written by Garry Wilson on 1 December 2014 in Amazon, devops

moving the endpoints

In my previous post, I talked about how we're moving away from Apache, and embracing Nginx for Atlas and Voila — our two main API endpoints. Doing so is no quick or easy task; we've put a lot of effort into optimising the Nginx configuration that we'll be using to make sure it can do everything Apache is currently doing, but *better*.

Atlas and Voila are busy APIs. We average over 12 million requests between them each day, and both services are covered by comprehensive SLAs to various clients, where we offer some guarantees against expected response time of a request and what uptime we strive to maintain.

So, given that we have to ensure Atlas and Voila are available around the clock, and perform quickly, how do we manage to completely scrap the web servers allowing that to happen in our move to Nginx?

cheap, unlimited testing infrastructure

To allow us to test the changes, we first spun up a disposable EC2 instance for Nginx, where we could play around with the jfryman Puppet module for Nginx mentioned in the previous post. It's also where we worked out the bugs around Dnsmasq, and the settings we'd need to resolve IP addresses properly. We got a basic but working vhost template that could do in Nginx what we were currently doing in Apache.

Once we were happy the basic config was stable and not going to cause an issue alongside an Apache install, we moved to our stage Voila servers, and then to production.

elb - elegant load balancing

The basic config that we put on our stage and production environment had one small difference; we'd specified different listening ports in Nginx, so as not to conflict with the Apache service that was in place. We could then also add those same ports to our production ELB — in other words, we can run testing of our draft Nginx configuration alongside live production traffic to Apache on 'real' hardware. This allows us to simulate what would happen on the production hardware with little risk.

From then, we simply call voila.metabroadcast.com:81/1.0/schedules/, where 81 is the port Nginx is set to listen to. We ran load testing and checked every possible scenario and rewrite rule to verify our configuration was correct, and matching what was being returned on the default Apache port 80.

Once all the testing was done, we worked to have Nginx log in a way that could be handled by our existing logging infrastructure. This will help us evaluate how Nginx is performing over time, now that we're putting it live to production Voila traffic. We'd love to hear your views about the newly Nginx'd Voila, and if you notice any differences — get in touch on Twitter to let us know!