Running a MEAN stack on Google Cloud Platform with App Engine and MongoLab

Sandeep Dinesh
Nov 12, 2015 · 4 min read

In two of my previous posts, I talk about running a MEAN stack with Docker and then making it more robust by using Kubernetes.

In this post, I'm going to go on a tangent and take a different approach. Let's say you don't care about all that complicated stuff, you just want your MEAN application to run and scale without worrying about VMs and Pods and Docker and blah blah blah…

To do that, it makes sense to use a Platform as a Service (PaaS), which is a fancy way of saying you write the code and we run the code. There is no server or database management needed, which lets you focus on building a great product!

. . .

Tools

Google's PaaS is called App Engine. App Engine can run Node.js and handle all the scaling and maintenance so you don't have to.

For the database, I'm going to use MongoLab. It's the MongoDB you know and love, but you don't need to worry about hosting it.

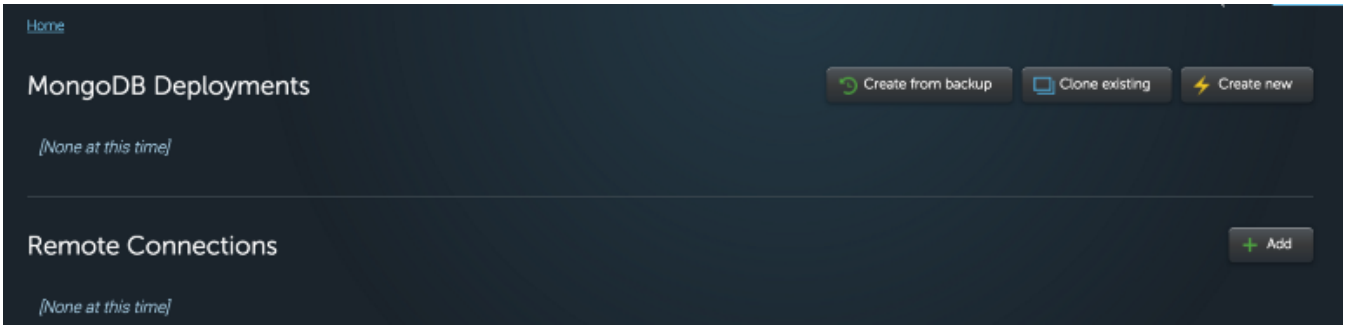
. . .

Step 1: Create the Database

Sign up for MongoLab.

Click that orange button!

Once your account is created, you should see a dashboard.

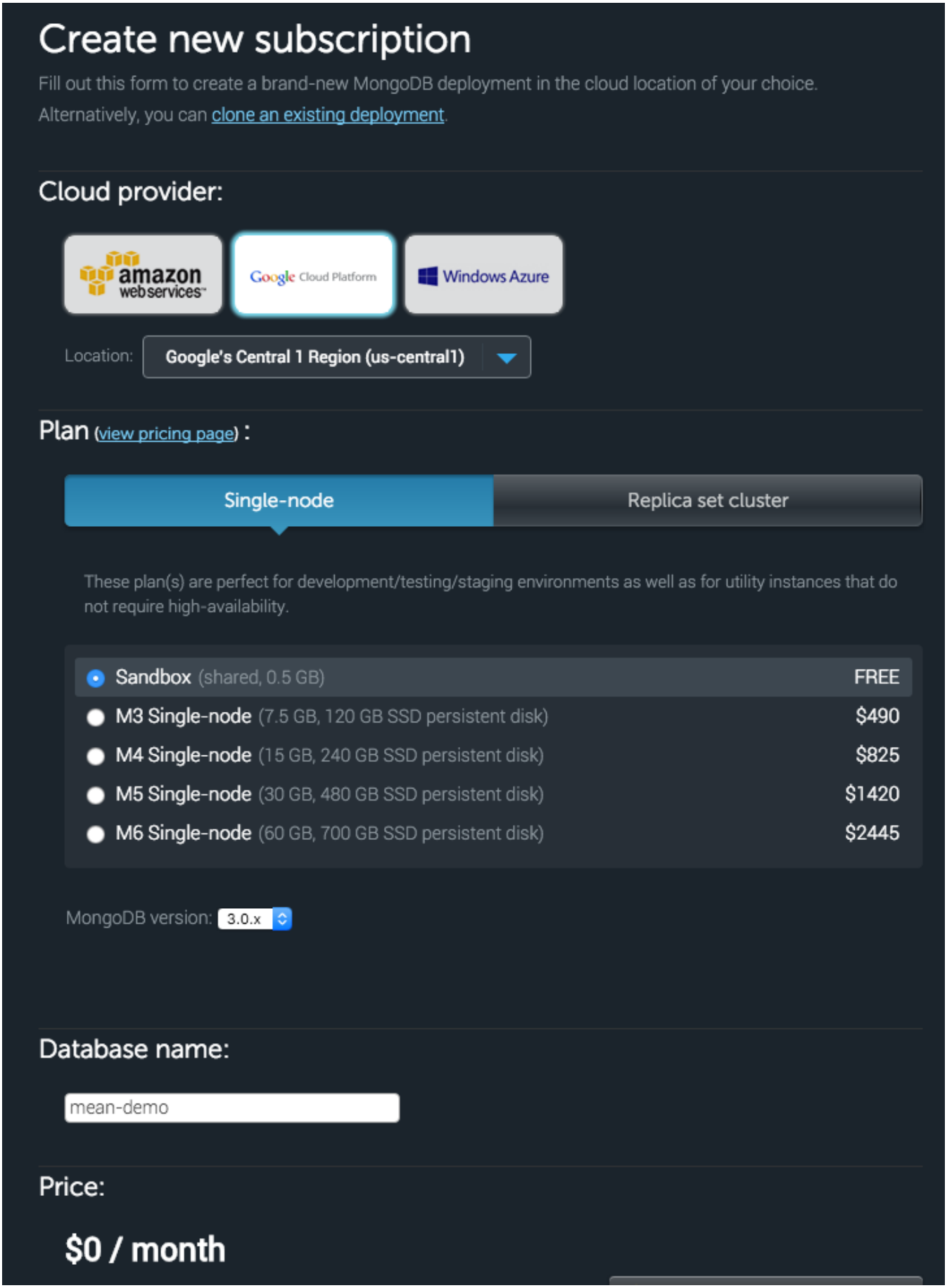


Click "Create New".

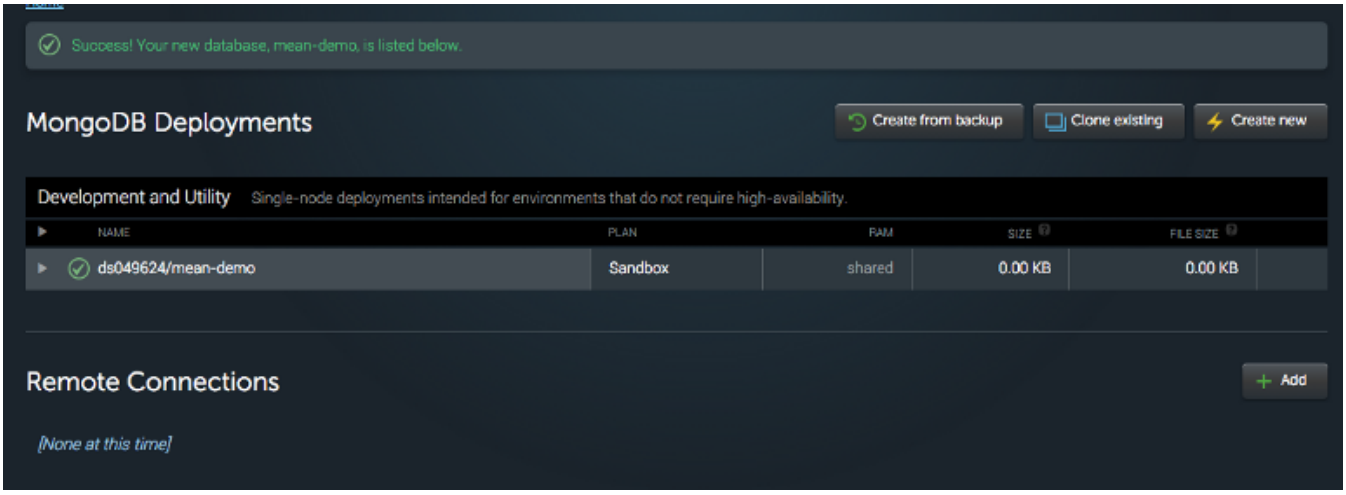
I'm using the free plan with Google Cloud Platform.

Also, make sure the **Database Name is *mean-demo***, because that's what our sample app expects.

Your config should look like this:

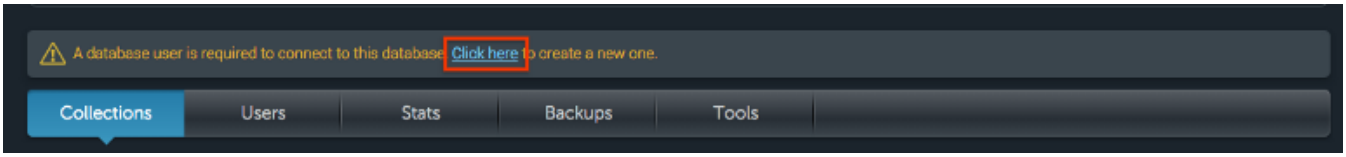


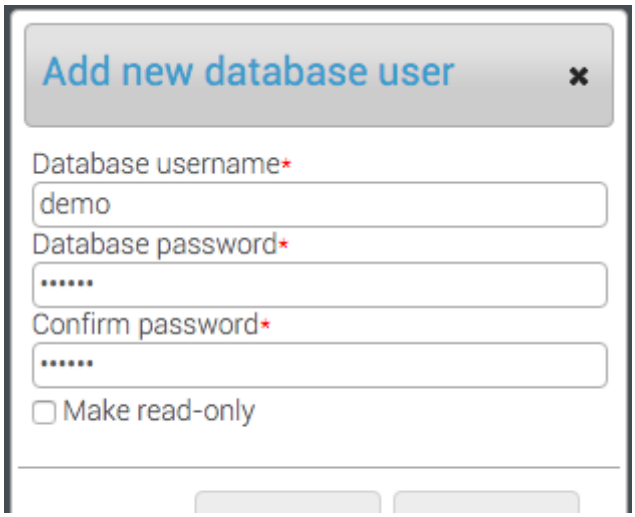
Once you click “Create new MongoDB deployment”, you should see your new Database up and running:

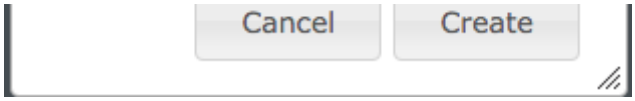


Click the database, and note the connection URI.

Now, create a user so the app can connect to the database:





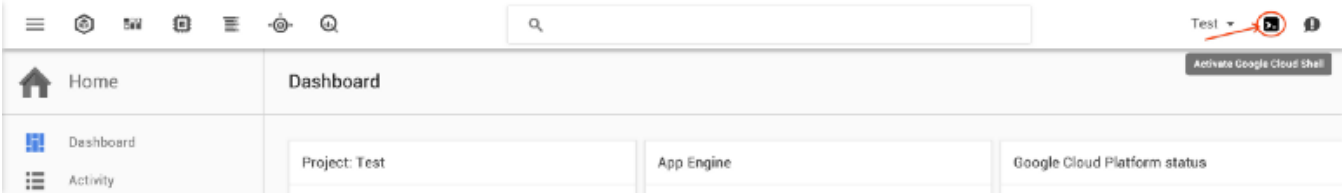


Done.

. . .

Step 2: Get the Code

Go to console.developers.google.com and select (or create) your project. For the rest of this tutorial, I'm going to be using Cloud Shell, which is basically a command line in your browser that has all the tools you need already installed. It is really cool!



Activate your Cloud Shell

Now, pull in the sample code:

```
$ git clone https://github.com/thesandlord/mean-demo.git app
$ cd app
```

And fix the configuration so you can connect to your MongoLab instance:

```
$ URI=<dbuser>:<dbpassword>@XXXXX.mongolab.com
$ DBPORT=####
$ sed -i -- "s/localhost:27017/$URI:$DBPORT/g" app.js
```

Replace *<dbuser>* and *<dbpassword>* with what you specified in Step 1.

Replace XXXXX with your Database ID (in my case, *ds049624*).

Replace #### with your Database port (in my case, *49624*).

All you are doing here is changing the MongoDB client to use your MongoLab instance.

. . .

Step 3: Test the Code

With Cloud Shell, you can easily test your code just like it was running on a local machine.

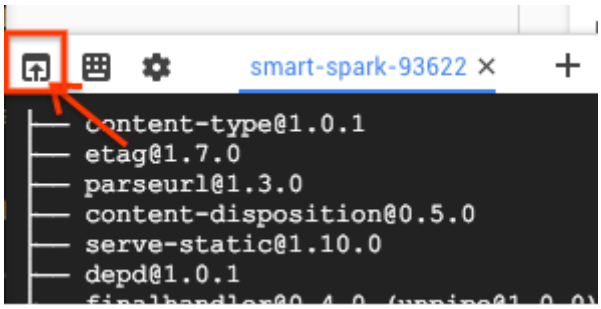
```
$ npm install

$ npm start
```

You should see this:

```
body-parser deprecated bodyParser: use individual json/urlencoded middlewares server.js:9:9
body-parser deprecated undefined extended: provide extended option node_modules/body-parser/index.js:105:29
I'm Listening...
```

Now, launch the Web Preview (Click the little button on your Cloud Shell):



And ta-da, it works!

If you add some data, it will add it into your MongoLab database. Try it!

. . .

Step 4: Deploy the Code

While Cloud Shell is great for testing, it can't host your app! It's time to deploy to production!

In the code, there is a file called **app.yaml,** which contains all the deployment details. Let's take a look:

```
runtime: nodejs
vm: true
```

Pretty self-explanatory. The ***vm: true*** option lets App Engine know it should use the Managed VM runtime (i.e. Docker).

To deploy, run:

```
$ gcloud preview app deploy app.yaml --promote
```

Or you can use the shorthand (thanks to the **package.json** file):

```
$ npm run deploy
```

And the deployment will begin.

gcloud will pull the project configuration from the Cloud Shell environment and your **app.yaml** file.

App Engine will copy your code, create a Docker container, spin up VMs and load balancers, and launch your code. You just sit back and relax.

Once you see this:

```
$ Deployed module [default] to [https://XXXX.appspot.com]
```

You are done. Go to the URL to see your MEAN app running!

That's it! Now you don't need to worry about servers crashing or database management. Of course, you lose some flexibility because you are not nitpicking every little detail, but for most applications App Engine works great!

For more info, check out the Node.js getting started guide and the MongoLab docs.

Thanks to Jack Wilber.

Google Cloud Platform Nodejs Mongodb