Cloudflare for Gaming provides a solid foundation for developers to deploy their game in a way that prioritizes user experience and security. However, in order for Cloudflare to become an end-to-end solution for its users – in that developers build, deploy, and scale their app using Cloudflare – we must dive deeper into how developers think.

1) What tools are developers using to build their games right now? Why are they using that platform, service, or tool?
2) What are the drawbacks to the server they're using now? What factors would they take into consideration before using Cloudflare Workers? Right now, what causes their games to fail?
3) Are there different types of developers? If so, how can we categorize them? What are each category's needs, and which segment can we serve the best?
4) What are emerging trends in the online video game industry? What is a wave we can capitalize on based on our current offerings?
5) Who are we competing against? Who provides similar services, and why are they succeeding? How are they different from us?

Given Cloudflare's current client base, they already may have access to a trove of information about medium to large sized developers – including answers to some of the questions above. Other segments of the market, which would be uncovered through more research, could include independent developers. Reaching out to the Independent Game Development Association on Twitter, finding successful developers on Reddit's r/gamedev thread, or even penning the frequent contributors to online communities/blogs would be excellent ways of finding information and points of contact. From research, interviews, and surveys, we could produce a persona for an aggregate and target the market based off of that. For the rest of the paper, I'll be giving recommendations based on a certain set of assumptions.

With 200+ data centers around the world, a robust CDN, and effective load balancing, Cloudflare has the solutions to many of the problems that multiplayer online games encounter when trying to scale. And with the rise in popularity of e-sports, almost every multi-player game is trying to build its fanbase & optimize for maximum competitiveness and playability in hopes of reaching "e-sport status" – a title that garners millions of spectators and billions of dollars in revenue. The speed at which developers can push updates, meaningful code fixes, and tweak aspects of a game largely dictate a game's success (or potential demise). The nature of the serverless platform already allows developers to accelerate their build-to-deploy time. But in order to improve their game and give it an edge above others in an already over-saturated market, developers must understand what works and what doesn't.

Games fail because they don't respond to their users fast enough.

Small details like changing the current level ordering from 4, then 5 to 5, then 4 may actually make the game more engaging for users. Testing takes the guesswork out of a lot of decisions – but usually, the time and cost required to actually test a game play decision probably outweigh the marginal benefits that a developer could receive. Cloudflare Workers for Gaming should have an established A/B testing guide for gaming. A basic Workers function assigns a new client to a "test" or "control" group, sets a response path for each group (two copies of the game's URL could be stored under different directories at the origin server), and triggers response based on assigned group. While this is usually used to test tweaks on static sites, the framework could be expanded on to facilitate a testing process for games. Some game development engines allow for the modularization of code into assets, so that developers can re-arrange or reuse assets to create scenes faster. Cloudflare could publish a set of game-friendly guides that help developers A/B test their code, learn how to cache user permissions to at the network edge to minimize latency when administering these tests, and down the line, help developers make content customized to a particular user type/geo-location.

The most important way developers can utilize A/B testing is through microtransaction placement. Some of the most common, household-name games - from League of Legends to Pokémon Go - utilize a free download & pay as you go model. Last year, revenues from free-to-play games made up 80% of online game revenues, and microtransactions are a huge part of this phenomenon, especially in e-sports. The scenes at which microtransactions are introduced, how offerings are packaged, even where they're placed on the screen all factor into gameplay experience & earnings – arguably the most important metric to an independent developer or publishing company.

Games fail because they can't monetize.

This is the biggest reason why indie developers fold, but also plays a major role in big players do too. Let's use the game Tribes Ascend as an example. A few years ago, Tribes Ascend was one of the rising players in the e-sports ecosystem – with a great playability-difficulty balance, rave reviews from critics, and a cult popularity. Almost inexplicably, it folded in two years. Why? Their production company later admitted that there were faults in their monetization structure – for example, they added too many "pay-to-win" options with overpowered weapons. Perhaps, testing paid weapon releases would have uncovered the reason for their declining user base.

In conjunction with the testing framework, could provide developers with an information or analytics dashboard built off of Cloudflare's GraphQL Analytics API. It would allow the developer to monitor usage and latency levels, transaction occurrences, and security threats. This would help developers effectively load balance, understand the results of their A/B tests, and respond faster to threats. Close microtransaction monitoring has the additional benefit of helping developers detect early signs of gold mining or credential stuffing.

An assumption I'm making is that developers aren't making applications in-house with Workers, but that they're using existing game engines to code and develop their games, probably with C++ and JS on engines like Unity, Unreal, Elm, etc. In offering the product to the market, I would provide extensive documentation around how to integrate Workers with their existing game, or provide a Software Developer Kit that makes onboarding to Workers as smooth as possible. I would also want to provide discounts or financial incentives for smaller developers to use our platform. While testing & micro-transaction placement could appeal to mid- to large-scale publishing companies who want to carve their niche in the e-sports ecosystem, these features could also largely appeal to a rising independent developer struggling with monetization. "Built with Workers" is great because it not only showcases people's interesting work, but it 1) demonstrates to users the potential use cases of Workers 2) adds to the feeling of community support with the platform 3) helps people learn how to use the product itself. By starting off as open-source nature, we're emulating the community aspect of the gaming industry in our product, demonstrating to mid-sized companies what the platform can do, and building a user base that we can utilize to garner feedback and improve our offering before a final release. And in order to roll out changes to the market as fast as possible, first iteration for Cloudflare Workers for Gaming should only include the SDK/ Documentation and a robust A/B testing framework, since these are the closest to the products we have currently. Based on feedback, the second iteration will include analytics provided to developers on purchasing & other game metrics measured through A/B testing. I would also bundle Cloudflare Workers for Gaming with discounts on Spectrum, Bot Management and Argo-smart Routing on its final release to give developers a more comprehensive package.

Metrics to Measure Success: # of Games that start using Cloudflare Workers, # of Games that stop using Cloudflare Workers, # of transactions, % increase in # of transactions per game, # of contributions to Built with Workers, Amt. of Spillover to other Cloudflare Products, Peak-time latency levels, security breaches, and qualitative feedback from developers

Risks: 1) Competitors: Although Cloudflare has been proven to be faster than its competitor Lambda@Edge by 192%, Amazon has an advantage in terms of overall tooling. Amazon DynamoDB and S3 allow nonrelational database storage that's utilized by multi-player games to store game data, like microtransactions. While clients could use Workers with an external database, other companies that provide a more comprehensive service for gamers could have higher retention rates. This risk could be mitigated by creating an entirely new Cloudflare product – a secure, distributed relational database (although the cost and timeline must be considered first), establishing partnerships, or pursuing a different direction for the product.
2) Market: Right now, the market is exploding with multi-player games, e-sports, and microtransactions. However, these trends could change in the future. In the future, security attacks could occur to such a degree that free-to-pay models won't even be used anymore. Additionally, multiplayer games and e-sports could grow to require such high capacity that current number of data centers aren't enough. Cloudflare may have to look

into increasing its number of data centers, dedicating certain servers entirely to gaming based on usage levels and latency levels. If cost doesn't permit, it may look into targeting the segment of independent developers.
3) Users: With my limited time for user research, I picked a segment that was interesting to me: mid to large sized gaming companies – with the popularity of e-sports and increasing competitiveness of the gaming market, I think that now is the time to capitalize on the fact that a lot of these companies could optimize their performance with Cloudflare Workers. However, another emerging customer segment is independent developers, which should be explored further before making product decisions.