Exam Ref 70-535
Architecting Microsoft Azure Solutions

Haishi Bai
Dan Stolts
Santiago Fernández Muñoz
I would like to dedicate this book to editors, technical reviewers and co-authors. It’s been a long and collaborative process to get the book out. I appreciate your dedication, professionalism and persistence to complete the quest.

——HAISHI BAI

I would like to dedicate this book to my son Brad. His love, encouragement, drive and motivation gave me the strength to get to the finish line.

——DAN STOLTS

I would like to dedicate this book to my wife Rocio, for supporting me all the time I spent on this and other projects and being the most important reason on my life to be a better person.

——SANTIAGO FERNÁNDEZ MUÑOZ
Contents at a glance

Introduction xv

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

CHAPTER 1 Design compute infrastructure 1
CHAPTER 2 Design data implementation 85
CHAPTER 3 Design networking implementation 149
CHAPTER 4 Design security and identity solutions 193
CHAPTER 5 Design solutions by using platform services 249
CHAPTER 6 Design for operations 279

Index 319
# Contents

Introduction

Acknowledgments ........................................... xi
Organization of this book ...................................... xv
Microsoft certifications ........................................ xv
Microsoft Virtual Academy ..................................... xvi
Quick access to online references ............................. xvi
Errata, updates, & book support ................................ xvii
Stay in touch ...................................................... xvii

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

## Chapter 1  Design compute infrastructure  1

**Skill 1.1: Design solutions using virtual machines** ................. 2
- Design VM deployments by leveraging Availability sets, Fault Domains, and Update Domains in Azure 3
- Design for compute-intensive tasks using Azure Batch 21
- Define a migration strategy from cloud services 23

**Skill 1.2: Design solutions for serverless computing** .................. 32
- Use Azure Functions to implement event-driven actions 32
- Design for serverless computing using Azure Container Instances 36
- Design Application Solutions by using Azure Logic Apps, Azure Functions, or both 38
- Determine when to use API Management service 40

**Skill 1.3: Design microservices-based solutions** .......................... 41
- Determine when a container-based solution is appropriate 41
- Determine when container-orchestration is appropriate 43
- Determine when Azure Service Fabric (ASF) is appropriate 44
- Determine when Azure Functions is appropriate 44
- Determine when to use the API Management service 44
- Determine when Web API is appropriate 45
Determine which platform is appropriate for container orchestration
Consider migrating existing assets versus cloud native deployment
Design lifecycle management strategies

Skill 1.4: Design web applications
Design Azure App Service Web Apps
Design custom web APIs
Secure Web API
Design Web Apps for scalability and performance
Design for high availability using Azure Web Apps in multiple regions
Determine which App Service Plan to use
Design Web Apps for business continuity
Determine when to use Azure App Service Environment (ASE)
Design for API apps
Determine when to use Web Apps on Linux
Determine when to use a CDN
Determine when to use a cache, including Azure Redis Cache

Skill 1.5: Create compute-intensive applications
Design high-performance computing (HPC) and other compute-intensive applications using Azure Services
Determine when to use Azure Batch
Design stateless components to accommodate scale
Design lifecycle strategy for Azure Batch

Thought experiment

Thought experiment answers

Chapter summary

Chapter 2  Design data implementation
Skill 2.1: Design for Azure Storage solutions
Skill 2.2: Design for Azure Data Services
Skill 2.3: Design for relational database storage
Chapter 4  Design security and identity solutions  193

Skill 4.1: Design an identity solution ............................................. 193
  Claim-based architecture ................................. 194
  Basic authentication and authorization workflow 196
  Working with Native Clients ................................. 197
  Working with multi-tiered applications .............. 198
  Additional scenarios ........................................ 199
  Azure Active Directory .................................. 199
  Sample scenario with Azure Active Directory
  Authentication and Visual Studio .................. 201
  Authentication frameworks ....................... 205
  Microsoft Graph API ................................ 206
  Secure resources by using hybrid identities .... 210

Skill 4.2: Secure resources by using identity providers ............ 216
  Sample scenario with external Identity Provider
  and ASP.NET Core .................................. 216
  Azure B2C ........................................ 220
  Azure B2B ........................................ 220

Skill 4.3: Design a data security solution ............................... 221
  Data protection ...................................... 222
  Data encryption ...................................... 222
  Access Control .................................... 224
  Data reliability and disaster recovery .......... 227
  Azure Rights Management Services ............. 231
  Azure Key Vault .................................. 231

Skill 4.4: Design a mechanism of governance and polices
  for administrating Azure resources .................... 232
  Access control challenges faced by large enterprises 233
  Role Based Access Control (RBAC) ................. 233
  RBAC for Azure Resources ....................... 235
  Empowering a user with self-service ............ 238
  Azure AD Application Access Panel .......... 240
## Chapter 6  Design for operations  279

Skill 6.1: Design an application monitoring and alerting strategy  

Determine the appropriate Microsoft products and services for monitoring applications on Azure  280  
Define solutions for analyzing logs and enabling alerts using Azure Log Analytics  282  
Define solutions for analyzing performance metrics and enabling alerts using Azure Monitor  290  
Define a solution for monitoring applications and enabling alerts using Application Insights  293

Skill 6.2 Design a platform monitoring and alerting strategy  

Determine the appropriate Microsoft products and services for monitoring Azure platform solutions  297  
Define a monitoring solution using Azure Health, Azure Advisor, and Activity Log  300  
Define a monitoring solution for Azure Networks using Log Analytics and Network Watcher service  301  
Monitor security with Azure Security Center  303

Skill 6.3 Design an operations automation strategy  

Determine when to use Azure Automation, Chef, Puppet, PowerShell, Desired State Configuration (DSC), Event Grid, and Azure Logic Apps define a strategy for auto-scaling  307  
Define a strategy for enabling periodic processes and tasks  314

Thought experiment  

Thought experiment answers  

Chapter summary  

Index  319
Acknowledgments

DAN STOLTS: I’d like to thank the following people: Brad Stolts, Leslie Stolts, Kathy Vieira, John Ross, and Ronald Thibeau.

SANTIAGO FERNÁNDEZ MUÑOZ: I’d like to thank my mentor and, much more important, my friend Rafa, for always helping and advising me on the right path. He always makes me think twice, helping me make the correct questions that have helped me to become the person that I am today. Thank you Rafa!

About the authors

HAISHI BAI, principal software engineer at Microsoft, focuses on the Microsoft Azure compute platform, including IaaS, PaaS, networking, and scalable computing services. Ever since he wrote his first program on an Apple II when he was 12, Haishi has been a passionate programmer. He later became a professional software engineer and architect. During his 21 years of professional life, he’s faced various technical challenges and a broad range of project types that have given him rich experiences in designing innovative solutions to solve difficult problems. Haishi is the author of a few cloud computing books, and he’s an active contributor to a few open-source projects. He also runs a technical blog(http://blog.haishibai.com) with millions of viewers. His twitter handle is @HaishiBai2010.

DAN STOLTS “ITProGuru” is a proven leader in business and technology with over 30 years of experience. He is a technology expert and leader who is a master of systems management, DevOps, and security. He is Chief Technology Strategist for Microsoft, owns Bay State Integrated Technology, Inc. and is a published author. Reach him on his primary blog http://itproguru.com or twitter @ITProGuru. He is a proven leader of teams, people and projects. He is proficient in many datacenter technologies (Windows Server, System Center, Virtualization, Cloud, SQL, etc.) and holds many certifications including MCT, MCITP, MCSE, TS, etc. Dan is currently specializing in DevOps and cloud technologies. Dan is and has been a very active member of the user group community. He is an enthusiastic advocate of technology and is passionate about helping others. See more at LinkedIn https://www.linkedin.com/in/danstolts or on his blog http://itproguru.com/about.
SANTIAGO FERNÁNDEZ MUÑOZ started his career as a trainee in a training center in Seville where he started working with Unix and Windows systems. He followed his passion and taught other people, but it was not what he wanted to do with the rest of his life, so he moved to other companies where he started to work with bigger and bigger projects, with more people and countries involved. He's been working as an Infrastructure Solution Architect for the last six years. He has always been passionate about Microsoft technologies, starting with Windows Server 2003 through to Windows Server 2016 and Azure. He is focused on the automation of cloud infrastructure and continuous integration and delivery for software development.
Introduction

This book teaches you how to design and architect secure, highly-available, performant, monitored and resilient solutions on Azure. This book guides you through leveraging functional, operational and deployment requirements to deploy best in class solutions running in Azure or a hybrid environment. DevOps, automation, monitoring and hands-off management are all key foundations of the highly resilient systems you will be able to design after understanding the material covered.

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

Organization of this book

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

Microsoft certifications

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

**MORE INFO ALL MICROSOFT CERTIFICATIONS**

For information about Microsoft certifications, including a full list of available certifications, go to [http://www.microsoft.com/learning](http://www.microsoft.com/learning).

Check back often to see what is new!

**Microsoft Virtual Academy**

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

[http://www.microsoftvirtualacademy.com](http://www.microsoftvirtualacademy.com)

**Errata, updates, & book support**

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

[https://aka.ms/examref535/errata](https://aka.ms/examref535/errata)

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

If you need additional support, email Microsoft Press Book Support at [msinput@microsoft.com](mailto:msinput@microsoft.com).
Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to http://support.microsoft.com.

Stay in touch

Let’s keep the conversation going! We’re on Twitter: http://twitter.com/MicrosoftPress.
Important: How to use this book to study for the exam

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

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

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

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

Note that this Exam Ref is based on this publicly available information and the author’s experience. To safeguard the integrity of the exam, authors do not have access to the exam questions.
CHAPTER 3

Design networking implementation

The foundation of the cloud is a large pool of storage, compute, and networking resources, allowing you to acquire any amount of cloud resources at any time, from anywhere, without managing any underlying infrastructure. Once resources are complete, return them to the cloud to avoid any unnecessary costs. Azure resources are managed by Azure Resource Manager (ARM), providing a unified API to management tools and automation scripts for provisioning, monitoring and releasing Azure resources.

Some cloud services give access to the infrastructure, such as Virtual Machines (VMs) and virtual networks, and are called Infrastructure as a Service (IaaS). Platform as a Service (PaaS) provides support for building your own services on the cloud. And, Software as a Service (SaaS), makes it possible to handle workloads on the cloud.

Azure provides networking features similar to on-premises datacenters. This chapter provides coverage on networking, introducing key components, services, and tools used to implement various networking scenarios.

Skills covered in this chapter:

- Skill 3.1: Design Azure Virtual Networks
- Skill 3.2: Design external connectivity for Azure Virtual Networks
- Skill 3.3: Design security strategies
- Skill 3.4: Design connectivity for hybrid applications

Skill 3.1: Design Azure Virtual Networks

Today, just about any computer you see is connected to a network. Computers on Azure are no exception. Provisioning a new VM on Azure prevents physical access to the hosting machine. Instead, you can operate the machine through remote connections, such as remote desktop or Secure Shell (SSH), which is made possible by Azure’s networking infrastructure.

Azure Virtual Network, introduced here, creates virtualized private networks on Azure. VMs deployed on a virtual network can communicate like they do on an on-premises local area network (LAN).
Connect virtual networks with on-premises networks, or with other virtual networks, through cross-network connections. Skill 3.2 covers hybrid networks.

This section covers how to:
- Create and manage virtual networks
- Implement load balancing
- Use User Defined Routes (UDRs)

Create and manage virtual networks

It’s easy to create a new virtual network on Azure. Here we will set up a new virtual network with two subnets on Azure, covering the differences between a virtual network and an on-premises network required when designing network infrastructures in the cloud.

NOTE REVIEW OF BASIC NETWORKING CONCEPTS
A deep networking knowledge isn’t required here, since you may not routinely maintain networks. We provide refreshers of basic networking concepts in notes found throughout this chapter. Feel free to skip these notes if you’re already familiar with the concepts.

Creating a virtual network by using the Azure management portal

There are ways to create a new virtual network on Azure, including using the Azure management portal, Azure PowerShell, and Azure CLI. Here you will use the management portal to create a new virtual network, and scripting options are discussed later in this chapter.

2. Click on the New link at the upper-left corner, and then select Networking | Virtual Network, shown in Figure 3-1.
3. On the Create Virtual Network blade, type a name for the virtual network. In the Address space box, change the CIDR to 10.0.0.0/16. You can pick any address space you like. In this example, we’ll use an address space and create two subnets on the network.

**NOTE** ABOUT ADDRESS SPACE CONFLICT WARNINGS

When entering the CIDR, you might see a warning message that says the address space ‘10.0.0.0/16’ overlaps with another existing address space. This is because you’ve already created another virtual network whose address space overlaps with the current address space. This is not a problem if you use the two virtual networks in isolation. You’ll face problems, however, when you try to connect them through cross-network connections (see Skill 3.2).

4. Change the subnet name to **frontend** and the subnet address range to **10.0.0.0/24**. Later in this exercise, you’ll create a backend subnet. When managing a large virtual network, create multiple subnets to improve performance. To describe this briefly, a network is like a web of roads. When you have more computers sending and receiving packets on the same network, packets can collide and must be resent. Using subnets, you can control and limit traffic in different areas. It’s similar to using local roads for a short commute and using shared highways to travel longer distances.

In many cases, subnets are created not only for performance but also for manageability. You can create subnets in alignment with business groups, such as creating one subnet for the sales department and another subnet for engineering. You can also create subnets based on server roles. Create a subnet here for a frontend and another subnet for a backend.
NOTE  ABOUT CIDR NOTATION

Classless Inter-Domain Routing (CIDR) notation is a shorthand representation of a subnet mask. It uses the number of bits to represent a subnet mask. For example, a subnet mask of 255.0.0.0 uses 8 bits hence, it’s written as /8. And a subnet mask of 255.255.0.0 uses 16 bits, which is written as /16 in CIDR notation. With CIDR, 10.0.0.0/16 in this exercise represents a network ID of 10.0.0.0 and a subnet mask of 255.255.0.0, which corresponds to the address range 10.0.0.0 to 10.0.255.255.

1. If you have multiple Azure subscriptions, pick the subscription to use in the Subscription dropdown box.

2. All of your Azure resources are organized in resource groups. You can choose to create a new resource group or put the virtual network into an existing resource group.

3. Pick the Azure region where you want to deploy your network and then click on the Create button.

4. Once the virtual network is created, click on the Subnets menu and then the +Subnet icon, as shown in Figure 3-2.

5. On the Add subnet blade, type in **backend** as the subnet name, and verify that the CIDR block is 10.0.1.0/24. Then, click on the OK button to add the subnet.
NOTE AVAILABLE ADDRESSES

In Figure 3-2, the number of available addresses is 251 instead of 256. This is because Azure reserves the first three available IP addresses in the range for internal uses. Subtract the first network address and the last broadcast address, and you are five short from 256. Why are there 256 addresses to begin with? When you create a subnet, you are borrowing a number of bits from the host ID and adding them to the network ID. For example, if you borrow 8 bits (24-16=8), you can create 256 (2^8) subnets. Each subnet has 256 (2^8) addresses. If you borrow 3 bits instead, you can create 8 (2^3) subnets. Because the bits borrowed are high bits, they correspond to 0, 32, 64, 96, 128, 160, 192 and 224. In this case, the first address on the second subnet will be 10.32.0.0.

Managing virtual networks with Azure Cloud Shell or Azure CLI

Azure CLI is a cross-platform command-line tool for managing Azure resources. You can download and install Azure CLI for macOS, Linux, and Windows. You can also access Azure CLI directly from Azure management portal through a feature called Cloud Shell. In this exercise, you’ll use Cloud Shell to perform a couple of administrative tasks. You’ll first inspect an existing virtual network and then create and delete another virtual network.

1. On Azure management portal, click on the Cloud Shell icon in the upper-right corner, as shown in Figure 3-3. This launches a new Cloud Shell instance at the bottom of the portal screen.

   ![FIGURE 3-3 Launching Cloud Shell](image)

2. If you have multiple Azure subscriptions, use the following command to choose the subscription you want to use. Otherwise, skip to step 3.

   `az account set --subscription '<your subscription name>'`

3. To list existing virtual networks in a table format, use the following command.

   `az network vnet list --output table`

4. To add a new virtual network with address space 192.168.0.0/16 to your resource group, use the following command.

   `az network vnet create --name <virtual network name> --address-prefix 192.168.0.0/16 --resource-group <existing resource group name>`

5. To delete the virtual network, use the following command.

   `az network vnet delete --name <virtual network name> --resource-group <resource group name>`
Quick check

As you’ve seen, you can create, update and destroy virtual networks freely on Azure. And as you are doing so, there are many other Azure users working on the same environment, allocating address spaces that are same as yours. How could this be possible on the same networking infrastructure?

Quick check answer

The answer is Software Defined Network (SDN). SDN defines a virtualized networking environment on top of the physical network infrastructure. It decomposes the network control plane and the data plane. The control plan is programmable. This allows network administrators to design logical network topologies as well as network policies at an abstracted level without needing to worry about exact physical implementations. The network topology and polices are then pushed down to the data plane and implemented on Azure host machines.

IP Addresses

You’ve learned how to perform basic virtual network management tasks, so let’s dig deeper into how the Azure Virtual Network works, beginning with IP addresses.

You can assign two types of IP addresses to an Azure resource: public IP and private IP. Public IPs are used for communication with the Internet. Private IP addresses are used for communication within a single Azure Virtual Network or connected virtual networks.

NOTE AZURE DEPLOYMENT MODELS

Azure provides two deployment models for managing resources: resource and classic. It’s recommended to use Resource Manager to provision any new Azure resources. The classic deployment model is not discussed further in this book.

Private IP Addresses

Private IPs can be assigned to resources sitting in a virtual network and provide addresses for them to reach one another. For example, you can assign a private IP to the network interface card (NIC) of a VM, making the VM addressable within the scope of the virtual network. If the VM has multiple NICs, you can assign a private IP to each of the NICs.

Figure 3-4 depicts a sample topology of a virtual network (10.0.0.0/16) with two subnets (10.0.0.0/24 and 10.0.1.0/24). A default system route table allows VMs (10.0.0.4, 10.0.1.4 and 10.0.1.5) to communicate with each other. When a VM tries to reach the Internet, its private IP address is Source Network Address Translated (SNAT) into a public IP address by the Azure infrastructure.
NOTE  SNAT AND OUTBOUND TRAFFIC

When a VM with only a private IP initiates an outbound connection, Azure translates its private IP to a routable public IP so that the return traffic can reach the originating VM. This is done by Source Network Address Translation (SNAT). SNAT uses ephemeral ports of the public IP address to distinguish individual flows. And these ports are referred to as SNAT ports. Flows to different destination IP addresses share the same SNAT port, while multiple flows to the same destination IP address use different SNAT ports. When SNAT ports are exhausted, new outbound connections will fail. When a VM with a public IP address initiates an outbound connection, SNAT is not used because Azure can directly route the returning traffic to the IP address.

A private IP address can be either dynamic or static. A dynamic IP address is automatically allocated from the resource's subnet address space using DHCP. When a resource is stopped and restarted, it may get a different private IP address. A static IP address, on the other hand, remains the same throughout the lifetime of the resource.

Dynamic private IP addresses are allocated by an Azure-provided DHCP server. The DHCP server also assigns a DNS server and a default gateway. The DNS server is by default the host server IP address, and the gateway is an address on your Azure virtual network subnet. Later in this chapter, you’ll learn about how to set up a custom DNS server.

Public IP Addresses

A public IP address is an independent Azure resource with its own properties. It can be associated to resources such as VMs, Azure Load Balancers, VPN gateways and Application Gateways to allow direct communications to and from the Internet.
Public IPs come from an IP address pool owned by Microsoft, and they are assigned to Azure resources either dynamically or statically. A dynamic public IP is allocated when you start or create the associated resource and is released when you stop or delete the resource. A static IP is released only when you delete the resource.

Available public IPs on the Internet are limited resources. You can allocate up to 60 dynamic public IP addresses and 20 static public IP addresses per Azure subscription. You need to contact Azure support to raise the limit if you need more.

## Name resolution

Azure provides a built-in recursive resolver for name resolutions. This virtual DNS server is located at a fixed virtual public IP address of 168.63.129.16. It filters DNS lookup requests so that you can only resolve hostnames of your own deployments. Because VMs residing on the same virtual network share the same DNS postfix (*.internal.cloudapp.net), they can reach one another directly by their hostnames.

You can bring your own DNS server when needed. For example, when you need cross-virtual network name resolutions, you can set up a DNS server on each of the virtual networks and cross-forward DNS queries, as shown in Figure 3-5. In this scenario, a DNS server is configured in each of the virtual networks and forwards DNS queries to the other DNS server. The two virtual networks are connected by an inter-network connection, which will be discussed later in Skill 3.2. The DNS servers also forward DNS queries to Azure-provide name resolution servers for other name resolutions. Please note that when you set up your own DNS servers, you should assign static IP addresses to them.

![Custom DNS servers](image)

**FIGURE 3-5** Custom DNS servers

You can assign DNS name labels to public IPs that you allocate. These DNS name labels are scoped with system-defined postfixes in the format of `<region>.cloudapp.azure.com`. For example, a DNS name label in the West US region has the name `<label>.westus.cloudapp.azure.com`. If you
prefer your own domain name, you need to set up corresponding DNS records pointing to the public IP in your own domain DNS servers. Or, you can use Azure DNS to host DNS records for your domain.

In the following exercise, you'll use Cloud Shell to create three virtual machines on the virtual network you've provisioned in the virtual network management exercise. One of the VMs is put in the frontend subnet and assigned a public IP. The other two VMs are put in the backend subnet with only private IPs. Then, you'll assign a DNS label to the public IP of the frontend machine. And finally, you'll use Azure DNS to set up a custom domain record for the frontend machine.

1. Open Cloud Shell.
2. Use the following command to create the frontend machine, which creates an Ubuntu LTS server using the `Standard_DS1_v2` machine size. You can choose a different VM size if needed. If you don't specify a `--public-ip-address` parameter, the `az vm create` command automatically allocates a public IP (using dynamic allocation method) for you. If you don't want a public IP address to be allocated, you need to pass an empty string to the parameter. Please also note the machine is placed in the frontend subnet by the `--subnet` parameter.

   ```bash
   az vm create --resource-group <your resource group name> --name frontend-vm --admin-password <machine admin password> --vnet-name <virtual network name> --authentication-type password --admin-username <machine admin name> --size Standard_DS1_v2 --image UbuntuLTS --subnet frontend --nsg ""
   ```

3. Once the machine is provisioned, you can observe the assigned public IP in the command’s output JSON. To query the public IP at a later time, use the following command:

   ```bash
   az vm list-ip-addresses --name <virtual machine name> | grep ipAddress
   ```

4. Create an availability set. You don't need it in this exercise, but you'll need to use it in later exercises. In step 5, you'll create two backend VMs on the availability set.

   ```bash
   az vm availability-set create --name <availability set name> --resource-group <resource group name>
   ```

5. Create two backend VMs on the backend subnet:

   ```bash
   az vm create --resource-group <your resource group name> --name backend-vm-1 --admin-password <machine admin password> --vnet-name <virtual network name> --authentication-type password --admin-username <machine admin name> --size Standard_DS1_v2 --image UbuntuLTS --subnet backend --nsg "" --public-ip-address "" --availability-set <availability set name>
   ```

   ```bash
   az vm create --resource-group <your resource group name> --name backend-vm-2 --admin-password <machine admin password> --vnet-name <virtual network name> --authentication-type password --admin-username <machine admin name> --size Standard_DS1_v2 --image UbuntuLTS --subnet backend --nsg "" --public-ip-address "" --availability-set <availability set name>
   ```
6. Once all of the machines are created, you can SSH into the frontend machine and do some experiments, by using the following command:

   ssh <machine admin name>@<frontend machine public IP>

7. Once connected to the frontend machine shell, try a domain lookup on one of the backend machines:

   nslookup backend-vm-1

   You should see an output like the following output:

   Server:     168.63.129.16
   Address: 168.63.129.16#53
   Name:   backend-vm -1.mtji5hmdhvgudmao4fdntopbwe.dx.internal.cloudapp.net
   Address: 10.0.1.5

8. Exit the SSH by typing in **exit** and pressing Enter. Now let’s assign a DNS label to the frontend VM’s public IP address. To do this, you’ll need to get the resource name for the public IP address. Use the vm list-ip-addresses command again and note the name of the public IP address resource:

   [
   {
       "virtualMachine": {
           "name": "frontend-vm-3",
           "network": {
               ...
           },
           "publicIpAddresses": [
               {
                   "ipAddress": "40.78.109.13",
                   "ipAllocationMethod": "Dynamic",
                   "name": "frontend-vm-3PublicIP",
                   ...
               ]
           }
       }
   ]

9. Assign a DNS label to the public IP address:

   az network public-ip update --name <public IP address name> --resource-group <resource group name> --dns-name <DNS label>

   The above command associates a <DNS label>.<region>.cloudapp.azure.com FQDN to the public IP address.

10. Now, the frontend VM can be accessed from a client by either its public IP address or its FQDN.

    Azure DNS provides high-performance and high-availability DNS hosts in Azure. You can manage and query your DNS records over the Azure global infrastructure. In the following exercise, you’ll create a new Azure DNS zone and then set up a custom domain record for the frontend machine.
1. In Cloud Shell, use the following command to create a new Azure DNS zone. The DNS zone name should have at least two parts, separated by ".", for example: haishi.com.

   `az network dns zone create --name <DNS zone name> --resource-group <resource group name>`

2. Add a new A record that points a `<DNS prefix>.<DNS zone name>` label to the frontend VM's public IP address:

   `az network dns record-set a add-record --record-set-name <DNS prefix> --zone-name <DNS zone name> --ipv4-address <frontend vm public IP> --resource-group <resource group name>`

3. Now, you can query your records using the following command:

   `az network dns record-set ns show --name @ --zone-name <DNS zone name> --resource-group <resource group name>`

   For example, the above command generates the following output in a given system:

   ```
   {
     "etag": "c75...",
     "id": "/subscriptions/46.../resourceGroups/70-534/providers/Microsoft.Network/dnszones/mydnszone.com/NS/@",
     "metadata": null,
     "name": "@",
     "nsRecords": [
       {"nsdname": "ns1-06.azure-dns.com."},
       {"nsdname": "ns2-06.azure-dns.net."},
       {"nsdname": "ns3-06.azure-dns.org."},
       {"nsdname": "ns4-06.azure-dns.info."}
     ],
     "resourceGroup": "70-534",
     "ttl": 172800,
     "type": "Microsoft.Network/dnszones/NS"
   }
   ```

   When you purchase your domain name, you should configure the name servers with the domain name registry.

**Load balancing**

So far, you've got a frontend VM and two backend VMs that can communicate by private IP addresses and hostnames. However, for the two backend VMs to provide a high-available backend, you don't want the frontend VM to address them separately. Instead, you need a load balancer in front of the backend VMs, as shown in Figure 3-6.
The internal load balancer provides the following benefits: First, it provides high-availability. As long as one of the backend VMs is functioning, they can provide continuous backend services to support the frontend. Second, it provides scalability. When you need more capacity in the backend, you can join more VMs behind the load balancer, and the load balancer will evenly distribute workloads to running VMs. This allows you to scale out backend without impacting the frontend.

**Availability set**

The availability of the system in Figure 3-6 is jeopardized if both backend VMs fail. Azure uses a concept of availability set to minimize the risk. An availability set is a management boundary that defines multiple fault domains (FDs) and update domains (UDs).

A fault domain is roughly a rack in an Azure datacenter. In Azure datacenters, unplanned maintenances are triggered by unexpected physical infrastructure problems such as network failures, rack-level failures, and other hardware failures. When such a failure is detected, Azure automatically moves your VMs to a healthy host. Fault domains don’t share a common power source or network switch, so the probability of two fault domains failing at the same time is low. When you add VMs to an availability set, they are evenly distributed into available fault domains.

Azure periodically performs maintenances on the hosting infrastructure. Many of these maintenances occur at the hosting operation system level and the platform software level without any impacts to the hosted VMs. However, some of these updates will require your VMs to be shut down or rebooted. When Azure updates VMs, it guarantees that not all machines in the same availability set will be shut down at the same time. Instead, it walks through update domains, bringing VMs down group by group. This process ensures that at any given time during updates there are at least a few machines running to provide continuous services.
Internal Load Balancer

Azure Internal Load Balancer provides load-balancing capacity within a virtual network. An internal load balancer is defined by the following components:

- **Frontend IP configuration** You can associate multiple IP addresses to a load balancer. The load balancer can be reached via any of the associated IP addresses. This allows a many-to-many mesh between the frontend VMs and the backend VMs.

- **Backend pools** Backend pools hold the candidate VMs to which the traffic is routed. You can add individual VMs and available sets into backend pools.

- **Health probes** A load balancer uses its health probes to detect the health of VMs. Traffic is routed only to VMs that pass health checks. Health probes periodically use either the TCP protocol or the HTTP protocol through the configured port to talk to the VMs. For TCP protocol, a successful TCP handshake is considered a positive response. For the HTTP protocol, a response with 200 return codes is considered a positive response. If a VM fails to provide a positive response within configured iterations, it's considered unhealthy and is taken out of the load-balancing pool. Please note that even if a VM is taken out of the load-balancing pool, health probes keep checking the VM and rejoin the VM back to the pool when the VM comes back.

- **Load balancing rules** Load balancer behaviors are driven by load balancing rules. A load balancing rule binds front IP address, backend pool, and the health probe together. Traffic from the front IP address is evenly distributed to healthy VMs in the backend pool, which is determined by the health probe. You can also specify session persistence behavior of a load balancing rule. If this value is set to “None,” successive requests from the same client during a session may be handled by any virtual machines in the backend pool. If the value is set to “Client IP,” requests from the same client IP address that are handled by the same virtual machine. “Client IP and protocol,” on the other hand, specifies that successive request from the same client IP address and protocol combination are handled by the same virtual machine.

  Session persistence is useful to route traffic to stateful services. For example, when you add multiple items to a server-held shopping cart, you want all the add requests to be routed to the same server that holds the shopping cart instance for your session. If the requests are scattered to different servers, you’ll get inconsistent shopping cart state across the servers, unless the state is replicated across the VMs.

- **Inbound NAT rules** You can set up NAT rules to route traffic to <load balancer IP>:<port> to different VMs in the backend pool based on port numbers. For example, to enable SSH connection to each VM instance, you can set up a few NAT rules that maps a series of ports to port 22 on individual VMs, so that a client can use the load balancer IP with a different port to select different VMs.

  An internal load balancer resides in a virtual network subnet and is assigned a private IP address from the network address space. To facilitate security configurations, you may want to assign static private IPs to internal load balancers.
Public load balancing

You can setup public-facing load balancers, which are associated with public IP addresses and can be directly accessed over the Internet. A public load balancer is configured in the same way as configuring an internal load balancer. The difference is that a public load balancer can route traffic to VMs across multiple virtual networks.

Figure 3-7 shows an extended topology of the previous samples. In this case, two frontend VMs form the frontend and join the public load balancer. Two backend VMs form the backend and join the internal load balancer. Internet traffic is routed to the frontend VMs via the public load balancer. The frontend VMs talk to the backend via the internal load balancer. And the internal load balancer distributes requests to backend VMs.

A public load balancer is an independent ARM resource that sits in the Azure infrastructure and is different from an internal load balancer that resides in your virtual network.

ARM object model

Before moving on to further discussions, it’s worthwhile to review the ARM object model behind the diagram in Figure 3-7. ARM treats all Azure artifacts it manages as resources. These resources are organized into resource groups. Resources can have dependencies among them either by reference or by association. Reference is a hard dependency. If resource A depends on resource B, it can’t be provisioned until resource B is already in place. Association, on the other hand, is a weak dependency. Association is often established by setting one resource as another resource’s property.

Figure 3-8 shows the object model behind Figure 3-7. Rectangles in the figure represent different Azure resource types. Ovals in the figure represent child properties of a resource type. The object model also includes Network Security Groups (NSGs), which will be covered later in this chapter. It’s important to understand the ARM object model and how resources are linked together. This will help you to focus on correct resources when working with a complex ARM topology. For example, although public IP addresses are often directly denoted on VMs, they are assigned to network interface resources, which is a different resource type.
NOTE AZURE RESOURCE EXPLORER

Azure Resource Explorer (https://resources.azure.com) provides a simplified UI for you to quickly navigate through your Azure resource tree. It’s a powerful tool that can help you to browse underlying details of Azure resources. Figure 3-8 is not a complete ARM object model diagram. It contains only resources that are relevant to this chapter.

**FIGURE 3-8** ARM object model (partial)

**Traffic Manager**

When you implement a globally-available web application, you need a global routing service. Azure Traffic Manager routes incoming traffic to your application deployments at different geographic locations based on performance and availability.

To use Traffic Manager, you define a Traffic Manager profile that consists of a domain name, a list of endpoints, and a load-balancing policy. When a user tries to access a service, the following activities happen:

1. The user accesses the service by the domain name provided by Traffic Manager (*.trafficmanager.net). If a custom domain is used, another DNS resolution is performed to first resolve the custom domain name to the Traffic Manager domain name.
2. When Traffic Manager receives the DNS resolution request, it evaluates its policy and picks an endpoint address based on availability, performance, weighted priorities or geo locations.
3. Traffic Manager returns a CNAME record that maps the Traffic Manager domain name to the selected endpoint.
4. The user’s DNS server resolves the endpoint address to its IP address and sends it to the user.
5. The user calls the endpoint directly by the IP address.
A couple of points are worth discussing here. First, Traffic Manager works during the DNS resolution phase. The actual traffic doesn’t go through Traffic Manager. Second, because DNS records are often cached, Traffic Manager isn’t involved in every service request. Third, the endpoints don’t need to be on Azure. They can be on other cloud platforms, or even in on-premises datacenters.

Traffic Manager picks endpoints based on one of the following four methods:

- **Priority** Traffic is routed to a primary service endpoint and routed to backup endpoints when the primary fails.
- **Weighted** Traffic is distributed across a set of endpoints based on weights.
- **Performance** Traffic Manager periodically updates a table that records the response time between various IP ranges to Azure datacenters. When a new request comes in, it picks the datacenter with the best response time in corresponding IP range.
- **Geographic** Traffic is routed to the same region where the DNS query originates from. This feature is important to meet data sovereignty requirements in some regions.

You can also nest Traffic Manager profiles, which means a profile at a higher level uses other Traffic Manager endpoints as candidate endpoints. Using nested profiles, you can implement more complex policies. For example, you can have a top-level profile that uses the priority method to establish a primary site and a secondary site, and a second-level profile that distributes user traffic based on performance. You can have up to 10 levels of nested profiles.

---

**Quick check**

As mentioned above, Traffic Manager can manage endpoints from different environments, as long as the endpoints are reachable over Internet. Can you leverage this feature in your hybrid deployment scenarios?

**Quick check answer**

Traffic Manager can help you to achieve hybrid scenarios such as failover-to-cloud. In this case, you’ll set up your primary endpoint to be your public on-premises entry point, and a secondary endpoint on Azure. The on-premises services provide fast, private services to your customers. And when the on-premises system fails, your customers are automatically redirected to Azure hence, the service continuity is ensured. Similarly, you can set up cross-cloud failovers to have your primary system running on your preferred cloud platform while having a backup system running on a secondary cloud.

---

**CDN**

Azure operates out of facilities located in over 30 regions around the world, and the number is increasing every year. In addition, Azure also strategically places CDN point of presence (POP) locations to deliver content to end users. You can cache content from Azure Storage, Azure Web Apps, and Azure Cloud Services.
When a user requests content by the CDN URL, the content is directly served from the CDN node if the content exists. Otherwise, the content will be retrieved from the content origin and stored at the CDN node for future requests.

Using CDN has two major benefits. First, because content is served directly from the CDN node that is closest to the user, user experience can be greatly improved. Second, because a large portion of requests will be served from CDN nodes instead of from the original service nodes, the loads on the original service nodes are greatly reduced, making it possible for the service to scale out to support a much greater number of users.

CDN is mostly used to cache static contents. However, you can cache dynamic outputs from your websites and Cloud Services, as well because CDN content is identified by URLs, including the query parameters. For example, http://<identifier>.vo.msecnd.net/chart.aspx?item=1 and http://<identifier>.vo.msecnd.net/chart.aspx?item=2 represent two different cached objects. You need to be careful not to cache volatile data in CDN, because doing so can adversely affect your performance or even cause content problems, all at increased cost.

Routes

Now, let’s dig deeper into how network packets are routed on Azure. When you place multiple VMs on an Azure Virtual Network, the VMs can communicate with one another by their private IP addresses or host names. The VMs can also reach the Internet. All of these are made possible by Azure-provided system routes.

System routes

When you provision a VM on a virtual network subnet, a few default routes are defined, as listed in Table 3-1.

<table>
<thead>
<tr>
<th>Address Prefixes</th>
<th>Next Hop Type</th>
<th>Next Hop Type IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual network CIDR, for example: 10.0.0.0/16</td>
<td>Virtual network</td>
<td>-</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>Internet</td>
<td>-</td>
</tr>
<tr>
<td>10.0.0.0/8</td>
<td>None</td>
<td>-</td>
</tr>
<tr>
<td>100.64.0.0/10</td>
<td>None</td>
<td>-</td>
</tr>
<tr>
<td>172.16.0.0/12</td>
<td>None</td>
<td>-</td>
</tr>
<tr>
<td>192.168.0.0/16</td>
<td>None</td>
<td>-</td>
</tr>
</tbody>
</table>

In Table 3-1, other than how the Virtual Network hop type is always set to the virtual network’s address space, all other system routes are static. Routes are processed via the Longest Prefix Match (LPM) method; thus, the most specific routes have the highest priority to be applied. Being the least specific, 0.0.0.0/0 is the “catch-all” route when no other routes apply.
VM-TO-VM ROUTES
Figure 3-9 depicts how traffic between two VMs on different subnets is routed. In this case, the most specific 10.0.0.0/16 route is activated, and the packet is routed through local virtual network to the destination. Please note that although 10.0.0.0/8 and 0.0.0.0/0 could also apply, they are not used because they are less specific.

Consider another example: a packet is targeted at address 10.1.1.2. In this case, 10.0.0.0/8 would be the most specific matching route, and the packet is dropped because the next hop type of the route is “None.”

VM-TO-INTERNET ROUTES
Private IPs are not routable outside the virtual network (or connected virtual networks). For a VM to communicate with the Internet, it needs to be associated with a routable public IP. As introduced earlier in this chapter, if a VM doesn’t have a public IP, Azure uses SNAT to associate a public IP so that it can participate in packet routing. If a VM is in a load balancer pool, Azure uses the public IP associated with the load balancer for SNAT. If a VM has a public IP address, there’s no SNAT needed. Packets are directly routed in this case.

Figure 3-10 shows the details of how a packet from a VM with private IP 10.0.0.4 is routed to bing.com. From the perspective of bing.com, the request comes from the load balancer public IP. From the perspective of the VM, it never knows about the intermediate hop and thinks it’s directly communicating with bing.com.
User Defined Routes

You can define custom routes to implement complex routing scenarios. For example, you may want to inspect all packets flowing through your network. Or you may want to implement a custom firewall and force all traffic to go through the firewall. You may even want to set up a common NAT. For these scenarios, you can create a custom routing table, using User Defined Routes (UDR) to gain fine-grained control over network traffic.

DEFINING USER DEFINED ROUTES

The best way to learn UDR is to see it in action. In the following exercise, you’ll extend your virtual network topology and insert a simple virtual network appliance between the frontend subnet and the backend subnet. Then, you’ll define a UDR to force all traffic from the frontend to the backend to go through the virtual network appliance you configure. The finished topology is shown in Figure 3-11.

**FIGURE 3-10** VM-to-Internet Routing
First, create a new subnet to host your virtual appliance. Name this subnet “appliance” and set its CIDR to 10.0.3.0/24:

```bash
az network vnet subnet create --name appliance --vnet-name <virtual network name> --resource-group <resource group name> --address-prefix 10.0.3.0/24
```

2. Create a NIC. There are a few things worth noticing in the following command. First, the NIC is assigned with a static private IP 10.0.3.4. When we set up UDR, this IP address will be the routing destination. Second, the NIC is created with the ip-forwarding flag, which allows the NIC to forward network packets.

```bash
az network nic create --vnet-name <virtual network name> --private-ip-address 10.0.3.4 --name appliance-nic --ip-forwarding --subnet appliance --resource-group <resource group name>
```

3. Now, create the virtual appliance VM with the above NIC.

```bash
az vm create --resource-group <resource group name> --name appliance-vm --admin-password <admin password> --authentication-type password --admin-username <admin user name> --size Standard_DS1_v2 --image UbuntuLTS --nsg "" --nics appliance-nic
```

4. Create a custom route table named appliance-route:

```bash
az network route-table create --resource-group <resource group name> --name appliance-route
```
5. Define a custom route on the route table to route all traffic targeted at 10.0.1.0/24 to the virtual appliance at 10.0.3.4:

```
az network route-table route create --resource-group <resource group name> --name route-to-backend --route-table-name appliance-route --address-prefix 10.0.1.0/24 --next-hop-type VirtualAppliance --next-hop-ip-address 10.0.3.4
```

6. Apply the custom routing table to the frontend subnet:

```
az network vnet subnet update --resource-group <resource group name> --vnet-name <virtual network name> --name frontend --route-table appliance-route
```

7. SSH into one of the frontend VMs via its public IP:

```
Ssh <admin user name>:<front VM public IP>
```

8. Use the traceroute command to trace the route from the frontend VM to 10.0.1.4 (if you don't have traceroute installed, use the command `sudo apt-get install traceroute` to install it):

```
traceroute 10.0.1.4 -m 3
```

The above command will fail with the following outputs:

```
traceroute to 10.0.1.4 (10.0.1.4), 3 hops max, 60 byte packets
1  * * * 
2  * * * 
3  * * * 
```

This is because the virtual appliance doesn't forward any traffic yet. You'll fix this next.

9. From the SSH session, use SSH again to connect to the appliance VM:

```
ssh <admin user name>@10.0.3.4
```

10. Enable IP forwarding:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

11. Type `exit` to exit the SSH session to 10.0.3.4 and return to the SSH session to the frontend VM.

12. Use the traceroute command again to trace the route to 10.0.1.4. This time, you should see how the packet is routed through the virtual appliance (10.0.3.4) and sent to the destination (10.0.1.4):

```
traceroute to 10.0.1.4 (10.0.1.4), 3 hops max, 60 byte packets
1 10.0.3.4 (10.0.3.4)  1.337 ms  1.326 ms  1.324 ms 
2 10.0.1.4 (10.0.1.4)  1.807 ms  1.796 ms  1.788 ms 
```

13. The traffic going from the backend to the frontend doesn't go through the virtual appliance. To verify this, SSH into the backend VM:

```
ssh <admin user name>@10.0.1.4
```
14. Use the traceroute command to trace the route back to 10.0.0.4:

```
traceroute 10.0.0.4
```

The above command should succeed showing a direct route to 10.0.0.4:

```
traceroute to 10.0.0.4 (10.0.0.4), 30 hops max, 60 byte packets
1  10.0.0.4 (10.0.0.4)  1.883 ms  1.859 ms  1.849 ms
```

**NOTE  EXPERIMENT ON WINDOWS**

If you’re using Windows Server instead of Ubuntu, you should use the tracert command instead of `traceroute`. Furthermore, to enable a Windows server to route traffic, install the Routing and Remote Access role, and then configure the role to enable the server as a IPv4 router for Local area network (LAN) routing.

**VIRTUAL NETWORK APPLIANCE**

You created a simple IP forwarder in the above exercise. In a real-life deployment, you should use existing virtual network appliances from your favorite brands to deliver advanced networking features. Azure Marketplace provides many popular networking products such as Barracuda Web Application Firewall (WAF), Cisco Cloud Services Router (CSR) 1000V, F5 WAF Solution for ASC and FortiGate Next Generation Firewall for HA. Please visit https://azuremarketplace.microsoft.com/en-us/marketplace/apps/category/networking?page=1&subcategories=appliances for a complete list of virtual network appliances.

**Skill 3.2: Design external connectivity for Azure Virtual Networks**

Microsoft realizes that for many of its existing enterprise customers, a migration to the cloud is a long process that might take years or even decades. In fact, for some of these customers, a complete migration might never be feasible. To ensure smooth cloud transitions, Azure provides a pathway for enterprises to adopt the cloud at their own pace. This means that for the foreseeable future, many enterprises will be operating hybrid solutions that have components running both on-premises and in the cloud. Thus, reliable, secure, and efficient connectivity between on-premises datacenters and cloud becomes a necessity. This skill discusses two of the connectivity options: Azure Virtual Network and Azure ExpressRoute. Then, we briefly introduce some other hybrid solution options.

This section covers how to:

- Design hybrid solutions with Virtual Network and ExpressRoute
- Leverage other hybrid solution options
Hybrid connectivity
The virtual network offers several types of hybrid connections that bridge resources located at different facilities. You can choose one or several connection options that best suit your requirements. Note that this skill does not focus on detailed steps of setting up the connections. Instead, it describes the steps in general and then focuses on how each connection type suits different scenarios.

Point-to-Site VPN
Point-to-Site VPN is the simplest hybrid connection that enables you to securely connect your local computer to an Azure virtual network. No specific VPN devices are needed in this case. Instead, you install a Windows VPN client through which you can connect to any VMs and Cloud Services within the virtual network. Figure 3-12 shows the topology of a Point-to-Site VPN.

![FIGURE 3-12 Point-to-site connectivity](image)

Establishing a point-to-site connection involves several steps:

1. Specify an IP address range. When your VPN clients connect, they will receive IP addresses from this range. You need to ensure that this range doesn’t overlap with IP ranges within your on-premises network.
2. Add a gateway subnet.
3. Create a Route Based VPN (previously known as dynamic routing gateway). You can create up to 128 P2S connections through a single gateway.
4. Create a client certification to be used for client authentication. The client machine that makes the VPN connection needs to have the certificate installed.
5. Download the VPN client configuration package from your virtual network’s dashboard page. When the client is installed, you’ll see a new VPN connection with the same name as your virtual network.

With Point-to-Site connection, you can connect to your VMs on Azure from anywhere. It uses Secured Socket Tunneling Protocol (SSTP), which means that you can establish the connection through firewalls and Network Address Translation (NAT). It works well to support a small mobile workforce. However, because each client PC in this case establishes a separate connection to the gateway, you are limited to the number of P2S connections that the gateway can support.
Point-to-Site enables scenarios such as remote administration of cloud resources, troubleshooting, monitoring, and testing. It can be applied to use cases such as remote education, mobile office, and occasional command and control. However, for bridging on-premises networks and Azure Virtual Networks, you’ll probably want to use Site-to-Site VPN.

**Site-to-Site VPN**

Site-to-Site VPN is designed for establishing secured connections between site offices and the cloud or bridging on-premises networks with virtual networks on Azure. To establish a Site-to-Site VPN connection, you need a public-facing IPv4 address and a compatible VPN device or Routing and Remote Access Service (RRAS) running on Windows Server 2012. (For a list of known compatible devices, go to [https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpn-devices](https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpn-devices).) You can use either policy-based or route-based gateways for Site-to-Site VPN. However, if you want to use both Site-to-Site VPN and Point-to-Site VPN at the same time, you’ll need a route-based gateway. Figure 3-13 shows the topology of a Site-to-Site VPN.

![Site-to-Site connectivity](image)

**FIGURE 3-13** Site-to-Site connectivity

Site-to-Site VPN extends your local network to the cloud. As you gradually move your workloads to the cloud, you often need the servers in the cloud and the local servers to still work together before the migration is complete. Using Site-to-Site VPN, these servers can communicate with each other as if they are on the same local network. This becomes handy when you move some domain-joined servers to the cloud, but you still want to keep them on your local Active Directory.

Site-to-Site works in the other direction as well: it brings your VMs in the cloud into your local network. You can join these servers into your local domain and apply your security policies on them. In many migration cases, moving the application servers is easier compared to moving a large amount of data. And some enterprises prefer to keep their data local for various reasons. With Site-to-Site VPN, your cloud VMs can reach back to your on-premises data. They also can be joined to Azure Load Balancer to provide high-availability services.
Although Site-to-Site connections provide reasonable reliability and throughput, some larger enterprises require much more bandwidth between their datacenters and the cloud. Moreover, because VPNs go through the public Internet, there’s no SLA to guarantee the connectivity. For these enterprises, ExpressRoute is the way to go.

ExpressRoute

ExpressRoute provides private connections between your on-premises datacenters and Azure datacenters. You can achieve up to 10 Gbps of bandwidth with the dedicated, secure, and reliable connections. These connections don’t go through the public Internet, and you can get connectivity SLAs from your selected service providers. If you have frequent large-volume data transfers between your on-premises datacenters and Azure, ExpressRoute provides a faster solution that in some cases is even more economical.

There are two ways to use ExpressRoute to connect to Azure. One way is to connect to Azure through an exchange provider location. The other way is to connect Azure through a network service provider. The exchange provider option provides up to 10 Gbps of bandwidth. The network service provider option provides up to 1 Gbps of bandwidth. In either case, Azure configures a pair of cross-connections between Azure and the provider’s infrastructure in an active-active configuration to ensure availability and resilience against failures. Figure 3-14 shows the topology of an ExpressRoute connection.

![ExpressRoute connectivity](image)

ExpressRoute’s fast and reliable connection is ideal for scenarios such as data storage access, backups, and disaster recovery. For example, you can transfer and store a large amount of data to Azure Storage service while keeping your applications running on your own datacenter. For backup and disaster recovery, ExpressRoute makes data replication faster and more reliable, improving the performance as well as the reliability of your disaster recovery strategies. Moreover, you can access other Azure-hosted services such as Office 365 by using the same private connection for fast, secure access.

When working together, many servers need frequent exchanges of data. When some of the servers are moved to the cloud, the additional latency introduced by Internet connections can have a serious impact on the performance of the overall system and sometimes render the
entire system unusable. ExpressRoute provides a fast connection between your on-premises datacenters and Azure so that you can extend your local infrastructure to the cloud without having to make significant architecture or code changes.

vNet-to-vNet VPN

Just as you can establish Site-to-Site connections between your on-premises datacenters and Azure, you also can connect two virtual networks on Azure by using a VPN connection. Figure 3-15 shows the topology of a vNet-to-vNet connection.

![Figure 3-15 vNet-to-vNet connectivity](image)

You can use vNet-to-vNet VPN to support geo-redundancy and geo-presence. For example, you can use vNet-to-vNet VPN to set up SQL Always On across multiple Azure regions. Figure 3-16 shows another example, which is a cross-region three-node MongoDB replica set with a primary node and a secondary node in West US, and another secondary in West Europe. The West Europe node is for disaster recovery and is not allowed to be elected as a primary.

![Figure 3-16 Cross-region mongodb replica set](image)
You also can use vNet-to-vNet VPN in business integration scenarios. With global corporations, business units sometimes remain independent from one another, but at the same time some workflows need to be integrated. Using vNet-to-vNet, resources owned by different business units can communicate with one another while maintaining isolations between the resources (refer to the later discussions on ACLs and NSGs). Some multitiered applications need such kind of isolations as well. For instance, a new corporate website might need to consume services and data from multiple regional sites, which have their own virtual networks and security policies.

**Multi-site VPN**

You can use an Azure Virtual Network gateway to establish multiple Site-to-Site connections. This capability makes it possible to join multiple on-premises networks. Figure 3-17 shows the topology of a Multi-site VPN.

![Multi-site VPN Diagram](image)

**FIGURE 3-17** Multi-site VPN

Using Multi-site VPN, branch offices from different geographic locations can connect with one another to exchange data and share Azure-based resources, such as a common hosted service. This topology is also referred to as a *hub-and-spoke* topology, which is quite common for scenarios in which a head office connects to multiple branch offices.
Network peering
When you connect two virtual networks without overlapping IP ranges in the same region together, you can use virtual network peering to connect the two networks through the Azure backbone network without gateways. Network peering has a few advantages over vNet-to-vNet connections:

- Traffic between virtual machine in peered virtual network is routed directly through the Azure backend infrastructure instead of the gateway. The network latency for a round trip between two virtual machines in peered virtual network is the same as for a round trip within a single virtual network.
- There aren’t additional bandwidth constraints other than the bandwidth allowed for the virtual machine.

Network peering has some limitations:

- Peered virtual networks must be in the same Azure region (at the time of writing cross-region peering is in preview).
- Peering is between two networks and is not transitive. For example, if virtual network A is peered with network B and virtual network B is peered with network C, network A and network C are not peered.
- Azure-provided DNS name resolution for virtual machines doesn’t work across peered network. VMs with internal DNS names are resolvable only within the local virtual network. You can set up your own DNS server for name resolutions across the peered networks.

Skill 3.3: Design security strategies
Network security is always an important factor in enterprise IT operations. One of the main responsibilities of a network administrator is to ensure the enterprise network remains effective, extensible, auditable, and secure. Azure virtual network environment replicates the features you usually see in on-premises networks, including connectivity, isolation, firewall, access control, and application gateways. In this section, you’ll review some of the Azure Virtual Network security features, and study a couple of security scenarios.

This section covers how to:

- Apply Network Security Groups (NSGs)
- Deploy Azure Application Gateway

Network Security Groups
A Network Security Group (NSG) defines a list of prioritized security rules that allow or deny traffic to subnets or individual network interfaces attached to VMs (see Figure 3-8).
NSG rules

An NSG rule is defined by the following properties:

- **Name**  A region-wide unique name.
- **Protocol**  Protocol to match for the rule. You can use TCP, UDP, or * that matches UDP, TCP as well as ICMP.
- **Source port range**  Source port range to match for the rule. It can be a single port number from 1 to 65535, a port range such as 8080-8088, or * for all ports.
- **Destination port range**  Destination port range to match for the rule.
- **Source address prefix**  Source address prefix or tag to match for the rule. It can be a single IP address such as 10.0.2.4, a subnet such as 192.168.1.0/24, a default tag, or * for all addresses.
- **Destination address prefix**  Destination address prefix or tag to match for the rule. It can be a single IP address, a default tag, or * for all addresses.
- **Direction**  Direction of traffic. It can be either inbound or outbound.
- **Priority**  Priority is a number between 100 to 4096, inclusive. A smaller priority number indicates a higher priority. When Azure applies NSG rules, it goes from highest priority rule to the lowest priority rule. Once it finds a matching rule, the rule is applied, and no further matchings are tested. If no matching rules are found, the packet is dropped.
- **Access**  Allow or deny access.

**NOTE  DEFAULT TAGS**

Default tags are system-provided identifiers that indicate IP address categories. There are three default tags: VirtualNetwork, AzureLoadBalancer, and Internet. VirtualNetwork includes virtual network address spaces, connected on-premise address spaces as well as connected virtual networks. AzureLoadBalancer indicates Azure’s load balancers. It resolves to an IP where Azure’s health probes originate. Internet denotes the IP address space outside the virtual network address space and reachable by the Internet.

- **Access**  Allow or deny access.

**DEFAULT RULES**

Each NSG comes with a set of default rules that can’t be deleted. These rules are assigned with low-priority numbers so that they can be easily overridden by your own custom rules. Table 3-2 lists the default inbound rules of an NSG. The first rule allows all internal traffic within the same virtual network. The second rule allows health probes from Azure Load Balancer health probes. And the third rule denies all other inbound traffic.

**TABLE 3-2  Default inbound rules of an NSG**

<table>
<thead>
<tr>
<th>Priority</th>
<th>Source IP:Port</th>
<th>Destination IP:Port</th>
<th>Protocol</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>65000</td>
<td>VirtualNetwork:*</td>
<td>VirtualNetwork:*</td>
<td>*</td>
<td>Allow</td>
</tr>
<tr>
<td>65001</td>
<td>AzureLoadBalancer:*</td>
<td><em>:</em></td>
<td>*</td>
<td>Allow</td>
</tr>
<tr>
<td>65500</td>
<td><em>:</em></td>
<td><em>:</em></td>
<td>*</td>
<td>Deny</td>
</tr>
</tbody>
</table>
Table 3-3 lists the default outbound rules of an NSG. The first rule allows outbound traffic to the virtual network. The second rule allows outbound traffic to Internet. And the third rule denies all other outbound traffic.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Source IP:Port</th>
<th>Destination IP:Port</th>
<th>Protocol</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>65000</td>
<td>VirtualNetwork:*</td>
<td>VirtualNetwork:*</td>
<td>*</td>
<td>Allow</td>
</tr>
<tr>
<td>65001</td>
<td><em>:</em></td>
<td>Internet:*</td>
<td>*</td>
<td>Allow</td>
</tr>
<tr>
<td>65500</td>
<td><em>:</em></td>
<td><em>:</em></td>
<td>*</td>
<td>Deny</td>
</tr>
</tbody>
</table>

**SPECIAL RULES**

Besides the default rules, a NSG also contains two special rules that allow essential Azure infrastructural traffic to go through. If you block the following two rules, your VMs won’t work properly as expected.

- **Traffic to 168.63.129.16**  As introduced earlier in this chapter, 168.63.129.16 is a special virtual IP address where infrastructural services such DHCP, DNS and health monitoring services reside. This IP maps to the physical IP of the host where the VM is hosted. The host acts as a DHCP relay, DNS recursive resolver, and probe target for load balancer and health probes.

- **Outbound traffic through port 1688** All Windows images running in VMs need to be licensed. Queries to a hosted Key Management Service are sent through port 1688. To ensure proper licensing, you should not block outbound traffic through port 1688.

**RULE FOR REMOTE ACCESS**

When you associate a public IP to a NIC, Azure automatically creates an NSG with an inbound rule that allows SSH at port 22 for a Linux VM, or RDP at port 3389 for a Windows VM. If a VM has multiple NICs, you can associate different NSGs to different NICs.

**Apply NSGs**

NSGs are applied in different orders for inbound traffic and outbound traffic. For inbound traffic, subnet-level NSGs are applied first and then the NIC-level NSGs. For outbound traffic, the order is reversed: NIC-level NSGs are applied first and then the subnet-level NSGs.

Now, let’s consider how to add network protections to the topology depicted in Figure 3-7. Logically, the overall deployment is split into two tiers and each corresponds to a separate subnet (frontend and backend). Let’s consider each layer separately.

**THE FRONTEND TIER**

Assume the frontend is comprised of two load balanced Windows web servers that are accessible through both port 80 for HTTP and port 443 for HTTPS. The two web servers are joined to an Azure Load Balancer with HTTP-based probes periodically querying the root path “/”. In addition, one of the web servers is double-purposed as a jump box to access VM instances in
the virtual network. A Load Balancer NAT rule is configured to map RDP connections at port 3389 to the selected server.

You don’t need to define a rule for load balancing because it’s already covered by the default rules. For HTTP and HTTP traffic and RDP connection, however, you’ll need to define additional inbound rules, as listed in Table 3-4.

**TABLE 3-4** Inbound rules for the frontend

<table>
<thead>
<tr>
<th>Priority</th>
<th>Source IP:Port</th>
<th>Destination IP:Port</th>
<th>Protocol</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Internet:80</td>
<td>*:80</td>
<td>TCP</td>
<td>Allow</td>
</tr>
<tr>
<td>1001</td>
<td>Internet:443</td>
<td>*:443</td>
<td>TCP</td>
<td>Allow</td>
</tr>
<tr>
<td>1002</td>
<td>Internet:3389</td>
<td>*:3389</td>
<td>TCP</td>
<td>Allow</td>
</tr>
</tbody>
</table>

Please note that for the RDP access, the destination IP is set to * instead of a specific IP. This is because in the case of using dynamic private IPs, you can’t predict or guarantee a stable IP address for a VM. Alternatively, you can define a separate NSG with the third rule only and apply it directly to the jump box VM. This avoids accidentally granting access to other frontend server’s remote access if the NAT rules on the load balancer is configured incorrectly.

Because there are no needs for the web servers to send outbound traffic, all outbound traffic to the Internet should be blocked, as shown in Table 3-5.

**TABLE 3-5** Outbound rules for the frontend

<table>
<thead>
<tr>
<th>Priority</th>
<th>Source IP:Port</th>
<th>Destination IP:Port</th>
<th>Protocol</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td><em>:</em></td>
<td>Internet:*</td>
<td>*</td>
<td>Deny</td>
</tr>
</tbody>
</table>

**THE BACKEND TIER**

Assume the backend runs two SQL database servers with data replication enabled. You need to open port 1433 for database accesses. Moreover, you may want to open port 3389 to allow RDP traffic from the jump box. The inbound rules are listed in Table 3-6.

**TABLE 3-6** Inbound rules for the frontend

<table>
<thead>
<tr>
<th>Priority</th>
<th>Source IP:Port</th>
<th>Destination IP:Port</th>
<th>Protocol</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>10.0.0.0/24:3389</td>
<td>*:3389</td>
<td>TCP</td>
<td>Allow</td>
</tr>
<tr>
<td>1001</td>
<td>10.0.0.0/24:1433</td>
<td>*:1433</td>
<td>TCP</td>
<td>Allow</td>
</tr>
</tbody>
</table>

Please note that in both rules, only connections from the frontend subnet are allowed. The outbound rule for the backend is the same as the outbound rule for the frontend—it blocks all outbound Internet traffic.
Figure 3-18 shows the updated topology with NSGs.

![Network topology with NSGs](image)

**FIGURE 3-18** Network topology with NSGs

In the following exercise, you’ll implement the NSG described in Table 3-3 using the Azure management portal. This exercise assumes you’ve already implemented the network topology in Figure 3-7.

1. Logon to the Azure management portal.
2. In the left navigation bar, click on the Resource groups icon. And then, in the Resource Group List, click on your resource group name to open the resource group.
3. On the resource group Overview blade, click on the +Add icon at the top of the screen.
5. On the Network Security Group blade, click on the Create button.
6. On the Create Network Security Group blade, enter a name for your NSG. Leave all other boxes at default values, and click on the Create button.
7. Once the NSG is created, the NSG blade should open. If the blade doesn’t open, click on the notification icon at the top of the portal (see Figure 3-3) and click on the latest notification to bring the blade up.
9. Click on the +Add icon to add a new inbound rule.
10. On the Add Inbound Security Rule blade, select Service Tag in the Source field. Then, select Internet in the Source service tag field. Change the Destination port ranges to 80, and Protocol to 80. Then, click on the OK button to create the rule, as shown in Figure 13-9.
11. Repeat the above step and create rules for HTTPS and RDP. The result NSG is as shown in Figure 3-20 (you can toggle default rules visibility by clicking on the Default Rules icon).
Azure Application Gateway

Azure load balancers are level-4 load balancers that provide round-robin load balancing. Azure Application Gateway is a level-7 load balancer with many features such as HTTP(S) round-robin load balancing, cookie-based session affinity, URL path-based routing, web application firewall, and SSL termination. In this section, we cover several different scenarios, and you’ll learn how to use Azure Application Gateway to implement these scenarios.

Azure Application Gateway Components

The Azure Application Gateway configuration is comprised of the following components:

- **Frontend IP configurations** An application gateway can be bound to a public IP, a private IP, or both. This address is the entry point of your web applications behind the gateway.

- **Listeners** A listener listens to incoming traffic on a given port and triggers associated routing rule. For a multi-site listener, it’s activated only when the request’s host name matches with its host name property.

- **Backend pools** A backend pool represents a routing target. VMs added to the same backend pool are treated as equal peers, and they share the traffic load that is routed to the pool.

- **HTTP settings** HTTP settings describe a specific set of HTTP characteristics of a backend pool such as if cookie-based affinity should be enabled, request timeout, protocol, and port.

- **Rules** A rule binds a listener with backend pools. A basic rule binds a listener to a single backend pool. With a basic rule, when a listener is activated, its captured traffic is routed to the backend pool. A path-based rule binds a listener to multiple backend pools. Once a listener captures a request, the request is analyzed, and the request path is retrieved. Then, different backend pools are selected based on the patterns in the request path.

- **Health probes** By default, Application Gateway probes the root folder of your web application for health updates. If you need to use a different health report page, configure custom health probes and assign them to your backend pools.

Scenario: Simple load balancing

In the simplest case, you can set up Azure Application Gateway for simple round-robin load-balancing. For example, if you have two VMs supporting a website at port 80, you can set up your Azure Application Gateway as shown in Figure 3-21.
Scenario: Serving contents from different servers

In this scenario, you host a media website. You are hosting regular website artifacts such as web pages and images from a pool of regular web servers. You have also configured specialized media server that can serve up video clips efficiently. What you need in this case is URL path-based routing. Specifically, when a request with a “/video” path segment in the URL comes in, it will be routed to your special media server pool instead of the default server pool that serves up the rest of the web content. Figure 3-22 shows the architecture of such a deployment.

To implement the above scenario, use the following walkthrough.

1. Add a new subnet named “gateway” with CIDR 10.0.2.0/24 to your virtual network. Application Gateway needs to be put into an empty subnet.
2. Click on the + icon at upper-left corner of the portal. Then, select Networking, Application Gateway.
3. On the Basics blade, enter a name for the gateway. Leave the tier operation as Standard. We’ll introduce Web Application Firewall (WAF) later in this chapter.
4. Select the Medium SKU size. Application Gateway is offered in three sizes: Small, Medium and Large (VM sizes). The small instance size is intended for development and testing scenarios only. Larger SKU sizes provide more throughput. For example, a large instance can provide 50Mbps to 200Mbps with SSL offload enabled.

5. Set Instance count to 2. To ensure gateway availability, it’s recommended to have at least two gateway instances for a production environment.

6. Select the subscription, resource group, and location you want to use. For this exercise, use the same resource group used to host the topology in Figure 3-9. Then, click the OK button to continue.

7. On the Settings blade, select the virtual network you’ve been using. Then, select the subnet you created for the gateway in Step 1.

8. Set the IP address type to Public for Frontend IP configuration. Then, click on the Choose a public IP address link, and then click on the Create new link. Give a name to the public IP, and click on the OK button to continue.

9. Set Listener configuration to HTTP at port 80. Then, click the OK button.

10. On the Summary blade, click on the OK button to create the gateway.

11. Once the gateway is created, open its blade and click on the Backend pools link. You’ll see there’s already an empty backend pool created. Click on the pool name to open it.

12. Click on the Add target button. Select the Virtual machine option. Then, pick one of the frontend VMs. And, click on the Save button.

13. Back to the backend pools list, click on the +Add button to add another backend pool.

14. Name the new pool “videobackendpool.” Add the other frontend VM to the new pool.

15. Now, you need to delete the default basic rule and replace it with a URL path-based rule. However, because the portal doesn’t allow deleting the default rule because it’s the last rule associated with the gateway, you need to create a temporary rule so that you can delete the default rule. To create the temporary rule, create a temporary listener, because a listener can be bound to only one rule.

16. Back to the gateway blade. Click on the Listeners link.

17. Click on the +Basic button to add a temporary listener. On the Add basic listener blade, set the name as “temp,” and the Frontend port and name to 88. The exact port doesn’t matter, as long it’s not 80, which has been used by the default listener. Click on the OK button to create the temporary listener.

18. Back to the gateway blade. Click on the Rules link.

19. Click on the +Basic button to add a temporary rule. On the Add basic rule blade, set the rule’s name to “temp,” select the temp listener, and then click on the OK button to create the temporary rule.

20. Now back to the rule list. Click on the default rule (rule1), and then click on the Delete button to delete it.
21. Back to the rule list again. Click on the +Path-based button to add a new path-based rule.

22. On the Add path-based rule blade, enter rule name as **pathbased**. Choose the default HTTP listener. Set the Default backend pool to the default backend pool in Step 12. Then, add a new row to the path-based rule with name “video”, paths “/video/*” and backend pool “videobackendpool”, as shown in Figure 3-23. Click on the OK button to create the rule.

![Add path-based rule](image)

**FIGURE 3-23** Path-based rule

23. Now you can delete the temp rule and the temp listener.

### NOTE THE UNIFORM ARM OPERATION PARADIGM

ARM provides a consistent operation model for managing all types of Azure resources. Creating, updating, and deleting resources via the Azure management portal follows the same operation path regardless of resource types. Interacting with resources via Azure CLI also follows a consistent `az <resource type> <action> <options>` paradigm. Once you’ve learned to manipulate one type of resource, you can apply similar operation steps to managing other types of resources. Both the CLI and portal use the same backend API, so they provide functionality parity in most cases. ARM also provides a unified, JSON-based resource templating language called ARM template. You can use ARM template to describe a group of related Azure resources and treat them as an integrated deployment unit.
Scenario: Protecting against common web vulnerabilities

The Internet is a hostile environment. Hackers and malicious users are like sharks swimming around to find victims that have exploitable vulnerabilities. Web Application Firewall (WAF) is an Application Gateway feature that protects your web applications from common exploits and vulnerabilities such as SQL injection and XSS attack. Figure 3-24 shows how WAF can block malicious requests before they reach your sites.

![WAF protects your sites](image)

**FIGURE 3-24** WAF protects your sites

WAF provides protections against the following attacks:

- SQL injection
- Cross site scripting
- Common web attacks such as command injection, HTTP request smuggling, HTTP response splitting, and remote file inclusion
- HTTP protocol violations
- HTTP protocol anomalies such as missing host user-agent and accept headers
- Bots, crawlers and scanners
- Common application misconfiguration (Apache and IIS etc.)

WAF provides such protections by implementing the OWSAP Core Rule Set (3.0 and 2.2.9). OWSAP Core Rule Set is a collection of generic rules for WAFs written in a SecRules language. These rules are used by WAF to provide in-depth protections against common attacks on web applications.

**NOTE** DEFENDING AGAINST DDOS

A Distributed Denial of Service (DDoS) attack is a continually rising threat. To define the massive-scale DDoS attack, significant infrastructure needs to be deployed to detect and disrupt such attacks. Microsoft Azure provides global-scale, built-in DDoS protection across all Azure datacenters that prevents common DDoS attacks such as UDP floods, SYN-ACK attacks and reflection attacks. Azure has also been investing in intelligent monitoring and analysis across Internet environments using advanced technologies such as machine learning. The collected intelligence is then used to identify and mitigate potential risks and attacks, making Azure an ever stronger and more secure cloud platform.
Scenario: End-to-end SSL

Azure Application Gateway terminates SSL connections and sends unencrypted data to back-end servers. This relieves the backend servers from the burden of encryption and decryption. However, for some customers, sending unencrypted data is unacceptable. For these customers, Azure Application Gateway supports end-to-end SSL. After it terminates the SSL connection and uses decrypted data to decide packet routes, it encrypts the data again using whitelisted backend server certificates to ensure secured communication with the backend servers. Figure 3-25 shows a sample deployment of the end-to-end SSL scenario.

Figure 3-25 also shows how you can configure SSL negotiation policies:

- SSL 2.0 and SSL 3.0 are disabled by default. These policies are not configurable.
- You can disable any of the three protocols: TLSv1.0, TLSv1.1 and TLSv1.2.
- If no SSL policy is defined, then all three protocols are enabled.

Skill 3.4: Design connectivity for hybrid applications

Skill 3.2 has introduced Azure’s hybrid networking capabilities in detail. However, for applications to thrive in a hybrid environment, additional hybrid supports are needed at higher levels. Additional services are required to facilitate accessing data across cloud and on-premises datacenters.
**This section covers how to:**
- Connect to on-premises data by using Service Bus Relay
- Use Azure App Service Hybrid Connections
- Use Azure websites’ virtual private network capability
- Identify constraints for connectivity with VPN
- Identify options for domain-joining VMs

**Connect to on-premises data by using Azure Service Bus Relay**

You can use Service Bus Relay to build hybrid applications that run in both an Azure datacenter and your own on-premises enterprises environment. Service Bus Relay facilitates this by enabling you to securely expose Windows Communication Foundation (WCF) services that reside within a corporate enterprise network to the public cloud. And it does this without requiring additional incoming firewall rules. Figure 3-26 shows the data flow using Service Bus Relay to access an on-premises WCF service.

![Diagram of Service Bus Relay architecture](image)

**FIGURE 3-26** The Service Bus Relay architecture

Using Service Bus Relay, you can host WCF services within your existing enterprise environment. You can then delegate the task of listening for incoming sessions and requests to these WCF services to the Service Bus Relay service running within Azure. This makes it possible for you to expose these services to application code running in Azure or to mobile workers or extranet partner environments though outbound connections only. With Service Bus Relay, you can securely control who can access these services at a detailed level. It provides a powerful and secure way to expose application functionality and data from your existing enterprise solutions and take advantage of it from the cloud.
Hybrid Connections

Hybrid Connections provides an easy and convenient way to connect Azure App Services to on-premises resources. Hybrid Connections is a feature of App Services. You must config minimal TCP ports to access your network. Using Hybrid Connections, you can make connections to on-premises resources that use static TCP ports, such as SQL Server, MySQL, Web APIs, and most web services. As of this writing, Hybrid Connections does not support services that use dynamic ports, such as SQL Express. Figure 3-27 shows the setup of a Hybrid Connection.

![Figure 3-27 BizTalk API App Hybrid Connections](image)

You can use Hybrid Connections with all frameworks supported by App Services (.NET, PHP, Java, Python, and Node.js). A feature of using Hybrid Connections to work with resources that are on-premises is that when you move an application such as a website from on-premises to Web Apps, you do not need to change the connection string. Connecting to on-premises resources is exactly the same as if the website were running locally. This makes it possible for Web Apps to be moved to Azure faster because there are no changes required to access the needed data.

Enterprise administrators can keep control over what resources are available to the applications in Azure and can have access to other tools to help them monitor that access. Administrators can set up group policies to determine what resource applications can be accessed through the Hybrid Connections. Event and audit logs can provide visibility into what resources are being accessed. Administrators also do not need to make changes to incoming firewall rules, because the traffic for the Hybrid Connections requires only outbound TCP and HTTP connectivity. Many administrators are reluctant to open up additional ports for security issues. Table 3-7 lists the ports that Hybrid Connections use.

<table>
<thead>
<tr>
<th>Port</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>HTTP port; used for certificate validation.</td>
</tr>
<tr>
<td>443</td>
<td>HTTPS port.</td>
</tr>
<tr>
<td>5671</td>
<td>Used to connect to Azure. If TCP port 5671 is unavailable, TCP port 443 is used.</td>
</tr>
<tr>
<td>9352</td>
<td>Used to push and pull data. If TCP port 9352 is unavailable, TCP port 443 is used.</td>
</tr>
</tbody>
</table>
To configure Hybrid Connections, follow these steps:

1. Sign in to the management portal.
2. On the left side, go to App Services.
3. Select the App Service instance you want to configure.
4. Click on the Networking link.
5. On the Networking blade, click on the Configure Your Hybrid Connection Endpoints link under the Hybrid connections section.
6. Download and install the Hybrid Connection Manager.
7. Follow the step for the Hybrid Connection Manager setup and connect it to the App Services instance in Azure.

**Web Apps virtual private network capability**

Web Apps have the ability to integrate directly into Azure Virtual Network as well as to Hybrid Connections. This does not allow you to put the website into the Virtual Network, but it does allow for the website to reference other services in the Virtual Network. If the Virtual Network is connected to an on-premises network with a site-to-site virtual private network, the website can access all of the on-premises systems such as databases.

Azure ExpressRoute is another option that you can use to connect resources in Azure to the corporate network. This is a feature that uses an ExpressRoute Partner to set up a direct connect between your corporate WAN and Azure. This is the fastest connection to create a Hybrid Connection.

**Identifying options for domain-joining Azure Virtual Machines**

You can add Virtual Machines to a domain in several different ways. First, you can set up your own domain controllers (also as Virtual Machines) on your Virtual Network in Azure. Once you have your domain controllers configured, additional virtual machines on the Virtual Network can join your domain, like how local services join the on-premises domain. Second, if you have ExpressRoute or VPN configured, you can join an Azure virtual machine back to the on-premises domain. Third, Azure Active Directory Domain Services provides a managed domain service on the cloud with Windows AD compatible domain services such as domain join, group policy, Kerberos/NTLM authentication. You can join your VMs into managed domains backed by Azure Active Directory Domain service.

You also can synchronize your on-premises directories with your cloud-based directories using Azure AD Connect, which will be introduced in Chapter 4, “Design Security and Identity Solutions.”
**Thought experiment**

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

Using isolated security zones is an effective way for enterprises to reduce many types of risks on their networks. For example, many enterprises use a perimeter network to isolate their Internet-facing resources from other parts of their internal network. You can implement the same level of protection in Azure Virtual Network as well. In this case, you have a number of VMs that will be exposed to the Internet. And you have a number of application servers and database servers on the same virtual network.

With this in mind, answer the following questions:

1. What technologies would you use to implement a perimeter network in Azure Virtual Network?
2. How would you design your network topology?

**Thought experiment answers**

This section contains the solutions to the thought experiment.

1. You can use NSGs to control network traffic to VMs. Alternatively, you can use UDR to route traffic through a virtual network appliance that implement a firewall or advanced packet filtering.

2. One possible way to design the topology is to put Internet-facing resources, application services, and database servers into different subnets. The Internet-facing resources can communicate only to application servers through specific ports. And only application servers can access database servers governed by another set of rules.

**Chapter summary**

- Azure provides rich virtual network features that simulate what you’ve been familiar with in on-premises datacenters. This helps you to lift-and-shift your existing on-premises services to cloud without redesigning your network topology.
- Azure uses Software Defined Network (SDN) to enable multi-tenant virtual networks on top of Azure datacenter infrastructures.
- You can use a combination of Azure-provided DNS and custom DNS to achieve different name resolution scenarios.
- Virtual machines added to an availability set are distributed across fault domains and update domains to reduce probability of total failure.
- You can use Internal load balancers and public load balancers to provide entry points to scaled-out services.
- Traffic Manager is designed to support cross-region, cross-site, high-available service endpoints.
- CDN is an effective way to improve response time by serving cached data directly from edge servers.
- Azure provides rich hybrid connectivity options for you to bridge your on-premises resources with your cloud-based resources.
- You can use Point-to-Site VPN, Site-to-Site VPN, ExpressRoute, vNet-to-vNet VPN, and Network Peering to bridge different networks.
- NSG rules are used to allow or deny network traffics between sources and destinations.
- Application Gateway provides level-7 load balancing for web applications.
- Azure Service Bus Relay and Azure App Service Hybrid Connections allow you to connect your cloud services with on-premises services.
Index

Symbols

2-Factor Authentication (2FA) 234
5 WAF Solution for ASC 170

A

access control 131, 224–227
    Azure AD Application Access Panel 240–242
    challenges in large organizations 233
    governance and policies for 232–242
    role-based 199, 233–238
    self-service 238–240
Active Directory Domain Services (AD DS) 200–201
Active Directory Federation Services (ADFS) 201, 210, 213–214
Active Directory Right Management Services (AD RMS) 223
activity logs 281–282, 299, 300
address spaces 151
administrator roles 225–226
AES Clear Key dynamic encryption 275
agile development 56
alerting strategy
    application 279–296
    platform 296–306
alerts 299–300
    Application Insights for 293–296
    automating 296
    Log Analytics for 282–289
    not triggering 288
    Number Of Results alert rule 288
ALTER DATABASE command 124
alternative services 63
API Apps 73
API Management 40–41, 44–45
append blobs 86
Application Insights 293–296, 297
application packages 22
Application Performance Management (APM) 293
applications
    API Apps 73
    ASP.NET 202–204, 216–219
    Azure Logic 38–40
    compute-intensive 74–79
deployable 15
development 54–57
high-performance 74–76
hybrid 187–190
integrating AI into 252–254
Linux based 17
Logic apps 311–312
migration of 47–49
monitoring 279–296
multi-tiered 198–199
refactoring 48
SaaS 233, 240–241
scaling 10–11, 17–21, 68, 71
supporting infrastructures 15–16
web 16–17, 253
Applications Solutions
    for serverless computing 38–40
App Service Plans 70–71
archive access tier 90
ARM object model 162–163
artificial intelligence (AI) 249–254
deep learning 251
defined 250
integrating into applications 252–254
machine learning and 250–251, 251–252
neural networks 251
artificial neurons 251
ASP.NET 205
ASP.NET applications 202–204, 216–219
ASP.NET Core 205
ASP.NET Core Identity 205, 206
authentication 67, 193
  Azure Active Directory 199–205
  by identity provider 195
defined 195
frameworks 205–206
Microsoft Account 217–218
multi-factor 131, 195, 234
multi-tiered applications 198–199
SAS 226
  with native clients 197–198
workflow 196
authorization
  Azure Active Directory 199–205
defined 195
  multi-tiered applications 198–199
  with native clients 197–198
workflow 196
automated testing 56
automatic backups 122
automatic failover 139
automation 2, 14, 52, 135
  about 306
  autoscaling 313
Azure Automation 307–308
Azure Logic apps 311–313
Chef 308
Desired State Configuration 309–311
Event Grid 312–313
periodic processes/tasks 314–316
Puppet 308
strategy design 306–315
Windows PowerShell for 308
autoscale 17
autoscaling 313
availability 59–60
  system-level 62–63
availability sets 3–7, 10, 160–161
AzCopy 93
Azure
  DevOps streamlining with 2
  replicating to 28
  workload hosting on 2
Azure Active Directory 199–205, 210, 225–226, 228
authentication/authorization workflow 201–205
tiers 200–201
vs. on-premises AD 200–201
Azure Active Directory (AAD) 67
Azure Active Directory Identity Protection 245–246
Azure Active Directory Sync (AAD Sync) 210
Azure AD Application Access Panel 240–242
Azure AD Authentication Library (ADAL) 199
Azure AD Connect 200, 210
  components 211–213
  installation and configuration 215
  with Express installation 214–216
Azure AD Connect Health 213–214
Azure AD Connect Sync 212
Azure Advisor 297, 300, 301
Azure Analysis Services 107–108
Azure Application Gateway 182–187
  components 182
  end-to-end SSL 187
  load balancing 182–185
  protection against web vulnerabilities 186–187
  serving contents from different servers 183–185
Azure App Service Environment (ASE) 72
Azure App Service Web Apps 58–64
Azure Automation 14, 307–308
Azure Autoscale 313
Azure B2B 220–221
Azure B2C 220
Azure Backup 26–32, 229–230
Azure Backup Agent 29
Azure Batch
  compute-intensive tasks with 21–23
  lifecycle strategy for 78–79
  when to use 77–78
Azure Bot Services 252, 254
Azure CLI
  managing virtual networks with 153–154
Azure Cognitive Services 252
Azure Container Instances 36–38
Azure Container Services 43
Azure Cosmos DB 138–146
Azure Custom Script Extension 14
Azure Database for MySQL 116–118, 122–123
Azure Database for PostgreSQL 122–123
Azure Database Migration Service 26
Azure Data Box 97–98
Azure Data Catalog 100–102
Azure Data Factory 102–103, 135–137
Azure Data Lake Analytics 105–107
Azure Data Lake Store 130–133
Azure Data Services 99–109
  Analysis Services 107–108
  Data Catalog 100–102
  Data Factory 102–103
  Data Lake Analytics 105–107
  HDInsight 109–110
  SQL Data Warehouse 103–105
Azure Disk Encryption 223
Azure Files 85, 92–94
Azure File Sync 93
Azure Functions 32–39, 44, 271–272
Azure HDInsight 109–110
Azure Health 300–301
Azure Hybrid Benefit for Windows Server 11
Azure Import/Export 93, 96
Azure IoT Edge 258
Azure IoT Hub 257
Azure Key Vault 97, 132–133, 223, 231–232
Azure Logic apps 38–40, 311–313
Azure Machine Learning 253
Azure Media Services 274–276
Azure Messaging Services 267–269
Azure Migrate 23–25
Azure Migration Center 23
Azure Monitor 280–282, 297
Azure Monitoring Metrics 290–293
Azure Performance Diagnostics VM extension 286
Azure Portal 63
Azure Quickstart Templates 12, 16
Azure Recovery Services 123
Azure Redis Cache 73, 78, 127–129
Azure Resource Explorer 163
Azure Resource Manager Templates 15–16
Azure Resources 235
Azure Rights Management Services. See Rights Management Services
Azure Search 133–134
Azure Security Center 243, 303–306
Azure Service Fabric 120
Azure Service Fabric (ASF) 44, 46
Azure Service Health 297
Azure Site Recovery 25–28
Azure SQL Database 110–113, 118–122
Azure SQL Data Warehouse 103–105, 124–126
Azure Storage 85–99, 222–223, 224–225, 227–228
  accounts 85–86, 87
  Azure Data Box 97–98
  Azure Files 92–94
Azure StorSimple 98–99
  Blob storage 86–89
  disks 94–96
  storage tiers 90–91
Azure Storage Service Encryption (SSE) 97
Azure StorSimple 98–99
Azure Stream Analytics 135
Azure subscriptions 236
Azure Table Storage 129–130, 138, 144–145
Azure Templates 8
Azure Threat Protection (ATP) 245
Azure Time Series Insights 258
Azure Toolkit for Intellij 17
Azure Traffic Manager 63
Azure Virtual Networks. See virtual networks (VNETs)
Azure VM gallery 11

backups
  application 71
  Azure Backup 26–32, 229–230
  Azure Site Recovery 26–28
  Cosmos DB 146
  recovery vaults 30
  relational databases 122–124
  size limits 31
  solution features 28–29
  StorSimple 230
  virtual machines 26–32
  website 72
  Windows Server Backup 29
Backup Vault 229–230
Barracuda Web Application Firewall (WAF) 170
basic monitoring 297
Batch. See Azure Batch
Batch accounts 23
Big Compute 21, 74
Big Data 22, 102–103, 130
  bindings 33–34, 38
  BitLocker 223
  Blob storage 29, 85, 86–89
  tiers 90–91
  Blob triggers 34, 36
  block blobs 86
  blue team kill chain 243–244
  bots 252, 254
bring your own devices (BYOD) 233  
browser metrics 296  
built-in images 17  
business continuity 71–72, 146  
business continuity and disaster recovery (BC/DR) plans 26–27  

C  
cache 73, 78, 127–129  
centralized message dispatchers 264  
Chef 14, 308–309  
Cisco Cloud Services Router (CSR) 1000V 170  
claim-based architecture 194–195, 234  
claims 194  
Classless Inter-Domain Routing (CIDR) 152  
cloud  
datacenters 3  
managing run-time environments in 14  
native deployment 47–49  
resources 1  
cloud services  
migration strategy from 23–32  
Cloud Shell 91, 117  
managing virtual networks with 153–154  
code repository 37  
columnar storage 103, 124–125  
Column-Level Encryption (CLE) 224–225  
column size 125  
columnstore tables 125  
common encryption scheme (CENS) 275  
Competing Consumer pattern 264–266  
Compute category 8  
compute generation 118  
compute-intensive applications 74–79  
compute-intensive instances 74–75  
compute-intensive tasks 21–23  
ComputeNode 23  
Compute Plane 255  
configuration files 37  
connection strings 63  
Connector Space 211  
consumption plans 34  
containers  
advantages of 41–42  
Azure Container Instances 36–38  
blobs in 87  
configuration files 37  
images 42  
orchestration 43–44, 45–46  
Web apps for 16–17  
when to use 41–42  
Content Delivery Network (CDN) 69, 164  
Content Distribution Network (CDN) 73  
continuous integration/continues deployment (CI/CD) 54  
Control Plane 255  
Convolutional Neural Network (CNN) 251  
cookbooks 14  
cool access tier 90  
cool data storage 113–115  
Cosmos DB 138–146  
advantages of 138–139  
am automatic failover 139  
Azure Table Storage 144–145  
backups 146  
business continuity 146  
consistency models 141, 145  
database creation 142  
Graph API 144  
high availability 139, 146  
latency 140–141, 145  
MongoDB API 141–142  
planning 145–146  
programming for 143  
resources relationships in 143  
SQL API 142–143  
credential validation 204  
cross-region deployments 63  
custom images 12  
custom roles 236–238  
Custom Script Extension 13–14  

dashboards 299  
data  
access control 224–227  
backing up 229–230  
Big Data 102–103, 130  
corruption 146  
deleting 90, 95–96  
persistence 127  
protection 222
recovery 124, 230–231
reliability 227–228
Database Transaction Units (DTUs) 112, 115
data disks 95
data encryption 29, 131, 222–224, 231, 275
  Azure Storage Service Encryption 97
data implementation 85–148
  Azure Data Services 99–109
  Azure Storage solutions 85–99
  NoSQL storage 126–137
  relational database storage 109–126
Data Lake Analytics 105–107
Data Lake Store 130–133
data loss 27
Data Migration Assistant (DMA) 26
data recovery 29
data security 221–232
  encryption 222–224, 231
  protection 222
datasets 103
data storage. See storage
Data Warehouse Units (DWUs) 104
deep learning 251
default tags 177
delegated user identity 198
deployment
  cloud native 47–49
  cross-region 63
  models 154
  slots 63
  to Azure Container Instances 37
  virtual machines 7–11
descriptive metadata 101
design pipelines 135–137
Desired State Configuration (DSC) 13–14, 308, 309–311
development pipeline 54, 55
  streamlining 2
device twins 257
DevOps 2, 15, 17, 42, 49–57, 286, 301, 306
DHCP server 155
directory synchronization 210–211
DirSync 210
disaster recovery (DR) 118–122
disasters 27
distributed denial of service (DDoS) attacks 186
DM-Crypt for Linux 223
DNS Analytics 298
DNS name labels 156–157
DNS server 155, 156
Docker Hub 37
Docker Swarm 46
domains
  adding VMs to 190
DRM technology 275
D-series 76
DS-Series 76
dual-datacenter scenarios 28
dynamic enterprise environment 233
dynamic IP addresses 155
dynamic scaling 61
E
eDTUs 112
Elastic database client library 116
elasticity 270
elastic pools 111, 112
Elastic Scale 298
embarrassingly parallel workloads 21
Enabling Diagnostics With Resource Manager
  templates 285
Encrypting File System (EFS) 223
event-driven actions
  Azure Functions for 32–36
event-driven triggers 314
Event Grid 269, 312–313
Event Sourcing 267
ExpressRoute 173–174, 190
ExpressRoute Monitor 298
extensible key management (EKM) 224
external state store 60
external users 238
F
failovers 26
fault domains 3–7, 61, 160
fault tolerance 228
feedback loop 57
FIM + Azure Active Directory Connector 210
firewalls 112, 117
  Web Application Firewall 186–187
FortiGate Next Generation Firewall for HA

- Functions
  - Azure Functions 32–39, 44, 271–272
  - vs. Logic Apps 39
- Fuzzy inferences 250

G

- Generalized images 12
- General-purpose storage accounts 86, 95, 130
- Geo-presence 174
- Geo-redundant storage (GRS) 227
- Geo-redundancy 174
- Geo-redundant storage (GRS) 88, 146
- Geo-replication 118–122, 127, 228
- Get-AzureVMAvailableExtension cmdlet 13
- Graph API 144
- Graph data models 144
- Graph Explorer 206–208, 209–210
- Group management
  - Self-service 239–240
- Groups 234

H

- Hadoop Distributed File System (HDFS) 130
- Hardware failures 3
- HDInsight 109–110
- Header versioning 67
- Health monitoring 62
- Health probes 161
- Heuristics 250
- High availability (HA) 2, 69–70, 111, 118–122, 139, 146, 228
- High-performance computing (HPC) applications 74–76
- Homogenous instances 60
- Horizontal scaling 313
- Hosting plans 34
- Hot access tier 90
- HPC clusters 75–76
- Hybrid applications
  - Connectivity for 187–190
- Hybrid Connections 189–190
- Hybrid connectivity 171–177, 187–190
- Hybrid identities 210–216
- Hypertext as the Engine of Application State (HATEOAS) 66
- Hyper-V Replica 28

I

- Identities
  - Delegated 198
  - Hybrid 210–216
  - Managed 199–202
- Identity providers 195, 216–221
- Identity solution
  - Authentication and authorization workflow 196
  - Authentication frameworks 205–206
  - Azure Active Directory 199–205
  - Azure Active Directory Identity Protection 245–246
  - Azure AD B2B 220–221
  - Azure B2C 220
  - Claim-based architecture 194–195
  - Designing 193–216
  - Governance and policies for 232–242
  - Hybrid identities 210–216
  - Identity providers 216–221
  - Managed identities 193–194
  - Microsoft Graph API 206–210
  - Multi-tiered applications 198–199
  - With native clients 197–198
- Images
  - Built-in 17
  - Container 42
  - Custom 12
  - In Azure Container Registry 37
  - Managing 11–12
  - Marketplace 20
  - Platform 20
- Import/Export service 223
- Independent Software Vendor (ISV) 15
- Infrastructure as a Service (IaaS) 1, 149
- Infrastructure as code (IaC) 15–16, 56
- Infrastructure availability 279
- Input files 23
- Instances
  - Compute-intensive 74–75
  - Homogenous 60
  - Primary-secondary 61
  - State-ful 60
  - Stateless 60
- Integration patterns 264–267
- Integration testing 56
- IntelliJ 17
- Intel MPI Library 75
- Interactive design 254
Internal Load Balancer 161–162
internal load balancer (ILB) address 72
Internet Information Services (IIS) 309
Internet of Things (IoT) 254–262
architecture 255
Azure IoT Edge 258
Azure IoT Hub 257
Azure Time Series Insights 258
Compute Plane 255
Control Plane 255
device twins 257
message routes 257
messaging systems 260–261
Microsoft IoT Suite 256–257
processing units 260–262
server pipeline 255
server-side pipelines 258–262
intrinsically parallel workloads 21
IP addresses 153, 154–156
dynamic 155
private 154–155
public 155–156
iterative development 56

J

jobs 23
Job Schedule 23
JSON Resource Template 56
JSON Web Token (JWT) 275

K

Katana 205
key management service (KMS) 223, 224
Key Vault. See Azure Key Vault
Kim, Gene 51
Kubernetes 46

L

Language Understanding Intelligent Service (LUIS) 252
licenses 9
lifecycle management strategies 49, 49–57, 78–79
lift-and-shift scenarios 76
Linux based apps 17, 73
live streaming 275–276
load balancers 10
load balancing 70
availability sets 160–161
Azure Application Gateway 182–187
Internal Load Balancer 161–162
public 162
round-robin 182, 182–185
rules 161
virtual networks 159–161
load leveling 61
locally redundant storage (LRS) 88, 227
Local Storage (LS) 119
Log Analytics 282–289, 298, 301–303
Logic apps 311–312
Logic Apps 38–40, 272–273
Log Search API 283
Longest Prefix Match (LPM) 165
loose coupling 61–62, 264

M

machine learning 245, 253
AI and 250–251
challenges of 251–252
Managed Disks 19
managed identities 193–194, 199–202
management portal
creating virtual networks using 150–153
management solutions 283–285, 298
man-in-the-middle attacks 275
Marketplace images 20
Master Encrypt Keys (MKEs) 131
Mean Time To Recover (MTTR) 72
Media Indexer 276
media services
Azure Media Services 274–276
designing 274–276
key components 274–276
live streaming 275–276
Media Indexer 276
Monitor Services 276
Mesosphere Enterprise DC/OS 46
Message Mapper 266
messaging gateway 266–267
messaging systems
Azure Messaging Services 267–269
benefits of 264–265
designing 262–273
for system integrations 263–267

325
metadata

integration patterns 264–267
pub/sub pattern 269–270
queuing options 268–269
reactive systems 270–273
metadata 100
metaverse 211
Metrics Explorer 299
microservices 249–278, 259, 261–262
microservices-based solutions
  API Management for 44–45
  Azure Functions for 44
  Azure Service Fabric for 44
  container-based 41–42
  lifecycle management strategies 49–57
  migration of existing assets 47–49
  Web API for 45
Microsoft Account
  authentication 217–218
  middleware 219
Microsoft Graph API 206–210
Microsoft HPC Pack 76
Microsoft IoT Suite 256–257
migration
  Azure Database Migration Service 26
  Azure Migrate 23–25
  Azure Site Recovery 25–26, 28
  of existing assets 47–49
  scenarios 48–49
  strategy 23–32
  minimal viable product (MVP) 56
mobile devices
  push notifications to 273
modeling 21
modularity 249–278
MongoDB API 141–142
monitoring systems
  activity logs 300–301
  application 279–296
    products and services for 280–282
  Application Insights 293–296, 297
  Azure Advisor 300, 301
  Azure Health 300–301
  Azure Monitor 280–282
  Azure Monitoring Metrics 290–293
  Azure Security Center 303–306
  basic 297
  Log Analytics 282–289, 297, 301–303
  Network Watcher 298, 301–303
platform 296–306
premium 297–298
Service Map 298
shared functionality 299–300
System Center 299–300
monitors 23
Monitor Services 276
multi-container architectures 43
multi-container groups 36
multi-factor authentication 131
multi-factor authentication (MFA) 195, 234
multi-shard queries (MSQ) 116
Multi-site VPN 175
multi-tiered applications 198–199
MySQL 116–118, 122–124

N

name resolution 156–159
native clients
  identity management with 197–198
NAT rules 161
Network Address Translation (NAT) 171
networking 149–192
  connectivity for hybrid applications 187–190
  connectivity options 170–176
  routes 165–170
  security 176–187
  virtual networks 149–170, 170–176
network interface card (NIC) 154
network peering 176–177
Network Performance Monitor (NPM) 283–285, 298
Network Security Groups (NSGs) 162, 176–181, 226
  applying 178–181
  backend tier 179–181
  frontend tier 178–179
  rules 176–178
network topologies 252
Network Watcher 298, 301–303
neural networks 251, 252
neurons 251
New-AzureRmResourceGroupName cmdlet 16
NoSQL storage 126–137
  Azure Data Factory 135–137
  Azure Data Lake Store 130–133
  Azure Search 133–134
  Azure Table Storage 129–130
features 126
relational databases

MongoDB API 141–142
Redis Cache 127–129
SQL API 142–143
Time Series Insights 134–135
Number Of Results alert rule 288

O

OAuth 2.0 On-Behalf-Of specification 199
Office 365 data querying 208–209
Open API Initiative 67
Open Web Interface for .NET (OWIN) 205
operating system disk 95
Operation Management Suite (OMS) 243–244
operations 56
operations design 279–318
application monitoring and alerting 279–296
automation strategy 306–315
monitoring and alerting systems 296–306
Operations Management Suite (OMS) 283–285
output files 23
overfitting 252
overprovisioning 20

P

Packer 12
page blobs 87
partitions 125, 134
passwords 9
self-service reset 239
patches 5, 37
people 50–51
performance
web apps 68–69
periodic processes/tasks 314–316
Platform as a Service (PaaS) 1, 149
platform images 20
platform independence 64
platform monitoring and alerting 296–306
platform services 249–278
artificial intelligence 249–254
Internet of Things 254–262
media services 274–276
messaging systems 262–273
PlayReady 275
Point in Time Restore feature 228
Point-to-Site VPNs 171–172
Polybase 125
Pool application packages 22
POSIX ACLs 131
PostgreSQL 116–118, 122–123
premium storage 76
pre-production 57
primary-secondary instances 61
private IP addresses 154–155
process 50, 52–53
processing units 260–262
production 57
public IP addresses 155–156
public load balancing 162
pub/sub pattern 269–270
Puppet 14, 308–309
push notifications 273

Q

quality assurance (QA) 56
query pools 108
query string versioning 66

R

Reactive Manifesto 270
reactive systems 270–273
characteristics of 270–271
Serverless and 271–273
read-access geo-redundant storage (RA-GRS) 88, 89, 228
recovery
relational databases 122–124
Recovery Point Objective (RPO) 27, 29, 72, 120–121, 267
Recovery Time Objective (RTO) 27, 29, 72, 120, 267
recovery vaults 30
Recurrent Neural Network (RNN) 251
Red Dog Front End (RDFE) 281
Redis Cache 73, 78, 127–129
red team kill chain 243–244
redundancy 59–60, 227
relational databases
Azure Database for MySQL 116–118, 122–123
Azure Database for PostgreSQL 116–118, 122–123
Azure SQL Database 110–113, 118–122
Azure SQL Data Warehouse 124
backup and recovery strategy 122–124
disadvantages of 126
go-replication 118–122
high availability/disaster recovery 118–122
retention policy 122–123
scalability and features 115–116
SQL Server Stretch Database 113–115
storage 109–126
Relying Party (RP) 194
remote connections 149
remote direct memory access (RDMA) 74
Remote Storage (RS) 119
replicas 134
replication
to Azure 28
Request Unit (RU) 145
resiliency 270
resource classes 125
Resource Groups 15, 16, 236
Resource Manager 8
resources 236
creating 7–11
deployment models 154
scaling 10–11, 17–21
resource scopes 235
Resources Metrics 290
responsiveness 270
representation state transfer (REST) 64
REST APIs 64–65
retention policy 122–123
reusability 249–278
Rights Management Services (RMS) 223, 231
robocopy 93
role-based access control 131
role-based access control (RBAC) 199, 233–238oles 234, 235
administrator 225–226
custom 236–238
Round Robin distribution 125
round-robin load balancing 182, 182–185
routes 165–170
system 165–166
user defined 167–170
VM-to-Internet 166
VM-to-VM 166–167
Routing and Remote Access Service (RRAS) 172
Runbook Gallery 307
runbooks 14

S
scalability
autoscaling 313
designing for 115–116
stateless components for 77–78
scale-out 115
scale sets 17–21
scale-up 115
scaling out 10–11, 68, 313
scaling up 11
scheduled-driven triggers 314–315
SCSI disks 95
Secret Manager tool 218
securable entities 194
Secured Socket Tunneling Protocol (SSTP) 171
Secure Token Systems (STS) 275
Secure Transfer Required 89
security. See also authentication;
See also authorization;
See also data encryption;
See also identity solution
access control 224–227
Azure Active Directory Identity Protection 245–246
Azure Application Gateway 182–187
Azure Security Center 243, 303–306
Azure Threat Protection 245
data 221–232
identity solution 193–216
Key Vault 231–232
monitoring 303–306
networking 176–187
Network Security Groups 176–181
Operation Management Suite 243–244
Rights Management Services 231
risk management 242–245
security tokens 194, 195
self-service 238–240
Serverless compute 271–273
serverless computing
about 32
API Management and 40
Applications Solutions for 38–40
Azure Container Instances and 36–38
Azure Functions and 32–36
design solutions for 32–40
tables

- Azure Table Storage 129–130, 138, 144–145
- temporary 125
- Task application packages 22
- task output 23
- tasks 23
  - in containerized environment 37
- templates
  - Azure 8
  - Azure Quickstart Templates 12, 16
  - Azure Resource Manager Templates 15–16
  - JSON Resource Template 56
  - template (JSON) images 12
- temporary disks 9, 95
- temporary tables 125
- testing
  - automated 56
  - integration 56
  - quality assurance 56
  - user acceptance 57
- test scripts 56
- Time Series Insights 134–135
- tools 50, 53
- Traffic Analytics 298
- Traffic Manager 63, 163–165
- transaction size 125
- transient errors 61
- transient-fault handling application block 61
- Transparent Data Encryption (TDE) 223, 231
- triggers 32–33, 34, 36, 38, 136, 314–315
- trusted subsystems 198
- trust relationships 195
- Twitter 241–242

U

- uniform ARM operation paradigm 185
- update domains 4–7, 61, 160
- upgrade domains. See update domains
- upgrades/updates 5, 37
- URI versioning 66
- User Acceptance Testing (UAT) 57
- user agents 196, 197
- user defined routing 167–170
- user-managed storage 19–20

username 9
U-SQL scripts 105–107

V

- vCore 118
- versioning 66–67
- vertical scaling 313
- Virtual Hard Disks (VHDs) 94
- virtual hard drives (VHDs) 9
- Virtual Machine Agent (VM Agent) 12
- virtual machines (VMs) 1
- backing up 26–32
- deployment 7–11
- design solutions using 2–32
- discovery of 24
- disk type 9
- domain-joining 190
- features 19
- images 11–12
- just in time access 306
- migration of existing 47–48
- monitoring 286
- reconfiguring 305
- routing between 166–167
- scale sets 17–21
- scripts for 13–14
- sizing 9–10
- state management 14
- use cases 10
- virtual network appliances 170
- virtual networks (VNETs) 1, 40, 149–170
  - access control 226
  - ARM object model 162–163
  - Content Delivery Network 164
  - creating 150–153
  - external connectivity for 170–176
  - IP addresses 154–156
  - load balancing 159–161
  - managing 153–154
  - name resolution 156–159
  - network peering 176–177
  - routes 165–170
  - Traffic Manager 163–165
- virtual private networks (VPNs)
  - multi-site 175
  - point-to-site 171–172
site-to-site 172–173
vNet-to-vNet 174–175
Web Apps capability 190
Visual Studio 53
Visual Studio Team Services (VSTS) 53
VM Extensions (VM Extensions) 12
vNet-to-vNet VPN 174–175

W
waste 52
waterfall development 56
Web API 45
custom 64–67
secure 67
versioning 66–67
Web App for Containers 17–18
Web Application Firewall (WAF) 186–187
web applications
AI in 253
alternative services 63
App Service Plans 70–71
availability 59–60
Azure App Service 58–64
business continuity and 71–72
cache 73
Content Distribution Network with 73
cross-region deployments 63
custom Web API 64–67
deployment slots 63
features of 58–59
health monitoring 62
high availability 69–70
loose coupling 61–62
on Linux 73
performance 68–69
scalability 68–69
scaling 71
secure Web API 67
staging environments for 63
system-level availability 62–63
VPN capability 190
WebHDFS API 131
web hooks 210
Web Services 58
websites 58, 69
backing up 72
Windows Communication Foundation (WCF) 188
Windows PowerShell
for automation 308
Windows Server AD 201
Windows Server Backup 29

X
XSS attacks 186

Z
zone redundant storage (ZRS) 227
zone-redundant storage (ZRS) 88
ZRS Classic 88