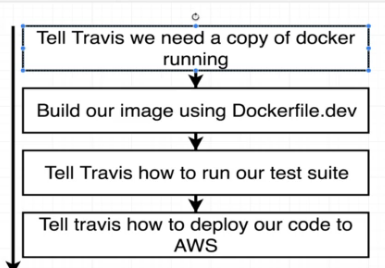
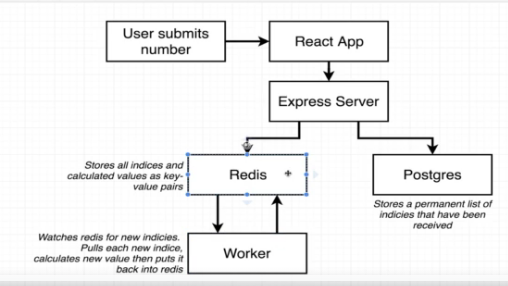
## Usage of Travis



Travis is same like Jenkins, used to build a image and



Its sample project,,worker is main where when a new value inserted into redis ,worker will monitor that

And it will cal fib and it will insert into redis

# Dockerizing multiple services

**Refer 118 (complex is the proj name)** project which contains only source code & doesn’t have any Dockerfiles or docker-compose files

**Refer 132** –which consists of all files including docker files and docker-compose files-refer this at last

Write docker files,docker-compose.yml files on your own

Refer checkpoint.zip and open that folder complex-where we will have 3 folders inside it

Got to folder named “client” and create a docker file and paste the below contents

In production docker files, container startup commands will be different-instead of “npm run dev”

It will be “npm run start” in “Dockerfile”

FROM node:16-alpine

USER node

RUN mkdir -p /home/node/app

WORKDIR /home/node/app

COPY --chown=node:node ./package.json ./

RUN npm install

COPY --chown=node:node ./ ./

CMD ["npm", "start"]

Create 2 similar “DockerFile.dev” in remaining folders but with different commands ,npm,run,dev

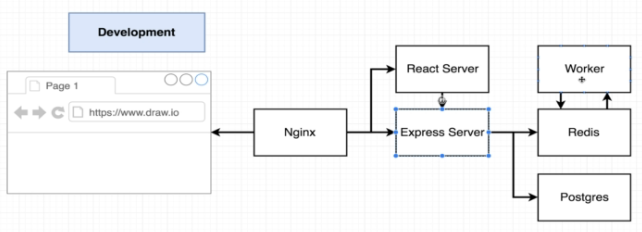
Due to a recent Postgres update, we need to make a change to our environment variables.

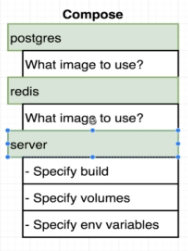
Add the following to your docker-compose.yml file in the **postgres** service:

1. postgres:
2. image: "postgres:latest"
3. environment:
4. - POSTGRES\_PASSWORD=postgres\_password

docker-compose down && docker-compose up --build

Please note, all of these changes are reflected in the downloadable checkpoint zip files.



here for postgres we used direct image sample from docker-compose.yml

version: '3'

services:

postgres:

image: 'postgres:latest'

I mean these are custom images, someone has already built those and pushed those images

To the repository with a custom name, generally if we built , we will get some random image name

But of course, we can build using custom name as “docker build –f <file name> -t <jar name/image name>”

### Setting environmental variable to container at run time

If your source code is using something like dollar

You need to provide ,and u can provide using docker-compose.yml file

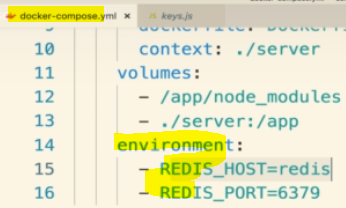
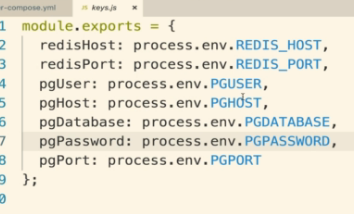
In docker-compose.yml file enter as below

Container name:

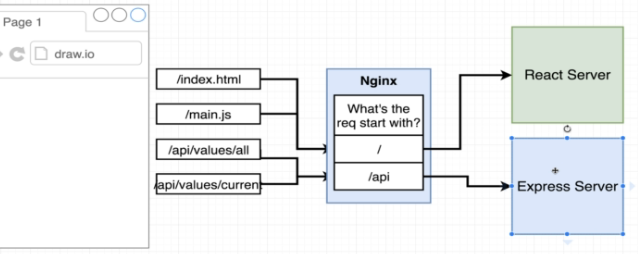
environment:

--key:value

--REDIS\_PORT:6379

  
 if your source is using environmental variables like in right side

U have to enter those in docker-compose.yml file as in left side



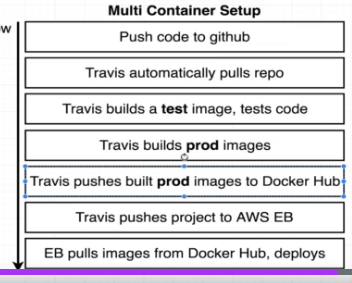
If request comes with /api it should redirect to express server

Else it should get redicted to react server

Refer default.conf

Refer project 132

### Multi container deployments



Create a travis ci yml file where we will keep/ write all docker commands

Refer 145,149 projects

If u use 149 then , u refer travis.yml file and u have to enter or configure those credentials in travis environmental

Variables

Sample travis.yml file

services:

- docker

before\_install:

- docker build -t stephengrider/react-test -f ./client/Dockerfile.dev ./client

script:

- docker run -e CI=true stephengrider/react-test npm test

after\_success:

- docker build -t stephengrider/multi-client ./client

# Log in to the docker CLI

- echo "$DOCKER\_PASSWORD" | docker login -u "$DOCKER\_ID" --password-stdin

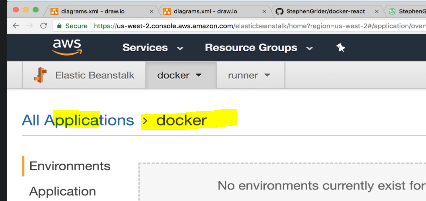
# Take those images and push them to docker hub

- docker push stephengrider/multi-client

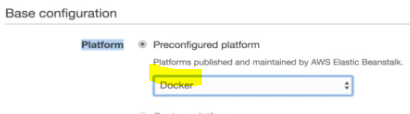
Login to aws.amazon.com

Elastic bean stalk is appropriate when we are using / running 1 container at a time

Search for it and click create an application refer video 89



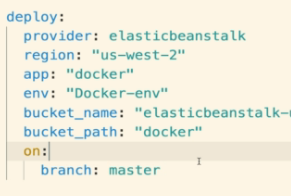
While creating environment ,select Docker



EBS is intelligent, if more traffic comes to that application then automcatically more instances

Will be added

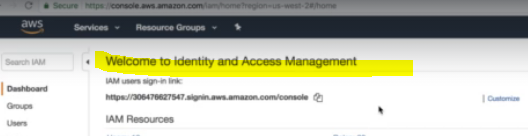
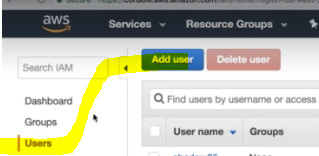
## Travis configurations to deploy in AWS



Now the word “on” says, when the code is merged to master branch, then once build over then it will creates a

Deployment

IAM is a service that is used to manage API keys

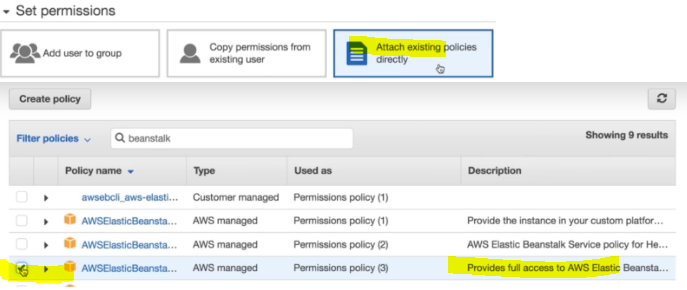
Same like service account credentials, we need to create credentials that will be used by travis CI

To push that image to the AWS server

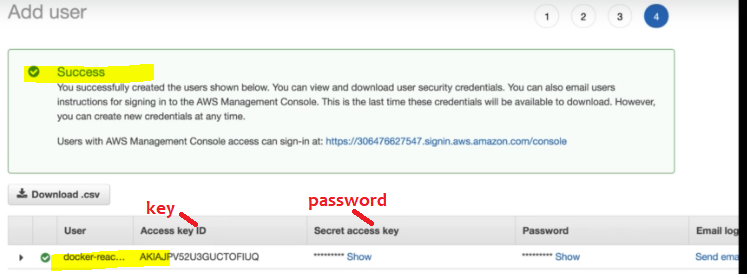
Steps to create service account credentials

1. Select “users”, select “Add user”
2. Select access type as “programmatic access”
3. Add an existing policy to the user, so that with those service account credentials, we should be

Able to push the image, to achieve that we have to add some existing policies to that user credentials



Once policies are added, u will get a api key



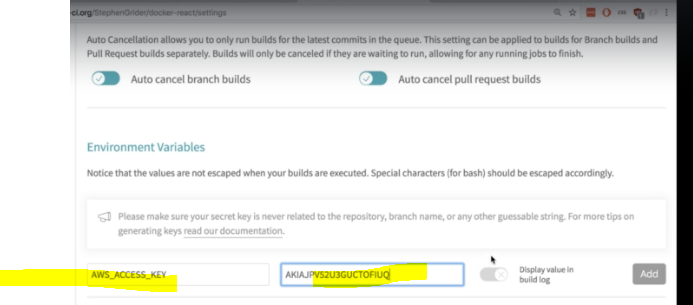
### Add/Configure AWS keys in Jenkins/Travis

Refer project number-93

And we have to keep those secrets in travis cI , ui only, if we keep those credentials in code, everyone will be

Able to see that , so don’t keep in code keep in travis CI tool itself

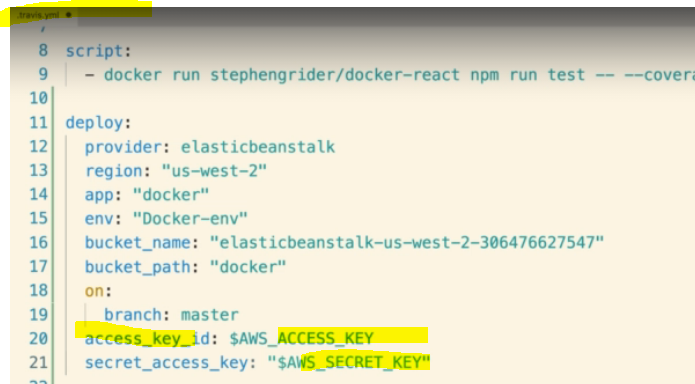
In real time we will keep those credentials in Jenkins tool





The above keys will be created either in Jenkins and in the code we will use those keys

### Use those keys in travis.yml



The travis CI must be able to execute this yml file, means it should also have a docker in it,.

Sample from “travis.yml” file

before\_install:

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

If travis is able to execute that command means ,travis also would have had docker in it

Else it can not execute “docker build “command at all