Project Introduction

This web app is a customizable news aggregator. It has login function, user can add their favorite websites that they want to follow. Our website will display several most recent and popular articles from them.

Front end uses Express.js with npm and handlebars. And backend it uses MongoDB, Redis Clustered Storage, cookies, and so on.

We have teams specializing in scraping web content, classifying and training data, designing and maintaining databases, network and server services, and full-stack developers in charge of making the web app.

Set up dev environment

It is advised to develop on AWS server as redis cluster is hard to set up on local machines. And all the domain data and session details stay up there.
All files are **cloned by github,** you need to learn basic **git operations** to track versions and branch.

- you will use git checkout, git push, git pull, git commit, git log

❗ If multiple people need to work in the same time, you may need to open a new user or new server.

The project is deployed on ec2 server with address ec2-user@ec2-3-83-81-66.compute-1.amazonaws.com (the first ec2-user is the user name). To view the source directory, first **ssh** into the server. Before typing in the below ssh command, you need to have the private key file "aws_cluster.pem" in the directory `~/.ssh/` . (**ASK 博飞 for the pem private key)** Aws needs to verify the identity by using this file. This file is determined when set up the server. By using

```
ssh -i "aws_cluster.pem" ec2-user@ec2-3-83-81-66.compute-1.amazonaws.com
```

you will be able to see the files on server.

You can type in `npm start` to start the express service in the project directory. To view the website (as it's outputted in localhost 3000 on server), you need to forward the port on server to your local machine. To do that, open a new terminal window and type in the below command and keep the terminal open as long as you need to view the website.

```
ssh -i ~/.ssh/aws_cluster.pem ec2-user@ec2-3-83-81-66.compute-1.amazonaws.com -L
3000:localhost:3000.
```

The above syntax is *called port forwarding. **-i** specifies the location of the private key file, and **-L** specifies to forward the port. The first 3000 is your local port number to be replcaed, and the latter 3000 is port # on server that is to be forwarded.*

Now, to view the website, type in `localhost:3000` in your brwoswer.

Redis cluster is deployed on ubuntu@ec2-34-235-162-191.compute-1.amazonaws.com. There are multiple ec2 server responsible for handling different cluster of data. Using visulization tools like rdm (redis desktop manager), you can usually only the data from one server. But commands in js (as `ioredis` package we used in Javascript) supports cluster (multiple servers), meaning you can get all data, so no worries.

(Mac instruction below) => To use rdm as introduced above, on your local terminal window, type

```
ssh -i ~/.ssh/aws_cluster.pem ubuntu@ec2-34-235-162-191.compute-1.amazonaws.com
-L 6379:localhost:6379
```

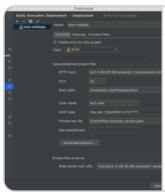
and keep the terminal open. This commands forward 6379 port on the server (that are used for redis) to your localhost 6379 so that you can see the content stored in redis using rdm.

In your rdm configuration, you can use 127.0.0.1 as server and 6379 as port number. (this is default) You will finally see the content on server as your local port 6379 is now actually 6379 on the server.

We also need to use MongoDB on the server, it's already set up. As long as the server is running, you don't need to restart mongo service. If the ec2 server hosting our web app is restarted, type in `sudo mongod` to start the mongo server.

It's convenient to use IntelliJ IDEA IDE editor for editing the files hosting on the server.

- To see the files on our ec2, open preference, deployment, set up the following:



Screen Shot 2019-07-27 at 12.18.23 AM

  - use SFTP type of connection.
  - SFTP host field is: `ec2-3-83-81-66.compute-1.amazonaws.com`
  - port # is 22
  - root path is /home/ec2-user/WebAppDevs
  - user name is `ec2-user`
  - auth type is Key pair and choose the `aws_cluster.pem` file
- Select Remote Host side bar, open the filees.

Screen Shot 2019-07-27 at 12.22.29 AM

  - the file name on the tab is <aws-webapp>/….(your file)
  - When saving the file using `command+s` , it's saved locally. You **MUST** also click `upload current remote file` to reflect it on server. (ctrl+shift+Q)
- Also, under Tools, click `Start SSH session. (so that you are ssh'ed into the server).

Recap: (short version)

- Check redis serve, (**ASK 博飞)**, and check mongo DB server (deployed in same server as this webapp)
- port forward localhost 3000 so that you can see the website in your browser using localhost:3000:

  `ssh -i ~/.ssh/aws_cluster.pem ec2-user@ec2-3-83-81-66.compute-1.amazonaws.com -L 3000:localhost:3000.`
- in IDEA IDE, after configuration, under tools, start a ssh session, and open the remote host and files so you can edit the files on the sever. (**DONT FORGET TO UPLOAD TO SERVER in addition to saving it locally)**
- if you want to see redis, use:

  `ssh -i ~/.ssh/aws_cluster.pem ubuntu@ec2-34-235-162-191.compute-1.amazonaws.com`

`-L 6379:localhost:6379`

- ❗ **before using `npm start` to start nodejs, check it's version using `npm -v` It must be 6.x.x.**
  - If not correct, use `nvm install 12` to use the 6.10.0 version npm.
- ❗ SSH pipes often break, if something doesnt work, check the ssh session. You may need to restart the ssh session.

Project Structure

The project is using express and its routings.

- server.js
- app.js
- public
  - fetch.js
  - css...
- routes:
  - api.js
  - index.js
  - ReadDB.js (deprecated, not currrenly in use)
- views:
  - index.handlebars (abbreviate hbs)

Comments in the code should help you understand the code.

Generally, `fetch.js` handles all interactive elements on webpages, including its fetch calls to internal service APIs. `api.js` handles JSON returning api calls (by using router.get) and cookies related operations. `index.js` handles page rendering of index.handlebars by getting '\' http request and responding by res.render.

For a more complete course on those, refer to the Stanford class web.stanford.edu/class/cs193x/lectures/.

Some reminders:

- cookies operations must not be in file server.js. Otherwise it will not work. **reason NOT**

CLEAR. TO BE investigated

Functionalities Planed

- fix all bugs regarding login, cookies and sessions, login condition handling (different login and cookies states correspond to different db access to get what cards (domains) to display.
- fix get details from redis as redis structure is changed recently
- add display of corresponding details of a suburl (
  - title, timestamp (how many minutes ago…), estimated reading time (word count), thumbnail pic (if no pic, don't display), author name, etc……..
  - design where to display
- Related articles for each suburl titles
- beautify the design and css.
- link to other pages in our webapp, currently there's only index.html (to be continued…

Known Issues

Domains and suburl:

- data access from redis cluster need to be revised (as data structure changed)
  - currently, no subruls or titles are displayed…
- including getting the initial 6 cards from the available domains on server *(Bofei recommends ZSCAN command)*

Cookie related:

- cookies delete implementation
- check conditions for whether to use mongodb, cookies, or both (determine login? local cookies present?)
- recover session if `sid` present in cookie, but browser quit. When a new one opens, there will be new session… recover the old one with sid.

Login related:

- when logout, need to refresh

- ASK 唐吴斌，王宇