MAKING SURE THE MOST IMPORTANT LAYERS OF API SPACE STAY OPEN

APIs are a really tough concept to help folks on-board with. I have come to realize that people see APIs very differently, and most of the time, I think this can be a good thing. In a perfect world it would be nice if everyone followed the same standards, and used only the best API patterns, when designing their APIs. This isn't the world we have, and to help better engge within the world we have, I am working to tell stories of this dysfunctional but beautiful API space we have accidently created for ourselves, develop visuals that help us better quantify things, and potentially give little nudges where I can to move things in a better direction.

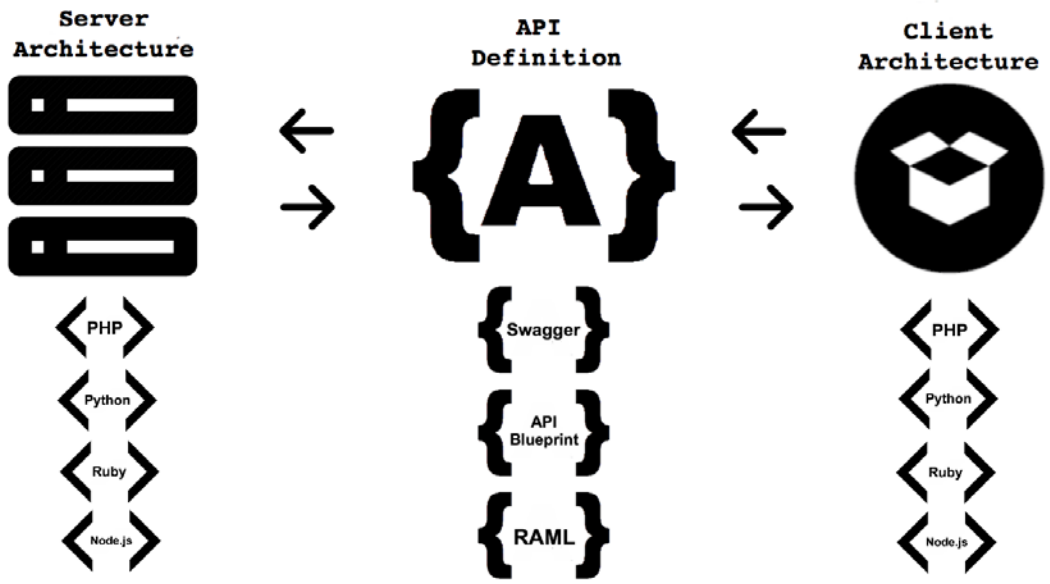
When I hear most technologists talk about APIs, I feel many are exclusively looking through a client / server lens. Your APIs is your backend server architecture, and then your client architecture that will be used to connect APIs to websites, mobile applications, devices, and other systems. When I try to talk about API copyright, projects like API Commons, andAPIs.json, or educate folks about modern approaches to API design, deployment, management and integration in this client / server mindset, it can be very difficult to articulate some very important concepts. Not all developers, software architects, and business leaders are familiar with this very important, and the rapidly evolving surface area being defined by API.

To help convey some of my thoughts about this very important layer of the API space, I've created some new visuals that I can use in stories, slide decks, and my research reports. First I want to elevate the discussion beyond just client / server architecture, and highlight the importance of the API definition surface area that is present in the middle.

When you are designing, developing, managing, and integrating with APIs, you use some very specific languages for developing your server architecture. We use programming languages like PHP, Python, Ruby ,Node.js and other languages to develop the server side of our API. In most scenarios, companies choose the programming language they are already using across their business, developing all server side architecture in a common language.

Inversely, when it comes to client side architecture, companies should work to provide as many possible programming languages possible, providing code samples, libraries, and SDKs for all popular platforms. This is the promise of simple web APIs, is that a well designed API will abstract away any language specific details present on the server, making it accessible in any programming language. Then the good API providers encourage integration, by providing a wide variety of client tooling as they possibly can, to make integration as frictionless as possible.

In the middle, for the API definition layer, modern API developers are defining their interfaces, using machine readable formats like Swagger, API Blueprint, and RAML. Providing a machine readable definition of an API is proving to be critical for just about every aspect of the API life-cycle.

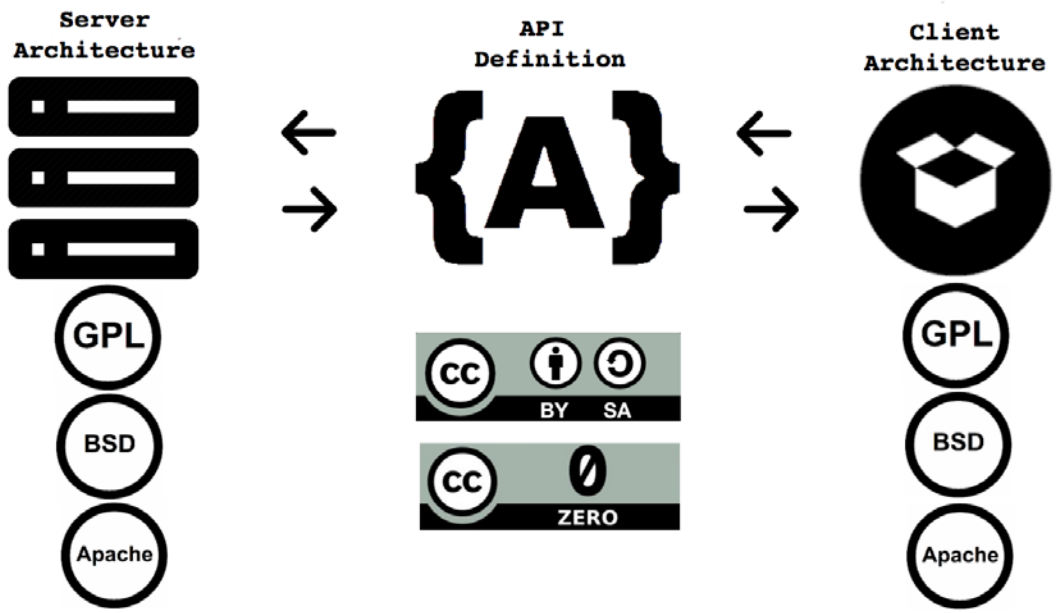


If you have a public API, or publicly available mobile or Internet of Things apps, you have a public API interface. This layer exists on the open Internet, and as we go into 2015, this layer for all public APIs should be well defined. Developers are using this definition to integrate with their existing systems, generating client tools, and are powering testing, monitoring, and other critical aspects of API integration.

Your server architecture is critical to API operations, and while I'm not in the business of telling pople how to license server side code, I will encourage anyone to make sure and at least apply a

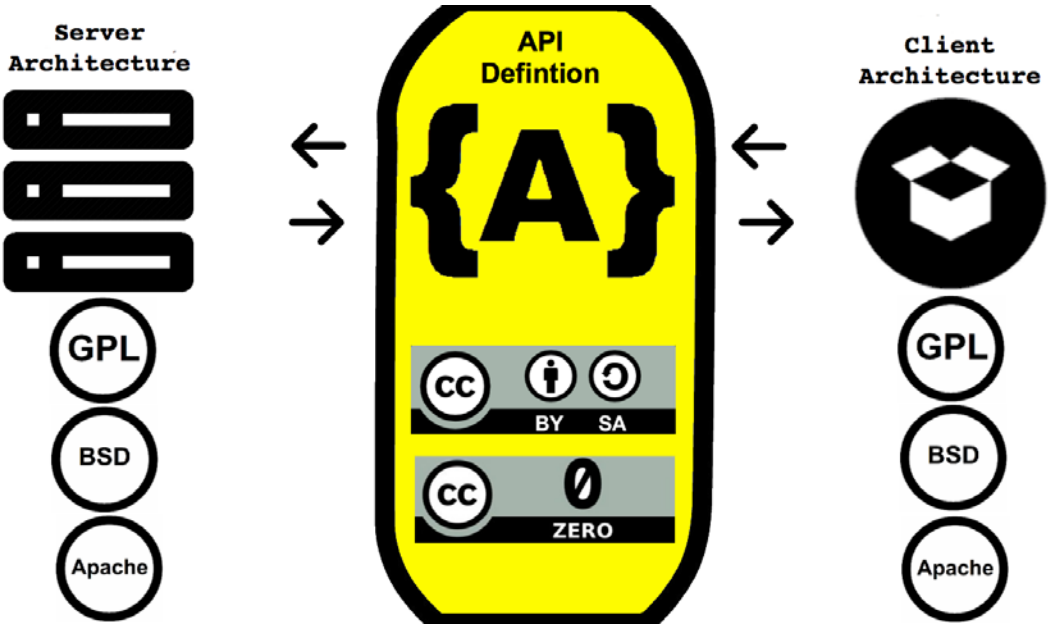
license, and I personally believe in using as liberal of license as you possibly can. When it comes to your client architecture, I will get a little more pushy about how things should be licensed--you are asking someone to integrate your API into their business, existing applications, and the code you provide should allow for as much flexibility as possible, not imposing restrictions.

This approach also applies to the API definition layer. Your API definition should be as accessible, open, and flexible as possible. You are expecting users to bake your Application Programming Interface (API) into their worlds, and this surface area is the point at which your two business meet. You can keep your server side code as a "secret sauce", but your API definition, and client architecture should be as open as possible.



I do not think that copyright should be able to be applied to API definitions, and my work with the EFF on the Oracle vs. Google Java API copyright case reflects this. With that said, we don't live in a perfect world, and embracing the one we have, I encourage applying a Creative Commons CC-BY-SA or CC0 license to your API definitions, regardless of whether or not you use Swagger, API Blueprint, or RAML. After you do, I encourage you to add it to the API Commons.

I've had conversations with people who claim that their API surface area is their secret sauce. First if it supports a public mobile app, is used on websites, or provides integration of external systems, and devices over the open Internet--it ain't a secret! Second your sauce should be in the delivery of services behind your API surface area, the actual wording, order, and other aspects of your definition should be open for anyone to use!



It is incredibly important that this layer of the API space is well defined, front and center in our conversations, and remains as openly accessible as possible. If you need a real world analogy to better understand the openness of this layer, read my API copyright -- restaurant menu story. Your menu is not your business, it will be the service behind the menu that truly defines your company.

The best example of this in action, within the API space, is around the Amazon S3 and EC2 APIs. If these API definitions were not openly available, and were protected under copyright, would cloud computing have ever happened? Would we have an ecosystem of tooling like Eucalyptus, to Chrome or Firefox extensions, that has helped us deploy global infrastructure, and build applications that have come to define cloud computing.

Please work with me to make sure the most important layers of the API space stay as open, and accessible as possible.

Source : http://apievangelist.com/2014/12/21/making-sure-the-most-important-layers-of-api-space-stay-open/