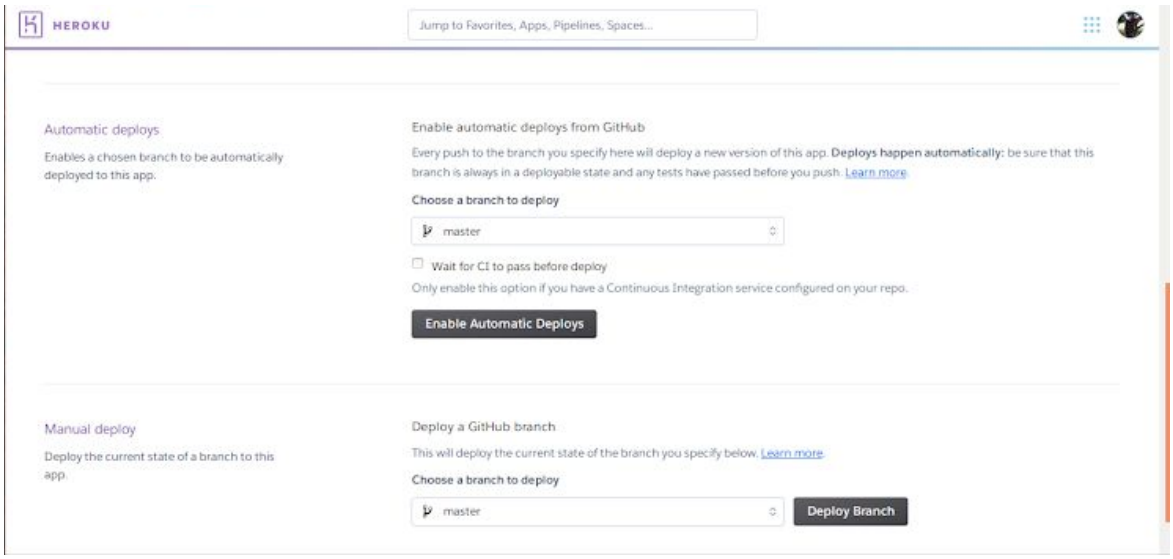
Continuous Integration and Deployment

Expected Time to Read and Implement: < 25 Minutes

Continuous Deployment

One of the easiest ways to set up continuous deployment is to enable automatic deploys on Heroku. (make sure to unselect the “Wait for CI to pass before deploying” option since we haven’t set up any CI yet). If you’ve connected Github and Heroku, after enabling automatic deploys, you can try making a change to the website and pushing the code to Github. You’ll see that that the website is rebuilt and deployed every single time.



After this is done, make sure to push code to the master branch only when you’re sure it’s working completely.

In the next part of the guide, we will setup TravisCI so that any change pushed to the master branch is run against **unit tests** before automatic deployment. Unit tests are used to automatically verify that code is working as it’s supposed to. While you’ll be learning and working with unit tests in detail post-Midsem exams, the guide in the next page will help you setup TravisCI on your app. Using TravisCI you can automatically run tests as soon you push a change to Github. *(Note: TravisCI works only on public Github repos)*

Setting Up Travis CI

Travis CI has been set up on [this git repository](#) (same repo as last two tutorials). It might be helpful to keep looking at the code there while reading these instructions.

1. Create a file .travis.yml in the root of your MERN project that looks something like this. The node version in your package.json and this should match.

```
.travis.yml
language: node_js
node_js:
  - 12.14.1
script:
  - "npm run test"
```

2. Install Mocha, Chai and Supertest using these commands:
   a. npm install mocha chai supertest --save-dev
3. The file tests/basic_tests.js in the git repository gives you an example of how unit tests look like. You can create something that looks like this in your repository as well (or just copy-paste it). The second test case is asserting that 1+1 = 2 and the first one is confirming that an element is not present in a list. These tests are not actually useful in checking code - we're just using these to show you how Continuous Integration (CI) works. For Sprint 2 of your semester project, you'll be writing unit test cases to actually test your code before deploying.

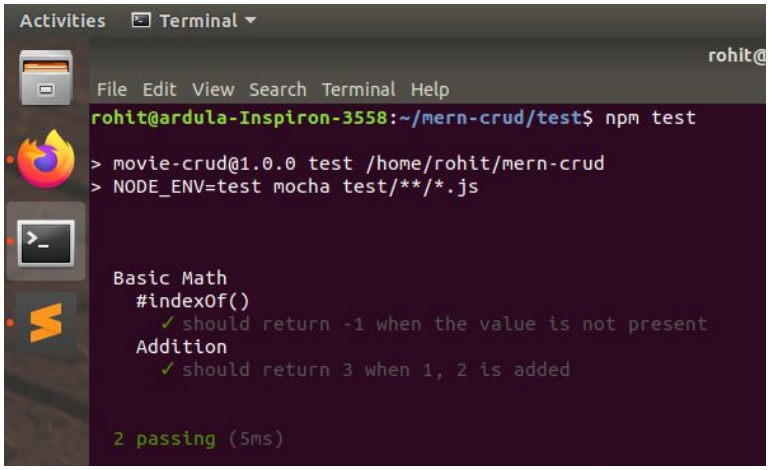
```
tests.js    index.js
var assert = require('assert');

describe('Basic Math', function() {
  describe('#indexOf()', function() {
    it('should return -1 when the value is not present', function() {
      assert.equal([1, 2, 3].indexOf(4), -1);
    });
  });
  describe('Addition', function() {
    it('should return 3 when 1, 2 is added', function() {
      assert.equal(1 + 2, 3);
    });
  });
});
```

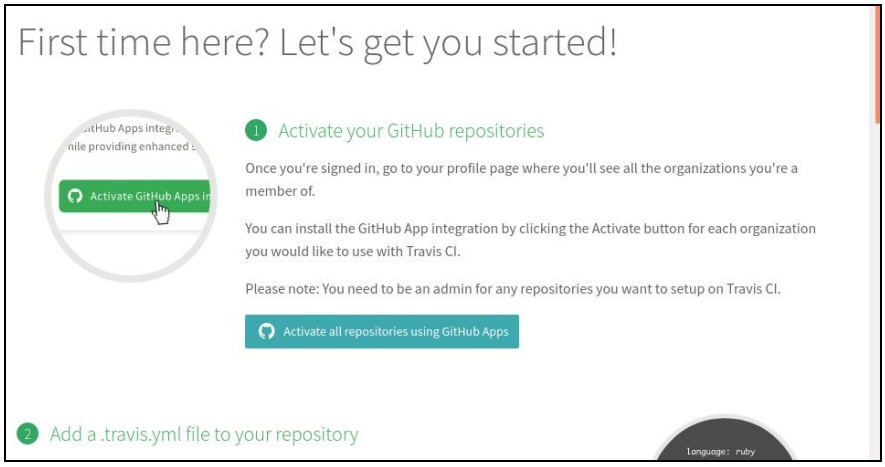
4. Edit the package.json to include a test script. (highlighted)

```
  "description": "MERN CRUD app for movies",
  "main": "index.js",
  "scripts": {
    "heroku-postbuild": "cd client && npm install && npm run build",
    "start": "npm install && node server.js",
    "test": "NODE_ENV=test mocha test/**/*.js"
  },
```

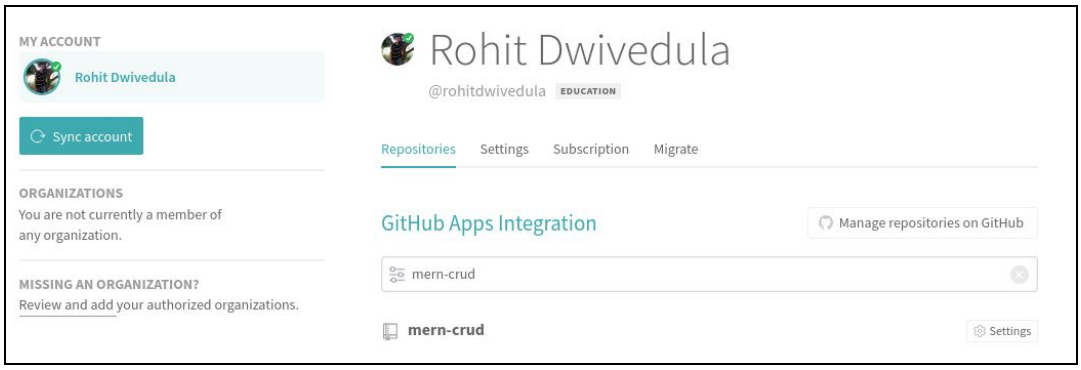
5. Try running `npm test` in the console. If everything's been configured right, we'll see that both the test cases have passed.



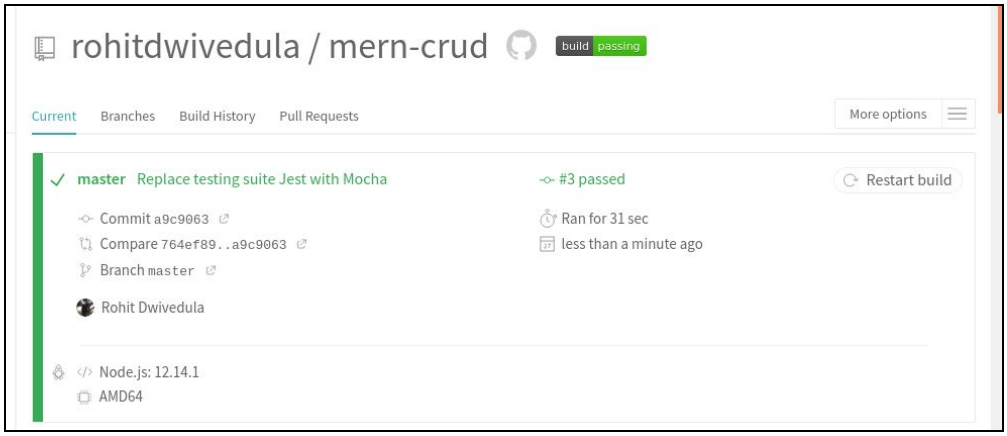
6. Commit all of this to your git repo and push it to Github.
7. With this done, we'll now go to travis-ci.org and create an account with Github.
8. Activate your Github repositories on Travis CI by giving the relevant permissions.



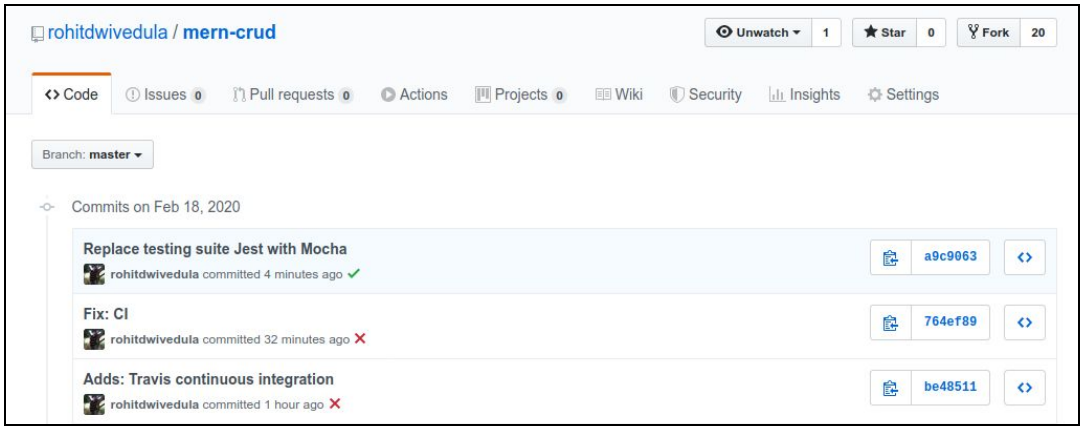
9. Wait for a while as Travis CI connects to Github. Search for the relevant app and click on it.



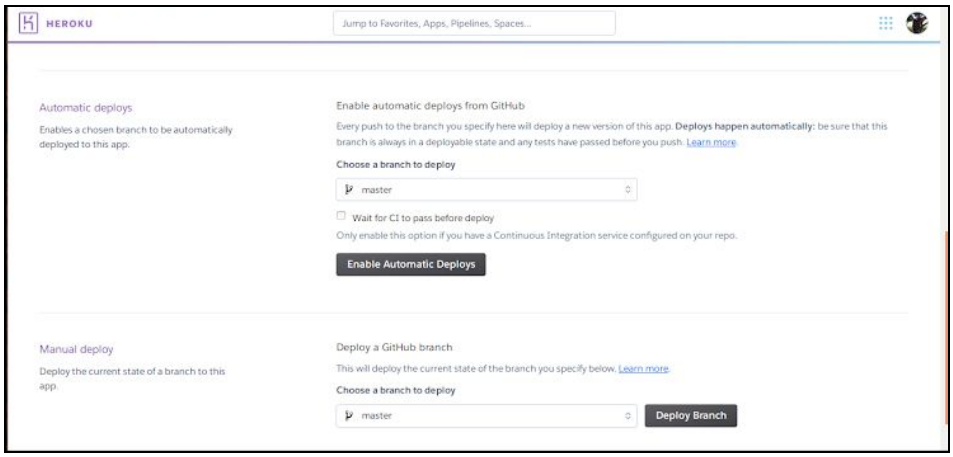
10. TravisCI will run the unit tests and if you've configured everything properly, you'll see a green screen of success.



11. See the commit history on Github. You'll see a green tick mark/red cross beside each commit telling you whether unit tests passed or not for that.



12. Go back to the Heroku, and open your app settings and go to the deploy section. Click on the "Enable Automatic Deploys" button. Make sure to select the tick box "Wait for CI to pass before deploy". Doing this will ensure that your app will be redeployed only if the unit tests pass.



Links

1. CI/CD with GitHub, Travis CI and Heroku: This guide explains how to set up continuous integration using TravisCI.
2. Build a Unit-Testing Suite with Mocha and Mongoose
3. Endpoint testing with Jest and Supertest | Zell Liew: A brief intro to unit testing using Jest.