Containerizing Node.js application on Docker

The primary goal of this example is to show you how to allow Node.js application into Docker container. For this example, you need Docker installed and a basic understanding of Node.js. I have used Ubuntu for this setup. Node.js container generally have Dockerfile that comprises of the app and will start when Docker engine initiates container off image.

Setup comprises of 1. Connect container to docker Host

2. Connect container to Database (PostgreSQL for database connectivity)

1. Update your system

   apt update && sudo apt upgrade

2. Install PostgreSQL

   apt install postgresql postgresql-contrib

3. Change the Postgres user's Password

   passwd postgres

4. Set password for postgres database user

   su - postgres
   psql -d template1 -c "ALTER USER postgres WITH PASSWORD 'yourpassword';"

5. Create Database for app and connect

   Created nodejs
   Psql nodejs
   Add hello world to the DB
   nodejs=# CREATE TABLE hello (message varchar);
   nodejs=# INSERT INTO hello VALUES ('Hello world');
   nodejs=# \q

6. Create Dump of DB for later use and sign out of the postgre Linux user using exit command:

   pg_dumpall > backup.sql

7. Copy the dump data to home directory

   sudo cp /var/lib/postgresql/backup.sql ~/.

8. In this scenario we will be connecting database from a container which will have IP address other than our localhost, for this edit PostgreSQL config file to allow connections from remote IP's.

   Open /etc/postgresql/9.5/main/postgresql.conf in a text editor.
   Uncomment the listen_addresses line and set it to '*':

```
listen_addresses = '*'                  # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
```

9. Enable and start the postgresql service:

   sudo systemctl enable postgresql
   sudo systemctl restart postgresql

10. Create Hello World app on Node.js
    Install Node and NPM and navigate to home directory and create a directory app.

    curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
    sudo apt-get install nodejs
    mkdir app && cd app

11. Use vim or nano to create app.js

```
const { Client } = require('pg')

const client = new Client({
  user: 'postgres',
  host: 'localhost',
  database: 'nodejs',
  password: 'yourpassword',
  port: 5432
})

client.connect()

client.query('SELECT * FROM hello', (err, res) => {
  console.log(res.rows[0].message)
client.end()
})
```

12. Install the pg module and test the app, database will display "Hello world " on the console. This app uses NPM module to establish database connectivity, it then later queries hello table which return "Hello world message".

    npm install pg

    node app.js

13. Connect container to Docker Host
    GO to home directory and create a Dockerfile to run Node.js application

```
1 FROM debian
2
3 RUN apt update -y && apt install -y gnupg curl
4 RUN curl -sL https://deb.nodesource.com/setup_8.x | bash - && apt install -y nodejs
5 COPY app/ /home/
6
7 ENTRYPOINT tail -F /dev/null
```

Image from docker file will copy the app/ directory to new image and edit app.js to allow application to connect with database instead of localhost.

14. Build image from Docker file from current directory

    docker build -t Node1_image .

15. Connect container to Database

    Docker sets default bridge network through docker0 network interface, use ifconfig to view interface

```
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:a8:d6:e8:40  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

16. Allow PostgreSQL to connect from Docker interface for that go to /etc/postgresql/10/main/pg_hba.conf

```
host all postgres 172.17.0.1/16 password
```

17. Since docker host contains IP address as 172.17.0.1 all containers of host will have an IP address in range of 172.17.0.0/16

18. Restart the database :

    Sudo Systemctl restart postgresql

19. Now start the container

docker run --add-host=database:172.17.0.1 --name container_node container_image

The add host will allow database host ,which will locate IP address of Docker host.

20. Now, within the container, use ping command to test connection to database host

    docker exec -it container_node ping database

21. Now, Test the container by pinging the address from Docker host

    docker exec -it container_node node home/app.js

    If the configuration is successful, the program will display “Hello World”

    Image Reference: https://medium.com/@simon_lee/how-i-learned-docker-and-deployed-a-node-js-server-200e742259e5