Ratelimiting Options

The idea is to offer a way for Clowder admins to configure ratelimits to the platforms that they manage.

Base Implementation Options

Here are some options for how to go about implementing rate limits in Clowder from a high-level.

Option 1: NGINX Ratelimiting

See https://www.nginx.com/blog/rate-limiting-nginx/ for an example of how to configure these limits

This seems like the most obvious approach, given that NGINX is already along the central entrypoint of traffic into the Clowder ecosystem.

Although NGINX configuration is notoriously complex, they do offer ratelimiting options there.

Pros:

- baked into NGINX by default
- would require no modifications to Clowder source code

Cons:

- NGINX configuration is notoriously complex

Option 2: Clowder API Ratelimiting

See https://github.com/sief/play-guard for a possible example of a generic plugin for handling this

This seems like a good idea if we can find a generic way to implement it.

That is, this is ***not*** a feasible approach if we need to add custom code to every API endpoint.

Pros:

- There may already be some generic plugin(s) we can leverage to handle this - this could (at the very least) serve as a pattern for how we could accomplish the same

Cons:

- Ratelimits would be on a per-instance basis - e.g. if you are running 4 replicas of Clowder, a user who is ratelimited on one machine could repeat their request to reach another machine where they are not ratelimited (is this acceptable?)

~~Option 3: Route53 Ratelimiting?~~

I looked into applying the ratelimit one level above NGINX, going to Amazon's Route53 which can be used to handle DNS resolution.

Sadly, it looks like Route53 applies its own ratelimiting, which is not necessarily configurable or even exposed to the user.

Error-Handling Options

From Google:

> *The **HTTP 429 Too Many Requests** response status code indicates the user has sent too many requests in a given amount of time ("rate limiting").*
>
> *A Retry-After header might be included to this response indicating how long to wait before making a new request.*

There are some outliers - for example, Google Drive uses **403 Forbidden** instead of 429. This could be confusing to users, as we already use 403 to indicate that the user lacks permission to perform the operation.

My recommendation would be to stick with 429 as an error code for ratelimiting because it is commonly used to indicate that a user is ratelimited, and to differentiate it from Forbidden (aka "Permission Denied").

Other Concerns

- How can we tell the difference between a user being ratelimited vs being blocked by the firewall (HTTP 429 should only be a symptom for ratelimit?)