Nearme CMS Documentation

nearmeCMS was developed with Parse Server, the open source version of the Parse backend that can be deployed to any infrastructure that can run Node.js.

We will deploy nearmeCMS to Heroku and host the database in MongoLab, but before we need the following:

- Appdata
  - App ID
  - App Name
  - Master Key
  - REST API Key
- MongoLab
  - Database URI
- Mailgun
  - API Key
  - Domain
  - From Address
- AWS S3
  - Access Key ID
  - Secret Access Key
  - Bucket Name

1. Create a MongoDB database in MongoLab

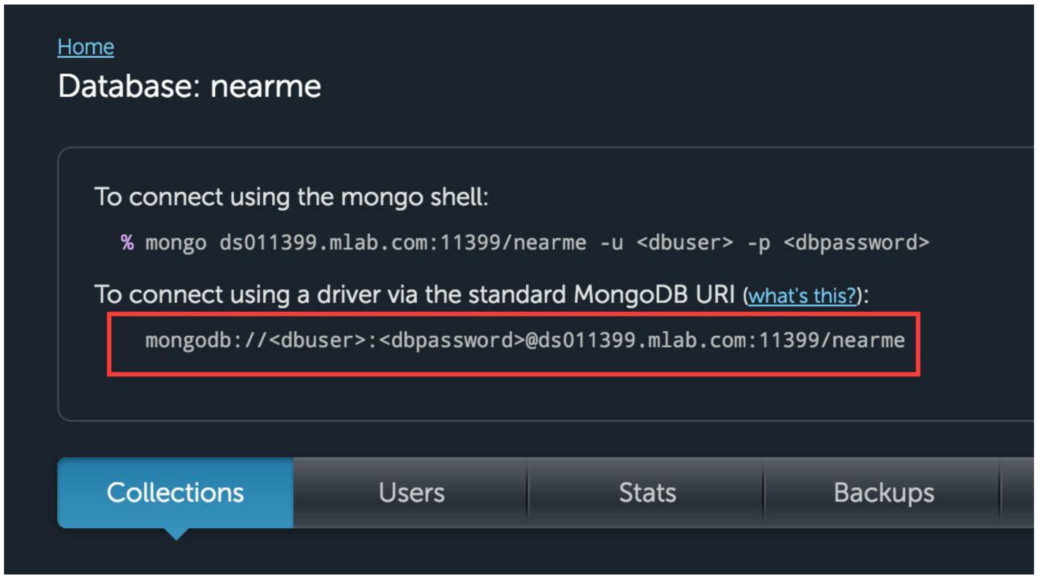
Create an account in MongoLab and log in.

On the dashboard, click the **Create New** button. On the page that appears, do the following:

1. Choose Amazon Web Services as cloud provider.
2. Choose Amazon's US East (Virginia) Region (us-east–1).
3. Select the Single Node tab and choose the Sandbox option. This is mLab's free tier.
4. Enter the database name.
5. Finally, click **Create new MongoDB deployment**.

The new database is created. You can of course choose different MongoDB options depending on the requirements of your app. Keep in mind that the original Parse service compressed your app's data. Parse recommends to 10x the size of your Parse data when creating a new MongoDB database.

If all went OK, you'll be redirected to your mLab dashboard which shows the newly created database. Click on the database to open its detail screen. Note the mongodb:// URI at the top, we'll need it later.



To connect using the mongo shell:

```
% mongo ds011399.mlab.com:11399/nearme -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard MongoDB URI (what's this?):

```
mongodb://<dbuser>:<dbpassword>@ds011399.mlab.com:11399/nearme
```

Next, do this:

1. Click the **Users** tab.
2. Click **Add database user**.
3. In the dialog that appears, enter an username and a strong password, then click **Create**. Make sure to leave the **Make read-only** checkbox unticked.

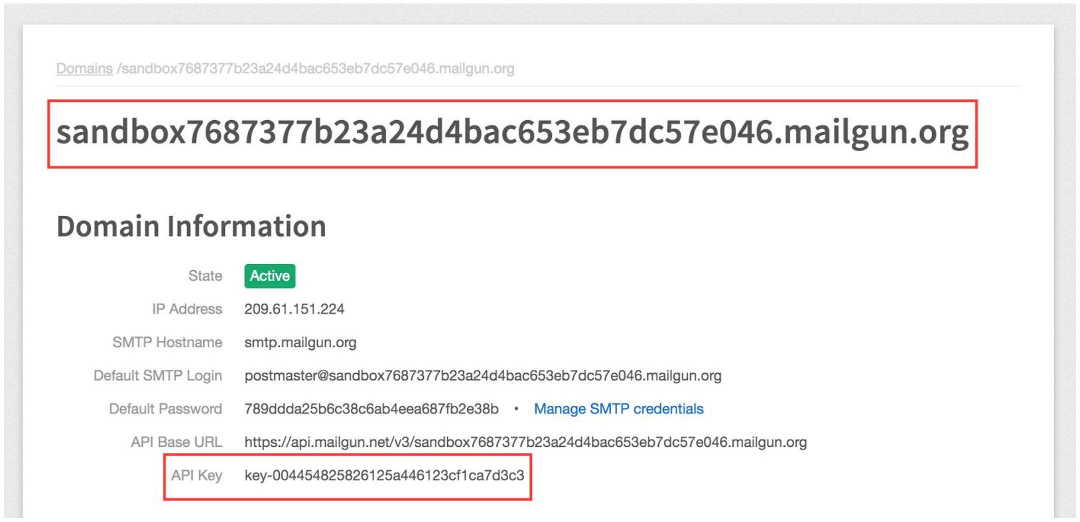
Make sure you note the username and its password somewhere safe. You will need it later.

2. Mailgun

Create an account in Mailgun and log in.

Next, do the following:

1. Click the default sandbox domain.
2. On the page that appears, take note the domain name and API key


Domains /sandbox7687377b23a24d4bac653eb7dc57e046.mailgun.org

sandbox7687377b23a24d4bac653eb7dc57e046.mailgun.org

Domain Information

State	Active
IP Address	209.61.151.224
SMTP Hostname	smtp.mailgun.org
Default SMTP Login	postmaster@sandbox7687377b23a24d4bac653eb7dc57e046.mailgun.org
Default Password	789ddda25b6c38c6ab4eea687fb2e38b • Manage SMTP credentials
API Base URL	https://api.mailgun.net/v3/sandbox7687377b23a24d4bac653eb7dc57e046.mailgun.org
API Key	key-004454825826125a446123cf1ca7d3c3

For testing purposes it's OK to use the sandbox domain, but later you should add your own domain.

3. AWS S3

To use Amazon S3, you need an AWS account. If you don't already have one, you'll be prompted to create one when you sign up for Amazon S3.

To sign up for Amazon S3

- Go to http://aws.amazon.com/s3/ and click **Sign Up**.
- Follow the on-screen instructions.

Now that you've signed up for Amazon S3, you're ready to create a bucket using the AWS Management Console. Every object in Amazon S3 is stored in a bucket. Before you can store data in Amazon S3, you must create a bucket.

To create a bucket:

1. Sign into the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3.
2. Click **Create Bucket**.
3. In the **Create a Bucket** dialog box, in the **Bucket Name** box, enter a bucket name.
4. The bucket name you choose must be unique across all existing bucket names in Amazon S3. One way to help ensure uniqueness is to prefix your bucket names with the name of your organization.
5. In the **Region** box, select a region. Select Oregon from the drop-down list.
6. Click **Create**.
7. When Amazon S3 successfully creates your bucket, the console displays your empty bucket in the **Buckets** panel.

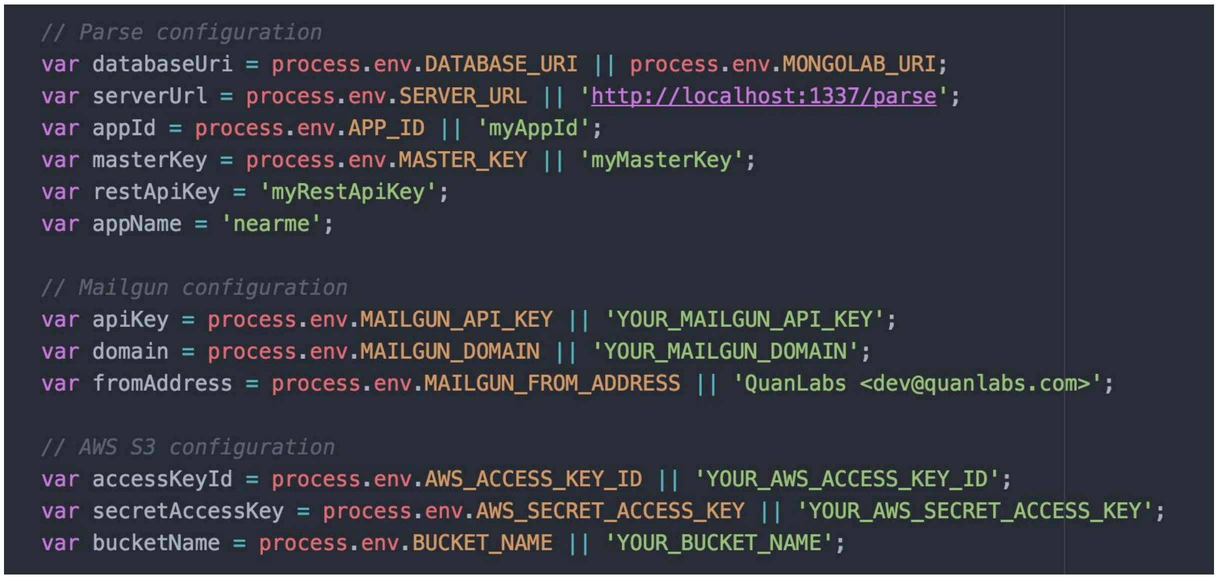
You now need to retrieve **Amazon Access Key ID** and **Secret Access Key**.

1. Go to Amazon Web Services console and click on the name of your account (it is located in the top right corner of the console). Then, in the expanded drop-down list, select **Security Credentials**.
2. Click the **Continue to Security Credentials** button.
3. Expand the **Access Keys (Access Key ID and Secret Access Key)** option. You will see the list of your active and deleted access keys.
4. To generate new access keys, click the **Create New Access Key** button.
5. Click **Show Access Key** to have it displayed on the screen. Note, that you can download it to your machine as a file and open it whenever needed. To download it, just click the **Download Key File** button.

Attention! If you do not write down the key or download the key file to your computer before you press **Close** or **Cancel** you will not be able to retrieve the secret key in future. Then you'll have to delete the keys which you created and start to create new keys.

4. Deploy nearmeCMS to Heroku

Once you have all the required information, open the **index.js** file and update it with your data.



```
// Parse configuration
var databaseUri = process.env.DATABASE_URI || process.env.MONGOLAB_URI;
var serverUrl = process.env.SERVER_URL || 'http://localhost:1337/parse';
var appId = process.env.APP_ID || 'myAppId';
var masterKey = process.env.MASTER_KEY || 'myMasterKey';
var restApiKey = 'myRestApiKey';
var appName = 'nearme';

// Mailgun configuration
var apiKey = process.env.MAILGUN_API_KEY || 'YOUR_MAILGUN_API_KEY';
var domain = process.env.MAILGUN_DOMAIN || 'YOUR_MAILGUN_DOMAIN';
var fromAddress = process.env.MAILGUN_FROM_ADDRESS || 'QuanLabs <dev@quanlabs.com>';

// AWS S3 configuration
var accessKeyId = process.env.AWS_ACCESS_KEY_ID || 'YOUR_AWS_ACCESS_KEY_ID';
var secretAccessKey = process.env.AWS_SECRET_ACCESS_KEY || 'YOUR_AWS_SECRET_ACCESS_KEY';
var bucketName = process.env.BUCKET_NAME || 'YOUR_BUCKET_NAME';
```

For instance, the databaseURI is set according to the environment variable we'll set later in Heroku.

At this point you need to generate three text strings of random characters.

1. First, go to https://www.random.org/strings/.
2. Then, set the strings to be 20 characters long and tick all the boxes for Numeric, Uppercase and Lowercase characters. Set the strings to be unique, then click **Get strings**.
3. Then, pick three of the 10 random strings you just generated.

If all went OK, you now have three random text strings. With the strings, do the following:

1. First, get back to that **index.js** file in your text editor and locate the right code block again.
2. Then, replace myAppId with one of the random text strings.
3. Then, replace myMasterKey with one of the random text strings.
4. Then, replace myRestApiKey with one of the random text strings.

Alright, at this point you need a Heroku account. Go to https://heroku.com and sign up for an account.

Next up, make sure the Heroku Toolbelt knows who you are. Type the following in Terminal:

```
heroku login
```

When Heroku asks for it, input your username and password.

Next, we're going to deploy nearmeCMS to a Heroku instance.

Make sure you're still in the ~/yourapps/nearme-cms with Terminal and run:

```
git init
```

Then, type this on the command line:

```
heroku create
```

The output at the command line will be similar to this:

```
Creating app... done, stack is cedar-14
https://vast-shelf-54389.herokuapp.com/ | https://git.heroku.com/vast-shelf-54389.git
```

Heroku will now create an app for you, assign it a random name and add a Git "remote" to the repository. This is one of the powers of Heroku: you can deploy and update your app's code with Git.

Next up, you need to add the recent code changes to your local repository. You changed the file, indeed, but these changes also need to be tracked in the Git repository. Input the following two commands in Terminal.

```
git add index.js
git commit -m "Updated config vars"
```

The index.js file is now added to the local repository, but the source code change is not uploaded to the Heroku instance we created. We'll do this now.

Type this in Terminal:

```
git push heroku master
```

If all goes OK, you'll see a bunch of lines racing across your screen. If you dig a little deeper, you'll see Heroku recognizes the NodeJS app, builds a whole bunch of dependencies (library code Parse Server uses), and then attempts to restart the app process (which doesn't exist).

Next up, we need to set that environment variable that points to your MongoDB database instance. You'll do that with the MongoDB URI. You can find it at the top of your mLab dashboard.

Copy your mLab URI to a text file and then replace <dbuser> and <dbpassword> with your own username and password.

Note: Don't use your account username and password, but use the username and password you created for the database.

Next, type in the following in Terminal on the command line (still in the same directory). Replace the mongodb://... part with your own MongoDB URI.

```
heroku config:set MONGOLAB_URI=mongodb://...
```

Also, you need to set the server URL. In this case, Heroku generated the following URL:

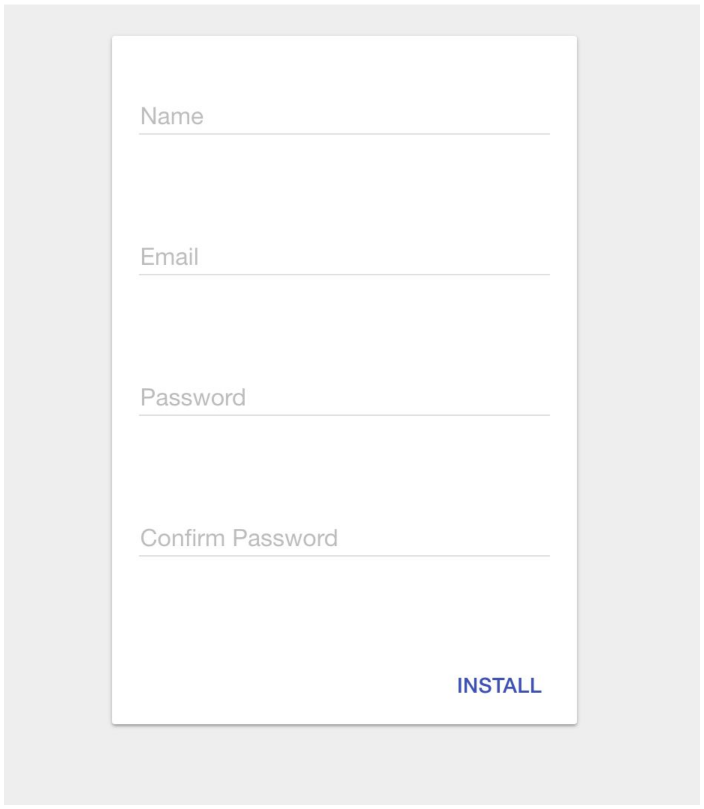
https://vast-shelf-54389.herokuapp.com

```
heroku config:set SERVER_URL=https://vast-shelf-54389.herokuapp.com/parse
```

Don't forget to add /parse at the end.

Awesome! Your Heroku instance is now complete.

Now go to https://vast-shelf-54389.herokuapp.com/install to complete the installation. You need to create the admin account to manage places, categories, reviews and users.



Fill the form and press **Install.** If everything goes well, you should now see the dashboard.

That’s it!