API Management Tools

Minnesota API Users Group - May 2016

Presented by Eric Caron

Why do we need API management tools?

If you have an API consumed by more than 1 person, at some point you'll:

- Want to know who they are and what they use
- Want to restrict access to certain endpoints to certain people
- Want to throttle requests by the second, hour, day, lifetime, etc
- Get tired of handling versions in each individual application
- Have multiple applications sharing the same hosts, userbase, throttles, etc
- Want to strip certain information in the request, or in the response

You could certainly extend your application to handle all of these needs.
You could also extend your application to exist as a Linux kernel module...

What solutions are available?

The solutions fall into 2 buckets:

- Someone hosts it for you
  - Costs more (since they run the hardware), but they support it and maintain it
  - Examples: Mashery, MuleSoft, AWS API Gateway, Apigee, Intel Expressway
- You host it yourself
  - Costs less (but you have to pay for the hardware), and there are different models for who does support and how. Everything from 100% free and you're on your own, to they do everything for you except supply power to the server
  - Examples: Kong, Tyk, Apigee, StrongLoop, API Umbrella

There are dozens more in each variety. You have to figure out what you need and what you're willing to spend.

High level overview

- Users have access credentials (keys)
- Keys belong to one-or-more packages
- Packages have one-or-more routes
- Routes are what tells the tool how to serve the request
    - It is at this point you can think if it acting like a web server, such as nginx or Apache.

```
server { # simple reverse-proxy
  listen       80;
  server_name  domain2.com www.domain2.com;
  access_log   logs/domain2.access.log  main;

  # serve static files
  location ~ ^/(images|javascript|js|css|flash|media|static)/  {
    root    /var/www/virtual/big.server.com/htdocs;
    expires 30d;
  }

  # pass requests for dynamic content to rails/turbogears/zope, et al
  location / {
    proxy_pass      http://127.0.0.1:8080;
  }
}
```

Other lessons from the real world

- If you're paying a provider for hosting the solution, they're probably offering caching. And that makes everyone happier.
- Make sure you own your userbase. And you can update your user's information on their behalf.
- If you're paying a provider, they should be a CDN & WAF (web application firewall). If you're hosting yourself, determine your CDN & WAF strategy.
- Weigh the pros-and-cons when having a "developer portal" and who hosts/manages/owns it
- Be intentional of the line between the management tool and your application (e.g. does the application control the content of error messages, or the tool)
- Your servers should only accept requests from the provider's whitelist of IPs

Why I'm going to show you Kong now

- At my current job, I attract an uncomfortable amount of attention from people selling things. Mashape reached out to me last June and I ignored them.
- In September I evaluated the marketplace, and found Kong (not recognizing they approached me months earlier)
  - http://management.apievangelist.com/organizations/
- In December The Changelog podcast featured the story behind Kong
- The Guardian moved to Kong and seem to be very happy with it
- It is built on NGINX, OpenResty (which is LUA) and Cassandra. So it is really crazy fast
- Although they're open source, they have corporate backing (so I can pay for features or support when I don't want to tackle it myself)