**Docker Lab**

On your VM:

1. From home directory:

**mkdir nodeapp**

**cd nodeapp**

**git clone ssh://bdfvks@solace.ist.rit.edu/var/courses/repos/341/dockerlab** (replace bdfvks with your RIT username.

**mv dockerlab/index.js .**

**mv dockerlab/package.json .**

**rm -rf dockerlab**

1. Create a docker image with our app:
   1. In a browser (either on host or VM) and go to: <https://hub.docker.com/explore/>
   2. Search for “node”
   3. Pick the “official” repository and look at the tags available.
   4. Go to <http://nodejs.org> in another tab/window to see what the current LTS version is.
   5. Read the information about choosing which image to use near the bottom of the page (same considerations apply to picking other images)
   6. In the VM and in the nodeapp folder:

**vscode Dockerfile** and make it:

# click on “tags” and pick a version that contains the latest stable version of Node.JS,

# you can always change your mind if it doesn't work

FROM node:carbon

# Create app directory

RUN mkdir -p /usr/src/app

WORKDIR /usr/src/app

# Install app dependencies

COPY package.json /usr/src/app/

RUN npm install

# Bundle app source

COPY . /usr/src/app

EXPOSE 3000

CMD [ "node", "index.js" ]

**<save it>**

* 1. Answer the following questions:  
     1. What does the FROM command do?

FROM command is used to specify the parent image

* + 1. What is the difference between RUN, CMD and ENTRYPOINT?

RUN execute the commands in a new layer over the image. CMD provides the default arguments for the ENTRYPOINT. ENTRYPOINT is used to configure a container to run as executable. ENTRYPOINT overrides the CMD values.

* + 1. What does the WORKDIR command do?  
          
       WORKDIR sets the working directory for a RUN, CMD, ADD and other commands which follows it.
    2. What does the COPY command do?  
         
       COPY copies files or directories from source and adds them to the filesystem of the container at the destination.
    3. What does the EXPOSE command do?  
         
       EXPOSE lets the Docker know that the container listens at the specified port at runtime. Default protocol is TCP but can be changed to UDP as well.
  1. Run: **sudo** **docker build -t {your-name}/{your-app-name}:{tag} .** Replace the {text} with your information. You can use “1.0” for the version and don’t forget the “.”.

You can ignore the warnings, etc.

* 1. Since we need Mongo as well:

**vscode docker-compose.yml** and make it:  
  
version: '3'

services:

mongo:

image: mongo

command: mongod

networks:

- all

web:

#use the values from Dockerfile above

image: **{your-name}/{your-app-name}:{tag}**

ports:

- "8080:3000"

depends\_on:

- mongo

restart: always

networks:

- all

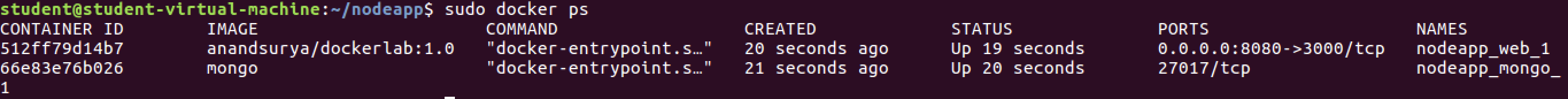
environment:

MONGODB\_URI: "mongodb://mongo:27017/accounts"

networks:

all:  
  
**<save it>**

* 1. **sudo docker-compose up -d**
  2. **sudo docker ps** 
     1. put a screen shot of the output here:



* 1. **ip addr show ens33** (copy the inet address)
  2. On your Mac, open Postman:

Create a “GET” request to http://<ip address>:8080/messages – SEND

Paste your reponse here:

Graphical user interface, text, application, email

Description automatically generated

Create a “POST” request to the same URL, click on “Body”, then “raw”, change MIME to “application/json”, then enter JSON similar to: {"message":"this is a test","name":"bryan french"} and click SEND.

Past your response here:

Graphical user interface, text, application, email

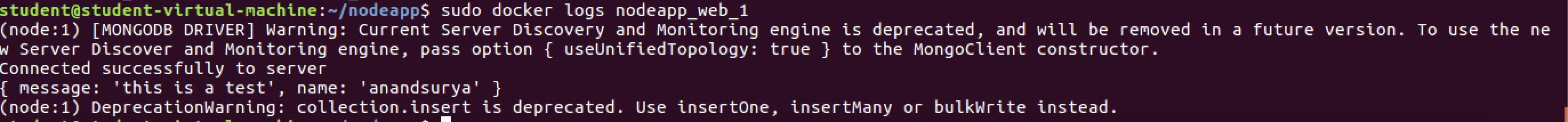
Description automatically generated

Send the GET request again and paste your response here:

Graphical user interface, text, application, email

Description automatically generated

Go to the VM: **sudo docker logs nodeapp\_web\_1** and paste the response here:

****