Hands-On Software Architecture with C# 8 and .NET Core 3

Architecting software solutions using microservices, DevOps, and design patterns for Azure Cloud

www.packt.com

Gabriel Baptista and Francesco Abbruzzese

Hands-On Software Architecture with C# 8 and .NET Core 3

Architecting software solutions using microservices, DevOps, and design patterns for Azure Cloud

Gabriel Baptista
Francesco Abbruzzese

Packt

BIRMINGHAM - MUMBAI

Hands-On Software Architecture with C# 8 and .NET Core 3

Commissioning Editor: Kunal Chaudhari
Acquisition Editor: Alok Dhuri
Content Development Editor: Ruvika Rao
Senior Editor: Afshaan Khan
Technical Editor: Ketan Kamble
Copy Editor: Safis Editing
Project Coordinator: Francy Puthiry
Proofreader: Safis Editing
Indexer: Rekha Nair
Production Designer: Jyoti Chauhan

First published: November 2019

Production reference: 1291119

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78980-093-7

`www.packt.com`

Contributors

About the authors

Gabriel Baptista is a software architect who technically leads a team in the most diverse projects for retail and industry, using a dozen varieties of Microsoft products. He has become a specialist in Azure **Platform-as-a-Service** (**PaaS**) solutions since designing a **Software-as-a-Service** (**SaaS**) platform in partnership with Microsoft. Besides all that, he is also a college computing professor who has published many papers and teaches different subjects related to software engineering, development, and architecture. He is also a speaker on *Channel 9*, one of the most prestigious and active community websites for the .NET stack. As well as that, he is a cofounder of a start-up for developing mobile applications, where Scrum, design thinking, and DevOps philosophy are the keys to delivering user needs.

To my incredible kids, Murilo and Heitor, and my dear wife, Denise, who have always allowed me to move forward.

Francesco Abbruzzese is the author of the book *MVC Controls Toolkit*. He has also contributed to the diffusion and evangelization of the Microsoft web stack since the first version of ASP.NET MVC through tutorials, articles, and tools. He writes about .NET and client-side technologies on his blog, *Dot Net Programming*, and in various online magazines. His company, Mvcct Team, implements and offers web applications, AI software, SAS products, tools, and services for web technologies associated with the Microsoft stack. He has moved from AI systems, where he implemented one of the first decision support systems for banks and financial institutions, to the video games arena, with top-10 titles such as Puma Street Soccer.

To my beloved parents, to whom I owe everything.

About the reviewers

Efraim Kyriakidis has almost 20 years of experience in software development. He got his diploma as an electrical and software engineer from Aristotle University of Thessaloniki in Greece. He has used .NET since its beginnings with version 1.0. In his career, he has mainly focused on Microsoft technologies. He is currently employed by Siemens AG in Germany as a senior software engineer.

Fabio Claudio Ferracchiati is a senior consultant and a senior analyst/developer who uses Microsoft technologies. He works for React Consulting. He is a Microsoft Certified Solution Developer for .NET, Microsoft Certified Application Developer for .NET, and Microsoft Certified Professional. He is also a prolific author and technical reviewer. Over the last 10 years, he's written articles for Italian and international magazines and has co-authored more than 10 books on a variety of computer topics.

Table of Contents

Preface

This book covers the most common design patterns and frameworks involved in software architecture. It discusses when and how to use each pattern by providing you with practical real-world scenarios. This book also presents techniques and processes such as DevOps, microservices, continuous integration, and cloud computing so that you can have a best-in-class software solution developed and delivered for your customers.

This book will help you to understand the product that your customer wants from you. It will guide you to deliver and solve the biggest problems you could face during development. It also covers the do's and don'ts that you need to follow when you manage your application in a cloud-based environment. You will learn about different architectural approaches, such as layered architectures, service-oriented architecture, microservices, and cloud architecture, and understand how to apply them to specific business requirements. Finally, you will deploy code in remote environments or on the cloud using Azure.

All the concepts in this book will be explained with the help of real-world practical use cases where design principles make the difference when creating safe and robust applications. By the end of the book, you will be able to develop and deliver highly scalable enterprise-ready applications that meet the end customers' business needs.

It is worth mentioning that this book will not only cover the best practices that a software architect should follow for developing C# and .NET Core solutions, but it will also discuss all the environments that we need to master in order to develop a software product according to the latest trends.

Who this book is for

This book is for engineers and senior developers who are aspiring to become architects or wish to build enterprise applications with the .NET stack. Experience with C# and .NET is required.

What this book covers

`Chapter 1`, *Understanding the Importance of Software Architecture*, explains the basics of software architecture. This chapter will give you the right mindset to face customer requirements, and then select the right tools, patterns, and frameworks.

`Chapter 2`, *Functional and Nonfunctional Requirements*, guides you in the first stage of application development, that is, collecting user requirements and accounting for all other constraints and goals that the application must fulfill.

`Chapter 3`, *Documenting Requirements with Azure DevOps*, describes techniques for documenting requirements, bugs, and other information about your applications. While most of the concepts are general, the chapter focuses on the usage of Azure DevOps.

`Chapter 4`, *Deciding the Best Cloud-Based Solution*, gives you a wide overview of the tools and resources available in the cloud, and in particular on Microsoft Azure. Here, you will learn how to search for the right tools and resources and how to configure them to fulfill your needs.

`Chapter 5`, *Applying a Microservice Architecture to Your Enterprise Application*, offers a broad overview of microservices and Docker containers. Here, you will learn how the microservices-based architecture takes advantage of all the opportunities offered by the cloud and you will see how to use microservices to achieve flexibility, high throughput, and reliability in the cloud. You will learn how to use containers and Docker to mix different technologies in your architecture as well as make your software platform-independent.

`Chapter 6`, *Interacting with Data in C# - Entity Framework Core*, explains in detail how your application can interact with various storage engines with the help of **Object-Relational Mappings** (**ORMs**) and Entity Framework Core 3.0.

`Chapter 7`, *How to Choose Your Data Storage in the Cloud*, describes the main storage engines available in the cloud and, in particular, in Microsoft Azure. Here, you will learn how to choose the best storage engines to achieve the read/write parallelism you need and how to configure them.

`Chapter 8`, *Working with Azure Functions*, describes the serverless model of computation and how to use it in the Azure cloud. Here, you will learn how to allocate cloud resources just when they are needed to run some computation, thus paying only for the actual computation time.

`Chapter 9`, *Design Patterns and .NET Core Implementation*, describes common software patterns with .NET Core 3 examples. Here, you will learn the importance of patterns and best practices for using them.

`Chapter 10`, *Understanding the Different Domains in a Software Solution*, describes the modern domain-driven design software production methodology, how to use it to face complex applications that require several knowledge domains, and how to use it to take advantage of cloud- and microservices-based architectures.

`Chapter 11`, *Implementing Code Reusability in C# 8*, describes patterns and best practices to maximize code reusability in your C# .NET Core applications.

`Chapter 12`, *Applying Service-Oriented Architectures with .NET Core*, describes service-oriented architecture, which enables you to expose the functionalities of your applications as endpoints on the web or on a private network so that users can interact with them through various types of clients. Here, you will learn how to implement service-oriented architecture endpoints with ASP.NET Core, and how to self-document them with existing OpenAPI packages.

`Chapter 13`, *Presenting ASP.NET Core MVC*, describes in detail the ASP.NET Core framework. Here, you will learn how to implement web applications based on the Model-View-Controller (**MVC**) pattern and how to organize them according to the prescriptions of domain-driven design, described in `Chapter 10`, *Understanding the Different Domains in a Software Solution*.

`Chapter 14`, *Best Practices in Coding C# 8*, describes best practices to be followed when developing .NET Core applications with C# 8.

`Chapter 15`, *Testing Your Code with Unit Test Cases and TDD*, describes how to test your applications. Here, you will learn how to test .NET Core applications with xUnit, and see how easily you can develop and maintain code that satisfies your specifications with the help of test-driven design.

`Chapter 16`, *Using Tools to Write Better Code*, describe metrics that evaluate the quality of your software and how to measure them with the help of all the tools included in Visual Studio.

`Chapter 17`, *Deploying Your Application with Azure DevOps*, describes how to automate the whole deployment process, from the creation of a new release in your source repository, through various testing and approval steps, to the final deployment of the application in the actual production environment. Here, you will learn how to use Azure Pipelines to automate the whole deployment process.

`Chapter 18`, *Understanding DevOps Principles*, describes the basics of the DevOps software development and maintenance methodology. Here, you will learn how to organize your application's continuous integration/continuous delivery cycle.

`Chapter 19`, *Challenges of Applying CI Scenarios in DevOps*, complements the description of DevOps with continuous integration scenarios.

`Chapter 20`, *Automation for Software Testing*, is dedicated to automatic acceptance tests – that is, tests that verify automatically whether a version of a whole application conforms with the agreed specifications. Here, you will learn how to simulate user operations with automation tools and how to use these tools together with xUnit to write your acceptance tests.

To get the most out of this book

Do not forget to have Visual Studio Community 2019 or higher installed.

Be sure that you understand C# .NET principles.

Download the example code files

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

You can download the code files by following these steps:

1. Log in or register at `www.packt.com`.
2. Select the **Support** tab.
3. Click on **Code Downloads**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

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/Hands-On-Software-Architecture-with-CSharp-8`. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Code in Action

You can see the code in action videos at `http://bit.ly/2Old2IG`.

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `https://static.packt-cdn.com/downloads/9781789800937_ColorImages.pdf`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "They are copied in the final string just once, when you call `sb.ToString()` to get the final result."

A block of code is set as follows:

```
[Fact]
public void Test1()
{
    var myInstanceToTest = new ClassToTest();
    Assert.Equal(5,     myInstanceToTest.MethodToTest(1));
}
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "In the **Solution Explorer**, you have the option to **Publish...** by right-clicking."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packtpub.com/support/errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packt.com`.

1
Section 1: Transforming Customer Needs in Real-World Applications

This section includes the first three chapters of the book. The idea is to make sure you understand how to transform user requirements into the actual architectural needs that are essential for project success, using the various architectural aspects and design considerations involved in developing enterprise applications with C# and .NET Core.

In `Chapter 1`, *Understanding the Importance of Software Architecture*, we will discuss the importance of software architecture and aspects related to .NET Core and C#. The chapter will also discuss the importance of analyzing software requirements and designing for principles such as scalability and robustness. No matter what software development cycle you decide to have in your project, analyzing requirements will help you adhere to the goal of the project. Without this, the success of your project is at risk. This chapter will present some use cases where a lack of understanding of the requirements gathered led to project failures. Besides this, in each case, we will provide practical advice that could help to protect your software from the scenarios presented.

Once you have understood the process of gathering system requirements, in `Chapter 2`, *Functional and Nonfunctional Requirements*, we will prompt you to think about the impacts that the requirements have on the architectural design. Scalability, performance, multithreading, interoperability, and other subjects will be discussed, both their theory and their practice.

To finish the first section, in `Chapter 3`, *Documenting Requirements with Azure DevOps*, we will present Azure DevOps, which is the tool currently being provided by Microsoft to enable an application development life cycle that follows the principles of the DevOps philosophy. There are a variety of good features that can help you document and organize your software, and the purpose of the chapter is to present an overview of those features.

This section includes the following chapters:

- `Chapter 1`, *Understanding the Importance of Software Architecture*
- `Chapter 2`, *Functional and Nonfunctional Requirements*
- `Chapter 3`, *Documenting Requirements with Azure DevOps*

1 Understanding the Importance of Software Architecture

Nowadays, software architecture is one of the most discussed topics in the software industry, and for sure, its importance will grow more in the future. The more we build complex and fantastic solutions, the more we need great software architectures to maintain them. That is the reason why you decided to read this book. That is the reason why we decided to write it.

For sure, it is not an easy task to write about this important topic, which offers so many alternative techniques and solutions. The main objective of this book is not just to build an exhaustive and never-ending list of available techniques and solutions, but also to show how various families of techniques are related and how they impact, in practice, the construction of a maintainable and sustainable solution.

The attention on how to create actual efficacious enterprise solutions increases as users always need more new features in their applications. Moreover, the need to deliver frequent application versions (due to a quickly changing market) increases the obligation to have sophisticated software architecture and development techniques.

The following topics will be covered in this chapter:

- The history of software development and the definition of software architecture
- Software processes currently used by success enterprises
- The process for gathering requirements

By the end of this chapter, you will be able to understand exactly what the mission of a software architecture is. You will also learn what Azure is and how to create your account in the platform. Besides considering this is an introductory chapter, you will get an overview of software processes, models, and other techniques that will enable you to conduct your team.

Technical requirements

This chapter will guide you on how to create an account in Azure, hence no code will be provided.

What is software architecture?

If you are reading this book today, you should thank the computer scientists who decided to consider software development as an engineering area. This happened in the last century and, more specifically, at the end of the sixties, when they proposed that the way we develop software is quite similar to the way we construct buildings. That is why we have the name **software architecture**. Like in the design of a building, the main goal of a software architect is to ensure that the software application is implemented well. But a good implementation requires the design of a great solution. Hence, in a professional development project, you have to do the following things:

- Define the customer requirements for the solution.
- Design a great solution to meet those requirements.
- Implement the designed solution.
- Validate the solution with your customer.
- Deliver the solution in the working environment.

Software engineering defines these activities as the software development life cycle. All of the theoretical software development process models (waterfall, spiral, incremental, agile, and so on) are somehow related to this cycle. No matter which model you use, if you do not work with the essential tasks presented earlier during your project, you will not deliver acceptable software as a solution.

The main point about designing great solutions is totally connected to the purpose of this book. You have to understand that great real-world solutions bring with them a few fundamental constraints:

- The solution needs to meet user requirements.
- The solution needs to be delivered on time.
- The solution needs to adhere to the project budget.
- The solution needs to deliver good quality.
- The solution needs to guarantee a safe and efficacious future evolution.

Great solutions need to be sustainable and you have to understand that there is no sustainable software without great software architecture. Nowadays, great software architectures depend on both tools and environments to perfectly fit users' requirements. To explain this, this book will use some great tools provided by Microsoft:

- **Azure**: This is the cloud platform from Microsoft, where you will find all of the components it provides to build advanced software architecture solutions.
- **Azure DevOps**: This is the application life cycle management environment where you can build solutions using the latest approach for developing software, that is, DevOps.
- **C#**: This is one of the most used programming languages in the world. C# runs on small devices up to huge servers in different operating systems and environments.
- **.NET Core**: This is an open source development platform that is maintained by the Microsoft and .NET community on GitHub.
- **ASP.NET Core**: This is an open source multi-platform environment developed using .NET Core to build web applications and is hosted in the cloud or even on standard servers (on-premises).

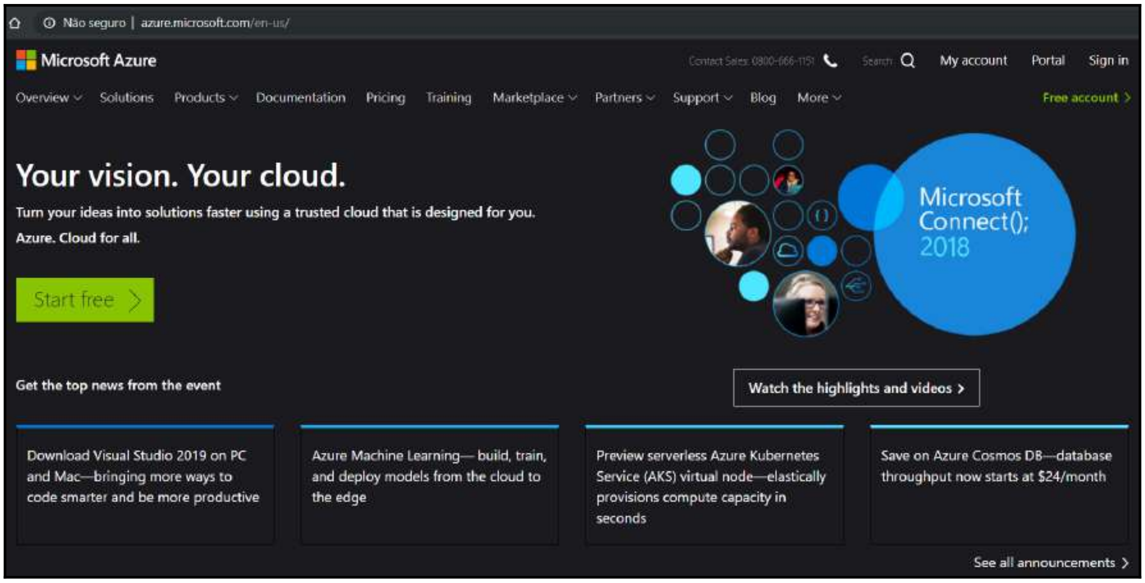
Being a software architect means understanding the aforementioned and a lot of other technologies. This book will guide you on a journey where you, as a software architect working in a team, will provide optimal solutions with the tools listed. Let's start this journey by creating your Azure account.

Creating an Azure account

Microsoft Azure is one of the best cloud solutions currently available on the market. It is important to know that, inside Azure, we will find a bunch of components that can help us in the architecture of twenty-first century solutions.

This subsection will guide you in creating an Azure account. If you already have one, you can skip this part:

1. You can access the Azure portal using this URL: `https://azure.microsoft.com`. Here, you will find a website, as follows. The translation to your native language will probably be set automatically:



Your vision. Your cloud.

Turn your ideas into solutions faster using a trusted cloud that is designed for you.

Azure. Cloud for all.

Get the top news from the event

Download Visual Studio 2019 on PC and Mac—bringing more ways to code smarter and be more productive

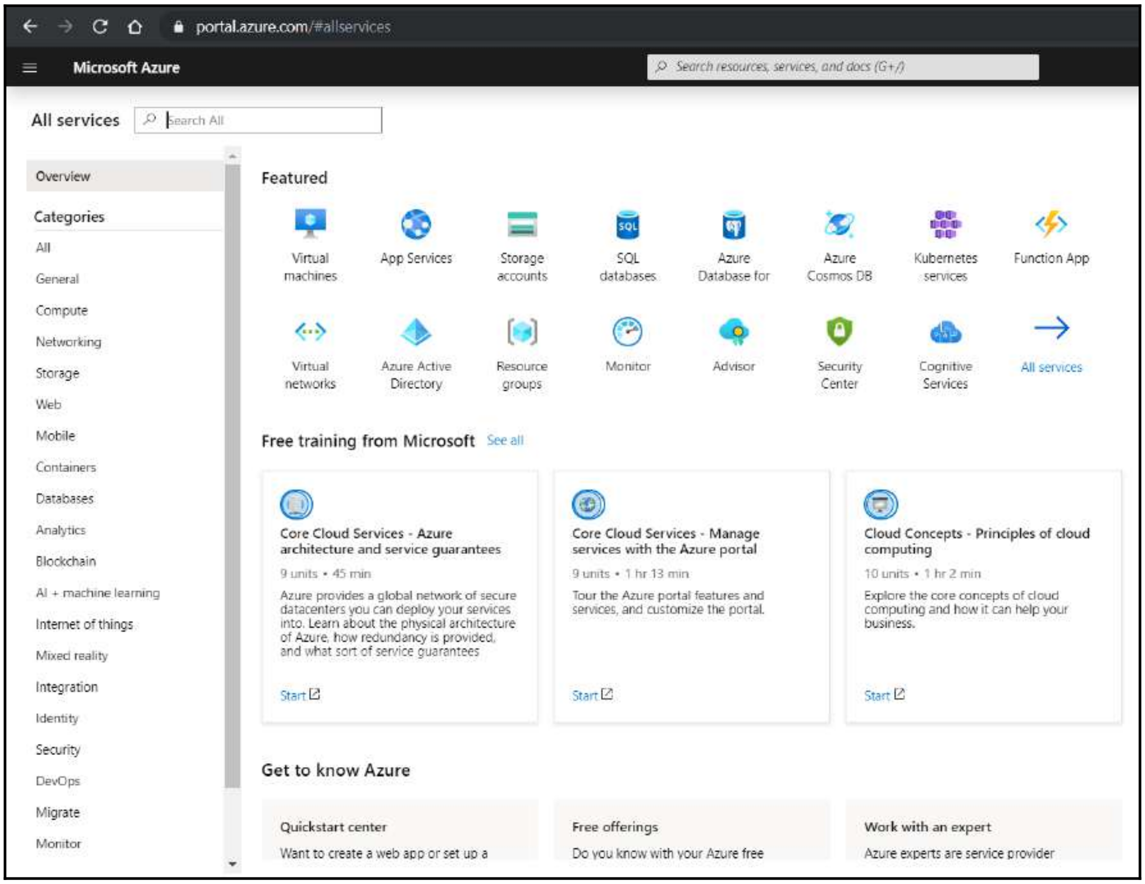
Azure Machine Learning— build, train, and deploy models from the cloud to the edge

Preview serverless Azure Kubernetes Service (AKS) virtual node—elastically provisions compute capacity in seconds

Save on Azure Cosmos DB—database throughput now starts at $24/month

2. Once you have accessed this portal, it is possible to sign up. If you have never done this before, it is possible to sign up for free, so you will be able to use some Azure features without spending any money.

3. Once you finish the form, you will be able to access the Azure panel. As you can see in the following screenshot, the panel shows a dashboard that you can customize, and a menu on the left, where you can set up the Azure components you are going to use in your solution. Throughout this book, we will come back to this screenshot to set up the components that create great opportunities for modern software architecture:



Core Cloud Services - Azure architecture and service guarantees

9 units • 45 min

Azure provides a global network of secure datacenters you can deploy your services into. Learn about the physical architecture of Azure, how redundancy is provided, and what sort of service guarantees

Core Cloud Services - Manage services with the Azure portal

9 units • 1 hr 13 min

Tour the Azure portal features and services, and customize the portal.

Cloud Concepts - Principles of cloud computing

10 units • 1 hr 2 min

Explore the core concepts of cloud computing and how it can help your business.

Want to create a web app or set up a

Do you know with your Azure free

Azure experts are service provider

Once you have your Azure account created, you are ready to understand how a software architect can conduct a team to develop software taking advantage of all of the opportunities offered by Azure. However, it is important to keep in mind that a software architect needs to understand something more than specific technologies because, nowadays, this role is played by people who are expected to define how the software will be delivered. A software architect not only architects the base of software, but they also determine how the whole software development and deployment process is conducted.

Software development process models

As a software architect, it is really important for you to understand some of the common development processes that are currently used in most enterprises. A software development process defines how people in a team produce and deliver software. In general, this process is connected with a software engineering theory, called **software development process models**. From the time software development was defined as an engineering process, many process models for developing software have been proposed. Let's take a look at the ones that are currently common.

Reviewing traditional software development process models

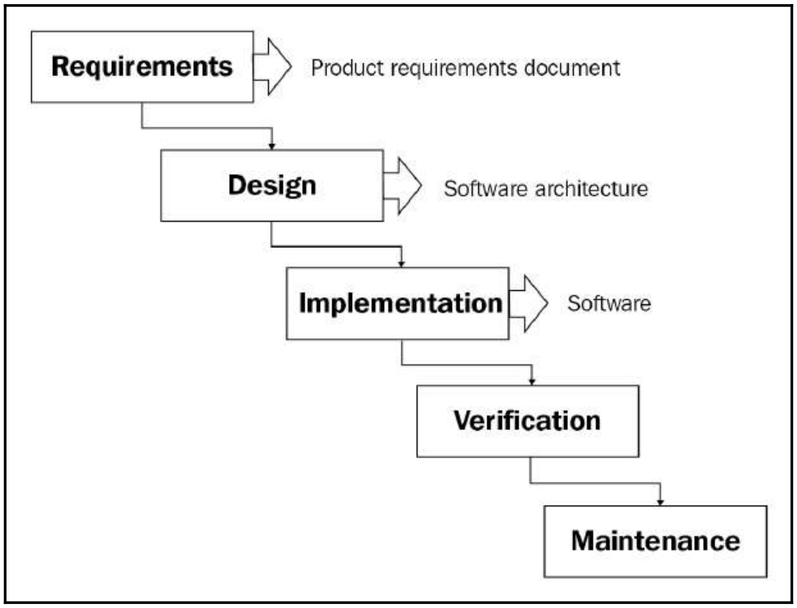
Some of the models introduced in the software engineering theory are already considered traditional and quite obsolete. This book does not aim to cover all of them, but here, we will give a brief explanation of the ones that are still used in some companies.

Understanding the waterfall model principles

This topic may appear strange in a software architecture book of 2019, but yes, you may still find companies where the most traditional software process model still remains the guideline for software development. This process executes all fundamental tasks in sequence. Any software development project consists of the following steps:

- Requirements specification
- Software design
- Programming
- Tests and delivery

Let's look at a diagrammatic representation of this:



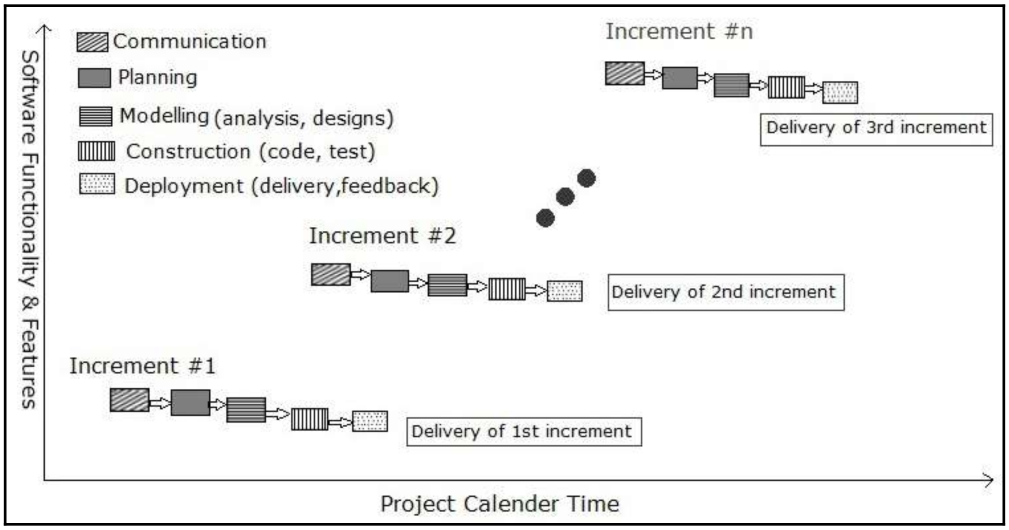
The waterfall development cycle (https://en.wikipedia.org/wiki/Waterfall_model)

Often, the use of waterfall models causes problems related to delays in the delivery of a functional version of the software and user dissatisfaction due to the poor quality of the final product.

Analyzing the incremental model

Incremental development is an approach that tries to overcome the biggest problem of the waterfall model: the user can test the solution only at the end of the project. The idea of this model is to give the users opportunities to interact with the solution as early as possible so that they can give useful feedback, which will help during the development of the software.

However, also in this model, the limited number of increments and the project's bureaucracy can causes problems in the interaction between developers and customers:


Communication
Planning
Modelling (analysis, designs)
Construction (code, test)
Deployment (delivery,feedback)
Increment #1
Delivery of 1st increment
Increment #2
Delivery of 2nd increment
Increment #n
Delivery of 3rd increment
Software Functionality & Features
Project Calender Time

The incremental development cycle (https://en.wikipedia.org/wiki/Incremental_build_model)

The incremental model was introduced as an alternative to the waterfall approach and it mitigated the problems related to the lack of communication with the customer. For big projects, fewer increments is still a problem. Besides, at the time the incremental approach was used on a large scale, mainly at the end of the last century, many problems related to project bureaucracy were reported, due to the large amount of documentation required. This scenario caused the rise of a very important movement in the software development industry—**agile**.

Understanding agile software development process models

At the beginning of this century, developing software was considered one of the most chaotic activities in engineering. The number of software projects that failed was incredibly high and this fact proved the need for a different approach to deal with the flexibility required by software development projects. So, in 2001, the Agile Manifesto was introduced to the world and, from that time, various agile process models were proposed. Some of them have survived up till now and are still very common.

Please check out this link for the Agile Manifesto: `https://agilemanifesto.org/`.

One of the biggest differences between agile models and traditional models is the way developers interact with the customer. The message that all agile models transmit is that the faster you deliver software to the user, the better. This idea is sometimes confusing for software developers who understand this as—*let's try coding and that's all folks!* However, there is an important observation of the Agile Manifesto that many people do not read when they start working with agile:

"That is, while there is value in the items on the right, we value the items on the left more."

– Agile Manifesto, 2001

A software architect always needs to remember this. Agile processes do not mean a lack of discipline. Moreover, when you use the agile process, you understand that there is no way to have good software developed without discipline. On the other hand, as a software architect, you need to understand that *soft* means flexibility. A software project that does not deal with flexibility tends to get ruined over time.

Getting into the Scrum model

Scrum is an agile model for the management of software development projects. The model comes from lean principles and is definitely one of the widely used approaches for developing software nowadays.

Please check this link for more information about the Scrum framework: `https://docplayer.net/78853722-Scrum-insights-for-practitioners.html`.

The basis of Scrum is that you have a flexible backlog of user needs that needs to be discussed in each agile cycle, called a **Sprint**. The Sprint Goal is determined by the Scrum Team, composed by the Product Owner, the Scrum Master, and the Development Team. The **Product Owner** is responsible for prioritizing what will be delivered in that sprint. During the sprint, this person will help the team to develop the required features. The person who leads the team in the Scrum process is called **Scrum Master**. All of the meetings and processes are conducted by this person.

It is important to notice that the Scrum process does not discuss how the software needs to be implemented and which activities will be done. So, again, you have to remember the software development basis, discussed at the beginning of this chapter. That means Scrum needs to be implemented together with a process model. DevOps is one of the approaches that may help you with the use of a software development process model together with Scrum. We will discuss this later in this book, in `Chapter 18`, *Understanding DevOps Principles*.

Enabling aspects to be gathered to design high-quality software

Fantastic! You just started a software development project. Now, it is time to use all of your knowledge to deliver the best software you can. Probably, your next question is—*how do I start?* Well, as a software architect, you are going to be the one to answer it. And be sure your answer is going to evolve in each software project you lead:

1. Defining a software development process is obviously the first thing to do. This is generally done during the project planning process.
2. Besides, another very important thing to do is to gather the software requirements. No matter which software development process you decide to use, collecting real user needs is a part of a very difficult and continuous job. Of course, there are techniques to help you with this. And be sure that gathering requirements will help you to detect important aspects of software architecture.

These two activities are considered by most experts in software development as the key to having success at the end of the development project journey. As a software architect, you need to enable them to happen so that you will not have problems while guiding your team.

Understanding the requirements gathering process

There are different ways to represent the requirements. The most traditional approach consists of you having to write a perfect specification before the beginning of the analysis. Agile methods suggest that you need to write stories as soon as you are ready to start a development cycle.

TIP

Remember: you do not write requirements for the user, you write them for you and your team. The user just needs the job done!

The truth is that no matter the approach you decide to adopt in your projects, you will have to follow some steps to gather requirements. This is what we call **requirements engineering.**

Please check out this image of the requirements engineering process for more information: `https://www.slideshare.net/MohammedRomi/ian-sommerville-software-engineering-9th-edition-ch-4`.

During this process, you need to be sure that the solution is feasible. In some cases, the feasibility analysis is a part of the project planning process too, and by the time you start the requirements elicitation, you will have the feasibility report already done. So, let's check the other parts of this process, which will give you a lot of important information for the software architecture.

Practicing the elicitation of user needs

There are a lot of ways to detect what exactly the user needs for a specific scenario. In general, this can be done using techniques that will help you to understand what we call user requirements. Here, you have a list of common techniques:

- **The power of imagination**: If you are an expert in the area where you are providing solutions, you may use your own imagination to find new user requirements. Brainstorming can be conducted together so that a group of experts can define user needs.
- **Questionnaires**: This tool is useful for detecting common and important requirements such as the number and kind of users, peak system usage, and the commonly-used **operating system** (**OS**) and web browser.
- **Interviews**: Interviewing the users helps you as an architect to detect user requirements that perhaps questionnaires and your imagination will not cover.
- **Observation**: There is no better way to understand the daily routine of a user than being with them for a day.

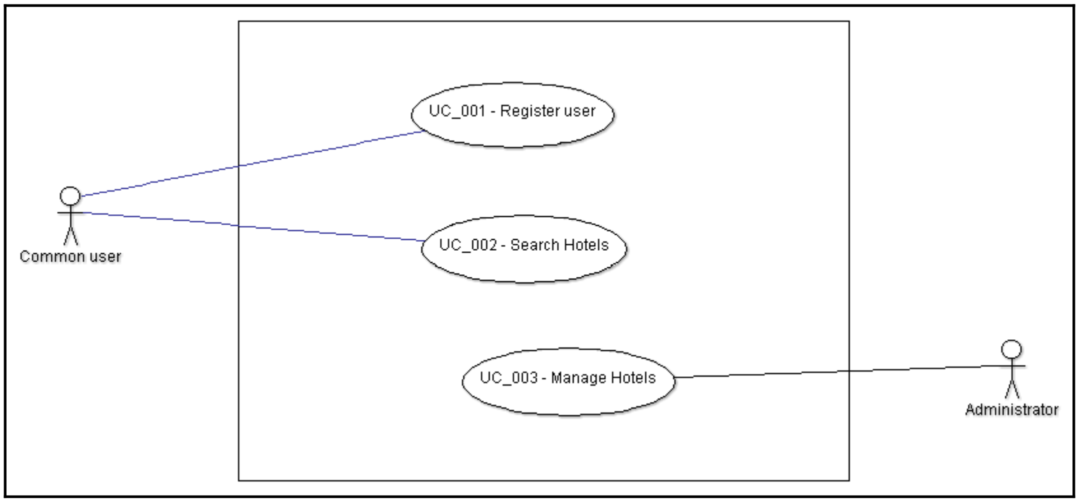
As soon as you apply one or more of these techniques, you will have great and valuable information, that is, the user's needs. At that moment, you will be able to analyze them and detect the user and system requirements.

Remember: You can use these techniques in any situation where the real need is to gather requirements, no matter if it is for the whole system or for a single story.

Analyzing requirements

As soon as you detect user needs, it is time to begin the analysis of the requirements. At that time, you can use techniques such as the following:

- **Prototyping**: Prototypes are really good to clarify and to materialize the system requirements. Today, we have many tools that can help you to mock interfaces. A really nice open source tool is the **Pencil Project**. You will find further information about it at `https://pencil.evolus.vn/`.
- **Use cases**: The **Unified Modeling Language** (**UML**) use case model is an option if you need detailed documentation. The model is composed of a detailed specification and a diagram. **Argo UML** is another open source tool that can help you out with this:



While you are analyzing the requirements of the system, you will be able to clarify exactly what the users' needs are. This is really helpful when you are not sure about the real problem you will solve and is pretty much better than just starting to program the system. It is time that you will invest in having better code in the near future.

Writing the specifications

After you finish the analysis, it is important to register it as a specification. This document can be written using traditional requirements or user stories, which are commonly used in agile projects.

Requirements specification represents the technical contract between the user and the team. There are some basic rules that this document needs to follow:

- All stakeholders need to understand exactly what is written in the technical contract, even if they are not technicians.
- The document needs to be clear.
- You need to classify each requirement.
- Use a simple feature to represent each requirement.
- Ambiguity and controversy need to be avoided.

Besides, some information can help the team to understand the context of the project they are going to work on. Here, you have some tips about it:

- Write an introductory chapter to give a full idea of the solution.
- Create a glossary to make understanding easier.
- Describe the kind of user the solution will cover.
- Write functional and non-functional requirements.
- Attach documents that can help the user to understand rules.

If you decide to write user stories, a good tip to follow is to write short sentences representing each moment in the system with each user, as follows:

```
As <user>, I want <feature>, so that <reason>
```

This approach will explain exactly the reason why that feature will be implemented. Besides that, you will have a good tool to later analyze the stories that are more critical and prioritize the success of the project.

Reviewing the specification

Once you have the specification written, it is time to confirm with the stakeholders whether they agree with it. This can be conducted in a review meeting or can be done online using collaboration tools.

This is when you present all of the prototypes, documents, and information you have gathered. As soon as everybody agrees with the specification, you are ready to start studying the best way to implement this part of your project.

Using design thinking as a helpful tool

During your career as a software architect, you will find many projects where your customer will bring you a solution *ready for development*. This is quite complicated once you consider that as the correct solution and, most of the time, there will be architectural and functional mistakes that will cause problems in the solution in the future. There are some cases where the problem is worse—when the customer does not know the best solution for the problem. Design thinking can help us with this.

Design thinking is a process that allows you to collect data directly from the users, focusing on achieving the best results to solve a problem. During this process, the team will have the opportunity to discover all *personas* that will interact with the system. This will have a wonderful impact on the solution since you can develop the software by focusing on the user experience, which can have a fantastic impact on the results.

The process is based on the following steps:

- **Empathize**: In this step, you have to execute field research to discover the user's concerns. This is where you find out about the users of the system. The process is good for making you understand why and for whom you are developing this software.
- **Define**: Once you have the users' concerns, it is time to define their needs to solve them.
- **Ideate:** The needs will provide an opportunity to brainstorm some possible solutions.
- **Prototype**: These solutions can be developed as prototypes to confirm whether they are good ones.
- **Test**: Testing the prototypes will help you to understand the prototype that is most connected to the real needs of the users.

What you have to understand is that design thinking can be a fantastic option to discover real requirements. As a software architect, you are committed to helping your team to use the correct tools at the correct time.

Understanding the principles of scalability, robustness, security, and performance

Detecting requirements is a task that will let you understand the software you are going to develop. However, as a software architect, you don't have to only pay attention to the functional requirements for that system. Understanding the non-functional requirements is really important and one of the primordial activities for a software architect.

We are going to discuss this more in `Chapter 2`, *Functional and Nonfunctional Requirements*, but at this point, it is good to know that the principles of scalability, robustness, security, and performance need to be applied for the requirements gathering process. Let's take a look at each concept:

- **Scalability**: As a software developer, globalization gives you the opportunity to have your solution running all over the world. This is fantastic, but you, as a software architect, need to design a solution that provides that possibility. Scalability is the possibility for an application to increase its processing power as soon as it is necessary, due to the number of resources that are being consumed.
- **Robustness**: No matter how scalable your application is, if it is not able to guarantee a stable and always-on solution, you are not going to get any peace. Robustness is really important for critical solutions, where you do not have the opportunity for maintenance at any time, due to the kind of problem that the application solves. In many industries, the software cannot stop and lots of routines run when nobody is available (overnight, holidays, and so on). Designing a robust solution will give you the freedom to live while your software is running well.
- **Security**: This is another really important area that needs to be discussed after the requirements stage. Everybody is worried about security and laws dealing with it are being proposed in different parts of the world. You, as a software architect, have to understand that security needs to be provided by design. This is the only way to cope with all of the needs that the security community is discussing right now.
- **Performance**: The process of understanding the system you are going to develop will probably give you a good idea of what your efforts will need to be to get the desired performance from the system. This topic needs to be discussed with the user to identify most of the bottlenecks you will face during the development stage.

It is worth mentioning that all these concepts are requirements for this new generation of solutions that the world needs. What will differentiate good software for incredible software surely is the amount of work done to meet the project requirements.

Some cases where the requirements gathering process impacted system results

All of the information discussed up to this point in the chapter is useful if you want to design software following the principles of good engineering. This discussion is not related to developing by using traditional or agile methods but focuses on building software professionally or as an amateur.

Besides, it is good to know about some cases where the lack of activities you read about caused some trouble for the software project. The following cases intend to describe what went wrong and how the preceding techniques could have helped the development team to solve the problems. In most cases, simple action could guarantee better communication between the team and the customer and this easy communication flow could transform a big problem into a real solution.

Case 1 – my website is too slow to open that page!

Performance is one of the biggest problems that you as a software architect will live through during your career. The reason why this aspect of any software is so problematic is that we do not have infinite computational resources to solve problems. Besides, the cost of computation is still high, especially if you are talking about software with a high number of simultaneous users.

You cannot solve performance problems by writing requirements. However, you won't end up in trouble if you write them correctly. The idea here is that requirements have to present the desired performance of a system. A simple sentence, describing this, can help the entire team that works on the project:

> *Non-functional requirement: Performance – any web page of this software will respond in at least 2 seconds.*

The preceding sentence just makes everybody (users, testers, developers, architects, managers, and so on) sure that any web page has a target to achieve. This is a good start, but it is not enough. With this, a great environment to both develop and deploy your application is important. This is where .NET Core can help you a lot. Especially if you are talking about web apps, ASP.NET Core is considered one of the fastest options to deliver solutions today.

If you talk about performance, you, as a software architect, should consider the use of the techniques listed in the following sections. It is good to mention that ASP.NET Core will help you to use them easily, together with some **Platform as a Service** (**PaaS**) solutions delivered by Microsoft Azure.

Understanding caching

Caching is a great technique to avoid queries that can consume time and, in general, give the same result. For instance, if you are fetching the available car models in a database, the number of cars in the database can increase but they will not change. Once you have an application that constantly accesses car models, a good practice is to cache that information.

It is important to understand that a cache is stored in the backend and that cache is shared by the whole application (*in-memory caching*). A single point of attention here is when you are working on a scalable solution, you can configure a *distributed cache* to solve it using the Azure platform. In fact, ASP.NET Core provides both of them, so you can decide on the one that bests fits your needs.

Applying asynchronous programming

When you develop ASP.NET Core applications, you need to keep in mind that this app needs to be designed for simultaneous access by many users. Asynchronous programming lets you do this simply, giving you the keywords `async` and `await`.

The basic concept behind these keywords is that `async` enables any method to run in a different thread from the one that calls it. On the other hand, `await` lets you synchronize the call of an asynchronous method without blocking the thread that is calling it. This easy-to-develop pattern will make your application run without performance bottlenecks and better responsiveness. This book will cover more about this subject in `Chapter 2`, *Functional and Nonfunctional Requirements*.

Dealing with object allocation

One very good tip to avoid a lack of performance is to understand how the Garbage Collector works. The Garbage Collector is the engine that will free memory automatically when you finish using it. There are some very important aspects of this topic, due to the complexity that the GC has.

Some types of objects are not collected by the GC. The list includes any object that interacts with I/O, such as files and streaming. If you do not correctly use the C# syntax to create and destroy this kind of object, you will have memory leaks, which will deteriorate your application performance.

The incorrect way of working with I/O objects:

```
System.IO.StreamWriter file = new System.IO.StreamWriter(@"C:\sample.txt");
file.WriteLine("Just writing a simple line");
```

The correct way of working with I/O objects:

```
using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:\sample.txt"))
{
    file.WriteLine("Just writing a simple line");
}
```

Even though the preceding practice is mandatory for I/O objects, it is totally recommended that you keep doing this in all disposable objects. This will help the GC and will keep your application running with the right amount of memory.

Another important aspect that you need to know about is that the time spent by the GC to collect objects that will interfere with the performance of your app. Because of this, avoid allocating large objects. This can cause you trouble waiting for the GC to finish its task.

Getting better database access

One of the most common performance Achilles' heel is database access. The reason why this is still a big problem is the lack of attention while writing queries or lambda expressions to get information from the database. This book will cover Entity Framework Core in `Chapter 6`, *Interacting with Data in C# – Entity Framework Core*, but it is important to know what to choose, the correct data information to read from a database, and filtering columns and lines is imperative for an application that wants to deliver performance.

The good thing is that best practices related to caching, asynchronous programming, and object allocation fit completely in the environment of databases. It is only a matter of choosing the correct pattern to get better-performance software.

Case 2 – the user's needs are not properly implemented

The more technology is used in a wide variety of areas, the more difficult it is to deliver exactly what the user needs. Maybe this sentence sounds weird to you, but you have to understand that developers, in general, study to develop software, but they rarely study to deliver the needs of a specific area. Of course, it is not easy to learn how to develop software, but it is even more difficult to understand a need in a particular area. Software development nowadays delivers software to all possible types of industries. The question here is *how can a developer, being a software architect or not, evolve enough to deliver software in the area they are responsible for?*

Gathering software requirements definitely will help you in this tough task. Moreover, writing them will make you understand and organize the architecture of the system. There are several ways to minimize the risks of implementing something different from what the user really needs:

- Prototyping the interface to achieve the understanding of the user interface faster
- Designing the data flow to detect gaps between the system and the user operation
- Frequent meetings to be updated on the current needs and aligned to the incremental deliveries

Again, as a software architect, you will have to define how the software will be implemented. Most of the time, you are not going to be the one who programs it, but you will always be the one responsible for this. For this reason, some techniques can be useful to avoid the wrong implementation:

- Requirements are reviewed with the developers to guarantee that they understand what they need to develop
- Code inspection to validate a predefined code standard
- Meetings to eliminate impediments

Case 3 – the usability of the system does not meet user needs

Usability is a key point for the success of a software project. The way the software is presented and how it solves a problem can help the user to decide whether they want to use it or not. As a software architect, you have to keep in mind that delivering software with good usability is mandatory nowadays.

There are basic concepts of usability that this book does not intend to cover. But a good way to meet the correct user needs when it comes to usability is by understanding who is going to use the software. Design thinking can help you a lot with that, as was discussed earlier in this chapter.

Understanding the user will help you to decide whether the software is going to run on a web page, or a cell phone, or even in the background. This understanding is very important to a software architect because the elements of a system will be better presented if you correctly map who will use them.

On the other hand, if you do not care about that, you will just deliver software that works. This can be good for a short time, but it will not exactly meet the real needs that made a person ask you to architect software. You have to keep in mind the options and understand that good software is designed to run on many platforms and devices.

You will be happy to know that C# is an incredible cross-platform option for that. So, you can develop solutions to run your apps in Linux, Windows, Android, and iOS. You can run your applications on big screens, tablets, cell phones, and even drones! You can embed apps on boards for automation or in HoloLens for mixed reality. Software architects have to be open-minded to design exactly what their users need.

Case study – detecting user needs

The case study of this book will take you on a journey of creating the software architecture for a travel agency called **World Wild Travel Club** (**WWTravelClub**). The purpose of this case study is to make you understand the theory explained in each chapter, plus to provide the during the process of reading this book to develop an enterprise application with Azure, Azure DevOps, C#, .NET Core, ASP.NET Core, and other technologies that will be introduced in this book.

Book case study – introducing World Wild Travel Club

World Wild Travel Club (**WWTravelClub**) is a travel agency that was created to change the way people make decisions about their vacations and other trips around the world. To do so, they are developing an online service where every detail of a trip experience will be assisted by a club of experts specifically selected for each destination.

The concept of this platform is that you can be both a visitor and a destination expert at the same time. The more you participate as an expert in a destination, the higher the points you will score. These points can be exchanged for tickets that people buy online using the platform.

The customer came with the following requirements for the platform. It is important to know that, in general, customers do not bring the requirements ready for development. That is why the requirements gathering process is so important:

- Common user view:
    - Promotional packages on the home page
    - Search for packages
    - Details for each package:
        - Buy a package
        - Buy a package with a club of experts included:
            - Comment on your experience
            - Ask an expert
            - Evaluate an expert
    - Register as a common user
- Destination expert view:
    - The same view as the common user view
    - Answer the questions asking for your destination expertise
    - Manage the points you scored answering questions:
        - Exchange points for tickets
- Administrator view:
    - Manage packages
    - Manage common users
    - Manage destination experts

To finish this, it is important to note that WWTravelClub intends to have more than 100 Destination Experts per package and will offer around 1,000 different packages all over the world.

Book case study – understanding user needs and system requirements

To summarize the user needs of WWTravelClub, you can read the following user stories:

- `US_001`: As a common user, I want to view promotional packages on the home page, so that I can easily find my next vacation.
- `US_002`: As a common user, I want to search for packages I cannot find on the home page so that I can explore other trip opportunities.
- `US_003`: As a common user, I want to see the details of a package, so that I can decide which package to buy.
- `US_004`: As a common user, I want to register myself, so that I can start buying the package.
- `US_005`: As a registered user, I want to buy a package, so that I can process the payment.
- `US_006`: As a registered user, I want to buy a package with a club of experts included, so that I can have an exclusive trip experience.
- `US_007`: As a registered user, I want to ask for an expert, so that I can get the best of my trip.
- `US_008`: As a registered user, I want to comment on my experience, so that I can give feedback from my trip.
- `US_009`: As a registered user, I want to evaluate an expert who helps me, so that I can share with others how fantastic they were.
- `US_010`: As a registered user, I want to register as a Destination Expert View, so that I can help people who travel to my city.
- `US_011`: As an expert user, I want to answer questions about my city, so that I can score points to be exchanged in the future.
- `US_012`: As an expert user, I want to exchange points for tickets, so that I can travel around the world more.
- `US_013`: As an administrator user, I want to manage packages, so that users can have fantastic opportunities to travel.

- `US_014`: As an administrator user, I want to manage registered users, so that WWTravelClub can guarantee good service quality.
- `US_015`: As an administrator user, I want to manage expert users, so that all of the questions regarding our destinations are answered.
- `US_016`: As an administrator user, I want to offer more than 1,000 packages around the world, so that different countries can experience WWTravelClub service.
- `US_017`: As an administrator user, I want to have more than 1,000 users simultaneously accessing the website, so that I can support all of the needs of my users.
- `US_018`: As a user, I want to access WWTravelClub in my native language, so that I can easily understand the package offered.
- `US_019`: As a user, I want to access WWTravelClub in the Chrome, Firefox, and Edge web browsers, so that I can use the web browser of my preference.
- `US_020`: As a user, I want to buy packages safely, so that only WWTravelClub will have my credit card information.

Notice that while you start writing the stories, information related to non-functional requirements such as security, environment, performance, and scalability can be included.

However, some system requirements may be omitted when you write user stories and need to be included in the software specification. These requirements can be related to legal aspects, hardware and software prerequisites, or even points of attention for the correct system delivery. They need to be mapped and listed as well as user stories. The list of WWTravelClub system requirements is presented in the following. Notice that requirements are written in the future because the system does not exist yet:

- `SR_001`: The system will use Microsoft Azure components to deliver the scalability required.
- `SR_002`: The system will respect **General Data Protection Regulation** (**GDPR**) requirements.
- `SR_003`: The system will run on the Windows, Linux, iOS, and Android platforms.
- `SR_004`: Any web page of this system will respond in at least 2 seconds.

Summary

In this chapter, you learned the purpose of a software architect in a software development team. Also, this chapter covered the basics of software development process models and the requirements gathering process. You also had the opportunity to learn about how to create your Azure account, which will be used during the case study of this book, which was presented to you in the previous section. Moreover, you even learned about functional and non-functional requirements and how to create them using user stories. These techniques will surely help you deliver a better software project.

In the next chapter, you will have the opportunity to understand how functional and non-functional requirements are important for software architecture.

Questions

1. What is the expertise that a software architect needs to have?
2. How can Azure help a software architect?
3. How does a software architect decide the best software development process model to use in a project?
4. How does a software architect contribute to gathering requirements?
5. What kind of requirements does a software architect need to check in a requirement specification?
6. How does design thinking help a software architect in the process of gathering requirements?
7. How do user stories help a software architect in the process of writing requirements?
8. What are good techniques to develop very good performance software?
9. How does a software architect check whether a user requirement is correctly implemented?

Further reading

Here, you have some books and links you may consider reading to gather more information about this chapter:

- https://www.packtpub.com/virtualization-and-cloud/hands-azure-developers
- https://azure.microsoft.com/en-us/overview/what-is-azure/
- https://azure.microsoft.com/en-us/services/devops/
- https://docs.microsoft.com/en-us/dotnet/core/about
- https://docs.microsoft.com/en-us/aspnet/core/
- https://www.packtpub.com/web-development/hands-full-stack-web-development-aspnet-core
- https://agilemanifesto.org/
- https://www.amazon.com/Software-Engineering-10th-Ian-Sommerville/dp/0133943038
- https://www.amazon.com/Software-Engineering-Practitioners-Roger-Pressman/dp/0078022126/
- https://scrumguides.org/
- https://www.packtpub.com/application-development/professional-scrummasters-handbook
- https://docs.microsoft.com/en-us/aspnet/core/performance/performance-best-practices
- https://www.microsoft.com/en-us/hololens
- https://en.wikipedia.org/wiki/Incremental_build_model
- https://en.wikipedia.org/wiki/Waterfall_model

2
Functional and Nonfunctional Requirements

Once you have gathered the system requirements, it is time to think about the impact they have on the architectural design. Scalability, performance, multithreading, interoperability, and other subjects need to be analyzed so that we can meet user needs.

The following topics will be covered in this chapter:

- What is scalability and how does it interact with Azure and .NET Core?
- Good tips for writing better code when it comes to performance improvement
- Creating a safe and useful multithreading software
- Software usability, that is, how to design effective user interfaces
- .NET Core and interoperability

Technical requirements

The samples provided in this chapter will require Visual Studio 2019 Community Edition or Visual Studio Code.

You can find the sample code for this chapter here: `https://github.com/PacktPublishing/Hands-On-Software-Architecture-with-CSharp-8/tree/master/ch02`.

How does scalability interact with Azure and .NET Core?

A short search on scalability returns a definition such as *the ability of a system to keep working well when there's an increase in demand*. Once developers read this, many of them incorrectly conclude *that scalability only means add more hardware to keep things working without stopping the app*.

Scalability relies on technologies involving hardware solutions. However, as a software architect, you have to be aware that good software will keep scalability in a sustainable model, which means that a well-architected software can save a lot of money. Hence, it is not just a matter of hardware but also a matter of overall software design.

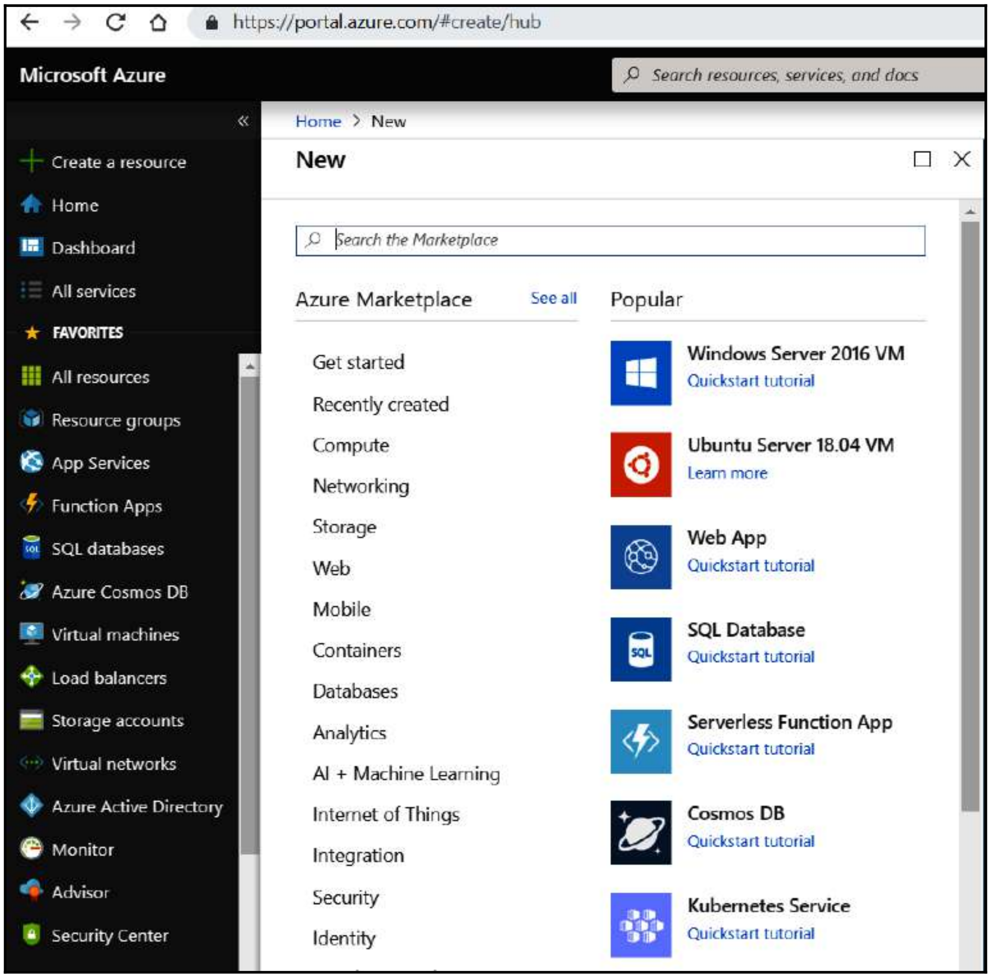
In `Chapter 1`, *Understanding the Importance of Software Architecture*, while discussing software performance, we proposed some good tips to overcome bad performance issues. The same tips will help you with scalability too. The fewer resources we spend on each process, the more users the application can handle.

It is worth knowing that Azure and .NET Core web apps can be configured to handle scalability too. Let's check this out in the following subsections.

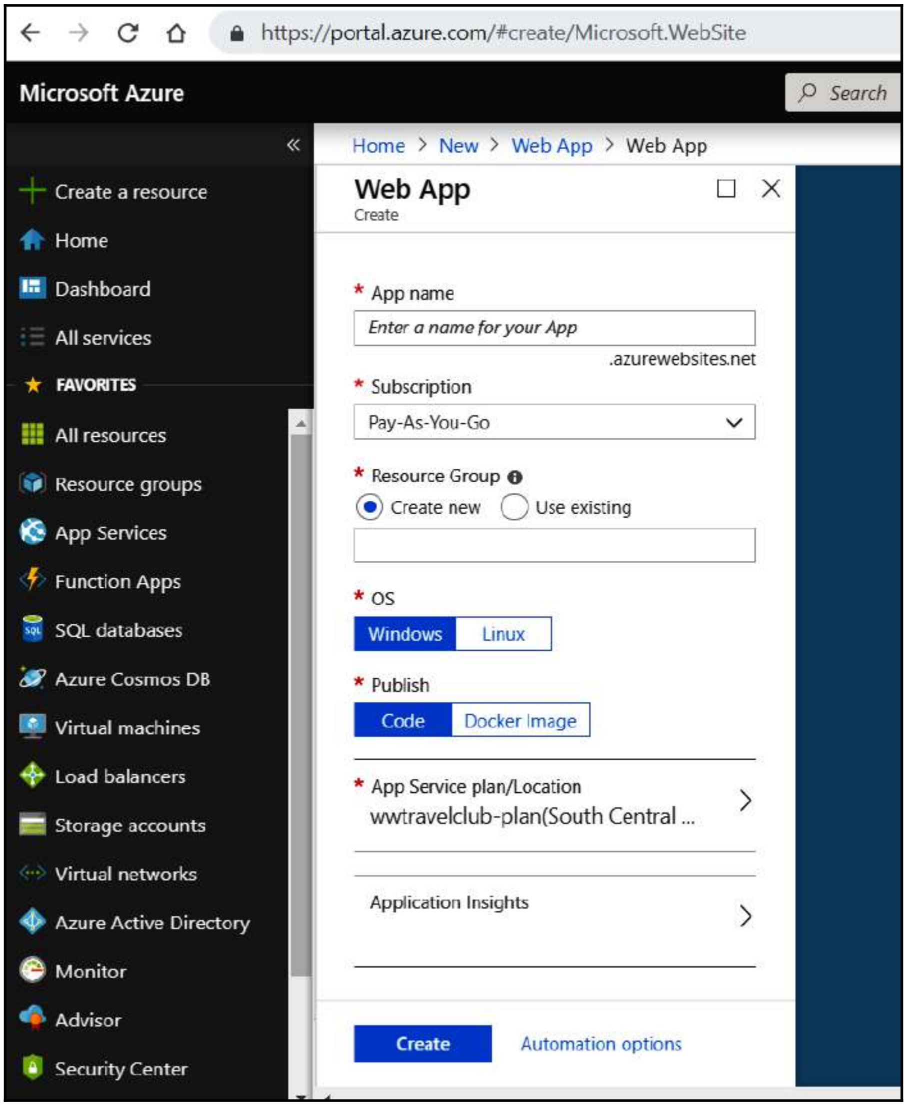
Creating a scalable web app in Azure

It is pretty simple to create a web app in Azure, ready for scaling. The reason why you have to do so is to be able to maintain different amounts of users during different seasons. The more users you have, the more hardware you will need. The following steps will show you how to create a scalable web application in Azure:

1. As soon as you log in to your Azure account, you will be able to create a new resource (web app, database, virtual machine, and so on), as you can see in the following screenshot:



2. After that, you can select **Web App**. This tutorial will take you to the following screen:



https://portal.azure.com/#create/Microsoft.WebSite

Microsoft Azure

Home > New > Web App > Web App

Web App
Create

* App name
Enter a name for your App
.azurewebsites.net

* Subscription
Pay-As-You-Go

* Resource Group
Create new [✓] Use existing

* OS
[✓ Windows] Linux

* Publish
[✓ Code] Docker Image

* App Service plan/Location
wwtravelclub-plan(South Central ...

Application Insights

Create Automation options

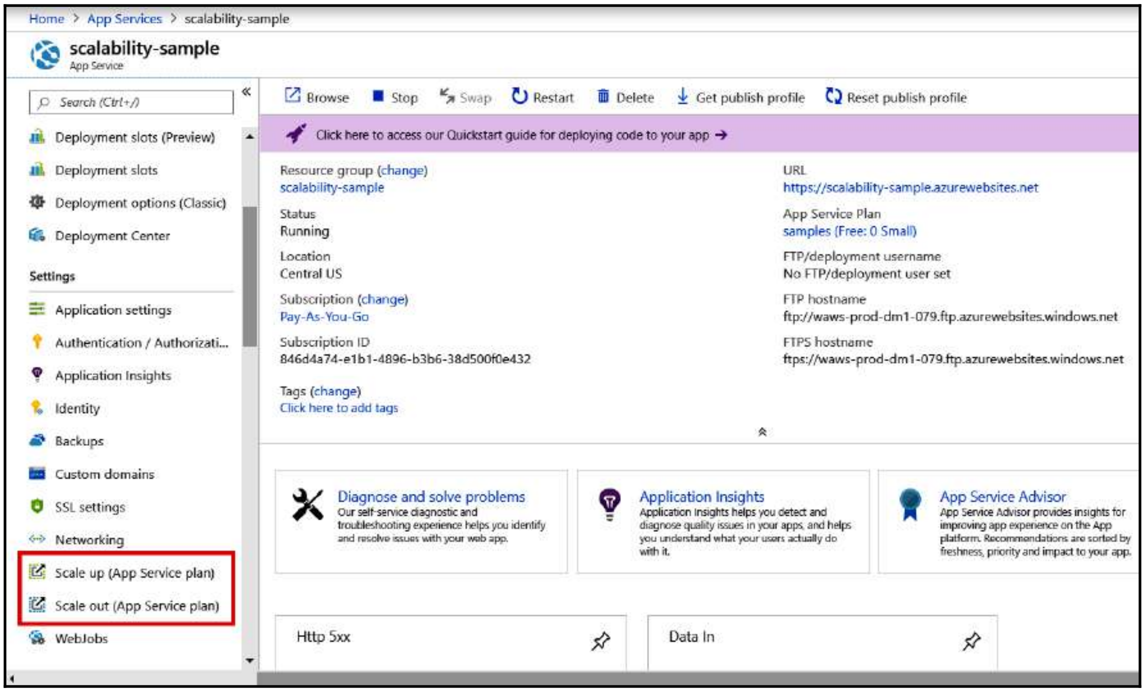
The required details are as follows:

- **App name**: As you can see, this is the URL that your web app will assume after its creation. The name is checked to ensure it is available.
- **Subscription**: This is the account that will be charged for all application costs.
- **Resource Group**: This is the collection of resources you can define to organize policies and permissions. You may specify a new resource group name or add the web app to a group specified during the definition of other resources.
- **OS**: This is the operating system that will host the web app. Both Windows and Linux may be used for ASP.NET Core projects.
- **Publish**: This parameter indicates whether the web app will be delivered directly or whether it is going to use Docker technology to publish content. Docker will be discussed in more detail in `Chapter 5`, *Applying a Microservice Architecture to Your Enterprise Application.*
- **App Service Plan/Location**: This is where you define the hardware plan that's used to handle the web app and the location of the servers. This choice defines application scalability, performance, and costs.
- **Application Insights**: This is a useful Azure toolset for monitoring and troubleshooting web apps.

Applications may be scaled in two conceptually different ways:

- Vertically (**Scale up**)
- Horizontally (**Scale out**)

Both of them are available in the web app settings, as you can see in the following screenshot:



Resource group (change)
scalability-sample

Status
Running

Location
Central US

Subscription (change)
Pay-As-You-Go

Subscription ID
846d4a74-e1b1-4896-b3b6-38d500f0e432

Tags (change)
Click here to add tags

URL
https://scalability-sample.azurewebsites.net

App Service Plan
samples (Free: 0 Small)

FTP/deployment username
No FTP/deployment user set

FTP hostname
ftp://waws-prod-dm1-079.ftp.azurewebsites.windows.net

FTPS hostname
ftps://waws-prod-dm1-079.ftp.azurewebsites.windows.net

Our self-service diagnostic and troubleshooting experience helps you identify and resolve issues with your web app.

Application Insights helps you detect and diagnose quality issues in your apps, and helps you understand what your users actually do with it.

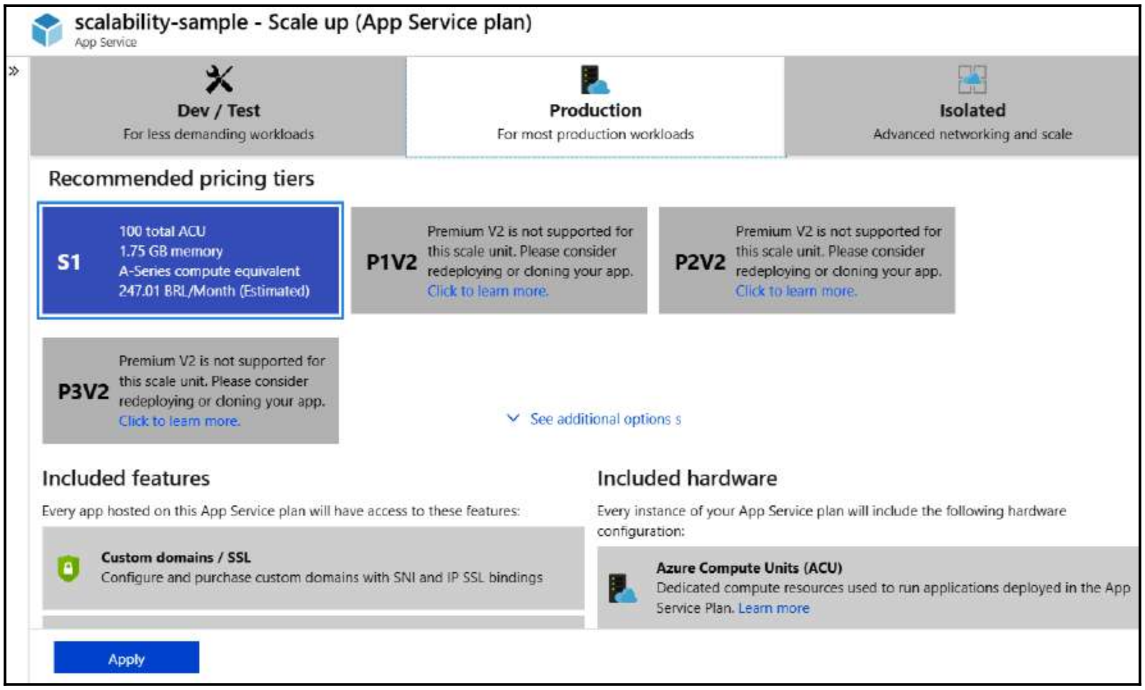
App Service Advisor provides insights for improving app experience on the App platform. Recommendations are sorted by freshness, priority and impact to your app.

Let's checkout the two types of scaling.

Vertical scaling (Scale up)

Scale up means changing the type of hardware that will sustain your application. In Azure, you have the opportunity of starting with free-shared hardware and moving to an isolated machine in a few clicks.

By selecting this option, you have the opportunity to select more powerful hardware (machines with more CPUs, storage, and RAM). The following screenshot shows the user interface for scaling up a web app:


scalability-sample - Scale up (App Service plan)
App Service

Dev / Test
For less demanding workloads
Production
For most production workloads
Isolated
Advanced networking and scale

Recommended pricing tiers

S1 100 total ACU
1.75 GB memory
A-Series compute equivalent
247.01 BRL/Month (Estimated)

P1V2 Premium V2 is not supported for this scale unit. Please consider redeploying or cloning your app. Click to learn more.

P2V2 Premium V2 is not supported for this scale unit. Please consider redeploying or cloning your app. Click to learn more.

P3V2 Premium V2 is not supported for this scale unit. Please consider redeploying or cloning your app. Click to learn more.

See additional options s

Included features

Every app hosted on this App Service plan will have access to these features:

Custom domains / SSL
Configure and purchase custom domains with SNI and IP SSL bindings

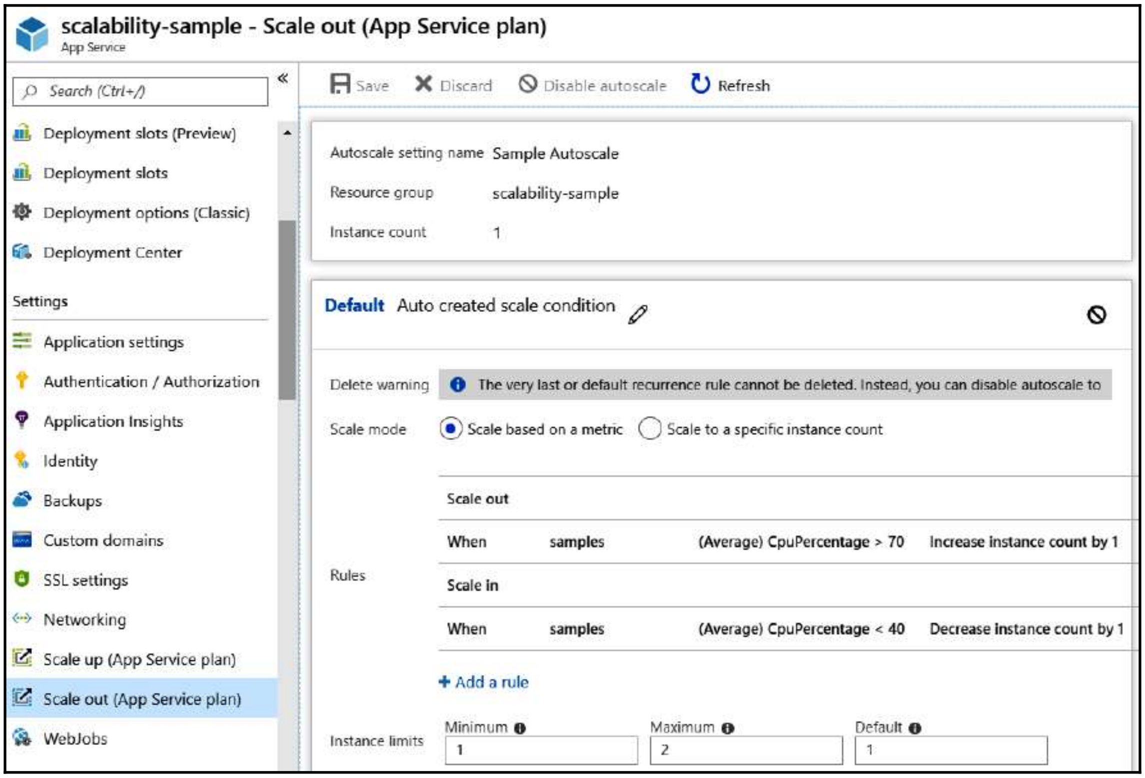
Included hardware

Every instance of your App Service plan will include the following hardware configuration:

Azure Compute Units (ACU)
Dedicated compute resources used to run applications deployed in the App Service Plan. Learn more

Horizontal scaling (Scale out)

Scaling out means splitting all requests among more servers with the same capacity instead of using more powerful machines. The load on all the servers is automatically balanced by the Azure infrastructure. This solution is advised when the overall load may change considerably in the future since horizontal scaling can be automatically adapted to the current load. The following screenshot shows an automatic **Scale out** strategy defined by two simple rules, which is triggered by CPU usage:



Autoscale setting name Sample Autoscale
Resource group scalability-sample
Instance count 1

Default Auto created scale condition

Delete warning: The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to

Scale mode: [✓ Scale based on a metric] Scale to a specific instance count

Rules

Scale out			
When	samples	(Average) CpuPercentage > 70	Increase instance count by 1
Scale in			
When	samples	(Average) CpuPercentage < 40	Decrease instance count by 1

Instance limits

Minimum	Maximum	Default
1	2	1

A complete description of all the available auto scale rules is beyond the purpose of this book. However, they are quite self-explanatory and the *Further reading* section contains links to the full documentation.

The **Scale out** feature is only available in paid service plans.

Creating a scalable web app with .NET Core

Among all the available frameworks for implementing web apps, ASP.NET Core ensures good performance, together with low production and maintenance costs. ASP.NET Core performance is comparable with the performance of Node.js, but production and maintenance costs are lower because of the usage of C# (which is a strongly typed and advanced pure object language) instead of JavaScript.

The steps that follow will guide you through the creation of an ASP.NET Core-based web app. All the steps are quite simple, but some details require particular attention.

First of all, during the web app's creation, you can choose between .NET Core Framework and .NET Framework. Pay attention, because only .NET Core can run on both Windows and cheaper Linux servers, while classic .NET runs only on Windows servers. On the other hand, with classic .NET, you will have access to a larger code base of legacy libraries that include both Microsoft and third-party packages.

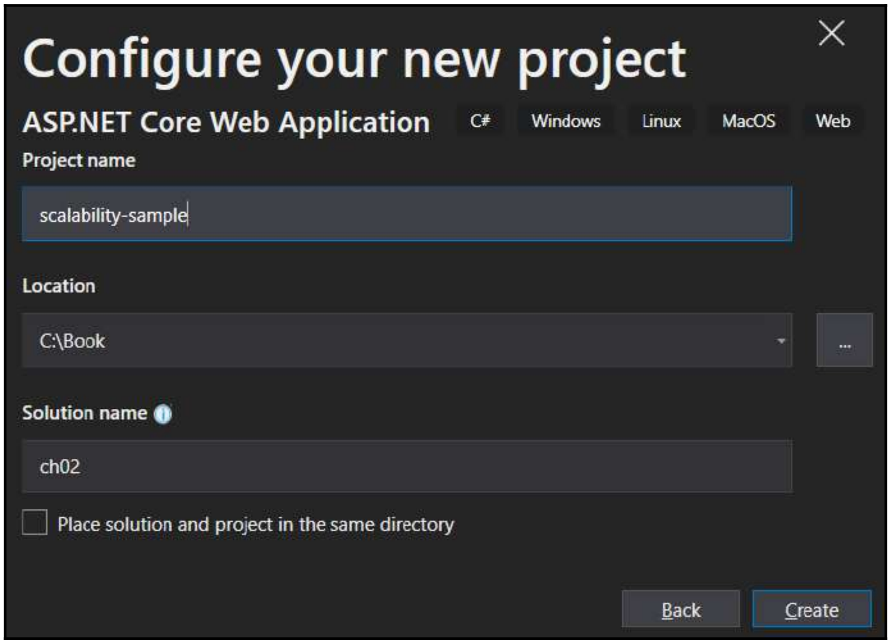
Nowadays, Microsoft recommends classic .NET, just in case the features you need are not available in .NET Core, or even when you deploy your web app in an environment that does not support .NET Core. In any other case, you should prefer .NET Core Framework because it allows you to do the following:

- Run your web app in Windows, Linux, or Docker containers
- Design your solution with microservices
- Have high performance and scalable systems

Containers and microservices will be covered in `Chapter 5`, *Applying a Microservice Architecture to Your Enterprise Application*. There, you'll get a better understanding of the advantages of these technologies. For now, it is enough to say that .NET Core and microservices were designed for performance and scalability, which is why you should prefer .NET Core in all of your new projects.

The following steps will show you how to create an ASP.NET Core web app in Visual Studio 2019 with .NET Core 3.0:

1. Once you select **ASP.NET Core Web Application**, you will be directed to a screen where you will be asked to set up the **Project name**, **Location**, and **Solution name**:


Configure your new project

ASP.NET Core Web Application

Project name: scalability-sample

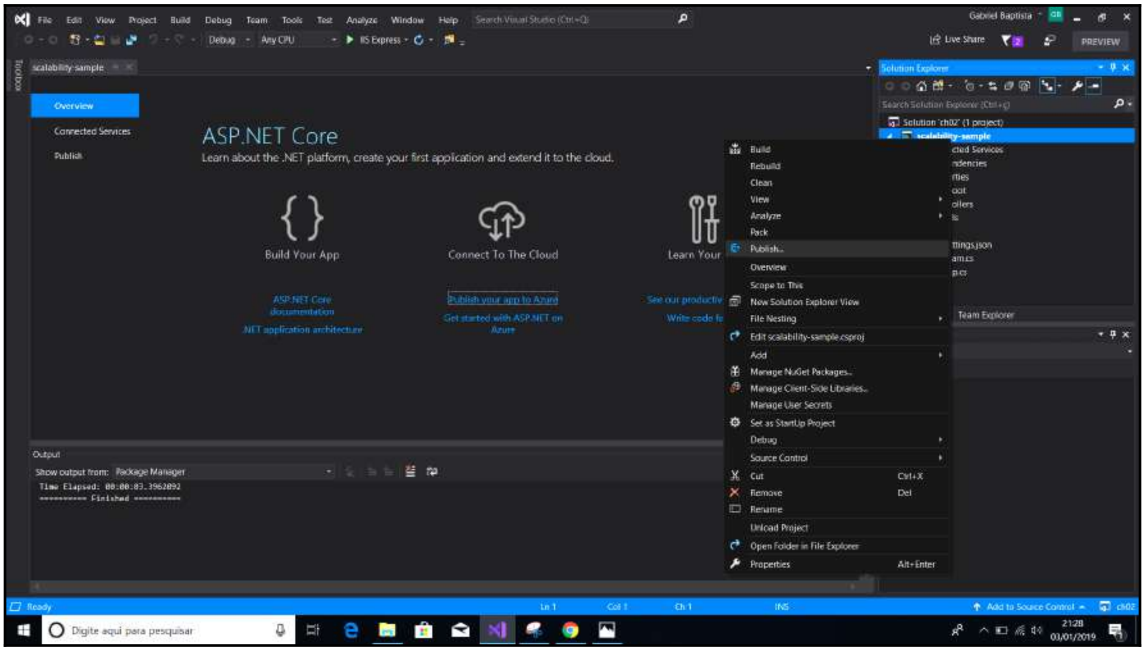
Location: C:\Book

Solution name: ch02

Place solution and project in the same directory

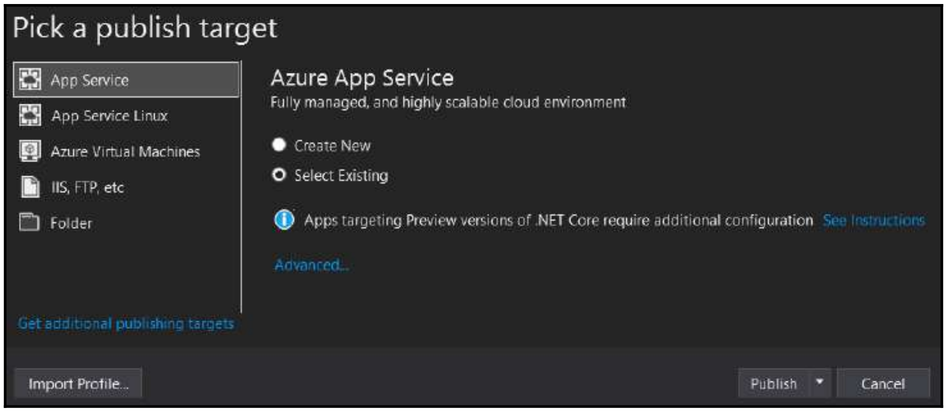
2. After that, you will be able to select the .NET Core version to use. At the time of writing, .NET Core 3.0 was still in its Preview 1 version.
3. Now that we are done with adding the basic details, you can connect your web app project to your Azure account and have it published.

4. In the **Solution Explorer**, you have the option to **Publish...** if you right-click anywhere in there:



```
Time Elapsed: 00:00:03.3962892
========== Finished ==========
```

5. After you select the **Publish...** menu item, you will be able to connect your Azure account and then select the web app you wish to deploy:



Azure App Service

Fully managed, and highly scalable cloud environment

Apps targeting Preview versions of .NET Core require additional configuration See Instructions