19 Continuous Integration and Continuous Deployment (CI/CD)

How does code get to the cloud?

19 Continuous Integration and Continuous Deployment (CI/CD)

How does code get to the cloud?

Tonight we'll see how to deploy our applications to servers where anyone can access them through the Internet!

19 Continuous Integration and Continuous Deployment (CI/CD)

How does code get to the cloud?

Tonight we'll see how to deploy our applications to servers where anyone can access them through the Internet!

How does the internet work?

19 Continuous Integration and Continuous Deployment (CI/CD)

How does code get to the cloud?

Tonight we'll see how to deploy our applications to servers where anyone can access them through the Internet!

How does the internet work?

Tell me what to draw. Let's diagram it.

19 Continuous Integration and Continuous Deployment (CI/CD)

Objectives

- Students will learn about continuous integration and continuous delivery
- Students will be able to configure TravisCI to run their tests on all pushes to GitHub
- Students will be able to deploy their application on Heroku with a continuous delivery pipeline

Continuous Integration

Continuous Integration (CI) is the process of regularly merging individually developed features of code into a shared repository as frequently as they are made. In the most basic setup, a team could simply use git to merge code into a master branch. However more advanced CI patterns can be developed. These patterns can automate static analysis (lining), running unit and integration tests, testing platform support, enforcing code reviews, and much more.

Continuous Deployment

Continuous Delivery (CD) is the process of deploying software in short cycles, by ensuring that software can be deployed at any time. CD pairs very will with advanced CI setups that help ensure the core product is always a stable code base.

Travis CI

Travis CI is a hosted and distributed continuous integration service (CI) that is most often used to build and test software projects hosted on GitHub.

Bare Bones .travis.yml

Travis CI is configured by adding a file named `.travis.yml` to the root directory of the repository. This file specifies the programming language used, the desired building and testing environment and various other parameters.

Example **.travis.yml**

```
language: node_js
node_js:
  - 'stable'
before_script: npm i
script:
  - npm test
```

Complicated .travis.yml

Some dependencies might require more and more configurations. Here's an example of a more robust `.travis.yml`. Try starting with a simple `.travis.yml` file and seeing what dependencies require additional configuration.

```
language: node_js
node_js:
  - 'stable'
services:
  - mongodb
addons:
  apt:
    sources:
      - ubuntu-toolchain-r-test
    packages:
      - gcc-4.8
      - g++-4.8
env:
  - CXX=g++-4.8
sudo: required
before_script: npm i
script:
  - npm test
```

Steps to Configure Travis CI

- Navigate to https://travis-ci.org/
- Sign in / sign up
- Add a new repository
- Connect your GitHub account and press `Sync Account`
- Filter the repositories until you see the one you want.
- Click on the toggle so it changes from an `X` to a checkmark.
- Click on the repo name to see the build status

Configure GitHub for Travis CI

- Navigate to GitHub page for the repo you want to configure
- Go to the `Settings` tab
- Choose `Integration and Services`
- You might see `Travis CI` already listed
  - If not, choose `Add Service` and find it
- Click on `Travis CI` to edit it

Protecting Branches

- Navigate to GitHub page for the repo you want to configure
- Go to the `Settings` tab
- Choose `Branches`
- Under `Protected Branches` choose the branch you want to protect (`master`)
- Check `Protect this branch`
- Check `Require status checks to pass before merging`
- Check `continuous-integration/travis-ci`

Now no one can marge code back into master unless the Travis CI tests pass.

Install Heroku Command Line Interface

Download the Heroku CLI to get access to Heroku tooling in your terminal.

OSX:

```
brew install heroku/brew/heroku
```

Ubuntu:

```
wget -qO- https://cli-assets.heroku.com/install-ubuntu.sh | sh
```

Windows: visit the download page linked above

Test to see if it's installed by running `heroku` in the terminal

```
heroku
```

Steps to Deploy to Heroku

- Navigate to http://heroku.com
- Sign In / Sign Up
- Choose New, Create new app
- Enter a globally-unique app name
- Set Deployment method to `Connect to GitHub`
- Search for a repo, click `Connect`
- Configure automatic deploy to happen from the `master` branch
- Check `Wait for CI to pass`
- Click `Enable Automatic Deploys`
- Click `Deploy Branch` in the manual deploy section to see things get started!
- Once it's deployed click the `View` button to see the app in action!

Debugging Heroku Deployment

When you try to view your page you might see a big error page. Heroku allows you to view the logs via the `heroku` command line tool.

```
heroku apps
heroku logs --app [[app_name]]
```

- Run `heroku apps` to see a list of all your apps
- Run `heroku logs --app [[app_name]]` to view logs for

```
➜  ci-cd-demo git:(master) heroku logs --app ci-cd-demo-401n6
2018-03-29T00:35:55.796043+00:00 app[api]: Initial release by user stevegeluso@gmail.com
2018-03-29T00:35:55.796043+00:00 app[api]: Release v1 created by user stevegeluso@gmail.com
2018-03-29T00:35:56.030577+00:00 app[api]: Enable Logplex by user stevegeluso@gmail.com
2018-03-29T00:35:56.030577+00:00 app[api]: Release v2 created by user stevegeluso@gmail.com
2018-03-29T00:39:34.000000+00:00 app[api]: Build started by user stevegeluso@gmail.com
2018-03-29T00:39:58.590283+00:00 app[api]: Deploy b027cdf6 by user stevegeluso@gmail.com
2018-03-29T00:39:58.590283+00:00 app[api]: Release v3 created by user stevegeluso@gmail.com
2018-03-29T00:39:58.607194+00:00 app[api]: Scaled to web@1:Free by user stevegeluso@gmail.com
2018-03-29T00:39:34.000000+00:00 app[api]: Build succeeded
2018-03-29T00:40:02.807716+00:00 heroku[web.1]: Starting process with command `npm start`
2018-03-29T00:40:05.107194+00:00 app[web.1]: npm ERR! missing script: start
2018-03-29T00:40:05.113863+00:00 app[web.1]:
2018-03-29T00:40:05.114089+00:00 app[web.1]: npm ERR! A complete log of this run can be found in:
2018-03-29T00:40:05.114217+00:00 app[web.1]: npm ERR!     /app/.npm/_logs/2018-03-29T00_40_05_108Z-debug.log
2018-03-29T00:40:05.227443+00:00 heroku[web.1]: State changed from starting to crashed
2018-03-29T00:40:05.230206+00:00 heroku[web.1]: State changed from crashed to starting
2018-03-29T00:40:05.186499+00:00 heroku[web.1]: Process exited with status 1
2018-03-29T00:40:09.877098+00:00 heroku[web.1]: Starting process with command `npm start`
2018-03-29T00:40:11.544726+00:00 heroku[web.1]: Process exited with status 1
2018-03-29T00:40:11.491591+00:00 app[web.1]:
```

Common Heroku Errors

- Make sure `package.json` has a `start` script
- Make sure your `start` script uses the plain `node index.js`
- If your `start` script uses `nodemon` then make sure `nodemon` is listed as a dependency in your `package.json`

Configuring MongoDB on Heroku

- Got to your app on Heroku
- Go to the Overview tab
- Click `Configgure Add-ons`
- Search for `mLab MongoDB` and choose it
- Choose the `Sandbox - Free` plan and press `Provision`
- Heroku and mLabs automatically set the `MONGODB_URI` environment property for your deployed app

For further documentation visit their article in the Heroku Dev Center:

https://devcenter.heroku.com/articles/mongolab