Microservices, IoT, and Azure

Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions

Bob Familiar

Apress®

Microservices, IoT, and Azure

Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions

Bob Familiar

Apress®

Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions

ISBN-13 (pbk): 978-1-4842-1276-9

ISBN-13 (electronic): 978-1-4842-1275-2

Managing Director: Welmoed Spahr
Lead Editor: James DeWolf
Development Editor: Douglas Pundick
Technical Reviewer: Jeff Barnes
Editorial Board: Steve Anglin, Gary Cornell, Louise Corrigan, James T. DeWolf, Jonathan Gennick, Robert Hutchinson, Michelle Lowman, James Markham, Susan McDermott, Matthew Moodie, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing, Matt Wade, Steve Weiss
Coordinating Editor: Melissa Maldonado
Copy Editor: Mary Behr
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.springer.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.

Apress and friends of ED books 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 Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

I dedicate this book to my incredible wife, Mandy, who is a continuous stream of inspiration and to my children, Ariana and Bobby, who never cease to amaze me with their talent, insight, and intelligence.

Contents at a Glance

Contents

About the Author

Bob Familiar is the Practice Director for Cloud & Services at BlueMetal. BlueMetal is a Modern Application company and the Cloud & Services team is a practitioner of lean engineering, a high velocity product development process that applies lean methodology, service-oriented patterns and practices, automation, and cloud platform capabilities for the design and development of modern applications.

Bob Familiar has been in the software industry for 30 years, having worked for both ISVs such as Dunn & Bradstreet Software and ON Technology and for Microsoft as a Principal Consultant, Architect Evangelist, and Director of Technology Evangelism. Bob holds a Masters in Computer Science from Northeastern and a patent for Object Relational Database and Distributed Computing.

About the Technical Reviewer

Jeff A. Barnes is a Cloud Solution Architect (CSA) on the Microsoft Partner Enterprise Architecture Team (PEAT), where he engages with leading cloud architects to present Microsoft's cloud vision. A 17+ year Microsoft veteran, Jeff brings over 30 years of deep technical experience to the CSA role. He typically works with key ISVs and global partners to demonstrate how Microsoft Azure technologies can be best leveraged to meet the current and future demands of an organization transitioning to the cloud. Jeff has deep practical experience in the retail, financial, and manufacturing industries and is a frequent speaker at Microsoft and third-party events. Jeff resides in Miami, Florida with his family, where his definition of "offshore development" usually equates to "fishing offshore."

Acknowledgments

I would like to thank Liam Spaeth, who has dedicated his life to both inspiring me with his creativity and keeping me employed for the past 20 years, and the leadership team at BlueMetal, Scott Jamison and Matt Jackson, for their support and encouragement.

A big thank you to my dear friend Jeff Barnes for his technical expertise and guidance throughout the process and to Vaclav Turecek and Mike Fussell from the Azure Service Fabric team for their input.

I would like to thank David McIntyre for his contribution to the sample code base, the Home Biomedical Dashboard.

Finally, I would like to thank Ron Bokleman, my PowerShell Sensei, who was instrumental in the creation of the automation scripts for the reference implementation.

Introduction

Microservices, IoT, and Azure make the case for adopting a high velocity, continuous delivery process to create reliable, scalable Software as a Service solutions that are designed and built using a microservice architecture, deployed to the Azure cloud, and managed through automation. SaaS applications are software products that are available 24x7, work on any device, scale elastically, and are resilient to change. This book provides software developers, architects, and operations engineers with practical guidance on this approach to software development through code, script, exercises, and a working reference implementation.

A working definition of microservices will be presented, and the approach will be contrasted with traditional, monolithic, layered architecture. A reference implementation for a fictitious home-biomedical startup will be used to demonstrate microservice architecture and automation capabilities for cross-cutting and business services as well as connected device scenarios for Internet of Things (IoT). Several Azure PaaS services will be detailed including storage, SQL Database, DocumentDb, Redis Cache, Cloud Services, Web APIs, API management, IoT Hub, IoT Suite, Event Hub, Stream Analytics. Finally, we will look to the future and examine Service Fabric to see how microservices are becoming the de facto approach to building reliable software in the cloud.

The Reference Implementation

The Reference Implementation provides automation scripts and source code for several microservices along with several client applications that play various roles in the context of the solution. The PowerShell scripts automate the provisioning, build, and deployment tasks that get the Home Biomedical solution up and running in Azure. In order to control costs of running the reference implementation, deprovisioning scripts are also provided.

The reference implementation consists of several independent microservices built using C#, ASP.NET Web API, DocumentDb, and Redis Cache and deployed as Azure websites. In addition, there is an IoT subsystem that is built using Event Hub, Stream Analytics, Cloud Services, and SQL Database. There is a sample real-time data visualization client that demonstrates how to orchestrate the microservices into a complete solution.

Viewed as a whole, the Reference Implementation demonstrates how to use several Azure PaaS services along with custom code and automation scripts to create a complete, modern application.

You will learn…

The combination of the book and the reference implementation provide a resource to learn the following:

- What microservices are and why they are a compelling architecture pattern for SaaS applications
- How to design, develop, and deploy microservices using Visual Studio, PowerShell, and Azure
- Microservice patterns for cross-cutting concerns and business capabilities
- Microservice patterns for Internet of Things and big data analytics solutions using IoT Hub, Event Hub, and Stream Analytics
- Techniques for automating microservice provisioning, build, and deployment
- What Service Fabric is and why it is the future direction for microservices on Microsoft Azure

Chapter 1: From Monolithic to Microservice - Shifting demographics and competitive pressure on business to drive impact at velocity is requiring us to evolve our approach to how we develop, deploy, and support our software products. This chapter lays out a roadmap to evolve not only application architecture but also process and organization.

Chapter 2: What Is a Microservice? - This chapter provides a working definition of microservices and details the benefits as well as the challenges to evolving to this architecture pattern.

Chapter 3: Microservice Architecture - Traditionally, we have used separation of concerns, a design principle for separating implementation into distinct layers in order to define horizontal seams in our application architecture. Microservice architecture applies separation of concern to identify vertical seams that define their isolation and autonomous nature. This chapter will compare and contrast microservice architecture with traditional layered architecture.

Chapter 4: Azure – A Microservice Platform - The Azure platform exudes characteristics of microservices. This chapter examines several Azure services to identify common patterns of services that are designed and implemented using microservices. Storage, SQL Database, DocumentDb, Redis Cache, Service Bus, API management, and app containers are reviewed.

Chapter 5: Automation - Automation is the key to being able to evolve to a continuous delivery approach and realize the benefits of SaaS. This chapter outlines a framework for defining and organizing your automation process and takes you through 10 exercises that will provision, build, and deploy the reference implementation using PowerShell.

Chapter 6: Microservice Reference Implementation - The epic story of Home Biomedical, a wholly owned subsidiary of LooksFamiliar, Inc., is detailed, and the implementation details of the reference microservices are revealed. The common libraries for ReST calls and DocumentDb and Redis Cache for data access are reviewed. Designing for both public and management APIs is discussed along with the implementation details for the model, interface, service, API, SDK, and console components.

Chapter 7: IoT and Microservices - IoT is becoming a more common solution pattern as we learn to incorporate streaming data into our solutions. This chapter outlines the capabilities needed to realize an IoT solution and maps them to the Azure IoT stack. IoT Hub, IoT Suite, Event Hub, and Stream Analytics are detailed, as is how to use Event Hub, Cloud Services, and Notification Hub to support mobile alerts. A working example of data visualization using a JavaScript client along with SignalR, ReST, and SQL Database is reviewed.

Chapter 8: Service Fabric - Service Fabric is the microservice management, runtime, and infrastructure that Microsoft uses to build, deploy, and manage their own first-class cloud services such as SQL Database, DocumentDb, Bing Cortana, Halo Online, Skype for Business, In Tune, Event Hubs, and many others. This chapter provides a primer and demonstrates Service Fabric by migrating one of the Web API microservices to Service Fabric.

CHAPTER 1

From Monolithic to Microservice

The days of paper-based transactions have passed and the days of small-scale web solutions for Intranet or online customer interaction are nearly forgotten. The monolithic software systems that were designed to function in those worlds are now struggling to keep pace with the expectations of both customers and the business. It is now imperative for companies to provide a modern digital experience for their customers and a platform for the business that can be used to drive impact and derive insight at velocity.

In order to meet the demands of a modern customer base, all companies, regardless of their stated business, must come to the realization that they are also in the software business. That may not be their primary persona but it has become the primary way that their customers expect to interact with them. Demographics are continually shifting, and the expectation of these new customers is that they will engage the companies they choose to do business with digitally. They expect that the experience is beautiful, feature-rich, and fast. They also expect that the experience is reachable and fully functional 24 hours a day, 7 days a week from any device.

It has fallen to us as the software developers, architects, and operations engineers to deliver new, scalable solutions that meet the expectations of customers and the business. We are being asked to deliver more features and functions and to do it with less: less time, less resources. And if you are like most software professionals, you are also responsible for a portfolio of aging, legacy systems that house the ever important business logic somewhere within impenetrable brambles of code and data.

There does exist a software product model that both meets the expectations of customers and provides a platform to drive business outcomes at velocity. That model is called Software as a Service (SaaS). Let's examine the characteristics of a SaaS solution to see how it may provide a means by which we can extricate ourselves from the thorns, spines, and prickles of monolithic legacy applications.

Software as a Service

The Software as a Service model implies that your software product is available 24/7, scales elastically, is highly available and fault tolerant, provides a responsive user experience on all popular devices, and does not require the user to install a client or perform updates or patches. The software product is always the most recent and up-to-date version and is deployed and maintained using a process called Continuous Delivery.

Continuous Delivery

The Continuous Delivery process is defined by the following capabilities:

- The software is developed in a high-velocity, iterative approach and is deployable throughout its lifecycle.
- Deployment to dev, test, staging, and production is managed using an on-demand automated process.
- If there is ever any issue with the deployment process, fixing it takes higher priority over delivering new features.

Continuous Delivery requires a product-oriented software development process that is guided by a set of unwavering principles that prioritizes the frequent release of high-quality working software.

Agile and Scrum

Agile and Scrum have become the prevalent methodology and process for high-velocity software development with teams organized into small cross-functional groups and whose goal is to deliver working, running software at the close of every sprint. Agile defines a set of core principles that are known as the Agile Manifesto:

- We value individuals and interactions over processes and tools.
- We value working software over comprehensive documentation.
- We value customer collaboration over contract negotiation.
- We value responding to change over following a plan.
- That is, while there is value in the items on the right, we value the items on the left more.

In order to apply these principles in the context of an actual project, Ken Schwaber and Jeff Sutherland introduced Scrum, an evolutionary, adaptive, and self-correcting approach to the software development process. Scrum is lightweight, simple to understand, but difficult to master. As with any endeavor, discipline is required in order to achieve a high-quality result.

Scrum is based on an empirical process control theory that asserts that knowledge comes from experience and decisions should be based on what is known. Scrum uses an iterative, incremental approach in order to forecast and reduce risk.

There are three pillars to the Scrum Process:

- **Transparency**: A common language is used to describe the process and is used by all members of the team.
- **Inspection**: Scrum artifacts are frequently inspected in order to detect variances in progress toward a goal.
- **Adaptation**: If it is determined that an aspect of the process will produce an undesirable outcome, the process must be adjusted as soon as possible to minimize damage.

The Scrum Team works together through the well-defined Scrum Process to develop a product backlog, identifying a set of backlog items that will be developed during a two- to four-week sprint called the Sprint Backlog. The Scrum Master and Development Team meet daily in the Daily Scrum to identify what was accomplished the previous day, what will be done today, and if there are any blocking items. This process is monitored closely to determine if the sprint is on track or not. At the close of the sprint, running software is delivered and a Sprint Retrospective meeting is held to review progress and provide input into the next phase of sprint planning.

Scrum, when combined with Agile Principles, provides a process that is more in harmony with the way that software developers work individually and as teams, and has resulted in increased velocity and quality over the traditional gated waterfall process. Software development techniques have emerged from this new way of teaming, such as the use of immutable interfaces and mock objects to support independent workstreams and early testing.

In addition to the need for a high-velocity, high-quality development process, Software as a Service requires a set of principles and processes related to bringing software products to market. Lean engineering is a methodology and process that looks to increase product quality and customer satisfaction by including the customer in the process and providing access to the product early and often. Customers can provide critical feedback that is used to guide product design.

Lean Engineering

Lean engineering has risen out of the startup space and defines a high-velocity product development approach that builds on Agile and Scrum to include deployment into production so as to gather telemetry from the product as well as from the customers using the product. This learning is then folded into the next development iteration (see Figure 1-1).

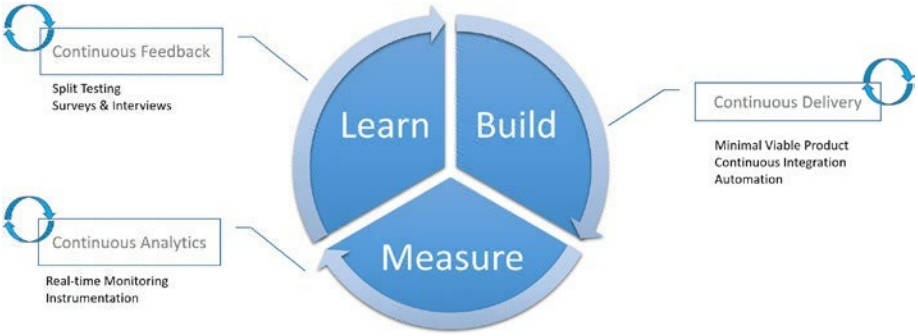

Figure 1-1. *Lean engineering's Build-Measure-Learn cycle*

The Lean engineering cycle is called Build-Measure-Learn and promotes Continuous Delivery, Continuous Analytics, and Continuous Feedback. The creation of dashboards, either developed or provided by third-party tools, are instituted to provide the real-time and historical analytics from which you can derive insights quickly and steer the product development effort in the direction that meets your customer's needs.

DevOps

In order to support a Lean Engineering, Continuous Delivery product development lifecycle, you must automate the development process, called DevOps. DevOps is both a culture and a set of technologies and tools used for automation.

The cultural aspect of DevOps can be the most challenging to organizations. DevOps implies the reorganization of teams combining developers, architects, and quality assurance together with operations. It also requires the adoption of new methodologies, processes, platforms, and tools. It is something that does not happen overnight and should be approached in a phased manner using small teams that adopt the new methods and then transition to become subject matter experts, transferring their knowledge to the rest of the staff.

Cloud

SaaS solutions require an infrastructure and software platform that can provide high availability, fault tolerance, elastic scale, and on-demand storage and compute. In other words, SaaS is a software model designed for the cloud. Cloud platforms such as Amazon's AWS and Microsoft's Azure are themselves Software as a Service platforms that provide all the building blocks needed for the creation of SaaS solutions.

Cloud computing introduced the concept of a managed virtual environment that offers levels of choice with respect to how much of the platform you want to be responsible for maintaining. The terms Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) were introduced to define these choices (see Figure 1-2).