Fix three: build your own proxy

The fix I recommend in situations like this, is to build your own proxy! Exactly like the previous solution, you're utilizing the fact that the same origin policy is not enforced within server-to-server communication. In addition, you eliminate the latency concern. You don't need to share the cors-anywhere proxy with other consumers, and you can dedicate as many resources as you need to your own servers.

Here's some quick Node.js code that uses the express web framework to create a proxy server around the same `https://joke-api-strict-cors.appspot.com/` from above:

If you want to see this in action, head to the source code for the above, along with relevant steps in the README: https://github.com/15Dkatz/beat-cors-server

How does this work? The proxy uses express middleware to apply a `Access-Control-Allow-Origin: *` header to every response from the server. At its own `jokes/random` GET endpoint, the proxy requests a random joke from another server. The same-origin policy doesn't step in to block the request, even though the domains are different. After all, this is a server-to-server request. Finally, the proxy creates a response to the original requester (an app on the browser) consisting of the resulting data and the middleware-applied `Access-Control-Allow-Origin: *` header.