Gaurav Kumar Aroraa, Lalit Kale, Kanwar Manish

Foreword by:
Ed Price
Senior Program Manager
Microsoft AzureCAT (Customer Advisory Team), Microservices and Cloud Development

Building Microservices with .NET Core

Transitioning monolithic architecture using microservices with .NET Core

Packt

Building Microservices with .NET Core

Transitioning monolithic architecture using microservices with .NET Core

Gaurav Kumar Aroraa
Lalit Kale
Kanwar Manish

BIRMINGHAM - MUMBAI

Building Microservices with .NET Core

First published: June 2017

Production reference: 1120617

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.
ISBN 978-1-78588-783-3

www.packtpub.com

Credits

Authors
Gaurav Kumar Aroraa
Lalit Kale
Kanwar Manish

Reviewers
Vidya Vrat Agarwal
Nishith Shukla

Commissioning Editor
Veena Pagare

Acquisition Editor
Denim Pinto

Content Development Editor
Vikas Tiwari

Technical Editor
Diwakar Shukla

Copy Editor
Gladson Monteiro

Project Coordinator
Ulhas Kambali

Proofreader
Safis Editing

Indexer
Tejal Daruwale Soni

Graphics
Abhinash Sahu

Production Coordinator
Shantanu N. Zagade

Foreword

"Our industry does not respect tradition – it only respects innovation."
- Satya Nadella

I've spent my last three years at Microsoft, running customer feedback programs for Azure microservice architectures and tooling. I believe this microservices framework is a crucial spark of innovation in web development. In an agile world, we need an agile framework on the cloud that is working for us, processing individual actors and services. With this new power, we can deploy a framework that scales, improves resiliency, greatly reduces latency, increases our control of security, and upgrades the system without downtime. Microservices becomes the optimal architecture in our new cloud-based development environment, and it can result in major cost benefits.

Gaurav Aroraa, Lalit Kale, and Manish Kanwar masterfully whisk us away on a journey to explore the history of microservices, and they carefully and thoroughly take us on a tour of the architectural design concepts that accompany the evolution of microservices, from when James Lewis first coined the term to our current tools and implementations. The book starts at a high level, with detailed diagrams and descriptions that explain the architectural scenarios and uncovers all the values you'll receive with a microservices design. At this point, you might ask whether the book is about microservices architecture or a how-to guide in .NET development. Importantly, the authors transition us into the practical knowledge of translating our current applications into this bold new world of microservices. On that journey, they do not speed up. In other books, you move so fast that you simply cannot enjoy the view (or understand what you're supposed to be learning). You might just implement the code and pick up a few tactics along the way, mostly copying and coding by autopilot. But the authors teach each concept and step in the development process with the attention and focus that it deserves.

Personally, I have had the privilege of knowing Gaurav for a few years now. He's a Visual Studio and Development MVP (Microsoft's Most Valuable Professional award) and a key leader in the Microsoft cloud development community. I've worked closely with him on his powerful contributions on TechNet Wiki. In this book, I see a dedication and passion from Gaurav, Lalit, and Manish shine through. This book needs to be written. I am excited when I find gems like this. The authors thoroughly go through every detail, every parameter, and every consideration in tackling this weighty concept of a microservices architecture in .NET development. Read this book, skip ahead where you're knowledgeable about the given information, absorb the authors' knowledge, and share the book with your business contacts. The development community needs to adopt a microservices approach, and this book is a powerful advocate on that journey.

Ed Price

Senior Program Manager

Microsoft AzureCAT (Customer Advisory Team), Microservices and Cloud Development

Co-Author of *Learn to Program with Microsoft Small Basic*

About the Authors

Gaurav Kumar Aroraa has done M.Phil in computer science. He is a Microsoft MVP, certified as a scrum trainer/coach, XEN for ITIL-F, and APMG for PRINCE-F and PRINCE-P. Gaurav serves as a mentor at IndiaMentor, webmaster of dotnetspider, contributor to TechNet Wiki, and co-founder of Innatus Curo Software LLC. In the 19+ years of his career, he has mentored thousands of students and industry professionals. You can reach Gaurav via his blog, LinkedIn, and twitter handle (`@g_arora`).

Book writing is not an easy job, as it takes a lot of time. Sometimes, it needs your personal/family time. So, I want to thank all who motivated me and allowed me to spend time on this book, time that I was supposed to spend with them. My first thank you is to my wife, Shuby Arora, for her support in all ways. Then, I would like to thank my angel, Aarchi Arora (the newest member of our family). A great thanks to my parents whose blessings are always with me; this is because of them. I would like to thank the entire Packt team, especially Vikas Tiwari and Denim Pinto for their overnight support. A great thank you to Ed Price for his in-depth knowledge and his suggestions to improve various sections of the book. Finally, I want to say thanks to both Lalit and Manish for their full support as co-authors and their reply when I need for the book discussion.

Lalit Kale is a technical architect and consultant with more than 12 years of industry experience. Lalit has helped clients achieve tangible business outcomes through the implementation of best practices in software development. He is a practitioner of TDD and DDD, and a big believer in agile and lean methodologies. He has worked with several organizations, from start-ups to large enterprises, in making their systems successful, be it in-house or mission critical, with clients in the USA, the UK, Germany, Ireland, and India. His current interests include container technologies and machine learning using Python. He holds a bachelor's degree in engineering (IT).

I would like to take this opportunity to thank my coauthors, Gaurav and Manish, and the entire Packt team, without whom this book would never have existed. I would also like to thank Lord Ganesha and my parents. Without their support, I would never have been creative and wouldn't have pursued my passion with computers. I would like to pay my respect to my source of inspiration--my beloved grandfather, Raghunath Savdekar, who passed away during the writing of this book. Grandpa, this book is for you.

Lastly, I'd like to acknowledge the support from my wife, Sonal, and my kid, Aaryan, who had to tolerate my demands for endless cups of coffee and peaceful silence during long writing nights.

Kanwar Manish completed his masters of science in computer applications from MD University, India, and is a cofounder of Innatus Curo Software LLC, with a presence in India. He has been working in the IT industry across domains for the last 17 years. He started exploring .NET right from the first release and has been glued to it ever since. His range of experience includes global wealth management (financial service industry, USA), life insurance (insurance industry, USA), and document management system (DMS), ECMS, India. Manish does his bit for the community by helping young professionals through the IndiaMentor platform.

I would like to thank my wife, Komal, and my young boys, Aadi and Veda, who had to bear my absence while I was still around and for giving me that crucial support. And a big thanks to the rest of my family for always encouraging me. Gaurav played a vital role in giving his valuable input in guiding me. Also, I'd like to acknowledge the support from Packt's editors.

About the Reviewers

Vidya Vrat Agarwal is software technology enthusiast, Microsoft MVP, C# Corner MVP, TOGAF Certified Architect, Certified Scrum Master (CSM), and a published author. He has presented sessions at various technical conferences and code camps in India and the USA. He lives in Redmond, WA with his wife Rupali and two daughters, Pearly and Arshika. He is passionate about .NET and works as a software architect/.NET consultant. He can be followed on Twitter at `@DotNetAuthor`.

Nishith Shukla is a seasoned software architect and has been a leader in developing software products for over 15 years. Currently, he is working in Bay Area, California for BlackBerry. He joined BlackBerry through the acquisition of AtHoc and is playing a key technical leadership role in transmitting BlackBerry from a hardware to a software company.

Nishith has played a key role in various software products through his extensive knowledge on OOP, design patterns, and architectural best practices, including microservices, and through his balanced approach between business goals and technical goals. Outside work, Nishith plays an active role in software community building, playing Golf, travelling the world, and spending time with his family.

www.PacktPub.com

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com`and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www.packtpub.com/mapt`

Get the most in-demand software skills with Mapt. Mapt gives you full access to all Packt books and video courses, as well as industry-leading tools to help you plan your personal development and advance your career.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Customer Feedback

Thanks for purchasing this Packt book. At Packt, quality is at the heart of our editorial process. To help us improve, please leave us an honest review on this book's Amazon page at `https://www.amazon.com/dp/1785887831`. If you'd like to join our team of regular reviewers, you can e-mail us at `customerreviews@packtpub.com`. We award our regular reviewers with free eBooks and videos in exchange for their valuable feedback. Help us be relentless in improving our products!

Table of Contents

Preface

Distributed systems are always difficult to get complete success with. Lately, microservices have been getting considerable attention. With Netflix and Spotify, microservices implementations have some of the biggest success stories in the industry. Microservices is quickly gaining popularity and acceptance with enterprise architects. On the other hand, there is another camp that thinks microservices as nothing new or only as a rebranding of SOA.

In any case, microservices architecture has critical advantages, particularly with regard to empowering the nimble improvement and conveyance of complex venture applications.

However, there is no clear practical advice on how to implement microservices in the Microsoft ecosystem and especially with taking advantage of Azure and the .NET Core framework.

This book tries to fill that void. It explores the concepts, challenges, and strengths of planning, constructing, and operating microservices architectures built with .NET Core. This book discusses all cross-cutting concerns, along with the microservices design. It also highlights the more important aspects to consider while building and operating microservices through practical *how tos* and best practices for security, monitoring, and scalability.

What this book covers

`Chapter 1`, *What Are Microservices?*, makes you familiar with microservices architectural styles, history, and how it differs from its predecessors, monolithic architecture and service-oriented architecture (SOA).

`Chapter 2`, *Building Microservices*, gives you an idea of the different factors that can be used to identify and isolate microservices at a high level, what the characteristics of a good service are, and how to achieve the vertical isolation of microservices.

`Chapter 3`, *Integration Techniques*, introduces synchronous and asynchronous communication, style of collaborations, and the API gateway.

`Chapter 4`, *Testing Strategies*, explores how testing microservices is different from testing a normal .NET application. It gets you acquainted with the testing pyramid.

`Chapter 5`, *Deployment*, covers how to deploy microservices and the best practices for it. It also takes into account the isolation factor, which is the key success factor, along with setting up continuous integration and continuous delivery to deliver business changes at a rapid pace.

`Chapter 6`, *Security*, describes how to secure microservices with OAuth and, also, container security and best practices in general.

`Chapter 7`, *Monitoring*, explains that debugging and monitoring microservices is not a trivial problem but a quite challenging one. We have used the word, *challenging*, on purpose--there is no silver bullet for this. There is no single tool in the .NET ecosystem that is, by design, made for microservices; however, Azure monitoring and troubleshooting is the most promising one.

`Chapter 8`, *Scaling*, explains that scalability is one of the critical advantages of pursuing the microservices architectural style. In this chapter, we will see scalability by design, and by infrastructure as well, with respect to the microservices architecture.

`Chapter 9`, *Reactive Microservices*, gets you familiar with the concept of reactive microservices. You will learn how you can build reactive microservices with the use of reactive extensions. The chapter will help you focus on your main task and free you from the chores of communicating across services.

`Chapter 10`, *Creating a Complete Microservices Solution*, will walk you through all the concepts of microservices that you have learned so far. Also, we will develop an application from scratch while putting all our skills to use.

What you need for this book

All supporting code samples in this book are tested on .NET Core 1.1, using Visual Studio 2015 update 3 as IDE and SQL Server 2008R2 as database on the Windows platform.

Who this book is for

This book is for .NET Core developers who want to learn and understand microservices architecture and implement it in their .NET Core applications. It’s ideal for developers who are completely new to microservices or just have a theoretical understanding of this architectural approach and want to gain a practical perspective in order to manage application complexity better.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Here we are trying to showcase how our `Order` module gets abstracted."

A block of code is set as follows:

```
namespace FlixOne.BookStore.ProductService.Models
   {
     public class Category
     {
       public Guid Id { get; set; }
       public string Name { get; set; }
       public string Description { get; set; }
      }
   }
```

Any command-line input or output is written as follows:

```
Install-Package System.IdentityModel.Tokens.Jwt
```

New terms and important words are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Clicking the **Next** button moves you to the next screen."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book-what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for this book from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

1. Log in or register to our website using your e-mail address and password.
2. Hover the mouse pointer on the **SUPPORT** tab at the top.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box.
5. Select the book for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this book from.
7. Click on **Code Download**.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Building-Microservices-with-DotNET-Core`. We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books-maybe a mistake in the text or the code-we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
What Are Microservices?

The focus of this chapter is to get you acquainted with microservices. We will start with a brief introduction. Then, we will define its predecessors: monolithic architecture and **service-oriented architecture** (**SOA**). After this, we will see how microservices fare against both SOA and the monolithic architecture. We will then compare the advantages and disadvantages of each one of these architectural styles. This will enable us to identify the right scenario for these styles. We will understand the problems that arise from having a layered monolithic architecture. We will discuss the solutions available to these problems in the monolithic world. At the end, we will be able to break down a monolithic application into a microservice architecture. We will cover the following topics in this chapter:

- Origin of microservices
- Discussing microservices
- Understanding the microservice architecture
- Advantages of microservices
- SOA versus microservices
- Understanding problems with the monolithic architectural style
- Challenges in standardizing the .NET stack

Origin of microservices

The term *microservices* was used for the first time in mid-2011 at a workshop of software architects. In March 2012, James Lewis presented some of his ideas about *microservices*. By the end of 2013, various groups from the IT industry started having discussions on *microservices*, and by 2014, it had become popular enough to be considered a serious contender for large enterprises.

There is no official introduction available for *microservices*. The understanding of the term is purely based on the use cases and discussions held in the past. We will discuss this in detail, but before that, let's check out the definition of microservices as per Wikipedia (`https://en.wikipedia.org/wiki/Microservices`), which sums it up as:

> *Microservices is a specialization of and implementation approach for SOA used to build flexible, independently deployable software systems.*

In 2014, James Lewis and Martin Fowler came together and provided a few real-world examples and presented *microservices* (refer to `http://martinfowler.com/microservices/`) in their own words and further detailed it as follows:

> *The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.*

It is very important that you see all the attributes James and Martin defined here. They defined it as an architectural style that developers could utilize to develop a single application with the business logic spread across a bunch of small services, each having their own persistent storage functionality. Also, note its attributes: it can be independently deployable, can run in its own process, is a lightweight communication mechanism, and can be written in different programming languages.

We want to emphasize this specific definition since it is the crux of the whole concept. And as we move along, it will come together by the time we finish this book.

Discussing microservices

Until now, we have gone through a few definitions of *microservices*; now, let's discuss *microservices* in detail.

In short, a microservice architecture removes most of the drawbacks of SOA architectures. It is more code-oriented (we will discuss this in detail in the coming sections) than SOA services.

Slicing your application into a number of services is neither SOA nor microservices. However, combining service design and best practices from the SOA world along with a few emerging practices, such as isolated deployment, semantic versioning, providing lightweight services, and service discovery in polyglot programming, is microservices. We implement microservices to satisfy business features and implement them with reduced time to market and greater flexibility.

Before we move on to understand the architecture, let's discuss the two important architectures that have led to its existence:

- The monolithic architecture style
- SOA

Most of us would be aware of the scenario where during the life cycle of an enterprise application development, a suitable architectural style is decided. Then, at various stages, the initial pattern is further improved and adapted with changes that cater to various challenges, such as deployment complexity, large code base, and scalability issues. This is exactly how the monolithic architecture style evolved into SOA, further leading up to microservices.

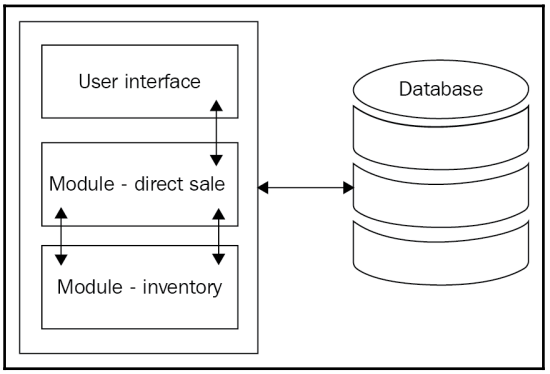
Monolithic architecture

The monolithic architectural style is a traditional architecture type and has been widely used in the industry. The term *monolithic* is not new and is borrowed from the Unix world. In Unix, most of the commands exist as a standalone program whose functionality is not dependent on any other program. As seen in the succeeding image, we can have different components in the application such as:

- **User interface**: This handles all of the user interaction while responding with HTML or JSON or any other preferred data interchange format (in the case of web services).

- **Business logic**: All the business rules applied to the input being received in the form of user input, events, and database exist here.
- **Database access**: This houses the complete functionality for accessing the database for the purpose of querying and persisting objects. A widely accepted rule is that it is utilized through business modules and never directly through user-facing components.

Software built using this architecture is self-contained. We can imagine a single .NET assembly that contains various components, as described in the following image:



As the software is self-contained here, its components are interconnected and interdependent. Even a simple code change in one of the modules may break a major functionality in other modules. This would result in a scenario where we'd need to test the whole application. With the business depending critically on its enterprise application frameworks, this amount of time could prove to be very critical.

Having all the components tightly coupled poses another challenge: whenever we execute or compile such software, all the components should be available or the build will fail; refer to the preceding image that represents a monolithic architecture and is a self-contained or a single .NET assembly project. However, monolithic architectures might also have multiple assemblies. This means that even though a business layer (assembly, data access layer assembly, and so on) is separated, at run time, all of them will come together and run as one process.

A user interface depends on other components' direct sale and inventory in a manner similar to all other components that depend upon each other. In this scenario, we will not be able to execute this project in the absence of any one of these components. The process of upgrading any one of these components will be more complex as we may have to consider other components that require code changes too. This results in more development time than required for the actual change.