**Implementing the Sample Rails application using Docker**

Two approaches to implement,

1. **Bundle the application code with the container**

This could be the preferred approach for production environments provided the size of the application code is reasonably small and does not change frequently. This has the benefits of faster start up times as the code is already bundled with the container and is also not dependent on paths in the host environment (which could change across environments).

1. **Use bind mounts with the container to mount the application code from a path on the host**

This is the preferred approach for development environments as most changes to the application code as part of debugging can be tested quickly by making changes on the host machine and restarting the container. There is no need to re-build the image after every change to the source code. This could also be a potential solution if the application source code is too large in size to be bundled with each container, provided the organization has implemented version control to track source changes.

I will explain both the approaches here,

**Approach 1: Bundling application code with the container (Steps to implement)**

1. Clone the git repository in some path (say **/myapp**) on the host machine

git clone <https://bitbucket.org/railstutorial/sample_app_4th_ed.git> sample\_app\_reference

1. Create a folder for **nginx** in **/myapp/sample\_app\_reference** and create two files in it, **Dockerfile** and **nginx.conf**

Include the following in the **nginx.conf** file to configure nginx to listen to application port 3000 which will be exposed by the rails application

worker\_processes 4;

events { worker\_connections 1024; }

http {

upstream web-app {

least\_conn;

server app:3000 weight=10 max\_fails=3 fail\_timeout=30s;

}

server {

listen 80;

location / {

proxy\_pass http://web-app;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

}

Include the following in the **Dockerfile** of nginx to copy the nginx.conf to the container and override the default configuration of nginx

# Nginx Web Server

# Version 1.0

FROM nginx

# Overwrite config to listen to app service on port 3000

COPY nginx.conf /etc/nginx/nginx.conf

1. Modify the Gemfile present in **/myapp/sample\_app\_reference** to include the following ruby gem dependency for MySQL.

gem 'mysql2', '0.4.10'

1. Modify the following details in **database.yml** file present in **/myapp/sample\_app\_reference/config** to include details for the MySQL db.

default: &default

adapter: mysql2

pool: 5

timeout: 5000

encoding: utf8

username: root

password: sampleapp

host: db

port: 3306

Ensure the same password is included in the **docker-compose.yml** file later

1. Create a **Dockerfile** in **/myapp/sample\_app\_reference** with the following contents

# Rails Sample App

# Version 1.0

FROM ruby:latest

# Include the library for mysqlclient

RUN apt-get update -qq && apt-get install -y build-essential libpq-dev nodejs \ default-libmysqlclient-dev

# Create a directory in the container for the application

RUN mkdir /sampleapp

WORKDIR /sampleapp

# Copy the Gemfile to container and run bundle to install required dependencies

COPY Gemfile /sampleapp/Gemfile

COPY Gemfile.lock /sampleapp/Gemfile.lock

RUN bundle install

# Copy the application code to the container

COPY . /sampleapp

1. Create file **.dockerignore** with the below contents in **/myapp/sample\_app\_reference** which has the files/folders that need not be included as part of the image

# Ignoring nginx file/folder

nginx

1. Create the **docker-compose.yml** file in **/myapp/sample\_app\_reference** with the following content

version: "3"

services:

web:

build: ./nginx

ports:

- "80:80"

depends\_on:

- app

networks:

- frontend

db:

image: mysql:latest

expose:

- "3306"

environment:

- MYSQL\_ROOT\_PASSWORD=sampleapp

networks:

- backend

volumes:

- mysqldata:/var/lib/mysql

app:

build: .

command: bin/rails server --port 3000 --binding 0.0.0.0

depends\_on:

- db

expose:

- "3000"

networks:

- frontend

- backend

networks:

frontend:

backend:

volumes:

mysqldata:

The persistent storage required for the mysql db is managed using a **named volume** called **mysqldata**

The **db** service is isolated from the **web** service as they are on different networks. Only the **app** service which shares both the networks (frontend and backend) can talk to both services. This further improves security.

The **db** service is started first, followed by the **app** service and finally the **web** service, enforced by the **depends\_on** options.

1. Build the containers and setup the **db** using the following commands

docker-compose build

docker-compose up -d

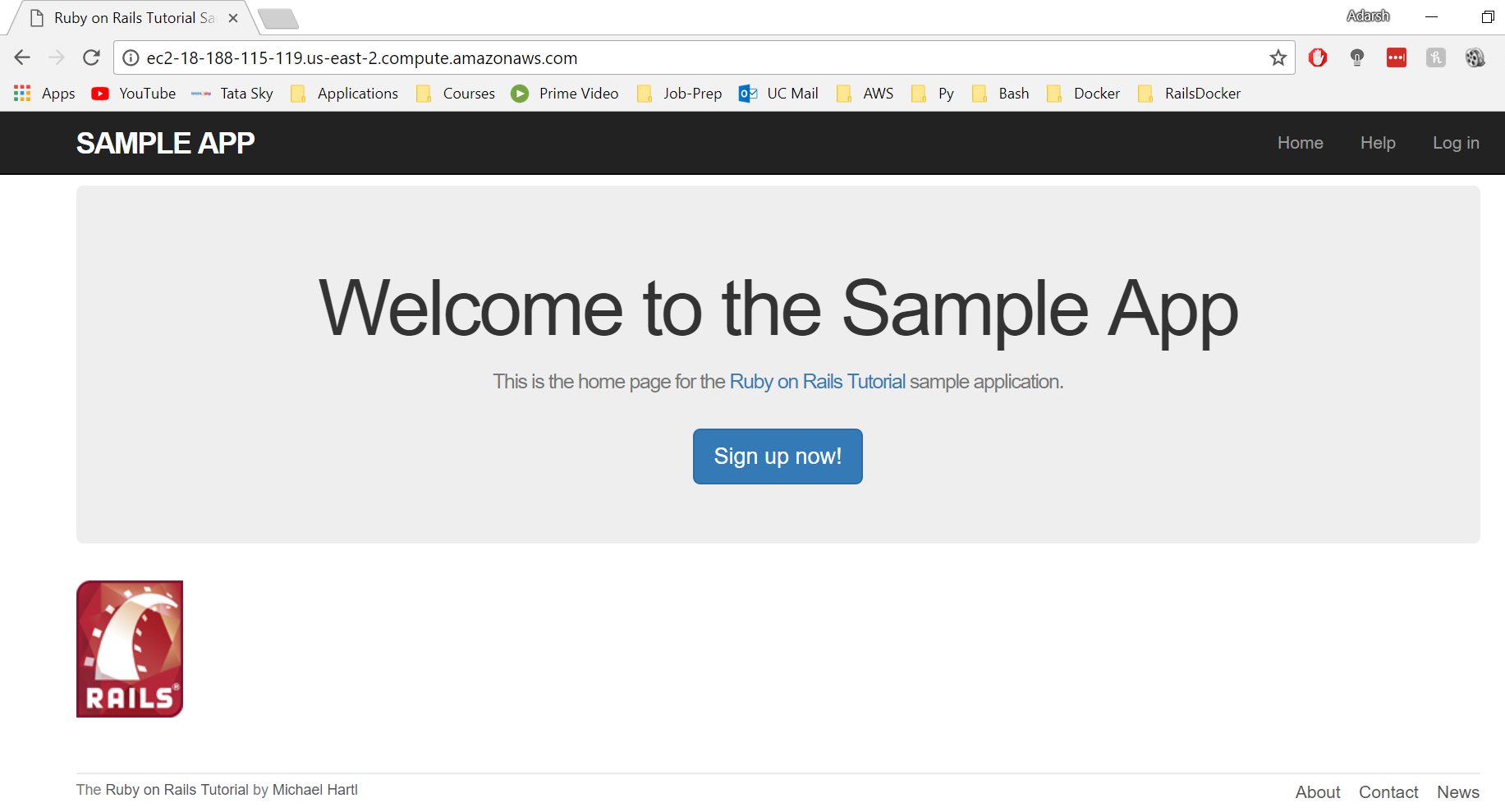
docker-compose run app bash

rake db:create

rails db:migrate

This creates three containers **sampleappreference\_web\_1**, **sampleappreference\_app\_1** and **sampleappreference\_db\_1**

1. Access the application at <http://localhost:80> (replace localhost with the dns name or IP of the VM if necessary)



1. Once the **db** has been setup for the first time, the application stack can be brought up/down using **docker-compose up** and **docker-compose down** commands. The db does not have to be migrated again as long as the named volume persists in the host machine.

**Approach 2: Use bind mounts with the container to mount the application code from a path on the host (Steps to implement)**

1. Clone the git repository in some path (say **/myapp\_dev**) on the host machine

git clone <https://bitbucket.org/railstutorial/sample_app_4th_ed.git> sample\_app\_reference

1. Create a folder for **nginx** in **/myapp\_dev** and create two files in it, **Dockerfile** and **nginx.conf**

Include the following in the **nginx.conf** file to configure nginx to listen to application port 3000 which will be exposed by the rails application

worker\_processes 4;

events { worker\_connections 1024; }

http {

upstream web-app {

least\_conn;

server app:3000 weight=10 max\_fails=3 fail\_timeout=30s;

}

server {

listen 80;

location / {

proxy\_pass http://web-app;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

}

Include the following in the **Dockerfile** of nginx to copy the nginx.conf to the container and override the default configuration of nginx

FROM nginx

COPY nginx.conf /etc/nginx/nginx.conf

1. Modify the Gemfile present in **/myapp\_dev/sample\_app\_reference** to include the following ruby gem dependency for MySQL.

gem 'mysql2', '0.4.10'

1. Modify the following details in **database.yml** file present in **/myapp\_dev/sample\_app\_reference/config** to include details for the MySQL db.

default: &default

adapter: mysql2

pool: 5

timeout: 5000

encoding: utf8

username: root

password: sampleapp

host: db

port: 3306

Ensure the same password is included in the **docker-compose.yml** file later

1. Create a **Dockerfile** in **/myapp\_dev/sample\_app\_reference** with the following contents

# Rails Sample App

# Version 1.1

FROM ruby:latest

# Include the library for mysqlclient

RUN apt-get update -qq && apt-get install -y build-essential libpq-dev nodejs \ default-libmysqlclient-dev

RUN mkdir /sampleapp

WORKDIR /sampleapp

# Copy the Gemfile to container and run bundle to install required dependencies

COPY Gemfile /sampleapp/Gemfile

COPY Gemfile.lock /sampleapp/Gemfile.lock

RUN bundle install

Notice that we are **not** copying the contents of the directory to the image above

1. Create the **docker-compose.yml** file in **/myapp/sample\_app\_reference** with the following content

version: "3"

services:

web:

build: ../nginx

ports:

- "80:80"

depends\_on:

- app

networks:

- frontend

db:

image: mysql:latest

expose:

- "3306"

environment:

- MYSQL\_ROOT\_PASSWORD=sampleapp

networks:

- backend

volumes:

- mysqldata:/var/lib/mysql

app:

build: .

command: bin/rails server --port 3000 --binding 0.0.0.0

depends\_on:

- db

expose:

- "3000"

volumes:

- .:/sampleapp # Current host directory mounted on to container

networks:

- frontend

- backend

networks:

frontend:

backend:

volumes:

mysqldata:

A key change in the docker-compose.yml file compared to previous approach is the addition of a **bind mount** for the **app** service, where the current directory in the host (**/myapp/sample\_app\_reference)** is mounted as the application directory in the container.

1. Build the containers and setup the **db** using the following commands

docker-compose build

docker-compose up -d

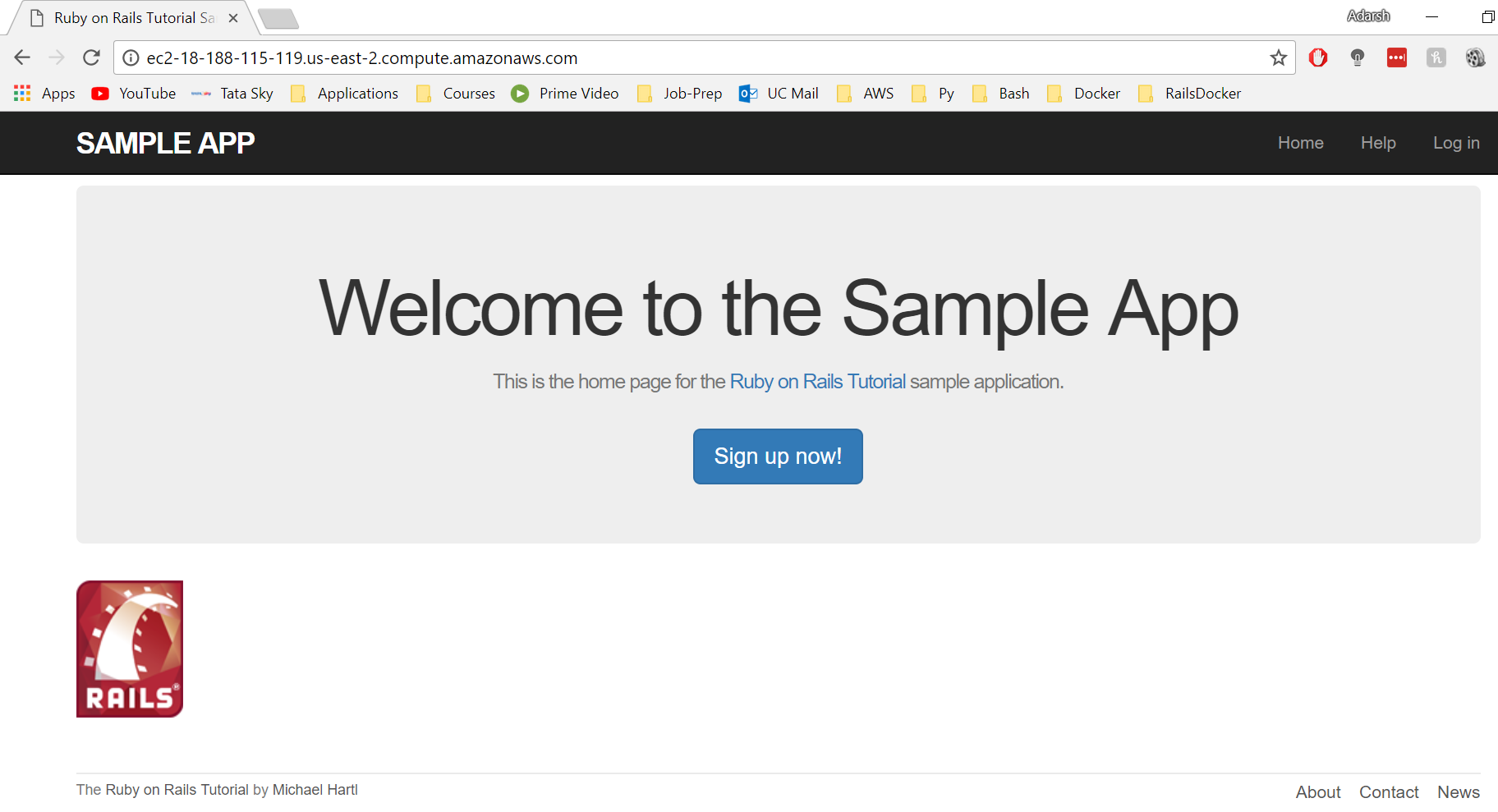
docker-compose run app bash

rake db:create

rails db:migrate

This creates three containers **sampleappreference\_web\_1**, **sampleappreference\_app\_1** and **sampleappreference\_db\_1**

1. Access the application at <http://localhost:80> (replace localhost with the dns name or IP of the VM if necessary)



1. Once the **db** has been setup for the first time, the application stack can be brought up/down using **docker-compose up** and **docker-compose down** commands. The db does not have to be migrated again as long as the named volume persists in the host machine.

Using this second approach, the application files can be changed dynamically for debugging and the results of the change can be seen on the application with or without a restart of the container (depending on the file changed). The application logs can also be viewed on the host machine itself.

**Some Observations**

1. The **rails test** command failed with the below error

ERROR["test\_micropost\_interface", Minitest::Result, 0.4482935449996148]

test\_micropost\_interface#Minitest::Result (0.45s)

ActiveRecord::StatementInvalid: ActiveRecord::StatementInvalid: Mysql2::Error: Incorrect datetime value: '2015-03-17 02:46:45 UTC' for column 'created\_at' at row 1: INSERT INTO `microposts` (`content`, `created\_at`, `updated\_at`, `id`, `user\_id`) VALUES ('Check out the @tauday site by @mhartl: http://tauday.com', '2015-03-17 02:46:45 UTC', '2018-03-17 02:46:45', 517500819, 762146111)

E 66/0: [ ] 0% Time: 00:00:00, ETA: ??:??:??

ERROR["test\_password\_resets", Minitest::Result, 0.4805328920010652]

test\_password\_resets#Minitest::Result (0.48s)

ActiveRecord::StatementInvalid: ActiveRecord::StatementInvalid: Mysql2::Error: Incorrect datetime value: '2018-03-17 02:50:30 UTC' for column 'activated\_at' at row 1: INSERT INTO `users` (`name`, `email`, `password\_digest`, `admin`, `activated`, `activated\_at`, `created\_at`, `updated\_at`, `id`) VALUES ('Michael Example', 'michael@example.com', '$2a$04$iDpf5u/lQYArIrWP/wn9mu4/BgQz5ok21p2sdx/y4bJ531RxKPjmu', 1, 1, '2018-03-17 02:50:30 UTC', '2018-03-17 02:50:30', '2018-03-17 02:50:30', 762146111)

E 66/0: [ ] 0% Time: 00:00:00, ETA: ??:??:??

This was fixed by making the following changes to **microposts.yml** and **users.yml** files in **/myapp/sample\_app\_reference/test/fixtures**

Changed all TIMEZONE date values for ‘**created\_at’** column in **microposts.yml** to DATETIME value

**Example:** Changed **created\_at: <%= 3.years.ago %>** to **created\_at: <%= 3.years.ago.to\_datetime %>**

Changed all TIMEZONE date values for ‘**activated\_at’** column in **users.yml** to DATETIME value

**Example:** Changed **activated\_at: <%= Time.zone.now %>** to **activated\_at: <%= Time.zone.now.to\_datetime %>**

The test passed successfully after the above changes,

[root@ip-172-31-7-243 fixtures]# docker exec -it sampleappreference\_app\_1 bash

root@cf3710e04b8f:/sampleapp# rails test

Running via Spring preloader in process 65

Started with run options --seed 55780

Run options: --seed 55780--=---=---=---=---=-] 0% Time: 00:00:00, ETA: ??:??:??

# Running:

... 66/66: [=================================] 100% Time: 00:00:02, Time: 00:00:021

.

Finished in 2.30699s

66 tests, 372 assertions, 0 failures, 0 errors, 0 skips

Finished in 2.306770s, 28.6114 runs/s, 161.2644 assertions/s.

66 runs, 372 assertions, 0 failures, 0 errors, 0 skips

root@cf3710e04b8f:/sampleapp#

1. To activate signed up users
   1. Change the following details in **/myapp/sample\_app\_reference/config/environments/development.rb**

host = 'localhost:3000' # Local server

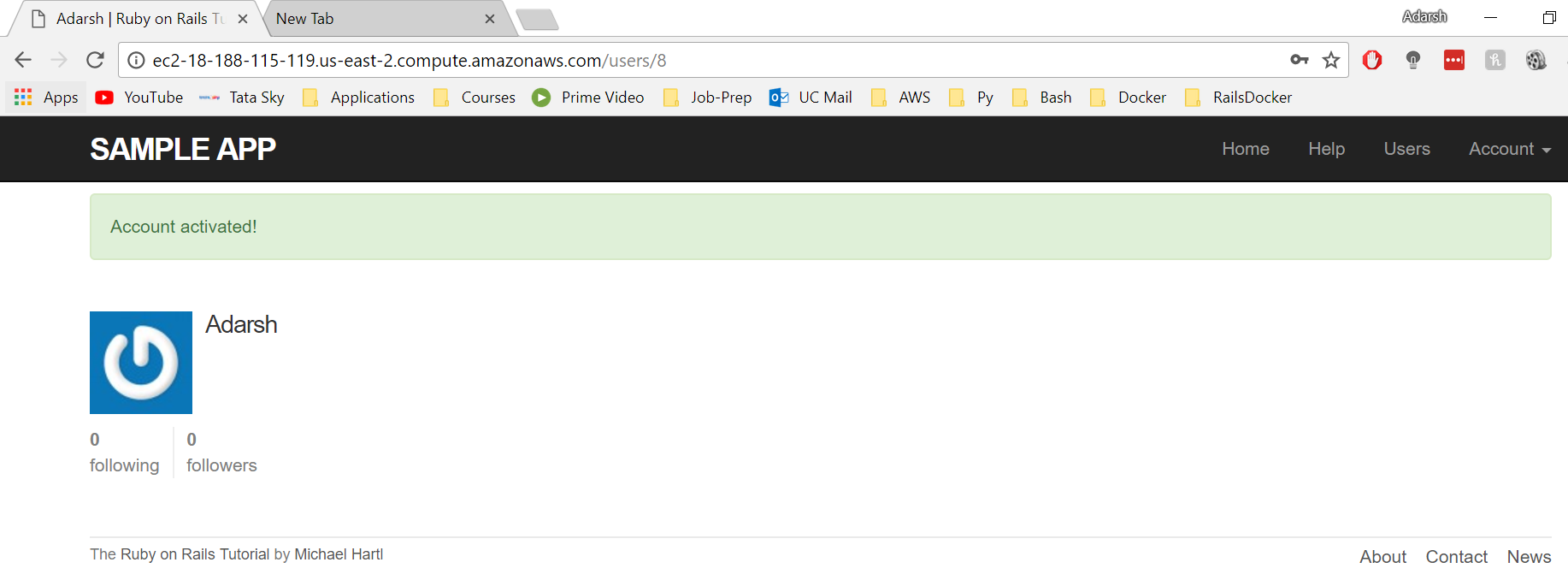
config.action\_mailer.default\_url\_options = { host: host, protocol: 'http' }

* 1. Register a new user using the web application.
  2. Check the development.log in **/myapp/sample\_app\_reference/log**

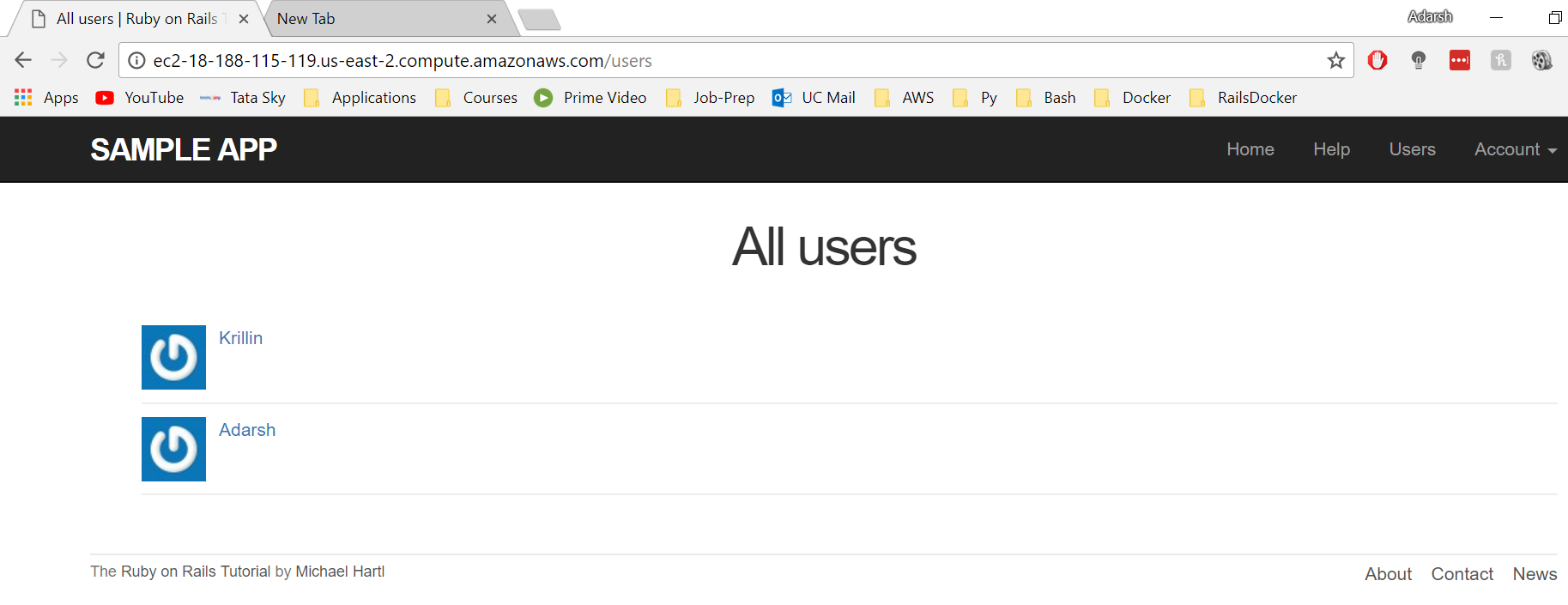
1. Welcome to the Sample App! Click on the link below to activate your account:
2. </p>
3. <a href="http://localhost:3000/account\_activations/d7CDFSdTIPy5b2pL3wfPOg/edit?email=adarsh%40example.com">Activate</a>
4. </body>
5. </html>
6. ----==\_mimepart\_5aad7bf05c7b6\_c2ab001494c342681b--
7. Redirected to http://ec2-18-188-115-119.us-east-2.compute.amazonaws.com/
8. Completed 302 Found in 487ms (ActiveRecord: 2.4ms)
9. Visit the highlighted URL to activate the user (replace localhost with the dns name or IP of the VM if necessary)

**Sample Screens**

1. New user signed up and account activated



1. Users page



1. Home page for Logged in user - with a post

