Taking Cloudflare Workers for Gaming to Market.

By Dylan Robertson

Executive Summary:

I propose that Cloudflare targets the gaming market by expanding the existing Cloudflare worker (CFW) systems to create Cloudflare Workers for Gaming (CFWG). These expansions would include the ability to create custom calls to a worker, a number of pre-built templates ready to be implemented, and expansion of the community site. Once successful, Cloudflare stands to increase their market share against market rivals such as AWS Lambda, Microsoft Azure, and Google Cloud functions.

Market Research:

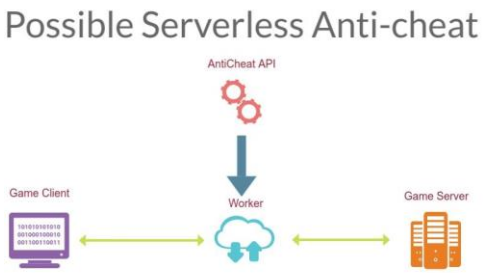
Based on my market research, I believe CFWG should focus on developing solutions for both indie developers and small development studios in the gaming market, such as Snowman and Voodoo. These developments would allow CFWG to become a part of the mobile gaming market, which makes up 45% or 68.5Bn. Going forward, more research and justification would need to be completed. As the product manager, I would conduct a number of interviews with smaller developers, and run surveys to collect qualitative data to help justify my hypothesis.

Market Analysis:

The Workers platform currently has a number of direct competitors. These include AWS Lambda, Google Cloud Functions, and Microsoft Azure Functions. These competitors currently hold a large market share above Cloudflare Workers. However, if successful, this development will help spread awareness and adoption of both the CFW and CFWG platforms helping the business gain market share.

Proposed Changes:

I am proposing two changes to the current CFW platform. Currently the CFW platform provides the ability to deploy serverless code to Cloudflare data centers across 200 cities in 90 countries. This can be improved on for gaming by offering the following changes:



1) The first change is allowing custom calls. Currently CFW can only run a FetchEvent on an HTTP call. If the team was able to design a system to handle customized calls, we could have our workers run between the client machines and the game servers. This opens up a number of possible solutions for game developers. One such solution would be the deployment of serverless anti-cheat. By having the worker catch the incoming packet and carry out a validation before passing it from the client to the server we could verify the input was accurate. This would be helpful in real time online games where there are currently two possibilities:
   - Client side anticheats (which are vulnerable to hacking and exploitation.)
   - Server side anticheats (which could slow down game servers, or limit the number of gamers per server.)

2) The second possible change is the development of a premade content delivery network (CDN) for mobile games through advertisement. Currently while playing most mobile games, when the ad event is triggered, it makes a call to the server to provide the most recent ad. With the proliferation of the mobile market, your user could be in Australia while your ad is hosted in the United States. This could have the user waiting up to 220ms. This can leave the end user waiting for several seconds increasing the odds that they will close the apps. If we developed a premade system that would allow game designers to cache ads on their workers, this could greatly reduce the amount of time that the end user spends waiting as the typical latency for a worker is about 28ms, and therefore helping the developers increase profit for the games they make.

Risk:
There are a number of risks with implementing CFWG. These include but are not limited to the following:

- Little community adoption within the developer world.
- Tech incapability.
- Direct competitors like AWS beating us to market or developing better solutions.

In order to mitigate these risks, it will be important for both the project manager and the rest of the team to be highly engaged with the customers, move quickly to deploy and reach the market, and make sure that our toolkits work with the most popular developer tools. Such as Unity, GameMaker, and CryEngine.

Beta Plans:
In order to have a successful launch it is important to test all of the team's designed systems, both for technique errors, and customer feedback. That is why I would recommend both an open and closed beta. These would allow the team to get crucial feedback both on the ideas and the implementation. In order to participate in the closed beta, we could request that the developers provide feedback and allow us to run tests to collect data.
After the closed beta feedback is processed, to verify that we have created the right tool set, we could invite all developers to use CFWG in an open beta. This would allow the team to stress test the new platform and would also allow the team to collect useful data about how the systems are being used and to gather user feedback.
The most important metric during this time is user feedback. In order to collect this, the team lead will need to conduct both interviews and surveys.

Goals:
It is important to have clearly described goals and measurements of success. Since this is a new product the initial goal should be to attract new users to the product and drive engagement with our services. This will allow us to pivot and monetize at a later date, after we have increased the value of the proposition by responding to the developers' wants. The key metrics to measure are listed below.

- Number of accounts linked to CFWG.
- Number of active workers deployed per account.
- Number of request workers handle per day.