stores into Azure and Power BI. It contains rich, out-of-the box views you can get insights into key scenarios, including:
 - Client and server errors reported by your application gateway
 - Requests per hour per application gateway
 - Failed requests per hour per application gateway
 - Client and server errors by user agent
 - Count of healthy and unhealthy hosts per application gateway
 - Failed requests per application gateway

Note

You will want to liaise with the Azure administrator to setup Azure monitor. They will likely have a monitoring strategy in place that you will want to align to

Diagnose and solve problems

Within the blade of a data platform technology in the Azure portal, there is a section named **Diagnose and solve problems** that can be used to help you troubleshoot issues with a data store. In this area, you can perform the following tasks:

- **Resource Health**

This shows the availability of a given data platform service

- **Recent Activity**

This provides Quick Insights of the following areas:

- Errors
- Failed deployments
- Alerts
- Service Health
- Role Assignments

In addition you can view all activities.

Note

This capability is not available in Azure Databricks

Summary

In the module, you have learned the capabilities of Azure monitoring to help you proactively baseline and measure the performance or operations of your data estate. You have learned how network analytics can be used to rule out networking issues. Finally, you have seen how data platform technologies contain a diagnose and solve problems section to help you troubleshooting issues.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

You want to respond to the critical condition and take corrective automated actions using Azure Monitor. Which feature would you use?

- Metric data
- Log data
- Alerts

Question 2

You are receiving an error message in Azure Synapse Analytics. You want to view information about the service and help to solve the problem, what can you use to quickly check the availability of the service?

- Network performance monitor
- Diagnose and solve problems
- Azure monitor

Troubleshoot Common Data Platform Issues

Troubleshoot Common Data Platform Issues

In this section, you will explore the common issues that can hinder the data platform. Whilst it does not provide an exhaustive coverage of the issues, you will explore issue that impact connectivity, performance and storage issues.

Learning Objectives

In this module, you will learn:

- Connectivity issues
- Performance issues
- Storage issues

Connectivity issues

There are a range of issues that can impact connectivity issues, and these will be reported back as the error occurs. Below is a list of the common issues:

Unable to connect to a data platform engine

If you are having issues connecting to a data store, the first area that you should check is the firewall configuration. In here you should confirm if the IP address of your client has been added to the approved IP list. If your IP address is added, then test the connection policy. Alternatively, you can test the connection by accessing it from a location external to your network. There are also examples when you are unable to connect due to pre determined maintenance schedules of a service. Watch our for emails that announce these schedules in advance.

Authentication failures

Another reason for perceived network connection loss is a failure in authentication. The first check is to ensure that the user name and password is correct for any connections that makes use of SQL Server or Active Directory authentication. From an Azure Blob perspective, check the storage account keys and ensure that they match in the connection string.

Cosmos DB Mongo DB API connection errors

Whenever a Mongo client is initialized to a remote address, the driver establishes more than one connection. On the server side, connections which are idle for more than 30 minutes are automatically closed down. It is possible that some of the connections in the connection pool would timeout (if that connection was not picked by driver to issue user commands for sometime). To avoid connectivity messages, change the connection string to set maxConnectionIdleTime to 1-2 minutes.

SQL Database fail-over

Should you receive an “unable to connect” message (error code 40613) in the Azure SQL Database, this scenario commonly occurs when a database has been moved because of deployment, failover, or load

balancing. This is likely to be a transient connectivity issue because reconfigurations are generally expected to complete in less than 60 seconds.

If all else fails

Check the **Microsoft Azure Service Dashboard**¹ for any known issues. And check and make sure there are no physical errors in the networking, for example damaged cables.

Performance issues

There are a range of issues that can impact performance, and you may not necessarily receive these issues as an error. Slow performance is typically reported as a symptomatic problem from reports being slow to data loads taking too long. You should consider checking the following areas:

Queries are slow

Data Lake storage

When there are query performance issues with Data Lake Storage, ensure that **hierarchical namespace** is enabled.

Cosmos DB

To speed up queries, try one or more of the following steps.

- Whenever possible, avoid full scans on the collection.
- All user defined functions (UDFs) and built-in functions will scan across all documents within the query scope; for example,
SELECT * FROM c will have a broader query scope than SELECT * FROM c.foo = 'bar'. To optimize performance, add a WHERE clause to your queries with UDFs and built-in functions to reduce the query scope.
- Use direct mode as your connection mode when configuring your connection policy.
- Tune the page size for queries and read feeds for better performance using the x-ms-max-item-count header.
- For partitioned collections, query in parallel to increase performance and leverage more throughput.

To speed up your database, try one or more of the following steps:

- Install the latest DocumentDB SDK. DocumentDB SDKs are constantly being improved to provide the best performance.
- Use direct mode as your connection mode when configuring your connection policy.
- Increase the number of threads/tasks to decrease the wait time while fulfilling requests.
- For .NET, increase the System.Net MaxConnections to have multiple simultaneous connections to DocumentDB, which has a default of 50 threads for .NET SDK 1.8.0 and above.

¹ <https://azure.microsoft.com/en-gb/status/>

- When possible, deploy your application and DocumentDB database to the same Azure region. For a ballpark comparison, calls to DocumentDB within the same region complete within 1-2ms, but latency between the West and East coast of the US is more than 50ms.
- Try increasing the allocated Request Units (RUs) for your collection.

In order to achieve the best performance for applications using Azure Cosmos DB, ensure that you are following below tips:

- For .NET clients use Direct/TCP connectivity mode and for Java clients use Direct/HTTPS connectivity mode for best performance.
- For .NET clients, call OpenAsync() to avoid increased latency for first few requests due to warm up time.
- For high transaction workloads, increase the number of threads/connection pool size to maximize concurrency of requests.
- Use singleton client instance for the lifetime of your application.
- Review your indexing policy and explicitly include paths you need for queries (if possible).
- Larger documents consume higher RUs for reads and writes, so try to keep your documents small for best performance.

SQL Data Warehouse

The following are the most common issues for slow query performance:

- Ensure table statistics are created and kept up to date. SQL Data Warehouse does not currently support automatic creation or update of statistics. As you load data into your data warehouse, query plans can regress if statistics are not up to date.
- The SQL Data Warehouse query optimizer is a cost-based optimizer. It compares the cost of various query plans, and then chooses the plan with the lowest cost, which is in most cases the plan that executes the fastest. The cost-based optimized relies on table statistics to ensure the most optimized query plan is selected.
- Ensure you consistently have high quality row groups and rebuild your indexes to improve rowgroup quality. Use the appropriate resource class, service level, and an appropriate number of partitions as you load data into your data warehouse to prevent poor columnstore index quality.
- A row group is a chunk of rows that are compressed together in columnstore. To get full speed from your data warehouse, it is important to maximize columnstore row group quality. For best compression and index efficiency, the columnstore index needs to compress the maximum of 1,048,576 rows into each rowgroup.
- Reduce query data movement operations by investigating your query plan. You can have a suboptimal plan with:
 - A poor selection of a table distribution key causing data skew.
 - Broadcast move operations which can be avoided by replicating certain tables.
 - Monitor to ensure your query is not queued and your data warehouse has enough concurrency slots.
- SQL Data Warehouse has a fixed number of concurrency slots depending on the current service level. Queries require several concurrently slots based on their resource class to ensure adequate resources are provided for optimal and efficient query performance. They become queued when there are not enough slots available.

- Ensure enough tempdb and memory have been allocated during query execution.
- It is recommended to scale if you see your datawarehouse is close to maxing out its tempdb capacity during high activity.
- Tempdb is used as a staging area for data movement operations as well as when a query reaches its memory grant.
- Each query has a specific memory grant depending on its resource class and the data warehouse service level. When queries consume their grant, they must spill into tempdb slowing down query performance.
- Avoid directly issuing queries against external tables by loading data first.
- External tables are not optimal for queries. External tables residing in Blob storage or Azure data lake does not have compute resources backing them; therefore the data warehouse cannot offload this work. Therefore, your data warehouse will be required to read the entire file into tempdb to scan the data.

Azure SQL Database

- Identifying and adding missing indexes

The most common source of this issue is the lack of appropriate indexes to satisfy filters or other restrictions in a query. Often, missing indexes manifests as a table scan when an index seek could suffice.

- Query tuning and hinting

The query optimizer in Azure SQL Database is similar to the traditional SQL Server query optimizer. Most of the best practices for tuning queries and understanding the reasoning model limitations for the query optimizer also apply to Azure SQL Database. If you tune queries in Azure SQL Database, you might get the additional benefit of reducing aggregate resource demands. Your application might be able to run at a lower cost than an un-tuned equivalent because it can run at a lower compute size.

Throughput and latency issues

Don't forget to collocate clients in same Azure region for better performance.

Storage Issues

Data Consistency

Cosmos DB

Azure Cosmos DB supports five consistency levels:

- Strong
- Bounded staleness
- Session
- Consistent prefix
- Eventual

This provides a balance between consistency and concurrency. If there is data that does not appear to be consistent, then consider changing the consistency level to Strong to ensure that the data integrity is

maintained. Of course, the priority may be to ensure maximizing concurrency, in which case Eventual will meet this need.

SQL Data Warehouse

The intention with Azure Synapse Analytics is a cloud-based Enterprise Data Warehouse (EDW) that leverages Massively Parallel Processing (MPP) to quickly run complex queries across petabytes of data. It is not intended to be transaction in nature, as a result, the focus is to control data loads and maximize concurrency by selecting the appropriate DWU.

SQL Database

Azure SQL Database supports the following transaction isolation levels:

- SERIALIZABLE
- SNAPSHOT
- REPEATABLE READ
- READ COMMITTED
- READ UNCOMMITTED

SERIALIZABLE maximizes data integrity, READ UNCOMMITTED concurrency. SQL Database default database wide setting is to enable read committed snapshot isolation (RCSI) by having both the READ_COMMITTED_SNAPSHOT and ALLOW_SNAPSHOT_ISOLATION database options set to ON. You cannot change the database default isolation level. However, you can control the isolation level explicitly on a connection. You do this by setting SET TRANSACTION ISOLATION LEVEL controls before you BEGIN TRANSACTION:

Data Corruption

Data corruption can occur on any of the data platforms for a variety of reasons. Recovering from these scenarios will be explored in the disaster.

Summary

In this section, you have explored the common issues that can hinder the data platform in the areas of connectivity, performance and storage issues.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

You want to maximize the data integrity of data that is stored in a Cosmos DB. Which consistency level should you choose?

- Session
- Strong
- Eventual

Question 2

You perform a daily SQL Data Warehouse using Polybase with CTAS statements to load the data. User are reporting that the reports are running slow. What should you do to improve the performance of the query?

- Try increasing the allocated Request Units (RUs) for your collection.
- Install the latest DocumentDB SDK.
- Add table statistics are created and kept up to date

Troubleshoot Common Data Processing Issues

Troubleshoot Common Data Processing Issues

In this section, you will explore the common issues that can hinder data processing from both a streaming and a batch perspective. You will also explore the common issues that impact Azure Data Factory.

Learning Objectives

In this module, you will learn:

- Troubleshoot streaming data
- Troubleshoot batch data loads
- Troubleshoot Azure Data Factory

Troubleshooting Stream Analytics

When using Stream Analytics, a Job encapsulates the Stream Analytic work and is made up of three components:

- A job input
- A Job query
- A job output

Each component may cause errors and require troubleshooting in the following ways

Stream Analytic Job Input

The job input contains a **Test Connection** button to validate that there is connectivity with the input. However, most errors associated with a job input is due to the malformed input data that is being ingested. When a Stream Analytics job receives a malformed message, it drops the message and gives a warning in the portal. A warning symbol is shown on the Inputs tile of your Stream Analytics job, and will remain there as long as the job is in running state.

You can view diagnostics logs and execution logs to view the details of the warning. A malformed message could be caused by something as simple as a missing parenthesis or a brace in a JSON object, or an incorrect timestamp format.

You can also exceed event reader limits on a Stream Analytics Job that will cause an error. If your streaming query syntax references the same input Event Hub resource multiple times, the job engine can use multiple readers per query from that same consumer group. When there are too many references to the same consumer group, the job can exceed the limit of five and thrown an error. Multiple select statements, UNION queries and self joins are the types of queries that can make multiple references to a job input and cause it to exceed its limit.

Stream Analytic Query

A common issue associated with Stream Analytics query is the fact that the output produced is not expected. In this scenario it is best to check the query itself to ensure that there is no mistakes on the code there. Items to look out for include the wrong where clause being used, incorrect timestamps in the

query. There is a **Test** button in the Job Query pane that will allow you to test queries against sample data. If this fails you can use the Audit Logs, and job diagnostic logs to identify and debug errors.

Stream Analytic Job Output

As with the job input, there is a **Test Connection** button to validate that there is connectivity with the output, should there be no data appearing. You can also use the **Monitor** tab in Stream Analytics to troubleshoot issues. Here you should look out for the following metrics

- If Input Events > 0, the job is able to read input data. If Input Events is not > 0, then:
 - To see whether the data source has valid data, check it by using Service Bus Explorer. This check applies if the job is using Event Hub as input.
 - Check to see whether the data serialization format and data encoding are as expected.
 - If the job is using an Event Hub, check to see whether the body of the message is Null.
- If Data Conversion Errors > 0 and climbing, the following might be true:
 - The output event does not conform to the schema of the target sink.
 - The event schema might not match the defined or expected schema of the events in the query.
 - The datatypes of some of the fields in the event might not match expectations.
- If Runtime Errors > 0, it means that the job can receive the data but is generating errors while processing the query.
 - To find the errors, go to the Audit Logs and filter on Failed status.
- If InputEvents > 0 and OutputEvents = 0, it means that one of the following is true:
 - Query processing resulted in zero output events.
 - Events or its fields might be malformed, resulting in zero output after query processing.
 - The job was unable to push data to the output sink for connectivity or authentication reasons.

Batch Data load issues

When trying to resolve data load issues, it is first pragmatic to make the holistic checks on Azure, as well as the network checks and diagnose and solve issue check. This ensures that the obvious issues are eliminated. Beyond this, there are specific scenarios you would want to check for a given data platform.

Azure Blob and Data Lake Store

Notwithstanding network errors; occasionally, you can get timeout or throttling errors that can be a symptom of the availability of the storage accounts. Should a client receive regular error code 500, this denotes an operational timeout of the storage service. If this issue is intermittent, then it could be an indicator of a load balancing operation. If it is persistent, then you should check the application timeouts within the client application, and if this is within acceptable levels, then it indicates that the application is hitting scalability limits of the performance tier of the storage account.

SQL Data Warehouse

Sometimes you can have issues with loading data into SQL Data Warehouse. You should always leverage Polybase to batch load into your data warehouse and avoid singleton inserts. This means using Create Table As Select (CTAS) statements rather than INSERT INTO to avoid full logging into staging heap tables

(especially on Gen2). Partition switching is also a technique that should be used to optimize load performance. You can also maximize throughput when using gzip text files by splitting files up into 60 or more files to maximize parallelism. You should also define the appropriate resource class and service level to ensure enough memory. As you scale your service level, SQL Data Warehouse increases the numbers of readers and writers for parallelism. Finally, make sure to co-locate your staging area (blob storage or data lake store) and your data warehouse to minimize latency.

Cosmos DB

With Cosmos DB, you may have not provisioned enough RUs to meet your per-second request rates. The throughput (RUs) can be increased via the Azure portal or the Cosmos DB SDKs. If your application is not latency sensitive, you can keep the RUs fixed and increase the number of retries in your client application. For the core (SQL) API, this is available as a configuration in the native SDKs. If your throughput utilization is low overall, but you still see rate limiting, you may have to review your partition key selection for skews. See Synthetic partition keys for some solutions to deal with skews.

If your RU consumption is predominantly from writes, you may be able to reduce RU consumption by modeling as smaller documents, tuning indexing policy, or by writing in batches. See Request units for details. If your RU consumption is predominantly from reads, you may be able to reduce RU consumption by tuning individual queries, lowering consistency level, or by reading in batches. See Request units for details.

SQL Database

There are a wide range of issue that should be considered with Azure SQL Database when it comes to the batch loading of data. There are also a wide range of tools that can help you to troubleshoot these issues including:

- Metrics and logging
- Performance recommendations
- Intelligent Performance Insight
- Automatic Tuning
- Query Performance Insights
- Azure SQL Analytics

Full details of each of these services can be reviewed here²

Troubleshooting Azure Data Factory

Troubleshooting Azure Data Factory (ADF) is complex. You must begin by first assessing that there are no issues with the general Azure environment, check the networking and then finally check that the data stores to which ADF will be transferring data have no issues either. Beyond this, many of the issues that occur with Azure Data Factory can be found by understanding the capabilities and limits of Azure Data Factory itself.

Examples of limits that are exceeded include the maximum activities per pipeline, which includes inner activities for container which is limited to 40. Another is the maximum number of linked integration runtimes that can be created against a single self-hosted integration runtime which is limited to 20.

² <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-monitoring-tuning-index>

You can monitor pipelines and activity runs visually by logging into the Azure Portal, navigating to the Data Factory instance, and click on the **Manage & Monitor** blade to view the information. The following table lists the columns that can be viewed for the pipeline runs, including:

| Column Name | Description |
|---------------|---|
| Pipeline Name | Name of the pipeline. |
| Actions | Single action available to view activity runs. |
| Run Start | Pipeline run start date time (MM/DD/YYYY, HH:MM:SS AM/PM) |
| Duration | Run duration (HH:MM:SS) |
| Triggered By | Manual trigger, Schedule trigger |
| Status | Failed, Succeeded, In Progress |
| Parameters | Pipeline run parameters (name, value pairs) |
| Error | Pipeline run error (if/any) |
| Run ID | ID of the pipeline run |

The following table lists the columns that can be viewed for the activities, including:

| | |
|---------------|---|
| Activity Name | Name of the activity inside the pipeline. |
| Activity Type | Type of the activity, such as Copy, HDInsightSpark, HDInsightHive, etc. |
| Run Start | Activity run start date time (MM/DD/YYYY, HH:MM:SS AM/PM) |
| Duration | Run duration (HH:MM:SS) |
| Status | Failed, Succeeded, In Progress |
| Input | JSON array describing the activity inputs |
| Output | JSON array describing the activity outputs |
| Error | Activity run error (if/any) |

Another common issue that is hit is the limit on the bytes per object for pipeline objects which currently stands at 200KB. The limit does no relate to the data itself, but to the JSON definitions of the Pipeline, data set, and linked service objects that represent the workload created. Should this limit be reached it is recommended that the pipeline is split into several smaller pipelines.

Summary

In this section, you have learned the common issues that can hinder data processing from both a streaming and a batch perspective, and learned the common issues that impact Azure Data Factory.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

What is the maximum number of activities per pipeline in Azure Data Factory?

- 40
- 60
- 80

Question 2

You are monitoring the job output of a streaming analytics job. The monitor is reporting back that there is a "Runtime Errors > 0". What does this mean?

- The data serialization format and data encoding are incorrect
- The job can receive the data but is generating errors while processing the query.
- The datatypes of some of the fields in the event might not match expected format

Managing Disaster Recovery

Managing Disaster Recovery

Ensuring the availability of data in Azure is a responsibility that is taken seriously by Microsoft, and should be taken seriously by an organization. As a data engineer, understanding the capabilities of data redundancy and disaster recovery for a given data platform is an imperative.

Learning Objectives

In this module, you will learn:

- Data redundancy
- Disaster recovery

Data redundancy

Data redundancy is the process of storing data in multiple locations to ensure that it is highly available. Microsoft Azure provides this capability automatically in many data platform technologies.

Azure Blob Store and Data Lake Storage (Gen 2)

Azure storage by default is replicated to provide protection against transient hardware failures, network or power outages, and massive natural disasters. You have the ability to choose where the data is replicated to by choosing one of the following options when creating the storage account:

- Locally redundant storage (LRS)
- Zone-redundant storage (ZRS)
- Geo-redundant storage (GRS)
- Read-access geo-redundant storage (RA-GRS)

Locally redundant storage (LRS)

Locally redundant storage is available as a premium storage option and provides the lowest-cost replication option. LRS replicates data three times within the region in which the storage account is created.

Zone-redundant storage (ZRS)

Zone-redundant storage replicates data three times across two to three facilities, either within a single region or across two regions.

Geo-redundant storage (GRS)

Geo-redundant storage replicates data to a secondary region that is hundreds of miles away from the primary region.

Read-access geo-redundant storage (RA-GRS)

Read-access geo-redundant storage replicates your data to a secondary region, as with GRS, but also then provides read-only access to the data in the secondary location.

The following table provides a quick overview of the differences between the storage replication types:

| Replication strategy | LRS | ZRS | GRS | RA-GRS |
|---|-----|-----|-----|--------|
| Data is replicated across multiple facilities. | No | Yes | Yes | Yes |
| Data can be read from the secondary location and from the primary location. | No | No | No | Yes |
| Number of copies of data maintained on separate nodes. | 3 | 3 | 6 | 6 |

You can change your storage account's replication strategy by using the Azure portal, Azure Powershell, Azure CLI, or one of the Azure client libraries. Changing the replication type of your storage account does not result in down time.

SQL Data Warehouse

SQL Data Warehouse performs a **geo-backup** once per day to a paired data center. The RPO for a geo-restore is 24 hours. You can restore the geo-backup to a server in any other region where SQL Data Warehouse is supported. A geo-backup ensures you can restore data warehouse in case you cannot access the restore points in your primary region. Geo-backups are on by default. If your data warehouse is Gen1, you can opt out if you wish. You cannot opt out of geo-backups for Gen2 as data protection is a built-in guaranteed.

Cosmos DB

Azure Cosmos DB is a globally distributed database service. You can configure your databases to be globally distributed and available in any of the Azure regions. You can add or remove the regions associated with your account at any time. Your application doesn't need to be paused or redeployed to add or remove a region. It continues to be highly available all the time because of the multihoming capabilities that the service provides.

SQL Database

SQL Database provides a wide range of features that provides availability of data. This includes

- Temporal tables enable you to restore row versions from any point in time.
- Built-in automated backups and Point in Time Restore enables you to restore complete database to some point in time within the last 35 days.
- You can restore a deleted database to the point at which it was deleted if the SQL Database server has not been deleted.

- Long-term backup retention enables you to keep the backups up to 10 years.
- Active geo-replication enables you to create readable replicas and manually failover to any replica in case of a data center outage or application upgrade.
- Auto-failover group allows the application to automatically recover in case of a data center outage.

A full review of this capability can be read from this location³

Disaster recovery

As a part of the disaster recovery strategy, there should be processes that are involved in backing up or providing failover for databases in an Azure data platform technology. Depending on circumstances, there are numerous approaches that can be adopted. These strategies accommodate the numerous business scenarios ranging from 24/7 business operations to non-continual operations. Understanding the numerous strategies will help you safely back up your data and restore it in a timely manner.

Azure Blob Store and Data Lake Storage (Gen 2)

Azure Storage supports account failover for geo-redundant storage accounts. With account failover, you can initiate the failover process for your storage account if the primary endpoint becomes unavailable. The failover updates the secondary endpoint to become the primary endpoint for your storage account. Once the failover is complete, clients can begin writing to the new primary endpoint.

SQL Data Warehouse

SQL Data Warehouse performs a **geo-backup** once per day to a paired data center. The RPO for a geo-restore is 24 hours. You can restore the geo-backup to a server in any other region where SQL Data Warehouse is supported.

In addition, there is a data warehouse snapshot feature that enables you to create a restore point to create a copy of the warehouse to a previous state. SQL Data Warehouse takes snapshots of your data warehouse throughout the day creating restore points that are available for seven days. This retention period cannot be changed. SQL Data Warehouse supports an **eight-hour** recovery point objective (RPO). You can restore your data warehouse in the primary region from any one of the snapshots taken in the past seven days. To see when the last snapshot started, run this query on your online SQL Data Warehouse.

```
select top 1 *
from sys.pdw_loader_backup_runs
order by run_id desc;
```

There are also user defined restore points that enables you to manually trigger snapshots to create restore points of your data warehouse before and after large modifications. This capability ensures that restore points are logically consistent, which provides additional data protection in case of any workload interruptions or user errors for quick recovery time.

Cosmos DB

With Azure Cosmos DB, not only your data, but also the backups of your data are highly redundant and resilient to regional disasters. It takes a backup of your database every **4 hours** and at any point of time, only the latest 2 backups are stored. If you have accidentally deleted or corrupted your data, you should

³ <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-business-continuity>

contact Azure support within 8 hours so that the Azure Cosmos DB team can help you restore the data from the backups. If a database or container is deleted, this is retained for 30 days.

SQL Database

SQL Database automatically creates database backups that are kept between 7 and 35 days, and uses Azure read-access geo-redundant storage (RA-GRS) to ensure that they are preserved even if the data center is unavailable. You can configure a long-term retention period of up to 10 years should you require it. SQL Database can create a full, differential, and transaction log backups to perform a point-in-time restore.

Summary

In this section, you have learned how to ensure the availability of data in Azure by understanding the capabilities of data redundancy and disaster recovery for a given data platform.

Important

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

Review Questions

Question 1

How long is the Recovery Point Objective for Azure Synapse Analytics?

- 4 hours
- 8 hours
- 12 hours

Question 2

How often is a backup taken for Azure Cosmos DB?

- 4 hours
- 8 hours
- 12 hours

Module Summary

Module Summary

In this module, you have been given an overview of the range of monitoring capabilities that are available to provide operational support should there be issue with a data platform technology. You learned the common data storage and data processing issues. Finally, you explored disaster recovery options to ensure business continuity.

Learning Objectives

In this module, you have learned:

- General Azure monitoring capabilities
- Troubleshoot common data storage issues
- Troubleshoot common data processing issues
- Manage disaster recovery

Post Course Review

After the course, watch **Rahul Bagaria joins Lara Rubbelke discuss Azure monitor in this 16 minute video⁴**.

Important

Remember

Should you be working in your own subscription while running through this content, it is best practice at the end of a project to identify whether or not you still need the resources you created. Resources left running can cost you money. You can delete resources one by one, or just delete the resource group to get rid of the entire set.

⁴ <https://youtu.be/B5X7M0WI0tY>

Answers

Question 1

You want to respond to the critical condition and take corrective automated actions using Azure Monitor. Which feature would you use?

- Metric data
- Log data
- Alerts

Explanation

Alerts enables you to respond to the critical condition and take corrective automated actions. Metric data provides quantifiable information about a system. Log data provides information in a log format that can be analysed in Log Analytics.

Question 2

You are receiving an error message in Azure Synapse Analytics. You want to view information about the service and help to solve the problem, what can you use to quickly check the availability of the service?

- Network performance monitor
- Diagnose and solve problems
- Azure monitor

Explanation

Diagnose and solve problems. Diagnose and solve problems can quickly show you the service availability. Azure Monitor allows you in collecting, analyzing, and acting on telemetry from both cloud and on-premises environments. Network performance monitor measure the performance and reachability of the networks that you have configured.

Question 1

You want to maximize the data integrity of data that is stored in a Cosmos DB. Which consistency level should you choose?

- Session
- Strong
- Eventual

Explanation

Strong. Strong ensures that the data integrity is maintained. Eventual maximises concurrency to the cost of data integrity. Session is a consistency level between Strong and Eventual.

Question 2

You perform a daily SQL Data Warehouse using Polybase with CTAS statements to load the data. User are reporting that the reports are running slow. What should you do to improve the performance of the query?

- Try increasing the allocated Request Units (RUs) for your collection.
- Install the latest DocumentDB SDK.
- Add table statistics are created and kept up to date

Explanation

Add table statistics are created and kept up to date. Trying to increase the allocated Request Units (RUs) for a collection and Install the latest DocumentDB SDK is a change that is made to improve Cosmos DB.

Question 1

What is the maximum number of activities per pipeline in Azure Data Factory?

- 40
- 60
- 80

Explanation

40 is the maximum number of activities per pipeline in Azure Data Factory.

Question 2

You are monitoring the job output of a streaming analytics job. The monitor is reporting back that there is a "Runtime Errors > 0". What does this mean?

- The data serialization format and data encoding are incorrect
- The job can receive the data but is generating errors while processing the query.
- The datatypes of some of the fields in the event might not match expected format

Explanation

The job can receive the data but is generating errors while processing the query. The data serialization format and data encoding are incorrect would report back as Input Events is not > 0. If Data Conversion Errors > 0 and climbing, it could indicate that the datatypes of some of the fields in the event might not match expected format

Question 1

How long is the Recovery Point Objective for Azure Synapse Analytics?

- 4 hours
- 8 hours
- 12 hours

Explanation

8 hours. Azure Synapse Analytics has a Recovery Point Objective of 8 hours.

Question 2

How often is a backup taken for Azure Cosmos DB?

- 4 hours
- 8 hours
- 12 hours

Explanation

4 hours. Azure Cosmos DB takes a backup every 4 hours