Xamarin Continuous Integration and Delivery

Team Services, Test Cloud, and HockeyApp

—

Optimizing your mobile software development process

—

Gerald Versluis

Foreword by Donovan Brown

Apress®

Xamarin Continuous Integration and Delivery

Team Services, Test Cloud, and HockeyApp

Gerald Versluis

Foreword by Donovan Brown

Apress®

Xamarin Continuous Integration and Delivery: Team Services, Test Cloud, and HockeyApp

Gerald Versluis
Sittard, The Netherlands

ISBN-13 (pbk): 978-1-4842-2715-2 ISBN-13 (electronic): 978-1-4842-2716-9
DOI 10.1007/978-1-4842-2716-9

Library of Congress Control Number: 2017941246

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Technical Reviewer: Adam Pedley
Coordinating Editor: Jill Balzano
Copy Editor: Kim Wimpsett
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

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 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 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/9781484227152. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

Contents at a Glance

Contents

Foreword

As a principal DevOps program manager for Microsoft, I was given the opportunity to define *DevOps*. After a month of soul searching, I came up with this single cohesive definition: DevOps is the union of people, process, and products to enable continuous delivery of value to our end users. This is true for all types of development, including mobile. Visual Studio Team Services (VSTS) is the product from Microsoft that enables DevOps for any language targeting any platform. With the help of this book, you are taken from creating your VSTS account all the way to deploying your application. The author gives a quick tour of all the functionality provided by VSTS and then jumps right in to helping with your first build.

Gerald Versluis touches on all three *P*'s of DevOps: people, process, and products. Most of the book covers the products—specifically VSTS—but also discusses other products, including HockeyApp, Xamarin Test Cloud, and Microsoft partner products. Throughout the book, Gerald explains the process that guides teams on their DevOps journey. Additionally, he discusses how to address the toughest *P*: people.

DevOps has benefits for projects of all sizes. Gerald does a great job of not just telling you how DevOps can benefit you and your organization but also explaining why these practices are so important. I always describe VSTS as everything you need to turn an idea into a working piece of software, and *Xamarin Continuous Integration and Delivery* proves that for mobile.

—Donovan Brown
Principal DevOps Program Manager
Microsoft

About the Author

Gerald Versluis is a full-stack software developer and Microsoft MVP from Holland. After years of experience working with Xamarin and .NET technologies, he has been involved in numerous projects, in all kinds of roles. A great number of these projects are Xamarin apps.

Not only does he like to code, but he is keen on spreading his knowledge, as well as gaining some in the bargain. Gerald involves himself in speaking, providing training sessions, and writing blogs (`https://blog.verslu.is`) and articles in his spare time. You can find him on Twitter with the handle `@jfversluis` and can find more information on his web site at `https://gerald.verslu.is`.

About the Technical Reviewer

Adam Pedley is a Microsoft MVP and actively engaged in the Xamarin community. He speaks at technical events and is a contributor and author to several open source projects, including Xamarin Forms. He currently runs the largest community blog dedicated to Xamarin and helps thousands of Xamarin developers around the world, through several online channels.

As a 15-year .NET developer and architect, he has helped government, large businesses, and startups with all aspects of development. Adam is a Xamarin Certified developer and has a bachelor's degree in computer science from Edith Cowan University.

You can read Adam's blog at `www.xamarinhelp.com` and follow him on Twitter `@adpedley`.

Acknowledgments

There are a few people I would like to thank for making this all possible.

First, thank you to the people at Apress, especially Jonathan and Jill who have been guiding me through the adventure of writing my first book. Without them it wouldn't look as good as it does now.

I would also like to thank Erwin, Patrick, Steven, and Ben, colleagues and friends. Each of them has helped me with early feedback and has been an inspiration at several stages while writing.

The technical review was in the hands of the skillful Adam Pedley, my friend from afar. Thank you for being willing to proofread my ramblings and ensure everything looks OK.

Despite his busy schedule, Donovan Brown of Microsoft found some spare time to not only proofread my book and provide some invaluable feedback but also write a foreword for it. Thank you for that.

Finally, I would like to thank Laurie for being patient with me and supporting me in whatever dream I am pursuing this time.

Although I did my best to name everyone who has helped me or has otherwise been of any importance to me during this process, I might have forgotten to mention you. For that I am sorry, but I want to thank you nevertheless.

Introduction

Continuous integration and delivery is a concept that we cannot do without anymore. In a world where good software is being delivered at a very high pace, it becomes more and more important to keep up. If you do not, others will steal away your users. Several books already describe this subject, but none of them focuses on Xamarin apps.

As a developer who loves Xamarin, I wanted to fill this gap. I have set up numerous CI/CD pipelines, and it never gets boring. The challenge is always different because no two apps are the same. Also, continuous integration and delivery for an app is different from "normal" applications. You must code sign your applications, abide by the rules for each app store, and let them go through a review process, to say the least. It requires some specific knowledge on how to achieve this in the first place and second on how to weave the steps into an automated pipeline.

With this book, I have created a hands-on guide for setting up a professional, automated building pipeline for yourself. I have focused on Microsoft technologies such as Visual Studio Team Services, HockeyApp, and of course Xamarin. Not only have I tried to describe the concepts behind these tools, but I take you through the actual steps to get things done. By following along from start to finish, you should be able to configure a pipeline for yourself.

To enable you to get the most out of this book, I also cover testing and specifically automated UI tests with Xamarin Test Cloud.

I hope you will enjoy reading my first book as much as I enjoyed writing it. Happy reading and learning!

CHAPTER 1

Why an Automated Pipeline?

An *automated pipeline*, *continuous integration* (CI), and *continuous delivery* (CD) are all terms that describe the process and tools that help you compile and deliver your software in an automated way. Since the software publication process remains largely the same, you do not have to repeat it manually every single time you want to push out a new version. Repetition is something people are not really good at, but computers are well suited for doing the same thing over and over again.

In this chapter, you will learn about the high-level terms that will be used in this book. Moreover, the chapter will explain why it is a good idea to set up a development pipeline of your own.

While the concept of an automated pipeline is far from new, it does have some interesting challenges when it is combined with the new world of mobile apps. The ecosystem of the different app stores differs from "traditional" desktop and web applications in terms of the freedom you have as a developer to control the environment on which your app will land. Also, it requires some specific handling to create a package that is accepted by the different app store vendors. These are all topics that we will look at throughout the course of this book.

What Is Continuous Integration?

"It works on my machine" is a phrase you will hear a lot when working in software development.

You can get away with that while working alone, when you are responsible for every single piece of code in your app. But when you start working in a team or have to set up a new development environment, there is a chance that you will forget to check in a file, reference a library from somewhere on your hard drive, and so on. These examples would all be causes for your app not being able to build anywhere besides within your specific development environment.

In addition, like every developer, you integrate your full code each time, including the unit tests for your app. Every time before you check in a new piece of code, you run it to see whether everything still lights up green, review the code coverage stats, and write new tests as you go. (You do that, right?) How awesome would it be if all of that testing work could be done for you automatically? You do not have to memorize a list of things you have to do or must not do before you are about to check in. Instead, you just get a signal when something is going wrong.

G. Versluis, *Xamarin Continuous Integration and Delivery*,
DOI 10.1007/978-1-4842-2716-9_1

Finally, one of the things each developer learns over time is that no matter how good you are, you still have bad days and make mistakes like a real person. There really is just one thing you can do about that, and that is to get some feedback about the mistakes as early on as possible so you can fix them before anyone notices.

Well, continuous integration can help you with all three of these situations. CI encompasses a great number of possibilities to ensure the quality of your app. Basically, it does what it says: CI integrates your code continuously with the rest of the code already there, whether that code has been enriched by other members of your team or is just code that you checked in earlier.

With CI, a version of your app will be created with all the latest sources available at that time, and while doing so, the process will let you know if it still compiles.

From there you can start expanding your build process by letting your unit tests run, sending your app through automated UI tests, loading tests, copying a file to another location, transforming your configuration files, generating a build number, and doing anything else you can think of. The rule of thumb is that anything you can run from a command line can be integrated in your builds.

One of the best and most complete solutions to support you with CI and CD (along with many more features) is Visual Studio Team Services (VSTS) by Microsoft. If you have ever worked with Team Foundation Server (TFS) as a source control tool, you should feel right at home.

VSTS is available online, hosted in an Azure variant of TFS, and actually gets new features before the on-premise TFS does. In addition, it is free (to some extent, of course!).

Besides acting as a source control solution, VSTS has a lot of possibilities. One is the automatic building of your source code, which is one of the aspects this book will be focusing on. Besides that, you can manage your work with Scrum or Kanban boards, edit your code from the web portal, work with branches, search code, push NuGet packages, and do much, much more.

While VSTS is a complete environment and leaves you with little to wish for, there are other good services out there like Bitrise and Fastlane. The focus of this book will be on Microsoft products and thus VSTS, but the appendix includes some pointers to alternative tools that are worth checking out.

What Is Continuous Delivery?

Imagine you are in the middle of coding the next functionality in your software or maybe starting a whole new project altogether and suddenly the tester on your team pops up at your desk with a possible bug in your app. Or, just when some big new functionality is only halfway done, your manager finds you and requests the latest version to show to the client's chief executive officer (CEO). While this is something you want to do, you cannot deliver a stable version without rolling back hours of work, and even then you are hoping that you did not forget anything.

These are just two examples in which continuous delivery can prove to be very useful.

As a logical extension of the continuous integration pipeline, CD takes the bits that have been generated while building (also known as *artifacts*) and delivers them to a certain audience of your choosing. This audience can be a tester on your team, a group of predetermined beta testers, or maybe your girlfriend, but it can also be no one. The fact

that you *can* deliver continuously does not mean you *have* to. The fact that delivering to specific environments works is a quality check in itself.

The environments can be the different channels with which you supply your shippable app to end users. Which channel you want to use depends on the requirements you have and the end users you are targeting.

In Chapter 7 you will look at HockeyApp, which is great for distributing your apps in a unified way for all platforms to your end users. It also helps you to collect user feedback, and it has a great software development kit (SDK), which allows you to retrieve detailed crash reports remotely, even for desktop apps.

HockeyApp is an essential tool for the modern developer to detect bugs almost before the user does. However, it cannot be used as a production environment in the case of apps because of the different vendors; Apple, Google, and Microsoft require you to distribute your app though their channels.

The different app stores do have some functionality that allows you to send out test versions. However, the features they offer vary, and more importantly, you, as the developer, need to go through three kinds of app store hoops to get what you want.

As mentioned earlier, there will be a point that you cannot get around the app stores anymore as you will have to distribute your production version through them. However, this does not mean you cannot provide these app stores with your app in an automated way.

Why Do You Need an Automated Build Pipeline?

A "fully featured build pipeline" sounds like something that only big companies can take advantage of, or can even afford. While this may have been true at one point, it sure is not true these days.

So, whether you are a sole developer working on your million-dollar idea, you are already working at a startup with limited funds, or you are working at an established company, you can get started for free on everything that will be described in this book.

There is a possibility that you have one question at this point: why do I even want this? If you are not convinced by the two previous sections, then keep reading.

As mentioned, an automated build pipeline takes a lot of repetitive work out of your hands instantly. From the moment you set up automatic builds, you will benefit, because you will be notified in real time if and when something goes wrong while building your new code. No more will you go hours, days, or even weeks without noticing that the code on your versioning system is in fact unbuildable. That alone should be reason enough to want to set this up.

But the benefits do not stop there. Building is just one thing; there are infinite more tasks that you can add to automated builds. Microsoft has added a marketplace to VSTS, which enables developers to create their own tasks that can be incorporated into a build. There are already a lot of useful packages in there, but if that specific one is not there, you can just create it yourself. This enables you to run any kind of test or diagnostic on your code or app to ensure its quality.

This brings us to the next big advantage: integrating tests. With continuous integration comes testing. While unit testing is probably the first type that comes to mind, you can also integrate functional tests with Xamarin Test Cloud to test your app on thousands of physical devices. This way you can ensure that with each check-in the quality of your software is at least as good as it was before (and ideally the quality has gone up!).

With an automated build pipeline, you get all of this integrated into one place where you can consolidate all the results and build dashboards to inspect the state of your app with just one glance.

When you have this all worked out, then why not do the same for delivering all this to your users? Constructing this process is less work than setting up the automatic builds but takes a lot of work out of your hands nevertheless. To clarify, Figure 1-1 shows an ideal situation.

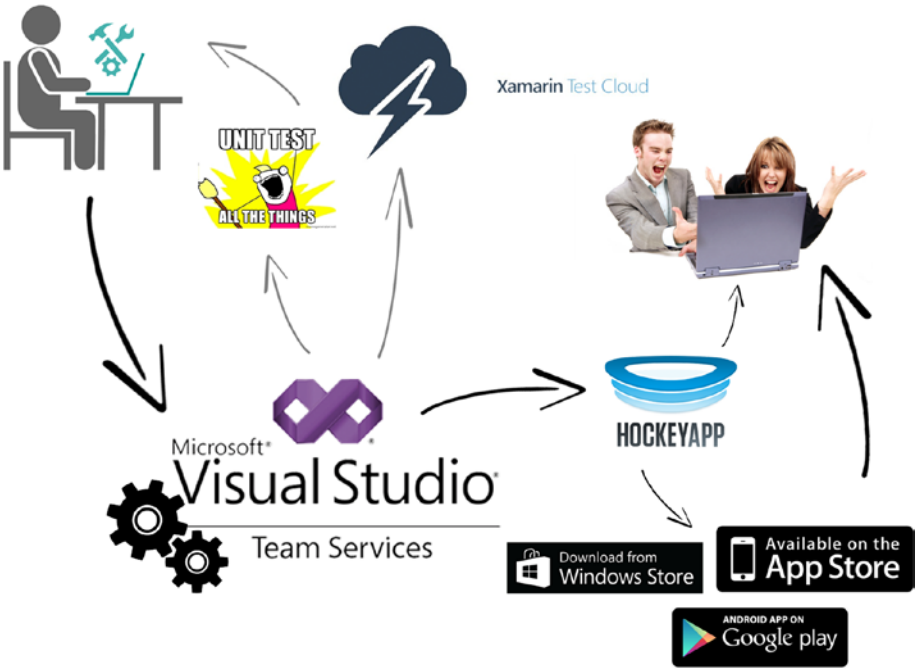

Figure 1-1. *The ideal situation*

In the top left, there is you, working on code and creating great apps. Whenever you are ready, you check your code into VSTS, which will trigger a build. In that build, your app will be compiled to see whether it still works. When it does, the unit tests you have in place will run. If everything is still OK, your app is transferred to Xamarin Test Cloud, and the automated UI test scripts are executed. If anything goes wrong at this point, it will be reported to you.

In that case, you can fix it, and the whole cycle will start again.

If all the builds and tests finish successfully, the resulting app will be released to HockeyApp. Depending on how you want to configure things, you can send your app to users from there, which can be testers of your company, and then gradually roll it out to several other groups. If and when all of these groups approve, you can send it to the actual stores. The final step in the upper right of Figure 1-1 is what it is all about: ecstatic users who love you.

If you have published apps before, you know how much work it can be. It includes identifying all the test users, making sure the right people get the right version, uploading a separate version for each platform, and compiling all the gazillion different icon sizes and screenshots to go with it.

Then, when you finally wrestle through that whole process and decide to send your app to production, you wait for the review process to complete, which can take days before showing up in the app store.

No more! You go outside of the normal app stores with HockeyApp so you can deliver your test versions in a unified way, instantaneously, while gaining a singular way to collect feedback from your users and detailed crash reports.

Also, with the release features in VSTS, you can set up a chain of command to transport your app from environment to environment. This is a powerful tool to give only designated people control over when and where to release the app, even all the way up to production! If you set up VSTS properly, you do not even have to worry about releasing the app yourself anymore.

All of this can be done automatically in a repeatable way so you never forget any step, because it won't be you who is doing it. How to set up an automated pipeline and use it to your full advantage is what this book is all about.

Final Thoughts

Now that you know what continuous integration and continuous delivery are all about, you can start getting your hands dirty. In this chapter, you learned at a high level what CI and CD can do for you, learn about the terminology that comes with them, and saw a scenario to work toward. By the end of this book, you can look back at this scenario and conclude that this is what you have achieved.

In the next chapter, you will learn how to prepare your development environment to begin implementing continuous integration and delivery for your own apps. You will set up some accounts, get an introduction to VSTS, and install the prerequisites needed.

CHAPTER 2

Establishing the Prerequisites

Before getting your hands on all the goods, there are some logistics you have to take care of first. You will need Visual Studio Team Services (VSTS), which is where all of your processes will start from. VSTS also consolidates your results into one convenient and clear overview for you.

If you already have a VSTS account, you can use that one. If you do not have one, you will find out how to set one up in this chapter. While some other accounts may be necessary, you will set them up in later chapters when necessary.

Setting Up a Visual Studio Team Services Account

To set up a VSTS account, you will need a Microsoft account. Creating one is beyond the scope of this book but should be pretty straightforward. Visit https://www.microsoft.com/account to get one. You should just have to click through a few screens, entering your information, like you would expect from any Microsoft product.

Once you have a Microsoft account, go to https://www.visualstudio.com/team-services/. If you want to see for yourself what VSTS has to offer, you can scroll through all the information there. But what you are after is the button in the top right called Free Account.

Like I promised, the account is absolutely free, which is pretty amazing once you find out what it has to offer you. Of course, free always means free to some extent." There are limits. For instance, you can invite up to five people to work with you on one account, so when your team grows to more than that, you will be charged for each user you want to add.

Note that if you or the person you want to add to your account is lucky enough to have a (company) Microsoft Software Developer Network (MSDN) subscription, that user will not count toward the limit of five. Therefore, if your company has a Microsoft partnership with an accompanying MSDN subscription, you can probably use VSTS without worrying about the user limit.

Further, there are some other limits that have to do with the computing minutes you can spend on load testing and hosted, automatic building. For more information about this, refer to the information available at https://www.visualstudio.com/team-services/pricing/.

G. Versluis, *Xamarin Continuous Integration and Delivery*,
DOI 10.1007/978-1-4842-2716-9_2

At the time of this writing, paid accounts start at $30 per month, which gives you ten users on your account. This, however, does not remove the limit for shared services such as build agents, and so on. These can be purchased separately, and prices per month could rise very quickly. However, since I've been using VSTS, I have not had a need for a paid account.

Let's get back to creating your account. In Figure 2-1 you see what the web site looks like at the time of this writing. Web sites do tend to change from time to time, so what you see might differ from Figure 2-1, specifically, the position of the Free Account button.

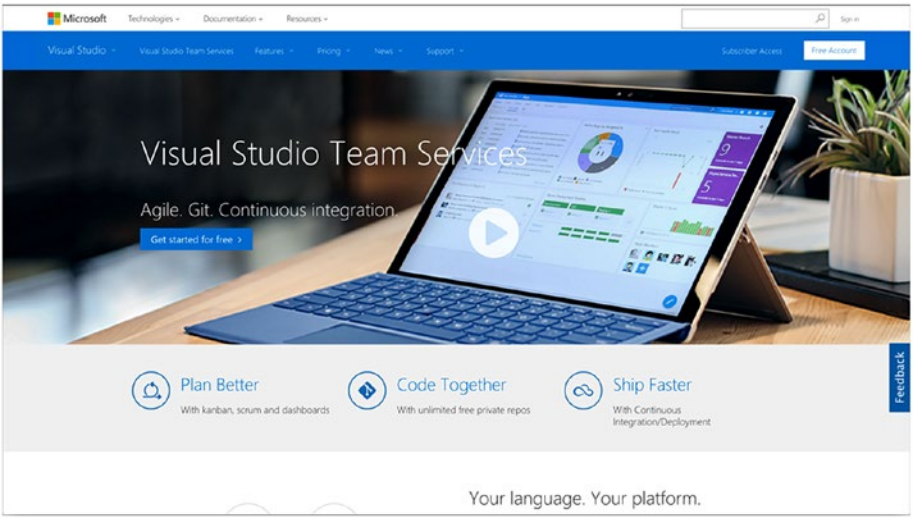

Figure 2-1. *Visual Studio Team Services web site, where you can sign up for an account*

After clicking the Free Account button, you will need to log in with your Microsoft account, after which a new screen will appear, as shown in Figure 2-2. On this screen, the most important thing you must do is pick a name for your VSTS account. The name will result in your own subdomain on the `visualstudio.com` domain.