



ULTIMATE GUIDE TO GAME DATA AND AI



DATA TEAMS EDITION

UNRESTRICTED

4-1-20-1-2-18-9-3-11-19



TABLE OF CONTENTS

Welcome to Data, Analytics and AI	02	Activating Your Playerbase	15
Do you know what you're getting into?	02	• Player Recommendations.....	15
How to use this book	02	• Next Best Offer/Action.....	15
Business Value	03	• Churn Prediction & Prevention.....	16
Talking to the business (feels like combat)	03	• Real-time Ad Targeting.....	16
Creating Value Alignment	03	Operational Use Cases	17
Goals and Outcomes	04	• Anomaly Detection	17
Ultimate Class Build Guide	04	• Build Pipeline.....	17
Creating a character	04	• Crash Analytics.....	18
• Data Engineers	04	Things to Look Forward To	19
• Data Scientists	05	Appendix	21
• Data Analysts.....	05	Ultimate Class Build Guide	21
Diving In	05	• Creating a Character	21
Producing game data.....	05	• Data Engineers	21
And receiving it in cloud	08	• Data Scientists	21
Getting data from your game to the cloud	08	• Data Analysts	22
The Value of Data Throughout the Game Developer Lifecycle	09	Data Access and the Major Cloud Providers	23
Lifecycle overview.....	09	• Cloud Rosetta Stone	23
Use data to develop a next-generation customer experience	09	• Jargon Glossary	23
Getting Started with Gaming Use Cases	10	• Getting started with the major cloud providers.....	23
Where do I start? Start with Game Analytics.....	10	Getting Started with Detailed Use Cases	25
Understand your audience.....	11	• Game analytics.....	25
• Player Segmentation.....	11	• Player Segmentation	25
• Player Lifetime Value.....	12	• Player Lifetime Value.....	26
• Social Media Monitoring	12	• Social Media Monitoring.....	28
• Player Feedback Analysis	13	• Player Feedback Analysis	29
• Toxicity Detection.....	13	• Toxicity Detection.....	30
Find your audience	14	• Multi-Touch Attribution and Media Mix Modeling	31
• Multi-Touch Attribution	14	• Player Recommendations.....	32
		• Next Best Offer/Action.....	33
		• Churn Prediction & Prevention.....	34
		• Real-time Ad Targeting.....	35
		Getting Started with Operational Use Cases	36
		• Anomaly Detection	36
		• Build Pipeline	37
		• Crash Analytics.....	39



Welcome to Data, Analytics, and AI

Do you know what you're getting into?

You may have heard the stories of game studios spending countless hours trying to more effectively acquire, engage, and retain players. Well, did you know that data, analytics, and AI plays a central role in the development and operation of today's top-grossing video games? Studios globally struggle with fragmented views of their audience, with data often outpacing legacy technologies. Today, the need for real-time capabilities and the leap from descriptive to predictive analytics has made it so that data, analytics, and AI are no longer a "nice-to-have", but table stakes for success.

The objective of this handbook is to guide you on the role data, analytics, and AI plays in the development and operations of video games. We'll cover who the key stakeholders are and how to align people across business units. Then we'll talk through strategies to help you successfully advocate for data, analytics, and AI projects internally. Finally, we dive deep through the most common use cases. We want to give you enough information to feel empowered to make a demonstrable impact. Just by reading this you are adding incredible insight and value to yourself as an industry professional. Quest on!

How to use this book

This book is primarily intended for technical professionals who are engaging with data within game studios. No matter your role in the gaming industry, you will be able to glean key takeaways that will make you more effective in your individual role and within the larger team — be that production, art, engineering, marketing, or otherwise.

Begin your journey by reviewing the **"Data, Analytics, and AI Ground Rules"** section to the right, which presents some This section presents some rules and guidelines for interpreting the role that data plays in the game development lifecycle.

Next, it's time to learn about the key professions (aka character classes) that interact and engage with data, analytics, and AI on a consistent basis within a game studio. This section breaks down each of the classes, providing an overview of each character's strengths and weaknesses as

well as helpful tips when operating as or working with one of these classes.

We follow this with the fundamentals for building a Proof of Concept (POC) or Minimum Viable Product (MVP). That is, connecting to the cloud; accessing your data; and most importantly, being able to represent the value you're seeking to unlock as you sell your project into your team and broader organization.

Finally, we'll dive into the most common use cases for data, analytics, and AI within game development. Similar to a tech-tree in a video game, we begin with the most basic use cases — setting up your game analytics. Then we progress through more advanced data use cases such as player segmentation, assessing lifetime value, detecting and mitigating toxicity, multi-touch attribution, recommendation engines, player churn prediction and prevention, and more.

Don't forget to review the Appendix. You'll find a handy ["Jargon Glossary"](#), ["Cloud Rosetta Stone"](#), and ["get started guide for the three major cloud providers"](#). All incredibly helpful assets to keep as hotkeys.

Data, Analytics, and AI Ground Rules

This guide assumes you understand the following:

- You understand the basics of data, analytics, and AI: How and why data is stored in a system, why data is transformed, the different types of output that data can feed into — such as a report, an analysis answering a question, or a machine learning model. If this is the first time you're creating a character, we highly recommend reviewing our data, analytics, and AI tutorial — aka getting started training and documentation, available at dbricks.co/training
- You have a basic understanding of cloud infrastructure. Specifically what it is, who are the key players, and associated terms (e.g., virtual machines, APIs, applications)
- You are generally aware of the game development lifecycle; pre-production, production, testing/QA, launch, operation



Business Value

Demonstrating business value is important when working on data, analytics, and AI projects because it helps ensure that the efforts of the project are aligned with the goals and objectives of the business. By showing how the project can positively impact a game's key performance indicators (KPIs) and bottom-line metrics, such as game revenue, player satisfaction, and operational efficiency, studio stakeholders are more likely to support and invest in the project. Additionally, demonstrating business value can help justify the resources, time, and money that are required to execute the project, and can also help prioritize which projects should be pursued. By focusing on business value, data, analytics, and AI projects can become strategic initiatives that contribute to the long-term success of your game studio.

Talking to the business (feels like combat)

While we highly encourage everyone to read this section, you may already feel confident understanding the needs and concerns of your internal stakeholders, and how to sell-in a project successfully. If so, feel free to skip this section.

We would love to dive into the data to explore and discover as much as possible, unfortunately in most environments, we are limited by resources and time. Understanding both the businesses pain points and strategic goals is crucial to choosing projects that will benefit the business, create value and make your message much easier to sell.

Whenever we embark on a proof-of-concept (PoC) or minimum viable product (MVP) — to prove out a new

methodology or technology — we will need to pitch it back for adoption. The technology could be revolutionary and absolutely amazing, but without the value proposition and tie back to goals, it is likely to land flat or fail to be adopted.

It is key to talk to your stakeholders to understand their perception of pain points and positions on potential projects to add value. Much like stopping at the Tavern when the adventuring party gets to town, these can be informal conversations where you socialize potential solutions while gathering information about what matters.

Creating value alignment

So what are your strategic goals and pain points and how might they be addressed through a use case from a PoC or MVP leveraging your data?

A few examples of strategic goals that are top of mind for our customers at the beginning of any fiscal or calendar year:

- Reduce costs
- Simplify your infrastructure
- Acquire more players
- Monetize your playerbase
- Retain your players (aka prevent churn)

Here are four ways the Databricks Lakehouse can provide value that aligns with your strategic goals and pain points:

- 1. Improved collaboration:** Databricks platform allows everyone to share and collaborate on data, notebooks and models between data scientists, engineers and business users. This enables for a more efficient and streamlined process for data analysis and decision making.
- 2. Find and explore your data:** The data in the Lakehouse is cataloged and accessible, which enables business users to explore and query the data easily and discover insights by themselves.
- 3. Uncover actionable business insights:** By putting your game's data into a Lakehouse architecture, it can be better analyzed using various tools provided by Databricks such as SQL, dashboards, notebooks, visualization and machine learning to better understand your playerbase, providing valuable insights into player behavior and performance. These insights can help the

Questions to ask:

- What other strategic goals and pain points can you list out and how would you prioritize them as a business leader?
- Does your prioritization match how your team, manager and/or leadership would prioritize? Typically the closer the match, the easier initial projects will be to "sell".



business capitalize on player preferences, engagement and retention, and use that information to improve the game and grow monetization.

4. **Lead with data-driven decisions:** A Lakehouse architecture provides a single source of truth for your organization's data. Data engineers write once, data analysts interpret the data, and data scientists can run machine machine learning models on the same data. *This cannot be understated in the value this provides an organization from a total cost of ownership perspective.* With the ability to access and analyze all the data in one place, the business can make unified data-driven decisions, rather than relying on intuition or fragmented data.

Goals and outcomes

Like many projects, starting with a strong foundation of 'what success looks like' will significantly improve your likelihood of achieving your objectives. Here are a few best-practices we recommend:

1. **Set goals:** Define your hypothesis, then use your data and process to prove or disprove your hypothesis. You have a goal in mind, make it part of the experiment. If the outcome differs from the expectation, that is part of experiments and we can learn from it to improve the next experiment. This is all about shortening the feedback loop between insight and action.

2. **Be realistic:** Honor your constraints and scope the project appropriately. For example, are you doing this as a side project? Do you have 2 sprints to show progress? It's important to scope your project based on the time, resources, and quality needed for the said project to be a success.
3. **Scope down:** Ruthlessly control scope for any PoC or MVP. Prioritization is your best friend. Stakeholders and your own internal team will naturally want to increase scope because there's no shortage of good ideas. But by controlling scope, you improve your chances of shipping on time and on budget. Don't let perfection be the enemy of good. There are always exceptions to this, but that is what the next sprint is for.
4. **Deliver on time:** Recovering lost goodwill is incredibly difficult - strive to always deliver on time. Make sure your goals, constraints and scope creep will not explode your timeline as creating tight feedback loops and iteration cycles is what will make you more agile than the competition.
5. **Socialize early, and often:** Show quantifiable value as quickly as possible, both to your immediate team and business stakeholders. Measure the value as frequently as makes sense, and socialize early and often to promote visibility of the project and ensure tight alignment across teams. This will empower you to create tighter feedback loops that will help improve any future iterations of your product, platform, or technology.

Ultimate Class Build Guide

Creating a character

Have you rolled your character already? Data engineers, data scientists, and data analysts form the heart of mature game data teams. Though, depending on studio size and resources, game developers may also be pulled in from time to time to perform data engineering and or data science tasks. Though for the sake of this guide, we'll keep focus on roles of data engineers, data scientists, and data analysts. There are many aspects to these roles, but they can be summarized in that Data Engineers create and maintain critical data workflows, Data Analysts interpret data and create reports that keep the business teams running seamlessly, and Data Scientists are responsible for

making sense of large amounts of data. Depending on the size of the organization, individuals may be required to multiclass in order to address needs of the team. In smaller studios, it's often developers who wear multiple hats, including those in data engineering, analytics and data science. Key characters include:

Data Engineers

Data engineers build systems that collect, manage, and convert source data into usable information for data scientists and business analysts to interpret. Their ultimate goal is to make data accessible so that teams can use it to evaluate and optimize a goal or objective.



Data Scientists

Data scientists determine the questions their team should be asking and figure out how to answer those questions using data. They often develop predictive models for theorizing and forecasting.

Data Analysts

A data analyst reviews data to identify key insights into a game studio's customers and ways the data can be used to solve problems.

Whether you're looking to stand-up an analytics dashboard to report on the health of a title or building a recommendation engine for your players, this guide will help you better understand the unique classes required to develop and maintain an effective data, analytics, and AI platform.

[Learn more about these character classes →](#)

Diving In

Before we get to the primary use cases of game data, analytics, and AI, we need to cover some basics. That is, the different types of game data and how they are produced. And the subsequent receiving of that data in the cloud to collect, clean, and prepare for analysis.



Producing game data...

Speaking in generalities, there are four buckets of data as it relates to your video game.

1. Game Telemetry

Game telemetry refers to the data collected about player behavior and interactions within a video game. The primary data source is the game engine. And the goal of game telemetry is to gather information that can help game developers understand player behavior and improve the overall game experience.

Some of the primary metrics that are typically tracked in game telemetry include:

- **Player engagement:** Track the amount of time players spend playing the game, and their level of engagement with different parts of the game.
- **Game progress:** Monitor player progress through different levels and milestones in the game.
- **In-game purchases:** Track the number and value of in-game purchases made by players.
- **Player demographics:** Collect demographic information about players, such as age, gender, location, and device type.
- **Session length:** Monitor the length of each player session, and how often players return to the game.
- **Retention:** Track the percentage of players who return to the game after their first session.



- **Player behavior:** Track player behavior within the game, such as the types of actions taken, the number of deaths, and the use of power-ups.
- **User Acquisition:** Track the number of new players acquired through different marketing channels.

2. Business KPIs

The second bucket of data is business key performance indicators (or KPIs). Business KPIs are metrics that measure the performance and success of a video game from a business perspective. The primary data source for business KPIs include game telemetry, stores, and marketplaces. These KPIs help game studios understand the financial and operational performance of their games and make informed decisions about future development and growth.

Some of the primary business metrics that are typically tracked include:

- **Revenue:** Track the total revenue generated by the game, including sales of the game itself, in-game purchases, and advertising.
- **Player Acquisition Cost (CAC):** Calculate the cost of acquiring a new player, including marketing and advertising expenses.
- **Lifetime Value (LTV):** Estimate the amount of revenue a player will generate over the course of their time playing the game.
- **Player Retention:** Track the percentage of players who continue to play the game over time, and how long they play for.
- **Engagement:** Measure the level of engagement of players with the game, such as the number of sessions played, time spent playing, and in-game actions taken.
- **User Acquisition:** Track the number of new players acquired through different marketing channels and the cost of acquiring each player.
- **Conversion Rate:** Measure the percentage of players who make an in-game purchase or complete a specific action.
- **Gross Margin:** Calculate the profit generated by the game after subtracting the cost of goods sold, such as the cost of game development and server hosting.

3. Game Services

Similar to game telemetry, game services provide critical infrastructure that requires careful monitoring and management. These services include things like game server hosting,

multiplayer networking, matchmaking, player voice and chat, and more. Here the source of data is the game services used.

Some of the common metrics game teams typically track for these services include:

- **Concurrent Players:** Track the number of players who are simultaneously connected to the game servers to ensure that the servers have enough capacity to handle the player demand.
- **Server Availability:** Monitor the uptime and downtime of the game servers to ensure that players have access to the game when they want to play, particularly important for global live service games where demand fluctuates throughout the day.
- **Latency:** Measure the time it takes for data to travel from the player's device to the game server and back, to ensure that players have a smooth and responsive gaming experience.
- **Network Bandwidth:** Monitor the amount of data being transmitted between the player's device and the game server to ensure that players have a high-quality gaming experience, even on slow internet connections.
- **Live Operations:** Monitor the success of in-game events, promotions, and other live operations to understand what resonates with players and what doesn't.
- **Player Feedback:** Monitor player feedback and reviews, including ratings and comments on social media, forums, and app stores, to understand what players like and dislike about the game.
- **Chat Activity:** Track the number of messages and interactions between players in the game's chat channels to understand the level of social engagement and community building in the game.

4. Data beyond the game

The last bucket comes from data sources beyond the video game. These typically include the following:

- **Social Media Data:** Social media platforms, such as Facebook, Twitter, TikTok and Instagram, can provide valuable insights into player behavior, feedback and preferences, as well as help game teams understand how players are talking about their games online with different communities.
- **Forum Data:** Online forums and discussion boards, such as Reddit and Discord, can be rich sources of player feedback and opinions about the game.



The secret to success is bringing all of the disparate data sources together, so you have as complete a 360-degree view as possible of what's happening in and around your game.

- **Player Reviews:** Ratings and reviews on app stores, such as Steam, Epic, Google Play and the Apple App Store, can provide valuable feedback on player experiences and help game teams identify areas for improvement.
- **Third-Party Data:** Third-party data sources, such as market research firms and industry data providers, can provide valuable insights into broader gaming trends and help game teams make informed decisions about their games and marketing strategies.

This is a lot of data. And it's no wonder that studios globally struggle with fragmented views of their audience, with data often outpacing legacy technologies. Today, the need for real-time capabilities and the leap from descriptive to predictive analytics has made it so that data, analytics, and AI are now table stakes for a game to be successful. Tapping into these four buckets of data sources, you'll find actionable insights that drive better understanding of your playerbase, more efficient acquisition, stronger and longer lasting engagement, and monetization that deepens the relationship with your players.

That's what we're going to dig into throughout the rest of this book.

Let's begin with how to get data out of your game!

There are a variety of ways to get data out of the game and into cloud resources. In this section, we will provide resources for producing data streams in Unity and Unreal. In addition, we will also provide a generic approach that will work for any game engine, as long as you are able to send HTTP requests.

Unity

Since Unity supports C#, you would use a .NET SDK from the cloud provider of your choice. All three major cloud providers have .NET SDKs to use and I have linked the documentation for each below.

No matter the cloud provider, if you want to use a SDK you install it through the NuGet package manager into your Unity project. [A walkthrough of how to implement the .NET SDK](#)

using AWS is provided here.

- **AWS:** [AWS .NET SDK - Unity considerations](#)
- **GCP:** [GCP .NET SDK Documentation](#)
- **Azure:** [Azure .NET SDK Overview](#)
- **Kafka (Open-source alternative):** [Kafka .NET connector](#)

From here, the SDK is used to send data to a messaging service. These messaging services will be covered in more detail in the next section.

Unreal Engine

Unreal supports development with C++, so you could use C++ SDKs or Blueprint interfaces to those SDKs.

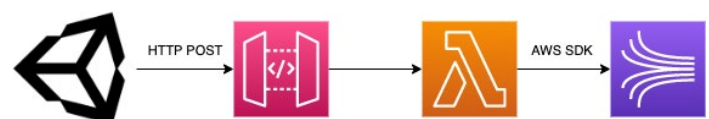
The resources for each SDK are provided here

- **AWS:** [How to integrate AWS C++ SDK with Unreal Engine](#)
- **Azure:** [Azure C++ SDK with PlayFab](#)
- **Kafka (Open-source alternative):** [Getting started with Kafka and C++](#)

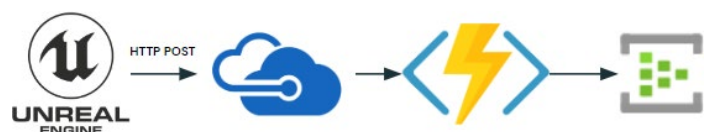
Just like with the Unity example above, from here the data is sent to a messaging streaming service.

Other engines may not support C++ or C#, but there is still a way to get your data into the cloud, no matter the language! By hitting an API Gateway with a HTTP POST request, you are able to send data to cloud services from many more types of applications. A sample high level architecture of this solution in AWS and Azure can be seen below:

AWS:



Azure:



...and receiving it in cloud

Once the data has been sent from the game into an event-streaming service, how do we get that data to a more permanent home? Here we will start by outlining what these messaging services do and how we can use them to point our data to a desired location.

Messaging services ingest real-time event data, being streamed to them from a number of different sources, and then send them to their appropriate target locations. These target locations can be databases, compute clusters or cloud object stores. A key property of the messaging services is to preserve the time in which the events arrive, so that it is always known the order that events occurred.

Examples of cloud messaging services include AWS Kinesis Firehose, Google PubSub, and Azure Event Hubs Messaging. If you prefer to use open-source products, Apache Kafka is a very popular open-source alternative.

Getting data from your game to the cloud

Moving to the cloud platform part of the journey involves building a gaming Lakehouse. The gaming Lakehouse allows gaming companies to store, manage, and analyze large volumes of gaming data, such as player behavior, performance metrics, and financial transactions, to gain valuable insights and make data-driven decisions to improve their business outcomes.

Next here are the basics of the Databricks platform simplified.

Data Ingestion:

- Data can be ingested into the Gaming Lakehouse using various built-in data ingestion capabilities provided by Databricks such as Structured Streaming and Delta Live Tables for a single simple API that handles streaming or batch pipelines.
- Data can be ingested in real-time or batch mode from various sources such as game clients, servers or APIs.
- Data can be cleaned, transformed and enriched with additional data sources, making it ready for analysis.

Data Storage:

- Data is stored in object storage such as S3, Azure Storage or GCP Buckets using Delta Lake.
- Delta Lake is an open-source storage framework that makes it easy to maintain data consistency and track changes.

Data Governance & Cataloging:

- Unity Catalog in Databricks provides tools for data governance that helps with compliance and controlling access to data in the lake.
- Unity Catalog also allows to track data lineage, auditing and data discovery with the use of data catalogs and governance.
- Metadata about the data including the structure, format, and location of the data can be stored in a data catalog.

Data Quality:

- Databricks platform enables you to validate, clean and enrich data using built-in libraries and rule-based validation using Delta Live Tables.
- It also allows tracking data quality issues and missing values by using Databricks Delta Live Tables tables.

Data Security:

- Databricks provides a comprehensive security model to secure data stored in the lake.
- Access to data can be controlled through robust access controls on objects such as catalogs, schemas, tables, rows, columns, models, experiments, and clusters.

Analytics:

- The processed data can be analyzed using various tools provided by Databricks such as SQL Dashboards, Notebooks, visualizations and ML.
- Game studios can gain insights into player performance and behavior to better engage players and improve their games.

Get started with your preferred cloud →



The Value of Data Throughout the Game Development Lifecycle

Lifecycle overview

Over the last decade, the way games have been developed and monetized has changed dramatically. Most if not all top grossing games are now built using a games-as-a-service strategy, meaning titles shipped in cycles of constant iteration to increase engagement and monetization of players over time. Games-as-a-Service models have the ability to create sticky, high-margin games, but they also heavily depend on cloud-based services such as game play analytics, multiplayer servers and matchmaking, player relationship management, performance marketing and more.

Data plays an integral role in the development and operation of video games. Teams need tools and services to optimize player lifetime value (LTV) with databases that can process terabytes-petabytes of evolving data, analytics solutions that can access that data with near real-time latency, and machine learning (ML) models that can translate insights into actionable and innovative gameplay features.

A game's development lifecycle is unique to each studio. With different skillsets, resources, and genres of games, there is no

one model. Below is a simplified view of a game development lifecycle for a studio running a games-as-a-service model.

What's important to remember is that throughout your title's development lifecycle, there is data that can help you better understand your audience, more effectively find and acquire players, and more easily activate and engage them. Whether using game play data to optimize creative decision making during pre-production, tapping machine learning models to predict and prevent churn, or identifying the next best offer or action for your players in real-time, **data is your friend**.

Use data to develop a next-generation customer experience

In the game industry, customer experience (CX) is an important factor that can impact a player's enjoyment of a game and the length they choose to play that game over time. In today's highly competitive and fast-paced games industry, a game studio's ability to deliver exceptional and seamless customer experiences can be a strategic differentiator when it comes to cutting through the noise and winning a gamer's

Game Development Lifecycle

Game-as-a-service (GaaS) / Game-as-a-Community (GaaC)

1. Pre-Production

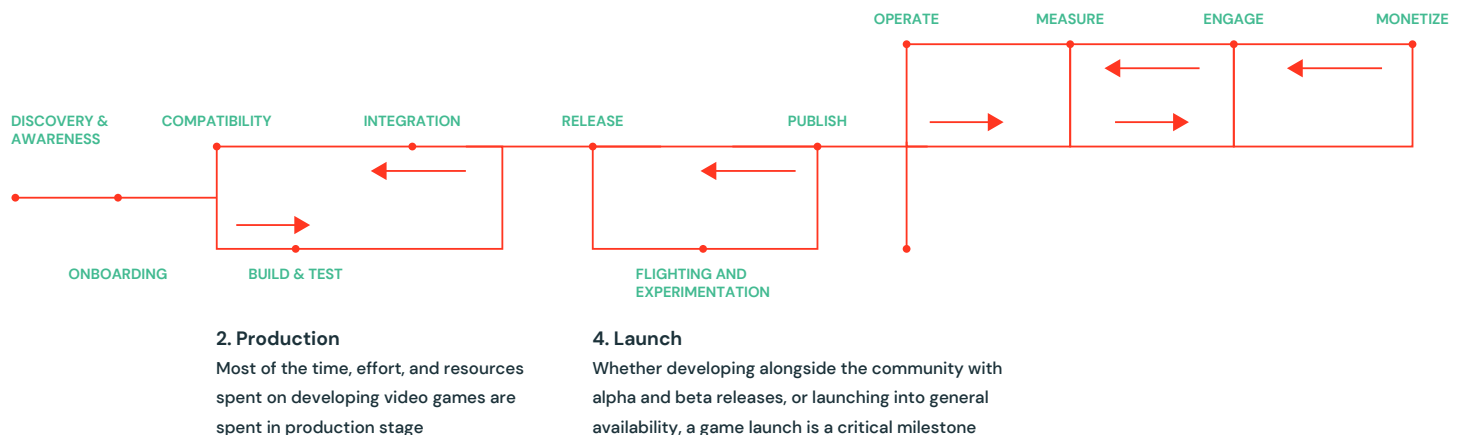
Brainstorm how to give life to the many ideas laid out in the planning phase

3. Testing

Every feature and mechanic in the game needs to be tested for game loop and quality control

5. Operation

As studios increasingly adopt games-as-a-service models, the ongoing operation of a video game is as critical as the launch itself



share of time and wallet. Here are a few examples of how data can help drive value through customer experience:

- 1. **Personalization:** Game studios can use data analytics and machine learning to personalize the game experience for each player based on their preferences and behavior. This can include personalized recommendations for content, in-game events, and other features that are tailored to the player's interests.
- 2. **Omnichannel support:** Players often use multiple channels, such as social media, forums, and in-game support, to communicate with game studios. Next generation customer experience involves providing a seamless and integrated support experience across all these channels in near-real time.
- 3. **Continuous improvement:** Game studios can use data and feedback from players to continuously improve

the game and the player experience. This can include gathering feedback on new features and using it to refine and optimize the game over time.

In summary, defining what a next generation customer experience looks like for your game is important because it can help you create a more personalized, seamless, and enjoyable experience for your players, which can lead to increased engagement, monetization, and loyalty. There are many ways teams can use data throughout a game's development lifecycle, but far and away the most valuable focus area will be in building and refining the customer experience.

Throughout the rest of this guide, we will dig into the most common use cases for data, analytics, and AI in game development, starting with where we recommend everyone begins: game analytics.

Getting Started with Gaming Use Cases

Where do I start? Start with game analytics

Overview


















Big question: Where's the best place to start when it comes to game data, analytics, and AI? For most game studios, the best place to start is with game analytics. Setting up a dashboard for your game analytics that helps you correlate data across disparate sources is infinitely valuable in a world

where there is no one gaming data source to rule them all. An effective dashboard should include your game telemetry data, data from any game services you're running, and data sources outside of your game such as stores, marketplaces, and social media. See below.

What we're trying to solve/achieve

Getting a strong foundation in game analytics unlocks more advanced data, analytics, and AI use cases. For example, concurrent player count plus store and marketplace data

Data Sources

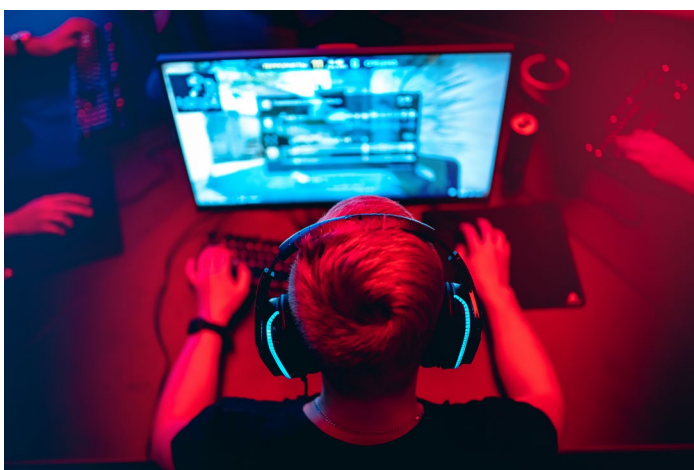
GAME TELEMETRY	GAME SERVICES	OTHER SOURCES
 UNREAL ENGINE	 ONLINE SERVICES	 
		
 CRYENGINE	Amazon GameLift	 
 android	PERFORCE	
 O3DE OPEN 3D ENGINE		 
 GameMaker		
 GODOT Game engine	Azure PlayFab	STEAMWORKS
	STEAMWORKS	



gives you the building blocks for player segmentation and lifetime value. Usage telemetry combined with crash reporting and social media listening helps you more quickly uncover where players might be getting frustrated. And correlating chat logs, voice transcriptions, and or discord and reddit forums can help you identify disruptive behavior before it gets out of hand, giving you the tools to take actionable steps to mitigate toxicity within your community.

[Get started and set up your Analytics Dashboard →](#)

Understand your audience



With your analytics pipelines set up, the first area of focus is to better understand your audience. This can help you inform a variety of key business decisions, from the highest macro order of “what game(s) to develop”, to how to market and monetize those games, and how to optimize the player experience.

By understanding the demographics, preferences, and behaviors of their audience, a game studio can create games that are more likely to appeal to their target market and be successful. You can also use this understanding to tailor your marketing and monetization strategies to the needs and preferences of your players.

Additionally, understanding your audience can help you identify potential pain points or areas for improvement within your games, allowing you to proactively make changes to address these issues and improve the player experience before a player potentially churns.

Understanding your audience is critical to creating games that are relevant and engaging to your players, giving you tools to effectively market and monetize with your audience.

Let's start with Player Segmentation.

Player Segmentation

Overview

Player segmentation is the practice of dividing players into groups based on shared characteristics or behaviors. Segmentation has a number of benefits. You can better understand your players, create more personalized content, improve player retention, and optimize monetization, all of which contributes to an improved player experience.

What we're trying to solve/achieve

The primary objective of segmentation is to ensure you're not treating your entire playerbase the exact same. Humans are different, and your players have different motivations, preferences and behaviors. Recognizing this and engaging with them in a way that meets them where they're at is one of the most impactful ways you can cultivate engagement with your game. As we mentioned above, the benefits of segmentation are broad reaching. Through better understanding of your playerbase, you can better personalize experiences, tailoring content and customer experience to specific groups of players that increases engagement and satisfaction. Better understanding of your players also helps in improving player retention. By identifying common characteristics of players who are at risk of churning (i.e., stopping play), you can develop targeted strategies that only reach specific audiences.

Create advanced customer segments to build out more effective user stories, and identify potential purchasing predictions based on behaviors. Leverage existing sales data, campaigns and promotions systems to create robust segments with actionable behavior insights to inform your product roadmap. You can then use this information to build useful customer clusters that are targetable with different promos and offers to drive more efficient acquisition and deeper engagement with existing players.

[Get started with Player Segmentation →](#)



Player Lifetime Value

Overview

Player lifetime value (LTV) is a measure of the value that a player brings to a game over the lifetime they play that game. It is typically calculated by multiplying the average revenue per user (ARPU) by the average player lifespan. For example, if the average player spends \$50 per year and plays the game for 2 years, their LTV would be $\$50 * 2 = \100 .

What we're trying to solve/achieve

Game studios care about LTV because it helps them understand the long-term value of their players and make informed decisions about how to invest in player acquisition and retention. For example, if the LTV of a player is higher than the cost of acquiring them (e.g., through advertising), it may be worth investing more in player acquisition. On the other hand, if the LTV of a player is lower than the cost of acquiring them, it may be more cost-effective to focus on retaining existing players rather than acquiring new ones.

LTV is one of the more important metrics that game studios, particularly those building live service games, can use to understand the value of their players. It is important to consider other metrics as well, such as player retention, monetization, and engagement.

[Get started with Player Lifetime Value →](#)

Social Media Monitoring

Overview

As the great Warren Buffet once said, "It takes 20 years to build a reputation and five minutes to ruin it. If you think about that, you'll do things differently." Now more than ever, people are able to use social media and instantly amplify their voices to thousands of people who share similar interests and hobbies. Take Reddit as an example. *r/gaming*, the largest video game community (also called a subreddit) has over 35 million members with nearly 500 new posts and 10,000 new comments per day, while over 120 game-specific subreddits have more than 10,000 members each, the largest being League of Legends with over 700,000 members. The discourse that takes place on online social platforms generates massive amounts of raw and organic

data coming from passionate and opinionated users that can be used to understand how customers think and discover exactly what they want.

The act and process of monitoring content online across the internet and social media for keyword mentions and trends for downstream processing and analytics is called media monitoring. By applying media monitoring to social media platforms, game developers are able to gain new advantages that previously might not have been possible, including:

- Programmatically aggregate product ideas for new feature prioritization
- Promote a better user experience by automatically responding to positive or negative comments
- Understand the top influencers in the industry who can sway public opinion
- Monitor broader industry trends and emerging segments such as free-to-play games
- Detect and react to controversies or crises as they begin
- Get organic and unfiltered feedback of games and features
- Understand customer sentiment at scale
- Make changes faster to keep customer satisfaction high and prevent churn

By failing to monitor, understand, and act on what customers are saying about the games and content you release as well as broader industry trends, you risk those customers leaving for a better experience that meets the demands and requirements of what customers want.

What we're trying to solve/achieve

By monitoring and listening to what existing and potential customers are saying on social media, game developers are able to get a natural and organic understanding of how customers actually feel about the games and products they release, or gauge consumer interest before investing time and money in a new idea. The main process for social media monitoring is to gather data from different social media platforms, such as Twitter or YouTube, process those comments or tweets, then take action on the processed data. While customer feedback can be manually discovered and processed in search of certain keyword mentions or feedback, it is a much better idea to automate it and do it programmatically.

[Get started with Social Media Monitoring →](#)



Player Feedback Analysis

Overview

Player feedback analysis is the process of collecting, analyzing, and acting on player feedback to inform game development. It involves collecting player feedback from multiple sources, such as in-game surveys, customer support tickets, social media, marketplace reviews, and forums, and using data analytics tools to identify patterns, trends, and insights. The goal of player feedback analysis is to better understand player needs, preferences, and pain points, and use this information to inform game development decisions and improve the overall player experience.

Player feedback analysis is an important part of game development as it helps ensure that the game continues to meet player needs and expectations. By regularly collecting and analyzing player feedback, game studios can make data-driven decisions to improve the game, increase player engagement and retention, and ultimately drive success and growth.

For this use case, we're going to focus on taking online reviews for your video game and categorizing the different topics players are talking about (bucketing topics) in order to better understand the themes (via positive or negative sentiment) affecting your community.

What we're trying to solve/achieve

This is incredibly helpful, providing data-driven customer insight into your development process. Whether used in pre-production, such as looking at games that are similar with reviews to learn where those games have strengths and weaknesses; or using player feedback analysis with a live service title to identify themes that can apply to your product roadmap, player feedback analysis helps teams better support and cultivate engagement with the player community.

Ultimately, player feedback analysis does two things. 1) It can help you stack rank themes according to positive and negative sentiment, and 2) you can weight those themes according to impact on player engagement, toxicity, monetization, churn, and more. We've all read reviews that are overly positive, or overly negative. The process of player feedback analysis helps to normalize feedback across the community (keeping in mind, only for those who have written a review), so you're not over indexing on one review, or a single theme that may seem in the moment very pressing.

Get started with Player Feedback Analysis →

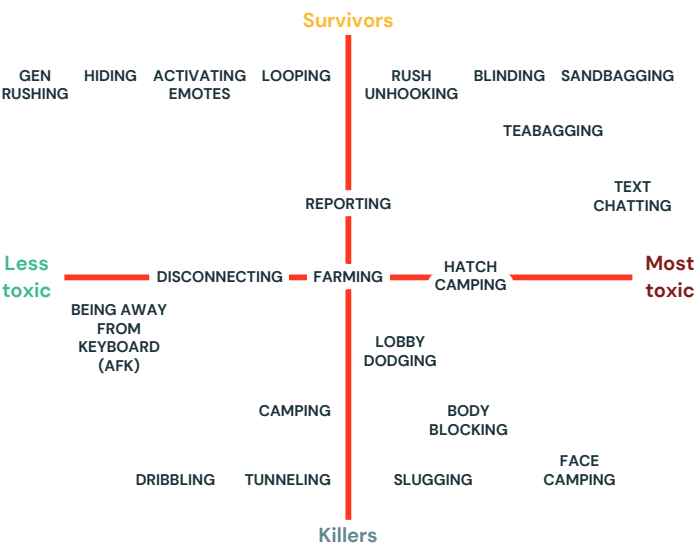
Toxicity Detection



Overview

Across massively multiplayer online video games (MMOs), multiplayer online battle arena games (MOBAs) and other forms of online gaming, players continuously interact in real time to either coordinate or compete as they move toward a common goal — winning. This interactivity is integral to game play dynamics, but at the same time, it's a prime opening for toxic behavior — an issue pervasive throughout the online video gaming sphere.

Toxic behavior manifests in many forms, such as the varying degrees of griefing, cyberbullying and sexual harassment that are illustrated in the matrix below from [Behaviour Interactive](#), which lists the types of interactions seen within the multiplayer game, *Dead by Daylight*.



In addition to the [personal toll](#) that toxic behavior can have on gamers and the community -- an issue that cannot be



overstated -- it is also damaging to the bottom line of many game studios. For example, a study from [Michigan State University](#) revealed that 80% of players recently experienced toxicity, and of those, 20% reported leaving the game due to these interactions. Similarly, a study from [Tilburg University](#) showed that having a disruptive or toxic encounter in the first session of the game led to players being over three times more likely to leave the game without returning. Given that player retention is a top priority for many studios, particularly as game delivery transitions from physical media releases to long-lived services, it's clear that toxicity must be curbed.

Compounding this issue related to churn, some companies face challenges related to toxicity early in development, even before launch. For example, [Amazon's Crucible](#) was released into testing without text or voice chat due in part to not having a system in place to monitor or manage toxic gamers and interactions. This illustrates that the scale of the gaming space has far surpassed most teams' ability to manage such behavior through reports or by intervening in disruptive interactions. Given this, it's essential for studios to integrate analytics into games early in the development lifecycle and then design for the ongoing management of toxic interactions.

What we're trying to solve/achieve

Toxicity in gaming is clearly a multifaceted issue that has become a part of video game culture and cannot be addressed universally in a single way. That said, addressing toxicity within in-game chat can have a huge impact given the frequency of toxic behavior and the ability to automate the detection of it using natural language processing (NLP). In summary, by leveraging machine learning to better identify disruptive behavior so that better-informed decisions around handling actions can be made.

[Get started with Toxicity Detection →](#)

Find your audience

In this section, we're going to talk about how to use your data to more effectively find your target audience across the web. Whether you're engaging in paid advertising, influencer or referral marketing, PR, cross promotion, community building, etc – use data to separate activity from impact. You want to focus on the channels and strategies that leverage your resources most effectively, be that time or money.

Say you have a cohort of highly engaged players who are spending money on your title, and you want to find more gamers just like that. Doing an analysis on the demographic and behavioral data of this cohort will give you the information needed to use an ad platform (such as Meta, Google, or Unity) to do lookalike modeling and target those potential gamers for acquisition.

Multi-Touch Attribution

Overview

Multi-touch attribution is a method of attributing credit to different marketing channels or touchpoints that contribute to a sale or conversion. In other words, it is a way of understanding how different marketing efforts influence a customer's decision to make a purchase or take a desired action.

There are a variety of different attribution models that can be used to assign credit to different touchpoints, each with its own strengths and limitations. For example, the last-click model attributes all credit to the last touchpoint that the customer interacted with before making a purchase, while the first-click model attributes all credit to the first touchpoint. Other models, such as the linear model or the time decay model, distribute credit across multiple touchpoints based on different algorithms.

What we're trying to solve/achieve

Multi-touch attribution can be useful for game studios because it can help them understand which marketing channels or efforts are most effective at driving conversions and inform their marketing strategy. However, it is important to choose the right attribution model for your title based on your business model (one-time purchase, subscription, free-to-play, freemium, in-game advertising, etc.) and regularly review and optimize your attribution efforts to ensure they are accurate and effective.

[Get started with Multi-Touch Attribution →](#)





Activating Your Playerbase

So far, we've discussed how to better understand your players, and how to acquire more of your target audience. Next, we're going to dig into how to better activate your players to create a more engaged and loyal playerbase that stays with your game for the long-term. Here, we're going to focus on strategies that differentiate your gamer experience.

Player Recommendations

Overview

Player recommendations are suggestions for content or actions that a game studio makes to individual players based on their interests and behaviors. These recommendations can be used to promote specific in-game items, encourage players to try new features, or simply provide a personalized experience.

What we're trying to solve/achieve

Player recommendations matter to game studios because they can help improve player retention, engagement, and monetization. By providing players with recommendations that are relevant and engaging, studios can increase the likelihood that players will continue to play their games

and make in-game purchases. Additionally, personalized recommendations can help improve the overall player experience and increase satisfaction.

Game studios can use a variety of techniques to create player recommendations, such as machine learning algorithms, collaborative filtering, and manual curation. It is important to regularly review and optimize these recommendations to ensure that they are effective and relevant to players.

[Get started with Player Recommendations](#) →

Next Best Offer/Action

Overview

Next best offer (NBO) and next best action (NBA) are techniques that businesses use to make personalized recommendations to their customers. NBO refers to the practice of recommending the most relevant product or service to a customer based on their past purchases and behaviors. NBA refers to the practice of recommending the most relevant action or interaction to a customer based on the same information.



For example, a game studio might use NBO to recommend an in-game purchase to a player based on their past spending habits and the items they have shown an interest in. They might use NBA to recommend a specific level or event to a player based on their progress and interests.

What we're trying to solve/achieve

It's important to remember that next best offer is a specific use case within personalization that involves making recommendations to players on the most valuable in-game item or action they should take next. For example, a next best offer recommendation in a mobile game might suggest that a player purchase a specific in-game currency or unlock a new character.

Both NBO and NBA can be used to improve customer retention, engagement, and monetization by providing personalized recommendations that are more likely to be relevant and appealing to individual customers. They can be implemented using a variety of techniques, such as machine learning algorithms or manual curation.

[Get started with Next Best Offer/Action →](#)

Churn Prediction & Prevention

Overview

Video games live and die by their player base. For Games-as-a-Service (GaaS) titles, engagement is the most important metric a team can measure. Naturally, proactively preventing churn is critical to sustained engagement and growth. Through churn prediction and prevention, you will be able to analyze behavioral data to identify subscribers with an increased risk of churn. Next, you will use machine learning to quantify the likelihood of a subscriber to churn, as well as indicate which factors create that risk.

What we're trying to solve/achieve

Balancing customer acquisition and retention is critical. This is the central challenge to the long-term success of any live service game. This is particularly challenging in that successful customer acquisition strategies needed to get games to scale tend to be followed by service disruptions or declines in quality and customer experience, accelerating player abandonment. To replenish lost subscribers, the acquisition engine continues to grind and expenses mount. As games reach for customers beyond the core playerbase they may have initially targeted, the title may not resonate

with new subscribers over the same durations of time or may overwhelm the ability of these players to consume, reinforcing the overall problem of player churn.

At some point, it becomes critical for teams to take a cold, hard look at the cost of acquisition relative to the subscriber lifetime value (LTV) earned. These figures need to be brought into a healthy balance, and retention needs to be actively managed, not as a point-in-time problem to be solved, but as a "chronic condition" which needs to be managed for the ongoing health of the title.

Headroom for continued acquisition-driven growth can be created by carefully examining why some players leave and some players stay. When centered on factors known at the time of acquisition, gaming studios may have the opportunity to rethink key aspects of their acquisition strategy that promote higher average retention rates, which can lead to higher average revenue per user.

Prerequisites for use case

This use case assumes a certain level of existing data collection infrastructure in the studio. Notably, a studio ready to implement a churn prediction and prevention model should have

- A cloud environment where player data is stored
 - This source data should contain player behavior and session telemetry events from within the game. This is the foundation that insights can be built on top of.

[Get started with Churn Prediction & Prevention →](#)

Real-time Ad Targeting

Overview

Real-time ad targeting in the context of game development focuses on using data to deliver personalized and relevant advertisements to players in near real-time, while they are playing a game. Real-time targeting is performance based, using highly personalized messages which are achieved by using data to precisely determine the most opportune moments to display ads, based on factors such as player behavior, game state, and other contextual information. Knowing when to send those ads is based on data. This use case is specific to titles using in-game advertising as a business model. It's important to note that in-game real-time ad targeting requires a sophisticated tech stack, with



data processing and analytics tools, as well as integrations with bigger ad ecosystem, ad networks and partners. The Databricks Lakehouse platform is an optimal foundation as it already contains many of the connectors required to enable this use case.

What we're trying to solve/achieve

The goal of in-game real-time ad targeting is to provide a more immersive and relevant advertising experience for players, while also increasing the effectiveness of the ads for advertisers. By delivering targeted ads that are relevant to each player's interests, game developers can create a more enjoyable and personalized gaming experience, which can help to reduce churn and increase the lifetime value of each player. Additionally, real-time ad targeting can also help game developers monetize their games more effectively, as advertisers are willing to pay a premium for hyper-targeted and engaged audiences.

[Get started with Real-time Ad Targeting →](#)

Operational use cases

In the game development industry, operational analytics are essential for ensuring a smooth and efficient production process. One common use case is anomaly detection, where data analytics is utilized to identify any unusual patterns or behaviors in the game, such as crashes or performance issues. This helps developers quickly identify and fix problems, improving the overall quality of the game. Another example is build pipelines, where data analytics can be used to monitor and optimize the process of creating new builds of the game. By tracking key metrics such as build time, error rates, and resource utilization, developers can make informed decisions about how to optimize the build process for maximum efficiency. Other operational use cases in game development include tracking player behavior, measuring server performance, and analyzing sales and marketing data. Lets explore a few of these below.

Anomaly Detection

Overview

Anomaly detection plays an important role in the operation of a live service video game by helping to identify and diagnose unexpected behaviors in real-time. By identifying patterns and anomalies in player behavior, system performance, and network traffic, this information can then be used to detect and diagnose server crashes, performance bottlenecks, and hacking attempts. The ability to understand if there will be an issue before it becomes widespread is immensely valuable. Without anomaly detection, which is a form of advanced analytics, you're always in a reactive (rather than proactive) state. Anomaly detection is a type of quality of service solution.

What we're trying to solve/achieve

The goal of anomaly detection is to ensure that players have a stable and enjoyable gaming experience. This has an impact across your game, from reducing downtime, to minimizing player churn, and improving your game's reputation and revenue. Additionally, the insights gained from anomaly detection can also be used to mitigate cheating and disruptive behavior.

[Get started with Anomaly Detection →](#)

Build Pipeline

Overview

A build pipeline is a set of automated processes that are used to compile and assemble the code, assets, and resources that make up a game project. The build pipeline typically includes several stages, such as code compilation, optimization, testing, and release. The purpose of a build pipeline is to streamline the game development process and ensure that each stage of development is completed efficiently and effectively. A build pipeline can be configured to run automatically, so that new builds are generated whenever changes are made to the code or assets. This helps to ensure that the game is always up-to-date and ready for testing and release. The logs are collected are in near-real time from build servers. A simplified example: Dev X is committing code on title Y, submitted on day Z, along with the log files from the pipeline and build server. Builds typically take multiple hours to complete, requiring significant amounts of compute via build farms. Being able to



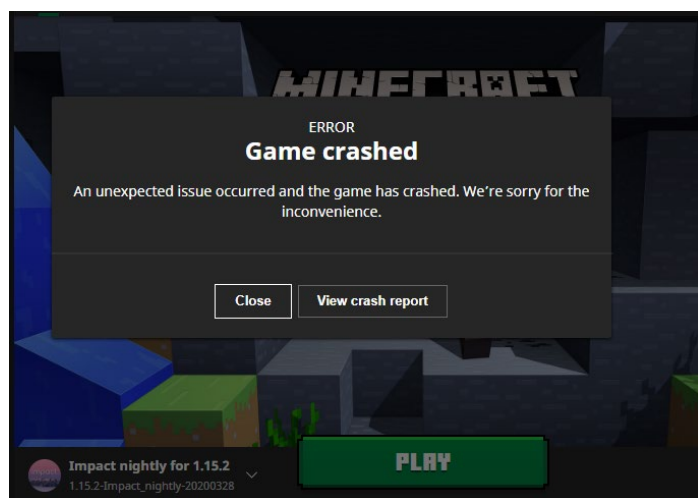
understand what builds are relevant for testing, which builds are wasting compute, and being able to predict which builds will fail as they go through the pipeline are ways to curb operational expenses.

What we're trying to solve/achieve

With this use case, we're seeking to reduce wasted compute and build a foundational view of what was developed, by who, when and how testing performed. In an ideal state, our automated build pipeline could send a notification to the developer with a confidence metric on the build making it through, allowing them to decide whether to continue or move another build through the pipeline. Often, developers do not have clear visibility until the build has completed or failed. By providing more insight to devs into the build pipeline process, we can increase the rate at which builds are completed efficiently and effectively.

[Get started with Build Pipeline →](#)

Crash Analytics



Overview

Games crash, it is a fact of game development. The combination of drivers, hardware, software, and configurations create unique challenges in tracking, resolving and managing the user experience.

Crash analytics and reporting is the process of collecting information about crashes or unexpected failures in a software application, in this case, a video game. A crash report typically includes information about the state of the game at the time of the crash, such as what the player was

doing, what assets were being loaded, and what system resources were being used. How long crash testing takes can vary, depending on the game's business model, amount of content, and scope. For a title with a one-time release, where there is a large amount of content and a complex storyline, the chances of hidden crashes causing errors while in development are high, making it require more time to perform testing before the game can be published. For titles built in a game-as-a-service model, i.e. a game shipped in cycles of constant iteration, crash detection should be done continuously, since errors in newly released content might affect the base game and lead to crashes.

Increasingly, titles are being released in alpha (where developers do the testing), closed beta (which includes a limited group of testers/sample-users who do the gameplay testing) and open betas (where anyone interested can register to try the game). All of which happens before the game is "officially" released. Regardless of alpha, beta, or GA, players may stumble over game crashes, which triggers crash reports that are sent to the developers for fixing. But sometimes, it can be challenging to understand the issue that caused the crash from crash reports provided by your game's platform.

What we're trying to solve/achieve

Ultimately, the purpose of crash analytics is to identify the root cause of a crash, and help you take steps to prevent similar crashes from happening in the future. This feedback loop can be tightened through automation in the data pipeline. For example, by tracking crashes caused on builds from committers, the data can provide build suggestions to improve crash rate. Furthermore, teams can automate deduplication when multiple players experience the same errors, helping to reduce noise in the alerts received.

[Get started with Crash Analytics →](#)





Things to look forward to

This eBook was created to help game developers better wrap their heads around the general concepts in which data, analytics, and AI can be used to support the development and growth of video games. **If you only have 5 minutes, these takeaways are critical to your success.**

For more information on advanced data, analytics, and AI use cases, as well as education resources, we highly recommend Databricks training portal [dbricks.co/training](https://databricks.co/training).

Top takeaways:

If you take nothing else from this guide, here are the most important takeaways we want to leave with you on your journey.

1. Data is fundamental. Data, analytics, and AI play a role throughout the entire game development lifecycle – from discovery to pre-production, development to operating a game as a live service. Build better games, cultivate deeper player engagements, and operate more effectively

by utilizing the full potential of your data.

- 2. Define your goals.** Start by establishing the goals of what you're hoping to learn and or understand around your game. Clear goals make it easier to identify key metrics to track, example goals include; developing high-quality games that provide engaging and satisfying player experiences, increasing player engagement and retention by analyzing and improving gameplay and mechanics, and building a strong and positive brand reputation through effective marketing and community outreach.
- 3. Identify and understand your data sources.** Spend time to identify and understand the breadth of data sources you are already collecting, be that game telemetry, marketplace, game services, or sources beyond the game like social media. It is critical to collect the right data, and track the right metrics based on the goals and objectives you have set for your game.
- 4. Start small, and iterate quickly.** Recognize that goals and objectives evolve as you learn more about the interaction



between your game and your player base. Data projects are most effective when scoped small with tight feedback loops, allowing you to quickly adapt with your community and alongside shifting market conditions.

5. **Game analytics forms the foundation.** Start by getting a game analytics dashboard up and running. The process of building out a dashboard will naturally require connecting and transforming your data in a way to unlock more advanced use cases down the road.
6. **Plan and revisit your data strategy frequently.** Once dashboarding is set up, you'll have a better picture of what downstream data use cases make the most sense for your game and business objectives. As you move to use cases such as player segmentation, churn analysis, and player lifetime value, revisit your data strategy frequently to ensure you're spending time on use cases that drive actionable insights for you and your team.
7. **Show value broad and wide.** Whether your data strategy is new or well established on the team, build the habit of communicating broadly to stakeholders across the company. Early in the process, it is important to gather critical feedback on what data is helpful and where there are opportunities for improvement. The worst thing that can happen is you create something that no one uses. That is a waste of everyone's time and money.
8. **Ask for help.** Engage with your technical partners. There are humans who can help ensure you're developing your data and analytics platform in a way that is efficient and effective. There are numerous partners with domain expertise in data science and data engineering that can accelerate your data journey – here is our recommended partner list for [data, analytics, and AI workloads](#).
9. **Participate in the community.** The community for game analytics is large and growing. It is important to research and

explore different communities to find the ones that best fit your needs and interests. Here are a few of our favorites:

- a. **IGDA Game Analytics:** The IGDA has a number of Special Interest Groups that bring together user researchers, designers, data engineers and data scientists focused on understanding player behavior and experiences. They offer resources and events for those working in games user research, including a yearly Games User Research Summit.
 - b. **Data Science Society:** The Data Science Society is a global community of data scientists and engineers. While not specifically focused on game development, they offer a wealth of resources and opportunities for learning, networking, and collaboration in the field of data science.
 - c. **Hugging Face:** is hub of open source models for Natural Language Processing, computer vision, and other fields where AI plays its role. They also provide an online platform where users can access pre-trained models and tools, share their own models and datasets, and collaborate with other developers in the community.
 - d. **Data Engineering subreddit:** The Data Engineering subreddit is a forum for data engineers to discuss topics related to building and managing data pipelines, data warehousing, and related technologies. While not specifically focused on game development, it can be a valuable resource for those working on data engineering in the gaming industry.
10. **Go beyond dashboards.** Looking at dashboards is only the first step in your data journey. Imagine how the output of your data can be presented in a way to help stakeholders across your company achieve more. For example, dropping data into an application that can help game designers make balancing decisions based on player events.



APPENDIX

Ultimate class build guide

Creating a character

The heart and soul of mature data teams are formed by this trio of classes. There are many aspects to these roles, but they can be summarized in that Data Engineers create and maintain critical data workflows, Data Analysts interpret data and create reports that keep the business teams running seamlessly, and Data Scientists are responsible for making sense of large amounts of data. Depending on the size of the organization, individuals may be required to multiclass in order to address needs of the team. In smaller studios, it's often developers who wear multiple hats, including those in data engineering, analytics and data science.

Whether you're looking to stand-up an analytics dashboard to report on the health of a title or building a recommendation engine for your players, this guide will help you better understand the unique classes required to develop and maintain an effective data, analytics, and AI platform.

Data Engineers

Data engineers build systems that collect, manage, and convert source data into usable information for data scientists and business analysts to interpret. Their ultimate goal is to make data accessible so that teams can use it to evaluate and optimize a goal or objective.

Responsibilities:

- Data Engineers are responsible for data migration, manipulation, and integration of data (joining dissimilar data systems)
- Setup and maintenance of ETL pipelines to convert source data into actionable data for insights. It is the responsibility of the data engineer to make sure these pipelines run efficiently and are well orchestrated.
- The Data Engineer sets up the workflow process to orchestrate pipelines for the studio's data and continuously validates it
- Managing workflows to enable data scientists and data analysts, and ensuring workflows are well-integrated with different parts of the studio (e.g., marketing, test/QA, etc)

Goals and Priorities of Data Engineers

- Enable access to usable data for real-time insights — data that both enables timely decision-making and is accurate and reproducible
- Increase user confidence and trust in data. This involves ensuring high consistency and reliability in ETL processes
- Limit the issues and failures experienced by other engineers and data scientists, allowing those roles to focus less on troubleshooting and more on drawing meaningful conclusions from data and building new products / features

What Data Engineers care about:

- Enabling access to data for real-time insights — data that both enables timely decision-making and is accurate and reproducible
- Building high-performance, reliable and scalable pipelines for data processing
- Delivering data for consumption from a variety of sources by Data Analysts and Data Scientists against tight SLAs
- A Data Engineer's biggest challenge? Collaboration across teams

Data Scientists

Data scientists determine the questions their team should be asking and figure out how to answer those questions using data. They often develop predictive models for theorizing and forecasting.

Responsibilities:

- Responsible for making sense of the large amounts of data collected for a given game title, such as game telemetry, business KPIs, game health and quality, and sources beyond the game such as social media listening
- The analytics portion of a Data Scientist's job means looking at new and existing data to try and discover new things within it
- The engineering component may include writing out pipeline code and deploying it to a repository
- Data Scientists are responding for building, maintaining, and monitoring models used for analytics and/or data products



Goals and Priorities:

- Developing new business capabilities (such as behavioral segmentation, churn prediction, recommendations) and optimizing processes around those capabilities
- Increase ROI by building algorithms and tools that are maintainable and reusable
- Exploring (or further expanding) the use of machine learning models for specific use cases
- Bridges the gap between engineering and analytics, between the technology teams and business teams
- Provides business side of studio with data that is crucial in decision-making, for example a churn model that helps predict the impact of a new feature set

What Data Scientists care about:

- Creating exploratory analysis or models to accurately predict business metrics, e.g., customer spend, churn, etc., and provide data-driven recommendations
- Enable team with actionable insights that are easy to understand and well curated
- Create and move models from experimentation to production
- A Data Scientist's biggest challenge? Keeping up with advancements and innovation in data science, and knowing which tools and libraries to use

Data Analysts

A data analyst reviews data to identify key insights into a game studio's customers and ways the data can be used to solve problems.

Responsibilities:

- Often serves as the go-to point of contact for non-technical business / operations colleagues for data access / analysis questions

- Analysts often interpret data and create reports or other documentation for studio leadership
- Analysts typically are responsible for mining and compiling data
- Streamline and or simplify processes when possible

Goals and Priorities:

- Empower stakeholder and business teams with actionable data
- "Catch things before they break". Proactively mitigate potential data issues before they occur (for internal and external customers)
- Analysts are often recruited to assist other teams (i.e., BI teams) with their domain knowledge
- Driving business impact through documentation and reliable data

What Data Analysts care about:

- Easy access to high quality data.
- Quickly find insights from data with SQL queries and interactive visualizations.
- The ability to easily share insights and while creating impactful assets for others to consume (dashboards, reports).
- A Data Analyst's biggest challenge? Working with complex processes and complicated technologies that are filled with messy data. While fighting these challenges, Analysts are often left alone or forced through paths that prevent collaboration with others across team/organization.
- Untrustworthy data: often Analysts get asked to provide answers to leadership that will leverage the data to determine the direction of the company. When the data is untrustworthy or incorrect due to previously mentioned challenges this can eventually lead to lack of trust in the data teams from leadership or the business.



Data access and the major cloud providers

Cloud Rosetta Stone

[AWS / Azure / GCP Service Comparison – Click Here](#)

If you are newer to the cloud computing space, it is easy to get lost between the hundreds of different services between the three major cloud providers. The table below is meant to highlight the important data, analytics, and AI services used by the various hyperscale service providers Amazon, Microsoft, and Google. In addition, it aims to pair up services from different cloud providers that serve the same purpose.

Getting started with the major cloud providers

Here are some quick ways to get started with the three major cloud providers: AWS, Azure, and GCP:

AWS:

1. **Create an AWS account:** The first step is to create an account on the AWS website. This will give you access to the AWS Management Console, which is the web-based interface for managing your AWS resources.

2. **Use the AWS free tier:** AWS offers a free tier of service that provides a limited amount of free resources each month. This is a great way to get started and try out various AWS services without incurring any charges.
3. **Explore the AWS Management Console:** Once you have an account and are logged in, take some time to explore the AWS Management Console and familiarize yourself with the various services that are available.
4. **Next you can search for Databricks:** In the AWS Management Console, use the search bar in the top-left corner of the page and search for “Databricks”.
5. **Navigate to the Databricks page:** Once you have found the Databricks page, you can access it to get started with the Databricks service.
6. **Launch Databricks Workspace:** To launch the Databricks Workspace on AWS, you can use the CloudFormation template provided by Databricks. Databricks CloudFormation template creates an IAM role, security group, and Databricks Workspace in your AWS account.

Azure:

1. **Create an Azure account:** The first step is to create an account on Azure portal. This will give you access to the Azure portal, which is the web-based interface for managing your Azure resources.

Service Type	Service Description	AWS Service	Azure Service	GCP Service
Storage	Object storage for various file types and artifacts (CSV, JSON, Delta, JAR). Objects can be retrieved by other services	Amazon Simple Storage Service (S3)	Azure Blob Storage	Google Cloud Storage
Compute	High-performance VMs to run applications. Platform where data transformations are run in Big Data apps.	Amazon Elastic Compute (EC2)	Azure Virtual Machines	Google Compute Engine
Messaging	Real-time event streaming services to write data to object stores or data warehouses. One OSS version is Kafka	Amazon Kinesis	Azure Service Bus Messaging	Google Pub/Sub
Data Warehouse	Traditional data storage layer for structured data, to then be used by data analysts. Often used to read from a Data Lake, which acts as a single source of truth	Redshift or Databricks	Synapse or Databricks	BigQuery or Databricks



Jargon Glossary

CDP	Customer Data Platform (CDP). A CDP is a piece of software that combines data from multiple tools to create a single centralized customer database containing data on all touch points and interactions with your product or service.
ETL	Extract, Transform, Load. In computing, extract, transform, load is a three-phase process where data is extracted, transformed and loaded into an output data container. The data can be collated from one or more sources and it can also be outputted to one or more destinations
KPI	Key Performance Indicator, a quantifiable measure of performance over time for a specific objective. KPIs provide targets for teams to shoot for, milestones to gauge progress, and insights that help people across the organization make better decisions.
POC	Proof of Concept (PoC). A proof of concept is a prototype or initial implementation of a solution that is developed to demonstrate the feasibility of a concept or idea. It is often used to test the effectiveness of a new tool or approach to data analysis or machine learning before investing in a full-scale implementation.
MVP	Minimum Viable Product (MVP). An MVP refers to the smallest possible solution that can be delivered to meet a specific business need. The goal of an MVP is to quickly validate assumptions and prove the potential value of a larger project. By delivering a smaller solution first, stakeholders can gain confidence in the project and see a return on investment sooner, while also providing feedback to improve the larger project.
ROI	Return on investment (ROI), which is calculated by dividing the profit earned on an investment by the cost of that investment.
Serverless computing	Using compute platforms that are completely managed by service providers. When using serverless computing, you simply execute queries or deploy applications and the service provider (AWS, Databricks, etc.) handles necessary server maintenance.
VPC	Virtual Private Cloud. A VPC is a virtual cloud networking environment, which helps organize and give you control of your resources. You also define how resources within your VPC can communicate with other regions, VPCs, and the public internet with traffic rules and security groups.

- 2. Take Azure tutorials:** Azure provides tutorials, documentation, and sample templates to help you get started. These resources can help you understand the basics of Azure and how to use its services.
- 3. You can search for Databricks:** In the Azure portal, use the search bar at the top of the page and search for “Databricks”.
- 4. Navigate to the Databricks page:** Once you have found the Databricks page, you can access it to get started with the Databricks service.
- 5. Create a new Databricks workspace:** To create a new Databricks workspace, you can use the Azure portal, Azure CLI or Azure Powershell. Once created, you’ll be able to access your Databricks Workspace through the Azure portal.
- 6. Other Azure Services:** Once you have a Databricks workspace setup, you can easily connect it to other Azure Services such as Azure Storage, Event Hubs, Azure Data Lake Storage, Azure SQL and Cosmos DB for example.

GCP:

- 1. Create a GCP account:** the first step is to create an account on GCP portal. This will give you access to the GCP Console, which is the web-based interface for managing your GCP resources.
- 2. Explore the GCP Console:** Once you have an account and are logged in, take some time to explore the GCP Console and familiarize yourself with the various services that are available.
- 3. Search for Databricks:** In the GCP Console, use the search bar in the top-left corner of the page and search for “Databricks”.
- 4. Navigate to the Databricks page:** Once you have found the Databricks page, you can access it to get started with the Databricks service.
- 5. Create a new Databricks workspace:** To create a new Databricks workspace, you can use the GCP Console or the gcloud command-line tool. Once created, you’ll be able to access your Databricks Workspace through the GCP Console.



Detailed Use Cases

Getting started with game analytics

Fortunately, standing up an effective analytics dashboard is getting easier. It all starts with getting your data into an architecture that sets your team up for success. Selecting any of the major cloud providers — [AWS](#), [Azure](#), [GCP](#) — you can land all your data into a cloud data lake, then use Databricks Lakehouse architecture to run real-time and reliable processing. Databricks can then help you visualize that data in a dashboard, or send to a visual analytics platform, such as Tableau.

1. **Sign up for a Databricks account:** You'll need to create an account on the Databricks website in order to use the platform.
2. **Access the Databricks portal:** Interact with the Databricks platform and run tasks such as creating clusters, running jobs, and accessing data.
3. **Set up a development environment:** You'll need a development environment where you can write and test your code, whether you're using a local IDE or the Databricks Workspace.
4. **Collect data:** Once you have your development environment set up, you can start collecting data from your game. This can involve integrating or building a SDK into your game code, or using another tool to send data to cloud storage.
5. **Process and analyze the data:** Once you have collected your data, you can use Databricks to process and analyze it. This can involve cleaning and transforming the data, running queries or machine learning algorithms, or creating visualizations.
6. **Monitor and optimize:** Regularly monitor your analytics to ensure that they are accurate and relevant, and use the insights you gain to optimize your game.

Keep in mind that these are just general steps to get started with Databricks for game analytics. The specific steps you'll need to take will depend on your specific use case and needs.

If you have any questions about how this solution can be deployed in your environment, please don't hesitate to [reach out](#) to us.

Tips / Best Practices

- **Define your goals:** What do you want to learn from your analytics data? Having clear goals will help you focus on collecting the right data and making meaningful use of it.
- **Plan your data collection:** Determine what data you need to collect, how you will collect it, and how you will store it.
- **Consider privacy:** Make sure you are transparent with your players about what data you are collecting and how you will use it, and give them the option to opt out if they wish.
- **Use analytics to inform design:** Leverage your analytics data to inform decisions around game design, such as any balance changes or new content targeting a specific audience.
- **Monitor and test your analytics implementation:** Regularly check your analytics to ensure that data is being collected correctly, and conduct tests to validate the accuracy of your data.
- **Visualize your data:** Dashboarding your data is one of the most effective ways to quickly and effectively make sense of what's happening at a given moment in time.
- **Use data to improve player retention:** Analyze player behavior and use the insights you gain to improve player retention, such as by identifying and addressing pain points or by providing personalized content.
- **Collaborate with your team:** Share your analytics findings with your team and encourage them to use the data to inform their work.
- **Keep it simple:** Don't try to collect too much data or create overly complex analytics systems. Keep it simple and focused on your goals.
- **Start where you are:** If you've yet to gather all of your data, don't go build some fancy model. Start with the data you have available to you and build from there.

Getting started with Player Segmentation

Player segmentation is crucial to studios as it allows them to better understand their audience and tailor their game experience to meet their specific needs and preferences. By dividing players into different segments based on factors such as demographics, playing styles, and in-game behavior,



studios can gain valuable insights into what motivates and engages their players. This information can then be used to design games that not only provide a more enjoyable experience for players, but also drive player retention and increase revenue for the studio. In a competitive industry where player satisfaction is key to success, player segmentation is an essential tool for studios to stay ahead of the game.

Start by evaluating the segmentation goals such as:

- **Personalize the experience:** Changing or creating experience specific designs to the player.
- **Create relevant content:** Surface the best content to players based on features and behaviors that will matter the most depending on the player's place in the games life cycle.
- **Monetization:** Create tailored monetization strategies that effectively reach and convert each player group. For example, you may have a group of highly engaged players who are more likely to make in-app purchases, while another group is less likely to spend money but may be more receptive to advertisements.

The next steps would be to identify, collect and analyze player data. By gathering information on player behavior, preferences, and demographics, you can gain insights into their motivations, pain points, and what drives their engagement with your game.

There are multiple types of player data to collect, including:

- **Player Behavior:** Track player behavior and actions within your game to gain insights into their play style, preferences, and patterns.
- **Surveys:** Ask players directly about their preferences, motivations, and feedback through in-game surveys, email questionnaires, or other forms of direct communication.
- **Focus groups:** Gather a small group of players to discuss and provide feedback on specific aspects of your game and player experience.
- **Social media listening:** Monitor social media platforms to gather insights into how players are engaging with and talking about your game.

[Customer Segmentation solution accelerator](#) →

Tips / Best Practices

Define your segmentation goals: Determine what you want to learn about your players and why. This will help you focus your analysis and ensure that your segments are meaningful and actionable.

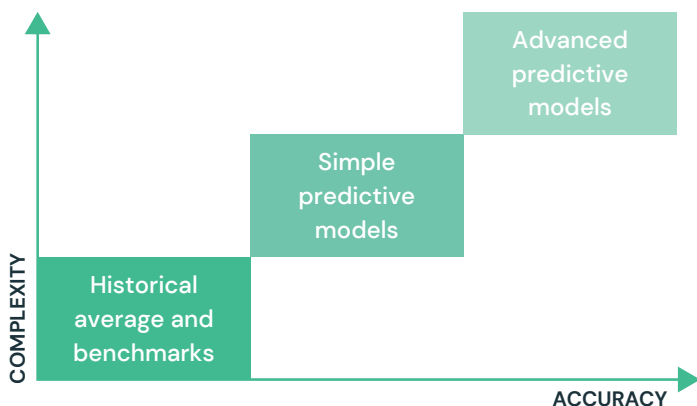
- **Use meaningful criteria:** Choose criteria that are relevant to your goals and that differentiate players in meaningful ways. This could include demographic information, in-game behavior, spending habits, or a combination of factors.
- **Analyze player data:** Use data from your players to inform your segmentation strategy. This could include data on in-game behavior, spending habits, or demographic information.
- **Use multiple methods:** We recommend using a combination of methods, such as clustering to create segments that are statistically meaningful and actionable to your game.
- **Validate your segments:** Test your segments to ensure that they accurately reflect the differences you observed in your player data. This could involve comparing the segments to each other, or validating the segments against external data sources.
- **Consider ethical and privacy concerns:** Ensure that your segmentation strategy is ethical and complies with privacy laws and regulations. This could involve anonymizing your player data, obtaining consent from players, or other measures to protect player privacy.
- **Monitor and refine your segments:** Regularly review your segments to ensure that they remain relevant and meaningful. Refine your segments as necessary to reflect changes in your player data or your goals.

Getting Started with Player Lifetime Value

Assuming you've followed the steps to collecting, storing, and preparing your player data for analysis; To calculate player lifetime value (LTV), the quick and dirty way of assessing overall player LTV is to divide the total revenue by the total number of registered players. Note, LTV is a critical calculation for return on investment, which is player lifetime spend versus the amount spent on player acquisition. Ideally, you want lifetime spend to be equal to or more than cost of acquisition.



As long as your game and its community are currently active, any player lifetime value calculations should be considered models, not exact numbers. This is because many of the players you're considering are likely actively registered and actively playing, so the exact player LTV number is a moving target.



The first calculation is relatively simple. We suggest using average revenue per user (ARPU), which is a game's daily revenue divided by the number of active users, to help you calculate lifetime value. First, you'll need to define what is an active player using retention values; which can be set to a week, multi-day, or multi-week period of time depending on how your game has performed to date. You can then look at the number of users who churn on a given day, averaging with the number of days from the player's first visit to the current date (or the specific date you've considered the end for said exercise). This is your playerbase lifetime value (note not Player Lifetime Value). To get Lifetime Value, divide daily revenue by the number of daily active users, and multiply that by the Lifetime Value to get your player LTV.

It's important to note that while calculating player lifetime value, the term is not entirely accurate since most player lifetimes are not over (particularly true for live service games). But for the purpose of modeling, we recommend keeping the amount of time that you consider a lifetime relatively short, allowing you to extrapolate. Keeping the time period shorter helps mitigate inaccuracies, specifically, the longer you stretch out what you consider a lifetime the more likely you are to collect inactive users in your count.

But these models are not entirely accurate since it doesn't take into account the players who are registered but have yet to generate any revenue. Instead, a data-driven approach pivoted around player segmentation or cohorts will generally yield more actionable insight, far more than calculating a single LTV for the entire player base.

You can define your game's cohorts in multiple ways. Perhaps the most obvious in terms of calculating LTV is going by daily active cohorts, or users who joined your game on the same day. You could also organize cohorts by users who joined your game through a certain ad campaign or promotional effort, by country or geographic location, or by the type of device used.

Lifetime Value solution accelerator →

Tips / Best Practices

- **Use multiple data sources:** To get a complete picture of a player's value, be sure to consider data from a variety of sources, including in-game purchases, ad revenue, and other monetization strategies.
- **Consider player retention:** Player retention is a key factor in LTV, so be sure to consider how long players are likely to play your game when calculating LTV.
- **Use accurate data:** Make sure you are using accurate data when calculating LTV. This might involve cleaning and processing your data, or using trusted sources such as in-game analytics tools.
- **Regularly review and update your LTV estimates:** Player LTV can change over time, so be sure to regularly review and update your estimates to ensure they are accurate.
- **Test and optimize:** Use experimentation methods such as A/B testing to see how different variables, such as in-game events or pricing strategies, affect LTV. Use the insights you gain to optimize your LTV calculations.
- **Be aware of outside factors:** Your calculations should consider the many outside factors that can affect your LTV, such as the virality of your game, any spikes or surge in visitors due to unexpected promotions (influencers, reviewers talking about your game), any significant changes to your game that users respond well to, and other organic lifts that are difficult to predict with existing data.



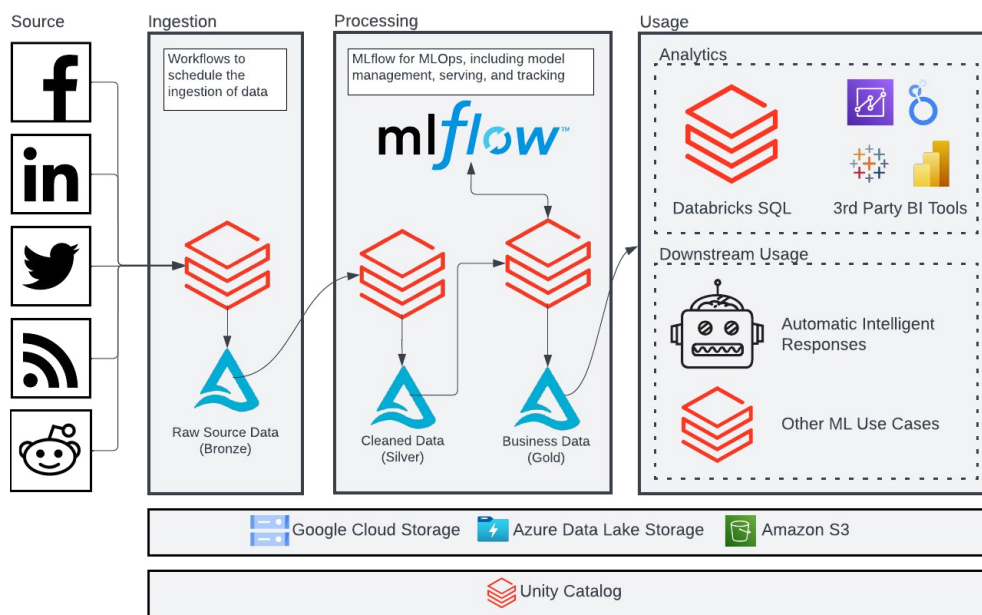
Getting Started with Social Media Monitoring

Social media monitoring has three primary components: collecting the data, processing the results, and taking action on the findings. When it comes to collecting the data, whether you're looking for tweets, YouTube comments, or Reddit posts, it can be very easy to get started since many social media platforms such as Twitter, YouTube, and Reddit all provide their own detailed and comprehensive APIs making it easy to start gathering data from those platforms with proper documentation and code examples to help along the way. Once the data has been collected, the next step is to process it and prepare it to be used in the next step. Processing your data can range in complexity from a simple keywords filter or more complicated approach such as filtering by location, removing emojis, and censoring and substituting words. With the data collected and processed, it can move to the final stage and be analyzed for downstream use and actionable insights by applying sentiment analysis or text mining.

If a game studio is looking to save time and have the above steps performed for them, it may be appealing to buy a pre-built tool. The primary benefits of buying an off the shelf solution is that it is often faster and easier to get started with, and the development of the tool is handled by a third party who will have experience in building media monitoring

solutions. On the other hand, building your own custom solution will provide more flexibility and control. Many pre-built media monitoring tools might not have the capabilities required to effectively process video, audio, and image data, and may not be able to control the frequency in which data is processed, whether it be near real-time or batch. Additionally, pre-built solutions tend to take a generalist approach for NLP, whether it be keyword extraction, topic filtering, or sentiment analysis, which often leads to poor results and feedback, especially for an industry as unique as the gaming industry where certain industry-specific slang or terminology is frequently used. Overall, building your own media monitoring tool will provide greater control and flexibility leading to a better tailored return on investment, and luckily Databricks makes it even easier to get started. With the Databricks Lakehouse platform, all data engineering, data science, machine learning, and data analytics can be done in a single place without having to stitch multiple systems and tools together.

Data engineers can use Workflows and Jobs to call social media platform APIs on a scheduled basis and use Delta Live Tables to create declarative data pipelines for cleaning and processing the data that comes in. Data scientists can use tools such as ML-specific Databricks runtimes (DBRs) that come with many of the most popular and common libraries already installed, MLflow which makes model development,



tracking, and serving easy and efficient, and various other tools such as AutoML and Bamboolib. Data analysts are able to create real-time alerts, dashboards, and visualizations using Databricks SQL. Each of the three personas will be able to effectively collaborate with each other and integrate each piece of their work into the broader data architecture.

If you have any questions about how this solution can be deployed in your environment, please don't hesitate to [reach out](#) to us.

Tips / Best Practices

While social media monitoring can be easy to get started with, there are a few key points to keep in mind.

- Remember the Pareto principle (roughly 80% of impact comes from 20% of activity) and diminishing returns. While it's important to monitor large platforms such as Reddit, Twitter, and YouTube, it might not be worthwhile to monitor smaller platforms (in terms of engagement) as the bulk of customer feedback will be on those major platforms.
- Monitor other sources of information. It is also useful to monitor mentions of key company personnel such as executives or public facing employees.
- While follower count does matter on platforms such as Twitter, don't ignore users with low-follower counts. It only takes one or two re-tweets from other users to become a large issue.
- On social media, customers can see through generic corporate responses to complaints, so it is important to get a clear understanding of the issue and provide a clear response.

Getting Started with Player Feedback Analysis

The easiest place to start is gathering your data. With accounts set up on Steam, Epic, Apple, Google, Xbox, Sony, Nintendo (or whatever platform you're using), identify the ID for your game(s), and pull the reviews corresponding to that game into Databricks through an API call.

From here, you clean the data using some of the pre-processing available in Python that removes any emojis and ASCII characters. Once complete, run through Spark NLP pipeline which does the basic natural language processing steps such as normalization, stemming, lemmatization. We recommend running through pre-trained models, such as Word Embeddings and Named Entity Recognition models from John Snow Labs. This should complete the pipeline and generates the aspects for the reviews provided by the community.

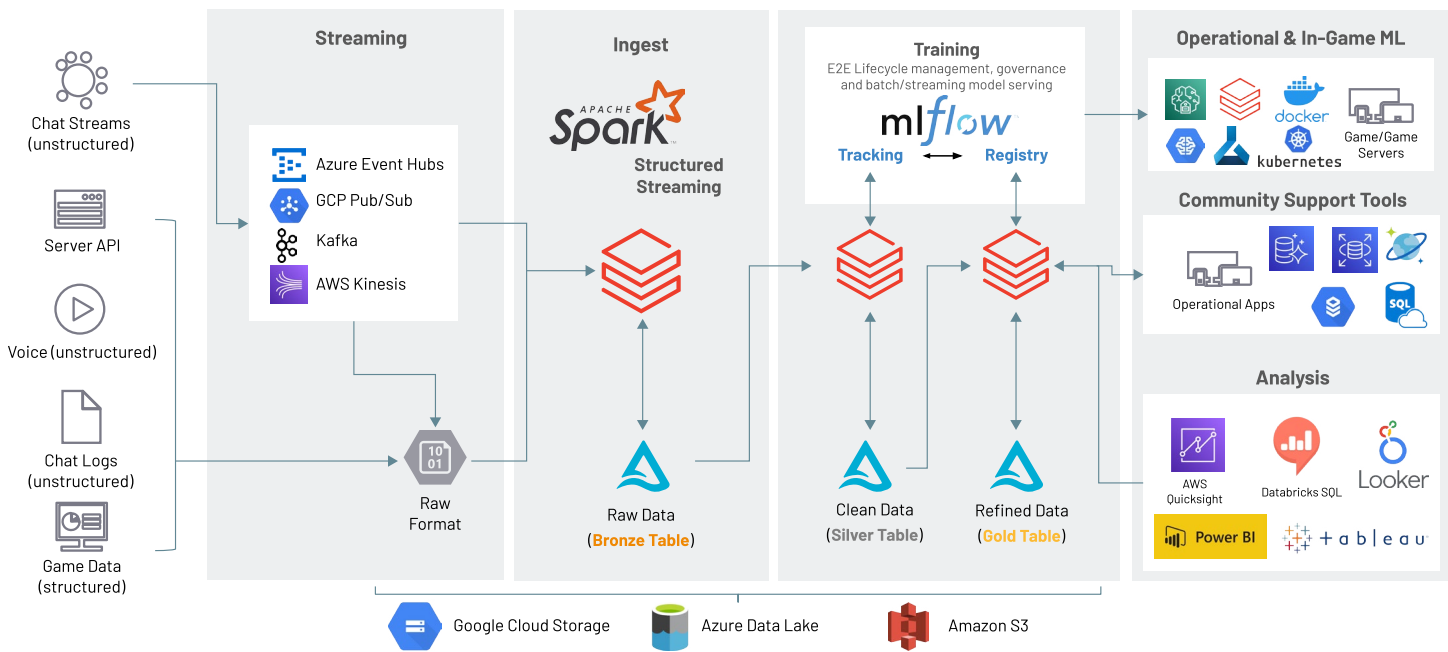
This data is then loaded into a Delta table for further analysis, such as using a visual dashboard (built on SQL queries inside Databricks) to analyze and understand the aspects the community is talking about, which can then be shared back with the development team for analysis and action. This is a great exercise to run once per month.

If you have any questions about how this solution can be deployed in your environment, please don't hesitate to [reach out](#) to us.

Tips / Best Practices

- **Check for word groupings:** Make sure your word groupings are accurate to improve the analysis. For example, if your game is called Football Manager, and the shorthand is FM, make sure both of those are grouped appropriately.
- **Leverage domain knowledge:** Clean the reviews based on your domain knowledge. There are generic steps one could take, but that will not be as effective as someone with domain, and specific game knowledge of your title.
- **Experiment with models:** Feel free to try multiple pre-trained models, and or tweak the pre-trained models based on your understanding of the domain to improve the accuracy of your results.
- **Work one title at a time:** This process works best when pulling reviews for a single title, specifically one version of one title at a time.
- **Let the model do the heavy lift, but use humans to double-check:** The sentiment corresponding to the aspects in the model will be labeled as Positive or Negative. In the case of a neutral review, the model will do its best to determine whether that is more positive or negative. A best practice is to spend time going back through the aspects early to determine model accuracy and make updates accordingly.





Getting Started with Toxicity Detection

Our recommendation on tackling the toxicity issue is to leverage cloud-agnostic and flexible tooling that can consume chat data from a variety of sources, such as chat logs, voice transcriptions, or sources like discord and reddit forums. No matter if the data is in log form from game servers or events from a message system, Databricks can provide quick and easy ways to ingest the data.

Leveraging a simplified architecture like the diagram above shows no matter the source, getting chat data for inferencing and model development can be as simple. While we leveraged a pre-built model from John Snow Labs to accelerate development, you can bring the ML framework of your choice to the platform.

[Gaming Toxicity solution accelerator](#) →

Tips / Best Practices – things to consider

- **Define what toxic and disruptive behavior looks like within your community:** Clearly define what you consider to be toxic behavior, as this will determine how you measure and detect it. This might include things like hateful language, harassment, or cheating.
- **Collect relevant data:** Make sure you are collecting the right data to help you detect toxicity. This might include data on in-game chat, player reports, and other sources.
- **Use machine learning:** Use machine learning algorithms to analyze your data and identify patterns of toxic behavior. This will allow you to more accurately detect toxicity and prioritize cases for review.
- **Test and optimize:** Regularly review and test your toxicity detection systems to ensure they are accurate and effective. Use experimentation methods such as A/B testing to see how different strategies impact toxicity rates.
- **Be transparent:** Make sure you are transparent with your players about how you are detecting toxicity, and give them the option to opt out if they wish.
- **Take action:** When toxic behavior is detected, take appropriate action to address it. The health and wellness of your community depends on it. This might involve banning players, issuing warnings, or taking other disciplinary measures.



Getting Started with Multi-Touch Attribution and Media Mix Modeling

To get started with multi-touch attribution, you need to first select an attribution model. There are a variety of different attribution models to choose from, each with its own strengths and limitations.

1. **Last-click model:** This model attributes all credit to the last touchpoint that the customer interacted with before making a purchase or taking a desired action.
2. **First-click model:** This model attributes all credit to the first touchpoint that the customer interacted with.
3. **Linear model:** This model attributes equal credit to each touchpoint that the customer interacted with.
4. **Time decay model:** This model attributes more credit to touchpoints that are closer in time to the purchase or desired action.
5. **Position-based model:** This model attributes a portion of the credit to the first and last touchpoints, and the remainder is distributed evenly among the other touchpoints.
6. **Custom model:** Some businesses create their own attribution model based on specific business needs or goals.

Each attribution model has its own strengths and limitations, and the right model for a particular video game will depend on a variety of factors, including the goals of your title, the customer journey, and the types of marketing channels being used. It is important to carefully consider the pros and cons of each model and choose the one that best aligns with the needs of your game.

Next, you're going to want to set up tracking. In order to attribute credit to different touchpoints, you'll need to set up tracking to capture data on customer interactions. This might involve integrating tracking code into the game, or using a third-party tracking tool.

With tracking set up, you'll start collecting data on player interactions and be able to use that information to calculate

attribution credit according to your chosen model (above). We highly recommend you regularly review and test your attribution efforts to ensure they are accurate and effective. Use experimentation methods such as A/B testing to see how different strategies impact conversion rates.

[Multi-Touch Attribution solution accelerator →](#)

Tips / Best Practices – things to consider

- **Define clear goals:** Sounds simple, but by clearly defining the goals of your acquisition campaign and what success looks like, you will be able to guide your decision-making and ensure that you are measuring the right metrics – such as cost per install, return on ad spend, conversion rate, lifetime value, retention rate, and more.
- **Use a data-driven approach:** Use data to inform your decision-making. Collect data on all touchpoints in the player journey, including ad impressions, clicks, installs, and in-game actions.
- **Choose the right attribution model:** Select the right attribution model that accurately reflects the player journey for your specific genre of game. This can be a complex process. A couple of things to keep in mind
 - Consider the touchpoints that are most important for your player journey, such as first ad impression, first click, or first in-game action
 - Consider the business goals you're trying to achieve. For example, if you are focused on maximizing return on investment, a last-click attribution model may be most appropriate. On the other hand, if you are looking to understand the impact of each touchpoint, a multi-touch attribution model may be more appropriate.
 - Consider the data you have available, including ad impressions, clicks, installs, and in-game actions.
- **Continuously monitor and optimize:** Continuously monitor and optimize your acquisition campaigns based on the data. Test different approaches, make adjustments as needed, and use A/B testing to determine what works best.



Getting Started with Player Recommendations

Recommendations is an advanced use case. We don't recommend (hehe) that you start here, instead, we're assuming that you've done the work to set up your game analytics (collecting, cleaning, and preparing data for analysis) and that you've done basic segmentation to place your players in cohorts based on their interests and behaviors.

Recommendations can come in many forms for video games. For this context, we're going to focus on the wide-and-deep learning for recommender systems, which has the ability to both memorize and generalize recommendations based on player behavior and interactions. First [introduced by Google](#) for use in its Google Play app store, the wide-and-deep machine learning (ML) model has become popular in a variety of online scenarios for its ability to personalize user engagements, even in 'cold start problem' scenarios with sparse data inputs.

The goal with wide-and-deep recommenders is to provide an intimate level of player understanding. This model uses explicit and implicit feedback to expand the considerations set for players. Wide-and-deep recommenders go beyond simple weighted averaging of player feedback found in some collaborative filters to balance what is understood about the individual with what is known about similar gamers. If done properly, the recommendations make the gamer feel understood (by your title) and this should translate into greater value for both the player and you as the business.



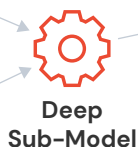
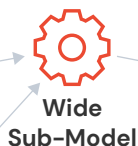
User A

- user identity
- user attributes



Product B

- product identity
- product attributes



**Probability of
User A + Product B**

Understanding the model design

To understand the concept of wide-and-deep recommendations, it's best to think of it as two separate, but collaborating, engines. The wide model, often referred to in the literature as the linear model, memorizes users and their past choices. Its inputs may consist simply of a user identifier and a product identifier, though other attributes relevant to the pattern (such as time of day) may also be incorporated.

The deep portion of the model, so named as it is a deep neural network, examines the generalizable attributes of a user and their choices. From these, the model learns the broader characteristics that tend to favor user selections.

Together, the wide-and-deep submodels are trained on historical product selections by individual users to predict future selections. The end result is a single model capable of calculating the probability with which a user will purchase a given item, given both memorized past choices and generalizations about a user's preferences. These probabilities form the basis for user-specific rankings, which can be used for making recommendations.

Building the model

The intuitive logic of the wide-and-deep recommender belies the complexity of its actual construction. Inputs must be defined separately for each of the wide-and-deep portions of the model and each must be trained in a coordinated manner to arrive at a single output, but tuned using optimizers specific to the nature of each submodel. Thankfully, the [Tensorflow DNNLinearCombinedClassifier estimator](#) provides a pre-packaged architecture, greatly simplifying the assembly of an overall model.



Training

The challenge for most teams is then training the recommender on the large number of user-product combinations found within their data. Using [Petastorm](#), an open-source library for serving large datasets assembled in Apache Spark™ to Tensorflow (and other ML libraries), one can cache the data on high-speed, temporary storage and then read that data in manageable increments to the model during training. In doing so, we limit the memory overhead associated with the training exercise while preserving performance.

Tuning

Tuning the model becomes the next challenge. Various model parameters control its ability to arrive at an optimal solution. The most efficient way to work through the potential parameter combinations is simply to iterate through some number of training cycles, comparing the models' evaluation metrics with each run to identify the ideal parameter combinations. By trials, we can parallelize this work across many compute nodes, allowing the optimizations to be performed in a timely manner.

Deploying

Finally, we need to deploy the model for integration with various retail applications. Leveraging [MLflow](#) allows us to both persist our model and package it for deployment across a wide variety of microservices layers, including Azure Machine Learning, AWS Sagemaker, Kubernetes and Databricks Model Serving.

While this seems like a large number of technologies to bring together just to build a single model, Databricks integrates all of these technologies within a single platform, providing data scientists, data engineers & [MLOps](#) Engineers a unified experience. The pre-integration of these technologies means various personas can work faster and leverage additional capabilities, such as the [automated tracking](#) of models, to enhance the transparency of the organization's model building efforts.

To see an end-to-end example of how a wide and deep recommender model may be built on Databricks, please check out the following notebooks: [Get the notebook](#)

Recommendation Engines solution accelerator →

Tips / Best Practices – things to consider

- **Use data to inform recommendations:** Use data from your analytics, player feedback, and other sources to understand what players like and dislike. This will help you create recommendations that are more likely to be relevant and engaging for individual players.
- **Segment your players:** Consider segmenting your players based on characteristics such as playstyle, spending habits, and demographic information. This will allow you to create more targeted recommendations for different groups of players.
- **Consider the player's current context:** When creating recommendations, consider the player's current context, such as what they are doing in the game and what content they have already consumed. This will help you create recommendations that are more likely to be relevant and timely.
- **Test and optimize your recommendations:** Use experimentation methods such as A/B testing to see how different recommendations perform with different player segments. Use the insights you gain to optimize your recommendations.
- **Be transparent:** Make sure you are transparent with players about how you are creating recommendations and give them the option to opt out if they wish.
- **Use recommendations to improve the player experience:** Use personalized recommendations to improve the player experience and increase engagement and satisfaction.

Getting Started with Next Best Offer/Action

Since NBO/NBA is a specific use case of personalization, how a team might get started implementing this will look very similar to how they would with broader personalization activities.

Begin with ensuring you are appropriately collecting player data (behavior, preferences, in-game purchases, etc), storing it in your cloud data lake using a service such as Delta Lake from Databricks. From here, you'll prepare the data using Databricks to clean, transform, and prepare for analysis. This may include aggregating data from multiple sources, removing duplicates and outliers, and transforming the data into a format suitable for analysis. As you analyze the player data, seek to identify patterns and trends in player behavior



and preferences that will give you signal on which actions are more likely to be successful.

From here, you can build a recommendation model based on the player data analysis, and incorporate information on in-game items and player preferences to make personalized recommendations.

If you have any questions about how this solution can be deployed in your environment, please don't hesitate to [reach out](#) to us.

Tips / Best Practices

- **Define your goals:** Like every use case, starting with clearly defined goals helps to ensure your implementation of NBO and NBA will be as effective and efficient as possible. Your goals will also help you determine what data to collect and how it will be used.
- **Collect relevant data:** Based on your goals, make sure you are collecting the right data to inform your NBO and NBA recommendations. This might include data on player behavior, engagement, and spending habits.
- **Leverage machine learning to scale your recommendations:** Use machine learning algorithms to analyze your data and make personalized recommendations to your players. This will allow you to identify trends and patterns that might not be immediately apparent.
- **Test and optimize:** THIS IS CRITICAL. Use experimentation methods such as A/B testing to see how different recommendations perform with different player segments. Past performance is not a perfect indicator of future success. Consistent testing allows you to tune your NBO and NBA recommendations so they evolve with your playerbase.
- **Consider the player's context:** When making recommendations, consider the player's current context, such as what they are doing in the game and what content they have already consumed. This will help you create recommendations that are more likely to be relevant and timely.
- **Be transparent:** Make sure you are transparent with your players about how you are using their data to make recommendations, and give them the option to opt out if they wish.
- **Collaborate with your team:** Share your NBO and NBA efforts with your team and encourage them to use the data to inform their work.

Getting Started with Churn Prediction & Prevention

The exciting part of this analysis is that not only does it help to quantify the risk of customer churn but it paints a quantitative picture of exactly which factors explain that risk. It's important that we not draw too rash of a conclusion with regards to the causal linkage between a particular attribute and its associated hazard, but it's an excellent starting point for identifying where an organization needs to focus its attention for further investigation.

The hard part in this analysis is not the analytic techniques. The Kaplan-Meier curves and Cox Proportional Hazard models used to perform the analysis above are well established and widely supported across analytics platforms. The principal challenge is organizing the input data.

The vast majority of subscription services are fairly new as businesses. As such, the data required to examine customer attrition may be scattered across multiple systems, making an integrated analysis more difficult. Data Lakes are a starting point for solving this problem, but complex transformations required to cleanse and restructure data that has evolved as the business itself has (often rapidly) evolved requires considerable processing power. This is certainly the case with the KKBox information assets and is a point noted by the data provider in their public challenge.

The key to successfully completing this work is the establishment of transparent, maintainable data processing pipelines executed on an elastically scalable (and therefore cost-efficient) infrastructure, a key driver behind the [Delta Lake pattern](#). While most organizations may not be overly cost-conscious in their initial approach, it's important to remember the point made above that churn is a chronic condition to be managed. As such, this is an analysis that should be periodically revisited to ensure acquisition and retention practices are aligned.

To support this, we are making the code behind our analysis available for download and review. If you have any questions about how this solution can be deployed in your environment, please don't hesitate to [reach out](#) to us.

Churn Prediction solution accelerator →



Tips / Best Practices

- **Define churn:** Clearly define what you consider to be player churn, as this will determine how you measure and predict it. For example, you might consider churn to be when a player stops playing your game for a certain number of days, or when they uninstall it.
- **Collect relevant data:** Make sure you are collecting the right data to help you predict and prevent churn. This might include data on player behavior, engagement, and spending habits.
- **Use machine learning:** Use machine learning algorithms to analyze your data and predict which players are at risk of churning. This will allow you to identify trends and patterns that might not be immediately apparent.
- **Test and optimize:** Use experimentation methods such as A/B testing to see how different strategies impact churn rates. Use the insights you gain to optimize your churn prevention efforts.
- **Focus on retention:** Implement retention strategies that are tailored to the needs and preferences of your players. This might involve providing personalized content, addressing pain points, or offering incentives to continue playing.
- **Be transparent:** Make sure you are transparent with your players about how you are using their data to predict and prevent churn, and give them the option to opt out if they wish.
- **Collaborate with your team:** Share your churn prediction and prevention efforts with your team and encourage them to use the data to inform their work.

Getting Started with Real-time Ad Targeting

Typically, implementing a real-time ad targeting strategy begins outside of your game (in services such as Google Ads, Unity Advertising), where your game becomes the delivery point for the advertisement. Here, you will need to integrate with Ad networks that provide real-time ad targeting capabilities. That will allow you to access a range of available ad assets to dynamically select and display the most relevant ads to players. Both Google AdMob and Unity Ads are great for banner ads, native ads, and rewarded video ads. Your role is to ensure that the data you're collecting is fed back into the advertising platform to better serve targeted ads to your playerbase.

To use a service like Databricks to manage the data needed to provide real-time ad targeting in your application, you can follow the below steps:

1. **Collect and store player data:** Collect data on player behavior, preferences, and demographics, and store it in a data lake using Databricks. Popular analytics tools such as Google Analytics or Mixpanel can be integrated into the game to collect data on player behavior. These tools, just like tracking website traffic, can track in-game events, provide insights on player behavior and demographics.. and they give you access to detailed reports and dashboards. Another option is to build in-house tracking systems to collect data on player behavior – logging events, e.g in-game purchases or player actions, activities such as “at which level does a player quit playing” and storing this in a database for analysis. The downside of building in-house tracking systems is you will need to host and maintain your own logging servers.
2. **Prepare the data:** Use Databricks to clean, transform, and prepare the player data for analysis. This may include aggregating data from multiple sources, removing duplicates and outliers, and transforming the data into a format suitable for analysis.
3. **Analyze the data:** Use Databricks’ built-in machine learning and data analytics capabilities to analyze the player data and identify patterns and trends.
4. **Create audience segments:** Based on the analysis, use Databricks to create audience segments based on common characteristics such as interests, behaviors, and preferences.
5. **Integrate with the ad server:** When an ad opportunity presents itself within the game, a call is made to the ad server. This call includes information about the player, such as the audience segment that they belong to. The ad server then uses this information to decide what ad to deliver to the player.
6. **Monitor and optimize:** Use Databricks to monitor the performance of the ad targeting and make optimizations as needed, such as adjusting the audience segments or adjusting the targeting algorithms.

By using a service like Databricks to manage the data needed for real-time ad targeting, game developers can effectively leverage their player data to create more personalized and engaging experiences, increase revenue, and reduce churn.



If you have any questions about how this solution can be deployed in your environment, please don't hesitate to [reach out](#) to us.

Tips / Best Practices

- **Focus on player data:** Make player data the center of your targeting strategy by collecting and storing comprehensive information on player behavior, preferences, and demographics. Here, it's critical to ensure the game code data trackers are properly implemented in order to collect this data (see Game Analytics section for detail).
- **Segment your audience:** Create audience segments based on common characteristics such as interests, behaviors, and preferences, and use these segments to deliver targeted ads.

- **Test and iterate:** Continuously test and iterate your targeting strategy to refine your audience segments and improve targeting accuracy.
- **Balance relevance and privacy:** Balance the need for relevant, personalized ads with players' privacy by only collecting and using data that is necessary for targeting and obtaining player consent.
- **Monitor performance:** Regularly monitor the performance of your targeting strategy to ensure that it is delivering the desired results and make optimizations as needed.
- **Partner with the right ad platform:** Choose an ad platform that is well-suited to your needs and aligns with your goals, and work closely with them to ensure that your targeting strategy is delivering the best results.

Operational use cases

Anomaly Detection

First thing is to begin collecting the data, game server / client logs out of your project. Then consume this into Databricks Delta, to have a continuous anomaly detection model running. Focus this on key pieces of information you want to monitor, for example – for live service games, this is going to be infrastructure and network-related metrics such as Ping and Server Health (# of clients connected, server uptime, server usage, CPU/RAM, # of sessions, time of sessions).

Once the model is ingesting and tuned specifically for the metrics based on the information you have above. You would build out alerts or notifications based on these specific metrics hitting a threshold that you define as needing attention. From here, you can build out automated systems to mitigate those effects – such as migrating players to a different server, canceling matches, scaling infrastructure, creating tickets for admins to review.

If you have any questions about how this solution can be deployed in your environment, please don't hesitate to [reach out](#) to us.

Tips / Best Practices

- **Define the problem and objectives clearly:** Before implementing an anomaly detection solution, it is important to define the problem you are trying to solve and your specific objectives. This will help ensure that you have the right data sources and use the appropriate algorithms to achieve your goals.
- **Choose the right data sources:** To effectively detect anomalies, you need to have the right data sources. Consider data from player behavior, system performance, and network traffic, as well as any other data sources that are relevant to your problem and objectives.
- **Clean and preprocess the data:** To ensure that the data you use for anomaly detection is accurate and meaningful, it is important to clean and preprocess the data. This includes removing any irrelevant or invalid data, handling missing values, and normalizing the data if necessary.
- **Choose the right algorithms:** There are many algorithms that can be used for anomaly detection, including statistical methods, machine learning algorithms, and rule-based systems. Choose the algorithms that are best



suited to your data and problem, and that provide the right level of accuracy, speed, and scalability.

- **Validate the results:** Before deploying the anomaly detection solution in production, it is important to validate the results by testing the solution on a small subset of data and comparing the results to expected outcomes.
- **Monitor and update the solution:** Once the anomaly detection solution is deployed, it is important to monitor its performance and accuracy, and update the solution as needed. This may include retraining the algorithms, adding or removing data sources, and updating the parameters and thresholds used by the algorithms.

Additionally, there are some key gotchas to look out for when implementing an anomaly detection solution.

- **Avoid overfitting:** Overfitting occurs when the anomaly detection solution is too complex and learns the noise in the data rather than the underlying patterns. To avoid overfitting, it is important to choose algorithms that are appropriate for the size and complexity of the data, and to validate the results using a separate test dataset.
- **False positive and false negative results:** False positive and false negative results can occur when the anomaly detection solution is not properly calibrated, or when the solution is applied to data that is significantly different from the training data. To minimize the risk of false positive and false negative results, it is important to validate the results using a separate test dataset, and to fine-tune the parameters and thresholds used by the algorithms as needed.
- **Scalability:** Scalability can be a concern when implementing an anomaly detection solution, especially when dealing with large amounts of data. To ensure that the solution can scale to meet the demands of a growing player base, it is important to choose algorithms that are fast and scalable, and to deploy the solution using a scalable infrastructure.

Getting Started with Build Pipeline

An operational goal game projects have is to make sure game project builds are generated, delivered quickly and efficiently to internal testing & external users.

A few of the key metrics and capabilities with analyzing your build pipelines are the below:

- **Build time and speed:** This includes metrics such as the time it takes to create a build, number of builds, and compute spent.
- **Build size and storage:** size of the builds, amount of storage, and network costs.
- **Bug tracking and resolution:** This includes metrics such as the number of bugs reported, the time it takes to resolve them, and the number of bugs that are resolved in each build.
- **Code quality and efficiency:** This includes metrics such as code complexity, code duplication, and the number of code lines written.
- **Collaboration and communication:** Such as the number of code reviews, the number of team meetings, and the number of code commits.
- **Advanced capabilities:** Such as Predicting real time build failure to reduce spend and combining build data with Crash Analytics (see below) to have “commit to build” visibility for accelerated bug fixing.

Before you start implementing your build pipeline, it’s important to define your requirements. What are the key goals of your build pipeline? Choosing the right CI/CD tools is critical to the success of your build pipeline. There are many different tools available, including Jenkins, Azure Devops, Perforce, gitlab and more. When choosing a CI/CD tool, consider factors such as ease of use, scalability, and cost. In addition, consider the specific needs of your game project, and choose a tool that can meet those needs.

The general recommendation is to look at automating your build process early. Once you’ve chosen your CI/CD tools, you can automate your build process by setting up a build server, configuring your CI/CD tool, and creating a script to build your game project. The build process should be automated as much as possible, and it should include steps to compile your code, run automated tests, and generate a build of your project.

Once you have automated your build process, often the next step is to implement CD (Continuous Delivery). This involves automating the deployment of your game builds delivery to stakeholders, such as QA testers, beta testers, or end-users via publishing platforms. CD can help ensure that stakeholders have access to the latest version of your game



as soon as possible, allowing them to provide feedback and help drive the development process forward.

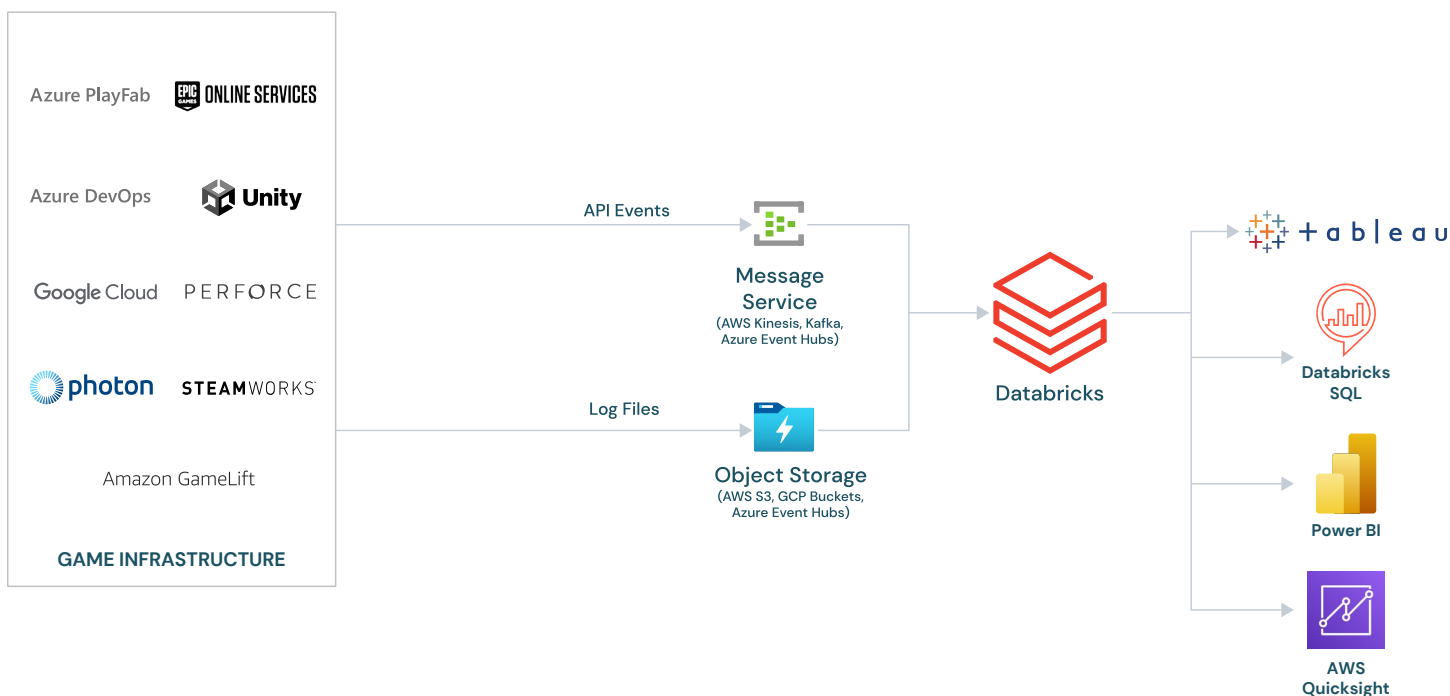
Finally, it's important to monitor and measure your build pipeline to ensure that it's working as expected. This can involve using tools such as Databricks Dashboards to visualize the status of your pipeline, or using metrics such as build times, test results, and deployment success rates to evaluate the performance of your pipeline. By monitoring and measuring your build pipeline, you can identify areas for improvement and make changes as needed to ensure that your pipeline continues to meet your needs.

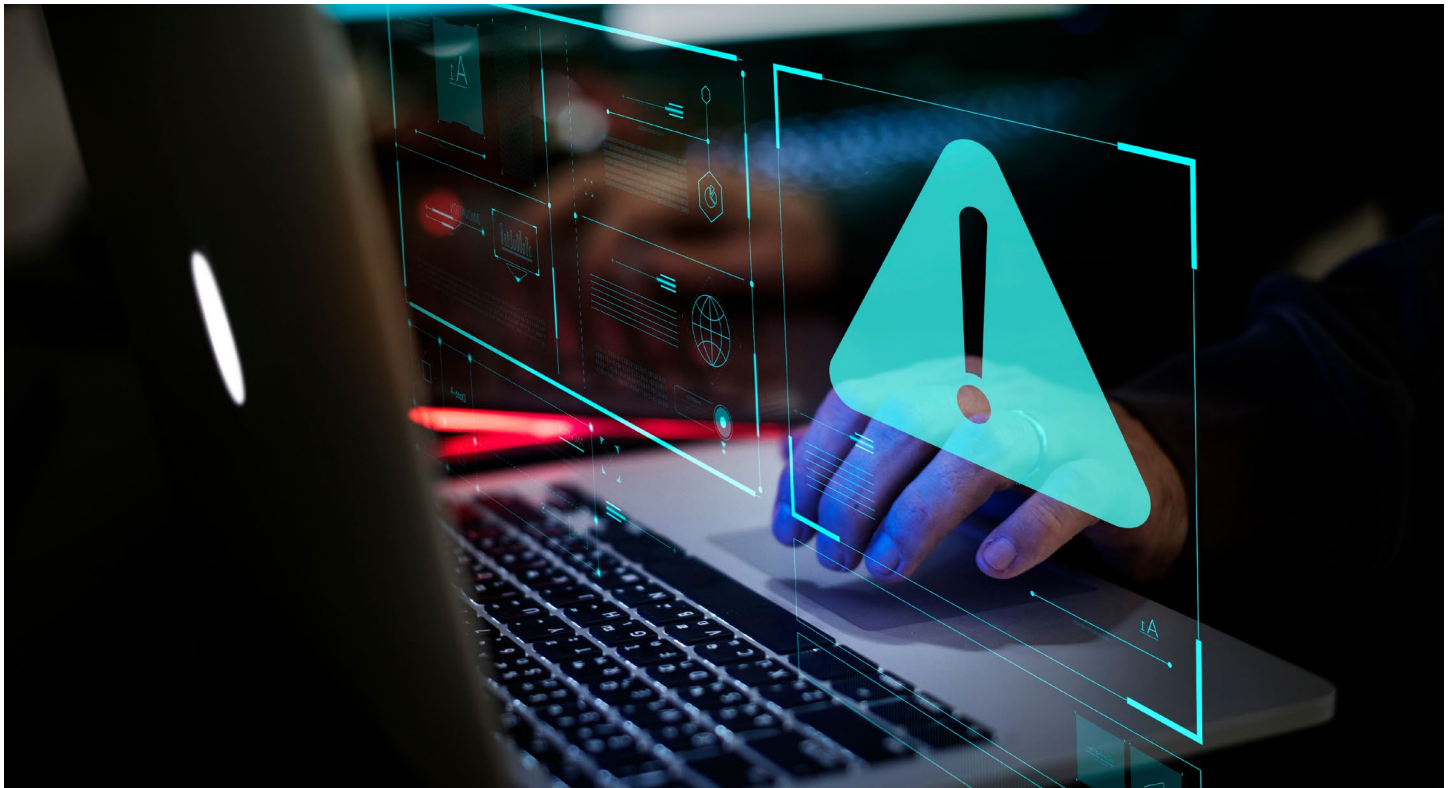
If you have any questions about how databricks can integrate into your devops solution, please don't hesitate to [reach out](#) to us.

Tips / Best Practices

- **Seek to automate early and often:** Automate as much of the build process as possible, from checking code into version control to generating builds and distributing them to stakeholders. This can help reduce errors and save time, allowing game teams to focus on more high value tasks.
- **Version control, version control, version control:** Use a version control system to manage the source code and other assets. This ensures that changes to the codebase are tracked and can be easily undone if needed.
- **Implement continuous integration and delivery:** Continuous integration (CI) involves automatically building and testing after code changes are checked into version control. With CI, new changes to the codebase do not break existing functionality. By automating the build process, CI helps to reduce errors and save time. CD, on the other hand, involves automatically delivering builds to stakeholders, such as QA testers, beta testers, or end-users, after they have passed the automated tests. By combining CI and CD, a video game project can ensure that builds are generated and delivered quickly and efficiently, without the need for manual intervention.
- **Build for scalability:** As your game project grows, you will need a build pipeline solution that is scalable and can handle the needs of your game team.
- **Integration with other tools:** Integrate the build pipeline solution with other tools and systems, such as issue tracking, testing, and deployment tools, to ensure a smooth and efficient workflow.

Reference Architecture





Getting Started with Crash Analytics

Building a pipeline to build a holistic view to support crash analytics means data coming from multiple different sources, different velocities and joining the data together.

The amount of data sources depends on your game projects publishing platforms, some may come from console based providers such as sony, xbox, and nintendo or pc platforms like Steam, Epic Games Marketplace, GoG and many others.

High level steps

- Determine what platforms your game is running on and how to interface to collect data.
- **Collect crash data:** Implement crash reporting tools in your game to collect data on crashes. The source data may be delivered in varying formats such as JSON or CSV.
- **Load crash data into Databricks:** Use Databricks' data ingestion tools to load the crash data into your workspace. This could involve using Databricks' built-in data source connectors, or programmatically ingest files to load the data.
- **Transform and clean the crash data:** Use Databricks' data processing and transformation tools to clean and prepare the crash data for analysis. This could involve using Databricks' capabilities like DLT, or using SQL to perform custom transformations.
- **Visualize crash data:** Use Databricks' dashboarding tools to create visualizations that help you understand the patterns and trends in your crash data. This could involve using Databricks' built-in visualization tools, or integrating with external visualization tools like Tableau or PowerBI.
- **Analyze crash data:** Use Databricks' machine learning and statistical analysis tools to identify the root causes of crashes. This could involve using Spark MLlib or many of the popular tools to build machine learning models, or using SQL to perform custom analyses.
- **Monitor and refine your pipeline:** Regularly review your pipeline to ensure that it remains relevant and useful. Refine your pipeline as necessary to reflect changes in your crash data or your goals.

If you have any questions about how this solution can be deployed in your environment, please don't hesitate to [reach out](#) to us.



Tips / Best Practices

- **Automated collection and aggregation of crash reports:** Collecting crash reports should be an automated process that is integrated into the output of the build pipeline for the game. The crash reports should be automatically aggregated and made available for analysis in near real-time.
- **Clear reporting and prioritization of issues:** The solution should provide clear reporting on the most common issues and allow game developers to prioritize fixing the most impactful problems first.
- **Integration with other analytics tools:** The crash analytics solution should integrate with other analytics tools, such as player behavior tracking, to provide a more complete picture of how crashes are impacting the player experience.
- **Flexibility and scalability:** As the game grows, the solution should be able to scale to accommodate an increasing number of players and crashes.
- **Data privacy and security:** It's important to consider data privacy and security when implementing a crash analytics solution. This may involve implementing measures to anonymize crash reports, or taking steps to ensure that sensitive information is not included in the reports.

Additionally, there are some key gotchas to look out for when implementing an anomaly detection solution.

- **Data privacy and security:** Ensure that crash reports do not contain sensitive information that could be used to identify individual players.
- **Scalability:** As the number of players and crashes increases, it may become difficult to manage and analyze the growing volume of data.
- **Integration with other tools:** Be aware when integrating crash analytics with other tools and systems, especially if the tools use different data formats or data structures.
- **Prioritization of issues:** Determine which crashes are the most impactful and prioritize fixes accordingly. This can be a complex process, especially if there are a large number of different crash types and causes.

Reference Architecture

