Microsoft

Microsoft Azure Architect Design

Exam Ref AZ-304

Ashish Agrawal
Avinash Bhavsar
MJ Parker
Gurvinder Singh

Microsoft

Exam Ref AZ-304 Microsoft Azure Architect Design

Ashish Agrawal
Avinash Bhavsar
MJ Parker
Gurvinder Singh

Exam Ref AZ-304 Microsoft Azure Architect Design

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

No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-13-726889-4
ISBN-10: 0-13-726889-0

Library of Congress Control Number: 2021936186

ScoutAutomatedPrintCode

TRADEMARKS

Microsoft and the trademarks listed at http://www.microsoft.com on the "Trademarks" webpage are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

WARNING AND DISCLAIMER

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author, the publisher, and Microsoft Corporation shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the programs accompanying it.

SPECIAL SALES

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

CREDITS

EDITOR-IN-CHIEF
Brett Bartow

EXECUTIVE EDITOR
Loretta Yates

SPONSORING EDITOR
Charvi Arora

DEVELOPMENT EDITOR
Rick Kughen

MANAGING EDITOR
Sandra Schroeder

SENIOR PROJECT EDITOR
Tracey Croom

COPY EDITOR
Rick Kughen

INDEXER
Cheryl Ann Lenser

PROOFREADER
Abigail Manheim

TECHNICAL EDITOR
Saabir Sopariwala

COVER DESIGNER
Twist Creative, Seattle

COMPOSITOR
codeMantra

GRAPHICS
codeMantra

Contents at a glance

This page intentionally left blank

Contents

Acknowledgments

Ashish Agrawal Firstly, I would like to thank my family—especially my Mom and Dad for their encouragement and blessings. I'd also like to thank my wonderful and ever-patient wife, Swapna, for putting up with my crazy schedule and letting me devote many of my weekends and evenings to this book, and my boys, Devansh and Yug, for being a continuous energy source, love, and inspiration. My Guru and inspiration, Sunil Poddar, always instilled in me the work ethic and dedication needed to get projects like this across the finish line. I'd also like to thank all my friends and elders in the family for their support and blessings.

I want to thank all people I have had an opportunity to work with at Kraft Heinz, Infosys, Patni, Cognizant, and Microsoft, as well as all my incredible client organizations for everything I learned along the way.

I am grateful to Microsoft Press and the editors at Pearson for taking on this project. Thank you, Loretta, for the opportunity to contribute to this and Charvi for excellent project management. This book has been a fantastic experience. I also want to thank my co-authors for their determination and outstanding teamwork. Finally, my special thanks go to the editors and entire Pearson team, for walking through the content with a fine-tooth comb and sharing incredible feedback.

Avinash Bhavsar First and foremost, I would like to thank my parents for bringing me in their difficult times and making me capable of seeing this day. Thanks to them and the Almighty for the blessings. Special thanks to my wonderful wife for her support and inspiration. I want to apologize to (and thank them for their patience) my lovely kids, Atharva and Aayush, for not participating in their playtime while I devote that time to completing this project. Huge thanks to Loretta Yates and Charvi Arora for their support during this journey. Finally, I would like to thank my co-authors and the Pearson team for the opportunity to work on this project. Cheers and happy reading.

MJ Parker I would like to thank Bryant Chrzan for having faith in me, and allowing me to work on so many wonderful Pearson projects and for introducing me to all the amazing people at Pearson that I have had the pleasure of working with, including James Manly, Loretta Yates, Laura Lewin, and Julie Phifer. I have enjoyed all my collaborations with Pearson. It has also been my honor to be counted among authors such as Ashish Agrawal, Avinash Bhavsar, and Gurvinder Singh. Another person who I am grateful for is Rick Kughen, who made all my work look great and who never gave up on me. Lastly, I would like to thank my family because they gave up the most precious commodity of all—time spent together—so that I could pursue my dream of writing.

Gurvinder Singh I am indebted to Microsoft Press for the opportunity to co-author this book in association with Ashish Agrawal, Avinash Bhavsar, and MJ Parker. All my co-authors are well known for their professional prowess and in-depth knowledge of the Microsoft Azure Platform and need no introduction.

A 'Big Thank You' to the editors and entire Pearson team for their well-coordinated efforts and due diligence, from conceptualization to publication of this volume. I am indeed grateful to the entire Pearson Team, especially Ms. Loretta Yates and Ms. Charvi Arora, for their cooperation, support, and patience throughout this journey.

I am indeed grateful to my mother, Daljeet Kour, my wife, Jaspreet Kaur, and daughter, Amritleen Kaur, for the tremendous encouragement that helped me walk the insanely tight rope of schedules and deadlines.

Finally, I submit myself in reverence to Guru Nanak, the great spiritual Guru, whose blessings enabled an incredibly small and nondescript individual like me with wisdom and opportunity.

About the authors

ASHISH AGRAWAL is a qualified technocrat, offering two decades of multifaceted experience as a technology leader, trusted advisor, and Enterprise Cloud Architect (Infra, Apps, Data, and Security). He drives a profound influence in the cloud technology landscape with provocative thought leadership and communicates his ideas with clarity and passion. Ashish has delivered numerous successful cloud engagements for global fortune 500 companies in cloud advisory, consulting, architecture, leadership, and delivery execution roles throughout his career and is considered an Azure subject matter expert since 2010. He is a change leader with transforming teams' experience to adopt and innovate best practices leading to critical customer impacting results.

AVINASH BHAVSAR is a Microsoft certified Azure Professional with about 18 years of hands-on experience in all facets of cloud computing, such as discovery, assessment, cloud foundation build, datacenter transformation, cloud-native application development for Azure, and migration of applications and databases from on-premises to the Azure platform. He has extensive Application Development background, which includes architecture, design, development, continuous integration, and continuous delivery to Azure platform (IaaS, PaaS, and serverless).

MJ PARKER has been a programmer for 30 years and is a Microsoft Certified Trainer who has been teaching various Microsoft technologies and other platforms for 25 years. Her passion, however, is writing absolutely anything. With the help of great editors, she has published several non-technical books, as well as other technical works, including content for exams, training sessions, and courseware.

GURVINDER SINGH is a Microsoft Certified Azure Solutions Architect with about 14 years of diversified software development experience. He has a strong programming background and hands-on experience on .NET and C#. In the past few years, Gurvinder has been guiding large enterprises in the transformation of legacy applications into cloud-native architecture with a focus on migration to Microsoft Azure. He is extremely passionate about technology, especially with the Microsoft Azure platform (PaaS, IaaS, and Serverless).

Introduction

The purpose of the AZ-304 certification exam is to test your knowledge and understanding of the Microsoft Azure platform. The exam is targeted for Azure Solution Architects, including advising stakeholders responsible for translating business requirements into secure, scalable, and reliable cloud solutions. This book provides comprehensive coverage of exam domain objectives, including in-depth explanation and demonstration of real-world design scenarios. Designed for modern IT professionals, this Exam Ref focuses on the critical thinking and decision-making acumen needed for success at the Microsoft Certified Expert level.

While we've made every effort possible to make the information in this book accurate, Azure is rapidly evolving, and there's a chance that some of the screens in the Azure portal are slightly different now than they were when this book was written, which might result in some figures in this book looking different than what you see on your screen. It's also possible that other minor interface changes have taken place, such as name changes and so on.

Azure supports a wide range of programming languages, frameworks, databases, and services. Given this, IT professionals need to learn a vast range of technical topics in a short span of time. There is an overabundance of content available, which makes it difficult to find just enough study material required to prepare for the AZ 304 exam. This book will serve as prescriptive guidance for people preparing for this exam.

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 that you'll find in the text to access more information. Take the time to research and study those topics. Great information is available on Microsoft Learn, docs.microsoft.com/azure, 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 Learn 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.

Preparing for the exam

Microsoft certification exams are a great way to build your resume and let the world know about your level of expertise. Certification exams validate your on-the-job experience and product knowledge. Although there is no substitute for on-the-job experience, preparation through study and hands-on practice can help you prepare for the exam. This book is *not* designed to teach you new skills.

We recommend that you augment your exam preparation plan by using a combination of available study materials and courses. For example, you might use the Exam Ref and another study guide for your "at home" preparation and take a Microsoft Official Curriculum course for the classroom experience. Choose the combination that you think works best for you. Learn more about available classroom training and find free online courses and live events at *http://microsoft.com/learn*. Microsoft Official Practice Tests are available for many exams at *http://aka.ms/practicetests*.

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

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/learn*.

Check back often to see what is new!

Quick access to online references

Throughout this book are addresses to webpages that the author has recommended you visit for more information. Some of these links can be very long and painstaking to type, so we've shortened them for you to make them easier to visit. We've also compiled them into a single list that readers of the print edition can refer to while they read.

Download the list at
MicrosoftPressStore.com/ExamRefAZ304/downloads

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

Errata, updates, & book support

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

MicrosoftPressStore.com/ExamRefAZ304/errata

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

For additional book support and information, please visit
MicrosoftPressStore.com/Support

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*.

CHAPTER 5

Design infrastructure

Azure provides a wide range of infrastructure services such as compute, network, and application services. These infrastructure services are among the most consumed services by Azure customers around the globe. As AZ-304 is an advanced level exam, you need to understand Microsoft's infrastructure services thoroughly, use your design skills, and your experience designing solutions on the Azure platform.

This chapter looks at various ways to design solutions on the Azure platform using compute, network, application, and migration services.

Skills covered in this chapter:

- Skill 5.1: Design a compute solution
- Skill 5.2: Design a network solution
- Skill 5.3: Design an application architecture
- Skill 5.4: Design migration

Skill 5.1: Design a compute solution

A compute service is a hosting model to host and run your application on the cloud. This service provides processing power, memory, and local storage. Compute is one of the fundamental building blocks of your workload. Microsoft Azure offers various compute services such as VMs, Azure App Service, function apps, Service Fabric, and so forth to cater to your needs.

As an Azure Solutions Architect, you need to be mindful of choosing the right compute service to balance your business needs and Azure spend optimally. In this skill, you learn the various Azure compute offerings available to host your application and the differences between them to make the right choice for your application scenario.

This skill covers how to:

- Recommend a solution for compute provisioning
- Determine appropriate compute technologies
- Recommend a solution for containers
- Recommend a solution for automating compute management

Recommend a solution for compute provisioning

The first step in using Azure compute services is to provision it. Imagine an on-premises world, and you need a high-end server with 64 core 128 GB memory. For such a high-end machine, you need to go through several steps such as procurement and installation. This typically takes days to get you a required server. In Azure, you can provision the same server in a few clicks. This is the beauty of the Azure Cloud platform.

Now suppose you need hundreds of Azure VMs, all the virtual machines need an antivirus agent, a few of them need Internet Information Server (IIS), and so forth. You can do it manually, but that is going to be time consuming and error prone. The Azure platform offers multiple solution options to automate the provisioning process. In this section, you learn the various options available to provision compute on the Azure Cloud platform.

Table 5-1 shows a high-level comparison of the automation tools:

TABLE 5-1 Automation tools

	ARM Templates	Ansible	Chef	Puppet	Terraform
Agent/agentless	No	No	Yes	Yes	No
Need extra infrastructure	No	No	Yes	Yes	No
Need master server	No	No	Yes	Yes	No
Declarative	Declarative	Procedural	Procedural	Declarative	Declarative
Immutable infrastructure	Mutable	Mutable	Mutable	Mutable	Immutable
Open source	Microsoft Automation Tool	Yes	Yes	Yes	Yes
Supported cloud providers	Azure Only	All	All	All	All

Azure Resource Manager (ARM) template

An ARM template is Microsoft's native solution to provision resources quickly in Microsoft Azure. The template is a JavaScript Object Notion (JSON) file, which you can use to write code for Azure infrastructure. In the ARM template JSON file, you use a declarative syntax to define what resources you want to provision, their names, properties, and dependencies. In a single template, you can deploy multiple Azure resources with their dependencies.

Let's look at the basic structure of the ARM template shown in the following code snippet, which is broken down in Table 5-2:

```
{
"$schema": "https://schema.management.azure.com/schemas/2019-04-01/
deploymentTemplate.json#",
"contentVersion": "",
"apiProfile": "",
"parameters": { },
"variables": { },
"functions": [ ],
"resources": [ ],
"outputs": { }
}
```

TABLE 5-2 ARM template syntax

Element Name	Description
`schema`	This is the location of the JSON schema file. This field is mandatory.
`content version`	This is a version of the template defined by you to manage your templates. This field is mandatory.
`parameters`	List of values that you need to provide while deploying a template, such as the name of the VM, username, and password.
`apiProfile`	This is a collection of API versions for resource types.
`variables`	These are like programming language variables used to store a value.
`functions`	These are user-defined functions that are available within the template.
`resources`	This is the actual collection of resources that you are going to provision.
`Outputs`	This is used to assign the output value of the deployment such as the IP address, which can be passed to another deployment.

ARM templates and their parameter files can be developed using Visual Studio Code or your choice of any JSON file editor. Visual Studio Code's key features are code snippets, Azure schema completion and validation, the ability to create and validate parameter files, and template navigation.

NEED MORE REVIEW? **DEVELOP AN ARM TEMPLATE WITH VISUAL STUDIO CODE**

For more information about developing an ARM template with Visual Studio, see *https://docs.microsoft.com/en-us/Azure/Azure-resource-manager/templates/quickstart-create-templates-use-visual-studio-code?tabs=CLI*.

NEED MORE REVIEW? **QUICKSTART TEMPLATES**

A library of QuickStart Azure ARM templates with templates developed by the community is available at *https://github.com/Azure/Azure-quickstart-templates*.

ARM templates can be deployed using the Azure portal, Azure PowerShell, Azure CLI, and VS Code or Visual Studio. You can also use Azure Pipelines to deploy ARM templates. When you deploy ARM templates using either of the above methods, they are submitted to Azure Resource Manager. Azure Resource Manager parses the JSON file, fills in the parameter values, validates, sorts, and calls REST APIs of the respective resources defined in the ARM templates. Key features of Azure Resource Manager templates:

- You can quickly develop ARM templates using familiar tools such as Visual Studio, and Visual Studio Code, using declarative syntax in JSON file format.
- You can quickly deploy ARM templates using your familiar tools such as Azure PowerShell, Azure CLI, the Azure portal, Azure Pipelines, Visual Studio, and Visual Studio code.
- Integration with Azure DevOps and Azure Pipelines for CI/CD, and the Azure portal to track your deployments. You can also deploy ARM templates directly from GitHub to your Azure subscription using the **"Deploy to Azure"** or **"Deploy to Azure Gov"** action.
- A library of ARM templates which contains hundreds of commonly used ARM templates to expedite your environment provisioning.

Ansible

Ansible is an open-source automation tool designed for provisioning, configuration management, deployment, orchestration, continuous delivery, and security automation. It is an agentless tool that manages remote machines using SSH (Linux and UNIX) or WinRM (Windows). It performs automation using playbooks. Playbooks contain automation tasks. You can author playbooks using YAML (Yet Another Markup Language). Key features of Ansible are:

- Ansible is easy to set up and use.
- Ansible is an agentless tool; no software or client is required to be installed on a remote machine. It manages remote machines using SSH (Linux and UNIX) or WinRM (Windows).
- Ansible is simple and easy to learn with a low learning curve for developers, IT managers, and administrators.
- Ansible provides more than 450 modules for day-to-day tasks.
- Ansible allows you to deploy multi-tier applications easily and quickly.
- Ansible provides simple, consistent, and reliable configuration management.

Chef

Chef is an open-source infrastructure automation tool for configuration management, deployment, and compliance. Chef uses Ruby, a domain-specific language (DSL) for writing system configuration called a recipe and cookbook. It provides a multi-cloud solution, multi-OS (operating system), or hybrid (cloud and on-premises) environments.

Chef uses a client-server architecture, and it also includes workstations. The workstation is the system in which cookbooks are created and tested. The workstation sends the cookbook to the Chef server using Chef Knife. The Chef server stores all the cookbooks, recipes, and metadata. The Chef client pulls the configuration from the server and updates nodes with the configuration present on the server. Key features of Chef are:

- It provides support for multiple operating systems such as Windows, RHEL/CentOS, FreeBSD, macOS, AIX, Solaris, and Ubuntu.

- It supports all major public cloud providers.
- With Chef, you can manage hundreds of servers with few employees.
- Chef has broad and growing community support.

Puppet

Puppet is an open-source automation tool for configuration management and continuous delivery. Puppet implementation is based on the master-slave architecture. The master and slave securely communicate with each other using SSL/TLS.

The Puppet agent sends a slave state in a key-value pair to the master. The Puppet master uses the client state information and compiles a catalog, which is a desired state of the slave. The Puppet slave implements the required configuration and reports back to the master. Key features of Puppet:

- Puppet has a large community of developers and hence better documentation and pre-built modules.
- Puppet also provides commercial support.
- It is scalable, reliable, consistent, and deploys faster.

Terraform

Terraform is an open-source automation tool by HashiCorp for provisioning and configuration management. Terraform uses a declarative language called the HashiCorp configuration language (HCL) to safely and efficiently manage the environment.

Terraform can manage infrastructure deployed on-premises or in the public cloud, such as Microsoft Azure, Google Cloud Platform, or Amazon Web Services. Key features of Terraform are:

- Terraform is platform agnostic.
- The planning step of Terraform allows you to generate an execution plan which shows what Terraform is going to change and in what order.
- You can implement complex automation with minimal human interaction.
- Terraform creates resources in parallel, based on the dependency of resources, and thus improves efficiency.

Determine appropriate compute technologies

Microsoft Azure Cloud platform offers many flavors of compute services. Each compute service has its own capabilities such as manageability, scalability, flexibility, control, and cost. The AZ-304 exam expects you to have deep insights into the various compute services to make the right decision when designing and architecting Azure compute solutions.

Let’s look at each Azure compute service, its capabilities, and reasons to choose it in your solution design.

Azure virtual machines

Azure VMs are fully Infrastructure as a Service (IaaS), which provides a virtual processor, memory, storage, and network interfaces, along with the operating system of your choice. You can connect to VM by using the Remote Desktop Protocol (RDP) connection for Windows and SSH for Linux VMs, and you can take full control of a VM. You can install the required software and all the necessary configuration of the server for your application. While you get full control of the VM, the VM's manageability is your responsibility, so you need to take care of backup and OS-patching activities.

When to use an Azure VM:

- When you need to quickly migrate servers/applications from on-premises to Azure; this is also called a "lift-and-shift" or rehost of the server from on-premises to Azure.
- For migrating legacy applications that you think would be challenging to redesign/ remediate and deploy them into Azure PaaS services.
- For deploying databases with features not supported in Azure PaaS, such as SQL Server database with the full database engine, SQL Server Integration Services (SSIS), SQL Server Reporting Services (SSRS), and SQL Server Analysis Services (SSAS).
- For deploying custom off the shelf (COTS) applications that you cannot remediate and deploy into Azure PaaS services.
- When you need full control over the application server, including the operating system and services.
- When you quickly need a development and test environment for your applications, you can provision an Azure VM quickly and use the VM's auto-shutdown feature to save costs. Once your development is complete, you can delete the VMs that are no longer required.
- When you need a secondary site for your disaster recovery, you can configure the Azure region as your secondary site using Azure Site recovery. If the primary datacenter fails, you can quickly provision VMs in the secondary region for your critical workloads and delete the VM when your primary datacenter becomes available again.

Azure App Service

Azure App Service is a fully managed platform (PaaS) to deploy enterprise-grade web applications and you need to focus on the application functionality. The load balancing, high availability, backup, security, and OS patching is taken care of by the Azure platform. Azure App Service also provides features that allow you to configure scalability of your application. You can manually scale up your Azure App Service Plan from, say, basic to standard tier, and then behind the scenes, the platform scales the infrastructure as per the plan and vice versa.

Azure App Service is useful for hosting web applications, REST APIs, and mobile backends. It provides Windows and Linux operating systems. You can develop applications using your choice of languages such as .NET, .Net Core, PHP, Ruby, Java, Python, and Node.js. You can also develop and deploy background tasks as web jobs in Azure App Service. You can run executables developed using a programming language such as .NET, Java, PHP, Python, or Node.js, or scripts such as .cmd, .bat, PowerShell, or Bash. The web jobs can be scheduled or triggered by a specific action.

Let’s look at the key features of Azure App Service:

- **Manageability** Automatic patching of the operating system and language framework.
- **Scalability** You can scale up or scale out manually, or you can also configure auto-scaling.
- **Availability** Microsoft provides 99.95 percent availability for Azure App Service excluding applications deployed in the free and shared tiers.
- **Security** You can protect your application by configuring Active Directory authentication, IP address restriction, encryption, and managed identity.
- **Compliance** App services are PCI-, ISO-, and SOC-compliant.
- **Ease of development** Microsoft provides a dedicated tool for rapid application development. It also offers ready-made templates in the Azure Marketplace, such as WordPress. You can also easily deploy other CMS solutions such as Drupal to Azure App Service using Web App for Containers service offering. This is a standard offering by Microsoft. Maybe we can provide a link for more info for readers - https://azure.microsoft.com/en-us/services/app-service/containers/.
- **Continuous Integration and Continuous Delivery (CI/CD)** It provides CI/CD support with Azure Pipelines, BitBucket, GitHub, Azure Container Registry, and Docker Hub.
- **Backup** Azure App Service provides manual as well as automatic backup at scheduled times. You can restore the app or create another app from the backup.

When to use Azure App Service:

- When you would like to offload manageability of your application’s underlying operating system and infrastructure to the Microsoft Azure Cloud platform and configure management aspects with ease such as automatic patching of OS and language framework, backup, security, and compliance
- When your application needs infrastructure to handle fluctuating traffic
- When migrating web applications from on-premises to Azure with the luxury of time and effort to remediate application code to fit the application into PaaS to get the most benefit of the cloud

Azure Service Fabric

Azure Service Fabric is a Platform as a Service (PaaS) offering, facilitating the development, packaging, deployment, and management of highly scalable microservices and containers. It is a distributed system that provides infrastructure designed to run stateless and stateful microservices across the Service Fabric cluster of machines. You could create a Service Fabric cluster using Windows or Linux operating systems in Azure, on-premises, or other cloud providers.

Let’s look at the key features of the Service Fabric:

- **Development and management** Simple and quick microservices development and application lifecycle management.
- **Near-real-time analysis** Service Fabric allows you to perform near-real-time data analysis, event processing, parallel transaction, and in-memory computation in your application.
- **Compliance** Azure Service Fabric is ISO-, PCI DSS-, SOC-, GDPR-, and HIPAA-compliant.

- **Ease of development** You could easily build a Service Fabric application using Visual studio or your choice of Integrated Development Environment (IDE). You could also use Service Fabric Explorer to visualize the node health and application state, such as warnings and errors.
- **Continuous integration and continuous delivery (CI/CD)** Azure Service Fabric provides CI/CD support with Azure DevOps, BitBucket, and GitHub.

When to use Azure Service Fabric:

- When you are developing a new application based on microservices architecture or event-driven architecture to develop highly available and scalable microservices
- For developing applications that require low-latency reads and writes, such as gaming and session-based integrative applications
- For IoT applications to collect and process data from millions of devices
- For data analytics and workflow processing applications that require optimized reads and writes to process events or streams reliably.

Azure Functions

Azure Functions is a Function as a Service (FaaS), which abstracts underlying infrastructure and operating systems and allows you to execute smaller tasks at a scheduled time or when triggered by external events.

You can develop Azure Functions in various languages, such as C#, F#, Java, JavaScript, Python, PowerShell, and TypeScript. You can write code and execute the function without worrying about the infrastructure to run the application.

Azure also provides the following templates to help you quickly get started with function development:

- `TimerTrigger` Schedule your code to execute at predefined times.
- `QueueTrigger` Run your function code when a new message arrives in the Azure Storage queue.
- `HTTPTrigger` Trigger the execution of code based on the HTTP request.
- `CosmosDBTrigger` Run your function code to process new or modified Azure Cosmos DB documents.
- `EventGridTrigger` Run your function code to respond to Azure Event Grid events.
- `EventHubTrigger` Respond to events delivered to an Azure Event Hub.
- `ServiceBusQueueTrigger` Run your function code when a new message arrives in the Azure bus queue.
- `ServiceBusTopicTrigger` Run your function code to respond to the service bus topic message.

Triggers can invoke Azure Functions. Triggers define how a function is called. Many triggers are available for Azure Functions such as `TimerTrigger`, which runs a function at a predefined time. Triggers have associated data that is passed as the payload to the function. An Azure function should have only one trigger and optional bindings. Bindings are a way of connecting other resources to the function. Bindings are optional, and a function can have one or more input/output bindings.

There are two types of binding:

- **Input bindings** The data that your function receives.
- **Output bindings** The data that your function sends.

Azure Functions has three hosting plans:

- **Consumption plan** As the name implies, you only pay for the consumption when your functions are running. Instances are dynamically added or removed based on the number of events. The Consumption plan's billing is based on the number of executions, execution time, and memory used.
- **Premium plan** Like a Consumption plan, Azure Functions dynamically adds or removes the host based on incoming events. The Premium plan's billing is based on the number of core seconds and memory allocated across instances. The Premium plan comes with additional features such as virtual network connectivity, pre-warmed instances, unlimited execution duration, and higher compute (up to 4 cores and 14 GB RAM).
- **Dedicated (App Service) plan** This is the same App Service plan that is mostly used with Azure App Service. The benefit of using the App Service plan is that you are using the existing underutilized App Service plan (running some other app services) for Azure Functions at no additional cost.

EXAM TIP

The AZ-304 exam typically includes one or more scenario questions to choose an appropriate answer to the given scenario. The following tips should help you select the right Azure Functions hosting plan:

- **The Premium and Dedicated plans offer virtual network integration.**
- **With the Consumption plan, you have the option to save costs because you do not need to pay for the idle compute or reserve capacity.**
- **The Premium plan is more costly than the Consumption plan.**

Let's look at the key features of the Azure Functions:

- You can build Azure Functions using various languages such as C#, F#, Java, JavaScript, Python, PowerShell, TypeScript, and Node.js. You can use NuGet and NPM libraries.
- It provides CI/CD support with Azure Pipelines, BitBucket, and GitHub.
- It is developed once and deployed into various hosting plans, Kubernetes clusters, or IoT devices for edge computing.
- You pay only when your code is running.
- It allows serverless development of serverless applications on Microsoft Azure.

- HTTP triggers can be protected using Azure Active Directory, Microsoft accounts, and Google, Facebook, and Twitter accounts.
- The Azure Functions runtime is open source and is available on GitHub.
- Integration with other Azure services such as blobs, queues, databases, Event Hub, and Event Grid.
- Auto-scaling based on the number of events/loads.
- Monitoring using Application Insights.

When to use Azure Functions:

- For infrequent tasks such as DB clean up and monthly archive.
- For the processing of service bus messages. For example, processing orders by reading messages from the service bus queue and storing the result into the database.
- For the processing of files (CSV, Images) when uploaded to Azure Storage.
- For big data processing with serverless MapReduce.
- For developing APIs with unpredictable traffic during events such as a concert/conference.
- IoT data processing where usage is high during the day and very low or nonexistent at night.
- Execution of small tasks/code using an event-driven serverless architecture.

Windows Virtual Desktop

Windows Virtual Desktop (WVD) is a desktop virtualization service on Microsoft's cloud platform. This WVD service can be accessed by your choice of a device such as Windows, Mac, Android, iOS, or any device having an HTML5 web client.

Let's look at the key features of Windows Virtual Desktop:

- A complete desktop virtualization environment in Azure without any additional gateway servers.
- Cost-efficient solution. Windows Virtual Desktop service is a cost-efficient service as you don't have to pay for this service separately; you use your existing Microsoft 365 or Windows per-user licenses. You can further optimize costs by leveraging Windows 10's multi-session capability.
- You can use your own operating system image.
- Publish multiple host pools for your workload.
- End users can use Teams and Microsoft Office and OneDrive, and get a local desktop experience.

Azure Batch

Azure Batch is a managed service designed to run large-scale parallel and high-performance computing (HPC) batch jobs in Microsoft's Azure Cloud platform.

In a typical workflow, you need to perform the following steps to run a parallel workload:

1. The client uploads files into Azure Storage. These files can include scripts or applications that process data.
2. You create a pool of compute nodes—which can be Windows or Linux VM images—and you define the size of the pool and a job to run on the workload.
3. You create a job and tasks. (A job is a collection of tasks. You associate your job with a specific pool.)
4. Azure Batch downloads input files and applications. After downloading, Azure Batch executes tasks on assigned nodes.
5. Your client application will monitor tasks that are being executed on the compute nodes.
6. Azure Batch uploads task output to Azure Storage.
7. Your client application downloads output files/data.

You don't have to pay for the Azure Batch service separately. You only need to pay for the underlying compute, network, and storage resources. An organization can use Azure Batch to deliver on-demand and high-end processing for their applications.

Let's look at the key features of the Azure Batch:

- It provides flexibility to run large-scale parallel workloads by using low-priority VMs.
- Integration with Azure Storage to upload/download data.
- Auto-scaling of the nodes allows you to add nodes, install applications, identify failures, and re-queue work.
- Monitoring using Batch Explorer and Azure Monitor's Application Insights feature.

When to use Windows Azure Batch:

- You need massive computing capacities, such as image processing and analysis, weather forecasting, and engineering simulations.
- For running intrinsically parallel workloads such as:
  - Financial risk modeling using Monte Carlo simulations
  - Data ingestion, processing, and ETL operations
  - VFX and 3D image rendering
  - Image analysis and processing
  - Media transcoding
  - Genetic sequence analysis
  - Optical character recognition (OCR)
  - Software test execution

High-performance computing (HPC)

High-performance computing (HPC)—also called "big compute"—uses many CPU- or GPU-based computers to solve complex mathematical tasks. With Azure HPC services, you get access to vast computing resources geared explicitly toward HPC workloads. For example, Azure provides various high-performance computing resources such as H-series virtual machines for memory-bound applications, N-series virtual machines for graphic intensive and CUDA/OpenCL based applications, and Cray fully dedicated and customized supercomputer delivered as a managed service.

With Azure HPC, you also have a choice to burst your HPC applications into Azure using data stored in on-premises NAS devices with HPC cache or using Azure NetApp files to access large amounts of I/O with sub-millisecond latency. You have an option to use a high-throughput storage solution, such as Cray ClusterSor, which is a Lustre-based, bare-metal HPC storage solution that is fully integrated with Azure.

Many industries use HPC to solve some of their most challenging problems, such as

- Genomics
- Oil and gas simulations
- Finance
- Semiconductor design
- Engineering
- Weather modeling

There are multiple ways to design and implement HPC in Azure. Typically, it includes the following components:

- **HPC head node** A VM that acts as a managing server and takes care of scheduling workload and jobs to a worker node.
- **Virtual machine scale sets** These are the worker nodes that execute the allocated tasks.
    - **Virtual network** This provides connectivity between the head node, compute node, and storage nodes.
    - **Storage** This node allows the storage of structured, unstructured, and executable files. This can be Azure Blob Storage, Azure Data Lake Storage Gen 2, Disk Storage, and Azure Files.
    - **Azure Resource Manager** Azure Resource Manager templates and script files used to deploy the application.

NEED MORE REVIEW? **COMMON SCENARIO TO BUILD HPC SOLUTION**

To learn more about common scenarios to build an HPC solution, see *https://docs.microsoft.com/en-us/azure/architecture/example-scenario*.

Let’s look at the key features of the HPC solution on Azure:

- A highly scalable solution on the Azure Cloud platform.
- A high-end CPU and GPU virtual machine and supercomputers from Cray.

- Azure's InfiniBand-enabled H-series and N-series VMs communicate over low latency and high bandwidth and provide the best HPC performance.
- Support for most common MPI libraries, including Intel MPI, OpenMPI, MPICH, MVAPICH2, Platform MPI, and all remote direct memory access (RDMA) verbs.
- Easily extends an on-premises HPC environment to Azure Cloud.

You should use HPC on Azure for all applications that require very intensive compute power, such as:

- Reservoir simulation in the oil and gas industry
- Market modeling in the finance industry
- Weather modeling in meteorology
- Gene sequencing in genetic science
- GPU-accelerated graphics applications such as 3D CAD modeling, 3D rendering, and scientific visualization

Containers

Containers provide immutable infrastructure for your application. It allows you to bundle your application code, libraries, dependencies, configuration as a container image. You can seamlessly deploy images into Azure, other cloud providers, and on-premises.

Let's look at the key features of Containers:

- Containers make your application deployment platform agnostic.
- Containers help with consistency across the environment by bundling application code and its dependencies.
- Containers are small, lightweight, and scalable.
- Containers are resilient; allow spinning up or down rapidly.
- You can run multiple applications on isolated containers on a single VM host.

NEED MORE REVIEW? **GUIDE TO CHOOSING AN AZURE COMPUTE SERVICE**

Microsoft Azure offers a variety of compute services to deploy your application. Microsoft's guidance for choosing the right compute service to meet the business needs of your application can be found at *https://docs.microsoft.com/en-us/azure/architecture/guide/technology-choices/compute-decision-tree*.

Recommend a solution for containers

Over the past few years, containerization has gained much traction. It has completely changed the IT industry, especially with organizations moving to the cloud with a multi-cloud strategy.

With that vision in mind, the Azure platform has made it incredibly simple to develop and deploy containerized applications, leveraging industry-leading container technologies.

In this section, you learn the following compute choices available in Azure to run containerized applications on Azure and understand when you would choose one over the other.

- **Azure Container Instances (ACI)** The Azure Container Instances service offering gives you the ability to spin up containers on demand without worrying about existing infrastructure such as Azure VMs. Azure manages all the underlying infrastructure mechanics transparently, and you just focus on building applications and deploying them in a readily available containerized environment. Azure Container Instances is best suited for scenarios that can operate in isolated containers and do not need orchestration. You can deploy and run small event-driven applications, simple web apps, and small batch processing jobs using Azure Container Instances, and you have the advantage of only paying for those containers. ACI is a managed service, and you get rid of infrastructure management and operational overhead, such as upgrading/patching the underlying operating system or Azure VMs.
- **Azure Kubernetes Services (AKS)** AKS is also a fully managed Kubernetes service that allows you to deploy and manage containerized applications with full-fledged container orchestration capabilities. AKS eliminates the operational and maintenance overhead, just as if you were to manage your Kubernetes deployments. As part of managed servers, Azure handles critical Kubernetes tasks such as health monitoring of underlying infrastructure, and it handles the desired state and lifecycle of containerized applications, including autoscaling, health monitoring of individual services, auto-discovery for interservice communication, and load balancing. The best part is that AKS is free, and you only pay for the agent nodes within your clusters; you do not pay for the masters controlling the AKS cluster.

EXAM TIP

Azure Kubernetes Service (AKS) provides different ways to expose your services running within the AKS cluster. To learn more about these options, see *https://docs.microsoft.com/en-us/azure/aks/ingress-basic*.

Recommend a solution for automating compute management

The first step in automating compute is provisioning. We have already seen automation of compute provisioning in the previous section. Now, we will look at the other computation automation aspects, such as configuration, update management, continuous delivery, and automation for compliance purposes. We can easily automate these jobs using Microsoft's native solutions such as Azure Automation, PowerShell desired state configuration, ARM templates, custom script extension, and Azure Pipelines. Also, there are multiple third-party solutions available in the market, such as Ansible, Chef, Puppet, Terraform, and Jenkin.

In the following sections, we'll look at Microsoft's native solution for compute automation.

Azure Automation

Azure Automation is a Cloud-based, cost-effective automation service on Microsoft's Azure Cloud platform. Azure Automation allows you to automate time-consuming, repetitive, and error-prone tasks across Azure and non-Azure environments. Following are the key features of Azure Automation:

- **Process automation** Azure Automation allows you to automate your day-to-day manual, repetitive, time-consuming, error-prone tasks. You can simply build your process logic into a PowerShell script or Python, or you can develop graphically (based on PowerShell) as a serverless runbook and schedule it as a job. It also offers hundreds of built-in PowerShell modules for everyday tasks that you can reuse in your runbook. You can also integrate easily with other systems by using these modules. You can also set up the Hybrid Runbook Worker at your on-premises location. Hybrid Runbook Worker allows you to run a runbook and connect to on-premises resources. An Automation runbook can also be exposed as a webhook and can be triggered by a monitoring system, DevOps, and ITSM.
- **Configuration management** Configuration management has two features:
  - **Change tracking and inventory** This allows you to track your infrastructure, including virtual machine states such as files, software, and registry, and you can generate alerts for unwelcome changes.
  - **Azure Automation state configuration** This allows you to manage the desired state configuration of virtual and physical machines.
- **Update management** This feature allows you to see the current compliance state of Windows and Linux VMs, create a deployment schedule, and install patches on the scheduled window.
- **Source control integration** Azure Automation supports GitHub, Azure Repos (Git), and Azure Repos Team Foundation Version Control (TFVC).
- **Heterogeneous support** Azure Automation supports Windows as well as Linux systems across a hybrid cloud environment.
- **Role-based access control** Azure Automation supports role-based access control (RBAC) to an Azure Automation account and its resources.
- **Integration** Azure Automation easily integrates with Azure services or other public systems.

Custom Script Extension

Azure custom script extension allows you to download and run a script on an Azure VM. The extension is useful for configuring the VM after provisioning. For example, you can install software, set up services, configure the server, automate the job, and so forth. The custom script can be applied using the Azure portal, Azure PowerShell, the REST API, or ARM templates. The script file can be downloaded from Azure Storage, GitHub, a local share using SMB protocol, or any other location (such as a public URL accessible from a VM). You need to ensure that Network Security Group (NSG) and firewalls are correctly configured to access the script location.

Key features of the Custom Script Extension include:

- This is a simple and easy way to run a script on a VM and configure it. You can apply custom script extensions on a VM with few clicks using the Azure portal.
- The custom script extension can be applied using Azure CLI, Azure PowerShell, ARM template, or the REST API.

Packer

Packer is HashiCorp's open-source automation tool for the creation of VM images. Packer helps automate the entire VM image creation process. You can install the necessary software/tools and customize a VM using a post-configuration script and then capture the VM as a managed disk.

Following are the key features of Packer:

- Use Packer when you need to build a hardened VM image.
- You can quickly set up an environment and use easy-to-understand JSON templates to build images.
- You can employ easy-to-use automation to create VM images that are supported on multiple clouds such as Azure, AWS, and Oracle Cloud.
- Packer works well with Terraform to create an image and install and deploy it with Terraform.
- Packer can create multiple images in parallel targeted for various platforms.
- Packer allows you to transform an artifact from the builder (AMI or VMWare image) into a Vagrant box file.

We covered other automation-related topics such as Ansible, ARM, Chef, Puppet, and Terraform earlier in this chapter. (See "Recommend a solution for compute provisioning.")

Skill 5.2: Design a network solution

With a spaghetti of cable running through the datacenter and the massive amount of networking gear such as ports, connectors, plugs, routers, and switches to manage, understanding a traditional datacenter network can be a daunting topic. Fortunately, the basic principles of cloud networking architecture are relatively straightforward.

As an Azure Solutions Architect taking the AZ-304 exam, you need to understand Azure networking services to set the foundation right because it is the glue between most of the Azure resources you must deal with for your solutions. In this skill, we are looking at various Azure networking services and their capabilities, so that you can recommend the right solutions.

This skill covers how to:

- Recommend a network architecture
- Recommend a solution for network addressing and name resolution
- Recommend a solution for network provisioning
- Recommend a solution for network security
- Recommend a solution for network connectivity
- Recommend a solution for automating network management
- Recommend a solution for load balancing and traffic routing

Recommend a network architecture

Azure Virtual Network is a foundational building block for your private network in Azure. Azure Virtual Network enables many Azure resources, such as VMs, VM scale sets, the App Service environment, App Service, and Azure Functions with virtual network integration and Kubernetes clusters, to communicate with each other securely via on-premises networks and on the Internet.

Azure provides virtual networks with the following capabilities:

- Secure communication for Azure resources to communicate with each other.
- You can configure endpoints on virtual networks for services that require Internet communication.
- A virtual network is a logical isolation that is dedicated to your Azure subscription.
- You can implement multiple virtual networks within Azure regions in your subscriptions.
- Isolation from other virtual networks.
- You can use private and public IP addresses defined in RFC 1918 and expressed in CIDR notation.
- If you use your public IP addresses as the virtual network's address space, those public IPs would not be routable from the Internet and are still private from an accessibility standpoint.
- You can connect two virtual networks by using virtual network peering. Once any two virtual networks peer, resources in one virtual network can connect to resources in other virtual networks.
- Peered virtual networks can be in the same or different regions.

By default, Azure learns routes from on-premises over ExpressRoute, routes for all peered virtual networks, and a default route to the Internet. Azure also allows customers to override these system routes with user-defined routes. You can assign user-defined routes at the subnet level.

Network topology is a critical element of enterprise-scale architecture because it defines how applications can communicate with each other. This section explores topology approaches for Azure enterprise deployments. There are three core approaches: Azure-only virtual networks, topologies based on the hub-and-spoke model, and topologies based on Azure virtual WAN.

Hub-and-spoke network topology

A hub-and-spoke network topology isolates workload while sharing services, such as identity, connectivity, and security. The hub virtual network, as the name suggests, is a central point of connectivity. Spoke virtual networks connect to the hub virtual network using virtual network peering or global virtual network peering. Typically, you would deploy network security gear, such as Azure Firewall or third-party firewall appliances in the hub. Shared services are typically deployed in the hub or as a separate spoke peered with the hub. In contrast, you would deploy individual production and non-production workloads as spoke virtual networks.

You can provision ExpressRoute gateway in the gateway subnet. Once you add an ExpressRoute gateway in the gateway subnet, you cannot deploy anything else in the gateway subnet.

In a hub-and-spoke topology, all the spoke-to-spoke communication transits through the hub. You also need to set your firewall (Azure Firewall or NVAs) as the next hop in your user-defined routes (UDR) attached to subnets in spoke virtual networks. With the UDR, you override system routes that would otherwise send all the traffic destined for an on-premises network through the gateway. With the UDR, you would set your virtual appliance as a next-hop address.

Figure 5-1 shows the implementation of the hub-and-spoke network topology. The spoke virtual networks typically host a management subnet and at least a workload subnet each. The hub virtual network hosts core networking and security solutions in subnets dedicated for gateway, management, firewalls, Active Directory, etc. You should use virtual network peering between hub-and-spoke virtual networks and express route circuit private peering connecting to an on-premises gateway and an Express Route gateway in the hub virtual network.

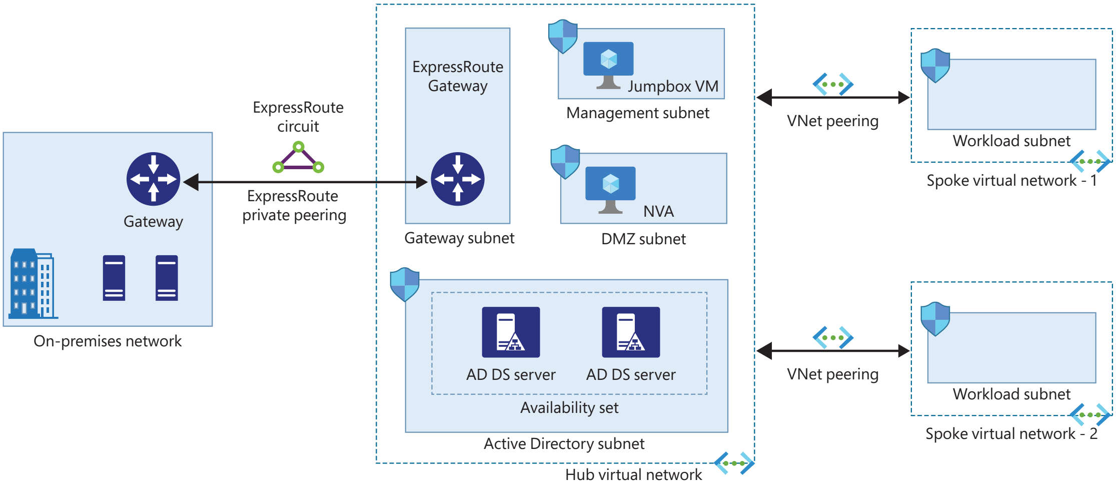

FIGURE 5-1 Hub-and-spoke topology

NEED MORE REVIEW? **HUB-AND-SPOKE TOPOLOGY**

To learn more about the hub-and-spoke topology, see *https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/hybrid-networking/hub-spoke*.

Following are the design considerations for the hub-and-spoke topology:

- Implementing a hub-and-spoke topology in Azure centralizes standard services, including connections to on-premises networks and firewalls.
- The hub virtual network acts as a central point of connectivity and hosts shared services used by workloads hosted in spoke virtual networks.
- Enterprises typically use a hub-and-spoke configuration.
- Spoke virtual networks isolate workloads; spoke-to-spoke communication goes through a hub; and a centralized firewall has visibility and can control traffic flow. Each workload can include multiple tiers.
- Azure lets you provision hub-and-spoke virtual networks in the same or different resource groups or subscriptions. You can also have spoke virtual networks in different subscriptions from that of the hub. Moreover, the subscriptions can be either associated with the same or different Azure Active Directory (Azure AD) tenants.
- This topology allows for decentralized management of each workload while sharing services maintained in the hub network.

You can use a virtual WAN to meet large-scale, multi-site interconnectivity requirements. Because a virtual WAN is a Microsoft-managed service, it reduces overall network complexity and modernizes your organization's network.

Use a traditional Azure network topology if these are your requirements:

- You intend to deploy resources across multiple Azure regions.
- You have a low number of branch locations per region.
- You need fewer than 30 IPSec tunnels.
- You require full control.
- You need granularity for configuring your Azure network.

Azure Virtual WAN topology

Azure Virtual WAN is a Microsoft-managed networking solution that provides end-to-end global transit connectivity. Virtual WAN hubs eliminate the need to configure network connectivity manually. For example, with Virtual WAN hubs, you are not required to configure user-defined routing (UDR) or network virtual appliances (NVAs) for hub-and-spoke connectivity. You can use NVAs with a virtual WAN if you require NVAs in your architecture.

Following are the design considerations for Azure Virtual WAN:

- Azure Virtual WAN simplifies end-to-end network connectivity in Azure and cross-premises by creating a hub-and-spoke network architecture with a Microsoft-managed hub. The architecture can span multiple Azure regions and multiple on-premises locations (any-to-any connectivity) out of the box, as shown in Figure 5-2. This diagram shows the global transit network with Azure Virtual WAN.

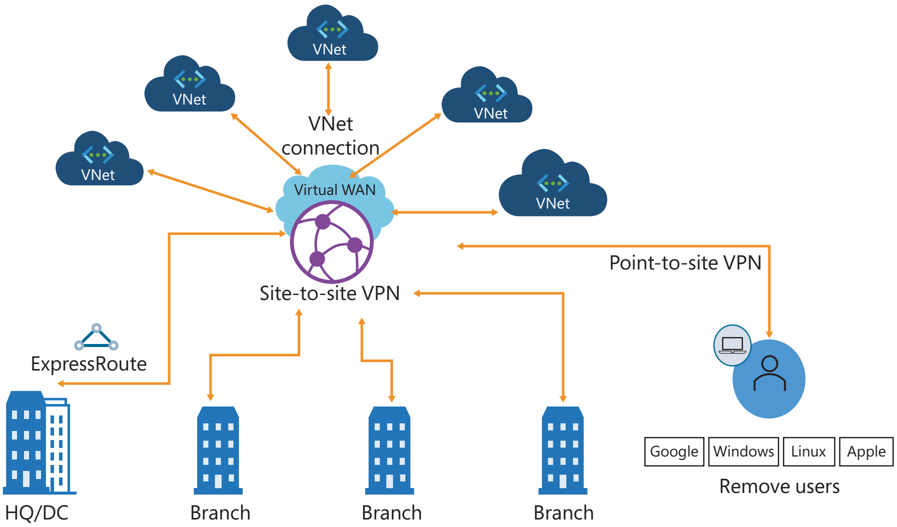

FIGURE 5-2 Global transit network with Azure Virtual WAN

- Virtual WAN hub virtual networks are locked down. You cannot deploy any resources in the WAN hub virtual network, except virtual network gateways (point-to-site VPN, site-to-site VPN, or Azure ExpressRoute); Azure Firewall through Firewall Manager; and route tables.

Azure Virtual WAN transitive connectivity supports the following:

- Virtual network to branch
- Branch to virtual network
- Branch to branch
- Virtual network to virtual network (same region and across regions)
- With Virtual WAN, you get an increased limit of prefixes advertised from Azure to on-premises via ExpressRoute private peering. The limit changes from 200 to 10,000 prefixes per virtual WAN hub. The limit of 10,000 prefixes includes prefixes advertised over site-to-site VPN and point-to-site VPN as well.
- Microsoft recently announced the general availability (GA) for virtual WAN hub-to-hub connectivity and network-to-network transitive connectivity (within and across regions) features.
- Because of the router in every virtual hub, Azure enables transit connectivity between the virtual networks in a standard virtual WAN. Every virtual hub router supports up to 50 Gbps aggregate throughput.
- Virtual WAN integrates with a variety of SD-WAN providers.

- You must use ExpressRoute circuits with the premium add-on, and they should be from an ExpressRoute Global Reach location.
- You can scale VPN gateways in Virtual WAN up to 20 Gbps and 20,000 connections per virtual hub.
- Azure Firewall Manager allows the deployment of Azure Firewall in the virtual WAN hub.

Virtual WAN is a recommended solution for new global network deployments in Azure when you need global transit connectivity across multiple Azure regions and various on-premises locations. Figure 5-3 shows an example of global deployment with datacenters spread across Europe and the United States and many branch offices across regions. The environment is connected globally via a virtual WAN and ExpressRoute Global Reach.

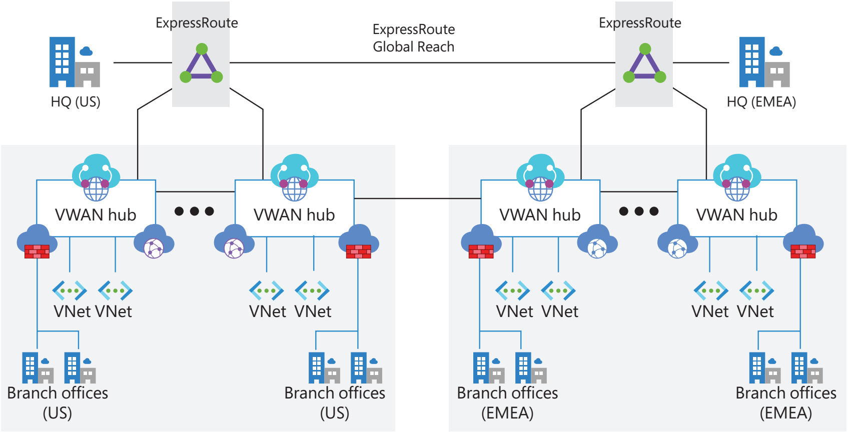

FIGURE 5-3 Global connectivity using a virtual WAN and ExpressRoute global reach

The recommended solution is to use Virtual WAN as a global connectivity resource. You can use one or many virtual WAN hubs per Azure region to connect multiple landing zones across Azure regions via local virtual WAN hubs.

Following are a few design recommendations that you should follow while implementing virtual WAN solutions:

- Connect virtual WAN hubs with on-premises datacenters using ExpressRoute.
- Deploy required shared services such as DNS or Active Directory domain controllers in a dedicated landing zone. Note that you cannot deploy such shared resources in the virtual WAN hub virtual network.
- You can connect branches and remote locations to the nearest virtual WAN hub using site-to-site VPN or branch connectivity to a virtual WAN through one of the SD-WAN partner solutions.
- You can connect users to the virtual WAN hub through a point-to-site VPN.
- We recommend that you follow the “traffic within Azure should stay in Azure” principle. With this solution, communication between Azure resources across regions occurs over the Microsoft backbone network.

- Azure Firewall in a virtual WAN hub helps with east-west and south-north traffic protection.
- Suppose you require third-party network virtual appliances for east-west or south-north traffic protection and filtering. In that case, you could choose to deploy the network virtual appliances in a separate virtual network, such as a shared virtual network. You can connect it to the regional virtual WAN hub and the landing zones that need access to NVAs.
- You do not need to build a transit network on top of an Azure Virtual WAN. The virtual WAN solution itself satisfies transitive network topology requirements. It would be redundant and increase complexity.
- Do not use existing on-premises networks such as multiprotocol label switching (MPLS) to connect Azure resources across Azure regions because Azure networking technologies support Azure resources' interconnection across regions through the Microsoft backbone.

Recommend a solution for network addressing and name resolution

You can use Azure Virtual Networks to provision and manage virtual private networks in Azure. Each Azure virtual network you create has its own CIDR block and can be linked to other virtual networks and on-premises networks if CIDR blocks do not overlap. You can segment the virtual network into subnets as needed. You can also configure your DNS setting for each virtual network.

You must provide a private IP address space using private (RFC 1918) addresses or public address space that your organization owns while provisioning an Azure virtual network. Azure assigns a private IP address to resources from the address space you assign to your virtual network. For example, when you deploy a VM in an Azure virtual network with an address space of 10.0.0.0/24, Azure assigns a VM's virtual network interface a private IP such as 10.0.0.4.

You can segment your virtual network into subnets so that you can allocate a portion of the virtual network's address space to each of those subnets. You can secure resources in subnets by associating network security groups to subnets and adding inbound and outbound NSG rules to allow or deny traffic as per your requirement.

Following are the design considerations for network addressing and name resolution:

Network addressing:

- Do not use overlapping IP address space across on-premises and Azure regions.
- You can add additional address spaces after you create a virtual network. However, when you are using virtual network peering, the process requires an outage. You are required to delete and re-create virtual network peering.
- Azure reserves five IP addresses for each subnet. You must factor in those addresses when you are sizing virtual networks and encompassed subnets.

- Azure allows you to delegate subnets to certain services to inject instances of such a service within that subnet.

Name resolution:

- Start with IP addresses from the address allocation for private networks (RFC 1918) for all your virtual network address spaces.
- Ensure that you are using non-overlapping IP address spaces across Azure regions and on-premises locations well in advance.
- In case you have limited availability of private IP addresses (RFC 1918), consider using IPv6.
- Avoid creating unnecessarily large virtual networks (for example, 10.1.0.0/16) to use available IP address spaces efficiently.
- Create virtual networks after planning the required address space and considering near-future expansion.
- Avoid using random public IP addresses for virtual networks unless those public IP addresses are owned by your organization and are not in use elsewhere on the network.
- The Domain Name System, or DNS, translates readable and easily memorable domain names or service names into its IP addresses. Azure DNS is a service for DNS domains that provides name resolution using the Microsoft Azure infrastructure.
- Resources that are deployed in virtual networks use one of the two methods to resolve domain names to internal IP addresses:
- Azure-provided name resolution (also includes Azure DNS private zones)
- Name resolution that uses a DNS server (which might forward queries to the Azure-provided DNS servers)

Azure-provided name resolution

Azure-provided name resolution provides only necessary authoritative DNS capabilities. If you use this option, Azure manages the DNS zone names and automatically records them, and you do not control the DNS zone names or the lifecycle of DNS records. If you need a fully featured DNS solution for your virtual networks, you must use Azure DNS private zones or customer-managed DNS servers.

Azure DNS supports private DNS zones in addition to supporting Internet-facing DNS domains. Azure Private DNS provides a reliable, secure DNS service to manage and resolve domain names in a virtual network without adding a custom DNS solution. By using private DNS zones, you can also use custom domain names rather than the Azure-provided names that are available by default.

You can also configure zone names with a split-horizon view, which allows a private and a public DNS zone to share the same name. To resolve DNS records of a private DNS zone from your virtual network, you must link a virtual network with that private DNS zone. Each linked virtual network can resolve all DNS records published in the private zone. You can also enable auto-registration on a virtual network link. When you enable auto-registration on a virtual network link, Azure registers VMs' DNS records on that virtual network in the private zone. When auto-registration is enabled, Azure DNS updates the zone records whenever a VM is created, changes its IP address, or is deleted.

Using your own DNS server

Domain Name System (DNS) is of the essential services in enterprise architecture. You can use your existing investments in DNS, or you can use cloud adoption as an opportunity to modernize your internal DNS infrastructure and use native Azure capabilities.

Typically, customers choose to use custom DNS servers when your name resolution needs to go beyond out-of-the-box features. Custom DNS servers within a virtual network can forward DNS queries to the Azure recursive resolvers to resolve hostnames within that virtual network. For example, a domain controller (DC) running in Azure or on-premises can respond to DNS queries for its domains and forward all other Azure queries. Forwarding queries allows VMs to see both your on-premises resources (via the DC) and Azure-provided hostnames (via the forwarder). Azure provides access to the recursive resolvers via the virtual IP 168.63.129.16.

The type of name resolution you use depends on how your resources need to communicate with each other. Below are the design considerations for a custom DNS:

- You might have a requirement to use your existing DNS solutions across both on-premises and Azure.
- You can only link just one private DNS zones to a virtual network with auto-registration enabled.
- You can link up to 1,000 private DNS zones to a virtual network without auto-registration enabled.
- You can use a DNS resolver along with Azure Private DNS for cross-premises name resolution.

Following are some design recommendations for DNS:

- If all you need is name resolution in Azure, then you can use Azure Private DNS. You can create a delegated zone for name resolution.
- If you need name resolution across Azure and on-premises, you can use the existing DNS solution (for example, Active Directory-integrated DNS) deployed on Azure VMs (two VMs for high availability). You would then configure DNS settings in virtual networks to use those custom DNS servers.
- Particular workloads such as OpenShift that require and deploy their own DNS should use their preferred DNS solution.
- You can enable auto-registration for Azure DNS to automatically manage the DNS records' lifecycle within a virtual network.
- Use a DNS on an Azure VM as a resolver for cross-premises DNS resolution with Azure Private DNS.

- Create the Azure Private DNS zone within a global connectivity subscription. You can create other Azure Private DNS zones (for example, `privatelink.database.windows.net` or `privatelink.blob.core.windows.net` for Azure Private Link) as needed.

Recommend a solution for network provisioning

Azure Virtual Network enables many Azure resources, such as Azure VMs, to securely communicate with each other, the Internet, and on-premises networks. All resources in an Azure virtual network can, by default, communicate outbound to the Internet. To communicate inbound from the Internet with a resource, you can provide a public IP address or a public Load Balancer.

When you plan to create your Azure landing zone, planning for virtual networks is usually in the first few steps. Network creation can be a daunting task in the physical world, but it is very straightforward in Azure. You can do it manually using various options such as the Azure portal, PowerShell, and CLI. However, the best practice is to use Infrastructure as Code (ARM templates or Terraform templates) to automate the provisioning process.

An ARM template is a JSON file that defines your project's Infrastructure as Code (IaC). The template uses declarative syntax, which lets you state what you intend to deploy without writing the sequence of programming commands to create it.

To create a `Microsoft.Network/virtualNetworks` resource, add the JSON shown in Listing 5-1 to the resources section of your template.

LISTING 5-1 ARM template for a virtual network

```
{
 "name": "string",
 "type": "Microsoft.Network/virtualNetworks",
 "apiVersion": "2020-06-01",
 "location": "string",
 "tags": {},
 "properties": {
  "addressSpace": {
   "addressPrefixes": [
    "string"
   ]
  },
  "dhcpOptions": {
   "dnsServers": [
    "string"
   ]
  },
  "subnets": [
   {
    "id": "string",
    "properties": {
     "addressPrefix": "string",
     "addressPrefixes": [
      "string"
     ],
```

Index