Programming Microsoft Azure Service Fabric

Haishi Bai

FREE SAMPLE CHAPTER
SHARE WITH OTHERS
Programming Microsoft Azure Service Fabric

Haishi Bai
## Contents at a glance

*Introduction*  

<table>
<thead>
<tr>
<th>PART I</th>
<th>FUNDAMENTALS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 1</td>
<td>Hello, Service Fabric!</td>
<td>3</td>
</tr>
<tr>
<td>CHAPTER 2</td>
<td>Stateless services</td>
<td>29</td>
</tr>
<tr>
<td>CHAPTER 3</td>
<td>Stateful services</td>
<td>53</td>
</tr>
<tr>
<td>CHAPTER 4</td>
<td>Actor pattern</td>
<td>69</td>
</tr>
<tr>
<td>CHAPTER 5</td>
<td>Service deployments and upgrades</td>
<td>95</td>
</tr>
<tr>
<td>CHAPTER 6</td>
<td>Availability and reliability</td>
<td>117</td>
</tr>
<tr>
<td>CHAPTER 7</td>
<td>Scalability and performance</td>
<td>145</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART II</th>
<th>SERVICE LIFECYCLE MANAGEMENT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 8</td>
<td>Managing Service Fabric with Windows PowerShell</td>
<td>167</td>
</tr>
<tr>
<td>CHAPTER 9</td>
<td>Managing Service Fabric with management portal</td>
<td>189</td>
</tr>
<tr>
<td>CHAPTER 10</td>
<td>Diagnostics and monitoring</td>
<td>203</td>
</tr>
<tr>
<td>CHAPTER 11</td>
<td>Testing</td>
<td>227</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART III</th>
<th>PATTERNS AND SCENARIOS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 12</td>
<td>Web applications</td>
<td>253</td>
</tr>
<tr>
<td>CHAPTER 13</td>
<td>Internet of Things</td>
<td>275</td>
</tr>
<tr>
<td>CHAPTER 14</td>
<td>Real-time data streaming</td>
<td>303</td>
</tr>
<tr>
<td>CHAPTER 15</td>
<td>Multitenancy and hosting</td>
<td>325</td>
</tr>
<tr>
<td>CHAPTER 16</td>
<td>Multiplayer gaming</td>
<td>345</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART IV</th>
<th>ADVANCED TOPICS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 17</td>
<td>Advanced service hosting</td>
<td>371</td>
</tr>
<tr>
<td>CHAPTER 18</td>
<td>Modeling complex systems</td>
<td>393</td>
</tr>
</tbody>
</table>
PART V  APPENDICES

Appendix A: Service Fabric subsystems and system services  421
Appendix B: Using Microsoft Azure PowerShell commands  425
Appendix C: Microsoft and containers  433
Appendix D: Pattern index  441

Index  445
Table of contents

Introduction ................................................................. xv

PART I  FUNDAMENTALS

Chapter 1  Hello, Service Fabric!  3
A modern PaaS .............................................................. 3
  Designed for agility ..................................................... 3
  Designed for QoS .......................................................... 5
  Separation of workload and infrastructure ......................... 6
Service Fabric concepts .................................................... 7
  Architecture ..................................................................... 7
  Nodes and clusters .......................................................... 9
  Applications and services ................................................. 10
  Partitions and replicas ..................................................... 10
  Programming modes ........................................................ 10
  Stateless vs. stateful ........................................................ 10
Getting started ................................................................. 11
  Setting up a development environment ............................... 12
  Provisioning a Service Fabric cluster on Azure .................... 12
Hello, World ................................................................. 17
Managing your local cluster ................................................ 22
  Visual Studio Server Explorer ......................................... 22
  Visual Studio Cloud Explorer .......................................... 23
  Service Fabric Explorer .................................................. 24
  Windows PowerShell ........................................................ 27
Additional information ....................................................... 28

What do you think of this book? We want to hear from you!
Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:

microsoft.com/learning/booksurvey
Chapter 2  Stateless services  29
Implement ASP.NET 5 applications ........................................ 29
Scalability and availability of a stateless service. ...................... 32
   Availability ............................................................... 32
   Scalability ............................................................... 32
Implement communication stacks ........................................ 33
   Default communication stack ........................................ 33
   WCF communication stack ............................................ 41
   Custom communication stack ........................................ 43
Additional information .................................................... 51

Chapter 3  Stateful services  53
Service Fabric state management ........................................ 53
Architecture of stateful services ......................................... 53
   Reliable collections .................................................... 54
   Reliable State Manager ............................................... 54
   Transactional Replicator .............................................. 55
   Logger ................................................................. 55
   Consistency ............................................................ 55
The Simple Store application ............................................. 56
   The shopping cart service ............................................ 56
   The Simple Store client .............................................. 59
   Service partition ..................................................... 61
   UniformInt64Partition ............................................... 61
   NamedPartition ...................................................... 64
Partitions and replicas ................................................... 65
   Replica roles .......................................................... 65
   Scaling ................................................................. 67
Additional information ..................................................... 68

Chapter 4  Actor pattern  69
Service Fabric Reliable Actors ........................................... 70
   Actors ................................................................. 70
| Contents |
|-----------------|-----------------|
| Actor lifetime.                                      | 70 |
| Actor states                                         | 71 |
| Actor communications                                 | 71 |
| Concurrency                                         | 71 |
| An actor-based tic-tac-toe game                      | 72 |
| Actor models                                        | 72 |
| Create the application                               | 73 |
| Define actor interfaces                              | 73 |
| Implement the Game actor                             | 74 |
| Implement the Player actor                           | 76 |
| Implement the test client                            | 76 |
| Test it                                             | 78 |
| Additional thoughts                                  | 78 |
| Timers, reminders, and events                        | 79 |
| Actor timers                                        | 79 |
| Actor reminders                                      | 80 |
| Actor events                                        | 81 |
| Actor internals                                      | 82 |
| Actor diagnostics and performance monitoring         | 82 |
| Actors and Reliable Services                         | 89 |
| Actor state providers                                | 91 |
| Additional information                               | 94 |
| **Chapter 5** Service deployments and upgrades      | 95 |
| Service Fabric application deployment process         | 95 |
| Package                                             | 95 |
| Upload                                              | 99 |
| Register/provision                                   | 100 |
| Create/replace/upgrade                               | 100 |
| Health model                                         | 100 |
| Health entities                                      | 101 |
| Health states                                        | 102 |
| Health policy                                        | 102 |
| Health reporting and aggregation                     | 103 |
Chapter 6  Availability and reliability  117

Service availability and reliability ....................................................... 117
  A broken service ............................................................................. 117
  Improving availability .................................................................... 118
  Improving reliability ...................................................................... 118

Service Fabric services availability .................................................. 119
  Service placements ........................................................................ 119
  Service failovers ........................................................................... 127
  Routing and load balancing ............................................................. 128
  Advanced rolling upgrades .............................................................. 128

Service Fabric services reliability ..................................................... 131
  Event Tracing for Windows ............................................................. 132
  Azure Diagnostics ....................................................................... 133
  Chaos tests ................................................................................... 137
  Service state backup and restore ...................................................... 142

Chapter 7  Scalability and performance  145

Scalability concepts ........................................................................... 145
  Vertical scaling vs. horizontal scaling .............................................. 145
  Scaling stateless services vs. scaling stateful services .................... 146
  Homogeneous instances vs. heterogeneous instances ................... 146
  Single-tenancy vs. multitenancy ....................................................... 147
Manual scaling vs. autoscaling ......................................... 148
Scaling a Service Fabric cluster .......................................... 148
Manually scaling a Service Fabric cluster ............................... 149
Autoscaling a Service Fabric cluster .................................... 149
Scaling with Content Delivery Network ................................. 153
Resolving bottlenecks ....................................................... 154
State bottlenecks ............................................................ 155
Communication bottlenecks ............................................... 159
Orchestration bottlenecks .................................................. 160

PART II SERVICE LIFECYCLE MANAGEMENT

Chapter 8 Managing Service Fabric with Windows PowerShell 167
Creating a secured Service Fabric cluster .............................. 167
Protecting your cluster by using a certificate ......................... 167
Client authentication using a certificate ............................... 171
Publishing applications from Visual Studio ......................... 172
Cluster management commands ......................................... 172
Queries ........................................................................... 172
Node operations .............................................................. 181
Additional cluster management commands ......................... 183
Application management commands ................................. 184
Deploying an application ................................................ 184
Upgrading an application ................................................ 185
Rolling back an application ............................................. 186
Decommissioning an application ....................................... 187
Additional information ..................................................... 187

Chapter 9 Managing Service Fabric with management portal 189
Anatomy of a Service Fabric cluster .................................... 189
Availability set ............................................................... 190
Virtual machines and NICs ............................................... 191
Virtual network ............................................................. 193
Load balancer ............................................................... 193
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Diagnostics and monitoring</th>
<th>203</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Diagnostics</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td>Configuring Service Fabric Diagnostics</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td>Using Elasticsearch and Kibana</td>
<td>208</td>
</tr>
<tr>
<td></td>
<td>Azure Operations Management Suite</td>
<td>216</td>
</tr>
<tr>
<td></td>
<td>Monitoring</td>
<td>219</td>
</tr>
<tr>
<td></td>
<td>Service Fabric Explorer</td>
<td>220</td>
</tr>
<tr>
<td></td>
<td>Visual Studio Application Insights</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td>Additional information</td>
<td>225</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Testing</th>
<th>227</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Software testability</td>
<td>227</td>
</tr>
<tr>
<td></td>
<td>Controllability</td>
<td>228</td>
</tr>
<tr>
<td></td>
<td>Observability</td>
<td>228</td>
</tr>
<tr>
<td></td>
<td>Isolateability</td>
<td>229</td>
</tr>
<tr>
<td></td>
<td>Clarity</td>
<td>230</td>
</tr>
<tr>
<td></td>
<td>Writing basic test cases</td>
<td>230</td>
</tr>
<tr>
<td></td>
<td>Setting up continuous integration</td>
<td>232</td>
</tr>
<tr>
<td></td>
<td>Preparing the Visual Studio Team Services project</td>
<td>232</td>
</tr>
<tr>
<td></td>
<td>Preparing the build machine</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>Creating a build definition</td>
<td>239</td>
</tr>
<tr>
<td></td>
<td>Running tests upon code check-ins</td>
<td>242</td>
</tr>
<tr>
<td></td>
<td>Running load tests with VSTS</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>Testability subsystem</td>
<td>246</td>
</tr>
<tr>
<td></td>
<td>Testability actions</td>
<td>247</td>
</tr>
<tr>
<td></td>
<td>Invoking testability actions using PowerShell</td>
<td>248</td>
</tr>
<tr>
<td></td>
<td>Additional information</td>
<td>250</td>
</tr>
</tbody>
</table>
## PART III
### PATTERNS AND SCENARIOS

#### Chapter 12  Web applications  253
- Azure PaaS ecosystem ........................................ 253
- App Services .................................................. 253
- Cloud Services ................................................ 256
- Service Fabric ............................................... 257
- Choosing PaaS platforms .................................. 258
- Azure Services for your web applications ............ 259
- Scenarios and patterns ..................................... 260
- E-commerce websites ....................................... 260
- Mass-source websites ...................................... 264
- Enterprise portals .......................................... 268
- Additional information ..................................... 274

#### Chapter 13  Internet of Things  275
- Azure IoT solutions ......................................... 275
- Data generation and feedback ............................ 276
- Command and control ...................................... 276
- Data ingress .................................................... 277
- Data transformation and analysis ........................ 277
- Storage .......................................................... 278
- Presentation and actions .................................... 278
- Scenarios and patterns ..................................... 279
- Remote monitoring .......................................... 279
- Other scenarios .............................................. 299
- Additional information ..................................... 301

#### Chapter 14  Real-time data streaming  303
- Real-time data streaming on Azure ..................... 303
- Five Vs of big data ........................................... 303
- Azure Stream Analytics .................................... 304
- Big data storages ............................................. 306
- Scenarios and patterns ..................................... 307
A big data solution .......................................................308
Responsive website with live data stream processing ..........317
Additional information ..................................................323

Chapter 15  Multitenancy and hosting  325
Multitenancy on Azure ..................................................325
Multitenancy vs. single tenancy .....................................326
Azure multitenant support ..........................................327
Building multitenant systems with Service Fabric ............330
Pattern: Tenant Manager .............................................331
Pattern: Cross-tenant aggregation ..................................332
Pattern: Self-service ..................................................333
Tenant by partitions ....................................................334
Pattern: Metadata-driven system ..................................335
Hosting multitenant systems ........................................339
Hosting service processes .........................................339
Pattern: Throttling Actor .............................................340

Chapter 16  Multiplayer gaming  345
Messy Chess .............................................................345
Chessboard and game goal ..........................................346
Challenges ...............................................................347
Game board ..............................................................348
Game pieces .............................................................352
Players ....................................................................355
Game hosting ............................................................357
A.I. Quests ...............................................................358
Game world .............................................................358
Player interactions .....................................................363

PART IV  ADVANCED TOPICS

Chapter 17  Advanced service hosting  371
A canonical PaaS platform ............................................371
Chapter 18  Modeling complex systems  393

Adaptive complex systems  .................................................. 393
Complex systems and complicated systems  ......................... 394
Emergence  ........................................................................... 394
A simple model  .................................................................. 396
Modeling and computational modeling  .............................. 396

The termite model  ............................................................... 397
Set up the solution  .............................................................. 397
Implement the Box service  .................................................... 398
Implement the termite actor  ............................................... 400
Implement the test client  ..................................................... 402
Test and analysis  ................................................................. 403

Service Fabric for complex systems .................................. 404
Distributed data structures  .................................................. 405
Actor Swarms  ..................................................................... 407
The spatial segregation model ........................................ 409
Set up the solution ..................................................... 409
Implement shared array with proposal supports .............. 409
Implement the virtual actor ........................................... 412
Implement the Actor Swarm ......................................... 414
Implement the test client .............................................. 416
Test the model .......................................................... 417
Future works ............................................................ 418

PART V APPENDICES

Appendix A: Service Fabric subsystems and system services 421
Appendix B: Using Microsoft Azure PowerShell commands 425
Appendix C: Microsoft and containers 433
Appendix D: Pattern index 441
Index 445

What do you think of this book? We want to hear from you!
Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:
microsoft.com/learning/booksurvey
Introduction

Azure Service Fabric is Microsoft’s new platform as a service (PaaS) offering for developers to build and host available and scalable distributed systems. Microsoft has used Service Fabric internally for years to support some of Microsoft’s cloud-scale applications and Azure services such as Skype for Business, Cortana, Microsoft Intune, Azure SQL Database, and Azure DocumentDB. The same platform is now available to you to write your own highly available and highly scalable services.

*Programming Microsoft Azure Service Fabric* is designed to get you started and productive with Azure Service Fabric quickly. The book covers fundamentals, practical architectures, and design patterns for various scenarios such as the Internet of Things (IoT), big data, and distributed computing. For fundamentals, the book provides detailed step-by-step walkthroughs that guide you through typical DevOps tasks. For design patterns, the book focuses on explaining the design philosophy and best practices with companion samples to get you started and moving in the right direction.

Instead of teaching you how to use Azure Service Fabric in isolation, the book encourages developers to make smart architecture choices by incorporating existing Azure services. When appropriate, the book briefly covers other Azure services that are relevant to particular scenarios.

Who should read this book

This book is intended to help new or experienced Azure developers get started with Azure Service Fabric. This book is also useful for architects and technical leads to use Azure Service Fabric and related Azure services in their application architecture.

Most of the book was written while the service was still in preview. So, this book is most suitable for readers who want to keep on the edge of Azure development. As one of the earliest Service Fabric books on the market, this book provides some early insights into a new service that is still under active development. Although the precise operational steps and programming APIs might change, the design patterns presented in this book should remain relevant into the foreseeable future.
Assumptions
This book expects that you are proficient in .NET, especially C# development. This book covers a broad range of topics and scenarios, especially in later chapters. Prior understanding of DevOps, application lifecycle management (ALM), IoT, big data, and big compute will help you understand these chapters.

Although no prior Azure knowledge is required, experience with the Azure software development kit (SDK), Azure management portal, Azure PowerShell, Azure command-line interface (CLI), and other Azure services definitely will be helpful.

This book might not be for you if...
This book might not be for you if you are a beginner in programming. This book assumes you have previous experience in C# development and ASP.NET development. Although this book covers topics in service operations, its primary audience is developers and architects, not IT pros.

Organization of this book

Finding your best starting point in this book
This book is an introduction to Service Fabric. It’s recommended that you read the chapters in the first two parts sequentially. Then, you can pick the topics that interest you Part III and Part IV.
If you are

Follow these steps

New to Service Fabric
Read through Part I and Part II in order.

Interested in applying Service Fabric in IoT scenarios
Focus on Chapter 13 and Chapter 14.

Interested in building scalable web applications
Focus on Chapters 12, 14, and 15. You may also want to skim through other chapters in Part III to discover some patterns that may be applicable to your scenarios.

Interested in gaming
Focus on Chapter 16. Also read Chapter 14, especially the section about the Web Socket communication stack, which provides satisfactory performance in many web-based multiplayer gaming scenarios.

Interested in operating a Service Fabric cluster
Chapters 8, 9, 10, and 11 introduce related tools and services. You may also want to browse through Chapters 5, 6, and 7 to understand application lifecycle management topics.

Interested in the Actor programming model
Focus on Chapter 4. Also browse through chapters in Part III because these chapters cover a number of Actor-based design patterns.

Interested in Service Fabric container integration
Focus on Chapter 17. Appendix C also gives you great background information on container integrations.

Interested in modeling complex systems with Service Fabric
Focus on Chapter 18.

Some of the book’s chapters include hands-on samples that let you try out the concepts just learned. No matter which sections you choose to focus on, be sure to download and install the sample applications on your system.

**System requirements**

You will need the following hardware and software to run the sample code in this book:

- Visual Studio 2015.
- Latest version of Azure SDK (2.8 or above, install via Web PI).
- Latest version of Azure PowerShell (1.0 or above, install via Web PI).
最新版本的 Azure CLI。

- 4 GB (64-bit) RAM。
- 30 GB 的可用硬盘空间。
- 活跃的 Microsoft Azure 订阅。你可以在 www.azure.com 上获取免费试用。
- 使用 Azure 以及下载软件或章节示例所需的互联网连接。

**Downloads: Code samples**

大多数章节的示例都包括让你可以互动地尝试新学习到的材料的示例代码。所有示例项目都可以在本书的下载网页上找到：

`http://aka.ms/asf/downloads`

**Using the code samples**

这本书的网页包含此书中的所有示例，按照对应章节的文件夹组织。它还包含两个额外的文件夹：ComplexSystems 和 AdditionalSamples。AdditionalSamples 文件夹包含额外的示例场景。ComplexSystems 文件夹包含框架和复杂系统的示例场景。

- **Chapter 1** 本文件夹包含来自第1章的示例。
  *HelloWorldApplication*: 这是一个简单的Hello World 应用程序。

- **Chapter 2** 本文件夹包含来自第2章的示例。
  *CalculatorApplication*: 这是提交堆栈示例中使用的计算器应用程序。

- **Chapter 3** 本文件夹包含来自第3章的示例。
  *SimpleStoreApplication*：简单的商店应用程序。
  *SimpleStoreApplication-NamedPartitions*: 使用命名分区的简单的商店应用程序。
Chapter 4  This folder contains samples from Chapter 4.

*ActorTicTacToeApplication*: The tic-tac-toe game using Actors.

Chapter 5  This folder contains samples from Chapter 5.

*ConsoleRedirectTestApplication*: The sample application used in package format samples.

Chapter 13  This folder contains samples from Chapter 13.

*SensorAggregationApplication-Pull*: The IoT scenario that aggregates sensor states using pull mode.

*SensorAggregationApplication-Push*: The IoT scenario that aggregates sensor states using push mode.

*IoTE2E End-to-end*: The IoT sample scenario.

Chapter 14  This folder contains samples from Chapter 14.

*NumberConverterApp*: The number converter service.

*ECommerceApplication*: The sample e-commerce application.

Chapter 15  This folder contains samples from Chapter 15.

*MetadataDrivenApplication*: The sample metadata-driven application (shows actor polymorphism).

*ThrottlingActorApplication*: The sample application shows the Throttling Actor pattern.

Chapter 16  This folder contains samples from Chapter 16. Both scenarios are under development. Please see release announcements in the repository for releasable versions.

*MessyChess*: The Messy Chess sample (under development).

*AIQuest*: The A.I. Quest sample (under development).

Chapter 17  This folder contains samples from Chapter 17.

*GuestApplication*: A simple guest application sample with a watchdog.

Chapter 18  This folder contains samples from Chapter 18.

*TermiteModel*: A simulation of termites moving and collecting wood chips.
ActorSwarmApplication: A simulation of people moving closer to neighbors with similar attributes. This sample shows a preliminary implementation of an actor swarm.

AdditionalSamples: This folder contains additional sample scenarios (see README.md under the folder).

ComplexSystems: This folder contains frameworks and samples for modeling complex systems.

To complete an exercise, access the appropriate chapter folder in the root folder and open the project file. If your system is configured to display file extensions, C# project files use .csproj as the file extension.

Acknowledgments

I’d like to thank my wonderful editor Karen Szall who has guided me through every single step along the way to get this book published. I’d also like to thank John McCabe for his insightful reviews. Especially, I’d like to thank Boris Scholl who, regardless of his busy schedule to get the service released on time, has helped me tremendously reviewing the book and providing me insights into container integrations.

I’d also like to thank the amazing team behind Service Fabric. I’ve personally worked with many of the team members including Mark Fussell, Matthew Snider, Vaclav Turecek, and Sean McKenna. The creativity and dedication of the team has inspired me while writing this book. I also gained a lot of knowledge from the Microsoft internal community, including the www.azure.com author group and the Yammer group.

Last but not least, I’d like to thank my wife Jing and my daughter Sabrina who have tolerated my late hours and busy weekends in the past six months. I couldn’t do this without your support.

Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

http://aka.ms/mspressfree

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 addresses (also known as URLs) can be painstaking to type into a web browser, so we’ve compiled all of them into a single list that readers of the print edition can refer to while they read.

The list is included in the companion content, which you can download here:

http://aka.ms/asf/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:

http://aka.ms/asf/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 mspininput@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.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://aka.ms/tellpress

We know you’re busy, so we’ve kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!
Stay in touch

Let’s keep the conversation going! We’re on Twitter: http://twitter.com/MicrosoftPress
PART I

Fundamentals

CHAPTER 1  Hello, Service Fabric! .............................. 3
CHAPTER 2  Stateless services................................. 29
CHAPTER 3  Stateful services................................. 53
CHAPTER 4  Actor pattern ...................................... 69
CHAPTER 5  Service deployments and upgrades .......... 95
CHAPTER 6  Availability and reliability ...................... 117
CHAPTER 7  Scalability and performance ................. 145
This page intentionally left blank
CHAPTER 1

Hello, Service Fabric!

This book is about Microsoft Azure Service Fabric, a distributed systems platform that makes it easy to build large-scale, highly available, low-latency, and easily manageable services. Service Fabric brings you the same technology that empowers cloud-scale applications such as Cortana, Skype for Business, and SQL databases so that you can easily design, implement, scale, and manage your own services leveraging the power of next-generation distributed computing.

Before you embark on the journey with Service Fabric, let’s reflect on what makes a great platform as a service (PaaS) and why you need a new PaaS to build the next generation of cloud-based services.

A modern PaaS

A PaaS is designed with agility, scalability, availability, and performance in mind. Microsoft Azure Service Fabric is a PaaS that is built from the ground up to support large-scale, highly available cloud applications.

Designed for agility

The software industry is all about agility. Developers have the privilege to work in a virtual world without physical constraints to drag us down. Innovations can happen at a speed that is unimaginable to other fields. And as a group, we’ve been in a relentless pursuit for speed: from software frameworks to automation tools, from incremental development to Heroku’s 12-factor methodology (Wiggins 2012), from minimum viable product (MVP) to continuous delivery. Agility is the primary goal so that developers can innovate and improve continuously.

Microservices

The essence of Microservices is to decompose complex applications into independent services. Each service is a self-contained, complete functional unit that can be evolved or even reconstructed without necessarily impacting other services.

All software can be abstracted as components and communication routes between them. Monolithic applications are hard to maintain or revise. An overly decomposed system, in contrast, is hard to understand and often comes with unnecessary overhead due to the complex interaction paths...
across different components. To a great extent, the art of a software architect is to strike a balance between the number of components and the number of communication paths.

A PaaS designed for Microservices encourages separation of concerns, emphasizes loose coupling, and facilitates flexible inter-component communication. While allowing other architectural choices, Service Fabric is designed for and recommends Microservices. A Service Fabric application is made up of a number of services. Each service can be revised, scaled, and managed as an independent component, and you still can manage the entire application as a complete logical unit. The Service Fabric application design is discussed in Chapter 2, “Stateless services,” Chapter 3, “Stateful services,” and Chapter 4, “Actor pattern.” We’ll review several application patterns and scenarios in Part III.

**Note** Architecture choices

Microservices is strongly recommended but not mandatory. You can choose to use other architectures such as n-tiered architecture, data-centric architecture, and single-tiered web applications or APIs.

### Simplicity

A PaaS platform is not just about scheduling resources and hosting applications. It needs to provide practical support to developers to complete the tasks at hand without jumping through hoops.

At a basic level, a PaaS platform helps developers deal with cross-cutting concerns such as logging, monitoring, and transaction processing. Taking it a step further, a PaaS platform provides advanced nonfunctional features such as service discovery, failover, replication, and load balancing. All these nonfunctional requirements are essential to a scalable and available system. And providing built-in constructs to satisfy these requirements leads to a significant productivity boost. Because PaaS takes care of all these troubles, developers can focus on building up core business logic. To achieve this, these nonfunctional features should be available without getting in the way. As you progress through this chapter and book, you’ll see how Service Fabric enables you to focus on business logic and to incorporate these features whenever you need them.

Can you go a step further? What if a PaaS platform provides easy programming models that help you tackle complex problems? And what if the PaaS platform also provides guidance and patterns for typical scenarios? We’ll come back to this in the discussion of different service types in Chapter 2, Chapter 3, and Chapter 4.

### Comprehensive application lifetime management

Continuous improvement is at the core of the agile software movement and the Lean movement in various industries. The faster you can iterate through revision cycles, the quicker you can innovate, reduce waste, and create additional value. A mature PaaS platform has to offer comprehensive application lifecycle management (ALM) functionalities to keep the innovation engine running without friction.
Because more companies are adopting continuous delivery, software is being released at a faster pace than in the past. Some companies claim they do hundreds of deployments on a daily basis. This calls for automated testing, continuous integration, rapid deployments, robust version management, and fast rollbacks. Only when a PaaS platform provides all these features can developers and independent software vendors (ISVs) realize such continuous delivery scenarios.

A comprehensive ALM strategy is critical to DevOps. If you look carefully, you’ll see that a lot of so-called friction between development and operations is rooted in discrepancies among different environments. PaaS platforms such as Service Fabric allow applications to be placed in self-contained packages that can be deployed consistently to different environments—such as development, test, QA, and production.

Part II of this book is dedicated to ALM.

**Designed for QoS**

A successful cloud service is based on a healthy partnership between the service developer and the cloud platform. The service developer brings business know-how and innovation, and the cloud platform brings Quality of Service (QoS) opportunities such as scalability, availability, and reliability.

**Scalability**

Through innovation, you can do unprecedented things. However, the increasing complexity of problems constantly challenges developers to improve methodologies to maintain momentum. A PaaS platform should be designed with scalability in mind so that applications can be scaled out naturally without much effort from the developers.

**Increasing complexity and scale**

Increasing complexity can be demonstrated easily with some examples. According to the “NASA Study on Flight Software Complexity” (NASA Office of Chief Engineer, 2009), flight software complexity has been increasing exponentially with a growth rate of a factor of 10 approximately every 10 years. Apollo 8 had about 8,500 lines of code in 1968. In contrast, the International Space Station (ISS) was launched with 1.5 million lines of code in 1989.

Besides software complexity, the sheer volume of data presents a new set of problems. According to Twitter statistics (Company Facts at [https://about.twitter.com/company](https://about.twitter.com/company)), Twitter is handling 500 million tweets every day. Data ingress, transformation, storage, and analysis at such a scale is an unprecedented challenge. Modern services also need to deal with the potential for rapid growth. Over the past five years or so, Azure Storage has grown into a service that needs to handle 777 trillion transactions per day (Charles Babcock, "Microsoft Azure: More Mature Cloud Platform," InformationWeek, Sept 30, 2015, [http://aka.ms/asf/maturecloud](http://aka.ms/asf/maturecloud)).
On a cloud platform, *scaling up*, which means increasing the processing power of a single host, is a less preferable approach. Typically, virtual machines are offered with preconfigured sizes. To scale up, you’ll need to migrate your workload to a virtual machine with a bigger size. This is a long and disruptive process because services need to be brought down, migrated, and relaunched on the new machine, causing service interruptions. Furthermore, because there are finite choices of machine sizes, scaling options run out quickly. Although Azure provides a large catalog of virtual machine sizes, including some of the largest virtual machines in the cloud, large-scale workloads still can exceed the processing power of a single machine.

In contrast, *scaling out* dynamically adjusts system capacity by adding more service instances to share the workload. This kind of scaling is not disruptive because it doesn’t need to shut down existing services. And theoretically, there’s no limit to how much you can scale because you can add as many instances as you need.

When scaling out, there are two fundamental ways to distribute workloads. One way is to distribute the workloads evenly across all available instances. The other way is to partition the workloads among service instances. Service Fabric supports both options, which we’ll discuss in detail in Chapter 7, “Scalability and performance.”

**Availability**

Availability commonly is achieved by redundancy—when a service fails, a backup service takes over to maintain business continuity. Although the idea sounds simple, difficulties can be found in the details. For example, when a service fails, what happens to its state that it has been maintaining locally? How do you ensure that the replacement service can restore the state and pick up wherever it left off? In a different case, when you apply updates, how do you perform a zero-downtime upgrade? And how do you safely roll back to previous versions if the new version turns out to be broken? The solution to these questions involves many parts such as health monitoring, fault detection, failover, version management, and state replication. Only a carefully designed PaaS can orchestrate these features into a complete and intuitive availability solution. Reliability and availability is the topic of Chapter 6, “Availability and reliability.”

**Reliability**

Reliability is compromised by system faults. However, in a large-scale, distributed system, monitoring, tracing, and diagnosing problems often are challenging. If a PaaS doesn’t have a robust health sub-system that can monitor, report, and react to possible system-level and application-level problems, detecting and fixing system defects becomes incredibly difficult.

We’ll examine what Service Fabric has to offer in terms of reliability in Chapter 6.

**Separation of workload and infrastructure**

The cloud era brings new opportunities and new challenges. One advantage of cloud infrastructure as a service (IaaS) is that it shields you from the complexity of physical or virtualized hardware management—and that’s only the starting point. To enjoy the benefits of the cloud fully, you need PaaS to
help you forget about infrastructure altogether. After all, for a program to run, all you need are some compute and storage resources such as CPU, memory, and disk space. Do you really need to control which host is providing these resources? Does it really matter if your program stays on the same host throughout its lifetime? Should it make a difference if the program is running on a local server or in the cloud? A modern PaaS such as Service Fabric provides a clear separation of workload and infrastructure. It automatically manages the pool of resources, and it finds and assigns resources required by your applications as needed.

**Placement constraints**

Sometimes, you do care how components in your application are laid out on a PaaS cluster. For example, if your cluster comprises multiple node types with different capacities, you might want to put certain components on specific nodes. In this case, your application can dictate where PaaS places different components by defining placement constraints. In addition, if you want to minimize the latency between two components that frequently interact with each other, you can suggest that PaaS keep them in close proximity. In some other cases, you might want to distribute the components far apart so that a failing host won’t bring down all the components. We’ll discuss placement constraints later in this book.

Such clear separation of concerns brings several significant benefits. First, it enables workloads to be transferred from host to host as needed. When a host fails, the workloads on the failing host can be migrated quickly to another healthy host, providing fast failovers. Second, it allows higher compute density because independent workloads can be packed into the same host without interfering with one another. Third, as launching and destroying application instances usually is much faster than booting up and shutting down machines, system capacity can be scaled dynamically to adapt to workload changes. Fourth, such separation also allows applications to be architected, developed, and operated without platform lock-in. You can run the same application on-premises or in the cloud, as long as these environments provide the same mechanism to schedule CPU, memory, and disk resources.

**Service Fabric concepts**

In this section, you first briefly review the architecture of Service Fabric. Then, you learn about some of the key concepts of Service Fabric in preparation for service development.

**Architecture**

An overview of Service Fabric architecture is shown in Figure 1-1. As you can see, Service Fabric is a comprehensive PaaS with quite a few subsystems in play. The discussion here gives you a high-level overview of these subsystems. We’ll go into details of each of the subsystems throughout this book, so don’t worry if you are not familiar with some of the terms.
The subsystems shown in Figure 1-1 are as follows:

- **Transport subsystem** The transport subsystem is a Service Fabric internal subsystem that provides secured point-to-point communication channels within a Service Fabric cluster and between a Service Fabric cluster and clients.

- **Federation subsystem** The federation subsystem provides failure detection, leader election, and consistent routing, which form the foundation of a unified cluster. We’ll examine these terms in upcoming chapters.

- **Reliable subsystem** The reliable subsystem manages state replication, failovers, and load balancing, which a highly available and reliable system needs.

- **Management subsystem** The management subsystem provides full application lifetime management, including services such as managing application binaries; deploying, updating and deprovisioning applications; and monitoring application health.

- **Hosting subsystem** The hosting subsystem is responsible for managing application life cycles on a cluster node.

- **Communication subsystem** The primary task of the communication subsystem is service discovery. With complete separation of workloads and infrastructure, service instances may migrate from host to host. The communication subsystem provides a naming service for clients to discover and connect to service instances.

- **Testability subsystem** The idea of test in production was popularized by the Netflix Chaos Monkey (and later the Netflix Simian Army). The testability subsystem can simulate various failure scenarios to help developers shake out design and implementation flaws in the system.
Nodes and clusters
To understand Service Fabric clusters, you need to know about two concepts: node and cluster.

- **Node**  Technically, a node is just a Service Fabric runtime process. In a typical Service Fabric deployment, there’s one node per machine. So you can understand a node as a machine (physical or virtual). A Service Fabric cluster allows heterogeneous node types with different capacities and configurations.

- **Cluster**  A cluster is a set of nodes that are connected to form a highly available and reliable environment for running applications and services. A Service Fabric cluster can have thousands of nodes.

Figure 1-2 is a simple illustration of a Service Fabric cluster. Notice that all nodes are equal peers; there are no master nodes or subordinate nodes. Also notice that although in the diagram the nodes are arranged in a ring, all the nodes can communicate directly with each other via the transport subsystem.

![A Service Fabric cluster](image)

**FIGURE 1-2**  A Service Fabric cluster

**Note**  Node and containers
In addition to physical machines and virtual machines, nodes can reside in Windows-based Docker containers, which are part of Windows Server 2016. Containerization is described in more detail in Chapter 17, “Advanced service hosting.”

A Service Fabric cluster provides an abstraction layer between your workloads and the underlying infrastructure. Because you can run Service Fabric clusters on both physical machines and virtual machines, either on-premises or in the cloud, you can run your Service Fabric applications without modifications in a variety of environments such as on-premises datacenters and Microsoft Azure.
Applications and services

A Service Fabric application is a collection of services. A service is a complete functional unit that delivers certain functionalities.

You author a Service Fabric application by defining the Application Type and associated Service Types. When the application is deployed to a Service Fabric cluster, these types are instantiated into application instances and service instances, respectively.

An application defines an isolation unit in Service Fabric. You can deploy and manage multiple applications independently on the same cluster. Service Fabric keeps their code, configuration, and data isolated from one another. You can deploy multiple versions of an application on the same cluster.

Partitions and replicas

A service can have one or more partitions. Service Fabric uses partitions as the scaling mechanism to distribute workloads to different service instances.

A partition can have one or more replicas. Service Fabric uses replicas as the availability mechanism. A partition has one primary replica and may have multiple secondary replicas. The states of replicas are synchronized automatically. When a primary replica fails, a secondary replica automatically is promoted to primary to keep service availability. And the number of secondary replicas is brought back to desired level to keep enough redundancy.

We’ll introduce partitions and replicas in more detail in Chapter 2, Chapter 3, and Chapter 7.

Programming modes

Service Fabric provides two high-level frameworks to build applications: the Reliable Service APIs and the Reliable Actor APIs.

- The Reliable Service APIs provide direct access to Service Fabric constructs such as reliable collections and communication stacks.
- The Reliable Actor APIs provide a high-level abstraction layer so that you can model your applications as a number of interacting actors.

With Reliable Service APIs, you can add either stateless services or stateful services to a Service Fabric application. The key difference between the two service types is whether service state is saved locally on the hosting node.

Stateless vs. stateful

Some services don’t need to maintain any states across requests. Let’s say there’s a calculator service that provides both an Add operation and a Subtract operation. For each of the service calls, the service takes in two operands and generates a result. The service doesn’t need to maintain any contextual information between calls because every call can be carried out based solely on given parameters. The
service behavior is not affected by any contextual information; that is, adding 5 and 3 always yields 8, and subtracting 6 from 9 always yields 3.

The majority of services, in contrast, need to keep some sort of states. A typical example of such a service is a shopping cart service. As a user adds items to the cart, the state of the cart needs to be maintained across different requests so that the user doesn’t lose what she has put in the cart.

Services that don’t need to maintain states or don’t save states locally are called stateless services. Services that keep local states are called stateful services. The only distinction between a stateful service and a stateless service is whether the state is saved locally. Continuing with the previous shopping cart example, the service can be implemented as a stateless service that saves shopping cart states in external data storage or as a stateful service that saves shopping cart states locally on the node.

Note  “Has state” and “stateful”

Most services have states. However, this doesn’t mean they are stateful. The only difference between stateful services and stateless services is where states are stored.

A stateful service can cause some problems. When a service is scaled out, multiple instances share the total workload. For a stateless service, requests can be distributed among the instances because it doesn’t matter which instance handles the specific request. For a stateful service, because each service instance records its own state locally, a user session needs to be routed to the same instance to ensure a consistent experience for the user. Another problem with a stateful service is reliability. When a service instance goes down, it takes all its state with it, which causes service interruptions for all the users who are being served by the instance.

To solve these problems, a stateful service can be transformed into a stateless service by externalizing the state. However, this means every service call will incur additional calls to an external data source, increasing system latency. Fortunately, Service Fabric provides a way to escape this dilemma, which we’ll discuss in Chapter 3.

**Getting started**

To get started with Service Fabric development, you need two things:

- A development environment
- A Service Fabric cluster

In this section, first you’ll set up a local development, which includes a local multinode cluster that allows you to deploy and test your applications. Then, you’ll provision a managed Service Fabric cluster on Microsoft Azure. This book primarily focuses on developments using C# in Visual Studio 2015. However, we’ll briefly cover developments using other languages such as Node.js.
Setting up a development environment

In addition, install the following tools:

- Latest version of Microsoft Azure SDK for .NET (using Web PI, this book uses 2.8.1)
- Latest version of Microsoft Azure PowerShell (using Web PI, this book uses 1.0)

Service Fabric SDK provides a local multinode Service Fabric cluster to which you can deploy and test your applications.

Provisioning a Service Fabric cluster on Azure
Although you can use the local cluster provided by Service Fabric SDK for local development and tests, you’ll want a hosted cluster on Azure for your production deployments.

Note  Microsoft Azure subscription
To use Microsoft Azure, you need a Microsoft Azure subscription. If you don’t have one, you can apply for a free one-month trial at https://azure.microsoft.com/pricing/free-trial/.

You can follow these steps to create a new Service Fabric cluster.

Microsoft Azure management portal terms
As you click links in the portal, the display areas that expand to the right are called blades. You also may see the following terms used in talks and articles:

- Hub  A hub gathers and displays information from multiple data sources. For instance, all notifications from different services are displayed in a centralized notification hub, which can be brought up by the Bell icon on the top command bar.

- Dashboard  The home page after you log in is called a dashboard, where you can pin various types of resources for quick access.

- Tile  Each item you pin on the dashboard is represented by a tile.

- Journey  As you go through a workflow, your navigation steps are recorded as a journey. You can see the history of your journey at the top of the page, and you can click any of the steps to track back or to jump ahead. Journeys are recorded automatically, and you can access previous journeys by clicking the down arrow icon beside the Microsoft Azure label, as shown in the following figure.
To provision a Service Fabric cluster, complete the following steps:


2. Click the New icon in the upper-left corner of the home page. Then, click Marketplace, as shown in Figure 1-3.

FIGURE 1-3 Create a new resource on Microsoft Azure
3. Under the Everything category, type **service fabric** in the Search box and press Enter. You’ll see a Service Fabric Cluster entry, as shown in Figure 1-4. Click the entry to create a new Service Fabric cluster.

![Figure 1-4 Service Fabric in Marketplace](image)

**FIGURE 1-4** Service Fabric in Marketplace

4. On the Service Fabric Cluster blade, click the Create button to continue, as shown in Figure 1-5.

![Figure 1-5 Service Fabric template blade](image)
5. On the Basics blade, enter a Cluster Name. Enter the user credentials for VM. Select the Azure Subscription you want to use, and type a name for the new Resource Group. Then, pick an Azure Location where you want the cluster to be hosted, and click OK to continue, as shown in Figure 1-6.

![Service Fabric Cluster creation blade](image)

**FIGURE 1-6** Service Fabric Cluster creation blade

**Note** Azure resource groups

A *resource group* is a collection of resources on Azure. On Azure, every resource, such as a virtual machine or a virtual network, belongs to a resource group. A resource group defines a management boundary and a security boundary. You can provision and deprovision all resources in a resource group as a logical unit. And you can apply group-level Role-Based Access Control (RBAC) policies, which are inherited by all members in the group.
6. Click Node Type and create a new node type configuration. (You’ll find more information about types of nodes later in this book.) In the Node Type Configuration blade, enter a name for the node type and pick a virtual machine size. Type 80 for the Custom Endpoints value, and then click OK, as shown in Figure 1-7.

![Service Fabric Cluster settings blade](image)

**FIGURE 1-7** Service Fabric Cluster settings blade

7. Change the Security mode to Unsecure, and follow the creation wizard to complete provisioning the cluster.

8. The provisioning process takes a few minutes. Once that is done, you’ll have a new tile on your dashboard to access the cluster. Figure 1-8 shows the cluster blade, on which you can find the cluster public address (in the format of <cluster name>.<region>.cloudapp.azure.com) and the port number (the default is 19000). You’ll need this information to connect to the cluster later.
Hello, World

This is the moment you’ve been waiting for—a chance to implement the beloved “Hello, World” in a new way.

A tribute to Hello, World

The following code was my first “Hello, World” program, which was written in BASIC about 27 years ago:

10 PRINT ”Hello, World"
20 END

What’s yours? I’m glad to see that such simplicity and elegance is being carried over throughout the years collectively by the community to get developers started with new languages and platforms. Of course, because Service Fabric is designed to tackle complex problems, you need to inherit some established frameworks and structures from the platform. However, as you’ll see in a moment, the “Hello, World” program still calls for only a couple lines of changes.
Now you are ready to create your first Service Fabric application. This application contains a stateless service that generates a time stamped “Hello World” string every five seconds.

1. Launch Visual Studio 2015 as an administrator.

   **Note** Launching Visual Studio as an administrator

   You need to launch Visual Studio as an administrator in this case because you are going to test the application with a local test cluster, which needs administrative rights to be launched.

2. Create a new project named HelloWorldApplication using the Cloud\Service Fabric Application template, as shown in Figure 1-9.

   ![New Project dialog box](image)

   **FIGURE 1-9** New Project dialog box

3. In the New Service Fabric Service dialog box, select the Stateless Service template, enter **HelloWorldService** as the Service Project Name, and then click OK to create the Hello World service, as shown in Figure 1-10.
Now, in the solution you have two projects: A HelloWorldApplication project for the Service Fabric application and a HelloWorldService project for the stateless service. You’ll go through the rest of the solutions in a moment. For now, focus on the `HelloWorldService` class in the service project:

```csharp
internal sealed class HelloWorldService : StatelessService
{
    public HelloWorldService(StatelessServiceContext context)
        : base(context)
    {
    }
    protected override IEnumerable<ServiceInstanceListener>
        CreateServiceInstanceListeners()
    {
        return new ServiceInstanceListener[0];
    }
    protected override async Task RunAsync(CancellationToken cancellationToken)
    {
        long iterations = 0;
        while (true)
        {
            cancellationToken.ThrowIfCancellationRequested();
            ServiceEventSource.Current.ServiceMessage(this, "Working-{0}", ++iterations);
            await Task.Delay(TimeSpan.FromSeconds(1), cancellationToken);
        }
    }
}
```

**FIGURE 1-10** New Service Fabric Service dialog box
4. To implement a stateless service, your service class needs to inherit from the *StatelessService* base class. Service Fabric doesn’t mandate a communication protocol. Instead, you can plug in different communication stacks by providing an ICommunicationListener implementation. You’ll see a number of communication stack implementations throughout this book. For this example, you’ll skip the communication stack, which means your service is a background service that doesn’t take any client requests.

5. Modify the *RunAsync* method to implement your service logic.

```csharp
protected override async Task RunAsync(CancellationToken cancellationToken)
{
    while (!cancellationToken.IsCancellationRequested)
    {
        ServiceEventSource.Current.ServiceMessage(this, "Hello World at " +
            DateTime.Now.ToLongTimeString());
        await Task.Delay(TimeSpan.FromSeconds(5), cancellationToken);
    }
}
```

As you can see in this code snippet, to implement a background service, all you need to do is override the *RunAsync* method and construct your processing loop.

**Note** Cancellation token

In the .NET asynchronous programming pattern, the CancellationToken structure is used to propagate notification that operations should be canceled. Because a cancellation token can be passed to multiple threads, thread pool work items, and Task objects, it can be used to coordinate cancellation across these boundaries.

When the token’s IsCancellationRequested property is set to True, the owning object has requested cancellation of associated tasks. You should stop processing when receiving the notification. If you are calling downstream tasks in your code, you should pass the cancellation token along so that the downstream tasks can be cancelled, like the Task.Delay call in the previous code sample.

6. Service Fabric SDK automatically generates an Event Tracing for Windows (ETW) event source implementation for you to generate trace events. Examine the generated *ServiceEventSource* class if you are interested.
Note Use ETW for tracing and logging

It’s recommended to use ETW for tracing and logging because ETW is fast and has minimum impact on your code performance. Furthermore, because Service Fabric also uses ETW for internal tracing, you can view your application logs interleaved with Service Fabric traces, making it easier to understand the relationships between your applications and Service Fabric. Last but not least, because ETW tracing works both in local environment and in the cloud, you can use the same tracing mechanism across different environments.

7. Now, you can press F5 to run the application. Visual Studio will launch a test cluster, deploy the application, and start the services. Once the service is launched, you can see the “Hello World” output in the Diagnostic Events window, as shown in Figure 1-11.

![Diagnostic Events window](image)

FIGURE 1-11 Diagnostic Events window

Note Bring up the Diagnostic Events view

If you don’t see the Diagnostic Events view, you can bring it up by using the View\ Other Windows\Diagnostic Event Viewer menu.

8. Click Stop Debugging on the Visual Studio toolbar or press Shift+F5 to stop the debugging session. If you bring back the Diagnostic Event view, you can see the “Hello World” strings are still coming in. This is because the Stop Debugging button only stops your debugging session; it doesn’t stop the service.

Congratulations! You’ve implemented, deployed, and tested your first Service Fabric application. Next, you’ll take a peek into the local cluster.
Managing your local cluster

There are multiple ways to manage your local cluster. You can use Visual Studio Server Explorer, Cloud Explorer, Service Fabric Local Cluster Manager, or PowerShell. You’ll walk through all the options in this section.

Visual Studio Server Explorer

You can bring up Server Explorer from the Visual Studio View\Server Explorer menu. Under the Azure node, you’ll see a Service Fabric Cluster (Local) node, along with other Azure resource types if you have Azure SDK installed. Figure 1-12 shows that on my cluster I have five nodes, and I have a single Hello World application deployed. You should note that applications and cluster nodes are presented separately, showing the clear separation between workloads and cluster resources that was introduced earlier in this chapter.

![Server Explorer](image)

**FIGURE 1-12** Server Explorer

The Nodes view in Figure 1-12 seems straightforward—it shows a cluster that consists of five nodes. The application view, however, needs some explanation. What do those levels of nodes mean? Don’t worry, you’ll go through each of them next.
Application Type node
At the top level, there is an Application Type node that represents an application type, which in this case is HelloWorldApplicationType. When you write your application code in Visual Studio, you define an application type. When the application is deployed, you get an application instance on the cluster. This relationship is similar to the relationship between a class and an object in Object Oriented Programming (OOP).

Application Instance node
Below the Application Type node is the Application Instance node, which is identified by the application's name (fabric:/HelloWorldApplication in this case). By default, an application is named after the corresponding application type name. But you can change to any other name in the format of fabric:/<string>; for example, fabric:/MyApplication.

Service Type node
Under the Application Instance node, you have Service Type nodes. Each node represents a registered service type, such as the HelloServiceType in this example. Each service in a Service Fabric application is named as fabric:<application name>/<service name>. This name also is the address of the service. Service Fabric built-in naming service resolves this name to the actual service instance address at run time.

Partition node
A partition is identified by a GUID, suggesting it's not something that a client would use to address a partition directly. Instead, the Service Fabric naming service will figure out the correct partition to which a service request should be sent.

Replica node
Figure 1-12 shows that by default, Service Fabric maintains multiple replicas for a partition for high availability. In this case, the Hello World service has a single partition with five replicas.

Note  Replica and service instance
A replica is a service instance. The term service instance often is used in logical descriptions of deployment topologies, and the term replica often is used when explicit replica behaviors or concepts are being discussed.

Visual Studio Cloud Explorer
Microsoft Azure SDK comes with a Visual Studio extension named Cloud Explorer, which you can access by the View\Cloud Explorer menu. Cloud Explorer is similar to Server Explorer in terms of Service Fabric cluster management functionalities, as shown in Figure 1-13.
Neither Server Explorer nor Cloud Explorer is designed as a full-fledged management tool. They are designed for you to navigate and view your cloud resources and server resources easily and with limited management functionality.

Service Fabric SDK provides a powerful tool named Service Fabric Explorer. To bring up the management UI, you can use any browser and navigate to http://localhost:19080/Explorer.

The left panel of the Service Fabric Explorer looks much like Server Explorer or Cloud Explorer. However, the tool also has a details panel to the right providing very detailed information on the currently selected item, as shown in Figure 1-14.
The management tool is loaded with features. You’ll use this tool frequently throughout this book for different scenarios. For now, perform a little exercise to familiarize yourself with the tool. In this exercise, you’ll delete the Hello World application from the cluster.

1. In the Service Fabric Explorer, select the fabric:/HelloWorldApplication node in the left pane. Then, in the right pane, click the Actions button and then click the Delete Application menu, as shown in Figure 1-15.

2. In the Confirm Application Deletion dialog box, shown in Figure 1-16, click the Delete Application button to continue.
3. The UI refreshes itself every few seconds. Once the UI is refreshed, you can see the application instance has been removed, as shown in Figure 1-17.

![Confirm Application Deletion dialog box](image1)

**FIGURE 1-16** Confirm Application Deletion dialog box

![Two application instances](image2)

**FIGURE 1-17** Two application instances

**Note** Resetting the cluster

Service Fabric SDK installs a Service Fabric icon on your taskbar, which provides a couple of shortcut menus for some cluster-level operations. This comes in handy when you want to reset everything and make a fresh start. To reset the cluster, right-click the Service Fabric icon on the taskbar and select the Reset Cluster menu.
Windows PowerShell

Windows PowerShell is a powerful automation and configuration management framework. Service Fabric SDK installs a number of PowerShell cmdlets for you to manage your applications and clusters using command lines or automation scripts.

If you look under the Scripts folder of the HelloWorldApplication, you can find a Deploy-FabricApplication.ps1, which in turn calls a number of prebuilt scripts under your Service Fabric SDK folder (C:\Program Files\Microsoft SDKs\Service Fabric\Tools\Scripts by default) to deploy, upgrade, and remove your applications.

To get started, open a new Windows PowerShell window and use the Connect-ServiceFabricCluster cmdlet to connect to the local cluster, as shown in Figure 1-18.

![Figure 1-18 Connect-ServiceFabricCluster cmdlet](image)

Next, try a few commands. You’ll be introduced to more cmdlets as you go through the chapters.

- List application instances.

To list application instances on the cluster, type the following cmdlet:

```powershell
Get-ServiceFabricApplication
```

The above cmdlet generates the following output on my environment (before the application instance is removed. To restore the instance, press F5 in Visual Studio to redeploy application):

```plaintext
ApplicationName : fabric:/HelloWorldApplication
ApplicationTypeName : HelloWorldApplicationType
ApplicationTypeVersion : 1.0.0.0

ApplicationStatus : Ready
HealthState : Ok
ApplicationParameters : { "HelloWorldService_InstanceCount" = "-1" }
```
List cluster node names and statuses.

To list all the nodes and their current statuses, use the following cmdlet:

```
Get-ServiceFabricNode | Format-Table NodeName, NodeStatus
```

A healthy cluster would look like this:

<table>
<thead>
<tr>
<th>NodeName</th>
<th>NodeStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node.1</td>
<td>Up</td>
</tr>
<tr>
<td>Node.2</td>
<td>Up</td>
</tr>
<tr>
<td>Node.3</td>
<td>Up</td>
</tr>
<tr>
<td>Node.4</td>
<td>Up</td>
</tr>
<tr>
<td>Node.5</td>
<td>Up</td>
</tr>
</tbody>
</table>

**Additional information**

At the time of this writing, Service Fabric is still in preview. It's likely that service APIs, tooling experiences, and management UIs will change over time. However, key concepts covered in Part I and Part II of this book and the patterns and applicable scenarios in Part III and Part IV of this book should remain valid in future releases.

For up-to-date Service Fabric documentations, please visit [https://azure.microsoft.com/documentation/services/service-fabric/](https://azure.microsoft.com/documentation/services/service-fabric/).
Index

A

Abort() method, 44
access control lists (ACLs), 113
ACLs (access control lists), 113
Active Secondary replica role, 65
Actor programming model
about, 69, 257
actor polymorphism, 442
Aggregation pattern, 160–162
communication considerations, 71, 159
diagnostics and performance monitoring, 82–89
events, 81–82
modeling complex systems, 69–70
processing topologies with actors pattern, 314–317, 441
Reliable Actors API, 70–72
Reliable Services API, 89–90
reminders, 80–81
responsive website scenario, 318–320
state providers, 90–94, 156–157
termite model, 400–402
tic-tac-toe game, 72–79, 230–231
timers, 79–80
actor proxy, 71, 82, 229
Actor Swarm, 407–408, 414–416
ActorMethodCallsWaitingForLock event, 83
ActorMethodStart event, 83
ActorMethodStop event, 83
ActorMethodThrewException event, 83
ActorProxy class, 71, 82
ActorSaveStateStart event, 83
ActorSaveStateStop event, 83
adaptive complex systems
about, 393
complicated systems and, 394
computational modeling, 396–397
emergence, 394–395
simple model, 396
Add-AzureKeyVaultKey cmdlet, 169
Add Outbound Security Rule dialog box, 201
administrators, launching Visual Studio, 18
Administrators user group, 113, 116
Advanced Message Queue Protocol (AMQP), 277
affinity (service), 124–125
agents. See Actor programming model
Aggregation pattern
about, 159–160, 281–286, 443
cross-tenant, 332–333, 442
global view and, 362–363
online voting system, 160–162
agile software movement, 4
agility
application lifetime management, 4–5
design considerations, 3
Microservices, 3–4
simplicity, 4
A.I. Quests (game)
about, 358
Aggregation pattern, 362
Coverage Map pattern, 366–367, 442
describing game world, 358–360
dynamic clustering, 367
Dynamic Game Room pattern, 363–366, 442
global map, 362–363
Partitioned world pattern, 360–361
player interactions, 363–367
spatial interpolation, 366
spatial model, 360
allow rules, 199–202
ALM (application lifecycle management), 4–5, 371–372, 376
Amazon Glacier cold storage, 278
Amazon Web Services (AWS), 371, 389–391
AMQP (Advanced Message Queue Protocol), 277
Apache ActiveMQ message broker, 277
Apache HBase database, 278
Apache Kafka message broker, 277
API Management service, 268
App Services
  about, 253–256
  choosing, 258–259
application deployment process
  about, 95
  creating/replacing/upgrading, 100
  implicit hosts, 111–116
  multiple environments, 110–111
  packaging, 95–99
  PowerShell commands, 184–185
  provisioning, 100
  registering, 100
  uploading, 99–100
application galleries, 375
Application Insights (Visual Studio)
  about, 84, 259
  alert rules, 223–225
  creating instances, 221–222, 254
  sending events, 222–223
Application Instance node, 23
application lifecycle management (ALM), 4–5, 371–372, 376
Application Manager, 142
application manifest files
  about, 37–39, 96
  diff packages, 131
  modifying replica size, 88
  multiple environment deployment, 110
  service affinity, 124
  upgrade process, 105, 185
application packages
  deployment process, 95–99, 142
  format considerations, 372–374
  rolling upgrades, 129–131
  service manifests, 37–38
  as services, 378–379
application publish profiles, 110–111
Application Type node, 23
application view (Server Explorer), 22–23
ApplicationParameters folder, 110
applications (Service Fabric). See also web applications
  about, 10
  ASP.NET 5, 29–31
  containerized, 384
  decommissioning, 187
  health model, 101
  hosting guest applications, 375–383
  launching, 18–21
  managing, 184–187
  model overview, 36–38
  PowerShell commands, 184–187
  programming modes, 10
  publishing from Visual Studio, 172
  query commands, 177–178
  rolling back, 186
  sample health policy, 103
  Simple Store example, 56–65
  tic-tac-toe game, 72–79, 230–231
  tuning, 88
  upgrading, 100, 104–110, 185–186
architecture
  OWIN, 44
  pattern index, 444
Service Fabric, 7–8
Arguments element, 98
ARM templates
  about, 150
  authoring, 150, 428–429
  defining custom event source, 204–207
  deploying clusters using, 430–431
ASA (Azure Stream Analytics)
  about, 298, 304
  defining jobs, 305
  query language, 305
  Service Fabric integration, 305–306
ASP.NET Web API, 29–31, 43–44
aura-nimbus model, 361
authentication
  claim-based, 269–271, 444
  client, 171–172
automated build process, 131
automated players (bots), 355–356
automated test cases, 228
autoscaling rules, 148–153, 256
availability (service). See also reliability (service)
  advanced rolling upgrades, 128–131
  bottlenecks and, 154–155
  broken service, 117–118
  CAP theorem, 53
design considerations, 6
Failover Manager, 36, 120, 421
guest applications, 376
improving, 118
redundancy and, 6, 10, 118–119
Resource Balancer, 120–123, 421
routing and load balancing, 128
service failovers, 119, 127–128
service placements, 119–126
stateful services, 146
stateless services, 32, 146
availability sets, 190–191
Average milliseconds per invocation performance
counter, 84
Average Milliseconds Per Save State Operation
performance counter, 85–88, 93–94
AWS (Amazon Web Services), 371, 389–391
Azure AD (Active Directory), 371, 389–391
Azure Backup, 278, 329
Azure Billing API, 327–329
Azure Blob service, 278, 307
Azure Cache, 307
Azure CLI, 435
Azure Cloud Services
  about, 34, 256
  choosing as platform, 258–259
Azure Container Service, 253, 387, 439
Azure Content Delivery Network service, 153–154, 259
Azure Data Lake database, 278, 307, 329
Azure Diagnostics
  about, 133–136, 203
capturing events, 229
  configuring, 203–208
Azure DocumentDB database, 307
Azure Event Hubs, 286, 309–312
Azure Functions, 306, 408
Azure Insights Autoscale, 149, 151–153
Azure IoT Hub
  about, 277, 279
  Azure Stream Analytics service, 298
event ingress, 286–288
  registering a device, 288–289
  saving sensor data, 293
Azure Key Vault service, 168, 259
Azure Marketplace, 258
Azure Operational Insights, 216
Azure Operations Management Suite
  about, 216
  linking to data source, 217–219
  provisioning workspace, 216–217
Azure PowerShell
  about, 187
  application management commands, 184–187
  cluster management commands, 172–183
  creating secured clusters, 167–172
discovering commands, 427
installing, 425
  invoking testability actions, 248–250
  managing resource groups, 427–431
  signing in, 167, 425–426
Azure Quickstart Templates, 150, 428–430, 439
Azure RateCard API, 328–329
Azure Resource Explorer, 206–207
Azure Resource Manager API, 151, 427
Azure Service Bus Queue, 160
Azure Service Fabric
  about, 3, 28, 257
  applications and services, 10
  architectural overview, 7–8
  ASA integration, 305–306
  choosing as platform, 258–259
  for complex systems, 404–408
  containers and, 385–388
  Hello, World, 17–21
  managing local clusters, 22–28
  multitenancy support, 331–338
  nodes and clusters, 9
  partitions and replicas, 10
  programming modes, 10
  provisioning clusters, 12–17
  setting up development environment, 11–12
  stateless vs. stateful, 10–11
Azure Service Management API, 142, 151, 427
Azure SQL Database, 306
Azure Stack, 389–390
Azure Storage, 5, 259, 310, 329
Azure Stream Analytics (ASA)
  about, 298, 304
  defining jobs, 305
  query language, 305
  Service Fabric integration, 305–306
Azure Table Storage, 278, 307
Azure Traffic Manager, 259, 357
Azure Usage API, 328–329
Azure Virtual Machine Scale Sets
  autoscaling clusters, 149–153
  managing clusters, 191–192, 374–375, 387
  storage accounts, 197
azure vm docker command, 435
azure vm image list command, 435
Azure Websites service, 253–254

B
BackupAsync() method, 142
backups, server state, 142–143
BatchAcknowledgementInterval configuration setting, 92, 94
batched data analytics, 277
batching pattern, 160, 443
BDD (Behavior-Driven Development) process, 260
Behavior-Driven Development (BDD) process, 260
BI (Business Intelligence), 268
big data
  composable data processing units pattern, 308–313, 441
  processing topologies with actors pattern, 314–317, 441
  storage considerations, 278, 306–307
  value, 304
  variety, 304
  velocity, 304
  veracity, 304
  volume, 303
blades (Microsoft Azure management portal), 12, 431
bots (automated players), 355–356
bottlenecks
  about, 154–155
  communication, 159–160
  orchestration, 160–164
  state, 155–159
  broken service, 117–118
  build machine
    creating build definition, 239–242
    preparing, 237–239
  Business Intelligence (BI), 268

C
C# language, chaos tests, 138–141
CA (certificate authority), 167
calculator service examples
  custom communication stack, 43–50
  default communication stack, 33–41
  WCF communication stack, 41–43
Call of Duty (game), 363
CancellationToken structure, 20
CAP theorem, 53
capacity planning, 148
CDN (Content Delivery Network) service, 153–154, 259, 268
CEP (Complex Event Processing), 278
certificate authority (CA), 167
certificates
  client authentication, 171–172
  protecting clusters, 167–171
  X509, 91
certmgr tool, 169
Chacko, Daniel, 187
Chaos Monkey (Netfix), 137
chaos swarm, 408
chaos tests
  about, 137, 247
  graceful and ungraceful faults, 141–142
  performing, 138–141
  performing failover tests, 141
  testing in production, 137–138
checkpoints (snapshots), 142
chessboard layouts
  designing, 346
  handling updates, 348–350
Chocolatey package manager, 114
CI (continuous integration)
  about, 232
  automated build process, 131
automated test cases, 228
creating build definition, 239–242
preparing build machine, 237–239
preparing VSTS project, 232–236
running load tests, 244–246
running testing upon code check-ins, 242–244
setting up, 232–246
claim-based authentication pattern, 269–271, 444
clarity (testing), 230
CleanTestState testability action, 247
CleanTestStateAsync API, 247
client authentication, 171–172
CloseAsync() method, 44
Cloud Explorer (Visual Studio), 23–24
cloud gateways, 277
Cloud Services (Azure)
   about, 34, 256
   choosing as platform, 258–259
Cluster Manager (system cluster manager service), 36, 142, 194–195, 422
cluster manifest
   defining health policy, 102
defragmentation metrics, 126
   image store location, 99–100
   validating, 183
   viewing contents, 173
cluster nodes
   about, 9
   containers and, 9
fault domains and update domains, 105
health model, 101
node operation commands, 181–183
placement constraints, 122–123
query commands, 172–176, 180–181
restarting nodes, 138–142
sample health policy, 102–103
scaling process, 67–68
clusters
   about, 9, 202
   administrative rights, 18
   advanced configuration, 197–202
   anatomy of, 189–197
   Cluster Manager, 36, 142, 194–195, 422
   creating, 195
   creating secured, 167–172
   customizing, 192
   deploying, 389, 430–431
deploying stand-alone, 389
dynamic clustering, 367
enabling Azure Diagnostics on, 204–205
health model, 101
heterogeneous, 146, 257
homogeneous, 146
local, 12–18, 22–28
managing with Azure VM Scale Sets, 191–192, 374–375
managing with Service Fabric Explorer, 167, 171, 194, 220–221, 389
protecting with certificates, 167–171
provisioning, 12–17, 122
query commands, 172–176
resetting, 26, 62
resource orchestration and, 374–375
sample health policy, 102–103
scaling, 67–68, 148–154
storage accounts, 196–197
code folder, 96
code packages
   deploying, 96–98, 142
   rolling upgrades, 129
   service manifests, 37–38
CodecPackage element, 37–38, 97
Command and Control scenario, 299–300
Command pattern, 356
CommitAsync() method, 58
communication patterns, 71, 159–160
communication stacks
   custom, 43–50
default, 33–41
   flexible, 257
   WCF, 41–43, 56–59
communication subsystem
   about, 8, 423
   Naming Service, 36, 128, 195, 423
competing consumers pattern, 162–163, 272, 441
Complex Event Processing (CEP), 278
complex systems
   adaptive, 393–397
   complicated systems and, 394
   computational modeling, 396–397
   emergence, 394–395
   future works, 418
complicated systems

- modeling, 69–70, 393
- Service Fabric support, 404–408
- simple model, 396
- spatial segregation model, 409–418
- termite model, 397–404
- complicated systems, 394
- composable data processing units pattern
  - about, 308–309, 441
  - creating converter service, 310–312
  - creating test client, 312–313
  - provisioning event hubs, 309–310
  - provisioning storage account, 310
  - testing solution, 313
- computational modeling, 396–397
- concurrency mode (actors), 71–72
- ConfigPackage element, 38
- configuration folder, 96
- configuration packages
  - deployment process, 98–99
  - rolling upgrades, 129
  - service manifests, 38
- Confirm Application Deletion dialog box, 25–26
- Connect-ServiceFabricCluster cmdlet, 27, 170–171, 248
- ConsiderWarningAsError flag, 103, 109
- consistency (CAP theorem), 53, 55–56
- Container Service (Azure), 253, 387, 439
- ContainerHost element, 385
- containerized applications, 384
- containers
  - guest, 385–386
  - history of, 383–385
  - nodes and, 9
  - orchestrating, 384, 438–439
  - Service Fabric support, 385
  - types supported, 388
  - Windows, 439–440
- Content Delivery Network (CDN) service, 153–154, 259, 268
- continuous integration (CI)
  - about, 232
  - automated build process, 131
  - automated test cases, 228
  - creating build definition, 239–242
  - preparing build machine, 237–239
  - preparing VSTS project, 232–236
  - running load tests, 244–246
  - running testing upon code check-ins, 242–244
  - setting up, 232–246
- Contributor role, 197
- controllability (testing), 228, 246
- ConvertTo-SecureString cmdlet, 168
- coordination pattern
  - about, 160, 162
  - competing consumers, 162–163
  - distributed computing, 163–164
  - self-driven workflows, 162
- Copy-ServiceFabricApplicationPackage cmdlet, 184–185
- Count metric, 120
- coupling pattern, 159–160, 444
- Coverage Map pattern, 366–367, 442
- CPU usage performance counter, 152
- Create Agent Pool dialog box, 236
- Create Network Security Group dialog box, 200
- Create New Build Definition dialog box, 239
- Create New Team Project dialog box, 233
- CreateReliableStateManager() method, 143, 159
- creating application instances, 100
- CRM (Customer Relation Management), 268, 273
- cross-queries on cluster nodes, 180–181
- cross-tenant aggregation pattern, 332–333, 442
- CRUD operations, 427
- custom communication stack, 43–50
- Customer Relation Management (CRM), 268, 273

D

dashboard (Microsoft Azure management portal), 12

data folder, 96

data manipulation
  - data collection, 276–277
  - data generation and feedback, 276–277
  - data ingress, 277, 286–288
  - data presentation and actions, 278–279
  - data storage, 278, 290–292, 310
  - data transformation and analysis, 277–278
  - end-to-end scenario, 286–299

data packages
  - deployment process, 99
  - rolling upgrades, 129
  - service manifests, 38
DataContractSerializer class, 71, 157–158
DataPackage element, 38, 99
deadlocks, 156
decommissioning applications, 187
dedicated logs, 55
dedicated resources, 326
default communication stack, 33–41
DefaultRunAsPolicy element, 113
DefaultServices element, 38, 121–122, 124
_DefaultServiceTypeHealthPolicy upgrade parameter, 109
defragmentation (service), 125–126
Deis container orchestration, 439
deny rules, 199–202
dependency injection technique, 229
dependent variables, 229
DeployedApplication entity, 102
DeployedServicePackage entity, 102
deployment process
about, 95
on Amazon AWS, 390
on Azure Stack, 389–390
creating/replacing/upgrading, 100
implicit hosts, 111–116
multiple environments, 110–111
packaging, 95–99, 142
provisioning, 100
registering, 100
stand-alone clusters, 389
stand-alone package, 390–391
uploading, 99–100
development environment
improving reliability, 118
setting up, 11–12
device orchestration, 300
DHTs (distributed hash tables), 423
Diagnostic Events Viewer
chaos test example, 140
EventSource events, 84
Hello, World output, 21
service replica migration, 127
simulated device messages, 296
viewing diagnostic events, 129–131, 382
viewing logged file content, 99
Diagnostics (Azure)
about, 133–136, 203
capturing events, 229
configuring, 203–208
diagnostics and performance monitoring
about, 219–220, 225
Actor pattern, 82–89
Azure Diagnostics, 133–136, 203–208, 229
Azure Operations Management Suite, 216–219
broken service and, 117–118
communication bottlenecks, 159–160
data presentation and actions, 278–279
Elasticearch engine, 208–215
game performance, 347
health monitoring, 376, 380–383
Kibana platform, 208, 212–213
monitoring sensor data, 298–299
orchestration bottlenecks, 160–164
pattern index, 443
performance counters, 84–85, 93–94
Service Fabric Explorer, 220–221
state bottlenecks, 155–159
Visual Studio Application Insights, 84, 221–225, 254
Windows Performance Monitor, 85–88, 94
diff packages, 131
Disable-ServiceFabricNode cmdlet, 181–182
DiskInMb metric, 121
distributed computing patterns, 163–164, 441
distributed data structures, 405–406
distributed file systems, 278
distributed hash tables (DHTs), 423
dizziness test, 227
Docker Containers, 388
Docker engine
Azure CLI, 435
Docker Machine tool, 435–436
Docker VM extension, 434–435
pre-built image, 433–434
Docker Hub, 375, 385, 437
Docker Machine tool, 435–436
Docker Swarm container orchestration, 387, 439
Docker Trusted Registry, 437–438
DocumentDB database, 259, 278, 307, 329
dynamic clustering, 367
Dynamic Game Room pattern, 363–366, 442
e-commerce website scenarios

E

e-commerce website scenarios
about, 260
live data stream processing, 317–323
Mock-based Service Design pattern, 260–261, 444
persistent shopping cart pattern, 263–264, 443
Personalization Actor pattern, 261–263, 444
Edit Upgrade Settings dialog box, 106–107
Elastic Database feature, 306
Elasticsearch engine, 208–215
Enable-ServiceFabricNode cmdlet, 182
EnabledForDeployment flag, 168
encryption, state serialization and, 157
end-to-end scenario
about, 286
defining device data storage, 290–292
event ingress, 286–288
monitoring sensor data, 298–299
registering a device, 288–289
saving sensor data, 292–297
simulating device data, 289–290
End-User License Agreement (EULA), 327
EndpointBindingPolicy element, 113–114
endpoints
ACLs and, 113
adding to profiles, 153–154
binding listeners to, 321
configuring, 47–48
HTTP, 113, 115
load balancing rules, 195–196
registering multiple, 357
replicator, 92
Tenant Manager, 331
Enter-PSSession cmdlet, 440
enterprise portal scenarios
about, 268–269
claim-based authentication pattern, 269–271, 444
Finite State Machine Actor pattern, 272–274, 443
message-based integration pattern, 271–272, 441
Enterprise Resource Planning (ERP), 268, 273–274
Enterprise Service Bus (ESB), 272
ERP (Enterprise Resource Planning), 268, 273–274
Error health state, 102
ESB (Enterprise Service Bus), 272
ESP (Event Stream Processing), 278
ETW (Event Tracing for Windows), 20–21, 132
EULA (End-User License Agreement), 327
Event Hubs (Azure), 309–312
Event Stream Processing (ESP), 278
Event Tracing for Windows (ETW), 20–21, 132
EventProcessorHost class, 310
events (Actor pattern), 81–82
EventSource attribute, 82–84, 132, 205–208
Excel (Microsoft), 148, 279
Exceptions thrown/Sec performance counter, 84

F

FabricBackupInProgressException exception, 142
FabricClient class, 142
FABRIC_PARTITION_SCHEME_SINGLETON partition scheme, 125
Failover Manager, 36, 120, 421
failover mechanisms
about, 127–128
allocating node buffers, 126
cloud gateways, 277
container isolation, 384
tenant bus, 272
Failover Manager, 36, 120, 421
performing tests, 141
relocating replicas, 67
service availability and, 119
shared nodes, 326
stateless services, 41
Tenant Manager, 331
FailureAction upgrade parameter, 108
Fan-out indexes pattern, 265–266, 443
fault domains, 104–105, 190–191
federation subsystem, 8, 423
field gateways, 277, 299
file links, adding, 35
finite state machine (FSM), 273–274
Finite State Machine Actor pattern, 272–274, 443
first-person shooter (FPS) games, 363
Flyweight pattern, 408
ForceRestart upgrade parameter, 109–110
FPS (first-person shooter) games, 363
FSM (finite state machine), 273–274
game board
  Aggregation pattern, 362
  describing world, 358–360
designing, 346
global map, 362–363
  handling updates, 348–350
Partitioned world pattern, 360–361
Game Piece Dispenser pattern, 353–355, 442
game pieces
  about, 352
  actor polymorphism, 353
  Game Piece Dispenser pattern, 353–355, 442
games and simulation
  A.I. Quests, 358–367
  Messy Chess, 345–357
  pattern index, 442–443
  simulating device data, 289–290
  tic-tac-toe game, 72–79, 230–246
garbage collection, 70, 80
Gateway service, 128
Get-AzureRm* cmdlets, 427
Get-AzureRmOperationalInsightsWorkspace cmdlet, 218
Get-AzureRmResourceGroup cmdlet, 428
Get-AzureRmStorageAccount cmdlet, 218
Get-AzureRmStorageAccountKey cmdlet, 218
Get-AzureRmSubscription cmdlet, 426
Get-AzureRmVMExtension cmdlet, 434–435
Get-AzureRmVMExtensionImage cmdlet, 434
Get-ContainerImage cmdlet, 440
Get-Help cmdlet, 427
Get-ServiceFabricApplication cmdlet, 27, 177, 186
Get-ServiceFabricApplicationHealth cmdlet, 178, 185
Get-ServiceFabricApplicationManifest cmdlet, 177
Get-ServiceFabricApplicationType cmdlet, 177, 184–185
Get-ServiceFabricApplicationUpgrade cmdlet, 178, 186
Get-ServiceFabricClusterConnection cmdlet, 172
Get-ServiceFabricClusterHealth cmdlet, 173–174
Get-ServiceFabricClusterLoadInformation cmdlet, 175
Get-ServiceFabricClusterManifest cmdlet, 173
Get-ServiceFabricClusterUpgrade cmdlet, 175
Get-ServiceFabricDeployedApplication cmdlet, 180–181
Get-ServiceFabricDeployedApplicationHealth cmdlet, 181
Get-ServiceFabricDeployedCodePackage cmdlet, 181
Get-ServiceFabricDeployedReplica cmdlet, 181
Get-ServiceFabricDeployedReplica Detail cmdlet, 181
Get-ServiceFabricDeployedService
  PackageHealth cmdlet, 181
  Type cmdlet, 181
Get-ServiceFabricDeployedServicePackage cmdlet, 181
Get-ServiceFabricNode cmdlet, 28, 176, 182, 248–249
Get-ServiceFabricNodeConfiguration cmdlet, 176
Get-ServiceFabricNodeHealth cmdlet, 176–177
Get-ServiceFabricNodeLoadInformation cmdlet, 177
Get-ServiceFabricPartition cmdlet, 180, 249
Get-ServiceFabricPartitionHealth cmdlet, 180
Get-ServiceFabricPartitionLoadInformation cmdlet, 180
Get-ServiceFabricRegisteredClusterConfigurationVersion cmdlet, 176
Get-ServiceFabricReigsteredClusterCodeVersion cmdlet, 176
Get-ServiceFabricReplica cmdlet, 180, 249–250
Get-ServiceFabricReplicaHealth cmdlet, 180
Get-ServiceFabricReplicaLoadInformation cmdlet, 180
Get-ServiceFabricService cmdlet, 179
Get-ServiceFabricServiceDescription cmdlet, 179
Get-ServiceFabricServiceHealth cmdlet, 179
Get-ServiceFabricServiceManifest cmdlet, 179
Get-ServiceFabricServiceType cmdlet, 179
GetConfigurationPackageObject() method, 98–99
GetDataPackageObject() method, 99
GetEvent() method, 82
GetReminder() method, 81
gossiping agents, 407
graceful faults, 141–142
guest applications
  about, 375–376
  application lifecycle management, 376
  density, 377
  health monitoring, 376, 380–383
  high availability, 376
  simple, 377–383
  guest containers, 385–386
Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS), 278
HDFS (Hadoop Distributed File System), 278
Health Manager, 422
health model
about, 100–101
health entities, 101–102
health monitoring, 376, 380–383
health policy, 102–103
health reporting and aggregation, 103
health states, 102
HealthCheckRetryTimeoutSec upgrade parameter, 108
HealthCheckStableDurationSec upgrade parameter, 108
HealthCheckWaitDurationSec upgrade parameter, 108
Hello, World program, 17–21, 96, 114–116
heterogeneous clusters, 257
heterogeneous instances, 146–147
high availability, 376
homogeneous clusters, 146–147
horizontal scaling, 145–146
hosted clusters, provisioning, 12–17
hosting guest applications
about, 375–376
application lifecycle management, 376
density, 377
health monitoring, 376
high availability, 376
simple, 377–383
hosting subsystem
about, 8, 422
game hosting, 357
hosting guest applications, 375–383
HTTP protocol, 195, 277
HttpListener class, 44
hub (Microsoft Azure management portal), 12
Hyper-V, 384
Hyper-V Containers, 388–389
I
IaaS (infrastructure as a service), 5–6
IActor interface, 70
IActorEventPublisher interface, 81
IActorEvents interface, 81
IActorTimer interface, 79–80
IAppBuilder interface, 44–45
ICommunicationListener interface
implementing communication stack, 20, 33, 44–45
stateful services, 53
stateless services, 44–45
Idle Secondary replica role, 65
image store, 99–100, 422
images, accessing from Docker Hub, 375, 437
implicit hosts
defining, 111–112
hosting Node.js applications, 114–116
RunAs policies, 112–114
Import-PfxCertificate cmdlet, 171
independent variables, 229
infrastructure as a service (IaaS), 5–6
InstanceCount property, 124
International Space Station (ISS), 5
Internet of Things (IoT)
about, 275–276, 301
command and control, 276–277
Command and Control scenario, 299–300
data generation and feedback, 276
data ingress, 277
data transformation and analysis, 277–278
device orchestration scenario, 300
presentation and actions, 278–279
remote monitoring scenario, 279–299
storage, 278
Invocations/Sec performance counter, 84
Invoke-ServiceFabricPartitionDataLoss cmdlet, 247
Invoke-ServiceFabricQuorumLoss cmdlet, 247
InvokeDataLoss testability action, 247
InvokeDataLossAsync API, 247
InvokeDataLossAsync() method, 144
InvokeQuorumLoss testability action, 247
InvokeQuorumLossAsync API, 247
IoT (Internet of Things)
about, 275–276, 301
command and control, 276–277
Command and Control scenario, 299–300
data generation and feedback, 276
data ingress, 277
data transformation and analysis, 277–278
device orchestration scenario, 300
presentation and actions, 278–279
remote monitoring scenario, 279–299
storage, 278
IoT Hub (Azure)
about, 277, 279
Azure Stream Analytics service, 298
event ingress, 286–288
registering a device, 288–289
saving sensor data, 293
IReliableDictionary interface, 54
IReliableStateManager interface, 142–143
IRemindable interface, 80
IsCancellationRequested property, 20
IService interface, 33
isolateability (testing), 229–230
isolation levels, 155–156
ISS (International Space Station), 5
IStateSerializer interface, 158

J
Java framework, 384
journey (Microsoft Azure management portal), 12

K
Katana Project, 44
key-value store (KVS) state provider, 91
Key Vault service (Azure), 168, 259
Kibana platform, 208, 212–213
Kubernetes container orchestration, 439
KVS (key-value store) state provider, 91

L
latency, stateless services, 11
Lean movement, 4
leasing manager, 293
licensing, perpetual vs. subscription, 326–327
Linux platform, 391
load balancing
availability and, 119
dedicated, 119
endpoints, 195
Gateway service, 128
scaling clusters, 149
service defragmentation, 125–126
stateless service, 32
virtual machines and, 193–194
load metrics reports, 122
load tests, 244–246
Local Cluster Manager, 97–98
local clusters
administrative rights, 18
managing, 22–28
provisioning, 12–17
local store settings, 92
loggers and log files
Azure Diagnostics, 133
reporting to Elasticsearch, 213–215
Transactional Replicator, 55
Login-AzureRmAccount cmdlet, 425
loose coupling pattern, 160, 444

M
management portal (Azure)
about, 12
accessing Docker images, 437
Applications Insight instance, 221–223
configuring continuous deployment, 254
configuring placement properties, 123
preparing build machine, 237–239
scaling clusters, 149
management subsystem
about, 8, 422
Cluster Manager, 36, 142, 194–195, 422
Health Manager, 422
image store, 99–100, 422
manual scaling, 148–149
Manual upgrade mode, 108
Marketplace (Azure), 258
mass-source website scenarios
about, 264
Fan-out indexes pattern, 265–266, 443
Service API pattern, 266–267, 444
Massive Multiplayer Online Game (MMOG)
about, 358
Aggregation pattern, 362
Coverage Map pattern, 366–367, 442
MaxAsyncCommitDelay local store setting
describing game world, 358–360
dynamic clustering, 367
Dynamic Game Room pattern, 363–366, 442
global map, 362–363
Partitioned world pattern, 360–361
player interactions, 363–367
spatial interpolation, 366
spatial model, 360
MaxAsyncCommitDelay local store setting, 92
MaxPercentUnhealthyDeployedApplications upgrade parameter, 109
MaxPrimaryReplicationQueueSize configuration setting, 92
MaxReplicationMessageSize configuration setting, 92
MaxSecondaryReplicationQueueSize configuration setting, 92
MaxVerPages local store setting, 92
mean time to repair (MTTR), 219–220
MemoryInMb metric, 122
Mesos container orchestration, 387
message-based integration pattern, 271–272, 441
Messy Chess (game)
about, 345
challenges, 347–348
chessboard and game goal, 346–347
game board, 348–352
game hosting, 357
game pieces, 352–355
game rules, 346–347
Observer pattern, 351–352, 443
players, 355–357
metadata-driven system pattern, 335–338, 442
metrics
App Services view, 254–255
batching load reports, 122
resource balancing, 120–122
Microservices
about, 385
additional services, 267–268
design considerations, 3–4
loose coupling, 160
reliability and, 118
Microsoft account, 216, 232, 238, 425
Microsoft Azure management portal. See management portal (Azure)
Microsoft Azure products. See beginning with Azure
Microsoft Azure Service Fabric
about, 3, 28
applications and services, 10
architectural overview, 7–8
Hello, World, 17–21
managing local clusters, 22–28
nodes and clusters, 9
partitions and replicas, 10
programming modes, 10
provisioning clusters, 12–17
setting up development environment, 11–12
stateless vs. stateful, 10–11
Microsoft CDN service, 268
Microsoft Excel, 148, 279
Microsoft-ServiceFabric-Actors event source provider, 82, 133, 205
Microsoft-ServiceFabric-Services event source provider, 133–134, 206
Microsoft Web Platform Installer (Web PI), 12
Microsoft.Diagnostics.Listeners library, 213
Microsoft.Owin.Host.HttpListener namespace, 45
MMOG (Massive Multiplayer Online Game)
about, 358
Aggregation pattern, 362
Coverage Map pattern, 366–367, 442
describing game world, 358–360
dynamic clustering, 367
Dynamic Game Room pattern, 363–366, 442
global map, 362–363
Partitioned world pattern, 360–361
player interactions, 363–367
spatial interpolation, 366
spatial model, 360
Mock-based Service Design pattern, 260–261, 444
modeling complex systems, 69–70
Mona Lisa (painting), 394–395
Monitored upgrade mode, 107, 110–111
monitoring. See diagnostics and performance monitoring
Monte Carlo simulation, 163–164
Move Primary testability action, 247
Move-ServiceFabricPrimaryReplica cmdlet, 247, 250
Move-ServiceFabricSecondaryReplica cmdlet, 247
MoveNextApplicationUpgradeDomainAsync() method, 108
MovePrimaryAsync API, 247
MoveSecondaryAsync API, 247
MQTT protocol, 277
MTTR (mean time to repair), 219–220
multiplayer gaming
  about, 345
  A.I. Quests, 358–367
  Messy Chess, 345–357
multiple environment deployment, 110–111
multiple partitions, 61
multitenancy
  about, 325
  Azure support, 327–330
  cross-tenant aggregation pattern, 332–333, 442
  data architectures, 329–330
  hosting service processes, 339–340
  metadata-driven system pattern, 335–338, 442
  pattern index, 442
  scalability and, 147
  self-service pattern, 333–334, 442
  Service Fabric support, 330–338
  single tenancy versus, 147, 326–327
  tenant by partitions, 334–335
  Tenant Manager pattern, 331–332, 442
  Throttling Actor pattern, 340–344, 442

N

NamedPartition element, 61, 64–65
naming conventions
  configuration sections, 91
  performance counters, 85
  service names, 147
Naming Service (system naming service), 36, 128, 195, 423
NAT rules, 193–194
nearest-neighbor interpolation, 366
.NET Framework, 384
Netflix Chaos Monkey, 137
network interface cards (NICS), 193
Network Security Groups (NSGs), 199–202
New-AzureResourceGroupDeployment cmdlet, 429
New-AzureRm* cmdlets, 427
New-AzureRmKeyVault cmdlet, 168
New-AzureRmOperationalInsightsStorageInsight cmdlet, 218
New-AzureRmResourceGroup cmdlet, 168, 428
New-AzureRmResourceGroupDeployment cmdlet, 208
New-AzureRmVirtualNetwork cmdlet, 427
New-AzureRmVM cmdlet, 427
New-AzureVM cmdlet, 427
New-Container cmdlet, 440
New Project dialog box, 18
New-SelfSignedCertificate cmdlet, 168
New Service Fabric Service dialog box, 18–19
New-ServiceFabricApplication cmdlet, 184
New-ServiceFabricNodeConfiguration cmdlet, 183
NICS (network interface cards), 193
NodeType Configurations blade, 122–123
NodeType Properties blade, 122–123
Node.js-based applications
  defining implicit hosts, 111–112
  hosting, 114–116
nodes (cluster)
  about, 9
  containers and, 9
  fault domains and update domains, 105
  health model, 101
  node operation commands, 181–183
  placement constraints, 122–123
  query commands, 176–177, 180–181
  restarting, 138–142
  sample health policy, 102–103
  scaling process, 67–68
Nodes view (Server Explorer), 22
NodeType Name property, 122–123
None replica role, 65
Notification Hubs service, 268
npm gallery, 375
NSGs (Network Security Groups), 199–202
NuGet for .NET gallery, 375
NuGet packages
  Microsoft.AspNet.WebApi.OwinSelfHost, 45
  Microsoft.Azure.ServiceBus.EventProcessorHost, 310, 312
# of actor calls waiting for actor lock performance counter, 85

O

observability (testing), 228–229
Observer pattern, 351–352, 443
Ok health state, 102
OMS (Operations Management Suite)
  about, 216
  linking to data source, 217–219
  provisioning workspace, 216–217
OnActiveAsync() method, 74, 79
OnActiveSync() method, 70–71
onDataLossEvent event, 143–144
OnDeactiveAsync() method, 70
online voting system, 161–162
OPC (Open Platform Communication), 301
  Open Platform Communication (OPC), 301
Open Web Interface for .NET (OWIN), 43–45
OpenAsync() method, 44
Operational Insights (Azure), 216
Operations Management Suite (OMS)
  about, 216
  linking to data source, 217–219
  provisioning workspace, 216–217
orchestration bottlenecks
  Aggregation pattern, 159–162
  coordination pattern, 160, 162–164
orchestration engine, 374–375
OWIN (Open Web Interface for .NET), 43–45
Owner role, 197

P

PaaS (platform as a service)
  about, 3, 371–372
  agility, 3–5
  application gallery, 374–375
  application package format, 372–374
  Azure ecosystem, 253–259
  choosing platforms, 258–259
  hosting guest applications, 375–383
  placement constraints, 7
  Quality of Service, 5–6
  separation of workload and infrastructure, 6–7
  stateless services, 146
  upgrade process, 107
PackageRoot folder, 37, 39
packing applications
  application package format, 372–374
  deployment process, 95–99, 142
  rolling upgrades, 129–131
  service manifests, 37–38
  as services, 378–379
  partial updates, 257
Partition node, 23
partition tolerance (CAP theorem), 53
Partitioned world pattern, 360–361
partitions and partitioning
  about, 10
  actor states, 90, 156–157
  cloud gateways, 277
  health model, 101
  implementing tenants, 334–335
  local environment constraints, 62
  multiple, 61
  multitenancy and, 147
  Partitioned world pattern, 360–361
  query commands, 180
  scaling by, 257
  Simple Store application, 61–65
  stateful services, 65–68, 146
  stateless services, 32
PartitionSelector class, 141
Paxos algorithm, 65, 68
performance counters, 84–85, 93–94
Performance Monitor (Windows), 85–88, 94
performance monitoring, See diagnostics and performance monitoring
perpetual licensing, 326–327
persistent shopping cart pattern, 263–264, 443
Personalization Actor pattern, 261–263, 444
.pfx file extension, 169
placement constraints (nodes), 122–123
Placement Properties blade, 122–123
platform as a service (PaaS)
  about, 3, 371–372
  agility, 3–5
  application gallery, 374–375
  application package format, 372–374
Azure ecosystem, 253–259
choosing platforms, 258–259
placement constraints, 7
Quality of Service, 5–6
separation of workload and infrastructure, 6–7
stateless services, 146
upgrade process, 107
players (games)
  about, 355
  automated, 355–356
  Command pattern, 356
  coordinating, 356–357
  Coverage Map pattern, 366–367, 442
dynamic clustering, 367
Dynamic Game Room pattern, 363–366, 442
staging area, 361
  Throttling Actor pattern, 357
Power BI tool, 279, 298–299
PowerShell (Azure)
  about, 187
  application management commands, 184–187
  cluster management commands, 172–183
  creating secured clusters, 167–172
discovering commands, 427
installing, 425
  invoking testability actions, 248–250
managing resource groups, 427–431
signing in, 167, 425–426
PowerShell (Windows)
  about, 27–28, 187
  application management commands, 184–187
  application upgrades, 105
  chaos tests, 138
  cluster management commands, 172–183
  creating secured clusters, 167–172
  invoking testability actions, 248–250
Windows containers, 439–440
Preview Database Update dialog box, 292
Primary Count metric, 120
primary replica, 10, 65, 83
Primary replica role, 65
Principals element, 112–113
probes (load balancing rules), 195–196
processing topologies with actors pattern, 314–317, 441
programming modes (applications), 10
Project Katana, 44
publish profiles, 50, 110–111
Publish Service Fabric Application dialog box, 34, 50, 106, 130, 172
publishing applications from Visual Studio, 172
PublishProfiles folder, 50
pull aggregation, 285–286, 443
push aggregation, 283–285
Python Package Index, 375

Q

QoS (Quality of Service)
  about, 5, 259
  availability and, 6
  performance tuning, 88
  reliability and, 6
  scalability and, 5–6
Quality of Service (QoS)
  about, 5, 259
  availability and, 6
  performance tuning, 88
  reliability and, 6
  scalability and, 5–6
queries (ASA), 305
queries (PowerShell cmdlets)
  application-level, 177–178
  cluster-level, 172–176
  on cluster nodes, 180–181
  node-level, 176–177
  partition-level, 180
  replica-level, 180
  service-level, 179
Quickstart Templates (Azure), 150, 428–430, 439

R

RabbitMQ message broker, 277
RBAC (Role-Based Access Control), 197–198
RBAC (Role-Based Access Control) policies

RBAC (Role-Based Access Control) policies, 15
RDFE (Azure Service Management), 427
read isolation level, 155–156
Reader role, 197
readOnly property, 72, 75, 80–81
real-time data streaming
about, 303, 323
Azure Stream Analytics, 298, 304–306
big data, 303–304, 306–307
composable data processing units pattern, 308–314, 441
processing topologies with actors pattern, 314–317, 441
responsive website scenario, 317–323
ReceiveReminderAsync() method, 80
Redis Cache, 259
redundancy, 6, 10, 118–119
Reentrant attribute, 72
Register-ServiceFabricApplicationType cmdlet, 184–185
Register-ServiceFabricClusterPackage cmdlet, 183
registering applications, 100
RegisterReminder() method, 80
RegisterTimer() method, 79
Registry Editor, 237
removable storage, 259, 278, 290–292, 298, 329–330
reliability (service). See also availability (service)
about, 131–132
Azure Diagnostics, 133–136
broken service, 117–118
chaos tests, 137–142
design considerations, 6, 132
Event Tracing for Windows, 132
improving, 118
pattern index, 443
response time, 89
server state backups, 142–143
server state restore, 143–144
stateful services, 11
reliability subsystem
about, 8, 421–422
Failover Manager, 36, 120, 421
Resource Balancer, 120–123, 421
Transactional Replicator, 53, 55, 421
Reliable Actors API, 10, 70–72, 99, 111
reliable collections
about, 54
isolation levels, 155–156
Reliable Dictionary, 54–55, 90, 155
Reliable Queue, 54, 155, 160
Reliable Services API, 10, 89–90, 111
Reliable State Manager
about, 53–55
service state backups, 142
state serialization, 157–159
relying parties, 269
reminders (Actor pattern), 80–81
Remote Desktop, 192
remote monitoring scenario
about, 279
Aggregation pattern, 159–162, 281–286, 443
distributed scenario, 286–299
Sensor Actor pattern, 279–281, 444
Removable Storage Management (RSM), 273–274
Remove-AzureRm* cmdlets, 427
Remove-AzureRmResourceGroup cmdlet, 428
Remove-ServiceFabricApplication cmdlet, 187
Remove-ServiceFabricApplicationPackage cmdlet, 187
Remove-ServiceFabricClusterPackage cmdlet, 183
Remove-ServiceFabricNodeConfiguration cmdlet, 183
Remove-ServiceFabricNodeState cmdlet, 182–183
Remove-ServiceFabricReplica cmdlet, 248
Remove-ServiceFabricTestsState cmdlet, 247
RemoveReplica testability action, 248
RemoveReplicaAsync API, 248
repeatable read isolation level, 155–156
replacing applications, 100
Replica Count metric, 120
Replica node, 23
ReplicaChangeRoleFromPrimary event, 83
ReplicaChangeRoleToPrimary event, 83
replicas
about, 23
availability and, 146
creating, 119
data changes, 55
distributing, 120, 125–126
events supported, 83
health model, 101
query commands, 180
removing, 142
replacing, 119
roles supported, 65–67
scaling process and, 67–68
service defragmentation, 125–126
replicator configuration, 91–92
ReplicatorEndpoint configuration setting, 92
resetting clusters, 26, 62
Resource Balancer, 120–123, 421
Resource Explorer (Azure), 206–207
resource groups
  about, 15, 150, 202
  creating, 168
  defining, 428–429
  deploying, 429–430
  modifying with Azure Resource template, 204
Resource Manager API (Azure), 151, 427
resource scheduling system, 146
resources
  about, 15, 150, 202
  creating in Microsoft Azure, 13
  dedicated, 326
  orchestrating, 374–375
  scheduling, 146
  shared, 326
response time
  bottlenecks and, 154–155
  state providers and, 89, 92–93
responsive website scenario
  about, 317–318
  Country/Region Actor, 319
  gateway implementation, 320–321
  Global Actor, 319–320
  Product Actor, 318–319
  test client, 322–323
  Web Socket listener, 321
REST API, 220
 Restart-ServiceFabricDeployedCodePackage cmdlet, 248
 Restart-ServiceFabricNode cmdlet, 248–249
 Restart-ServiceFabricPartition cmdlet, 248
 Restart-ServiceFabricReplica cmdlet, 248
 RestartDeployedCodePackage testability action, 248
 RestartDeployedCodePackageAsync API, 248
 RestartNode testability action, 248
RestartNodeAsync API, 248
RestartPartition testability action, 248
RestartPartitionAsync API, 248
RestartReplica testability action, 248
RestartReplicaAsync API, 248
restore, server state, 143–144
RestoreAsync() method, 143
Resume-ServiceFabricClusterUpgrade cmdlet, 183
RetryInterval configuration setting, 92
Role-Based Access Control (RBAC), 197–198
Role-Based Access Control (RBAC) policies, 15
rolling back applications, 186
rolling upgrades
  about, 100, 104, 257
  configuration and data changes, 129–131
  with diff packages, 131
  fault domains, 104–105
  update domains, 104–105
  upgrade modes, 107–108
  upgrade parameters, 108–110
  upgrade process, 105–107
RPC Proxy, 33
RSM (Removable Storage Management), 273–274
RunAs policies, 112–114
RunAsync() method, 20, 34, 99

scalability
  autoscaling, 148–153
  broken service and, 117–118
  complexity and, 5
  design considerations, 5–6
  heterogeneous instances, 146–147
  homogeneous instances, 146–147
  horizontal, 145–146
  manual scaling, 148–149
  multitenancy, 147
  pattern index, 443
  resolving bottlenecks, 154–164
  scaling clusters, 148–154
  single-tenancy, 147
  stateful services, 146
  stateless services, 32, 146
  vertical, 145–146
scale sets. See Azure Virtual Machine Scale Sets

scaling process
  relocating replicas, 67–68
  scaling out approach, 6, 11, 146, 156, 257
  scaling up approach, 6
secondary replicas, 10, 65
security tokens, 269–270
self-adapting actors, 443
self-adaptive scaling, 408
self-driven workflow pattern, 162, 441
self-service pattern, 333–334, 442
self-signed certificates, 167–171
Send-ServiceFabricClusterHealthReport cmdlet, 183
Sensor Actor pattern, 279–281, 444
Server Explorer (Visual Studio), 22–23
service affinity, 124–125
Service API pattern, 266–267, 444
service availability. See availability (service)
Service Bus Queue (Azure), 160
service defragmentation, 125–126
service discovery process, 128
Service Fabric (Azure)
  about, 3, 28, 257
  applications and services, 10
  architectural overview, 7–8
  ASA integration, 305–306
  choosing as platform, 258–259
  for complex systems, 404–408
  containers and, 385–388
  Hello, World, 17–21
  managing local clusters, 22–28
  multitenancy support, 331–338
  nodes and clusters, 9
  partitions and replicas, 10
  programming modes, 10
  provisioning clusters, 12–17
  setting up development environment, 11–12
  stateless vs. stateful, 10–11
Service Fabric Actor Method performance counters, 84
Service Fabric Actor performance counters, 85
Service Fabric Cluster blade, 122–123
Service Fabric Explorer
  about, 24–26
  application view, 30, 338
cluster map, 191
failover tests, 127–128
managing clusters, 167, 171, 194, 220–221, 389
partition view, 119, 122
service instances, 382
system services, 424
tree view, 66, 90
viewing replicas, 36, 40
Visual Studio Application Insights, 254
Web Socket port, 323
Service Fabric node runtime, 192
Service Fabric Reliable Actors API, 10, 70–72, 99, 111
Service Fabric Reliable Services API, 10, 89–90, 111
Service Fabric SDK, 12, 24
service failovers. See failover mechanisms
service instances
  about, 23
  multiple environment deployment, 110
  scaling, 145–146
  stateless actor events, 83
  testability considerations, 229
service-level queries, 179
Service Management API (Azure), 142, 151, 427
service manifest files
  about, 37–39, 96–97
  configuring endpoint, 47
  defining custom metrics, 121
  diff packages, 131
  guest containers, 385
  implicit hosts, 111, 115
  upgrade process, 105, 185
service placements
  about, 119
  batching load reports, 122
  Failover Manager, 120
  placement constraints, 122–123
  Resource Balancer, 120
  resource balancing metrics, 120–122
  service affinity, 124–125
  service defragmentation, 125–126
service proxy, 229
service reliability. See reliability (service)
service replicas. See replicas
Service Type node, 23
ServiceCorrelation element, 124
ServiceEventSource class, 20, 134
ServiceFabricReliableActorEventTable table, 133, 205
ServiceFabricReliableServiceEventTable table, 133, 206
ServiceFabricSystemEventTable table, 133
ServiceInitializationParameters attribute, 38, 48
ServiceInstanceClose event, 83
ServiceInstanceOpen event, 83
ServiceManifestImport element, 38
ServiceTemplates element, 121
ServiceTypeHealthPolicyMap upgrade parameter, 109
ServiceTypes element, 37
Set-AzureRm* cmdlets, 427
Set-AzureRmContext cmdlet, 426
Set-AzureRmKeyVaultAccessPolicy cmdlet, 168
Set-AzureRmVMExtension cmdlet, 435
Settings.xml file
  about, 36, 38
  actor state providers, 91
  custom sections and parameters, 99
  response time, 92
SetupEntryPoint element, 112–113
shared data structures, 443
shared locks, 156
shared resources, 326
shared-state swarm, 408
Simian Army tools, 137
Simple Store application
  about, 56
  client, 59–61
  partitions, 61–65
  shopping cart service, 56–59
simulated errors, 246–247
single points of failure (SPoF), 132
Single Sign-On (SSO), 269, 327
single-tenancy, 147, 325–327, 330
SingletonPartition element, 61
Smith, Adam, 395
snapshot isolation level, 155–156
snapshots (checkpoints), 142
software testability
  about, 227
  clarity, 230
state management
  actors, 70–71, 90
  application deployment, 111
  events supported, 83
  performance monitoring, 85–89
  Reliable State Manager, 53–55
  replica roles, 65–67
  resolving bottlenecks, 155–159
  state serialization, 157–159
state providers
  about, 91
  actor states, 91–94, 156–157
controllability, 228, 246
isolateability, 229–230
observability, 228–229
writing test cases, 230–231
solid-state drives (SSDs), 192
spatial segregation model
  about, 409
  Commit phase, 410
  Conflict resolution phase, 410
  implementing Actor Swarm, 414–416
  implementing shared array, 409–412
  implementing test client, 416–417
  implementing virtual actor, 412–414
  Proposal phase, 409
  Read phase, 409
  setting up solution, 409
  testing model, 417
  Think phase, 409
spatial swarm, 408
SPoF (single points of failure), 132
SQL Data Warehouse, 259, 329
SQL Database, 259, 278, 290–292, 298, 329
SQL Server Object Explorer, 292
SSDs (solid-state drives), 192
SSO (Single Sign-On), 269, 327
Start-Container cmdlet, 440
Start-ServiceFabricApplicationRollback cmdlet, 186
Start-ServiceFabricApplicationUpgrade cmdlet, 186
Start-ServiceFabricClusterRollback cmdlet, 183
Start-ServiceFabricClusterUpgrade cmdlet, 183
Start-ServiceFabricNode cmdlet, 248
StartNode testability action, 248
StartNodeAsync API, 248
stateful actors

- in-memory, 87
- reliable collections, 54

stateful actors
- about, 70–71
- events supported, 83
- tic-tac-toe game, 73–76

stateful services
- about, 10–11, 53, 65–68, 257
- architectural overview, 53–56
- big data storage, 307
- partitions and replicas, 65–68, 146
- Reliable Services APIs, 10
- scalability, 146
- service affinity, 124
- Simple Store application, 56–65
- state management, 53

StatefulActor class, 70
StatefulService class, 143
StatefulServiceBase class, 58

stateless actors
- about, 70–71
- events supported, 83
- tic-tac-toe game, 72

stateless services
- about, 10–11, 29, 51
- application deployment, 111
- ASP.NET 5 applications, 29–31
- availability, 32
- implementing, 20
- implementing communication stacks, 33–50
- latency, 11
- partitioning, 32
- Reliable Services APIs, 10
- scalability, 32, 146
- service affinity, 124

StatelessActor class, 70
StatelessService class, 20, 38

Stop-ServiceFabricNode cmdlet, 248
StopNode testability action, 248
StopNodeAsync API, 248
storage accounts, 196–197
storage tables, 133, 135–136
streamed data analytics, 277
SubscribeAsync() method, 82
subscription licensing, 326–327

TCP protocol, 195, 277
TDD (Test-Driven Development) process, 260
Team Explorer, 234–236
Team Foundation Server (TFS), 232
Team Foundation Service (TFS), 232
Tenant Manager pattern, 331–332, 442

termite model
- about, 396–397
- implementing Box service, 398–400
- implementing termite actor, 400–402
- implementing test client, 402–403
- setting up solution, 397
- test and analysis, 403–404

test cases
- automated, 228
- clarity of, 230
- writing, 230–231

Test-Driven Development (TDD) process, 260
Test Explorer, 231
Test-ServiceFabricApplication cmdlet, 248
Test-ServiceFabricClusterManifest cmdlet, 183
Test-ServiceFabricService cmdlet, 248

Testability subsystem
- about, 8, 227, 246–250, 423
- continuous integration, 131, 228, 232–246
- player simulation, 348
- software testability, 227–230
- writing basic test cases, 230–231

testing in production technique, 137–138
TFS (Team Foundation Server), 232
TFS (Team Foundation Service), 232
The Theory of Moral Sentiments (Smith), 395
Throttling Actor pattern, 340–344, 357, 442

throughput
  batching requests, 160
  latency and, 94
  real-time data streaming and, 303
  resolving bottlenecks, 154, 162–163
  response time and, 154, 162
  virtual machine size and, 192

thumbprint (certificates), 169–171

tic-tac-toe game
  actor models, 72–73
  creating application, 73
  defining actor interfaces, 73
  implementing Game actor, 74–76
  implementing Player actor, 76
  implementing test client, 76–78
  improving, 78–79
  setting up continuous integration, 232–246
  testing, 78
  writing test cases for, 230–231

tile (Microsoft Azure management portal), 12

time-to-live (TTL) limit, 273
timeouts, transaction, 156
TimeoutSec upgrade parameter, 109
timers (Actor pattern), 79–80
Traffic Manager (Azure), 259, 357
Transactional Replicator, 53, 55, 421
transport subsystem, 8, 423
troubleshooting tools, 254
TryAddStateSerializer() method, 159
TTL (time-to-live) limit, 273
Turecek, Vaclav, 45

Ubuntu Server, 433–434

UDFs (user-defined functions), 306
ungraceful faults, 141–142
UniformInt64Partition element, 61–64
Unknown health state, 102
Unknown replica role, 65
UnmonitoredAuto upgrade mode, 107
Unregister-ServiceFabricApplicationType cmdlet, 187
Unregister-ServiceFabricClusterPackage cmdlet, 183
UnregisterReminder() method, 81

UnRegisterTimer() method, 79
UnsubscribeAsync() method, 82
update domains, 104–105, 190–191
update locks, 156
Update-ServiceFabricClusterUpgrade cmdlet, 183
Update-ServiceFabricNodeConfiguration cmdlet, 183
UpgradeDomainTimeoutSec upgrade parameter, 109
UpgradeReplicaSetCheckTimeout upgrade parameter, 109
upgrading applications, 100, 104–110, 185–186
uploading applications, 99–100
UseImplicitHost flag, 111
User Access Administrator role, 197
user-defined functions (UDFs), 306
user groups, 113

V

ValidateApplication testability action, 248
ValidateApplicationAsync API, 248
ValidateService testability action, 248
ValidateServiceAsync API, 248
value (big data), 304
variety (big data), 304
velocity (big data), 304
veracity (big data), 304
vertical scaling, 145–146

virtual machines
  availability sets and, 191
  Azure Diagnostics extension, 205
  load balancing and, 193–194
  modifying configurations, 192
  NAT rules, 193–194
  NICs and, 191–193
  virtual networks, 193

virtualization technologies, 383–384

Visual Studio
  launching as administrator, 18
  publishing applications, 172
  upgrade process, 105–106

Visual Studio Application Insights
  about, 84, 259
alert rules, 223–225
creating instances, 221–222, 254
sending events, 222–223
Visual Studio Cloud Explorer, 23–24
Visual Studio Online (VSO), 232
Visual Studio Server Explorer, 22–23
Visual Studio Team Services (VSTS), 232–236
  about, 259
  creating build definition, 239–242
  preparing build machine, 237–239
  running load tests, 244–246
  running testing upon code check-ins, 242–244
Visual Studio Unit Test Framework, 230–231
VolatileActorStateProvider attribute, 87, 156
volume (big data), 303
Voronoi diagram, 365–367
VSO (Visual Studio Online), 232
VSTS (Visual Studio Team Services)
  about, 232, 259
  creating build definition, 239–242
  preparing build machine, 237–239
  preparing project, 232–236
  running load tests, 244–246
  running testing upon code check-ins, 242–244
W

WADServiceFabricReliableServiceEventTable table, 135
Warning health state, 102
watchdogs, 103, 380–383
WCF communication stack, 41–43
  Simple Store application, 56–59
web applications
  about, 253, 274
  Azure PaaS ecosystem, 253–259
  e-commerce websites, 260–264
  enterprise portals, 268–274
  mass-source websites, 264–268
Web Jobs, 255
Web PI (Web Platform Installer), 12
Web Platform Installer (Web PI), 12
web servers, console-based, 377–378
Web Socket listener, 318, 321
WebUser account, 113
Windows Performance Monitor, 85–88, 94

Windows PowerShell
  about, 27–28, 187
  application management commands, 184–187
  application upgrades, 105
  chaos tests, 138
  cluster management commands, 172–183
  creating secured clusters, 167–172
  invoking testability actions, 248–250
Windows containers, 439–440
Windows Server Containers, 388, 439–440
workflows, self-driven, 162, 441
WorkingDirectory element, 98
WorkingFolder element, 98
workloads
  distributing, 6
  separating infrastructure and, 5–6
writing test cases, 230–231

X

X509 certificates, 91

Z

Zen of Cloud (Bai), 195
zero-downtime upgrades, 100, 104–110