Microsoft

Microsoft Official Course

DP-201T01

Designing an Azure Data Solution

DP-201T01

Designing an Azure Data Solution

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.

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 contre-façon 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

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-201: Designing 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 201 certification
- About this course

Course Introduction

This module provides an overview of the following

- About this course
- Course agenda
- Course audience
- Course pre-requisites
- DP201: Designing an Azure Data Solution certification details

About this course

In this course, the students will design various data platform technologies into solutions that are in line with business and technical requirements. This can include on-premises, cloud, and hybrid data scenarios which incorporate relational, No-SQL or Data Warehouse data. They will also learn how to design process architectures using a range of technologies for both streaming and batch data.

The students will also explore how to design data security including data access, data policies and standards. They will also design Azure data solutions which includes the optimization, availability and disaster recovery of big data, batch processing and streaming data solutions.

This course is run in a workshop format. In this format, each module will be taught in approximately 20 - 30 minutes. This is then followed by a lab that is a group activity of approximately 50 - 60 mins. The module then finishes with a 20 - 30 min class review of the group solutions presented with the class discussing the relative merits of the solutions or architectures presented by each group.

Course Agenda

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

Module 1: Architecture Considerations

In this module, the students will learn how to design and build secure, scalable and performant solutions in Azure by examining the core principles found in every good architecture. They will learn how using key principles throughout your architecture regardless of technology choice, can help you design, build, and continuously improve your architecture for an organizations benefit.

Module Objectives:

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

- Describe the core principles for creating architectures
- Design with Security in mind
- Consider performance and scalability
- Design for availability and recoverability
- Design for efficiency and operations
- Understand the course Case Study

Module 2: Azure Batch Processing Reference Architectures

In this module, the student will learn the reference design and architecture patterns for dealing with the batch processing of data. The student will be exposed to dealing with the movement of data from on-premises systems into a cloud data warehouse and how it can be automated. The student will also be exposed to an AI architecture and how the data platform can integrate with an AI solution.

Module Objectives:

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

- Describe Lambda architectures from a Batch Mode Perspective

- Design an Enterprise BI solution in Azure
- Automate enterprise BI solutions in Azure
- Architect an Enterprise-grade conversational bot in Azure

Module 3: Azure Real-Time Reference Architectures

In this module, the student will learn the reference design and architecture patterns for dealing with streaming data. They will learn how streaming data can be ingested by Event Hubs and Stream Analytics to deliver real-time analysis of data. They will also explore a data science architecture the streams data into Azure Databricks to perform trend analysis. They will finally learn how an Internet of Things (IoT) architecture will require data platform technologies to store data.

Module Objectives:

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

- Lambda architectures for a Real-Time Mode Perspective
- Architect a stream processing pipeline with Azure Stream Analytics
- Design a stream processing pipeline with Azure Databricks.
- Create an Azure IoT reference architecture

Module 4: Security Design Considerations

In this module, the student will learn how to incorporate security into an architecture design and learn the key decision points in Azure provides to help you create a secure environment through all the layers of your architecture.

Module Objectives:

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

- Defense in Depth Security Approach
- Identity Protection
- Infrastructure Protection
- Encryption Usage
- Network Level Protection
- Application Security

Module 5: Designing for Scale and Resiliency

In this module, student will learn scaling services to handle load. They will learn how identifying network bottlenecks and optimizing your storage performance are important to ensure your users have the best experience. They will also learn how to handle infrastructure and service failure, recover from the loss of data, and recover from a disaster by designing availability and recoverability into your architecture.

Module Objectives:

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

- Adjust Workload Capacity by Scaling
- Optimize Network Performance
- Design for Optimized Storage and Database Performance
- Identifying Performance Bottlenecks
- Design a Highly Available Solution
- Incorporate Disaster Recovery into Architectures
- Design Backup and Restore strategies

Module 6: Design for Efficiency and Operations

In this module, students will learn how to design an Azure architecture that is operationally-efficient and minimizes costs by reducing spend, they will understand how to design architectures that eliminates waste and gives them full visibility into what is being utilized in your organizations Azure environment.

Module Objectives:

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

- Maximize the Efficiency of your Cloud Environment
- Use Monitoring and Analytics to Gain Operational Insights
- Use Automation to Reduce Effort and Error

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 fundamentals**[1]
- **DP200: Implementing an Azure Data Solution**[2]

1 https://docs.microsoft.com/en-us/learn/paths/azure-fundamentals/
2 https://www.microsoft.com/en-us/learning/course.aspx?cid=DP-200T01

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 201**[3].

3 https://docs.microsoft.com/en-us/learn/certifications/exams/dp-201

Module 1 Azure Architecture Considerations

Module Introduction

Introduction

Imagine you work for an organization as a Senior Data Engineer. Or you are a Solution Architect or a Consultant. Regardless of the industry, be it retail or a healthcare provider, how do you instill confidence in your customers that their data is secure? Can your architecture handle a spike in traffic if a media report goes viral? Can your architecture handle the failure of one or more critical components? Are you using resources in the most efficient manner possible?

A great architecture helps guide you to design, build, and continuously improve a secure, reliable, and efficient application and data platform. In this module, you will be introduced to the pillars and principles that are essential to a great Azure architecture.

NOTE The concepts discussed in this module are not all-inclusive, but represent some of the important considerations when building a solution on the cloud. Microsoft publishes a broad set of patterns, guidelines, and examples on designing applications on Azure. It is highly recommended that you look through the content in the Azure Architecture Center as you start planning and designing your architecture.

Learning objectives

In this module, you will:

- Identify key pillars for architecting a solution in the cloud
- Design for security
- Design for performance and scalability
- Design for availability and recoverability
- Design for efficiency and operations

Azure Architecture Considerations

Identify the Pillars of a Great Azure Architecture.

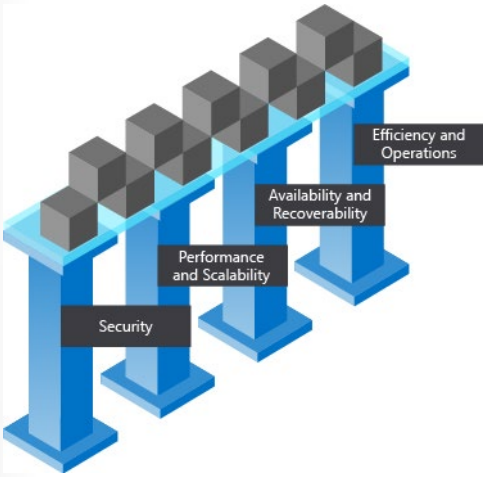
The cloud has changed the way organizations solve their business challenges, and how applications and systems are designed. Your role is not only to deliver business value through the functional requirements of an application and its data, but to ensure the solution is designed in ways that are scalable, resilient, efficient and secure. Solution architecture is concerned with the planning, design, implementation, and ongoing improvement of a technology system. The architecture of a system must balance and align the business requirements with the technical capabilities needed to execute those requirements. It includes an evaluation of risk, cost, and capability throughout the system and its components.

Design a great Azure Architecture

While there is no one-size-fits-all approach to designing a platform architecture, there are some universal concepts that will apply regardless of the architecture, technology, or cloud provider. While these are not all-inclusive, focusing on these concepts will help you build a reliable, secure, and flexible foundation for your application.

A great architecture starts with a solid foundation built on four pillars:

- Security
- Performance and scalability
- Availability and recoverability
- Efficiency and operations



Security

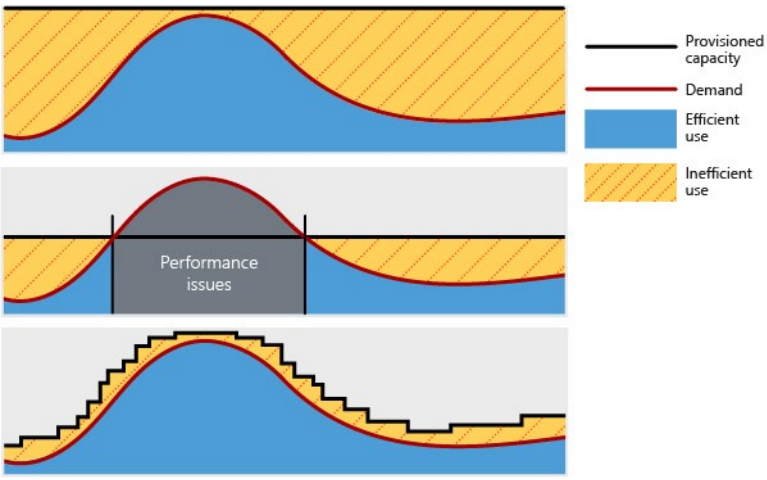
Data is the most valuable piece of your organization's technical footprint. In this pillar, you'll be focused on securing access to your architecture through authentication and protecting your application and data from network vulnerabilities. The integrity of your data should be protected as well, using tools like encryption.

You must think about security throughout the entire lifecycle of your application, from design and implementation to deployment and operations. The cloud provides protections against a variety of

threats, such as network intrusion and Dynamic Denial of Service (DDoS) attacks, but you still need to build security into your application, processes, and organizational culture.

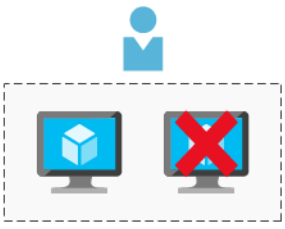
Performance and scalability

For an architecture to perform well and be scalable, it should properly match resource capacity to demand. Traditionally, cloud architectures do so by scaling applications dynamically based on activity in the application, or scaling data platforms during periods of large data loads. Demand for services change, so it's important for your architecture to have the ability to adjust to demand as well. By designing your architecture with performance and scalability in mind, you'll provide a great experience for your customers while being cost-effective.



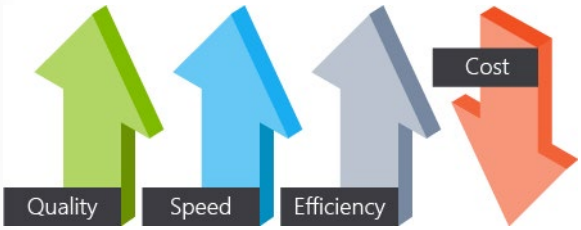
Availability and recoverability

Every architect's worst fear is having your architecture go down with no way to recover it. A successful cloud environment is designed in a way that anticipates failure at all levels. Both planned and un-planned. Part of anticipating these failures is designing a system that can recover from the failure, within the time required by your stakeholders and customers.



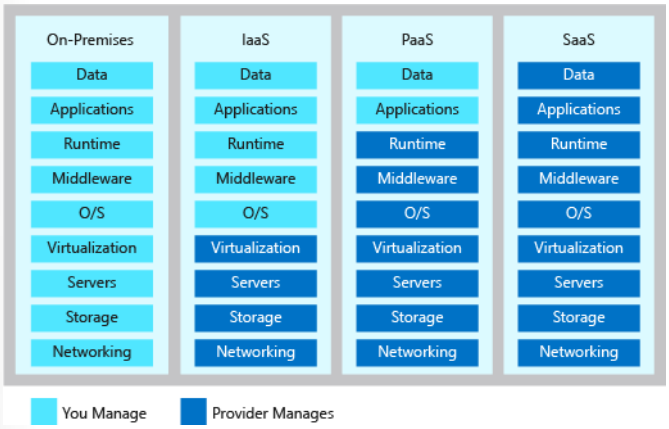
Efficiency and operations

You will want to design your cloud environment so that it's cost-effective to operate and develop against. Inefficiency and waste in cloud spending should be identified to ensure you're spending money where you can make the greatest use of it. You need to have a good monitoring architecture in place so that you can detect failures and problems before they happen or, at a minimum, before your customers notice. You also need to have some visibility in to how your application is using its available resources, through a robust monitoring framework.



Shared responsibility

Moving to the cloud introduces a model of shared responsibility. In this model, your cloud provider will manage certain aspects of your application, leaving you with the remaining responsibility. In an on-premises environment you are responsible for everything. As you move to infrastructure as a service (IaaS), then to platform as a service (PaaS) and software as a service (SaaS), your cloud provider will take on more of this responsibility. This shared responsibility will play a role in your architectural decisions, as they can have implications on cost, operational capabilities, security, and the technical capabilities of your application. By shifting these responsibilities to your provider you can focus on bringing value to your business and move away from activities that aren't a core business function.



Design choices

In an ideal architecture, we would build the most secure, high performance, highly available, and efficient environment possible. However, as with everything, there are trade-offs. To build an environment with the highest level of all these pillars, there is a cost. That cost may be in actual money, time to deliver, or operational agility. Every organization will have different priorities that will impact the design choices made in each pillar. As you design your architecture, you will need to determine what trade-offs are acceptable and which are not.

When building an Azure architecture, there are many considerations to keep in mind. You want your architecture to be secure, scalable, available, and recoverable. To make that possible, you'll have to make decisions based on cost, organizational priorities, and risk.

Design for Security

Your organization stores personal and potentially sensitive client data. A security incident could expose this sensitive data, which could cause personal embarrassment or financial harm. How do you ensure the integrity of their data and ensure your systems are secure?

Here, we'll talk about how to approach the security of an architecture.

What should I protect?

The data your organization stores or handles is at the heart of your securable assets. This data could be sensitive data about customers, financial information about your organization, or critical line-of-business data supporting your organization. Along with data, securing the infrastructure it exists on, and the identities we use to access it, are also critically important.

Your data may be subject to additional legal and regulatory requirements depending on where you are located, the type of data you are storing, or the industry that your application operates in. For instance, in the healthcare industry in the US, there is a law called the Health Insurance Portability and Accountability Act (HIPAA). In the financial industry, the Payment Card Industry Data Security Standard is concerned with the handling of credit card data. Organizations that store data that is in scope for these laws and standards are required to ensure certain safeguards are in place for the protection of this data. In Europe, the General Data Protection Regulation (GDPR) lays out the rules of how personal data is protected, and defines individuals' rights related to stored data. Some countries require that certain types of data do not leave their borders.

When a security breach occurs, there can be substantial impacts to the finances and reputation of both organizations and customers. This breaks down the trust customers are willing to instill in your organization, and can impact its long-term health.

Defense in depth

A multilayered approach to securing your environment will increase the security posture of your environment. Commonly known as *defense in depth*, we can break down the layers as follows:

- Data
- Applications
- VM/compute
- Networking
- Perimeter
- Policies & access
- Physical security

Each layer focuses on a different area where attacks can happen and creates a depth of protection, should one layer fail or be bypassed by an attacker. If we were to just focus on one layer, an attacker would have unfettered access to your environment should they get through this layer. Addressing security in layers increases the work an attacker must do to gain access to your systems and data. Each layer will have different security controls, technologies, and capabilities that will apply. When identifying

the protections to put in place, cost will often be of concern, and will need to be balanced with business requirements and overall risk to the business.

Defense in-depth security layers

There is no single security system, control, or technology that will fully protect your architecture. Security is more than just technology, it's also about people and processes. Creating an environment that looks holistically at security, and making it a requirement by default will help ensure your organization is as secure as possible.

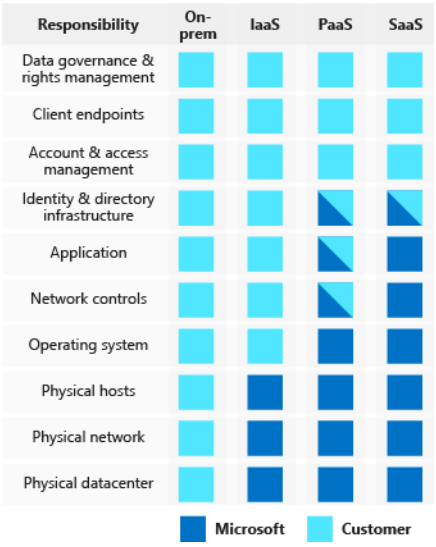
Common attacks

At each layer, there are some common attacks that you will want to protect against. These are not all-inclusive, but can give you an idea of how each layer can be attacked and what types of protections you may need to look at.

- **Data layer**: Exposing an encryption key or using weak encryption can leave your data vulnerable should unauthorized access occur.
- **Application layer**: Malicious code injection and execution are the hallmarks of application-layer attacks. Common attacks include SQL injection and cross-site scripting (XSS).
- **VM/compute layer**: Malware is a common method of attacking an environment, which involves executing malicious code to compromise a system. Once malware is present on a system, further attacks leading to credential exposure and lateral movement throughout the environment can occur.
- **Networking layer**: Unnecessary open ports to the Internet are a common method of attack. These could include leaving SSH or RDP open to virtual machines. When open, these could allow brute-force attacks against your systems as attackers attempt to gain access.
- **Perimeter layer**: Denial-of-service (DoS) attacks are often seen at this layer. These attacks attempt to overwhelm network resources, forcing them to go offline or making them incapable of responding to legitimate requests.
- **Policies & access layer**: This is where authentication occurs for your application. This could include modern authentication protocols such as OpenID Connect, OAuth, or Kerberos-based authentication such as Active Directory. Exposed credentials are a risk here and it's important to limit the permissions of identities. We also want to have monitoring in place to look for possible compromised accounts, such as logins coming from unusual places.
- **Physical layer**: Unauthorized access to facilities through methods such as door drafting and theft of security badges can be seen at this layer.

Shared security responsibility

Revisiting the model of shared responsibility, we can reframe this in the context of security. Depending on the type of service you select, some security protections will be built in to the service, while others will remain your responsibility. Careful evaluation of the services and technologies you select will be necessary, to ensure you are providing the proper security controls for your architecture.



Design for Performance and Scalability

Imagine a news story has just been published covering your organization's breakthrough. Such a milestone will undoubtedly bring a large influx of traffic to your website to purchase your goods or services, or just to seek out information. Will the website handle this traffic increase, or will the load cause the site to be slow or unresponsive?

Here, we'll look at some of the basic principles of ensuring outstanding application performance using scaling and optimization principles.

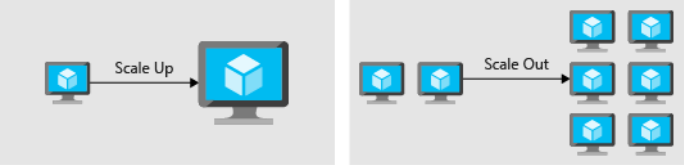
What is scaling and performance optimization?

Scaling and performance optimization are about matching the resources available to an application with the demand it is receiving. Performance optimization includes scaling resources, identifying and optimizing potential bottlenecks, and optimizing your application code and database access for peak performance.

Scaling

Compute resources can be scaled in two different directions:

- Scaling *up* is the action of adding more resources to a single instance.
- Scaling *out* is the addition of instances.



Scaling up is concerned with adding more resources, such as CPU or memory, to a single instance. This instance could be a virtual machine or a PaaS service. The act of adding more capacity to the instance increases the resources available to your application, but it does come with a limit. Virtual machines are limited to the capacity of the host they run on, and hosts themselves have physical limitations. Eventually, when you scale up an instance, you can run into these limits, restricting your ability to add further resources to the instance.

Scaling out is concerned with adding additional instances to a service. These can be virtual machines or PaaS services, but instead of adding more capacity by making a single instance more powerful, we add capacity by increasing the overall total number of instances. The advantage of scaling out is that you can conceivably scale out forever if you have more machines to add to the architecture. Scaling out requires some type of load distribution. This could be in the form of a load balancer distributing requests across available servers, or a service discovery mechanism for identifying active servers to send requests to.

In both cases, resources can be reduced, bringing cost optimization into the picture.

Performance optimization

When optimizing for performance, you'll look at network and storage to ensure performance is acceptable. Both can impact the response time of your application and databases. Selecting the right networking and storage technologies for your architecture will help you ensure you're providing the best experience for your consumers.

Performance optimization will also include understanding how the applications themselves are performing. Errors, poorly performing code, and bottlenecks in dependent systems can all be uncovered through an application performance management tool. Often, these issues may be hidden or obfuscated for end users, developers, and administrators, but can have adverse impact on the overall performance of your application.

Scalability and performance patterns and practices

Let's take a look at some patterns and practices that can be leveraged to enhance the scalability and performance of your application.

Data partitioning

In many large-scale solutions, data is divided into separate partitions that can be managed and accessed separately. The partitioning strategy must be chosen carefully to maximize the benefits while minimizing adverse effects. Partitioning can help improve scalability, reduce contention, and optimize performance.

Implement scale units

Scale as a unit. For each resource, determine the impact that a scaling activity may have on dependent systems. This makes applying scale-out operations easier, and less prone to negative impact on the application. For example, adding x number of web and worker roles might require y number of additional queues and z number of storage accounts to handle the additional workload generated by the roles. A

scale unit could consist of x web and worker roles, y queues, and z storage accounts. Design the application so that it's easily scaled by adding one or more scale units.

Autoscaling

Autoscaling is the process of dynamically allocating resources to match performance requirements. As the volume of work grows, an application may need additional resources to maintain the desired performance levels and satisfy service-level agreements (SLAs). As demand slackens and the additional resources are no longer needed, they can be de-allocated to minimize costs.

Autoscaling takes advantage of the elasticity of cloud-hosted environments while easing management overhead. It reduces the need for an operator to continually monitor the performance of a system and make decisions about adding or removing resources.

Caching

Use caching in your architecture can help improve performance. Caching is a mechanism to store frequently used data or assets (web pages, images) for faster retrieval. Caching can be used at different layers of your application. You can use caching between your application servers and a database, to decrease data retrieval times. You could also use caching between your end users and your web servers, placing static content closer to the user and decreasing the time it takes to return web pages to the end user. This also has a secondary effect of offloading requests from your database or web servers, increasing the performance for other requests.

Performance monitoring

Distributed applications and services running in the cloud are, by their nature, complex pieces of software that comprise many moving parts. In a production environment, it's important to be able to track the way in which users utilize your system, trace resource utilization, and generally monitor the health and performance of your system. You can use this information as a diagnostic aid to detect and correct issues, and also to help spot potential problems and prevent them from occurring.

Look across all layers of your application and identify and remediate performance bottlenecks in your application. These bottlenecks could be poor memory handling in your application, or even the process of adding indexes into your database. It may be an iterative process as you relieve one bottleneck and then uncover another that you were unaware of.

With a thorough approach to performance monitoring, you'll be able to determine what types of patterns and practices your architecture will benefit from.

Design for Availability and Recoverability

Your organization has little tolerance for downtime. They need to have access to IT systems around the clock to ensure they're providing the highest quality of service at all times. To meet the around-the-clock demands, the organizations applications and databases must be able to handle failures with minimal impact to their users. How do they ensure their applications remain operational, both for localized incidents and large-scale disasters?

Here, we'll talk through how to include availability and recoverability in your architecture design.

Availability and recoverability

In a complex application, any number of things can go wrong at any scale. Individual servers and hard drives can fail. A deployment issue could unintentionally drop all tables in a database. Whole datacenters may become unreachable. A ransomware incident could maliciously encrypt all your data.

Designing for *availability* focuses on maintaining uptime through small-scale incidents and temporary conditions like partial network outages. You can ensure your application can handle localized failures by integrating high availability into each component of an application architecture and eliminating single points of failure, such as in the database layer. Such a design also minimizes the impact of infrastructure maintenance. High-availability designs typically aim to eliminate the impact of incidents quickly and automatically and ensure that the system can continue to process requests with little to no impact.

Designing for *recoverability* focuses on recovery from data loss and from larger scale disasters. Recovery from these types of incidents often involves active intervention, though automated recovery steps can reduce the time needed to recover. These types of incidents may result in some amount of downtime or permanently lost data. Disaster recovery is as much about careful planning as it is about execution.

Including high availability and recoverability in the design of your architecture protects your business from financial losses resulting from downtime and lost data. They ensure your reputation isn't negatively impacted by a loss of trust from your customers.

Architecting for availability and recoverability

Architecting for availability and recoverability ensures that your application and databases can meet the commitments you make to your customers.

For availability, identify the service-level agreement (SLA) you're committing to. Examine the potential high-availability capabilities of your application relative to your SLA, and identify where you have proper coverage and where you'll need to make improvements. The goal is to add redundancy to components of the architecture so that you are less likely to experience an outage. Examples of high-availability design components include clustering and load balancing. Clustering replaces a single VM with a set of coordinated VMs. When one VM fails or becomes unreachable, services can fail over to another one that can service the requests. Load balancing spreads requests across many instances of a service, detecting failed instances and preventing requests from being routed to them.

For recoverability, perform an analysis that examines possible data loss and major downtime scenarios. The analysis should include an exploration of recovery strategies and the cost-benefit tradeoff for each. This exercise will give you important insight into your organization's priorities and help clarify the role of your application. The results should include the application's and/or database recovery point objective (RPO) and recovery time objective (RTO).

- **Recovery Point Objective**: The maximum duration of acceptable data loss. RPO is measured in units of time, not volume: “30 minutes of data”, "four hours of data", and so on. RPO is about limiting and recovering from data *loss*, not data *theft*.
- **Recovery Time Objective**: The maximum duration of acceptable downtime, where “downtime” needs to be defined by your specification. For example, if the acceptable downtime duration is eight hours in the event of a disaster, then your RTO is eight hours.

With RPO and RTO defined, you can design backup, restore, replication, and recovery capabilities into your architecture to meet these objectives.

Every cloud provider offers a suite of services and features you can use to improve your application's availability and recoverability. When possible, use existing services and best practices, and try to resist creating your own.

Hard drives can fail, datacenters can become unreachable, and hackers can attack. It's important that you maintain a good reputation with your customers using availability and recoverability. Availability focuses on maintaining uptime through conditions like network outages, and recoverability focuses on retrieving data after a disaster.

Design for Efficiency and Operations

Your organization has moved a majority of their systems to the cloud, but are now seeing cost increases in areas they didn't expect. After some observation, you realize that your organization is inefficient across your environment, and are still doing manual operational work.

Here, we'll take a look at some ways they can eliminate waste and improve operational efficiencies.

Importance of efficiency and operations

Efficiency is focused on identifying and eliminating waste within your environment. The cloud is a pay-as-you-go service and waste typically comes from provisioning more capacity than demand requires. There are operational costs that go along with this as well. These operational costs show up as wasted time and increased error. Focusing on these as you design your architecture will help you identify and eliminate waste across your environment.

Waste can show up in several ways. Let's look at a few examples:

- An Azure Synapse Analytics/Cosmos DB provisioned with too much SQL Pools/RTU than what is required
- A virtual machine that is always 90% idle
- Paying for a license included in a virtual machine when a license is already owned
- Retaining infrequently accessed data on a storage medium optimized for frequent access
- Manually repeating the build of a non-production environment

In each of these cases, more money is being spent than it should, and each presents an opportunity for cost reduction.

Operationally, it's important to have a robust monitoring strategy. This helps you identify areas of waste, troubleshoot issues, and optimize the performance of your application. A multilayered approach is essential. Gathering data points from components at every layer will let you alert when values are outside of acceptable values and track spending over time.

Efficiency best practices

There are several ways to optimize your environment and maximize efficiency. A great place to start is to look at cost optimization steps like sizing data services or virtual machines properly and deallocating compute that aren't in use. Now that you are paying for what you use, you want to be sure that you aren't wasting any of these resources.

Where possible, you want to move from IaaS to PaaS services. PaaS services typically cost less than IaaS, and typically bring reduced operational costs along as well. With PaaS services, you don’t have to worry about patching or maintaining VMs because those activities are typically handled by the cloud provider. Not all applications can be moved to PaaS, but with the cost savings that are there with PaaS services, it's certainly something worth considering.

Operational best practices

Automate as much as possible. The human element is costly, injecting time and error into operational activities. You can use automation to build, deploy, and administer resources. By automating common activities, you can eliminate the delay in waiting for a human to intervene.

Throughout your architecture, you want to have a thorough monitoring, logging, and instrumentation system. This will always give you the capability to see what's going on. It will ensure you know when something is not right before your users are impacted. With a comprehensive approach, you'll be able to identify performance issues, cost inefficiencies, correlate events, and gain a greater ability to troubleshoot issues.

Lastly, modern architectures should be designed with DevOps and continuous integration in mind. This will give you the ability to automate deployments using infrastructure as code, automate application testing, and build new environments as needed. DevOps is as much cultural as it is technical, but can bring many benefits to organizations that embrace it.

Efficiency is about identifying and eliminating waste within your environment. With cloud computing, you're paying for what you use, so you want to make sure you aren't wasting resources. To help ensure efficiency, try to automate as much as possible, move systems from IaaS to PaaS, and design your architectures with DevOps in mind.

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 fradulent 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 innovate 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 hours
- Bicycle Locked / Unlocked
- Bicycle parts and components information (on electrical bicycles)

First party and 3rd party applications can have access 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.

Module Summary

Summary

Architecture is the foundation of your application's design. A great architecture will give you the confidence that your databases and/or applications can sustainably meet the needs of your customers both now and in the future.

The architectural priorities and needs of every solution are different, but the four pillars of architecture are an excellent guidepost you can use to make sure that you have given enough attention to every aspect of your application:

- **Security**: Safeguarding access and data integrity and meeting regulatory requirements
- **Performance and scalability**: Efficiently meeting demand in every scenario
- **Availability and recoverability**: Minimizing downtime and avoiding permanent data loss
- **Efficiency and operations**: Maximizing maintainability and ensuring requirements are met with monitoring

Focusing on these pillars when designing your architecture will ensure you're laying a solid foundation for your applications in the cloud. With a solid foundation, you'll be able to drive innovation through your environment, build solutions that your users will love, and foster the trust of your customers.

Module 2 Azure Batch Processing Reference Architectures

Module Introduction

Azure Batch Processing Reference Architectures

There are a number of examples of how cloud solutions in the Data and AI space are taking shape in numerous industries. In this module you will explore some of the typical architecture patterns that are used to fulfill different types of batch mode operation including the population and automation of a business intelligence solution to facilitating data storage architectures for AI solutions including Bots.

NOTE The concepts discussed in this module are not all-inclusive, but represent some of the important considerations when building a solution on the cloud. Microsoft publishes a broad set of patterns, guidelines, and examples on designing applications on Azure. It is highly recommended that you look through the content in the Azure Architecture Center as you start planning and designing your architecture.

Learning objectives

In this module, you will learn:

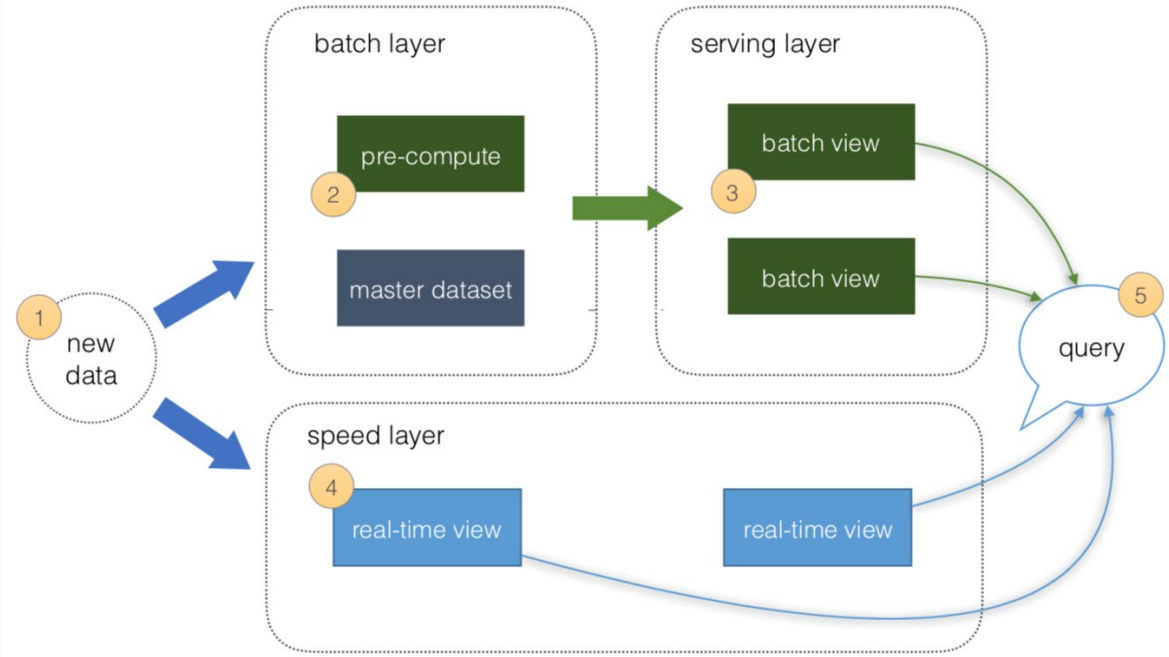
- Lambda architectures from a batch mode perspective
- Design an enterprise business intelligence solution
- Automate an enterprise business intelligence solution
- Architect an enterprise conversational bot

Azure Batch Processing Reference Architectures

Lambda Architecure from a Batch Mode Perspective

The Lambda architecture sets out a framework for data processing from both a speed and batch mode perspective. The advent and advances of cloud technologies has meant that many organizations are starting to derive actionable information from data almost immediately, or at least in a way that means that the data is not stale. Furthermore, a greater support for a wider range of data types has also meant that businesses are able to derive value from data that was once very difficult to extract information from.

Coupled with the advent of big data technologies, a shift is happening from the conventional data warehouse practice to cloud based data processing & management capabilities where high volume batch processing is possible at the optimized cost.



Lambda Architecture - Batch Layer

The Batch layer is used to pre-compute results on large volumes of data using a distributed data system. As this data is not being processed in real-time, the objective is to ensure that the results computed are as accurate as possible and updated on a regular schedule. This gives the opportunity to fix any errors before presenting a computed dataset, or recompute the data-set based on a refresh of the data.

There are a range of Azure technologies that can be used to fulfill the batch layer of a Lambda Architecture including

- Azure Databricks
- Azure Synapse Analytics
- Azure Cosmos DB

- Azure Data Lake Storage Gen II
- HDInsight

Lambda Architecture - Serving Layer

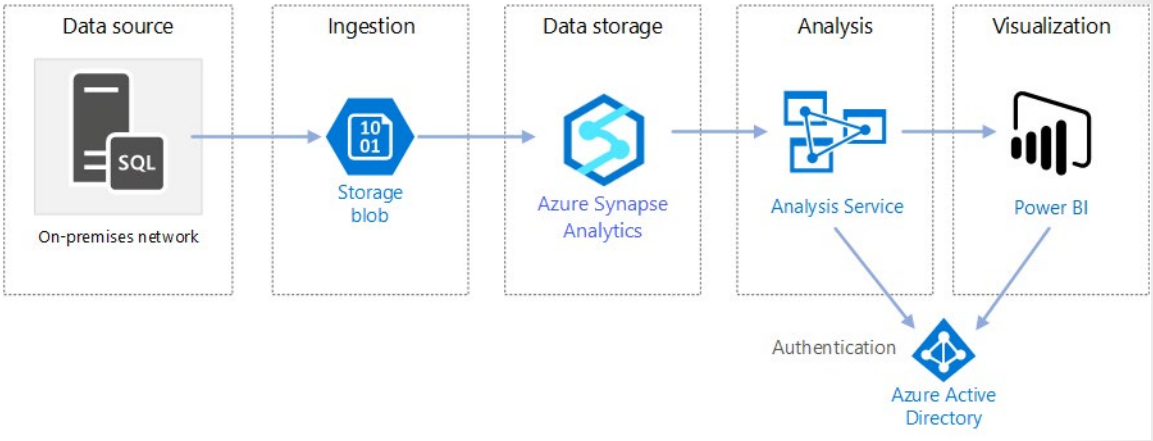
The serving layer acts as the storage output of either the Batch or Speed layer that is used by client applications to access the results of the data-sets. This enable ad-hoc queries to be issued against the Serving layer without negatively impacting either the Batch or Speed layer.

There are a range of technologies that can act as a Serving layer for a Lambda architecture including:

- Azure Analysis Services
- Power BI

Design an Enterprise Business Intelligence Architecture

In this architecture, data is moved from an on-premises data store into the cloud data store in a one-off of data transfer. This can be used by organizations that would like to transfer on-premises data to take advantage of the scale and cost controls that are available from Azure. In this lesson we will explore the architecture and data pipelines that would be setup up to enable this architecture



Data Source

In this example, the data is located in an OTP databases hosted on an on-premises SQL Server.

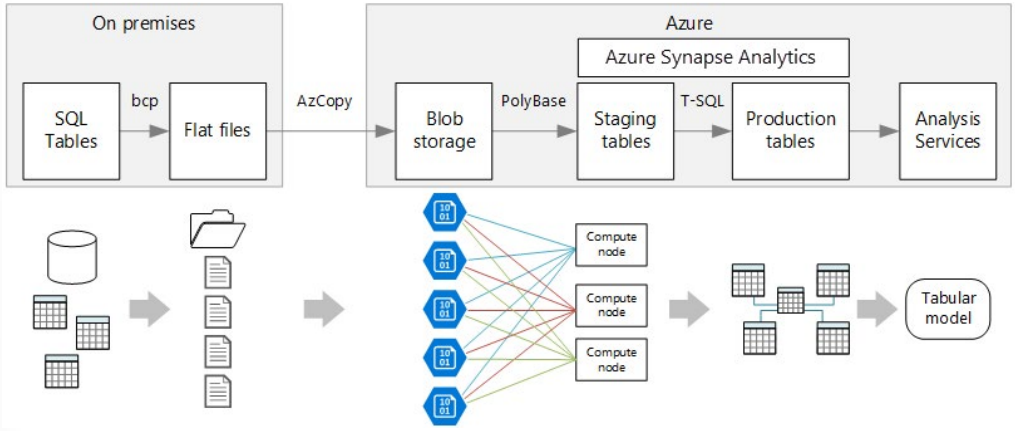
Ingestion and Data Storage

Azure Blob Storage is used as an intermediate staging area for the the OLTP data from the on-premises SQL Server. Whilst it is possible to transfer data directly from SQL Server to Azure Synapse Analytics, this would only work well for very small volumes of data.

Important: You can replace the staging area with Azure Data Lake Storage Gen II that can act as a raw data source store for Azure Databricks and HDInsight as well as Azure Synapse Analytics

The best approach would be to use Bulk Copy Program (bcp) to transfer the data from SQL Server into flat files in folders and then use AZCopy to transfer the data to Azure BLob Store. You can then use

PolyBase to perform a parallelized data load into staging tables in Azure Synapse Analytics, using Transact-SQL to transform the data into the final data warehouse tables.



Analysis and Visualization

Use Analysis Services to create a semantic model that users can query. Analysis Services is especially useful in a BI dashboard scenario. In this architecture, Analysis Services reads data from the data warehouse to process the semantic model, and efficiently serves dashboard queries. It also supports elastic concurrency, by scaling out replicas for faster query processing. Power BI is a suite of business analytics tools to analyze data for business insights. In this architecture, it queries the semantic model stored in Analysis Services.

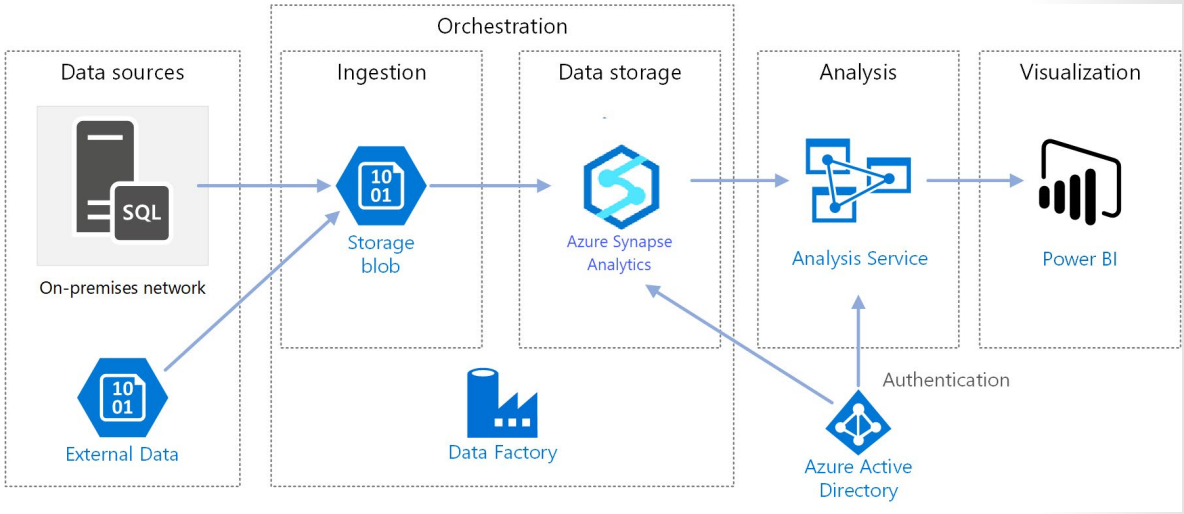
Authentication

Azure Active Directory (Azure AD) authenticates users who connect to the Analysis Services server through Power BI.

This architecture is designed for one-time execution. If you need to move data continually, it is recommended using Azure Data Factory to define an automated workflow.

Automate an Enterprise Business Intelligence Architecture

This architecture expands on the previous solution by introducing automation to the data pipelines using Azure Data Factory. This enables an organization to perform incremental data loads into an Azure Synapse Analytics using an Extract, Load and Transform (ELT) pipeline.



Data Source

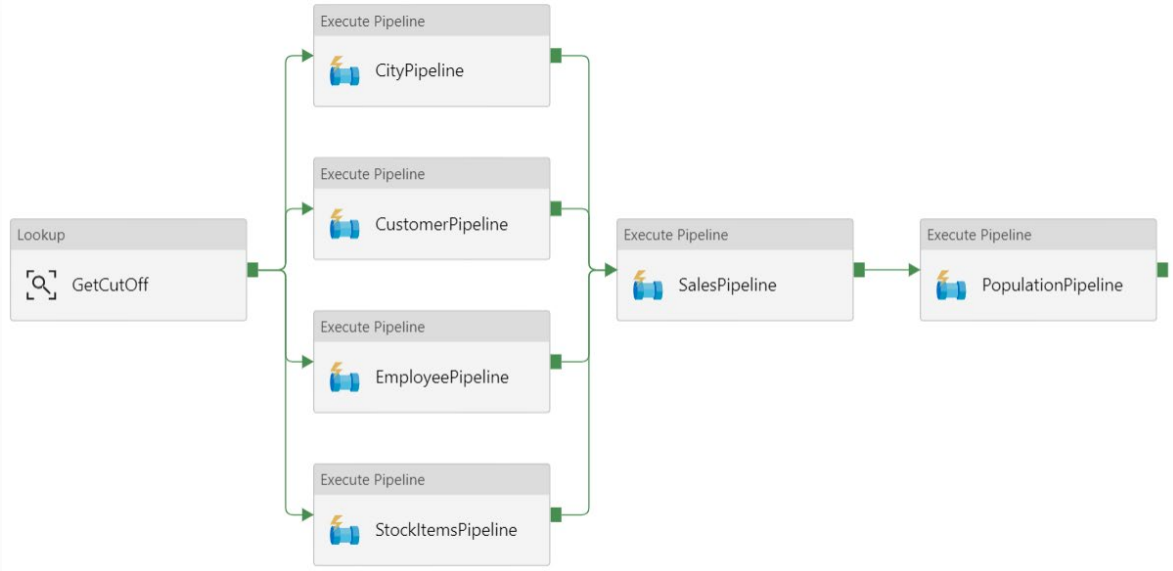
In this example, the data is located in an OTP databases hosted on an on-premises SQL Server. In addition, there are additional data stores that are used to collect data from. This is not uncommon in an Enterprise Business Intelligence Architecture, where data can come form other on-premises data stores or from external reference data such as weather or population data.

Ingestion and Data Storage

Azure Blob Storage (Data Lake Storage Gen II could be used instead) is again used as an intermediate staging area for both the OLTP and external data. Azure Synapse Analytics is used as the final destination for the data storage.

Azure Data Factory is a managed service that orchestrates and automates data movement and data transformation. In this architecture, it coordinates the various stages of the ELT process using pipelines to cleanse and load the data. A pipeline is a logical grouping of activities used to coordinate a task — in this case, loading and transforming data into Azure Synapse Analytics. This reference architecture defines a master pipeline that runs a sequence of child pipelines. Each child pipeline loads data into one or more data warehouse tables incrementally.

To perform an incremental load, you need a way to identify which data has changed. The most common approach is to use a high water mark value, which means tracking the latest value of some column in the source table, either a datetime column or a unique integer column.



Analysis and Visualization

Use Analysis Services to create a semantic model that users can query. Analysis Services is especially useful in a BI dashboard scenario. In this architecture, Analysis Services reads data from the data warehouse to process the semantic model, and efficiently serves dashboard queries. It also supports elastic concurrency, by scaling out replicas for faster query processing. Power BI is a suite of business analytics tools to analyze data for business insights. In this architecture, it queries the semantic model stored in Analysis Services.

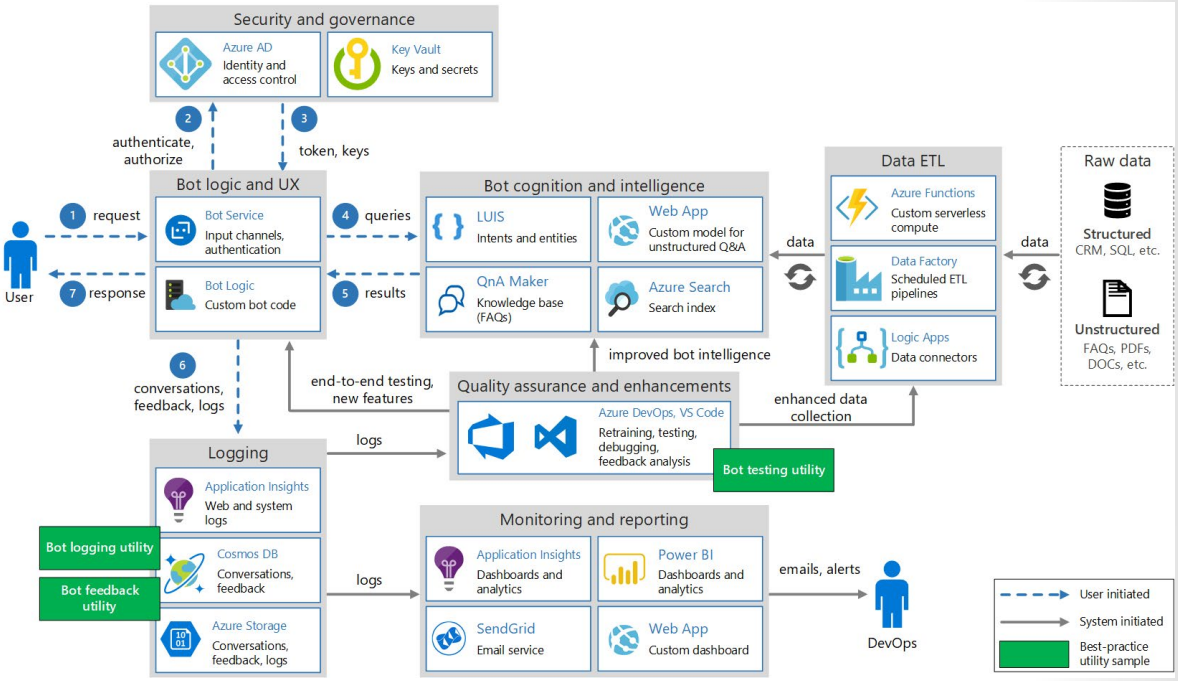
Authentication

Azure Active Directory (Azure AD) authenticates users who connect to the Analysis Services server through Power BI.

Architect an Enterprise Conversational Bot

This architecture is complicated. But the majority of this architecture is for an AI Engineer to design and build. In this instance, we are interested in the data components that we may be asked to support the AI Engineer with. Namely:

1. Creating a data store to hold bot logging data. (See the Logging section of the architecture)
2. Creating a data store to hold Cognitive Services metadata. (See the Data ETL of the architecture)



Bot Logging Data

By default, the Bot Framework will not perform any conversation logging for you automatically. This has privacy implications, and many bots simply can't allow that in their scenarios. In the advanced analytics, there are plenty of uses for storing log conversations. Having a corpus of chat conversations can allow developers to:

1. Build question and answer engines specific to a domain.
2. Determine if a bot is responding in the expected manner.
3. Perform text or sentiment analysis on specific topics or products to identify trends or patterns.

The Bot Framework SDK enables AI Engineers to create custom middleware to intercept and log messages as part of the middleware pipeline. Specifically, you can setup a Cosmos DB instance, and create middleware to store "entries" as users interact with the bot. Each entry will include a date/time stamp, the message the user sent, and the reply/replies that the bot sent. You can then use the capabilities of Cosmos DB to replicate this data globally if required. If planetary scale is not required, the you can simply use Azure Blob Store/Data Lake Store Gen II should you wish to use this as a Machine Learning source of data for text analytics.

Cognitive Services Metadata

A Bots intelligence is primarily provided by the capabilities of Cognitive Services. Typically data will be sent to a Cognitive Services API, which may return back data, sometimes in the form of metadata. To provide an example, should you use the Computer Vision API, the data that is sent to the API is an image. The data that is returned, will be Tag and Description data that is used to explain the content of an image.

You may want this data to be stored; in addition, you may also want to place an Azure Index on top of this data to speed up searches for the bot. For example, you may have a bot that has the ability to take a picture and provide a tag and description of the image. Rather than constantly making a call to the same API when the same image is uploaded, you could store the image in Azure Blob Store, and then store the

metadata returned in an Azure Cosmos DB instance to make the metadata available globally. An Azure Search can then be created on top of the Cosmos DB metadata, which the bot can then use to search for images based on a search og Tags or description data, this make the bot appear to be intelligent

Module Summary

Summary

Batch mode processing of data is well understood by many data professionals as it has been the most common approach to provisioning data analytics with on-premises data for a number of decades. This capability is available in the cloud providing data engineers with the ability to batch process both structured and semi-structured data. Azure Synapse Analytics enables organizations to realize the benefits of Azure through the migration of on-premises data warehouses to Azure, and organizations can start to see value from Big Data technologies using Data Lake Store and Databricks.

In this module, you have explored a number of batch mode architecture patterns that are common within a range of industries that could be applied to your organization including:

- Lambda architectures from a batch mode perspective
- Design an enterprise business intelligence solution
- Automate an enterprise business intelligence solution
- Architect an enterprise conversational bot

Module 3 Azure Real Time Processing Reference Architectures

Module Introduction

Azure Real-Time Processing Reference Architectures

Real time processing is becoming a more popular avenue for processing data as the opportunity becomes more cost-effective and accessible to more organizations. Azure provides a range of technologies that can be used to ingest data streams to perform real-time descriptive ort predictive analytics.

NOTE The concepts discussed in this module are not all-inclusive, but represent some of the important considerations when building a solution on the cloud. Microsoft publishes a broad set of patterns, guidelines, and examples on designing applications on Azure. It is highly recommended that you look through the content in the Azure Architecture Center as you start planning and designing your architecture.

Learning objectives

In this module, you will learn:

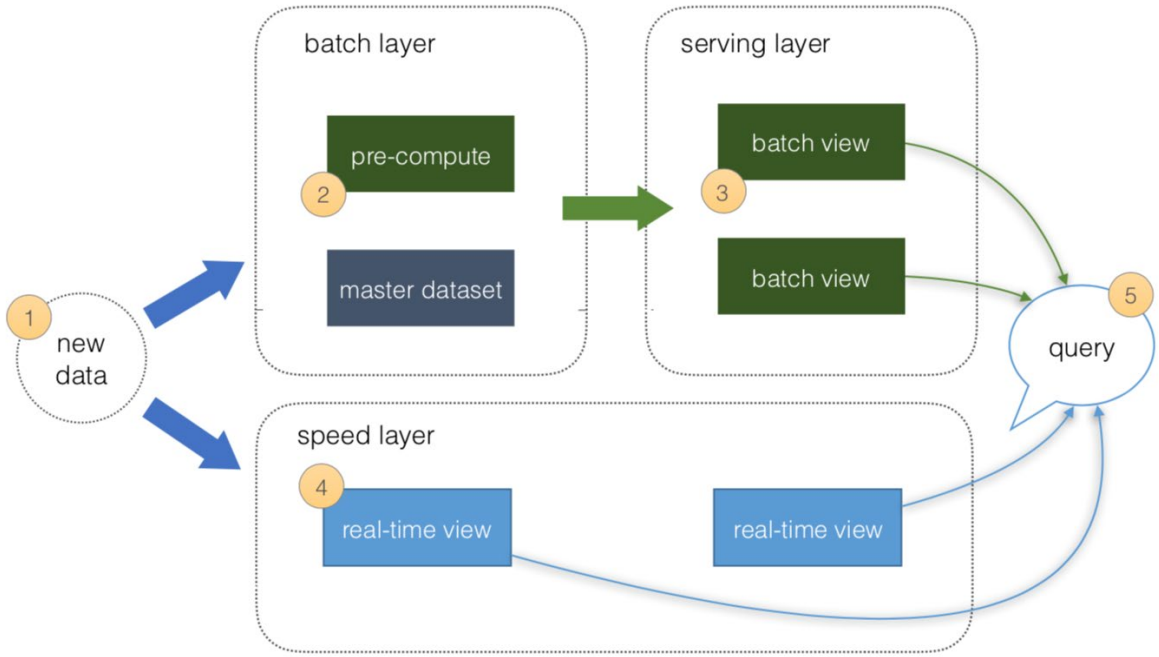
- Lambda architectures for a Real-Time Mode Perspective
- Architect a stream processing pipeline with Azure Stream Analytics
- Design a stream processing pipeline with Azure Databricks
- Create an Azure IoT reference architecture

Azure Real Time Processing Reference Architectures

Lambda architectures from a real-time mode processing perspective

The Lambda architecture sets out a framework for data processing from both a speed and batch mode perspective. The advent and advances of cloud technologies has meant that many organizations are starting to derive actionable information from data almost immediately, or at least in a way that means that the data is not stale. Furthermore, a greater support for a wider range of data types has also meant that businesses are able to derive value from data that was once very difficult to extract information from.

Coupled with the advent of big data technologies, a shift is happening from the conventional data warehouse practice to cloud based data processing & management capabilities where high volume batch processing is possible at the optimized cost.



Lambda Architecture - Speed Layer

The Speed layer processes data streams in real or near real time. This works well when the aim is to minimize the latency of the data ingestion to analysis. However, the cost of this is that there are no opportunities to clean or fix the data. It enables the organization to get a real time view of the data and can be used to fill in the gap of information until the batch data is processed.

There are a range of Azure technologies that can be used to fulfill the batch layer of a Lambda Architecture including

- Azure Stream Analytics
- Azure Databricks with Kafka
- Azure Kafka or Storm on HDInsight

Lambda Architecture - Serving Layer

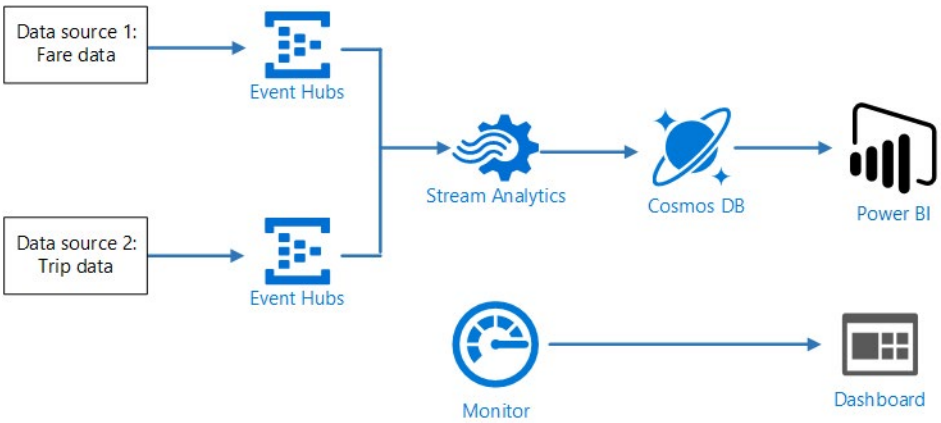
The serving layer is an optional layer for lambda real-time architectures and acts as the storage output of either the Batch or Speed layer that is used by client applications to access the results of the data-sets. This enable ad-hoc queries to be issued against the Serving layer without negatively impacting either the Batch or Speed layer.

There are a range of technologies that can act as a Serving layer for a Lambda architecture including:

- Azure Analysis Services
- Power BI

Architecting a Stream Processing Pipeline with Stream Analytics

This architecture shows an end-to-end stream processing pipeline. This is useful in use cases where an organization wants to ingest data from social media or custom application and analyze the data in near real time. In the example below, the pipeline ingests data from two sources, correlates records in the two streams, and calculates a rolling average across a time window. It is designed to simulate the a taxi company collects data about each taxi trip.



Data Source

In this architecture, there are two data sources that generate data streams in real time. The first stream contains ride information, and the second contains fare information. This is generated in real time by an application within the taxi cab.

Ingestion and Data Storage

Each data source sends a stream of data to an associated Azure Event Hub. Azure Event Hubs is a Big Data streaming platform and event ingestion service, capable of receiving and processing millions of events per second. it is one of three supported data sources for Azure Stream Analytics. The others being IoT Hub and Azure Blob Store.

Analysis and Visualization

The analysis of the streaming data is performed with Stream Analytics itself. A Stream Analytics job is created that contains the Event Hubs as input streams, with a query that performs stream processing of the two event hubs to correlate the records in the two data streams. An output is defined to Cosmos DB that stores the correlated results written as JSON documents to a Cosmos DB document database.

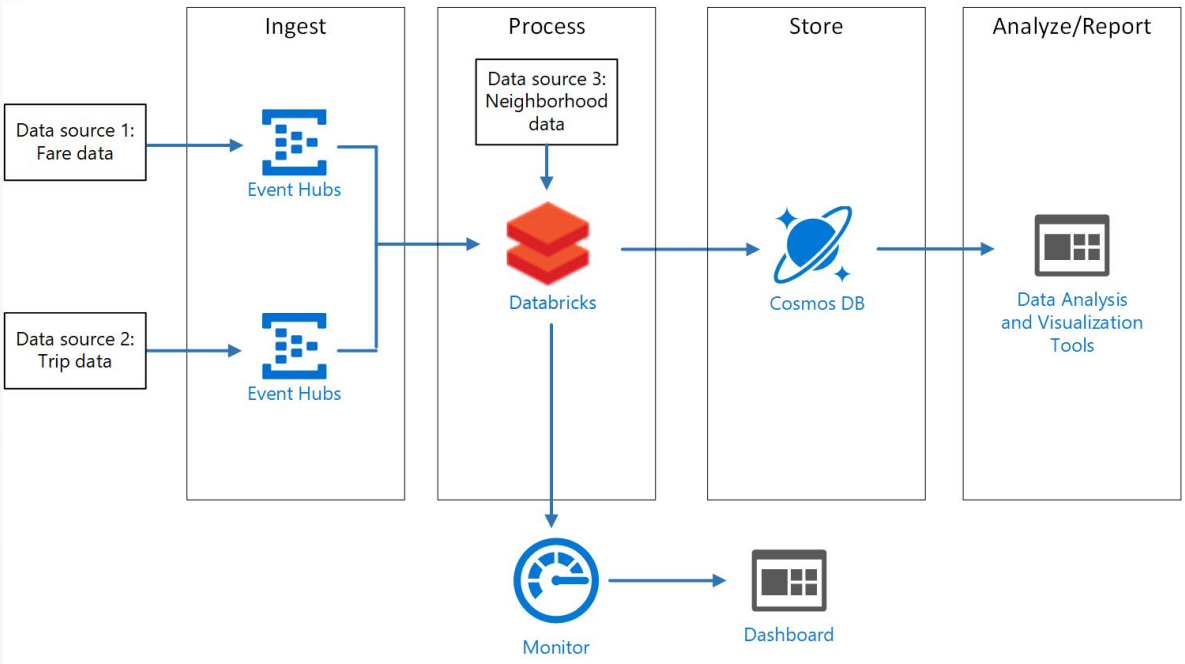
Power BI then uses Cosmos DB as a source for a dashboard of the correlated records. You could also have Power BI point directly from Stream Analytics; however, this data would not be persisted in a data store.

Additional technologies

Azure Monitor collects performance metrics about the Azure services deployed in the solution. By visualizing these in a dashboard, you can get insights into the health of the solution.

Design a Stream Processing Pipeline with Azure Databricks

This architecture shows an end-to-end stream processing pipeline. This is useful in use cases where an organization wants to ingest data from social media or custom application and analyze the data in near real time. In the example below, the pipeline ingests data from two sources, correlates records in the two streams, and performs a join on the related data streams. The results are stored for further analysis that go beyond the capabilities of Stream Analytics and enables Data Science experiments to be performed on the data using Azure Databricks. It is designed to simulate the a taxi company collects data about each taxi trip.



Data Source

In this architecture, there are two data sources that generate data streams in real time. The first stream contains ride information, and the second contains fare information. This is generated in real time by an application within the taxi cab.

Ingestion and Data Storage

Each data source sends a stream of data to an associated Azure Event Hub. Azure Event Hubs is a Big Data streaming platform and event ingestion service, capable of receiving and processing millions of events per second. It is then connected to Azure Databricks, which outputs its analysed data into an Azure Cosmos DB using the Cassandra API. The Cassandra API is used because it supports time series data modeling which can be used in the analysis of the trip and fare data.

Analysis and Visualization

The analysis of the streaming data is performed with Azure Databricks, which is is an Apache Spark-based analytics platform optimized for the Microsoft Azure cloud services platform. Databricks is used to correlate the taxi ride and fare data, and also to enrich the correlated data with neighborhood data stored in the Databricks file system.

Custom data analysis and visualization tools can be used to present the data to the end users

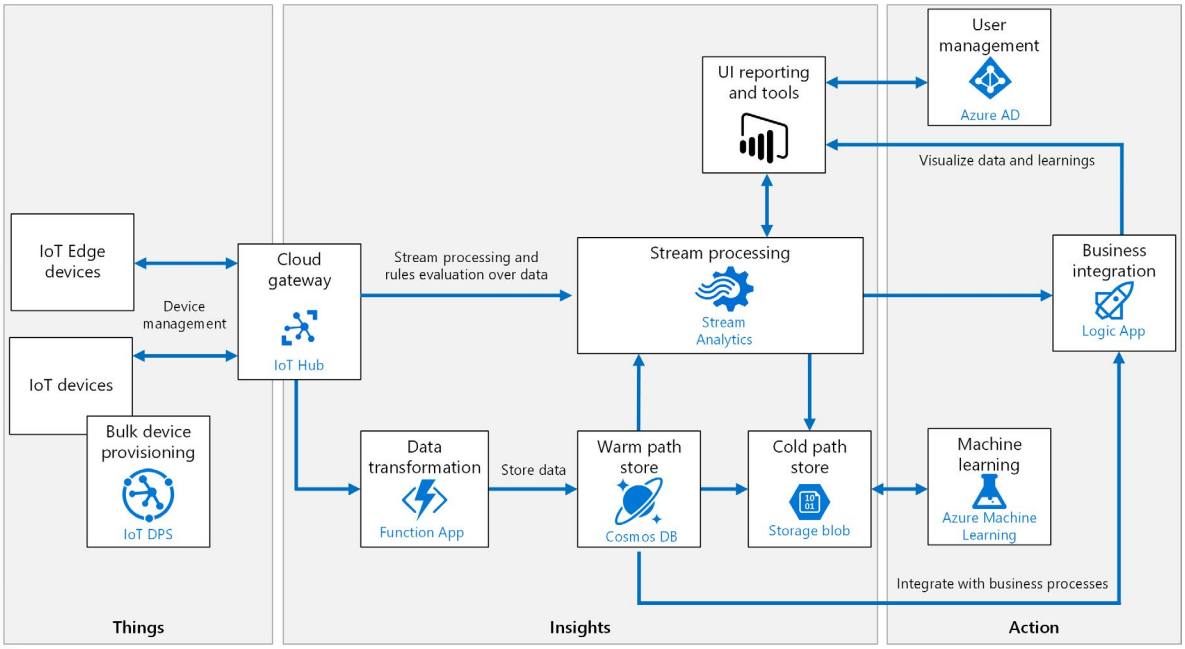
Additional technologies

Application log data collected by Azure Monitor is stored in a Log Analytics workspace. Log Analytics queries can be used to analyze and visualize metrics and inspect log messages to identify issues within the application.

Create an IoT Reference Architecture

This reference architecture shows an architecture for an Internet of Things (IoT) application on Azure using PaaS (platform-as-a-service) components. It is useful in use cases that takes source data from devices such as utility meters or field devices that have the capabilities to stream data in real-time. This example is shown in contrast to an architecture that make uses of data that is sourced from an application through Event Hubs.

Note: IoT is not part of the exam objectives, and this content is included to provide a holistic view of streaming data architectures.



Data Source

IoT applications can be described as things (devices) sending data that generates insights. An example is an engine (the thing) sending temperature data. This data is used to evaluate whether the engine is performing as expected (the insight). The insight is used to proactively prioritize the maintenance schedule for the engine (the action).

The are two types of devices. IoT devices and IoT edge devices. Typically, IoT devices are constantly connected to the cloud which provides the capability to perform data processing and analysis in Azure. Sometimes, there are occasions where devices cannot be constantly connected to the cloud. For example, a ship travelling through an ocean where there is no connectivity. In this case, Edge IoT devices contain some processing and analysis logic within the IoT edge device so that there is no dependency on a constant connection to the cloud.

Ingestion and Data Storage

A cloud gateway provides a cloud hub for devices to connect securely to the cloud and send data. It also provides device management, capabilities, including command and control of devices. Azure IoT Hub is an example of a cloud gateway on Azure. IoT Hub is a hosted cloud service that ingests events from devices, acting as a message broker between devices and backend services. IoT Hub provides secure connectivity, event ingestion, bidirectional communication, and device management. For registering and connecting large sets of devices, you can use the IoT Hub Device Provisioning Service (DPS). DPS lets you assign and register devices to specific Azure IoT Hub endpoints at scale.

From this point there are two ways in which the IoT data can be processed. Data can be processed in either a hot path, or a cold path. The hot path analyzes data in near-real-time, as it arrives. In the hot path, telemetry must be processed with very low latency. The hot path is typically implemented using a stream processing engine. The output may trigger an alert, or be written to a structured format that can be queried using analytical tools. The data storage for this data is typically place in a data store that is highly responsive and available. This makes Azure Cosmos DB a great choice to store hot path data

The cold path performs batch processing at longer intervals (hourly or daily). The cold path typically operates over large volumes of data, but the results don't need to be as timely as the hot path. In the cold path, raw telemetry is captured and then fed into a batch process. The data storage for this data is typically place in a data store that does not need to be queried directly, and does not need to be responsive in real-time. Azure Blob or Data Lake Store Gen II are useful data store in this scenario.

Analysis and Visualization

From a real-time analysis perspective, analysis of the streaming data is performed with Stream Analytics itself. A Stream Analytics job is created that contains the IoT Hubs as input streams, with a query that performs stream processing of the data streams. An output is defined to Cosmos DB that stores the results written as JSON documents to a Cosmos DB document database. As an alternative to Stream Analytics, another option is Apache Spark on Azure Databricks.

From a batch mode analysis perspective, the data in Azure Blob works with Machine Learning Services to allow predictive algorithms to be executed over historical telemetry data, useful in scenarios such as predictive maintenance. Power BI is used to visualize the data directly from Stream Analytics.

Additional technologies

The Azure Functions outlined in the architecture perform data transformation that manipulates or aggregates the telemetry stream. Examples of transformation include protocol transformation, such as converting binary data to JSON, or combining data points. Azure Functions built-in integration with IoT Hub, Cosmos DB, and Blob Storage.

Azure Logic Apps can be used for business process integration which can include performing actions based on insights from the device data. This could include storing informational messages, raising alarms, sending email or SMS messages, or integrating with CRM. Azure Active Directory is used to authenticate and authorize users.

Moudle Summary

Summary

The technologies that are available within Azure is making it easier for organizations to take advantage of real-time analytics to drive business transformation. Once the preserve of large enterprises, the affordability and ease of provisioning of services such as Event and IoT Hubs, Stream Analytics and Spark is now allowing organizations to view real-time analytics as a realistic option that they can use to help them make business decisions.

In this module, you have explored a number of real-time architecture patterns that are common within a range of industries that could be applied to your organization including:

- Lambda architectures for a Real-Time Mode Perspective
- Architect a stream processing pipeline with Azure Stream Analytics
- Design a stream processing pipeline with Azure Databricks
- Create an Azure IoT reference architecture

Module 4 Security Design Considerations

Module Introduction

Introduction

Security is one of the most important aspects of any architecture. Ensuring that your business data and customer data are secure is critical. A public data breach can ruin a company's reputation as well as cause significant personal and financial harm. In this module, we'll discuss key architectural security considerations as you design an environment on the cloud.

NOTE
The concepts discussed in this module are not all-inclusive, but represent some of the important considerations when building a solution on the cloud. Microsoft publishes a broad set of patterns, guidelines, and examples on designing applications on Azure. It is highly recommended that you look through the content in the **Azure Architecture Center**[1] as you start planning and designing your architecture.

Learning objectives

In this module, you will:

- Learn how to take a defense in depth approach to securing your architecture.
- Learn how to protect your identities.
- Learn what technologies are available to protect your Azure infrastructure.
- Learn how and where to use encryption to secure your data.
- Learn how to protect your architecture at the network level.
- Learn how to leverage application security best practices to integrate security into your application.

1 https://docs.microsoft.com/azure/architecture/

Security Design Considerations

Defense in depth

There's no easy approach for security and no solution that solves all your problems from a security perspective. Let's imagine that your organization has neglected security in its environment. The company has realized it needs to put some major focus in this area. It is not exactly sure where to start, or if it's possible to just buy a solution to make the environment secure. The company knows it needs a holistic approach, but is unsure what really fits into that. Here, we'll identify key concepts of defense in depth, identify key security technologies and approaches to support a defense in depth strategy, and discuss how to apply these concepts when architecting your own Azure services.

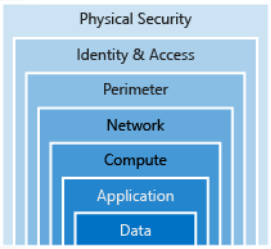
A layered approach to security

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 data. 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 our physical datacenters and across Azure services. The objective of defense in depth is to protect and prevent information from being stolen by individuals not authorized to access it. The common principles used to define a security posture are confidentiality, integrity, and availability, known collectively as CIA.

- **Confidentiality** - Principle of least privilege. Restricts access to information only to individuals explicitly granted access. This information includes protection of user passwords, remote access certificates, and email content.
- **Integrity** - The prevention of unauthorized changes to information at rest or in transit. A common approach used in data transmission is for the sender to create a unique fingerprint of the data using a one-way hashing algorithm. The hash is sent to the receiver along with the data. The data's hash is recalculated and compared to the original by the receiver to ensure the data wasn't lost or modified in transit.
- **Availability** - Ensure services are available to authorized users. Denial of service attacks are a prevalent cause of loss of availability to users.

Security layers

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.

Applications

- 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. Encourage all development teams to ensure their applications are secure by default. Make 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 through segmentation and access controls
- 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 only allow what is required. Segment your resources and use network level controls to restrict communication to only what is needed. 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 on them is important to keep your network secure.

Policies & access

- Control access to infrastructure, change control
- Use single sign-on and multi-factor authentication
- Audit events and changes

The policy & access layer is all about ensuring identities are secure, and that 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.

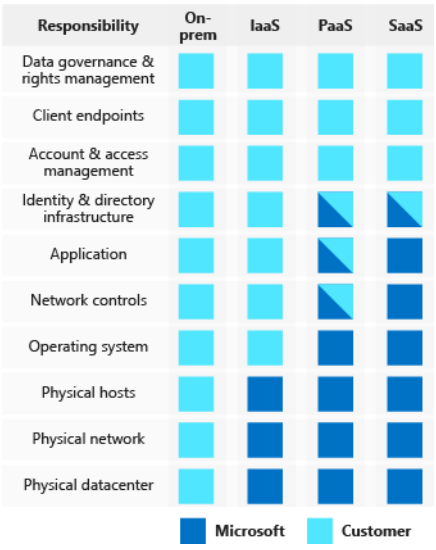
With physical security, the intent is to provide physical safeguards against access to assets. This ensures that other layers can't be bypassed, and loss or theft is handled appropriately.

Each layer can implement one or more of the CIA concerns.

#	Ring	Example	Principle
1	Data	Data encryption at rest in Azure blob storage	Integrity
2	Application	SSL/TLS encrypted sessions	Integrity
3	Compute	Regularly apply OS and layered software patches	Availability
4	Network	Network security rules	Confidentiality
5	Perimeter	DDoS protection	Availability
6	Policies & Access	Azure Active Directory user authentication	Integrity
7	Physical Security	Azure data center biometric access controls	Confidentiality

Shared responsibilities

As computing environments move from customer-controlled datacenters to cloud datacenters, the responsibility of security also shifts. Security is now a concern shared by both cloud providers and customers.



Continuous improvement

The threat landscape is evolving in real time and at massive scale, therefore a security architecture is never complete. Microsoft and our customers require the ability to respond to these threats intelligently, quickly, and at scale.

Azure Security Center[2] provides customers with unified security management and advanced threat protection to understand and respond to security events on-premises and in Azure. In turn, Azure customers have a responsibility to continually re-evaluate and evolve their security architecture.

Identity Management

In this lesson, we'll discuss identity as a security layer for access to applications and database systems, the benefits of single sign-on (SSO) to provide identity security, and why you should consider replicating on-premises identities to Azure Active Directory.

Identity as a layer of security

Digital identities (AKA Security Principles) are an integral part of today's business and social interactions on-premises and online. In the past, identity and access services were restricted to operating within a company's internal network, using protocols such as Kerberos and LDAP that were designed for this purpose. More recently, mobile devices have become the primary way people interact with digital services. Customers and employees alike expect to be able to access services from anywhere at any time, which has driven the development of identity protocols that can work at internet scale across many disparate devices and operating systems.

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

2 https://azure.microsoft.com/services/security-center/

applications, and as complexity requirements increase, it makes it more difficult for users to remember them.

On the other side is the management required for 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, users only need to remember one ID and one password. Access across database systems or 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. As a result, taking advantage of Single sign-on means that you can streamline your security management with the work of the Azure Administrator. You just assign a role to the database technology that you want the users to access.

Single Sign-On with Azure Active Directory

Azure Active Directory (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 data and applications using centralized rules and policies configured in Azure AD. In addition Azure Active Directory Business to Consumer (AAD B2C) can be used to provide identity management of external users.

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.

Infrastructure Protection

Here, we will explore utilizing capabilities on Azure to protect the access to infrastructure resources such as Azure Services and Data Stores.

Criticality of infrastructure

Cloud infrastructure is becoming a critical piece of many businesses. It is critical to ensure people and processes have only the rights they need to get their job done. Assigning incorrect access can result in data loss, data leakage, or cause services to become unavailable.

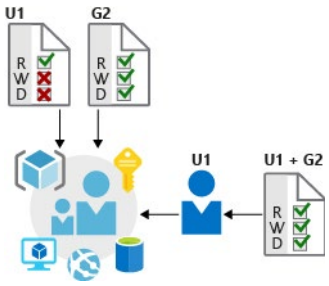
Data Engineers can be responsible for a variety of database systems, and permission that are required to access them. So correctly granting access can quickly become unmanageable and can lead to a 'one size fits all' approach. This approach can reduce the complexity of administration, but makes it far easier to inadvertently grant more permissive access than required.

Role-based access control

Role-based access control (RBAC) offers a slightly different approach. Roles are defined as collections of access permissions. Security principals are mapped to roles directly or through group membership.

Separating security principals, access permissions, and resources provides simplified access management and more fine-grained control.

On Azure, users, groups, and roles are all stored in Azure Active Directory (Azure AD). The Azure Resource Manager API uses role-based access control to secure all resource access management within Azure.

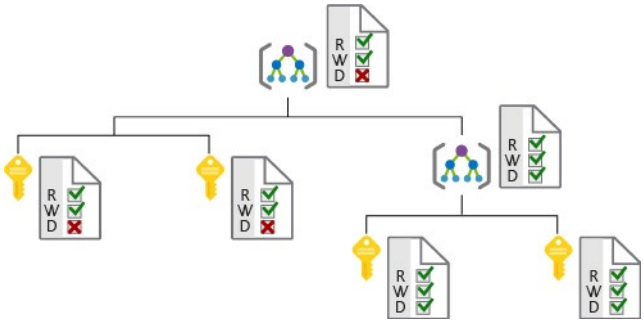


Roles and management groups

Roles are sets of permissions, like “Read-only” or "Contributor", that users can be granted to access an Azure service instance. Roles can be granted at the individual service instance level, but they also flow down the Azure Resource Manager hierarchy. Roles assigned at a higher scope, like an entire subscription, are inherited by child scopes, like service instances.

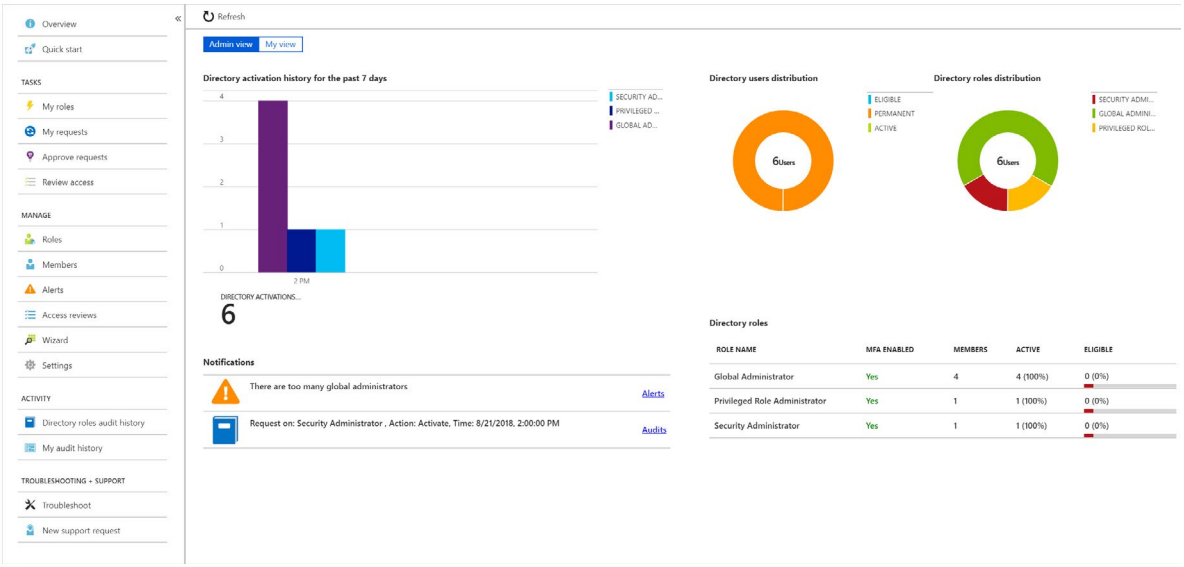
Management groups are an additional hierarchical level recently introduced into the RBAC model. Management groups add the ability to group subscriptions together and apply policy at an even higher level.

The ability to flow roles through an arbitrarily defined subscription hierarchy also allows administrators to grant temporary access to an entire environment for authenticated users. For example, an auditor may require temporary read-only access to all subscriptions.



Privileged Identity Management

In addition to managing Azure resource access with 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 & Azure resource access reviews.



Providing identities to services

It's often valuable for services to have identities. Often times, 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** as they are used in Identity management world.

An **identity** is just a thing that can be authenticated. Obviously, this includes users with username 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. Consider the use 'sudo' on a bash prompt or on Windows using "run as Administrator". In both of those cases, you are still logged in as the same identity as before, but you've changed the role under which you are executing.

So, a **Service Principal** is literally named. It is an identity that is used by a service or application. Like other identities, it can be assigned roles.

For example, your organization can assign its deployment scripts to run authenticated as a service principal. If that is the only identity that has permission to perform destructive actions, it could prevent incidents such as an accidental resource deletion.

Managed identities for Azure resources

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 resources 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 (the list is constantly growing). When you create a managed identity for a service, you are creating an account on the Azure AD tenant. Azure infrastructure will automatically take care of authenticating the service and managing the account. You can then use that account like any other AD account including securely letting the authenticated service access other Azure resources.

Encryption

Data is an organization's most valuable and irreplaceable asset, and encryption serves as the last and strongest line of defense in a layered security strategy. Here, we'll take a look at what encryption is, how to approach the encryption of data, and what encryption capabilities are available on Azure.

What is encryption?

Encryption is the process of making data unreadable and unusable. 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 cannot 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.

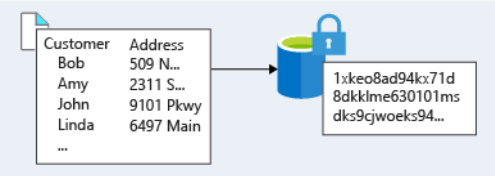
Both symmetric and asymmetric encryption play a role in properly securing your data.

Encryption is typically approached in two ways: encryption at rest and encryption in transit.

Encryption at rest

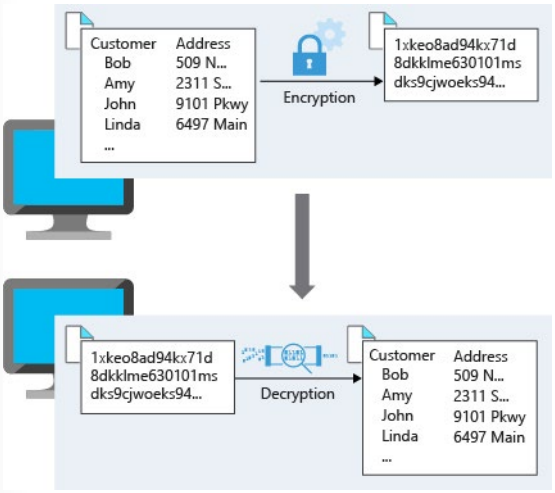
Data at rest is the data that has been stored on a physical medium. This could be data stored on the disk of a server, data stored in a database, or data stored in a storage account. Regardless of the storage mechanism, encryption of data at rest ensures that the stored data is unreadable without the keys and secrets needed to decrypt it. If an attacker were 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. In such a scenario, an attacker would have to attempt attacks against encrypted data, which are much more complex and resource consuming than accessing unencrypted data on a hard drive.

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 that the business stores about customers or employees, and even the keys and secrets used for the encryption of the data itself.



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 encrypting the data prior to sending it over a network, or setting up a secure channel to transmit unencrypted 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.



Identify and classify data

If an organization have had previous incidents that exposed sensitive data, there's a gap between what they are encrypting and what they should be encrypting. They need to start by identifying and classifying the types of data they are storing, and align this with the business and regulatory requirements surrounding the storage of data. It's beneficial to classify this data as it relates to the impact of exposure to the organization, its customers, or partners. An example classification could be as follows:

Data classification	Explanation	Examples
Restricted	Data classified as restricted poses significant risk if exposed, altered, or deleted. Strong levels of protection are required for this data.	Data containing Social Security numbers, credit card numbers, personal health records
Private	Data classified as private poses moderate risk if exposed, altered, or deleted. Reasonable levels of protection are required for this data. Data that is not classified as restricted or public will be classified as private.	Personal records containing information such as address, phone number, academic records, customer purchase records
Public	Data classified as public poses no risk if exposed, altered, or deleted. No protection is required for this data.	Public financial reports, public policies, product documentation for customers

By taking an inventory of the types of data being stored, they can get a better picture of where sensitive data may be stored and where existing encryption may or may not be happening.

A thorough understanding of the regulatory and business requirements that apply to data the organization stores is also important. The regulatory requirements an organization must adhere to will often drive a large part of the data encryption requirements. Industries will fall under different regulatory requirements. For example, a financial institution may store account information that falls within Payment Card Industry (PCI) standards. An organization doing business in the EU may fall under the General Data Protection Regulation (GDPR), which defines the handling of personal data in the EU. Business requirements may also dictate that any data that could put the organization at financial risk containing competitive information needs to be encrypted.

Once you have the data classified and your requirements defined, you can then take advantage of various tools and technologies to implement and enforce encryption in your architecture.

Encryption on Azure

Let's take a look at some ways that Azure enables you to encrypt data across services.

Encrypting raw storage

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.

Encrypting databases

As organizations migrate to the cloud, they may move databases to Azure SQL Database and want to ensure that their data is encrypted within their database. If the data files, log files, or backup files were stolen, they want to ensure they are unreadable without access to the encryption keys.

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 Databases.

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 and handles all the details. Bring-your-own-key is also supported with keys stored in Azure Key Vault.

Encrypting virtual machines

Storage Service encryption provides low-level encryption protection for data written to physical disk, but how do you protect the virtual hard disks (VHD) of virtual machines? If a malicious attacker gained access to your Azure subscription and exfiltrated the VHDs of your virtual machines, how would you ensure they would be unable to access data stored on the VHD?

Azure Disk Encryption (ADE) is a capability that helps you encrypt your Windows and Linux IaaS virtual machine disks. ADE 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 identity for Azure services for accessing the key vault).

Encrypting 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?

Azure Key Vault is a cloud service that works as a secure secrets store. Key Vault allows you to create multiple secure containers, called vaults. These vaults are backed by hardware security modules (HSMs). Vaults help reduce the chances of accidental loss of security information by centralizing the storage of application secrets. Key Vaults also control and log the access to anything stored in them. Azure Key Vault can handle requesting and renewing Transport Layer Security (TLS) certificates, providing the features required for a robust certificate lifecycle management solution. Key Vault is designed to support any type of secret. These secrets could be passwords, database credentials, API keys and, certificates.

Because Azure AD identities can be granted access to use Azure Key Vault secrets, applications using managed identities for Azure services can automatically and seamlessly acquire the secrets they need.

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.

What is network security

Network security is protecting the communication of resources within and outside of your network. The goal is to limit exposure at the network layer across your services and systems. By limiting this exposure, you decrease the likelihood that your resources can be attacked. In the focus on network security, efforts can be focused on the following areas:

- Securing traffic flow between applications and the internet
- Securing traffic flow amongst applications
- Securing traffic flow between users and the application

Securing traffic flow between applications and the internet focuses on limiting exposure outside your network. Network attacks will most frequently start outside your network, so by limiting the internet exposure and securing the perimeter, the risk of being attacked can be reduced.

Securing traffic flow amongst applications focuses on data between applications and their tiers, between different environments, and in other services within your network. By limiting exposure between these resources, you reduce the effect a compromised resource can have. This can help reduce further propagation within a network.

Securing traffic flow between users and the application focuses on securing the network flow for your end users. This limits the exposure your resources have to outside attacks, and provides a secure mechanism for users to utilize your resources.

A layered approach to network security

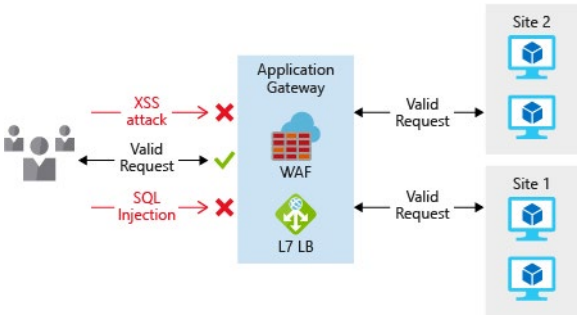
A common thread throughout this module has been taking a layered approach to security, and this approach is no different at the network layer. It's not enough to just focus on securing the network perimeter, or focusing on the network security between services inside a network. A layered approach provides 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

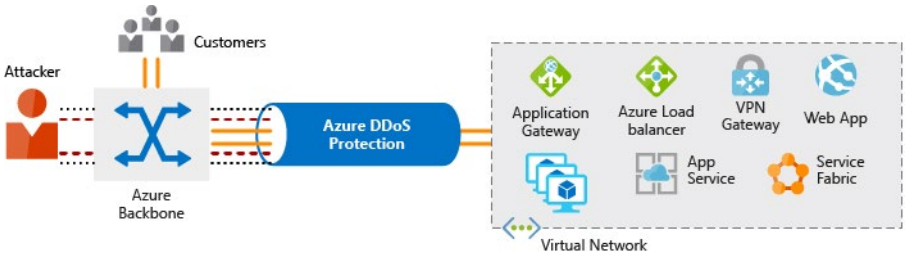
If we start on the perimeter of the network, we're focused on limiting and eliminating attacks from the internet. A great first place to start is to assess the resources that are internet-facing, and only allow inbound and outbound communication where necessary. Identify all resources that are allowing inbound network traffic of any type, and ensure they are necessary and restricted to only the ports/protocols required. Azure Security Center is a great place to look for this information, as it will identify internet-facing resources that don't have network security groups (NSG) associated with them, as well as resources that are not secured behind a firewall.

To provide inbound protection at the perimeter, there are a couple of ways to do this. Application Gateway is a Layer 7 load balancer that also includes a web application firewall (WAF) to provide advanced security for your HTTP-based services. The WAF is based on rules from the OWASP 3.0 or 2.2.9 core rule sets, and provides protection from commonly-known vulnerabilities such as cross-site scripting and SQL injection.



For protection of non-HTTP-based services or for increased customization, network virtual appliances (NVA) can be used to secure your network resources. NVAs are similar to firewall appliances you might find in on-premises networks, and are available from many of the most popular network security vendors. NVAs can provide greater customization of security for those applications that require it, but can come with increased complexity, so careful consideration of requirements is advised.

Any resource exposed to 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. To mitigate these attacks, Azure DDoS provides basic protection across all Azure services and enhanced protection for further customization for your resources. DDoS protection blocks attack traffic and forwards the remaining traffic to its intended destination. Within a few minutes of attack detection, you are notified using Azure Monitor metrics.



Virtual network security

Once inside a virtual network (VNet), it's important to limit communication between resources to only what is required.

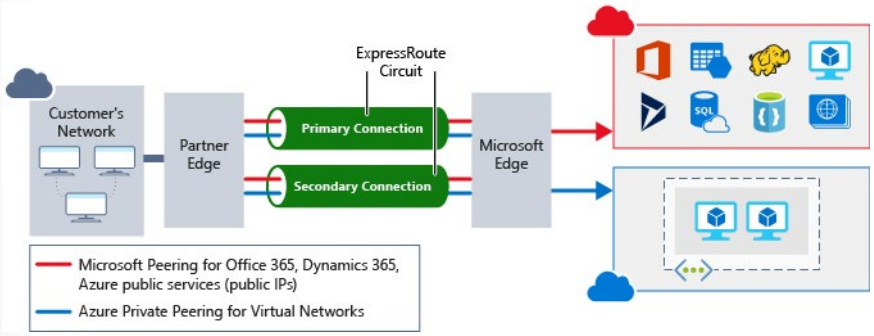
To isolate Azure services to only allow communication from virtual networks, use VNet service endpoints. With service endpoints, Azure service resources can be secured to your virtual network. Securing service resources to a virtual network provides improved security by fully removing public internet access to resources, and allowing traffic only from your virtual network. This reduces the attack surface for your environment, reduces the administration required to limit communication between your VNet and Azure services, and provides optimal routing for this communication.

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, and this is no different when working with virtual networking on Azure. Connection between Azure VNets and an on-premises VPN device is a great way to provide secure communication between your network and your virtual machines on Azure.

To provide a dedicated, private connection between your network and Azure, you can use 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 internet. You don't need to allow access to these services for your end users over the internet, and you can send this traffic through appliances for further traffic inspection.



To easily integrate multiple VNets in Azure, VNet peering establishes a direct connection between designated VNets. Once established, you can use NSGs to provide isolation between resources in the same way you secure resources within a VNet. This integration gives you the ability to provide the same fundamental layer of security across any peered VNets. Communication is only allowed between directly connected VNets.

Application Security

Hosting applications on a cloud platform provides a number of advantages when compared to traditional on-premises deployments. The cloud's shared-responsibility model moves security at the physical network, building, and host levels under the control of the cloud provider. An attacker trying to compro-

mise the platform at this level would see diminishing returns versus the considerable investment and insight providers make in securing and monitoring their infrastructure.

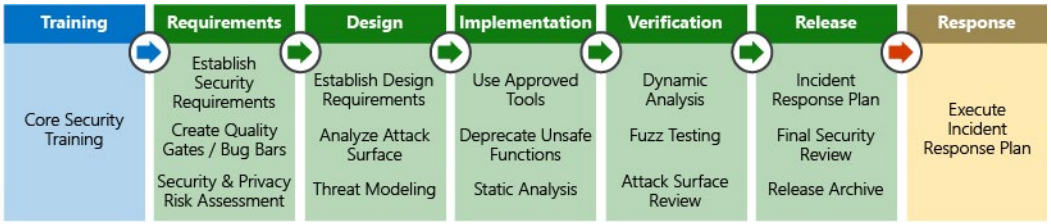
It's therefore far more effective for attackers to pursue vulnerabilities introduced at the application level by cloud-platform customers. Furthermore, by adopting Platform as a Service (PaaS) to host their applications, customers are able to free resources from managing operating system security and deploy them to harden application code and monitor the identity perimeter around the application. In this unit we will discuss some of the ways application security can be improved through design.

Security Development Lifecycle

Microsoft's **Security Development Lifecycle**[3] (SDL) process can be used during the application design stage to ensure security concerns are incorporated in the software development lifecycle. Security and compliance issues are far easier to address when designing an application and can mitigate many common errors that can lead to security flaws in the final product. Fixing issues early in the software development journey is also far less costly. The typical sequence of SDL steps a software project can use are as follows:

1. Training
   - Core security training
2. Requirements
   - Define requirements and quality gates
   - Analyze security and privacy risks
3. Design
   - Attack surface analysis
   - Threat modeling
4. Implementation
   - Specify tools to ensure code quality can be measured
   - Enforce banned APIs and functions
   - Perform static code analysis
   - Scan repositories for stored secrets
5. Verification
   - Dynamic/Fuzz testing
   - Verify threat models/attack surface
6. Release
   - Form security response plan
   - Perform a final security review
   - Release archive
7. Response
   - Execute threat response execution

3 https://www.microsoft.com/sdl



The SDL is as much a cultural aspect as it is a process or set of tools. Building a culture where security is a primary focus and requirement of any application development can make great strides in evolving an organization's capabilities around security.

Operational security assessment

Once an application and database system has been deployed, it's essential to continually evaluate its security posture, determine how to mitigate any issues that are discovered, and feed the knowledge back into the software development cycle. The depth to which this is performed is a factor of the maturity level of the software development and operational teams as well as the data privacy requirements.

Security vulnerability scanning software services are available to help automate this process and assess security concerns on a regular cadence, without burdening teams with costly manual processes, such as penetration testing.

Azure Security Center is a free service, now enabled by default for all Azure subscriptions, that is tightly integrated with other Azure application level services, such as Azure Application Gateway and Azure Web Application Firewall. By analyzing logs from these services, ASC can report on known vulnerabilities in real time, recommend responses to mitigate them, and even be configured to automatically execute play-books in response to attacks.

Identity as the perimeter

Identity validation is becoming the first line in defense for applications. Restricting access to a web application by authenticating and authorizing sessions can drastically reduce the attack surface area. Azure AD and Azure AD B2C offer an effective way to offload the responsibility of identity and access to a fully managed service. Azure AD conditional access policies, privileged identity management, and Identity Protection controls further enhance a customer's ability to prevent unauthorized access and audit changes.

Data protection

Customer data is the target for most, if not all attacks against web applications. The secure storage and transport of data between an application and its data storage layer is paramount.

To comply with these requirements, organizations can modify their applications to encrypt all data at rest and in transit. For example, Transport Layer Security (TLS) is used to encrypt data exchanged between the web application and back-end SQL databases. Data is also encrypted at rest in SQL Server using Transparent Data Encryption (TDE), ensuring that even if the environment is compromised, data is effectively useless to anyone without the correct decryption keys.

To encrypt data stored in blob storage, client-side encryption can be used to encrypt the data in memory before it's written to the storage service. Libraries supporting this encryption are available for .NET, Java, and Python, and enable the integration of data encryption directly into applications to enhance data integrity.

Secure key and secret storage

Separating application secrets (connection strings, passwords, etc.) and encryption keys from the application used to access data is vital. Encryption keys and application secrets should never be stored in the application code of configuration files. Instead, a secure store such as Azure Key Vault should be used. Access to this sensitive data can then be limited to application identities through Managed Service Identities, and keys can be rotated on a regular basis to limit exposure in the case of encryption key leakage. Customers can also choose to use their own encryption keys generated by on-premises Hardware Security Modules (HSM) and even mandate that Azure Key Vault instances are implemented in single-tenant, discrete HSMs.

Module Summary

Summary

We've covered a lot of topics in this module, but rightfully so, as security is such an important part of any architecture. Let's review what we've covered.

Defense in depth

We've talked through how to approach security in your architecture through defense in depth. Looking only at firewalls or antimalware alone isn't enough to slow down attackers. Use a layered approach and address security at each layer.

Identity management

We've talked through identity management, and how identity becomes an integral piece of the architectural puzzle. Azure AD has a number of features and capabilities to improve the identity security story for your environment.

Infrastructure protection

Protecting the access to your infrastructure ensures that the resources you create are administered by only those who should be administering them.

Encryption

Encryption is often the last layer of defense against access to your data. By using encryption, you make your data unreadable to anyone without the decryption keys. You should identify and classify your data, then align with encryption requirements from your business and any regulations your organization must adhere to.

Network security

Finally, we talked through securing your network. We looked at ways to secure traffic flow between applications and the internet. We described some ways to secure traffic flow amongst applications. And we wrapped up by looking at how to secure traffic flow between users and an application.

Module 5 Designing for Scale and Resiliency

Moudle Introduction

Introduction

Whether you're running a public facing application that handles massive amounts of traffic or an internal business API that manages critical data for internal systems, your users expect a system that performs well. Scaling your system to handle load, identifying network bottlenecks, and optimizing your storage performance are important to ensure your users have the best experience. Your business also relies on access to the systems and data that make it run. Every minute that your customers or your internal teams don't have access to what they need can result in a loss of revenue. It is your job to make sure that doesn't happen, and if it does, to minimize downtime.

NOTE: The concepts discussed in this module are not all-inclusive, but represent some of the important considerations when building a solution on the cloud. Microsoft publishes a broad set of patterns, guidelines, and examples on designing applications on Azure. It is highly recommended that you look through the content in the **Azure Architecture Center**[1] as you start planning and designing your architecture.

Learning objectives

In this module, you will:

- Adjust your capacity based on workload by scaling up and scaling out
- Optimize network performance
- Optimize storage and database performance
- Identify performance bottlenecks in your solution
- Leverage Azure services to design a highly available application
- Incorporate Azure disaster recovery capabilities into your architecture
- Backup and restore on Azure to protect your application from data loss or corruption

1 https://docs.microsoft.com/azure/architecture/

Designing for Scale and Resiliency

Scaling systems

It's rare that we can exactly predict the load on our system: public facing applications might grow rapidly or an internal application might need to support a larger user base as the business grows. Even then, databases may struggle under the load as well.

In a situation where we can predict load, it's rarely flat: retailers have more demand during the holidays and sports websites peak during playoffs. Here, we'll define *scaling up/down* and *scaling out/in*, cover some ways Azure can improve your scaling capabilities, and look at how serverless and container technologies can improve your architecture's ability to scale.

What is scaling?

Scaling is the process of managing your resources to help your application or databases meet a set of performance requirements. When we have too many resources serving users, we won't be using it efficiently and we'll be wasting money. Too few available resources means that the performance of our application or databases could be impacted. The goal is to meet our defined performance requirements while optimizing for cost.

"*Resources*" can refer to anything we need to manage to run our applications. Memory and CPU for virtual machines are the most obvious resources, but some Azure services might require you to consider bandwidth or abstractions, like Cosmos DB Request Units, or SQL Date Warehouse Units.

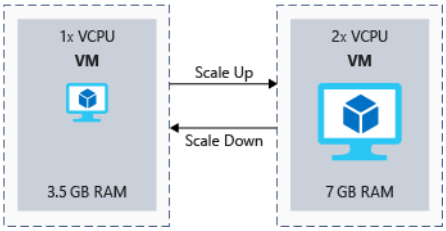
In a world where application demand is constant, it's easy to predict the right amount of resources you'll need. In the real world, the demands of applications change over time, so the right amount of resources you'll need can be harder to predict. If you're lucky, that change will be predictable or seasonal, but that is not typical of all applications. Ideally, you want to provision the right amount of resources to meet demand and adjust as demand changes.

Scaling is difficult in an on-premises scenario, where you purchase and manage your own servers. Adding resources can be costly and often takes too much time to bring online, sometimes longer than your actual need for the increased capacity. It can be just as difficult to then reduce capacity during times of low demand on the system, so you may be stuck with the increased cost.

Easy scaling is a key benefit of Azure. Most Azure resources let you easily add or remove resources as demand changes, and many services have automated options so they monitor demand and adjust for you. This automatic scaling capability, commonly known as autoscaling, lets you set thresholds for the minimum and maximum level of instances that should be available, and will add or remove instances based upon a performance metric (for example, CPU utilization).

What is scaling up or down?

Scaling up is the process where we increase the capacity of a given instance. A virtual machine could be increased from 1 vCPU and 3.5 GB of RAM to 2 vCPUs and 7 GB of RAM to provide more processing capacity. On the other hand, scaling down is the process where we lower the capacity of a given instance. For example, reducing a virtual machine's capacity from 2 vCPUs and 7 GB of RAM to 1 vCPU and 3.5 GB of RAM, reducing both capacity and cost. The following illustration shows an example of changing the size of a virtual machine.



Let's take a look at what scaling up or down means in the context of Azure resources:

- Azure SQL Database is a platform as a service (PaaS) implementation of Microsoft SQL Server. You can scale up a database based upon the number of database transaction units (DTUs) or vCPUs. DTUs are an abstraction of underlying resources and are a blend of CPU, IO, and memory. For instance, you could scale your Azure SQL database from a size of P2 with 250 DTUs up to a P4 with 500 DTUs to give the database more throughput and capacity.
- In Azure virtual machines, you scale based upon a virtual machine size. That size has a certain amount of vCPUs, RAM, and local storage associated with it. For example, we could scale up from a Standard_DS1_v2 virtual machine (1 vCPU and 3.5 GB of RAM) to a Standard_DS2_v2 virtual machine (2 vCPUs and 7 GB of RAM).

To have these capabilities in an on-premises environment you typically have to wait for procurement of the needed hardware and installation before you can start using the new level of scale. In Azure, the physical resources are already deployed and available for you. You simply need to select the alternate level of scale that you are looking to use.

You may need to consider the impact of scaling up in your solution, depending upon the cloud services that you have chosen.

For example, if you choose to scale up in Azure SQL Database, the service deals with scaling up individual nodes and continues the operation of your service. Changing the service tier and/or performance level of a database creates a replica of the original database at the new performance level, and then switches connections over to the replica. No data is lost during this process, and there's only a brief interruption (typically less than four seconds) when the service switches over to the replica.

Alternatively, if you choose to scale up or down a virtual machine, you do so by selecting a different instance size. In most cases this requires a restart of the VM, so it's best to have the expectation that a reboot will be required and you'll need to account for when performing this activity.

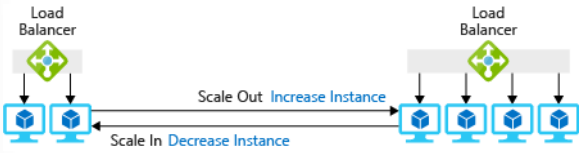
Finally, you should always look for places where scaling down is an option. If your application can provide adequate performance at a lower price tier, your Azure bill could be significantly reduced.

What is scaling out or in?

Where scaling up and down adjusts the amount of resources a single instance has available, scaling out and in adjusts the total number of instances.

Scaling out is the process of adding more instances to support the load of your solution. For example, if our website front end were hosted on virtual machines, we could increase the number of virtual machines if the level of load increased.

Scaling in is the process of removing instances that are no longer needed to support the load of your solution. If the website front ends have low usage, we may want to lower the number of instances to save cost. The following illustration shows an example of changing the number of virtual machine instances.



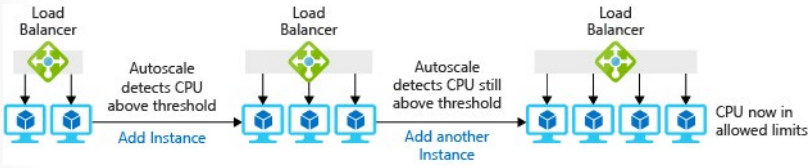
Here are some examples of what scaling out or in means in the context of Azure resources:

- For the infrastructure layer, you would likely use virtual machine scale sets to automate the addition and removal of extra instances.
    - Virtual machine scale sets let you create and manage a group of identical, load balanced VMs.
    - The number of VM instances can automatically increase or decrease in response to demand or a defined schedule.
- In an Azure SQL Database or Azure Synapse Analytics implementation, you could share the load across database instances by sharding. *Sharding* is a technique to distribute large amounts of identically structured data across a number of independent databases.
- In Azure App Service, the App Service plan is the virtual web server farm hosting your application. Scaling out in this way means that you're increasing the number of virtual machines in the farm. As with virtual machine scale sets, the number of instances can be automatically raised or lowered in response to certain metrics or a schedule.

Scaling out is typically easily performed in the Azure portal, command-line tools, or Resource Manager templates, and in most cases is seamless to the end user.

Autoscale

You can configure some of these services to use a feature called autoscale. With autoscale you no longer have to worry about scaling services manually. Instead, you can set a minimum and maximum threshold of instances and scale based upon specific metrics (queue length, CPU utilization) or schedules (weekdays between 5:00 PM and 7:00 PM). The following illustration shows how the autoscale feature manages instances to handle the load.



Considerations when scaling in and out

When scaling out, the startup time of your application can impact how quickly your application can scale. If your web app takes two minutes to start up and be available for users, that means each of your instances will take two minutes until they are available to your users. You'll want to take this startup time into consideration when determining how fast you want to scale.

You'll also need to think about how your application handles state. When the application scales in, any state stored on the machine is no longer available. If a user connects to an instance that doesn't have its state, it could force them to sign in or re-select data, leading to a poor user experience. A common pattern is to externalize state to another service like Redis Cache or SQL Database, making your web

servers stateless. Now that our web front ends are stateless, we don't need to worry about which individual instances are available. They are all doing the same job and are deployed in the same way.

Serverless

Serverless computing provides a cloud-hosted execution environment that runs your apps but completely abstracts the underlying environment. You create an instance of the service, and you add your code; no infrastructure management or maintenance is required, or even allowed. This approach is becoming increasingly common in Machine Learning solutions.

You configure your serverless apps to respond to events. This could be a REST endpoint, a timer, or a message received from another Azure service. The serverless app runs only when it's triggered by an event.

Infrastructure isn't your responsibility. Scaling and performance are handled automatically, and you are billed only for the exact resources you use. There's no need to even reserve capacity. Azure Functions, Azure Container Instances, and Logic Apps are examples of serverless computing available on Azure.

Containers

A container is a method running applications in a virtualized environment. A virtual machine is virtualized at the hardware level, where a hypervisor makes it possible to run multiple virtualized operating systems on a single physical server. Containers take the virtualization up a level. The virtualization is done at the OS level, making it possible to run multiple identical application instances within the same OS.

Containers are well suited to scale out scenarios. They are meant to be lightweight and are designed to be created, scaled out, and stopped dynamically as environment and demand change.

A benefit of using containers is the ability to run multiple isolated applications on each virtual machine. Since containers themselves are secured and isolated at a kernel level, you don't necessarily need separate VMs for separate workloads.

While you can run containers on virtual machines, there are a couple of Azure services that focus on easing the management and scaling of containers:

- **Azure Kubernetes Service (AKS)**

  Azure Kubernetes Service allows you to set up virtual machines to act as your nodes. Azure hosts the Kubernetes management plane and only bills for the running worker nodes that host your containers.

  To increase the number of your worker nodes in Azure, you could use the Azure CLI to increase that manually. At time of writing, there is a preview of Cluster Autoscaler on AKS available that enables autoscaling of your worker nodes. On your Kubernetes cluster, you could use the Horizontal Pod Autoscaler to scale out the number of instances of the container to be deployed.

  AKS can also scale with the Virtual Kubelet described below.

- **Azure Container Instances (ACI)**

  Azure Container Instances is a serverless approach that lets you create and execute containers on demand. You're charged only for the execution time per second.

  You can use Virtual Kubelet to connect Azure Container Instances into your Kubernetes environment, including AKS. With Virtual Kubelet, when your Kubernetes cluster demands additional container instances, those demands can be met from ACI. Since ACI is serverless, there is no need to have reserved capacity. You can therefore take advantage of the control and flexibility of Kubernetes scaling with the per-second-billing of serverless. At time of writing, the Virtual Kubelet is described as experimental software and should not be used in production scenarios

Optimizing Network Performance

Network performance can have a dramatic impact on a user's experience. In complex architectures with many different services, minimizing the latency at each hop can have a huge impact on the overall performance. In this unit, we'll talk about the importance of network latency and how to reduce it within your architecture.

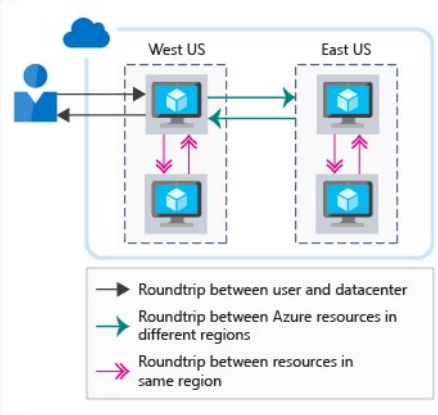
The importance of network latency

Latency is a measure of delay. Network latency is the time needed to get from a source to a destination across some network infrastructure. This time period is commonly known as a round-trip delay, or the time taken to get from the source to destination and back again.

In a traditional datacenter environment, latency may be minimal since resources often share the same location and a common set of infrastructure. The time taken to get from source to destination is lower when resources are physically close together.

In comparison, a cloud environment is built for scale. Cloud-hosted resources may not be in the same rack, datacenter, or even region. This distributed approach can have an impact on the round-trip time of your network communications. While all Azure regions are interconnected by a high-speed fiber back-bone, the speed of light is still a physical limitation. Calls between services in different physical locations will still have network latency directly correlated to the distance between them.

On top of this, the chattier an application, the more round trips that are required. Each round trip comes with a latency tax, with each round trip adding to the overall latency. The following illustration shows how the latency perceived by the user is the combination of the roundtrips required to service the request.



Now let's take a look at how to improve performance between Azure resources and from your end users to your Azure resources.

Latency between Azure resources

Imagine that your organization is piloting a new patient booking system running on several web servers and a database in the West Europe Azure region. This architecture minimizes the data time on the wire as resources are co-located inside an Azure region.

Suppose that the pilot of the system went well and has been expanded to users in Australia. Users in Australia will incur the round-trip time to the resources in West Europe to view the website, and their end-user experience will be poor due to the network latency.

Your team then decides to host another front-end instance in the Australia East region to reduce user latency. While this design helps reduce the time for the web server to return content to end users, their experience is still poor since there's significant latency communicating between the front-end web servers in Australia East and the database in West Europe.

There are a few ways we could reduce the remaining latency:

- Create a read-replica of the database in Australia East. This would allow reads to perform well, but writes would still incur latency. Azure SQL Database geo-replication allows for read-replicas.
- Sync your data between regions with Azure SQL Data Sync.
- Use a globally distributed database such as Azure Cosmos DB. This would allow both reads and writes to occur regardless of location, but may require changes to the way your application stores and references data.
- Use caching technology such as Azure Cache for Redis to minimize high-latency calls to remote databases for frequently accessed data.

The goal here is to minimize the network latency between each layer of the application. How this is solved depends on your application and data architecture, but Azure provides mechanisms to solve this on several services.

Latency between users and Azure resources

We've looked at the latency between our Azure resources, but we should also consider the latency between users and our cloud application. We're looking to optimize delivery of the front end-user interface to our users. Let's take a look at some ways to improve the network performance between end users and the application.

Use a DNS load balancer for endpoint path optimization

In your organizations example, we saw that the team created an additional web front-end node in Australia East. However, end users have to explicitly specify which front-end endpoint they want to use. As the designer of a solution, you want to make the experience as smooth as possible for their users.

Azure Traffic Manager could help. Traffic Manager is a DNS-based load balancer that enables you to distribute traffic within and across Azure regions. Rather than having the user browse to a specific instance of our web front end, Traffic Manager can route users based upon a set of characteristics:

- **Priority** - You specify an ordered list of front-end instances. If the one with the highest priority is unavailable, Traffic Manager will route the user to the next available instance.
- **Weighted** - You would set a weight against each front-end instance. Traffic Manager then distributes traffic according to those defined ratios.
- **Performance** - Traffic Manager routes users to the closest front-end instance based on network latency.
- **Geographic** - You could set up geographical regions for front-end deployments, routing your users based upon data sovereignty mandates or localization of content.

Traffic Manager profiles can also be nested. You could first route your users across different geographies (for example, Europe and Australia) using geographic routing and then route them to local front-end deployments using the performance routing method.

Consider that you have deployed a web front end in West Europe and Australia. Assume they have deployed Azure SQL Database with their primary deployment in West Europe, and a read replica in Australia East. Let's also assume the application can connect to the local SQL instance for read queries.

The team deploy a Traffic Manager instance in performance mode and add the two front-end instances as Traffic Manager profiles. As an end user, you navigate to a custom domain name (for example, lamna-healthcare.com) which routes to Traffic Manager. Traffic Manager then returns the DNS name of the West Europe or Australia East front end based on the best network latency performance.

It's important to note that this load balancing is only handled via DNS, there's no inline load balancing or caching that's happening here, Traffic Manager is simply returning the DNS name of the closest front end to the user.

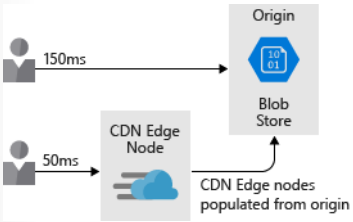
Use Azure Analysis Services as a Caching Service for Azure Data Warehouse

Azure Azure Synapse Analytics is limited with 1024 concurrency connections of and 64 active query connections. If there is a requirement to service more connections than the limits, you can consider populating Azure Analysis Services with data from the Data Warehouse to act as a caching service. This works very well in scenarios where the refresh of the warehouse is limited to hourly or daily refreshes.

Use CDN to cache content close to users

The website will likely be using some form of static content (either whole pages or assets such as images and videos). This content could be delivered to users faster by using a content delivery network (CDN) such as Azure CDN.

When Lamna deploys content to Azure CDN, those items are copied to multiple servers around the globe. Let's say one of those items is a video served from blob storage: `HowToCompleteYourBilling-Forms.MP4`. The team then configure the website so that each user's link to the video will actually reference the CDN edge server nearest them, rather than referencing blob storage. This approach puts content closer to the destination, reducing latency and improving user experience. The following illustration shows how using Azure CDN puts content closer to the destination which reduces latency and improves the user experience.



Content delivery networks *can* also be used to host cached dynamic content. Extra consideration is required, though, since cached content may be out of date compared with the source. Context expiration can be controlled by setting a time to live (TTL). If the TTL is too high, out-of-date content may be displayed and the cache would need to be purged.

One way to handle cached content is with a feature called **dynamic site acceleration**, which can increase performance of webpages with dynamic content. Dynamic site acceleration can also provide a low-latency path to additional services in your solution (for example, an API endpoint).

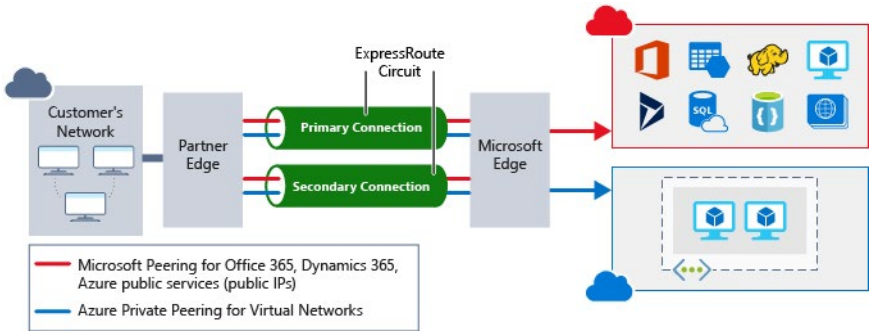
Use ExpressRoute for connectivity from on-premises to Azure

Optimizing network connectivity from your on-premises environment to Azure is also important. For users connecting to applications, whether they're hosted on virtual machines or on PaaS offerings like Azure App Service, you'll want to ensure they have the best connection to your applications.

You can always use the public internet to connect users to your services, but internet performance can vary and may be impacted by outside issues. On top of that, you may not want to expose all of your services over the internet, and you may want a private connection to your Azure resources.

Site-to-site VPN over the internet is also an option, but for high throughput architectures, VPN overhead and internet variability can increase latency noticeably.

Azure ExpressRoute can help. ExpressRoute is a private, dedicated connection between your network and Azure, giving you guaranteed performance and ensuring that your end users have the best path to all of your Azure resources. The following illustration shows how ExpressRoute Circuit provides connectivity between on-premises applications and Azure resources.



Once again looking at Lamna's scenario, they decide to further improve end-user experience for users who are in their facilities by provisioning an ExpressRoute circuit in both Australia East and West Europe. This gives their end users a direct connection to their booking system and ensures the lowest latency possible for their application.

Considering the impact of network latency on your architecture is important to ensure the best possible performance for your end users. We've taken a look at some options to lower network latency between end users and Azure and between Azure resources.

Optimizing Storage Performance

It's important to include storage performance considerations in your architecture. Just like network latency, poor performance at the storage layer can impact your end-users' experience. How would you optimize your data storage? What things do you need to consider to ensure that you're not introducing storage bottlenecks into your architecture? Here, we'll take a look at how to optimize your storage performance in your architecture.

Optimize virtual machine storage performance

Let's first look at optimizing storage for virtual machines. Disk storage plays a critical role in the performance of your virtual machines, and selecting the right disk type for your application is an important decision.

Different applications are going to have different storage requirements. Your application may be sensitive to latency of disk reads and writes or it may require the ability to handle a large number of input/output operations per second (IOPS) or greater overall disk throughput.

When building an IaaS workload, which type of disk should you use? There are four options:

- **Local SSD storage** - Each VM has a temporary disk that is backed by local SSD storage. The size of this disk varies depending on the size of the virtual machine. Since this disk is local SSD, the performance is high, but data may be lost during a maintenance event or a redeployment of the VM. This disk is only suitable for temporary storage of data that you do not need permanently. This disk is great for the page or swap file, and for things like tempdb in SQL Server. There is no charge for this storage. It's included in the cost of the VM.
- **Standard storage HDD** - This is spindle disk storage and may fit well where your application is not bound by inconsistent latency or lower levels of throughput. A dev/test workload where guaranteed performance isn't needed is a great use case for this disk type.
- **Standard storage SSD** - This is SSD backed storage and has the low latency of SSD but lower levels of throughput. A non-production web server would be a good use case for this disk type.
- **Premium storage SSD** - This SSD backed storage is well-suited for those workloads that are going into production, require the greatest reliability and demand consistent low latency, or need high levels of throughput and IOPS. Since these disks have greater performance and reliability capabilities, they are recommended for all production workloads.

Premium storage can attach only to specific virtual machine (VM) sizes. Premium storage capable sizes are designated with an “s” in the name, for example D2**s**_v3 or Standard_F2**s**_v2. Any virtual machine type (with or without an "s" in the name) can attach standard storage HDD or SSD drives.

Disks can be striped using a striping technology (such as Storage Spaces Direct on Windows or mdadm on Linux) to increase the throughput and IOPS by spreading disk activity across multiple disks. Using disk striping allows you to really push the limits of performance for disks, and is often seen in high-performance database systems and other systems with intensive storage requirements.

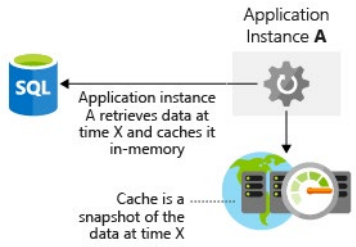
When relying on virtual machine workloads, you'll need to evaluate the performance requirements of your application to determine the underlying storage you'll provision for your virtual machines.

Optimize storage performance for your application

While you can use differing storage technologies to improve the raw disk performance, you can also address the performance of access to data at the application layer. Let's take a look at a few ways you can do this.

Caching

A common approach to improve application performance is to integrate a caching layer between your application and your data store. A cache typically stores data in memory and allows for fast retrieval. This data can be frequently accessed data, data you specify from a database, or temporary data such as user state. You'll have control over the type of data stored, how often it refreshes, and when it expires. By co-locating this cache in the same region as your application and database, you'll reduce the overall latency between the two. Pulling data out of the cache will almost always be faster than retrieving the same data from a database, so by using a caching layer you can substantially improve the overall performance of your application. The following illustration shows how an application retrieves data from a database, stores it in a cache, and uses the cached value as needed.

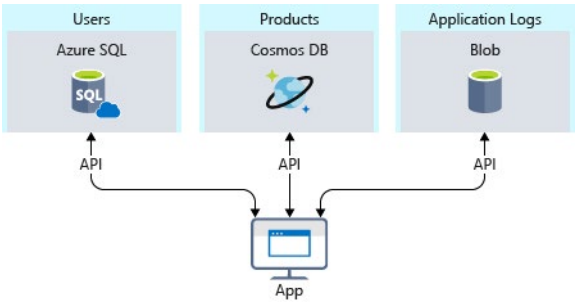


Azure Cache for Redis is a caching service on Azure that stores data in memory. It's based upon the open-source Redis cache and is a fully managed service offering by Microsoft. You select the performance tier that you require and configure your application to use the service.

Polyglot persistence

Polyglot persistence is the usage of different data storage technologies to handle your storage requirements.

Consider an e-commerce example. You may store application assets in a blob store, product reviews and recommendations in a NoSQL store, and user profile or account data in a SQL database. The following illustration shows how an application might use multiple data storage techniques to store different types of data.



This is important, as different data stores are designed for certain use cases or may be more accessible because of cost. As an example, storing blobs in a SQL database may be costly and slower to access than directly from a blob store.

Maintaining data consistency across distributed data stores can be a significant challenge. The issue is that strategies such as serialization and locking only work well if all application instances share the same data store, and the application is designed to ensure that the locks are very short-lived. However, if data is partitioned or replicated across different data stores, locking and serializing data access to maintain consistency can become an expensive overhead that impacts the throughput, response time, and scalability of a system. Therefore, most modern distributed applications do not lock the data that they modify, and they take a rather more relaxed approach to consistency, known as eventual consistency.

Eventual consistency means that replica data stores will eventually converge if there are no further writes. If a write is made to one of the data stores, reads from another may provide slightly out-of-date data. Eventual consistency enables higher scale because there is a low latency for reads and writes, rather than waiting to check if information is consistent across all stores.

Performance Bottlenecks

End users are expecting more from their applications. They want to have a great user experience and not be impacted by performance issues. How do you integrate performance bottleneck identification into your architecture? In this unit, we will look at both processes and tools that can help ensure that your application performs well, and help you track down why if it doesn't.

Importance of requirements

Before we talk about performance, it's important to talk about requirements. In theory, we could keep improving scalability and performance further and further without end. At some point, however, more improvement is prohibitively expensive, difficult, and doesn't have enough business impact to be worthwhile.

Our **non-functional requirements** help us find that point. These particular requirements don't tell us what our app must *do*. Instead, they tell us what quality levels it must meet. As examples, we can define these non-functional requirements to tell us:

- How fast a transaction must return under a given load.
- How many simultaneous connections we need to support before we start returning errors.
- In the event of server failure, what is the maximum amount of time our application is allowed be down before a back-up is online.

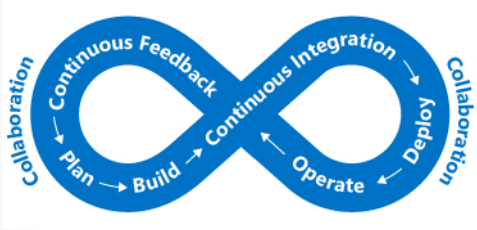
Defining these requirements in advance of building your solution is critical to ensure that the application meets expectations but doesn't require more effort or expend more money than necessary. We can also plan our monitoring and operations rules around these non-functional requirements.

Discuss requirements with your stakeholders or customers, document them, and communicate them broadly to ensure that everyone agrees on what "good performance" means.

DevOps and application performance

The idea behind DevOps is that we don't have development and infrastructure silos in our organization. Instead, they work together to effectively build, deploy, monitor, and maintain apps in streamlined process.

The planning, development, testing, and monitoring is carried out in an iterative approach. Performance and quality of our application become a part of our software development life cycle, rather than an afterthought as we deploy into a live environment. The following illustration shows where opportunities for collaboration exist in the software development lifecycle.



This approach aligns with a DevOps concept called "shifting left". In other words, bring your quality control checks earlier into your deployment and release process. This allows you to catch end-user impacting issues earlier in the process. As we operate in a continuous cycle, we limit the amount of manual interaction and automate as much as possible.

One way we make performance part of our DevOps process is to carry out performance or load tests to validate that the application meets the non-functional requirements prior to a deployment into production.

Ideally, we could carry out performance and load tests in an environment that is exactly like production while not impacting our actual production servers. When leveraging the cloud, you fully have this ability. You can automate the creation of a production-like environment, perform testing, and then destroy the environment to minimize cost. This approach to automation can provide reassurance that your application can handle the scale you require now, as well as respond to future growth.

Application performance monitoring becomes a core part of this. If we're running performance and load tests on our application or want to keep our production performance in check, we want to understand what parts of our application may be performing non-optimally. Let's take a look at some ways to do this.

Performance monitoring options in Azure

Monitoring is the act of collecting and analyzing data to determine the performance, health, and availability of your business application and associated resources.

We want to be kept informed that our application is running smoothly. Proactive notifications can be used to inform about critical issues that arise. There are many layers of monitoring to consider, mainly the infrastructure layer and the application layer.

Azure Monitor

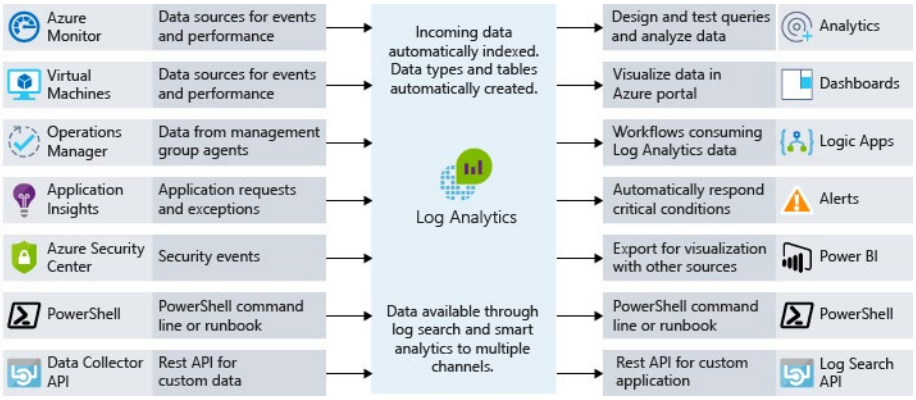
Azure Monitor provides a single management point for infrastructure-level logs and monitoring for most of your Azure services. It collects metrics, activity logs, and diagnostic logs and more. Azure Monitor provides us with a range of features including:

- Azure alerts to proactively notify or take action on any breaches to metrics or activities arising.
- Use Azure Dashboards to combine many monitoring sources into one view of our application.

Azure Monitor is the place to start for all your near real-time resource metric insights. Many Azure resources will start outputting metrics automatically once deployed. For example, Azure Web App instances will output compute and application request metrics. Metrics from Application Insights are also collated here in addition to VM host diagnostic metrics. VM guest diagnostic metrics will also appear once you opt in.

Log Analytics

Centralized logging can help you uncover hidden issues that may be difficult to track down. With Log Analytics you can query and aggregate data across logs. This cross-source correlation can help you identify issues or performance problems that may not be evident when looking at logs or metrics individually. The following illustration shows how Log Analytics acts as a central hub for monitoring data. Log Analytics receives monitoring data from your Azure resources and makes it available to consumers for analysis or visualization.



You can collate a wide range of data sources, security logs, Azure activity logs, server, network, and application logs. You can also push on-premises System Center Operations Manager data to Log Analytics in hybrid deployment scenarios and have Azure SQL Database send diagnostic information directly into Log Analytics for detailed performance monitoring.

Centralized logging can be massively beneficial for troubleshooting all types of scenarios, including performance issues. It's a key part of a good monitoring strategy for any architecture.

Application performance management

Deep application issues are often tricky to track down. This is where integrating telemetry into an application by using an application performance management solution (APM) to track down low-level application performance and behavior can be beneficial. This telemetry can include individual page request times, exceptions within your application, and even custom metrics to track business logic. This telemetry can provide a wealth of insight into what is going on within your application.

On Azure, Application Insights is a service that provides this deep application performance management. You install a small instrumentation package in your application, and set up an Application Insights resource in the Microsoft Azure portal. The instrumentation monitors your app and sends telemetry data to the portal.

Telemetry from the host environments, such as performance counters, Azure diagnostics, and Docker logs, can be ingested. You can also set up web tests that periodically send synthetic requests to your web service. You could even configure your application to send custom events and metrics that you write yourself in the client or server code. For example, application-specific events such as items sold or games won.

Application Insights stores its data in a common repository, and metrics are shared with Azure Monitor. It can take advantage of shared functionality such as alerts, dashboards, and deep analysis with the Log Analytics query language.

A common pattern used in determining the availability of a web application is the health endpoint monitoring pattern. This pattern is used to monitor web applications and associated back-end services, to ensure that they're available and performing correctly. The pattern is implemented by querying a particular uri. The endpoint checks on the status of many components, including the back-end services that the app depends on, rather than just the availability of the front end itself. This acts as a service-level health check that returns an indication of the overall health of the service.

Use an APM solution such as Application Insights to gain a deep understanding of your application and correlate activity across your application. This can help you understand how a specific action works in the client browser, on the server, and through to downstream services. It will also provide insight into trends,

provide notifications when there is a problem, and help identify where the problem is and how to fix it, before your users are aware.

High Availability

High availability (HA) ensures your architecture can handle failures. Imagine you're responsible for a system that must be always fully operational. Failures can and will happen, so how do you ensure that your system can remain online when something goes wrong? How do you perform maintenance without service interruption?

Here, you'll learn the need for high availability, evaluate application high-availability requirements, and see how the Azure platform helps you meet your availability goals.

What is high availability?

A highly-available service is a service that absorbs fluctuations in availability, load, and temporary failures in dependent services and hardware. The application remains online and available (or maintains the appearance of it) while performing acceptably. This availability is often defined by business requirements, service-level objectives, or service-level agreements.

High availability is ultimately about the ability to handle the loss or severe degradation of a component of a system. This might be due to a virtual machine that's hosting an application going offline because the host failed. It could be due to planned maintenance for a system upgrade. It could even be caused by the failure of a service in the cloud. Identifying the places where your system can fail, and building in the capabilities to handle those failures, will ensure that the services you offer to your customers can stay online.

High availability of a service typically requires high availability of the components that make up the service. Think of a website that offers an online marketplace to purchase items. The service that's offered to your customers is the ability to list, buy, and sell items online. To provide this service, you'll have multiple components: a database, web servers, application servers, and so on. Each of these components could fail, so you have to identify how and where your failure points are, and determine how to address these failure points in your architecture.

Evaluate high availability for your architecture

There are three steps to evaluate an application for high availability:

1. Determine the service-level agreement of your application
2. Evaluate the HA capabilities of the application
3. Evaluate the HA capabilities of dependent applications

Let's explore these steps in detail.

Determine the service-level agreement of your application

A service-level agreement (SLA) is an agreement between a service provider and a service consumer in which the service provider commits to a standard of service based on measurable metrics and defined responsibilities. SLAs can be strict, legally bound, contractual agreements, or assumed expectations of availability by customers. Service metrics typically focus on service throughput, capacity, and availability, all of which can be measured in various ways. Regardless of the specific metrics that make up the SLA, failure to meet the SLA can have serious financial ramifications for the service provider. A common component of service agreements is guaranteed financial reimbursement for missed SLAs.

Service-level objectives (SLO) are the values of target metrics that are used to measure performance, reliability, or availability. These could be metrics defining the performance of request processing in milliseconds, the availability of services in minutes per month, or the number of requests processed per hour. By evaluating the metrics exposed by your application and understanding what customers use as a measure of quality, you can define the acceptable and unacceptable ranges for these SLOs. By defining these objectives, you clearly set goals and expectations with both the teams supporting the services and customers who are consuming these services. These SLOs will be used to determine if your overall SLA is being met.

The following table shows the potential cumulative downtime for various SLA levels.

SLA	Downtime per week	Downtime per month	Downtime per year
99%	1.68 hours	7.2 hours	3.65 days
99.9%	10.1 minutes	43.2 minutes	8.76 hours
99.95%	5 minutes	21.6 minutes	4.38 hours
99.99%	1.01 minutes	4.32 minutes	52.56 minutes
99.999%	6 seconds	25.9 seconds	5.26 minutes

Of course, higher availability is better, everything else being equal. But as you strive for more 9s, the cost and complexity to achieve that level of availability grows. An uptime of 99.99% translates to about 5 minutes of total downtime per month. Is it worth the additional complexity and cost to reach five 9s? The answer depends on the business requirements.

Here are some other considerations when defining an SLA:

- To achieve four 9's (99.99%), you probably can't rely on manual intervention to recover from failures. The application must be self-diagnosing and self-healing.
- Beyond four 9's, it is challenging to detect outages quickly enough to meet the SLA.
- Think about the time window that your SLA is measured against. The smaller the window, the tighter the tolerances. It probably doesn't make sense to define your SLA in terms of hourly or daily uptime.

Identifying SLAs is an important first step when determining the high availability capabilities that your architecture will require. These will help shape the methods you'll use to make your application highly available.

Evaluate the HA capabilities of the application

To evaluate the HA capabilities of your application, perform a failure analysis. Focus on single points of failure and critical components that would have a large impact on the application if they were unreachable, misconfigured, or started behaving unexpectedly. For areas that do have redundancy, determine whether the application is capable of detecting error conditions and self-healing.

You'll need to carefully evaluate all components of your application, including the pieces designed to provide HA functionality, such as load balancers. Single points of failure will either need to be modified to have HA capabilities integrated, or will need to be replaced with services that can provide HA capabilities.

Evaluate the HA capabilities of dependent applications

You'll need to understand not only your application's SLA requirements to your consumer, but also the provided SLAs of any resource that your application may depend on. If you are committing an uptime to your customers of 99.9%, but a service your application depends on only has an uptime commitment of 99%, this could put you at risk of not meeting your SLA to your customers. If a dependent service is unable to provide a sufficient SLA, you may need to modify your own SLA, replace the dependency with

an alternative, or find ways to meet your SLA while the dependency is unavailable. Depending on the scenario and the nature of the dependency, failing dependencies can be temporarily worked around with solutions like caches and work queues.

Azure's highly available platform

The Azure cloud platform has been designed to provide high availability throughout all its services. Like any system, applications may be affected by both hardware and software platform events. The need to design your database architecture to handle failures is critical, and the Azure cloud platform provides you with the tools and capabilities to make your application highly available.

PaaS HA capabilities

PaaS services come with high availability built in. Services such as Azure SQL Database, Azure App Service, and Azure Service Bus include high availability features and ensure that failures of an individual component of the service will be seamless to your application. Using PaaS services is one of the best ways to ensure that your architecture is highly available.

When architecting for high availability, you'll want to understand the SLA that you're committing to your customers. Then evaluate both the HA capabilities that your application has, and the HA capabilities and SLAs of dependent systems. After those have been established, use Azure features, such as availability sets, availability zones, and various load-balancing technologies, to add HA capabilities to your application. Any PaaS services you should choose to use will have HA capabilities built in.

Disaster Recovery

Designing for high availability helps keep an application or process running despite unfavorable events and adverse conditions. But what do you do when something so significant happens that you've lost data and it's impossible to keep your apps and processes from going down? When disaster strikes, you need to have a plan to get your services running again. You should know what your goals and expectations are for recovering, what are the costs and limitations of your plan, and how to execute on it.

What is disaster recovery?

Disaster recovery is about *recovering from high-impact events* that result in downtime and data loss. A disaster is a single, major event with an impact much larger and long-lasting than the application can mitigate through the high-availability portion of its design.

The word “disaster” often evokes thoughts of *natural* disasters and external events (earthquakes, floods, tropical storms, and so on) but many other kinds of disasters exist as well. A failed deployment or upgrade can leave an app in an unrecognizable state. Malicious hackers can encrypt or delete data and inflict other kinds of damage that take an app offline or eliminate some of its functionality.

Regardless of its cause, the best remedy for a disaster once it has occurred is a well-defined, tested disaster recovery plan and an application that actively supports disaster recovery efforts through its design.

How to create a disaster recovery plan

A disaster recovery plan is a single document that details the procedures that are required to recover from data loss and downtime caused by a disaster, and identifies who's in charge of directing those procedures. Operators should be able to use the plan as a manual to restore application connectivity and

recover data after a disaster occurs. A detailed, written plan that's dedicated to disaster recovery is critical to ensuring a favorable outcome. The process of creating the plan will help to assemble a complete picture of the application. The resulting written steps will promote good decision-making and follow-through in the panicked, chaotic aftermath of a disaster event.

Creating a disaster recovery plan requires expert knowledge of the application's workflows, data, infrastructure, and dependencies.

Risk assessment and process inventory

The first step in creating a disaster recovery plan is performing a risk analysis that examines the impact of different kinds of disasters on the application. The exact nature of a disaster isn't as important to the risk analysis as its potential impact through data loss and application downtime. Explore various kinds of hypothetical disasters and try to be specific when thinking about their effects. For example, a targeted malicious attack may modify code or data that results in a different kind of impact than an earthquake that disrupts network connectivity and datacenter availability.

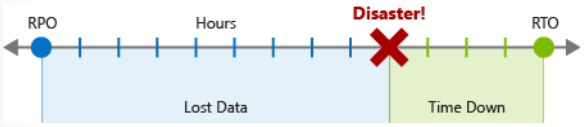
The risk assessment needs to consider *every* process that can't afford unlimited downtime, and every category of data that can't afford unlimited loss. When a disaster that affects multiple application components occurs, it's critical that the plan owners can use the plan to take a complete inventory of what needs attention and how to prioritize each item.

Some apps may only constitute a single process or classification of data. This is still important to note, as the application will likely be one component of a larger disaster recovery plan that includes multiple applications with the organization.

Recovery objectives

A complete plan needs to specify two critical business requirements for each process implemented by the application:

- **Recovery Point Objective (RPO)**: The maximum duration of acceptable data loss. RPO is measured in units of time, not volume: “30 minutes of data”, "four hours of data", and so on. RPO is about limiting and recovering from data *loss*, not data *theft*.
- **Recovery Time Objective (RTO)**: The maximum duration of acceptable downtime, where “downtime” needs to be defined by your specification. For example, if the acceptable downtime duration is eight hours in the event of a disaster, then your RTO is eight hours.



Each major process or workload that's implemented by an app should have separate RPO and RTO values. Even if you arrive at the same values for different processes, each one should be generated through a separate analysis that examines disaster scenario risks and potential recovery strategies for each respective process.

The process of specifying an RPO and RTO is effectively the creation of disaster recovery requirements for your application. It requires establishing the priority of each workload and category of data and performing a cost-benefit analysis. The analysis includes concerns, such as implementation and maintenance cost, operational expense, process overhead, performance impact, and the impact of downtime and lost data. You'll need to define exactly what “downtime” means for your application, and in some cases, you may establish separate RPO and RTO values for different levels of functionality. Specifying RPO and RTO

should be more than simply choosing arbitrary values. Much of the value of a disaster recovery plan comes from the research and analysis that goes into discovering the potential impact of a disaster and the cost of mitigating the risks.

Detailing recovery steps

The final plan should go into detail about exactly what steps should be taken to restore lost data and application connectivity. Steps often include information about:

- **Backups**: How often they're created, where they're located, and how to restore data from them.
- **Data replicas**: The number and locations of replicas, the nature and consistency characteristics of the replicated data, and how to switch over to a different replica.
- **Deployments**: How deployments are executed, how rollbacks occur, and failure scenarios for deployments.
- **Infrastructure**: On-premises and cloud resources, network infrastructure, and hardware inventory.
- **Dependencies**: External services that are used by the application, including SLAs and contact information.
- **Configuration and notification**: Flags or options that can be set to gracefully degrade the application, and services that are used to notify users of application impact.

The exact steps that are required will depend heavily on implementation details of the app, making it important to keep the plan updated. Routinely testing the plan will help identify gaps and outdated sections.

Designing for disaster recovery

Disaster recovery is not an automatic feature. It must be designed, built, and tested. An app that needs to support a solid disaster recovery strategy must be built from the ground up with disaster recovery in mind. Azure offers services, features, and guidance to help you create apps that support disaster recovery, but it's up to you to include them in your design.

Designing for disaster recovery has two main concerns:

- **Data recovery**: Using backups and replication to restore lost data.
- **Process recovery**: Recovering services and deploying code to recover from outages.

Data recovery and replication

Replication duplicates stored data between multiple data store replicas. Unlike *backup*, which creates long-lived, read-only snapshots of data for use in recovery, replication creates real-time or near-real-time copies of live data. The goal of replication is to keep replicas synchronized with as little latency as possible while maintaining application responsiveness. Replication is a key component of designing for high availability and disaster recovery, and is a common feature of production-grade applications.

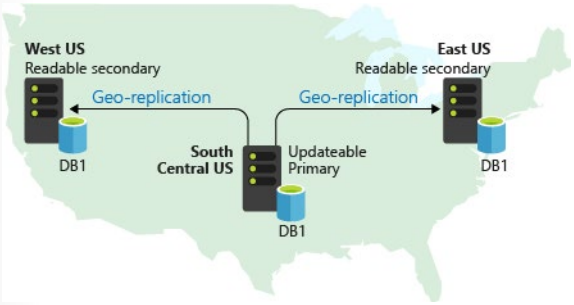
Replication is used to mitigate a failed or unreachable data store by executing a *failover*: changing application configuration to route data requests to a working replica. Failover is often automated, triggered by error detection built into a data storage product, or detection that you implement through your monitoring solution. Depending on the implementation and the scenario, failover may need to be manually executed by system operators.

Replication is not something you implement from scratch. Most fully featured database systems and other data storage products and services include some kind of replication as a tightly integrated feature due to its functional and performance requirements. However, it's up to you to include these features in your application design and make appropriate use of them.

Different Azure services support various levels and concepts of replication. For example:

- **Azure Storage** replication capabilities depend on the replication type of selected for the storage account. This replication can be local (within a datacenter), zonal (between data centers within a region), or regional (between regions). Neither your application nor your operators interact with it directly. Failovers are automatic and transparent, and you simply need to select a replication level that balances cost and risk.
- **Azure SQL Database** replication is automatic at a small scale, but recovery from a full Azure data-center or regional outage requires geo-replication. Setting up geo-replication is manual, but it's a first-class feature of the service and well supported by documentation.
- **Cosmos DB** is a globally distributed database system, and replication is central to its implementation. With Azure Cosmos DB, instead of configuring replication directly, you configure options related to partitioning and data consistency. In addition, Azure Cosmos DB automatically takes backups of your data at regular intervals. The automatic backups are taken without affecting the performance or availability of the database operations. All the backups are stored separately in a storage service
- **Azure Synapse Analytics** A data warehouse snapshot creates a restore point you can leverage to recover or copy your data warehouse to a previous state. Since Azure Synapse Analytics is a distributed system, a data warehouse snapshot consists of many files that are located in Azure storage. Snapshots capture incremental changes from the data stored in your data warehouse.

Many different replication designs exist that place different priorities on data consistency, performance, and cost. *Active* replication requires updates to take place on multiple replicas simultaneously, guaranteeing consistency at the cost of throughput. In contrast, *passive* replication performs synchronization in the background, removing replication as a constraint on application performance, but increasing RPO. *Active-active* or *multi-master* replication enables multiple replicas to be used simultaneously, enabling load balancing at the cost of complicating data consistency, while *active-passive* replication reserves replicas for live use only during failover.



IMPORTANTNeither replication nor backup are complete disaster recovery solutions on their own. Data recovery is only one component of disaster recovery, and replication will not fully satisfy many kinds of disaster recovery scenarios. For example, in a data corruption scenario, the nature of the corruption may allow it to spread from the primary data store to the replicas, rendering all the replicas useless and requiring a backup for recovery.

Process recovery

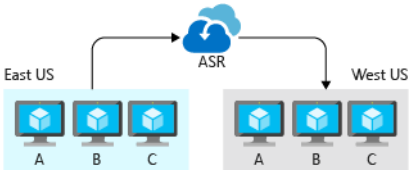
After a disaster, business data isn't the only asset that needs recovering. Disaster scenarios will also commonly result in downtime, whether it's due to network connectivity problems, datacenter outages, or damaged VM instances or software deployments. The design of your database or application needs to enable you to restore it to a working state.

In most cases, process restoration involves failover to a separate, working deployment. Depending on the scenario, geographic location may be a critical aspect. For example, a large-scale natural disaster that brings an entire Azure region offline will necessitate restoring service in another region. Your application's disaster recovery requirements, especially RTO, should drive your design and help you decide how many replicated environments you should have, where they should be located, and whether they should be maintained in a ready-to-run state or should be ready to accept a deployment in the event of disaster.

Depending on the design of your application, there are a few different strategies and Azure services and features that you can take advantage of to improve your app's support for process recovery after a disaster.

Azure Site Recovery

Azure Site Recovery is a service that's dedicated to managing process recovery for workloads running on VMs deployed to Azure, VMs running on physical servers, and workloads running directly on physical servers. Site Recovery replicates workloads to alternate locations and helps you to failover when an outage occurs and supports testing of a disaster recovery plan.

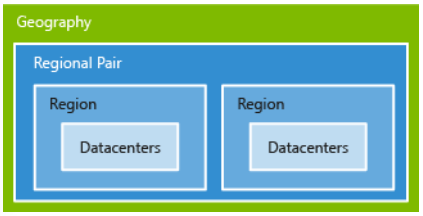


Site Recovery supports replicating whole VMs and physical server images as well as individual *workloads*, where a workload may be an individual application or an entire VM or operating system with its applications. Any application workload can be replicated, but Site Recovery has first-class integrated support for many Microsoft server applications, such as SQL Server and SharePoint, as well as a handful of third-party applications like SAP.

Any app that runs on VMs or physical servers should at least investigate the use of Azure Site Recovery. It's a great way to discover and explore scenarios and possibilities for process recovery.

Service-specific features

For apps that run on Azure PaaS offerings like App Service, most such services offer features and guidance for supporting disaster recovery. For certain scenarios, you can use service-specific features to support fast recovery. For example, Azure SQL Server supports geo-replication for quickly restoring service in another region. Azure App Service has a Backup and Restore feature, and the documentation includes guidance for using Azure Traffic Manager to support routing traffic to a secondary region.



Testing a disaster recovery plan

Disaster recovery planning doesn't end once you have a completed plan in hand. Testing the plan is a crucial aspect of disaster recovery, to ensure that the directions and explanations are clear and up-to-date.

Choose intervals to perform different types and scopes of tests, such as testing backups and failover mechanisms every month, and performing a full-scale disaster recovery simulation every six months. Always follow the steps and details exactly as they're documented in the plan, and consider having someone unfamiliar with the plan give perspective on anything that could be made clearer. As you execute the test, identify gaps, areas of improvement, and places to automate and add these enhancements to your plan.

Make sure to include your monitoring system in your testing as well. For example, if your application supports automated failover, introduce failures in a dependency or other critical component to ensure that the application behaves correctly end-to-end, including detection of the failure and triggering of the automated failover.

By carefully identifying your requirements and laying out a plan, you'll be able to determine what types of services you'll need to use to meet your recovery objectives. Azure provides several services and features to help you meet these objectives.

Backup and Restore

Backup is the final and most powerful line of defense against permanent data loss. An effective backup strategy requires more than simply making copies of data. It needs to take your application's data architecture and infrastructure into consideration. Your app may manage many kinds of data of varying importance, spread widely across filesystems, databases, and other storage services both in the cloud and on-premises. Using the right services and products for the job will simplify your backup process and increase recovery time if a backup needs to be restored.

Establish backup and restoration requirements

As with a disaster recovery strategy, backup requirements are based on a cost-benefit analysis. Analysis of your app's data should be guided by the relative importance of the different categories of data the app manages, as well as external requirements, such as data retention laws.

To establish backup requirements for your app, group your application's data based on the following requirements:

- How much of this type of data can afford to be lost, measured in duration
- The maximum amount of time a restore of this type of data should require
- Backup retention requirements: how long and at what frequency do backups need to remain available

These concepts map neatly to the concepts of Recovery Point Objective and Recovery Time Objective (RPO and RTO). The duration of acceptable loss will generally translate directly to required backup intervals and RPO. The maximum amount of time a restore takes corresponds to the RTO for the data component of your application. Both requirements should be developed relative to the cost of achieving them. Every organization would like to say that they truly can't afford to lose *any* data, but often that's not the case when the cost of achieving that requirement is considered.

Backup absolutely plays a role in disaster recovery (DR), but backups, restores and their associated scenarios extend beyond the scope of DR. Backups may need to be restored in non-disaster situations, including those where RTO and RPO aren't of great concern. For example, if a small amount of data older than your backup interval is corrupted or deleted, but the application doesn't experience downtime, your application may never be in danger of missing its SLA and a successful restore will result in no data lost. Your disaster recovery plan may or may not include guidance on performing restores in non-disaster situations.

TIP:
Don't confuse *archival*, *replication*, and *backup*. Archival is the storage of data for long-term preservation and read access. Replication is the near-real-time copying of data between replicas to support high availability and certain disaster recovery scenarios. Some requirements, such as data retention laws, may influence your strategies for all three of these concerns. Archival, replication, and backup all require separate analysis and implementation.

Azure backup and restore capabilities

Azure offers several backup-related services and features for various scenarios, including data in Azure as well as on-premises data. Most Azure services offer some kind of backup functionality. Here, we'll look at a few of the most popular backup-related Azure offerings.

Azure Backup

Azure Backup is a family of backup products that back up data to Azure Recovery Services vaults for storage and recovery. Recovery Service vaults are storage resources in Azure that are dedicated to holding data and configuration backups for virtual machines, servers, and individual workstations and workloads.

NOTE
Both Azure Backup and Azure Site Recovery use Azure Recovery Service vaults for storage. Azure Backup is a general-purpose backup solution. Azure Site Recovery can coordinate replication and failover and support low RPO and RTO disaster recovery operations.

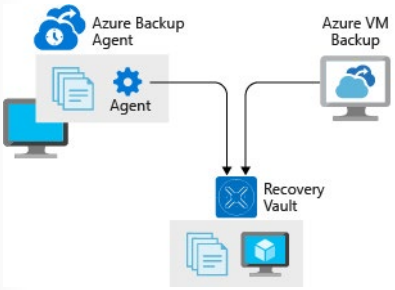
Azure Backup serves as a general-purpose backup solution for cloud and on-premises workflows that run on VMs or physical servers. It's designed to be a drop-in replacement for traditional backup solutions that stores data in Azure instead of archive tapes or other local physical media.

Four different products and services can use Azure Backup to create backups:

- **Azure Backup Agent** is a small Windows application that backs up files, folders, and system state from the Windows VM or server on which it's installed. It works in a way that's similar to many consumer cloud-based backup solutions, but requires configuration of an Azure Recovery vault. Once you download and install it onto a Windows server or VM, you can configure it to create backups up to three times a day.
- **System Center Data Protection Manager** is a robust, fully featured, enterprise-level backup and recovery system. Data Protection Manager is a Windows Server application that can back up filesystems and virtual machines (Windows and Linux), create bare-metal backups of physical servers, and

perform application-aware backup of many Microsoft server products, such as SQL Server and Exchange. Data Protection Manager is part of the System Center family of products and is licensed and sold with System Center, but it's considered part of the Azure Backup family because it can store backups in an Azure Recovery vault.

- **Azure Backup Server** is similar to Data Protection Manager, but it's licensed as part of an Azure subscription and doesn't require a System Center license. Azure Backup Server supports the same functionality as Data Protection Manager except for local tape backup and integration with the other System Center products.
- **Azure IaaS VM Backup** is a turnkey backup and restore feature of Azure Virtual Machines. VM backup supports once-per-day backups for Windows and Linux virtual machines. It supports recovery of individual files, full disks, and entire VMs, and can also perform application-consistent backups. Individual applications can be made aware of backup operations and get their filesystem resources into a consistent state before the snapshot is taken.



Azure Backup can add value and contribute to the backup and restore strategy for IaaS and on-premises applications of virtually any size and shape.

Azure Blob storage

Azure Storage doesn't include an automated backup feature, but blobs are commonly used to back up all kinds of data from various sources. Many services that provide backup capabilities use blobs to store their data, and blobs are a common target for scripts and tools in every kind of backup scenario.

General Purpose v2 storage accounts support three different blob storage tiers of varying performance and cost. **Cool** storage offers the best cost-to-performance ratio for most backups, as opposed to **hot** storage, which offers lower access costs but higher storage costs. **Archive**-tier storage may be appropriate for secondary backups or backups of data with low expectations for recovery time. It's low in cost, but requires up to 15 hours of lead time to access.

Immutable blob storage is configurable to be non-erasable and non-modifiable for a user-specified interval. Immutable blob storage was designed primarily to fulfill strict requirements for certain kinds of data, such as financial data. It's a great option for ensuring that backups are protected against accidental deletion or modification.

Azure SQL Database

Comprehensive, automatic backup functionality is included with Azure SQL Database at no extra charge. Full backups are created weekly, with differential backups performed every 12 hours, and log backups created every five minutes. Backups created by the service can be used to restore a database to a specific point in time, even if it's been deleted. Restores can be performed using the Azure portal, PowerShell, or the REST API. Backups for databases encrypted with Transparent Data Encryption, enabled by default, are also encrypted.

SQL Database backup is enterprise-grade, production ready, and enabled by default. If you're evaluating different database options for an app, it should be included as part of cost-benefit analysis, as it's a significant benefit of the service. Every app that uses Azure SQL Database should take advantage of it by including it in their disaster recovery plan and backup/restore procedures.

Azure App Service

Web applications hosted in the Azure App Service Standard and Premium tiers support turnkey scheduled and manual backups. Backups include configuration and file contents as well as contents of databases used by the app. They also support simple filters for excluding files. Restore operations can target different App Service instances, making App Service back up a simple way to move one app's contents to another.

App Service backups are limited to 10 GB total, including app and database content. They're a good solution for new apps under development and small-scale apps. More mature applications won't generally use App Service backup. They will instead rely on robust deployment and rollback procedures, storage strategies that don't use application disk storage, and dedicated backup strategies for databases and persistent storage.

Verify backups and test restore procedures

No backup system is complete without a strategy for verifying backups and testing restore procedures. Even if you use a dedicated backup service or product, you should still document and practice recovery procedures to ensure that they're well-understood and return the system to the expected state.

Strategies for verifying backups vary and will depend on the nature of your infrastructure. You may want to consider techniques, such as creating a new deployment of the application, restoring the backup to it, and comparing the state of the two instances. In many cases, this technique closely mimics actual disaster recovery procedures. Simply performing a comparison of a subset of the backup data with the live data immediately after creating a backup is enough. A common component of backup verification is attempting to restore old backups to ensure that they're still available and operational, and that the backup system hasn't changed in a way that renders them incompatible.

Any strategy is better than finding out that your backups are corrupted or incomplete while attempting to recover from a disaster.

A backup and restore strategy is an important part of ensuring your architecture can recover from the loss or corruption of data. Review your architecture to define your backup and restore requirements. Azure provides several services and features to provide backup and restore capabilities to any architecture.

Module Summary

Summary

Congratulations, you have completed the module on Performance and Scalability. Let's recap what we have covered:

Scaling up and scaling out

- The difference between scaling up/down (the level of resource provisioned on an instance) and scaling in/out (increasing or decreasing the number of available instances). We also discussed examples in Azure of each.
- The considerations to be made when scaling in/out around state management and application startup times.
- Autoscale, and how it can help you scale the number of instances based on a schedule or the demand of an application.
- Discussed alternate technologies that may help with scalability, including serverless and containers.

Optimize network performance

- Network latency is a measure in delay of data being sent from a sender to a receiver.
- In a cloud environment, chattier applications may see a performance impact compared to on-premises as resources are no longer immediately co-located.
- Co-locating APIs near to a database write endpoint could provide a performance benefit, as we are reducing the network latency between the two resources.
- Azure Traffic Manager could be used to route users to the deployed instance with the lowest network latency.
- Azure Content Delivery Networks (CDN) could be used to offload compute from the main application and speed up application load times by caching content on a CDN edge node near to a user.

Optimize storage performance

- There are three main types of disks available for IaaS deployments. Standard HDD disks (inconsistent latency and lower levels of throughput), Standard SSD disks (consistent latency and lower levels of throughput), and Premium SSD disks (consistent latency and high levels of throughput).
- Caching could be used in the application layer to improve the load times of an application. Frequently requested information could be stored in a cache in front of the database, which could then optimize for data load times of the most requested information.
- Using the appropriate back-end data store for the job (polyglot persistence) should be considered when building out your solution.

Identify performance bottlenecks

- Importance of understanding the expectations of the application before architecting or building any operations.

- Understanding how effective DevOps strategies can help build more robust and well-performing applications.
- Summarized the monitoring options available in Azure, including Azure Monitor (pane of glass for monitoring on Azure), Azure Log Analytics (log ingestion and IaaS monitoring), and Application Insights (application performance monitoring including availability, performance, and exception information).

High availability

- High availability is the ability to handle the loss or severe degradation of a component of a system.
- Evaluate your application for high availability by focusing on three key areas:
    - Your defined SLA.
    - The HA capabilities of your application.
    - The HA capabilities of dependent applications.
- Use availability sets and availability zones on Azure to provide HA for VM workloads.
- Use load balancing services such as Azure Traffic Manager, Azure Application Gateway, and Azure Load Balancer to distribute load across available systems.
- PaaS services have HA built in, so leverage these services in your architecture where possible.

Disaster recovery

- Disaster recovery is about *recovering from high-impact events* that result in downtime and loss of data.
- Create a disaster recovery plan to define the procedures, responsibilities, and activities needed to recover from a disaster.
- Define the RPO and RTO for your application to help determine the DR requirements for your application.
- Use backup and replication to provide copies of your systems and data to recover to.
- Use Azure Site Recovery for DR process recovery capabilities for your application.
- Test your disaster recovery plan to identify gaps and relevance of steps.

Backup and restore

- Use backups to restore your data as part of your DR strategy or for more isolated data loss scenarios.
- Use Azure Backup for full VM backup, file and folder backup, and backup of systems in an on-premises environment.
- Backup capabilities are often built into PaaS services, such as Azure SQL Database and Azure App Service. Take advantage of these backup capabilities when possible, understand their default configuration and overall capabilities.
- Test your restoration processes and procedures regularly to ensure they are valid and sufficient.

Module 6 Designing for Efficiency and Operations

Module Introduction

Designing for Efficiency and Operation

How efficient is your cloud environment? Are you maximizing your resources and minimizing your cloud spend? Are you able to assess the health of your infrastructure? Are adverse events visible and actionable? Are you provisioning resources manually, or have you automated them to reduce risk and increase efficiency? Here, you will learn how to answer these questions and improve the efficiency of your Azure cloud deployments.

As we learn about improving operational efficiency, we'll see how one fictional Azure customer puts these principles to work. Lamna Healthcare is a national healthcare provider with several thousand physicians and clinicians across multiple facilities throughout the country. Their IT organization has recently undertaken an effort to reduce their datacenter footprint and move the majority of their IT systems to Azure. They have a mixture of in-house developed applications, open source, and off-the-shelf applications, with varying architectures and technology platforms. They want to make their journey to the cloud successful and would like to learn what they need to focus on to help make it possible.

NOTE:
The concepts discussed in this module are not all-inclusive, but represent some of the important considerations when building a solution on the cloud. Microsoft publishes a broad set of patterns, guidelines, and examples on designing applications on Azure. It is highly recommended that you look through the content in the **Azure Architecture Center**[1] as you start planning and designing your architecture.

Learning objectives

In this module, you will:

- Maximize the efficiency of your cloud environment
- Use monitoring and analytics to gain operational insights

1 https://docs.microsoft.com/azure/architecture/

- Use automation to reduce effort and error

Designing for Effciency and Operations

Maximize the Efficiency of your Cloud Spend

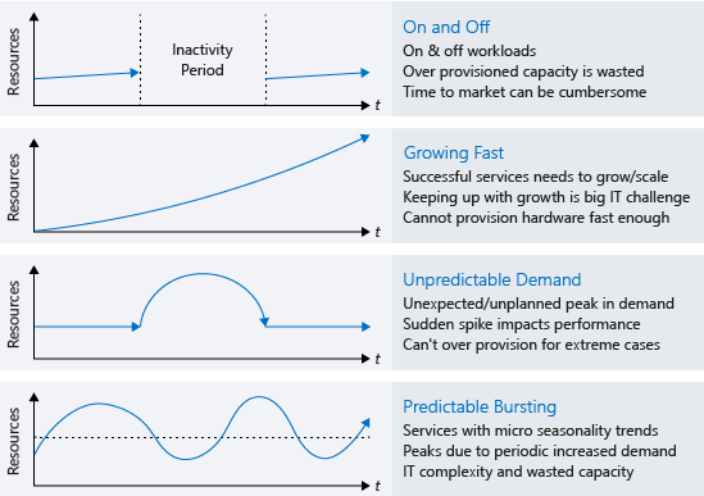
You are a solution architect. Your organization, Lamna Healthcare has moved its workloads to the cloud. Recently, the bill for these resources and workflows has increased more than Lamna had anticipated. You have been asked to determine whether the increase is natural, efficient growth, or whether the cost can be reduced by being more efficient with the organization's cloud resources.

How the cloud changes your expenses

One of the differences between the public cloud and on-premises infrastructure is how you pay for the services you use. In an on-premises datacenter, hardware procurement time is long, hardware is sized for maximum capacity, and some of the costs, such as power and space, can be hidden from the business unit consuming the resources. Purchasing physical infrastructure ties up investments in long-term assets, hindering your ability to be agile with your resources.

Shifting to the cloud introduces a pay-for-what-you-use cost model. You no longer need to tie up investments in assets, and if your resource requirements change, you can respond by adding, moving, or removing resources. Workloads vary between and within services, demand can be unpredictable, and your growth patterns shift over time. Since you only pay for what you use in the cloud, your cost structure can move in sync with the changes in resources.

Cloud infrastructure can handle fluctuating resource usage scenarios. Resources that have significant periods of inactivity can be shut down when not in use and not incur any cost at all. Resources can grow with a successful service as it grows, rather than having to wait for the next procurement cycle. More resources can be dynamically added and removed to respond to predictable and unpredictable bursts of demand. The following illustration shows why the on-premises infrastructure cannot handle all these fluctuating scenarios.



In an efficient architecture, provisioned resources match the demand for those resources. If a virtual machine is less than 10% utilized the majority of the time, you are wasting resources, both in compute and cost. Conversely, a virtual machine that is running 90% utilized is using the majority of the available resources and is an efficient use of money. Running a system to 100% utilization runs the risk of introducing performance issues. It is important to ensure that maximizing efficiency doesn't negatively impact the

performance of your system. Demand is rarely constant, so adjusting resources when possible to match demand is important to ensure efficiency.

Track your cloud spend

In order to make intelligent decisions, you need data. By looking at where your money is going, you can start comparing that to utilization to uncover where you may have waste within your environment.

An export of your billing data is available at any time. Using your billing data, you can track where your costs are going and how they're allocated across your resources. The challenge is that the billing data shows cost but not utilization. You'll have data that indicates you're paying for that large VM, but how much are you actually using it?

Azure Cost Management gives you insights where your spend is going, as well as underutilized resources. Azure Cost Management tracks your total spend, cost by service, and cost over time. You can drill down into resource types and instances. You can also break down your costs by organization or cost center by tagging resources with those categories.

Azure Advisor also has a cost component. It recommends VM resizing, buying reserved instances when more cost effective than pay-as-you-go instances. It identifies unused ExpressRoute circuits and idle virtual network gateways. Advisor makes additional recommendations in the areas of performance, high availability, and security.

The important part is to take time to review your spend and evaluate where your money is going. Identify areas of inefficiency to ensure you're operating as efficiently as possible.

Organize to optimize

Putting some organization to your resources can help track where some of your costs are going. There are ways to group resources together, establishing a relationship so you know where your costs are related. From a billing perspective, resources can be easily grouped by:

- Assigning resources to different subscriptions.
- Assigning resources to different resource groups.
- Applying tags to resources.

Using subscriptions and resource groups to organize resources is an easy way to logically group resources and can be leveraged when going through billing data. Tags come into play when resource relationships span the boundaries of subscriptions and resource groups. Tags are key/value pairs that can be added to any resource, and are exposed in billing data, allowing you to associate a department or cost center with your resource. Tags improve your ability to report on cost, as well as giving each department in your organization accountability for their own costs. The following illustration shows how you can apply the same tag to resources in different resource groups and even in different subscriptions.

Adding some organization to your resources can go a long way and can really aid in your ability to understand where your costs are going. Now let's take a look at some ways to optimize costs.

Optimizing IaaS costs

When using Infrastructure as a Service (IaaS) resources such as virtual machines (VM) as part of your solution, the cost associated with VMs is often the biggest portion of spend. The compute costs are typically the biggest piece, followed by storage. Taking time to optimize pay-for-what-you-use resources can have a large impact on the size of your monthly bill.

Let's take a look at best practices to reduce your compute and storage costs.

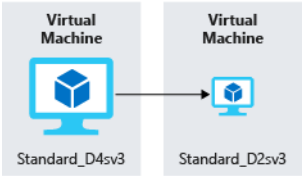
Compute

There are different options available to achieve cost savings for virtual machines.

- Choose a smaller virtual machine instance size.
- Reduce the number of hours a virtual machine runs.
- Use discounts for the compute costs.

Right size virtual machines

Right sizing a virtual machine is the process of matching the virtual machine size with the resource demand required of the VM. If a VM is running 25% idle, reducing the size of the VM will immediately reduce your cost. Virtual machine costs are linear within an instance family; each next size larger will double your cost. Conversely, reducing a VM by a single instance size will reduce your cost in half. The following illustration shows a 50% savings achieved by moving one size down within the same series.



Azure Advisor identifies which virtual machines are underutilized. Advisor monitors your virtual machine usage for 14 days and then identifies underutilized virtual machines. Virtual machines whose CPU utilization is 5 percent or less and network usage is 7 MB or less for four or more days are considered underutilized virtual machines.

Implement shutdown schedules for virtual machines

If you have VM workloads that are only used periodically, but are running continuously, you're wasting money. These VMs can be shut down when not in use saving you compute costs while the VM is deallocated. For example, a development environment is a good candidate because development generally happens only during business hours.

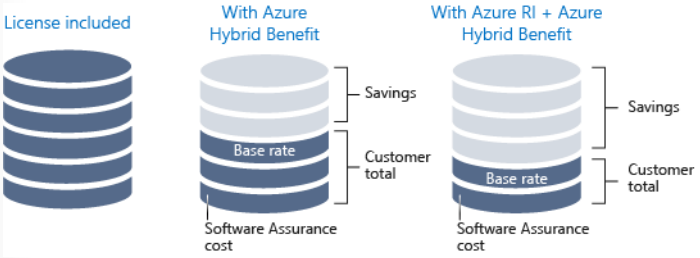
You have several options to deallocate a VM. You can use Azure Automation to run your VMs only during those times that your workloads require. You can use the auto-shutdown feature on a virtual machine to schedule a one-off automated shutdown. Finally, you can stop a VM manually in the Azure portal. You

should always use the Azure controls to stop your VMs; shutting down the OS from inside a VM does not deallocate its Azure resource so you will continue to accrue costs.

Apply compute cost discounts

The Azure Hybrid Benefit allows you to further optimize your costs for both Windows Server and SQL Server by allowing you to use your on-premises Windows Server or SQL Server licenses with Software Assurance to be used as a discount toward the compute cost of these VMs, eliminating the costs for Windows and SQL Server on enabled instances.

Some virtual machines need to be up and running all the time. Maybe you have a web application server farm for a production workload or maybe a domain controller supporting various servers on a virtual network. If you know with certainty that these virtual machines will run over the coming year or maybe longer, you can get further cost savings by purchasing a reserved instance. Azure Reserved Virtual Machine Instances can be purchased for one year or three years of compute capacity, at a discount compared to pay-as-you-go compute resources. Azure Reserved Virtual Machine Instances can significantly reduce your virtual machine costs, up to 72 percent on pay-as-you-go prices, with one-year or three-year upfront commitment. The following illustration shows savings achieved when you combine your on-premises license with the Azure Hybrid Benefit and when you combine your on-premises license with both Azure RI and the Azure Hybrid Benefit.



Virtual machine disk storage cost optimization

For workloads that do not require high reliability and performance disks, you can use the reduced-cost standard storage. You might choose to use standard storage for development and test environments that are not required to be an identical match for a production workload.

Ensure you don't have any orphaned disks remaining in your environment. Disks that aren't associated with a VM still incur storage costs. If you've removed a VM but not the disks, the orphaned disks may be a place to reduce your storage cost.

Similar to orphaned disks, if you have any orphaned snapshots lingering around, take some time to clean them up. Pricing for these is lower than the disks themselves, but it's still a good practice to eliminate costs of unnecessary resources.

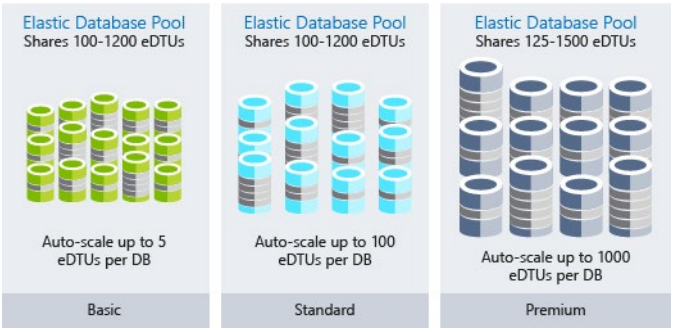
Optimizing PaaS costs

Platform as a Service (PaaS) services are typically optimized for costs over IaaS services, but there are opportunities to identify waste and optimize for minimal costs. Let's take a look at ways to reduce Azure SQL Database and Azure Blob storage costs.

Optimizing Azure SQL Database costs

When creating an Azure SQL database, you have to select an Azure SQL Server and decide on a performance tier. Each tier provides a performance level either in database transaction units (DTUs) or virtual cores (vCores). For database loads that are steady, it's easy to optimize by selecting the properly sized tier for the needed performance. But what if your database has unpredictable bursts or spikes in activity? Elastic pools can reduce costs for unpredictable workloads.

SQL Database elastic pools are a simple, cost-effective solution for managing and scaling several databases that have varying and unpredictable usage demands. The databases in an elastic pool are on a single Azure SQL Database server and share a set number of resources at a set price. Pools are well suited for a large number of databases with specific utilization patterns. For a given database, this pattern is characterized by low average utilization with relatively infrequent utilization spikes.
The more databases you can add to a pool, the greater your savings become. The following illustration shows the capabilities of the three types of Elastic Database Pools: basic, standard, and premium. Basic auto scales up to 5 eDTUs per DB, standard auto scales up to 100 eDTUs per DB, and Premium that auto scales up to 1000 eDTUs per DB.



Elastic pools are a great way to spread costs across multiple databases and can make a significant impact on reducing your Azure SQL Database costs.

Optimizing Blob storage costs

Blob storage is a cost-effective way to store data, but as the amount of data grows, your bill can benefit from optimizing how the data is stored.

Let's return to Lamna Healthcare. You have a medical-imaging application that stores images in blob storage. Due to the quantity and size of the images, the storage ends up being a notable cost for the application. When an image has been taken for a patient, it's likely that in the first week, that image will be viewed several times, and the performance of image retrieval is expected to be high. Conversely, an image taken two years ago may be accessed infrequently and has a lower retrieval performance expectation. You can use storage tiering to optimize the cost of image retrieval, given the reduced performance required as the image ages.

Azure Storage offers three storage tiers for blob object storage. The Azure hot storage tier is optimized for storing data that is accessed frequently. The Azure cool storage tier is optimized for storing data that is infrequently accessed and stored for at least 30 days. The Azure archive storage tier is optimized for storing data that is rarely accessed and stored for at least 180 days with flexible latency requirements.

- **Hot access tier** - Highest storage costs but the lowest access costs.
- **Cool access tier** - Lower storage costs and higher access costs compared to hot storage. This tier is intended for data that will stay in the cool tier for at least 30 days.

- **Archive access tier** - Lowest storage cost and highest data retrieval costs compared to hot and cool storage. This tier is intended for data that can tolerate several hours of retrieval latency and will stay in the archive tier for at least 180 days.

For Lamna Healthcare, keeping new images on the hot access tier for a month makes sense, so that viewing the most recent images performs as fast as possible. You could then move images over one year old to the archive tier since it is likely that these images will not be retrieved. This would reduce their costs associated with storing these images.

Leverage consumption pricing models

Moving to PaaS services can also take the pay-as-you-go model even further into a true consumption pricing model. Services such as Azure Functions have the ability to use *Consumption plans*. When you're using a Consumption plan, instances of the Azure Functions host are dynamically added and removed based on the number of incoming events. This serverless plan scales automatically, and you're charged for compute resources only when your functions are running. On a Consumption plan, a function execution times out after a configurable period of time.

Billing is based on number of executions, execution time, and memory used. Billing is aggregated across all functions within a function app.

Moving to services that use a consumption pricing model can bring a new approach to cost savings into your architecture.

Pause Compute Operations

PaaS services such as Azure SQL Data Warehouse provide the capability to pause the compute operations of the service when they are not in use. For example, in a daily load of a Data Warehouse, the daily load operation may be completed by 1:00am. If no one uses the data until 8:00am, then there is a 7 hour window where the compute layer can be paused. At this time it means that your organization is not charged for the compute, but there will be no access allowed to the data until the compute is resumed. This can optimize the cost of using Azure SQL Data Warehouse.

Cost optimization at Lamna Healthcare

Lamna Healthcare is making strides on reducing their costs. They have organized a monthly review of their costs, and each department has access to Azure Cost Management, where they can review their costs throughout the month. They've identified a number of places where reserved instances can be used and have purchased several to take advantage of this discount. They have implemented automated processes to stop development environments in off-hours, saving them additional costs during times when these resources were not being used.

Along with the optimization of blob storage for their imaging storage, they've managed to drop their bill notably over the past couple of months.

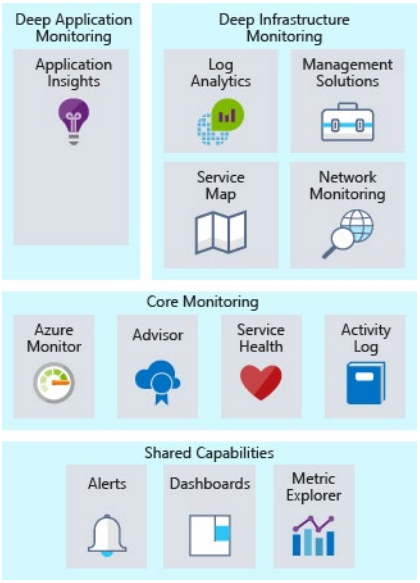
Summary

Optimizing the cost of your cloud infrastructure involves tracking your spend and ensuring that your resource utilization matches the demands of your workloads. Using the right quality and performance tier for your resources can further optimize your cloud spend.

Monitoring and Analytics to gain Operational Insights

Monitoring is the act of collecting and analyzing data to determine the performance, health, and availability of your business application and the resources that it depends on. What if you ran an operations team responsible for resources running on Azure? What would you do to ensure you had visibility into the health of your systems? If something happens, who finds out first, your team or your end users? An effective monitoring strategy helps you focus on the health of your application. It also helps you increase your uptime by proactively notifying you of critical issues, so that you can resolve them before they become problems.

When it comes to monitoring and analytics on Azure, we can bundle services into three specific areas of focus: deep application monitoring, deep infrastructure monitoring, and core monitoring. In this unit, we'll take a look through each of these bundles and how Azure services enable these capabilities for your architecture. Even though we've grouped these services together, there are several integration points between them, allowing for sharing of important monitoring data points between them. The following illustration shows the available monitoring services assembled into logical groups.



Core monitoring

Core monitoring provides fundamental, required monitoring across Azure resources. When we talk about fundamental monitoring, you can think of this as monitoring what is happening with your resources at the Azure platform level. This area of focus gives you insight into things like the health of the Azure platform, insight into changes being made to your resources, and performance metrics. Using services from this area gives you the ability to monitor the basic pieces you need to keep your application running.

Azure provides services to give you visibility into four key core monitoring areas: activity logging, the health of services, metrics and diagnostics, and recommendations on best practices. These services are built in to Azure and take little to no configuration to enable and set up. Let's take a closer look.

Activity logging

Activity logging is incredibly important to get information on what is happening with your resources at the Azure platform level. Every change submitted to the Azure platform is logged to the Azure Activity Log, giving you the ability to trace any action taken on your resources. Activity Log will contain detailed information on activities to help you answer questions like:

- Who has attached a disk to this virtual machine?
- When was this machine shut down?
- Who changed the load balancer configuration?
- Why did the autoscale operation on my virtual machine scale set fail?

Using Activity Log to answer these types of questions will help you troubleshoot issues, track changes, and provide auditing of what's happening in your Azure environment. Activity Log data is only retained for 90 days and can be archived to a storage account or sent to Azure Log Analytics for longer retention and further analysis.

Health of cloud services

At some point, any system can have issues, and that's true for Azure services as well. Staying informed of the health of Azure services will help you understand if and when an issue impacting an Azure service is impacting your environment. What may seem like a localized issue could be the result of a more wide-spread issue, and Azure Service Health provides this insight. Azure Service Health identifies any issues with Azure services that might affect your application. Service Health also helps you plan for scheduled maintenance.

Metrics and diagnostics

For issues that are more localized in nature, it's important to have visibility into what is happening on your system or service instance. The ability to view metrics and diagnostic information is critical to troubleshoot performance issues and stay notified when something goes wrong. To provide this visibility, Azure services have a common way of showing health, metric, or diagnostic information. Azure Monitor enables core monitoring for Azure services by allowing the collection, aggregation, and visualization of metrics, activity logs, and diagnostic logs.

Metrics are available that provide performance statistics for different resources and even the operating system inside a virtual machine. You can view this data with one of the explorers in the Azure portal and create alerts based on these metrics. Azure Monitor provides the fastest metrics pipeline (5 minutes down to 1 minute), so you should use it for time-critical alerts and notifications.

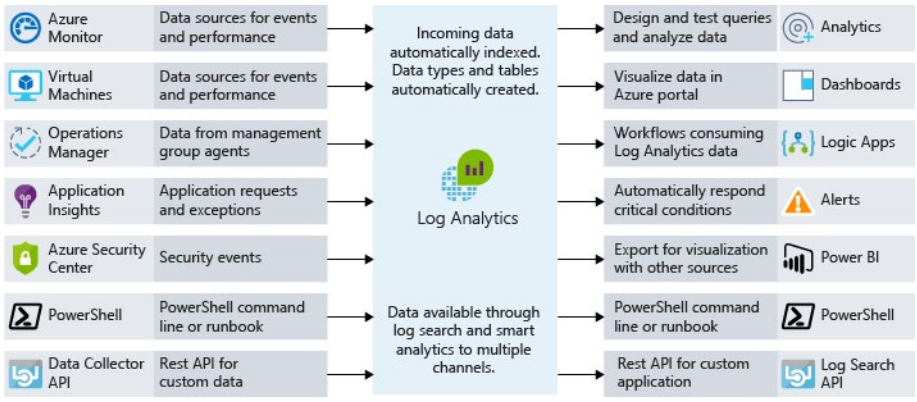
Recommendations on best practices

When we think of monitoring, we typically think of the current health of a resource. But even when a resource is healthy, there could be adjustments that would result in greater availability, reduced cost, or improved security. Azure Advisor can help by keeping an eye out for potential performance, cost, high availability, or security issues within your resources. Advisor makes personalized recommendations based on resource configuration and telemetry, providing guidance that most traditional monitoring platforms don't provide.

Deep infrastructure monitoring

While the monitoring components we've covered thus far are great at offering insights, they only give visibility to the Azure platform. For typical IaaS workloads, there's more metrics and diagnostic information to gather from the network or the actual operating systems. Pulling information from SQL Server to ensure it's properly configured, analyzing free disk space across all the servers in your environment, or visualizing the network dependencies between your systems and services are all examples where Log Analytics can provide deep insights.

When designing a monitoring strategy, it's important to include every component in the application chain, so you can correlate events across services and resources. For services that support Azure Monitor, they can be easily configured to send their data to a Log Analytics workspace. Virtual machines (both in the cloud and on-premises) can have an agent installed to send data to Log Analytics. You can submit custom data to Log Analytics through the Log Analytics API. The following illustration shows how Log Analytics acts as a central hub for monitoring data. Log Analytics receives monitoring data from your Azure resources and makes it available to consumers for analysis or visualization.



With this data in Log Analytics, you can query the raw data for troubleshooting, root cause identification, and auditing purposes. For several known services (SQL Server, Windows Server Active Directory), there are management solutions readily available that visualize monitoring data and uncover compliance with best practices.

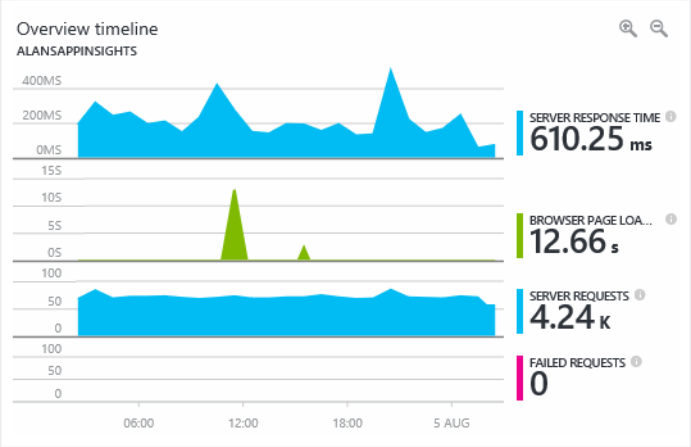
Log Analytics allows you to create queries and interact with other systems based on those queries. The most common example is an alert. Maybe you want to receive an email when a system runs out of disk space or a best practice on SQL Server is no longer followed. Log Analytics can send alerts, kick off automation, and even hook into custom APIs for things like integration with IT service management (ITSM).

Deep application monitoring

It's important to understand how core services and infrastructure are performing, but you can take your monitoring capabilities even further by looking deep into your applications to identify performance issues, usage trends, and overall availability of services you develop and depend on. By using an application performance management tool, you can better detect and diagnose issues that occur within your web apps and services.

Azure Application Insights allows you to do exactly that. Application Insights provides telemetry collection, query, and visualization capabilities. Little to no code changes are required. You only have to install a small instrumentation package into your application. Application Insights is cross platform, supporting .NET, Node.js, or Java.

For instance, the response time of the application might be complex to troubleshoot. Is it the web server being overloaded? Is it a specific SQL query that's not optimized? Is the API that you're calling performing slower than usual? Application performance monitoring solutions can help uncover the actual issues that basic metric monitoring can't expose. The following screenshot shows a graphical display of an application's performance details provided by Azure Application Insights.



An application performance monitoring solution will help you monitor usage, performance, and availability, allowing you to respond to failure much faster, and should be included in any monitoring strategy.

Monitoring at Lamna Healthcare

Lamna Healthcare has been revamping their monitoring strategy since moving their resources to the cloud. They're using Monitor for troubleshooting and alerting when performance issues may be impacting their resources. They have notifications configured to send any service health notifications to their operations team for immediate engagement. They have a process in place to regularly review Advisor to ensure the recommendations are implemented into their environment where applicable.

They send log data from all Azure and on-premises resources to a Log Analytics workspace, so they have the ability to search across log sources for event correlation and are using management solutions for Windows Server Active Directory and SQL Server.

Their development team has started integrating Application Insights into their applications, and they've already uncovered two defects that were impacting performance that had previously gone undetected.

Summary

A good monitoring strategy looks across multiple layers of an architecture, from supporting infrastructure to deep application telemetry. It will help you understand the detailed operation of the different components of your application. It increases your uptime by proactively notifying you of critical issues, so that you can resolve them before they become problems, and allows you to correlate logs and telemetry across systems to uncover issues. We've taken a look at a number of services on Azure that you can leverage in your monitoring strategy.

Automation to Reduce Effort and Error

Managing the infrastructure of any type of workload involves configuration tasks. This configuration can be done manually, but manual steps can be labor-intensive, error prone, and inefficient. What if you are assigned to lead a project that required the deployment of hundreds of systems on Azure? How would

you build and configure these resources? How long would this take? Could you ensure that each system was configured properly, with no variance between them? By using automation in your architecture design, you can work past these challenges. Let's take a look at some of the ways you can automate on Azure.

Infrastructure as code

Infrastructure as code is the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using a versioning system similar to what is used for source code. Like the principle that the same source code generates the same binary, an IaC model generates the same environment every time it is applied. IaC is a key DevOps practice and is often used in conjunction with continuous delivery.

Infrastructure as code evolved to solve the problem of environment drift. Without IaC, teams must maintain the settings of individual deployment environments. Over time, each environment becomes a snowflake, that is, a unique configuration that cannot be reproduced automatically. Inconsistency among environments leads to issues during deployments. With snowflakes, administration and maintenance of infrastructure involves manual processes which were hard to track and contributed to errors.

When automating the deployment of services and infrastructure, there are two different approaches you can take: imperative and declarative. In an imperative approach, you explicitly state the commands that are executed to produce the outcome you are looking for. With a declarative approach, you specify what you want the outcome to be instead of specifying how you want it done. Both approaches are valuable, so there's no wrong choice. What do these different approaches look like on Azure, and how do you use them?

Imperative automation

Let's start with imperative automation. With imperative automation, we're specifying *how* things are to be done. This is typically done programmatically through a scripting language or SDK. For Azure resources, we could use the Azure CLI or Azure PowerShell. Let's take a look at an example that uses the Azure CLI to create a storage account.

```
az group create --name storage-resource-group \
        --location eastus

az storage account create --name mystorageaccount \
        --resource-group storage-resource-group \
        --kind BlobStorage \
        --access-tier hot
```

In this example, we're specifying how to create these resources. Execute a command to create a resource group. Execute another command to create a storage account. We're explicitly telling Azure what commands to run to produce the output we need.

With this approach, we're able to fully automate our infrastructure. We can provide areas for input and output, and can ensure that the same commands are executed every time. By automating our resources, we've taken the manual steps out of the process, making resource administration operationally more efficient. There are some downsides to this approach though. Scripts to create resources can quickly become complex as the architecture becomes more complex. Error handling and input validation may need to be added to ensure full execution. Commands may change, requiring ongoing maintenance of the scripts.

Declarative automation

With declarative automation, we're specifying *what* we want our result to be, leaving the details of how it's done to the system we're using. On Azure, declarative automation is done through the use of Azure Resource Manager templates.

Resource Manager templates are JSON-structured files that specify what we want created. In the example below, we're telling Azure to create a storage account with the names and properties that we specify. The actual steps that are executed to create this storage account are left to Azure. Templates have four sections: parameters, variables, resources, and outputs. Parameters handle input to be used within the template. Variables provide a way to store values for use throughout the template. Resources are the things that are being created, and outputs are a way to provide details to the user of what was created.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "name": {
            "type": "string"
        },
        "location": {
            "type": "string"
        },
        "accountType": {
            "type": "string",
            "defaultValue": "Standard_RAGRS"
        },
        "kind": {
            "type": "string"
        },
        "accessTier": {
            "type": "string"
        },
        "httpsTrafficOnlyEnabled": {
            "type": "bool",
            "defaultValue": true
        }
    },
    "variables": {
    },
    "resources": [
        {
            "apiVersion": "2018-02-01",
            "name": "[parameters('name')]",
            "location": "[parameters('location')]",
            "type": "Microsoft.Storage/storageAccounts",
            "sku": {
                "name": "[parameters('accountType')]"
            },
            "kind": "[parameters('kind')]",
            "properties": {
```

```
                "supportsHttpsTrafficOnly": "[parameters('httpsTrafficOnlyEna-
bled')]",
                "accessTier": "[parameters('accessTier')]",
                "encryption": {
                    "services": {
                        "blob": {
                            "enabled": true
                        },
                        "file": {
                            "enabled": true
                        }
                    },
                    "keySource": "Microsoft.Storage"
                }
            },
            "dependsOn": []
        }
    ],
    "outputs": {
        "storageAccountName": {
            "type": "string",
            "value": "[parameters('name')]"
        }
    }
}
```

Templates can be used to create and manipulate most services on Azure. They can be stored in code repositories and source controlled, and shared across environments to ensure that the infrastructure being developed against matches what's actually in production. They are a great way to automate deployments and help ensure consistency, eliminate deployment misconfigurations, and can increase operational speed.

Automating your infrastructure deployment is a great first step, but when deploying virtual machines, there's still more work to do. Let's take a look at a couple of approaches to automating configuration post deployment.

Automation of operational tasks

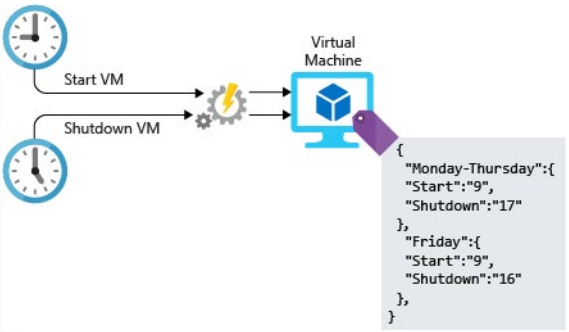
Once your solutions are up and running, there are ongoing operational activities that can also be automated. Automating these tasks with Azure Automation reduces manual workloads, enables configuration and update management of compute resources, centralizes shared resources such as schedules, credentials, and certificates, and provides a framework for running any type of Azure task.

For your Lamna Healthcare work, this might include:

- Periodically searching for orphaned disks.
- Installing the latest security patches on VMs.
- Searching for and shutting down virtual machines in off-hours.
- Running daily reports and producing a dashboard to report to senior management.

As a concrete example, suppose you want to run a virtual machine only during business hours. You can write a script to start the VM in the morning and shut it down in the evening. You can configure Azure

Automation to run the script at set times. The following illustration shows the role of Azure Automation in this process.



Automating development environments

At the other end of the pipeline of your cloud infrastructure are the development machines used by developers to write the applications and services that are the core of your business. You can use Azure DevTest Labs to stamp out VMs with all of the correct tools and repositories that they need. Developers working on multiple services can switch between development environments without having to provision a new machine themselves. These development environments can be shut down when not in use and restarted when they are required again.

Automation at Lamna Healthcare

Let's take a look at how Lamna Healthcare has improved by using automation. When you started your journey, infrastructure deployment and server builds were entirely manual. Engineers were deploying everything through the portal. This was introducing variance and errors between test and production environments, and the differences were hindering their ability to detect problems before code hit production.

They now deploy all their infrastructure through Resource Manager templates. These templates are checked into a GitHub repository, and a code review happens before they are released for deployment. They're also able to build the same infrastructure between dev, test, and production, ensuring they have validated their configuration across all environments.

For most services using virtual machines, they have a standard base image and use DSC to configure the systems post deployment. For web farms where they need the scalability of virtual machine scale sets, they have a fully automated process to check in code and build a new image with all required configuration built in before making it available in their scale sets.

They have an Automation job to shut down identified virtual machines in off-hours to reduce costs and have automated their VM patching as well.

Developers now have a self-service environment in DevTest Labs where they can develop against the latest images and configuration, ensuring that what they develop against matches the configuration in production.

All of this took some up-front effort, but the benefits have paid off in the long run. They've dramatically reduced error and the effort required by their operations teams to maintain their environments. Developers love that they can easily go provision resources to develop against, eliminating the back and forth to get environments created.

Summary

We've taken a look at a number of ways to bring automation capabilities into your architecture. From deploying infrastructure as code, to improving developer productivity with lab environments, there's a ton of benefit from taking time to automate your environment. Reducing error, reducing variance, and saving operational costs can be a significant benefit to your organization and help take your cloud environment to the next level.

Module Summary

Summary

We started out by talking about how to ensure you are in control of your consumption as well as optimize it. Here are some of the key aspects that were covered:

- Tracking cloud spend: the importance of gathering and looking at cloud consumption data to understand your cloud spend.
- Organizing to optimize: the relevance of a subscription and resource group hierarchy and how tags can further help provide various dimensions.
- There are several possibilities to optimize IaaS costs: right-sizing, reducing runtime hours, and compute cost discounts.
- Some examples were shown to optimize PaaS costs: for example, using Azure SQL Database elastics pools to optimize Azure SQL costs or Azure Blob storage tiering to optimize Azure Storage costs.

Next, we covered the operational insights you could gain from understanding and implementing monitoring and analytics. We determined that there are various levels on which we can apply monitoring.

- Monitoring is the act of collecting and analyzing data to determine the performance, health, and availability of your business application and the resources that it depends on.
- There are various depths of monitoring. Core monitoring covers aspects close to the Azure platform such as activity logging, cloud services health, and metrics and diagnostics.
- Deep infrastructure monitoring allows you to gather information besides what the Azure fabric can provide. For typical IaaS workloads, we can monitor at the network or operating system level.
- Deep application monitoring takes it even further by gathering metrics and diagnostics information close to the application, allowing you to identity performance issues, usage trends, and overall availability information.

We wrapped up by discussing that there are various ways to automate tasks to improve your operational capabilities. Here are some of the key things we covered:

- **Automating the deployment of resources can take two different approaches:** imperative (scripting) or declarative (templates).
- **After the VM is deployed, we can look at how to customize it.** Either by using custom images and thus remove the need for customization in the first place. Or by running a script post deployment.
- **Automation of Azure tasks:** Azure Automation can help with installing updates or shutting down virtual machines on a schedule.
- **Automation lab environments:** Azure DevTest Labs takes automation one step further by providing an easy-to-use interface that allows you to use various automation capabilities like images or shutting down VMs without having to worry about the actual implementation of those tasks.