Galvanize

Content Hub

Installation Instructions

April 6th, 2020
Version 1.0

Team Galvatron

David Guo	57734162
Pengwei Zhou	73569758
Peter Han	14912166
Tony Kong	47078150
Tyler Nee	22705157
Vickie Yen	13358156
Yuting Wen	44625168

WIP Setup Details

Current WIP Links

Our WIP is deployed at the URL: http://galvatron.dream.sh:3000/login
There are three test accounts, each with a different role:

User: galvatronuser123@gmail.com
Approver: galvatronapprover123@gmail.com
Admin: galvatronmaster123@gmail.com

Password for all accounts: Galvatron123

Setup Instructions

Content Hub can be hosted on AWS using any EC2 instance. It is however required to use Ubuntu 18.04 LTS as the Linux distribution.

1. Deploy EC2 instance on AWS, choose Ubuntu 18.04 LTS as the operating system.
2. Register for Google OAuth 2.0 client credentials. This is required to allow login via Google OAuth 2.0, which is the only way to log into Content Hub.
3. Register for a Mailgun account in order to obtain a valid Mailgun API key. This is required by the scheduler to send out digest emails to users.
4. Setup Mailgun account by registering a mail server domain name for sending emails. Details can be found at https://app.mailgun.com/app/sending/domains.
5. Clone project repository at https://gitlab.com/cpsc319-2019w2/galvanize/galvatron/content-aggregator.git.
6. Refer to README.md to first setup the configuration file. In particular pay attention to the following variables:
    - **GOOGLE_CLIENT_ID** and **GOOGLE_CLIENT_SECRET** are obtained from your Google OAuth 2.0 credentials from **step 2.**
    - **MAIL_APIKEY** is obtained from your Mailgun account described in **step 3.**
    - **MAIL_DOMAIN** is the domain name set in **step 4.**
7. Refer to README.md for installation instructions, specifically on how to use the provided deployment script, which will install all necessary libraries, dependencies, and systemd service files.
8. After installation, use the deployment script again to deploy and start all services.
9. To re-deploy, once again refer to the README.md for information on how to use the deployment script.

Note: A PDF version of the README.md has been attached to the end of this document.

References

Google OAuth:
https://developers.google.com/identity/protocols/oauth2
Mailgun Domain Registration:
https://help.mailgun.com/hc/en-us/articles/202256730-How-do-I-pick-a-domain-name-for-my-Mailgun-account-
Mailgun Batch Sending Documentation (choose Node.js on the NavBar):
https://documentation.mailgun.com/en/latest/user_manual.html#batch-sending

Local stories by local developers.

Please view the README directly here. Some dropdowns are not available in this PDF.

Production Environment

Requirements

Must be running **Ubuntu 18.04 LTS**. The deployment script will fail to run in other Linux distributions.

Dependencies

Node.js >= 12.x
MySQL >= 8.x
Serve >= 11.x

All three dependencies will be installed by the deployment script.

Configuration

The deployment script will look for a configuration file named `prod.conf` in the same directory:

▶ Sample production config file (click to expand)

Ensure the configuration file contains all necessary environment variables and is saved as `prod.conf`.
For a description of each variable, see Environment Variables.

Installation

To install Content Hub for the first time, run the deployment script using the `install` argument with sudo permissions:

```
$ sudo ./deploy.sh install
```

All required dependencies, along with any required systemd service files, will be installed.

Deployment

The deployment script handles all deployment activities.
To update the project repository and re-deploy, choose one of the following options:

```
$ ./deploy.sh update         # Update both frontend and backend
$ ./deploy.sh updatefrontend # Only update frontend
$ ./deploy.sh updatebackend  # Only update backend
```

To reload the database:

```
$ ./deploy.sh database
```

To start and stop Content Hub:

```
$ ./deploy.sh start
$ ./deploy.sh stop
```

To check backend logs:

```
$ ./deploy.sh log
```

Environment Variables

Server Side

```
DB_USERNAME                 MySQL server username
DB_PASSWORD                 MySQL server password
DB_HOST                     MySQL server hostname
DB_NAME                     Content Hub database name in MySQL

MAIL_DOMAIN                 Mail server domain name registered in Mailgun account
MAIL_APIKEY                 Mailgun API key retrieved from Mailgun account

JWT_SECRET                  Secret key used to sign the JSON Web Token

GOOGLE_CLIENT_ID            Client ID retrieved from Google OAuth 2.0 Admin Panel. Requir
GOOGLE_CLIENT_SECRET        Client secret retrieved from Google OAuth 2.0 Admin Panel. Re
```

Client Side

```
REACT_APP_SERVER_HOST       Hostname of the server. Do not include trailing '/'
REACT_APP_SERVER_PORT       Port of the server

REACT_APP_GOOGLE_CLIENT_ID Client ID from Google OAuth 2.0 Admin panel, should match GOC
```

References

Google OAuth 2.0 client credentials are required to enable login via Google OAuth 2.0. See more detailshere.
Content Hub uses Mailgun for the batch sending of emails. To use the Mailgun API, an API key from a verified mailgun account and a valid domain name are required.

Development Environment

Installation

Setup Node.js

Download and install Node.js >= 12.x

Setup local database

Download and install MySQL >=8.x
Use **MySQL** to run `db_create.sql` and `db_load.sql` scripts.

Setup environment variables

To set environment variables, refer to the following config file examples:

▶ Sample Linux config file (click to expand)

Save the file as `dev.conf` and source from it using:

```
$ source dev.conf
```

▶ Sample Windows config file (click to expand)

Save the file as `dev.bat` and execute from it using:

```
$ dev
```

Usage

Start client

```
$ cd client
$ npm install
$ npm start # Starts at localhost:3000
```

Start server

```
$ cd server
$ npm install
$ npm start # Starts at localhost:9000
```

Testing

Framework

Jasmine v3.5.0

Installation and usage **[optional]**:

```
 $ cd server
$ sudo npm install -g jasmine
$ jasmine
```

To run backend tests without installing Jasmine, use `npx`:

```
 $ cd server
$ npx jasmine
```