The Angular DevOps Series: CT/CI with Travis CI and GitHub Pages

Using Travis CI to implement Continuous Testing (CT) and Continuous Integration (CI) to deploy our Angular Application to GitHub Pages

Todd Palmer Follow
Oct 25, 2018 · 6 min read

Travis CI and GitHub

This article describes how to use Travis CI to watch the master branch on our GitHub repository for changes. We will configure Travis CI so that upon pushing a new commit for our Angular application it will:

- Run lint
- Run our Unit Tests
- Run our E2E Tests
- Build our application for production
- Deploy to GitHub Pages

The Angular DevOps Series

This post is the second post of the **Angular DevOps Series**. Make sure you take a look at the posts by my fellow Angular in Depth writers. Tim Deschryver explains how to leverage SemVer for your Angular Library. And of course, you all know how much I love

Angular Libraries! Also, Andrew Evans shows you how to use CircleCI to deploy to Firebase.

- Semantically release your Angular library
- CT/CI with Travis CI and GitHub Pages
- Deploying to Firebase with CircleCI

Definitions

To get this party started let's look at a few definitions. And yes I admit it, I copied these from Wikipedia:

Continuous Testing (CT)

Continuous testing (CT) is the process of executing automated tests as part of the software delivery pipeline to obtain immediate feedback on the business risks associated with a software release candidate.

Continuous Integration (CI)

In software engineering, **continuous integration (CI)** is the practice of merging all developer working copies to a shared mainline several times a day.

In Practice

OK, those definitions are fine. But, in practice what do they really mean? For the purposes of this article I am going to use some similar but very pragmatic and specific definitions:

CT
Our tests get run automatically before every deployment. And, our code will not be deployed if any tests fail.

CI
Pushing changes to a specific branch in our GitHub Repository initiates an automated deployment process.

Create an Angular Application

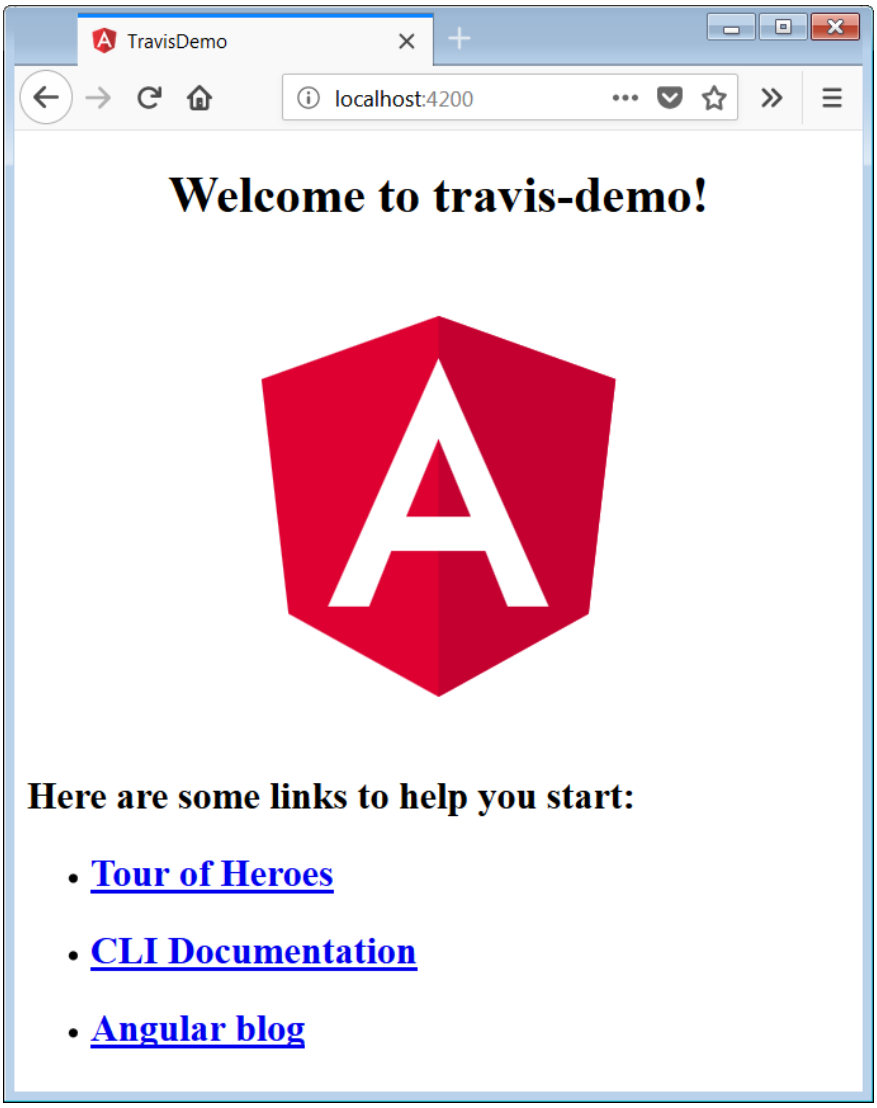
Like most of my articles I am going to walk through this with a sample application. To get started create a new Angular application using **Angular CLI** and test it locally.

```
ng new travis-demo
cd travis-demo
```

See my article on supporting Internet Explorer if you need it.

```
ng serve
```

Go to:
http://localhost:4200/



GitHub

Create a new GitHub repository: `travis-demo`

Now we need to add the GitHub repository as a remote to our local Git repository. You may need to commit first if you made any updates.

```
git commit -am "initial"
git remote add origin https://github.com/t-palmer/travis-demo.git
git push -u origin master
```

Make sure you replace **t-palmer** with your own GitHub ID.

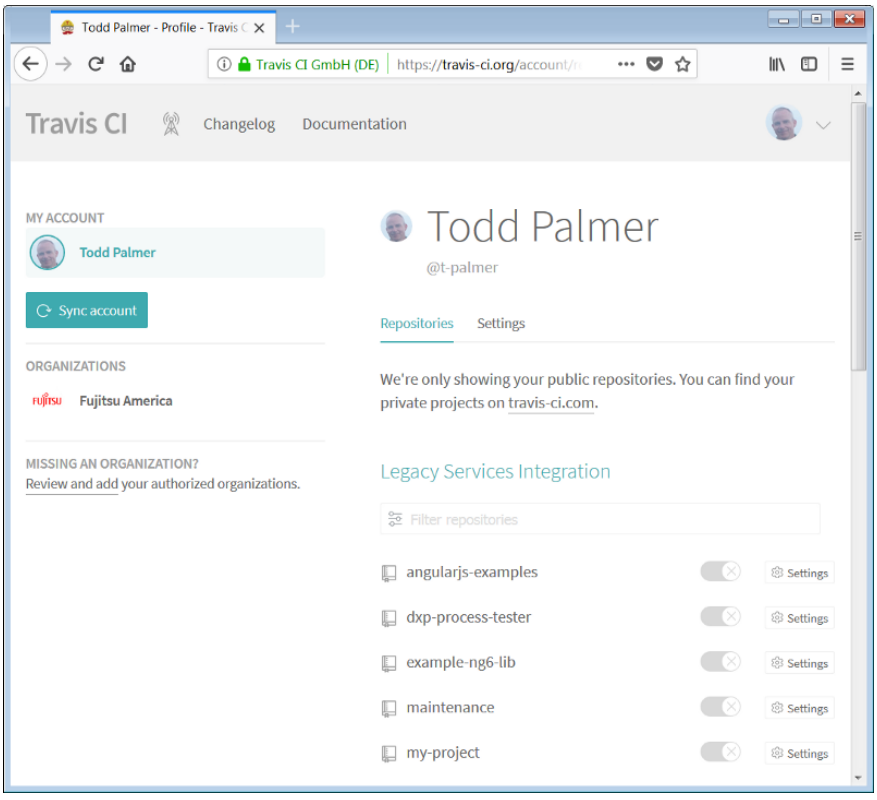
Travis CI

Register with Travis CI using your GitHub account.

Go to your profile to see the list of repositories.

If you already have an account, you will need to use **Sync account** to get your new repository to show up in the list. Also, you may need to use the filter in order to actually see your repository if you have a lot of them.

You can click on the **Settings** button to the right of the repository.



Add Travis CI Configuration

In order to tell Travis CI to do something with our repository, we need to add a Travis CI configuration file.

Create a file in the root of your workspace called **.travis.yml** with the following contents:

```
language: node_js
node_js:
  - "9"
dist: trusty
sudo: required

branches:
  only:
  - master

before_script:
  - npm install -g @angular/cli

script:
- ng lint
- ng build --prod --base-href https://t-palmer.github.io/travis-demo/
```

For the `--base-ref` you will need to use your own GitHub id.

This file tells Travis CI to do the following:

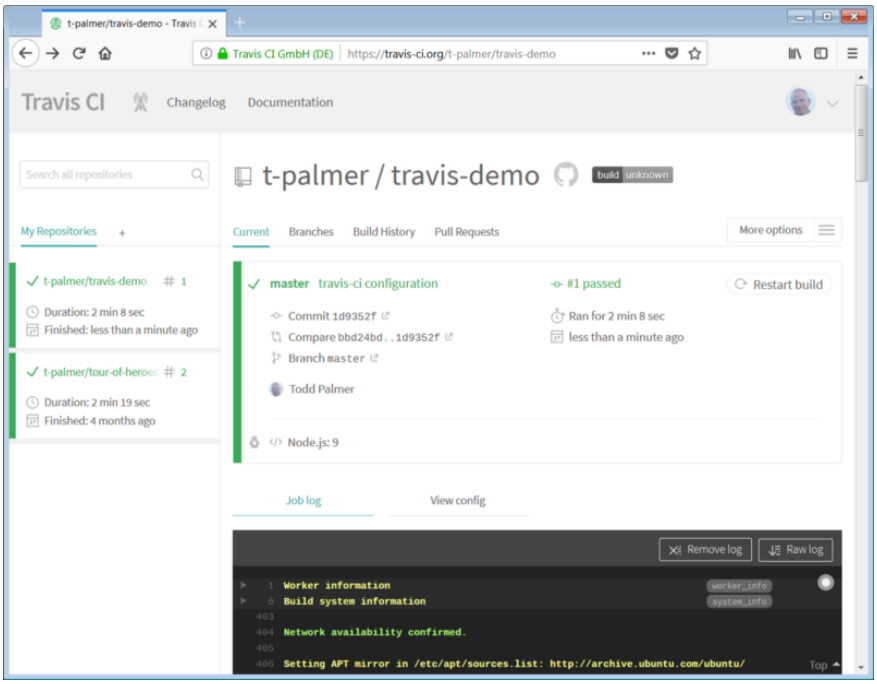
1. Use **Node.js**
2. Only watch the **master** branch for changes.
3. Before running any of the scripts install **Angular CLI**
4. Make sure the application passes our linting rules using `ng lint`.
5. Build our application using `ng build`.

After adding the Travis CI configuration file, commit and push.

```
git add .travis.yml
git commit -am "Travis CI configuration"
git push
```

After you push, jump over to the **Travis CI Dashboard** to watch the Travis CI log as it builds your application. You may need to click on the **Current** tab to see it building. Over in the far right of the log you can click on **Follow log** to make it more convenient to see the updated log.

When Travis CI has completed our script and built the application you will see the icon turn green to let you know your script has completed successfully.



Continuous Testing

Now that we have our application building let's venture into the world of **Continuous Testing** by having **Travis CI** run our **Unit Tests** whenever we push a new commit.

Unfortunately, we can't just `npm run test` on Travis CI. This is because we can't actually launch a browser GUI. So, we will use **Headless Chrome** to run our Unit Tests.

I have detailed the simple steps for setting up **Headless Chrome for Unit Testing** in my article:
Angular Testing with Headless Chrome

Please follow the instructions to create a new npm script like this:

```
"test-headless": "ng test --watch=false --
browsers=ChromeHeadless",
```

Now we can run our Unit Tests using Headless Chrome like this:

```
npm run test-headless
```

Now add `npm run test-headless` to our **.travis.yml** file in the `script` section like this:

```
script:
  - ng lint
  - npm run test-headless
  - ng build --prod --base-href https://t-
palmer.github.io/travis-demo/
```

Commit and push to GitHub:

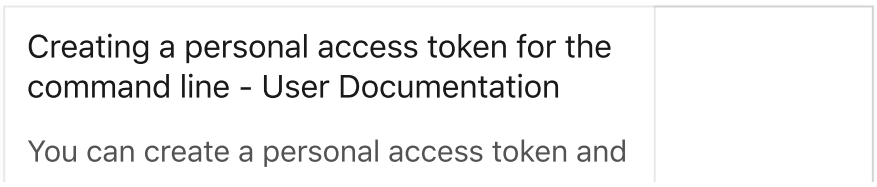
```
git commit -am "Added unit for Travis CI using Headless
Chrome"
git push
```

After pushing, switch to the **Travis CI Dashboard** to watch your project run the Unit Tests and then build.

Continuous Integration

Now that we are testing and building, we want to deploy our application.

We need to create a **GitHub Personal Access Token**. Follow these directions:

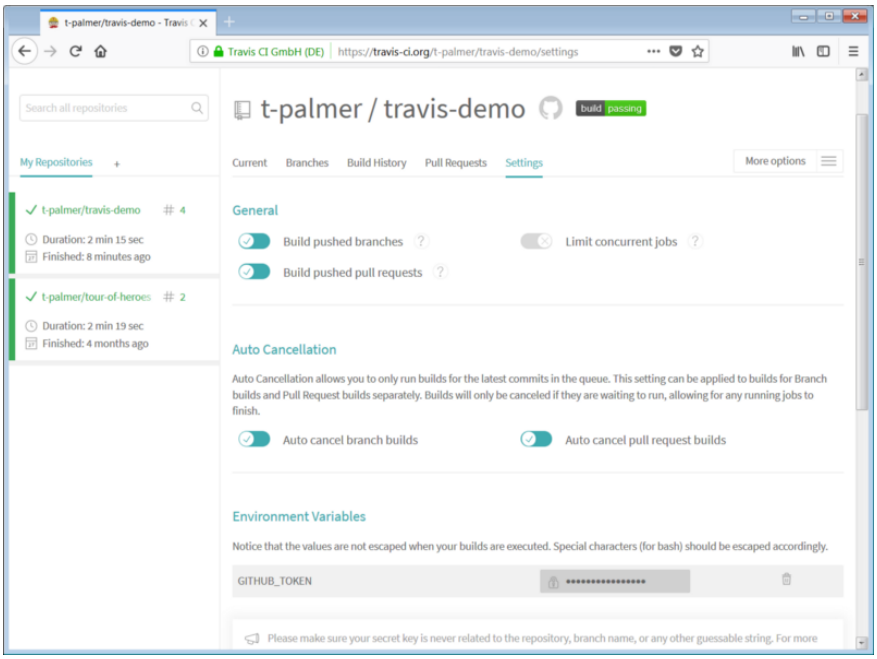


use it in place of a password when...
help.github.com

In the **Travis CI Dashboard** go to your project and over in the top right click on **More options : Settings**. Scroll down to the section for **Environment Variables**.

Add a new environment variable named: `GITHUB_TOKEN`

For the value, paste in the actual value of your GitHub Personal Access token that you generated. Click **Add**.



Deploying to GitHub Pages

Finally, let's set up Travis CI to deploy to GitHub Pages.

Add a deploy section to your **.travis.yml**:. At this point your file should look like this:

```
language: node_js
node_js:
  - "9"
dist: trusty
sudo: required

branches:
  only:
```

```
    - master

  before_script:
    - npm install -g @angular/cli

  script:
    - ng lint
    - npm run test-headless
    - ng build --prod --base-href https://t-
  palmer.github.io/travis-demo/

  deploy:
    provider: pages
    skip_cleanup: true
    github_token: $GITHUB_TOKEN
    local_dir: dist/travis-demo
    on:
      branch: master
```

Again, make sure you are using **your own GitHub id** instead of t-palmer.

This tells Travis CI to use our GitHub Token from the Environment Variable:
`$GITHUB_TOKEN`

Also, it specifies that the files to deploy come from our **dist** folder:
`dist/travis-demo`

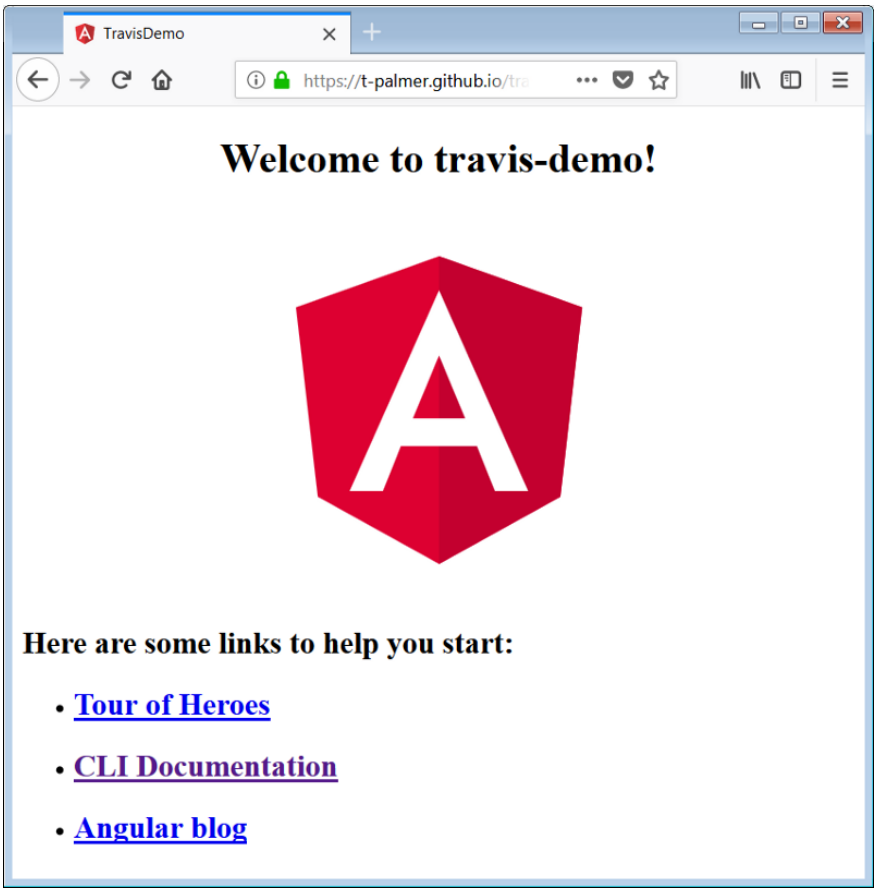
Commit and push to GitHub:

```
git commit -am "Deploy to GitHub pages"
git push
```

And, we should be able to go to our GitHub pages URL and see our application deployed.

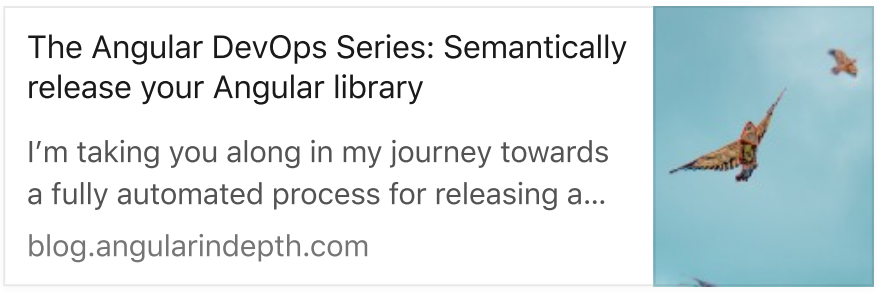
```
https://t-palmer.github.io/travis-demo/
```

Mine looks like this:



Further Reading

If you found this interesting, make sure you check out the article by my friend Tim Deschryver



If you want more information about **GitHub Pages**, you will find the documentation here: https://help.github.com/articles/user-organization-and-project-pages/

. . .


3 reasons why you should follow
Angular-In-Depth publication