Demo and Scaling Road Maps

# Demo

The Vehicle Visualization has been hosted at <http://d2d.shakkir.com/>

1. Open the Chrome browser and open URL <http://d2d.shakkir.com/>and then you will see Vehicle Visualizer app with the map and some additional controls to modify center, coverage etc.
2. send vehicle update requests to the application as explained in the [API specifications](https://github.com/door2door-io/d2d-code-challenges/tree/master/fullstack)
   1. POST <http://d2d.shakkir.com/vehicles>
   2. POST <http://d2d.shakkir.com/vehicles>
3. Download the simulator app and start it by executing node simulate d2d.shakkir.com:80 now you should see moving cars in the Visualizer. In addition, each location update will be saved to the DB.

### Demo Server details

Webserver:Apache2

The Node.js application is hosted on an AWS Lightsail instance. And it is managed by PM2 process manager module in order to leverage its auto restart on failure and load balancing capabilities

Since the Node.js application is listening to port 3001, a virtual host configuration has added in the apache webserver configuration to route inbound traffic to port 3001.

MongoDB is also installed on the same instance.

# Scaling Road map

When application grows to 1000s of parallel usage, we will need more computing power and load balancers etc. and when multiple servers are involved, some state variable that is currently used in the application will be essentially replaced by to a collection in the database so that all it can be synced across severs.

Application has to be deployed on more powerful multi CPU instances and the performance and availability can be improved by using forks/clusters in the node.js application to leverage computing power of all the CPU cores. Need to start the application on PM2 process manager in cluster/fork mode to enable its load balancing capabilities.

Multiple production servers has to be in place for load balancing and downtime while deployment is avoided this way.

# Technology Choices

## Node.js

1. Easier to setup
2. No code compilation need
3. Easiest to deploy
4. High availability of libraries and it is easy to include
5. Tremendous Help available online for Node.js and JavaScript

## MongoDB

1. Good performance
2. Well documented
3. Small learning curve
4. Easy to deploy.
5. JSON compatible hence a good choice with Node.js

## Google Maps API

1. It is discussed everywhere online
2. Clean looking and it is from google!
3. More features than any other map APIs
4. Many libraries for customization available

## JQuery at Front-end

The libraries used for customizing map API has dependency on JQuery. Since the UI requirement was simple very limited static components, there was no need for complex libraries like react.js or Angular.