Cloudflare Workers for Video Game Development

Serverless microservices for Game Developers

By Karan Gupta

There are many uses for Cloudflare's implementation of service workers, the Cloudflare Worker. By intercepting requests by the client before they are sent to the server (and vice versa) we introduce increased efficiency with triaging and caching as well as decreased attack surfaces with tokens instead of PII. However, these tools are not readily available to game developers; while AAA video game makers like Fortnite may have millions to pour into tool building, smaller development studios may be apprehensive to the idea of having to commit resources to building out complex netcode infrastructure. *Cloudflare Workers for Video Game Development* would be an Amazon Web Services-like portal listing with a wide range of tools. With so many different applications of Cloudflare Workers, a natural progression of this tech is to begin building the tools that go into this tool box with an end goal of a SaaS offering to integrate various microservices in between different clients and the server of a video game.

It is important to choose the correct tools to build first. With all the responses received from this prompt, there are a litany of ideas to implement instead of the ones provided below. We will explore how certain services are integrated, but note that in order to have a successful rollout, further testing and market validation is necessary to determine different pain points for game developers. After a catalogue of different problems have been identified, Cloudflare should test their product by offering a limited selection of tools. Based on feedback, tools are added, changed, or even deprecated as necessary. As the service gains popularity, revenue can be reinvested into expanding offerings. In the end, we expect an AWS-like experience: going to Route 53 for DNS, going to EC2 for instances, going to S3 for data storage, etc. Examples of *Cloudflare Workers for Video Game Development* tools could be entity triaging or game stores.

The experience of most online video games is affected by the player's ability to respond to an event as effectively as possible. Games with large maps and lobby sizes will have more data to process. In a new MMORPG FPS, *Cloudflare Workers for Video Game Development* offers an entity triaging tool: a service worker sitting between the server and client will intercept a response by the server and prioritize the transmission of different data. This is configured to send player characters first, hostile non-player characters second, and peaceful non-player characters last. Especially when clients are dealing with limited bandwidth, this mechanic allows the client to more readily render threats to the player, thus resulting in a quicker response time. This could be measured by tracking the number of client ticks between the client's initial request and the loading of entities. Once validated, the overhead for this tracking could be removed to further enhance efficiency.

Video games have become a huge economy with an accelerating trend of developers turning to downloadable content to tap into new sources of revenue. Most game developers also have to cut in the distribution platform, like Steam or Origin. For many developers, the exposure and audience they reach by using these platforms offsets a 30% cut of all in-game transactions. However, it isn't uncommon to see a game go viral pre-launch or to have a following from a crowdfund. These game developers may decide that using a distributor is too costly, and decide to cut out the middle man. The developer has built a store for their in-game economy and wants to use a service like Stripe or Plaid to allow users to shop. This traditionally means that the developer is now in possession of PII, exposing the development house to many compliance requirements. *Cloudflare Workers for Video Game Developers* steps in by offering a payment tokenization service- rather than processing the payment on the server side, our Cloudflare Worker intercepts the request by the client before it goes to the server. From here, card information is transmitted directly to the Stripe/ Plaid API, which returns a tokenized receipt. The service worker then sends this token to the server. As a result, we are able to process payments without a distribution platform and without handling or storing PII.

The largest risk to the success of this service is failure to build the right tools. It is already evident that service workers offer a unique power to developers; it is a matter of targeting the right pain points and building the first tools around those assumptions. Building the wrong tools will result in slow (if any) growth while competitors analyze Cloudflare Workers and implement advanced changes in their own products.