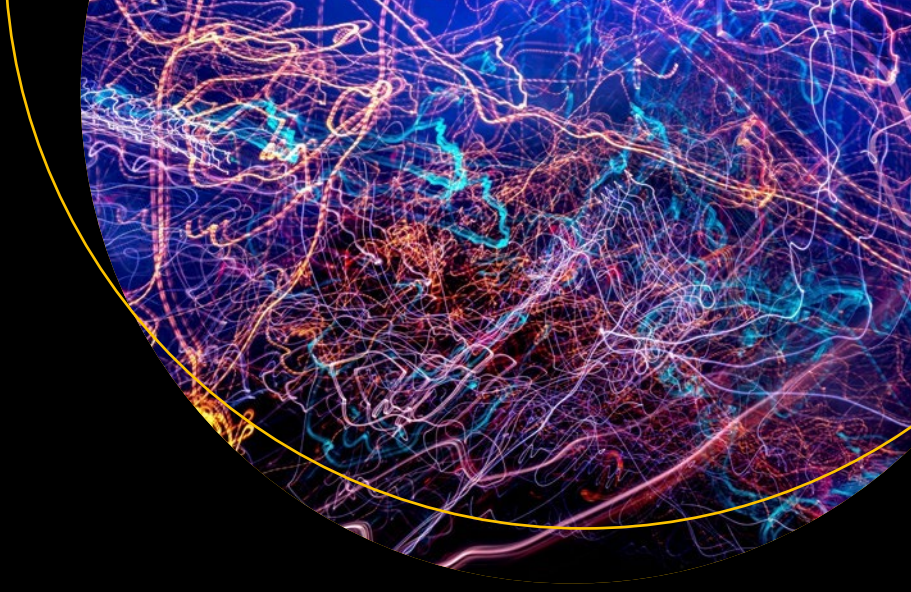


Game Backend Development

With Microsoft Azure and PlayFab

Balint Bors

Apress®

Game Backend Development

With Microsoft Azure and PlayFab

Balint Bors

Apress®

Game Backend Development: With Microsoft Azure and PlayFab

Balint Bors
Munich, Germany

ISBN-13 (pbk): 978-1-4842-8909-9 ISBN-13 (electronic): 978-1-4842-8910-5
https://doi.org/10.1007/978-1-4842-8910-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Joan Murray
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano
Copy Editor: Kezia Endsley

Cover designed by eStudioCalamar

Cover image by Wrongtog on Unsplash (www.unsplash.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. 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 booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

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 (https://github.com/Apress). For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To my wife and our three sons, for all the love, joy, and fun.

Table of Contents

About the Author

Balint Bors is a cloud solutions architect based in Munich, Germany. He has over 15 years of experience developing software and building IT infrastructures for many companies and industries. Balint also consults with and advises technical teams on applying cloud technologies. He is a Microsoft Certified Azure Solutions Architect Expert.

About the Technical Reviewer

Doug Holland is a software engineer and architect at Microsoft. He holds a master's degree in software engineering from the University of Oxford. Before joining Microsoft, he was honored with the Microsoft MVP and Intel Black Belt Developer awards.

Introduction

In recent years, with advances to networks and the cloud, massive backend support has become essential to all successful games. Having backend features enriches players' game experiences and gives your games a competitive advantage.

A typical example is support for multiplayer mode, which allows people to play with each other in the same game session at the same time. Even from far away, the network infrastructure makes this possible. A game stands out from the crowd immediately when it supports multiplayer mode.

Another classic question that game makers face is how to motivate players to play their game longer. One answer is by utilizing backend services. For example, you can implement a leaderboard for players to compete for the best place. Or, you can let players buy game items from a catalog. These are popular ways to engage players to stay with your game. And by the way, they also allow game creators to earn real money.

Further, many people publish their games, yet know nothing about how people experience them. Is there some level in the game that is impossible to accomplish? When do most players give up on your game? Insight into your players' behaviors helps you improve your game immensely.

For all of this, you need a backend infrastructure. You need servers to synchronize the world state among players in multiplayer mode. You need advanced analytics to gather, store, and evaluate game data and get insights into the players' behavior. You need to store catalogs, items, and statistics. You need networks, databases, storage, servers, and the cloud. These are all typical topics you have to deal with as a game developer.

How Does This Book Help?

The cloud and the backend infrastructure is a very complex topic. It's hard to find a good place to start. This book breaks this topic down and focuses only on the aspects relevant to gaming. Games have specific requirements, which differ from web applications, for example.

Through easy and practical examples, this book gives you a comprehensive guide on how to implement the most important backend features for your game.

You learn two ways to implement the backend. One is when you choose a traditional cloud provider; the other is when you are less interested in the backend and want a quick solution, using a Game Backend-as-a-Service (GBaaS) provider. Regardless of the method you choose, this book helps you understand the concepts and gives you step-by-step instructions.

You will learn about backend features, for example, what a virtual economy is and how game analytics help elucidate player behavior. You can use the provided source code to experiment and see how the concepts work in reality.

When you finish this book, you will be able to implement backend features into your game. For example, you'll be able to enable multiplayer mode or create an economy to monetize your game. There are also many other exciting features covered here that can bring your game to another level.

During the last 15 years, I developed infrastructures for many applications. I started my cloud journey with IBM Cloud, then Amazon WebServices (AWS) and Azure. I consulted clients, technical teams, and management, helping them to achieve their goals.

Years ago, when I started developing games, I realized that the game developer community did not have enough guides to incorporate backend services for games. So I decided to support the community and, using my background as an Azure architect, I helped game developers get started

with backend. I also started blogging (https://www.gamebackend.dev/) and engaged with the community.

I learned that game development is not easy. You need to learn a lot. This book aims to be part of this learning process by teaching you about implementing backend features, which are essential elements of modern games.

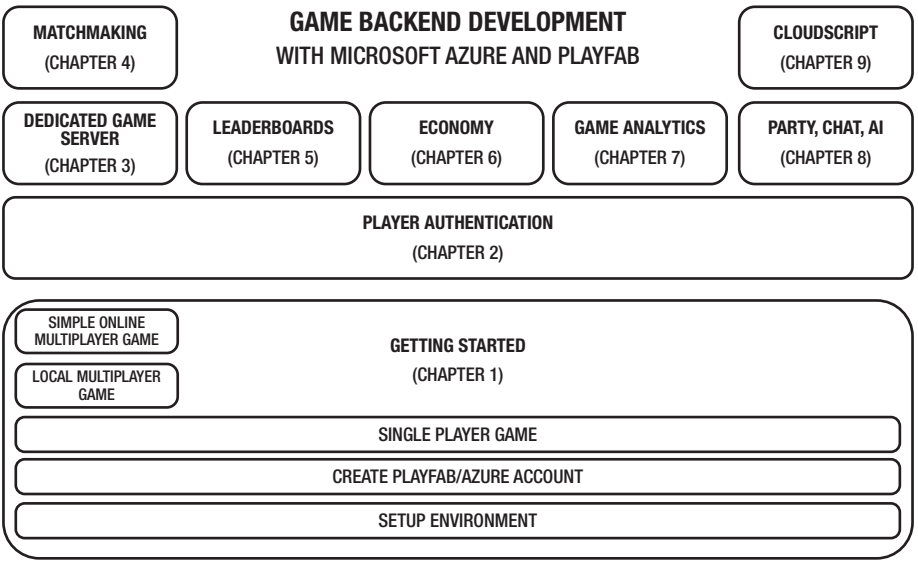
What Are the Benefits of Reading This Book?

- You will learn new and worthy skills**.** Tutorials and step-by-step instructions will guide you so you can implement sophisticated backend features.
- It will save you time and money. It is a comprehensive guide covering all of the most relevant information and will help you quick-start implementing game backends.
- It resolves uncertainty. It shows you where to start with certain backend topics and guide you through complex subjects in a simple way.
- It is practical. You can execute the source code in your environment and see right away how each feature works and contributes to your game.
- It is to the point. The product documentation is often overwhelming. The book focuses only on the main properties of each feature and describes practical steps.
- You can stand out from the crowd. A lot of games do not use backend features. Your game has a competitive advantage over others when you incorporate backend services.

- You learn that implementing the backend can be fun. It's enjoyable to create a game that uses advanced features.

What Does This Book Cover?

This book aims to describe the key game backend services. The following diagram shows the structure of this book and how each chapter is built on each other. In order to implement a specific topic, the underlying building blocks are prerequisites.



Each chapter deals with one backend topic, describing the concept, and then the implementation by Microsoft PlayFab and Azure cloud services.

- **Chapter 1, Getting Started.** You start by setting up the environment and creating a simple game in Unity. Then, you'll extend this to a local multiplayer game. You will also enable online multiplayer game mode, which requires backend support.
- **Chapter 2, Player Authentication.** In this chapter, you learn how to determine the player's identity and implement player authentication in different ways. This is a key feature, as only authenticated players can use other backend functions.
- **Chapter 3, Dedicated Game Servers.** You will build and use scalable backend infrastructures to run online multiplayer game servers.
- **Chapter 4, Matchmaking.** You will use and implement an automated process to bring players together in one multiplayer game session.
- **Chapter 5, Leaderboards.** In this chapter, you build leaderboards to store information like the highest scores achieved. This contributes to the competition among players and encourages player engagement.
- **Chapter 6, Economy.** You will build a virtual economy that allows players to purchase items using real or virtual currencies.
- **Chapter 7, Game Analytics.** In this chapter, you will investigate and implement ways to get insights into players' behavior and gather specific events so that you can optimize your games.

- **Chapter 8, Party, Chat, AI.** You will implement in-game player communication and use cognitive services to process the messages in order to improve your players' experience.
- **Chapter 9, CloudScript and Azure Functions.** You will investigate how to develop server-side logic so that you can implement all kinds of custom backend features.

Future-Proof Technology

This book uses the GBaaS provider Microsoft's Azure PlayFab and pure Azure services. These products are mature enough for implementing a game backend for games of any size.

PlayFab claims to host over 2.5 billion player accounts in over 5,000 games. Azure is the second largest cloud provider in the world. Microsoft heavily invests in the game industry, so you can assume the ecosystem will grow even more.

However, you can use other GBaaS and cloud providers as well. Cloud services are very similar and conceptually there should not be any difference. The book uses reference architectures and considers best practices.

Who Is This Book For?

This book is primarily for game developers who may be skilled in game development, but possess little to no skills in GBaaS and cloud computing. This book is also for professionals working in the cloud solutions space who want to learn about the specific challenges in the gaming domain.

Where Can You Find the Source Code?

All of the source code from this book is available on GitHub at:

https://github.com/apress/game-backend-development

What's Next?

After reading this book, you will understand the most important game backend functionalities—player authentication, enabling multiplayer games, matchmaking, buying items and creating virtual economies, leaderboards, chatrooms, and game analytics, as well as any other custom features you want to implement with the help of cloud scripts.

You will know their basic building blocks, how to implement them, and how to integrate them into your game. With those, you will learn a worthy skill and be able to bring your future games to the next level.

Every day, hundreds and thousands of new games are published. The best ones have integrated backends. Yours should not be left behind.

Let's get started. Read, learn, and implement. Have fun!

CHAPTER 1

Getting Started

Game engines such as Unity and the Unreal Engine simplify the development process and allow you to start developing quickly. In this chapter, you will create a simple game in Unity, which will help you see the backend features discussed in the following chapters in action.

First, however, you learn about and set up some key technologies that you will use throughout this book—Unity, PlayFab, Terraform, and Azure, to name the most important ones.

Then, you will set up Mirror Networking (or Mirror), which enables the basic networking functionality for this game. With the help of Mirror, you can turn this simple game into a local multiplayer one.

Thereafter, you will build a minimum infrastructure in the cloud to host your game server and enable an online multiplayer game.

By the time you complete this chapter, you will have a good foundation for building backend game features. Let's get started.

Game Frontend and Backend

To begin, it is important to define the game's *backend* and *frontend,* at least within the scope of this book.

When implementing a client-server topology for networked games, the software that runs on the clients is called the game's frontend. The server contains all the services (which can be a dedicated multiplayer server as well), and this is the game's backend. See Figure 1-1.

B. Bors, *Game Backend Development*, https://doi.org/10.1007/978-1-4842-8910-5_1

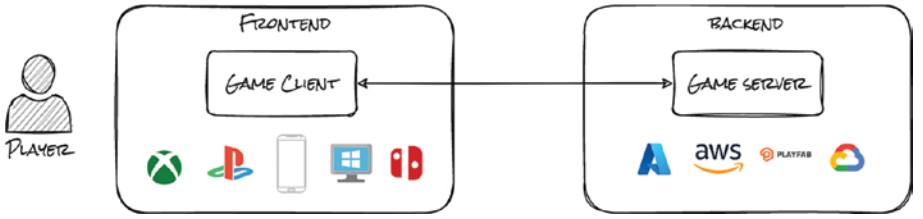


Figure 1-1. *Game frontend and backend*

This is analogous to a web application architecture, where the frontend is what you see in your browser (developed in Angular, React, etc.), and their backends (developed in PHP or Java) are running on a server in the cloud.

Game Frontend

A frontend developer of a game has to deal with many more graphical components and effects than a developer of a web application. The frontend includes your game running on your device. This device can be a PC, mobile phone, or console (Xbox, PlayStation, Nintendo Switch, etc.).

You'll use Unity in this book to implement the game's frontend. Unity is a cross-platform game engine that enables everyone to start creating games, even with limited resources. It provides out-of-the-box 2D and 3D graphics, animation, physics, virtual reality, and a lot more. You can start using it for free and pay only when your game gets traction. It is very popular among individual game developers.

Game Backend

All the servers and services are in the backend, and they are not directly visible to the clients (or players). Still, without them, the game can only operate in a very limited capacity. For example, backends allow buying items or playing with other players.

These backends are hosted in a central location, by a *cloud* or *GBaaS* (*Game Backend-as-a-Service*) providers, and the clients connect to them. Next, you'll review what a cloud is and the added value of GBaaS compared to pure cloud services.

Cloud Computing

Hosting dedicated servers was a difficult task in the early days. You had to invest in expensive IT server infrastructure, which then served the potential clients. In the worst case, the servers stood there without being used. It was hard to buy the exact necessary resources.

Cloud computing, among others, solved this problem. It allows you to reserve resources based on your actual needs. You can scale the servers up and down accordingly and pay as much as you use. A cloud gives you a flexible way to host your servers and provides a couple of important backend services that you can use for your game.

There are a lot of cloud providers nowadays. The three most prominent ones are Amazon Web Services (AWS), Microsoft Azure, and Google Cloud. The competition is fierce, and they keep on improving their services. They are always cheaper, and they give you a free tier where you can try their offerings for free or with discounts.

In this book, you'll implement the backend features on a Microsoft Azure cloud. You can use these ideas and concepts with any other cloud providers, as they are very similar.

Game Backend-as-a-Service (GBaaS)

The rising popularity of GBaaS providers enables game developers to achieve faster time-to-market and deal with the backend infrastructure more easily. Developers do not need to worry about building an infrastructure. With the help of APIs, game developers can easily access and utilize sophisticated backend features.

GBaaS is also backed by a cloud service, but it provides an additional layer on top, which implements specific services for games. In this book, you'll use PlayFab, which is backed by Azure.

Choosing GBaaS vs. a Cloud

Both have advantages and drawbacks, so it depends on your situation and requirements. Generally, building your own custom backend on the cloud is better for bigger projects and teams, while GBaaS is a great fit for smaller or individual developers. See Table 1-1.

Table 1-1. *Comparing Cloud and GBaaS (the More +, the Better)*

	Cloud	GBaaS	Comparison
Difficulty	+	+++	Learning the cloud takes more effort.
Flexibility	+++	+	Cloud's biggest advantage is that it is much more customizable
Implementation efforts	+	+++	With the cloud, you build your own infrastructure and develop the logic.
Maintenance	++	+++	GBaaS provides a fully managed backend infrastructure.
Price	++	+	The cloud provides more options to optimize cost and fit resources to your needs.
Quality	+++	++	Highly depends on providers. But big cloud providers can invest more in availability, security, etc.

In this book, you will find an implementation of each feature with both pure cloud and GBaaS providers. Sometimes you can integrate these two worlds; for example, in the case of CloudScripts, you can call Azure functions through the PlayFab API.

You can also create a hybrid solution, where you implement some of the features on the cloud and some of them with GBaaS. This book aims to help you choose the right situation for your needs and gives you a comprehensive view of both options.

Setting Up the Development Tools

Now you can move on to the practical part. First, you will learn how to set up your development environment and install the required tools.

Unity Editor

You'll start with the frontend. To experiment and learn about the features discussed in this book, you need to have Unity (at least version 2021.3.6f1). You can download it from:

https://unity3d.com/get-unity/download

Because this book focuses on the backend part, we do not discuss Unity any further here. The Unity game example is very simple, and the goal here is to learn how to use backend features. You can also use your own game as a frontend.

Visual Studio Code

Next, you'll get ready for the backend. You will develop the backend infrastructure with code, much in the same way you do for an application.

I suggest developing your infrastructure code in a separate place from your game code, which is in the sovereignty of the Unity Editor.

Basically, you can use any text editor to develop your infrastructure code. I use Visual Studio Code, as it is free, and it provides a convenient integrated development environment with its extensions.

https://code.visualstudio.com/

Note Install the HashiCorp Terraform extension for syntax highlighting, which will help you develop your code.

Terraforming Your Infrastructure

In this book, you use Terraform to provision the servers and accompanied services in the cloud. Terraform allows you to define your infrastructure as source code. It is also called Infrastructure-as-Code (IaC).

Terraform declares every resource (servers, networks, etc.) with all of their attributes in files. It supports multiple cloud providers.

Generally, learning the working mechanisms of Terraform is an extremely worthy skill. You can use it to easily build infrastructures, also on other clouds. It comes with the HashiCorp Configuration Language (HCL), which is easy to learn and use.

You just define the infrastructure components you need in one or more files. Then, you build it in the cloud with one command. With another command, you can destroy the whole thing. You can be sure that nothing is left behind. The next time, you can rebuild the same exact infrastructure. To achieve this manually through the portal or the command-line interface (CLI) is almost impossible. The more your infrastructure grows, the more you will need an IaC.

You can also use Azure Resource Manager (ARM) templates or other third-party tools (such as Ansible) to automate your infrastructure. ARM has better support for the latest Azure changes, but Terraform stands out with its cloud independence and simplicity.

Terraform comes with a single executable file. You can download and execute it:

`www.terraform.io/downloads.html`

Note Put Terraform into your PATH environment variable so that you can reach it from your project folder.

Creating a Single Player Game

After you install the development tools, you can start by creating your game. The intention here is to keep the game frontend as simple as possible. I used a freely downloadable Unity Asset, which includes some 3D characters. I use the Unity Asset called RPG Monster Dui PBR Polyart for the demonstration, because I found the characters hilarious. You can, of course, use your own game and apply the concepts described in this book.

First Steps to Building Your Game

If you choose to use the Unity Asset, here are the detailed steps to follow:

1. Create a new 3D Core project, called `MyGame`, with the help of the Unity Hub, as shown in Figure 1-2.

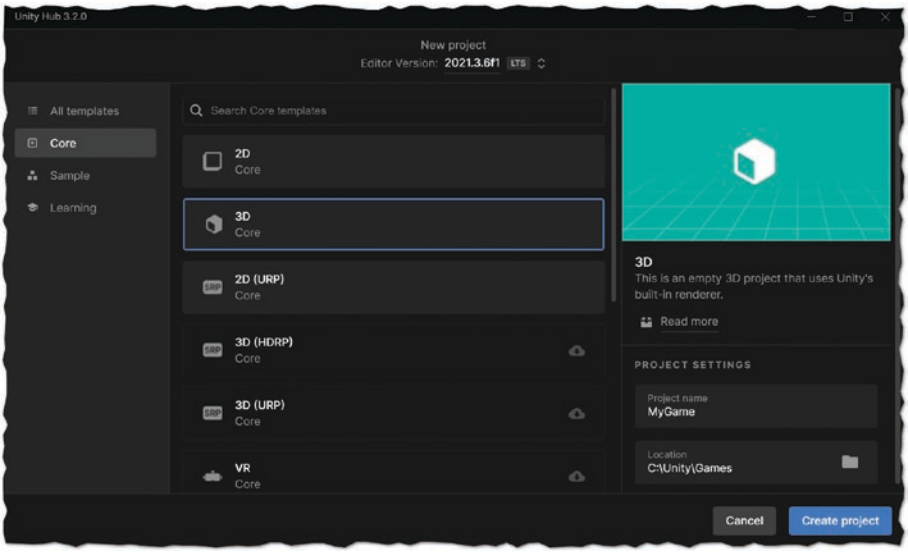


Figure 1-2. *Creating a new Unity project*

2. Go to the Unity Asset Store (choose Unity Editor ➤ Window ➤ Asset Store) and search for Unity Asset RPG Monster Duo PBR Polyart from this link: `https://assetstore.unity.com/packages/3d/characters/creatures/rpg-monster-duo-pbr-polyart-157762`.
3. Choose Add to My Assets ➤ Open in Unity ➤ Package Manager ➤ Import.
4. After you import the game into Unity, go to the sample scene (choose Project Panel ➤ Assets ➤ Scenes) and double-click it.

You can add a *plane* to the scene (right-click to choose Hierarchy Window ➤ 3D Object ➤ Plane) to be the floor where the players will move.

Go to a character prefab (choose Project Panel ➤ Assets ➤ RPG Monster Duo PBR Polyart ➤ Prefabs ➤ Slime) and select it. In the Inspector window, add a Rigidbody and a Box Collider component to it. You should make sure the collider fits with the character (0.5 on the Y value should be fine for this character).

Control and Animation of the Character

Create a new folder of scripts and add a new C# script called `Controller.cs`. You will put all the required logic to move and animate the character in this script. Copy the following code into it:

```
using UnityEngine;

public class Controller : MonoBehaviour
{
    private float speed = 0.01f;
    Animator animator;
    void Start()
    {
        animator = GetComponent<Animator>();
    }
    void Update()
    {
        Vector3 movement = new Vector3(Input.
        GetAxis("Horizontal") * speed, 0, Input.
        GetAxis("Vertical") * speed);
        transform.position = transform.position + movement;

        if ((movement.x != 0) || (movement.y !=0) ||
        (movement.z !=0))
        {
```

```
            transform.rotation = Quaternion.
            RotateTowards(transform.rotation, Quaternion.
            LookRotation(movement), 10000 * Time.deltaTime);
            animator.SetBool("moving", true);
        } else
        {
            animator.SetBool("moving", false);
        }
    }
}
```

Add this script to your `Slime` prefab as a new component.

The `Controller` script simply moves the character by reading the changes of the input and turns it in the right direction. It sets the `moving` parameter if the character moves.

To use this, create new transitions between Idle Normal and RunFWD in the Animator panel (choose Window ➤ Animation ➤ Animator) of the prefab. Right-click IdleNormal, click Make Transition, and choose RunFWD. Now click the transition arrow.

Then, go to the Parameters tab of the Animator and add the moving parameter as bool. After that, you can add the condition for the state transition. If the moving parameter is true, the character should start to run. You can see this configuration in Figure 1-3, after you drag-and-drop your character prefab into the Hierarchy window. See if you can control it.