Public Cloud Cost Visualizer Architecture

L. Breeman (s3212335)
B.G. Udrescu (s3136728)
M. Frikken (s3180417)
L. Visser (s2542501)
K. Tilman (s3229807)
E.J. Pasman (s2744457)
S. de Vries (s3186520)

TA: Charles Randolph
Client: Vasilios Andrikopoulos

Version: 1.1

March 16, 2018

Contents

1 Introduction **3**

2 Architectural overview **3**

3 Technology Stack **3**

4 Team Organization **4**

5 Change Log **5**

1 Introduction

Since we aim for a responsive environment the decision to go for bootstrap was pretty straightforward and easy to make. Concerning the 'interface' we want to go for more of a simplistic/minimal look, which preserves the property that our app is going to be easy to use. The ability drag and drop inputs, makes the app also very interactive with the user. Since we do not want to target only inexperienced people, we offer 'advanced options' sections for people that really know what they are looking for. Both experienced and inexperienced users are presented quite large data set(s). The best way to display these data sets in a user friendly manner to the users is by displaying the data in a graph with the *chart.js* javascript library. Moreover, using this technique we can make it easy to compare different data by just plotting them in the same 'graph'. The need for something like *NodeJS, Express, JSON & MongoDB* is self explanatory, because of the fact we need to simply serve webpages to our users.

2 Architectural overview

A web interface containing a drag & drop and canvas section. Be able to drag 'services' (e.g Virtual Machine, Storage, Database) onto the canvas. By clicking the service on the canvas, you can further specify (filter) options. Otherwise the default (average case) option is select for a service. Visualize the cost of the service(s) by showing the user a graph that shows the user a comparison of different options with their cost over time.

3 Technology Stack

Bootstrap **Frontend** an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

NodeJS **Backend** a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Express **Backend** a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. Provides a lot of HTTP utility methods and middleware which makes creating a robust API quick and easy.

MongoDB **Backend** MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. The document model maps to the objects in your application code, making data easy to work with. Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data. MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use

JSON **Backend** JavaScript Object Notation. JSON is a syntax for storing and exchanging data. JSON is text, written with JavaScript object notation.

Chart.js **Services** Simple, clean and engaging HTML5 based JavaScript charts. Chart.js is an easy way to include animated, interactive graphs on your website for free.

RESTful API **Services** provide interoperability between computer systems on the Internet. REST-compliant web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations. **Note: a RESTful API (db filled with the use of crawlers) with the price lists of the big providers is already available**

4 Team Organization

What teams are there and what are their responsibilities? Are the team responsibilities focused on different components?

Backend

Work on how to serve the webpages to the user. Has overlap with tasks of services team.

Frontend

Create a responsive web interface with bootstrap where you have the ability to drag & drop services on the canvas. Moreover, user should be able to further specifiy/filter the options they have selected. Display a graph after user has selected all their services.

Services

Work on how to pull data from the API, perform calculations on them and graph them.

5 Change Log

Descriptive list of changes made to the document tagged with date and author.

15-03-2018 Version: 1.0
Description: First draft
Author: S. de Vries

16-03-2018 Version: 1.1
Description: Changes based on TA's feedback
Author: S. de Vries