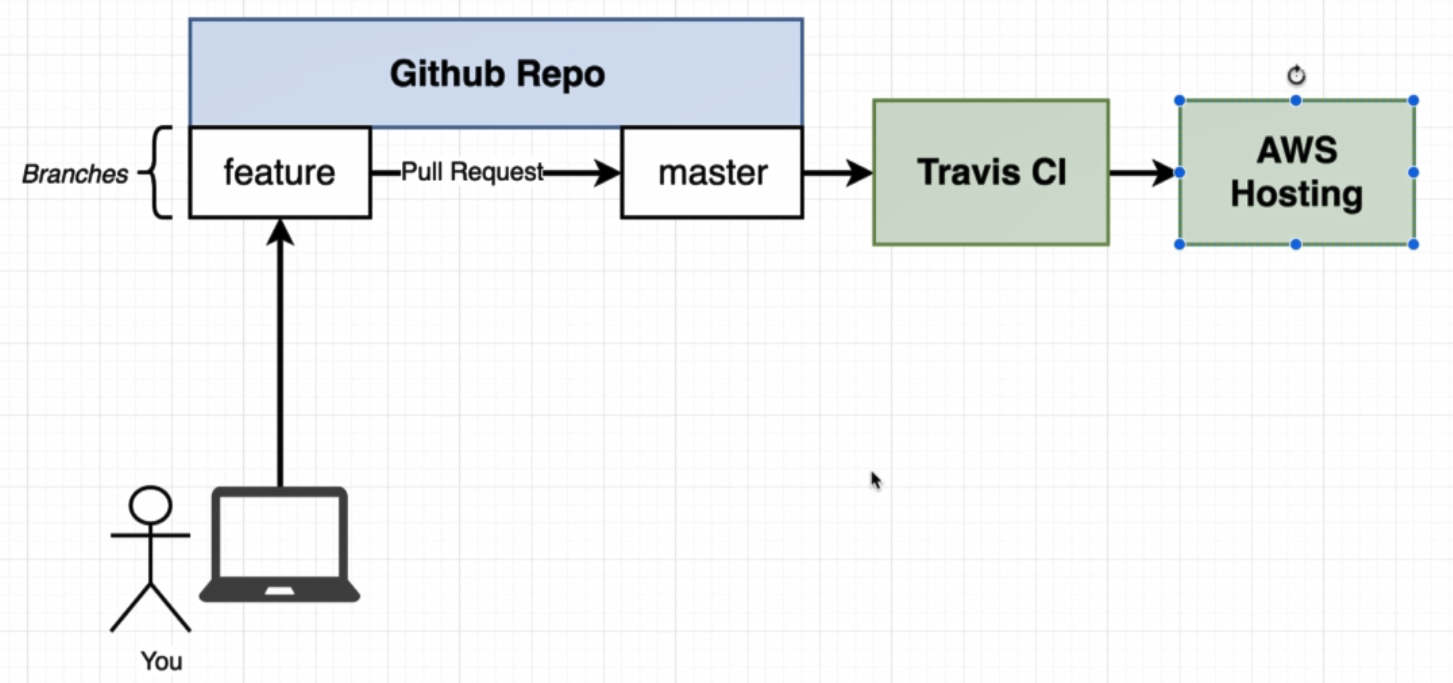
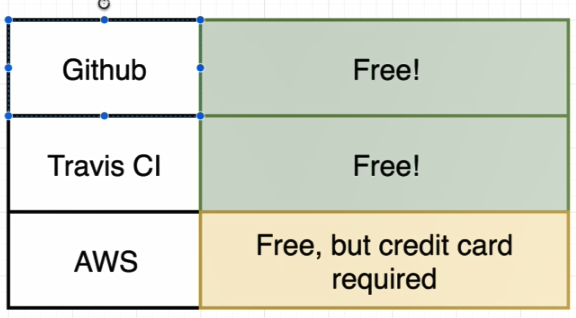
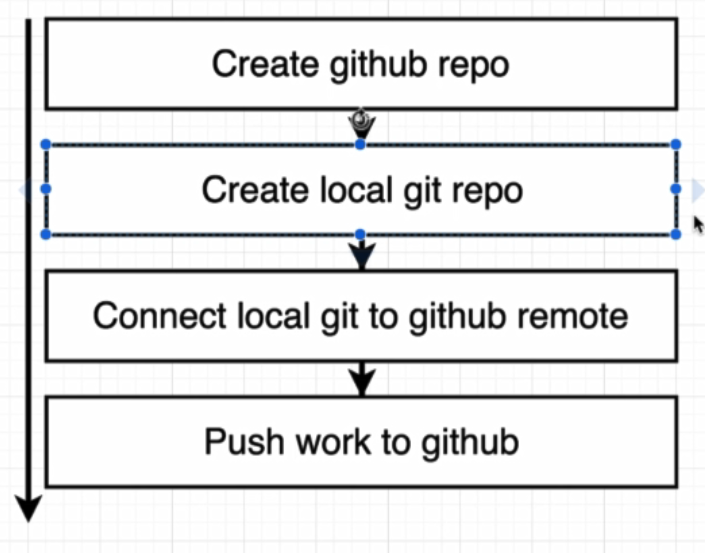
**07 Continuous Integration and Deployment with AWS**

In this section we will work on following image:



Setup Github repository as bellow:

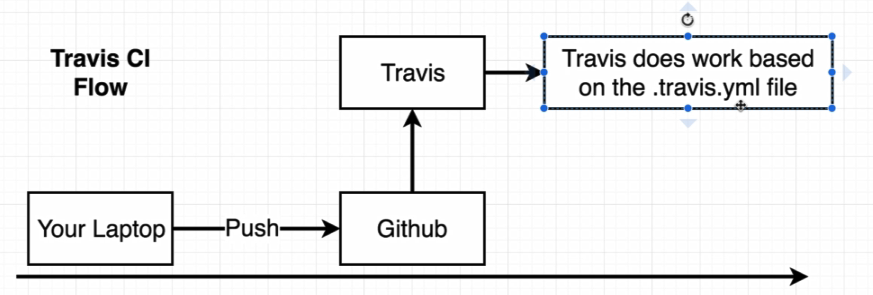


Inside of project folder write command bellow to initial project to github and push to it:

* git init
* git add .
* git commit –m “initial commit”
* git remote add origin <https://github.com/metehantagizade/docker-react.git>
* git push origin master

Setup Travis CI:

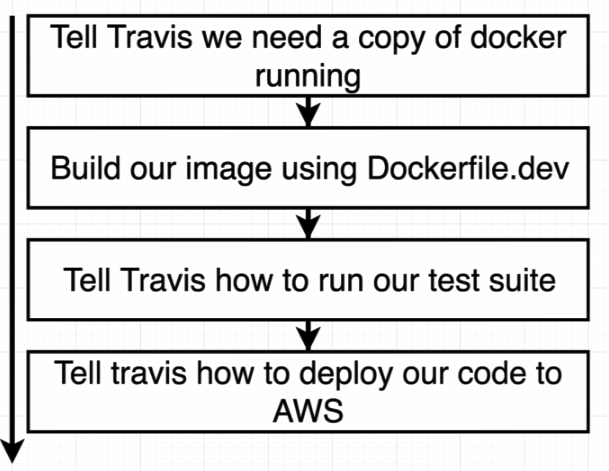
The entire idea of Travis CI is to watch for any time that you push some changes or some amount of code to the github project that we just created. On every commit and push to github, it tap on the sholder of Travis.



Travis CI then automatically pull down all of the code inside of out github repository and give as an ability to do some work. Traditionally people use Travis for either testing their codebase or for deployment.

**We will do the test on travis and after passing the test we will deploy it on AWS web server by using travis.**

First we have to create travis.yml file:



Create a file with name .travis.yml :

sudo: required # need super user level permission

services:

- docker # Install a copy of docker into our little running container.

befor\_install:

- docker build -t tohid1987/docker-react -f Dockerfile.dev .

# before do anything build an image

script: # contain all the different commands that need to be executed to

actually run our test suite.

- docker run tohid1987/docker-react npm run test -- --coverage

# coverage used to exit after test done

Now our travis file is ready and by pushing our code to Github, Travis CI will read .travis.yml file and do what the action described inside file.

* git add .
* git commit –m “travis file added to project”
* git push origin master

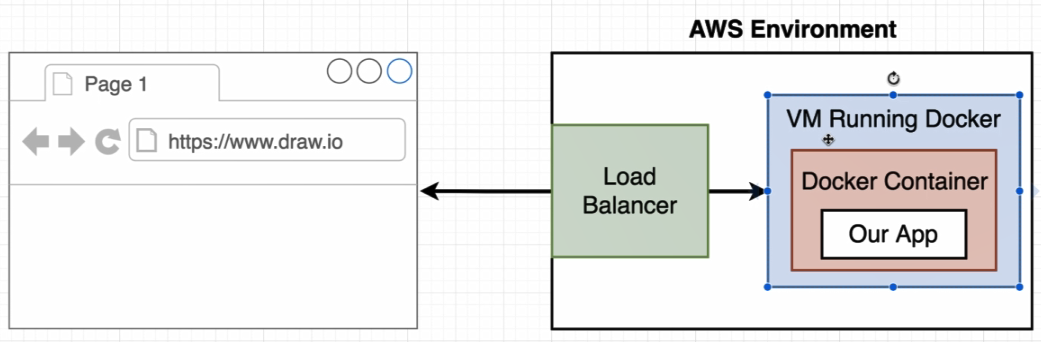
After push our code to Github, in <https://travis-ci.org/> page script inside .travis.yml will automatically run.

Now our application is ready to deploye to some outside hosting provider such as AWS, Windows Azure or Digital Ocean etc.

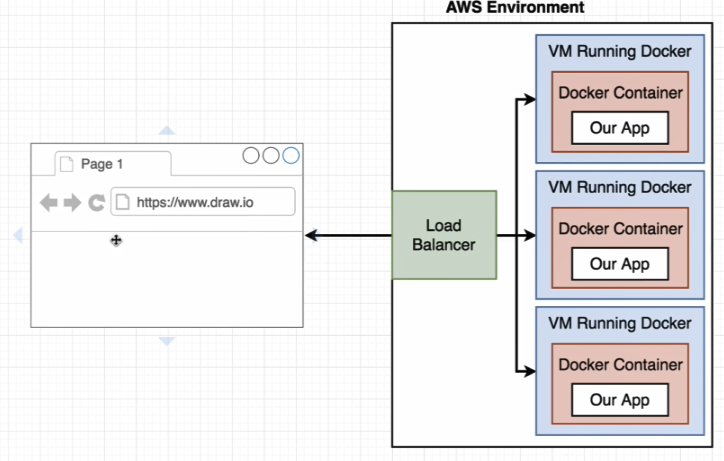
To deploy our project to AWS, we are going to use of **elastic beanstalk.**

* Create a new project with name docker-react.
* After that in opened page we have to create an environment for the project.
* In the new page select Web Server Environment
* In the next page and Base Configuration panel, change platform to Docker.
* Then click on Create Environment

What is going on inside elastic beanstalk:



User request from the browser come to loadbalancer inside elastic beanstalk. If the traffic reache a certain threshold, it going to automatically add in additional virtual machine to handle that traffic.



After create environment inside elastic beanstalk, it will give us a url to access to deployed project.

To automatically deploye our project after testing we have to add additional configuration inside .travis.yml file as bellow:

deploy:

provider: elasticbeanstalk

region: "region\_name" # get it from elasticbeanstalk created link. Docker-env.\*\*\*\*\*.region\_name.elasticbeanstalk.com

app: "docker-react"

env: "Docker-env"

# Name of the bucket that travis going to put our zip file into it. we can get it from S3 of AWS.

bucket\_name: "elasticbeanstalk-origin\_name-87687687687678"

bucket\_path: "docker-react"

on:

branch: master

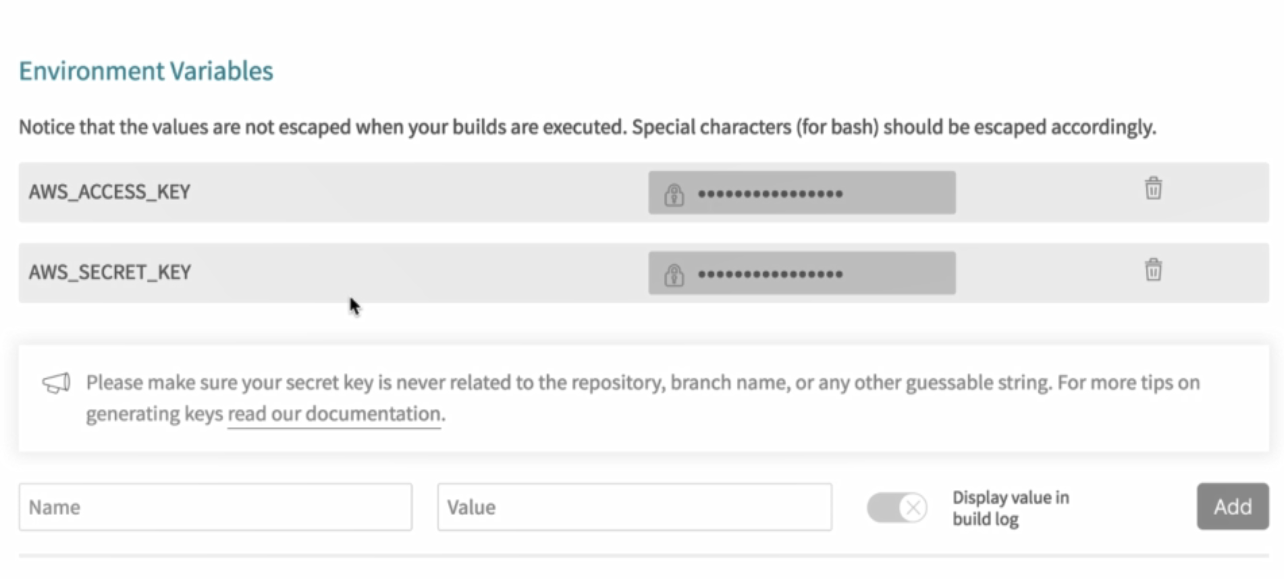
Now it’s time to generate API keys inside AWS to access our account over Travis.

* In AWS page search for IAM. It is a service to manage API keys.
* Click on users from opened page and add new user.
  + Username: docker-react-travis-ci
  + Check programmatic access
  + Click next permissions
  + Click on Attach existing policies directly
  + From last displayed on page select elastic beanstalk full access
  + Click on next review.
  + Click on create user

To avoid adding generated keys inside .travis.yml file we can use a feature **environment secret** provided by travis.

Inside travis-ci.org and setting section of recently created project we can find **Environment Variables** and fill text box as bellow:





And after define keys inside Travis, add lines bellow to .travis.yml file.

...

access\_key\_id: $AWS\_ACCESS\_KEY

secret\_access\_key:

secure: "$AWS\_SECRET\_KEY"

By commit and push code to github, the project will automatically deploy to AWS Web Provider.

If AWS return an error we should add line bellow inside Dockerfile after START nginx.

EXPOSE 80

**Github check out a new branch**

* git checkout –b feature
* git add .
* git commit –m “chaged app text”
* git push origin feature

to merge feature and master branch we can do it inside github:

