High Peak Solutions
Software Consultancy

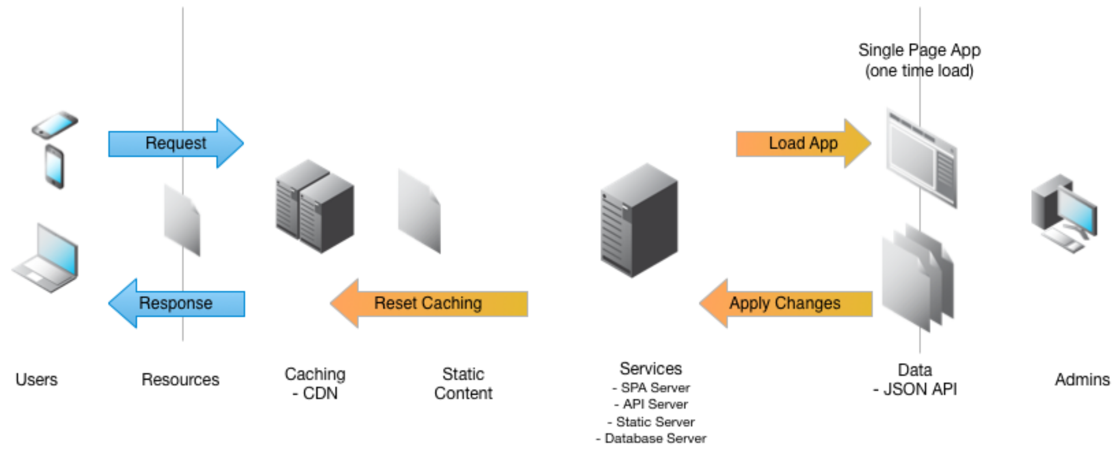
# "Almost Free" High Performance Stack - One Way Flow

There are countless advantages to having a high performance website including lower bounce rates, higher conversions and happier users. But like with every business decision there must be a cost-benefit analysis done when choosing projects; delivering high performance may or may not always be the highest priority. **Fortunately** a rising tide lifts all boats; and the rising tide for performance has been open source projects and services making the cost for making high speed mobile and responsive sites with proper architecture very minimal.

## A Low Cost High Performance Stack

To choose technologies based on performance and cost, I have narrowed the scope of this example to a generic Content Management System (CMS) for example a simple blog or e-commerce platform. On the **spectrum from interactive to informational**, there are two components - the admin side making changes to the content and the high visit user side only consuming the resulting content.

Architecture is simplified when we only have a single direction data (content) moves; from admin to user. The main advantages of this design is segmenting performance needs (admin and user) and allowing us to leverage free technologies and services by keeping the user loop as small as possible and avoiding load on our systems which further reduces costs.



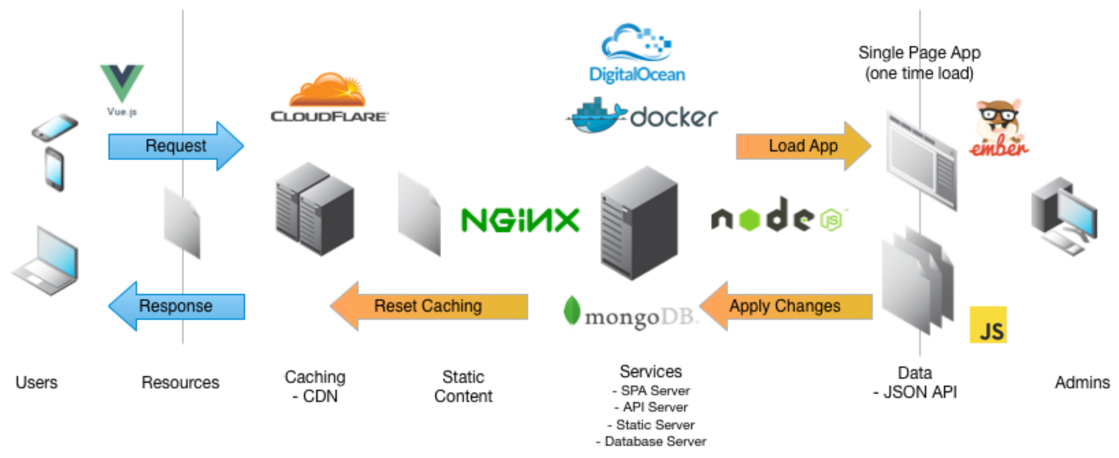
## Components

The stack is made up of two main components, the management Single Page Application (SPA) and the client facing static site. Different technologies can be used for both components because the difference in performance concerns and level of interactivity. The admin section can use a 'heavier' app framework because there is need for a smooth experience of managing, uploading, validating and changing content; and there is an expectation of a slower initial load or login process. Frameworks like these lend themselves to rapid prototyping and development and include lots of features such as built in security, resource connection management and validation. Managing content should be done through an API so that the system is modular; and so other applications or services can be used to perform the management if automation or integrations are needed from external sources (for example using data feeds to update content).

After content is validated and submitted by the admin application, a lightweight server can process and compile the updated assets into a static site format (or other high speed server side rendering format, see below for recommendations). The server itself actually hosts several services including the server for the SPA, a database for storing content, the API server for receiving changes and the server for the client app (you may also wish to have a separate service do the static site generation if you choose not to run that on the API thread).

Once the static site has been built (or rebuilt) the server process then clears the caching layer using APIs or calls to a caching service. Only the first (next) time a user requests the site will it actually reach and load your server, but this outgoing response will be cached and the new content will become readily available for all other users.

This caching strategy greatly reduces the load on infrastructure and moves the content closer to the users for a faster, more reliable experience. By doing this you can run multiple projects or sites on very lightweight infrastructure and at a much lower cost than when using a traditional self hosted server solution. Building sites in this way can increase the availability of your site or project by making it flexible, responsive and fast while keeping costs at a minimum.



Technologies I use for a low cost high speed site - os = open source, ft = free tier, f = free

- EmberJS (f, os) - An advanced client side framework for handling interactions, relationships, validation and easy prototyping. Integrates well with JSON API services.
- JSON API (f) - A growing web standard for object and response data formatting.
- NodeJS (f, os) - Javascript is quickly becoming a server side language running in NodeJS. I prefer it because it make client and sever-side data manipulation consistent.
- MongoDB (f, os) - Document database store lends itself to easily changing data format, objects and relationships, and has a language and format that is easily used with javascript.
- Digital Ocean ($5 month) - Hosting with easy scaling and management of cloud services.
- Docker (ft) - A container service that allows a server to run multiple independent services and easily manage across them.
- Docker Cloud (ft) - Easy continuous integration and hosting of docker containers (integrates nicely with Digital Ocean hosting).
- NGINX (f, os) - A highly optimized static content server (does not include a template engine, I also use NodeJS Express with Dust or Handlebar templates for ease of development).
- CloudFlare (ft) - A caching service and Content Delivery Network (CDN) with a free tier that allows easy setup, edge caching and analytics.
- Vue.js (f, os) - A lightweight client side interaction library (only does data binding and interaction. great for animation!).

Post By: Scott Traver scott@highpeaksolutions.com