**Docker Compose with Multiple Local Containers**

* **App Overview:**

We are going to create an app, which displays the number of times the page has visited.

Graphical user interface, application

Description automatically generated

We need two components for this app, Node server to show the page, and Redis to save the number of visits.

Diagram

Description automatically generated

There are two approaches to do this.

Have a dedicated redis for each instance of the container. But with this approach the number of visit will show wrong count as it hits the different instance and it is associated with dedicated Redis.

Graphical user interface, application

Description automatically generated

Diagram

Description automatically generated

Second approach is to have a n number of Node server instances and only one redis instance. In this approach we will increase the number of node server instances on need basis, not the redis.

Diagram

Description automatically generated

Our first target is to have a single node server instance and redis.

Graphical user interface

Description automatically generated

* **App Server Starter Code:**

Paste the code written here

* **Assembling Dockerfile:**

FROM node:alpine

WORKDIR '/app'

COPY package.json .

RUN npm install

COPY . .

CMD ["npm", "start"]

* **Introducing docker compose:**

Build and run the above image.

It will start and throw the below exception.

> @ start /app

> node index.js

Listening on port 8081

events.js:291

throw er; // Unhandled 'error' event

^

Error: connect ECONNREFUSED 127.0.0.1:6379

at TCPConnectWrap.afterConnect [as oncomplete] (net.js:1145:16)

Emitted 'error' event on RedisClient instance at:

at RedisClient.on\_error (/app/node\_modules/redis/index.js:406:14)

at Socket.<anonymous> (/app/node\_modules/redis/index.js:279:14)

at Socket.emit (events.js:314:20)

at emitErrorNT (internal/streams/destroy.js:106:8)

at emitErrorCloseNT (internal/streams/destroy.js:74:3)

at processTicksAndRejections (internal/process/task\_queues.js:80:21) {

errno: -111,

code: 'ECONNREFUSED',

syscall: 'connect',

address: '127.0.0.1',

port: 6379

}

npm ERR! code ELIFECYCLE

npm ERR! errno 1

npm ERR! @ start: `node index.js`

npm ERR! Exit status 1

npm ERR!

npm ERR! Failed at the @ start script.

npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:

npm ERR! /root/.npm/\_logs/2020-10-09T04\_37\_55\_190Z-debug.log

localhost:visists spandiyan$ clear

This error is thrown because we don’t have a redis instance.

To get the redis instance using docker, use the below command

docker run redis

Once the redis is started, now start the node server, still we will get the same error, this is because both are running as a different and isolated process.

To make a connection between node and redis there are two ways.

Diagram

Description automatically generated

A picture containing diagram

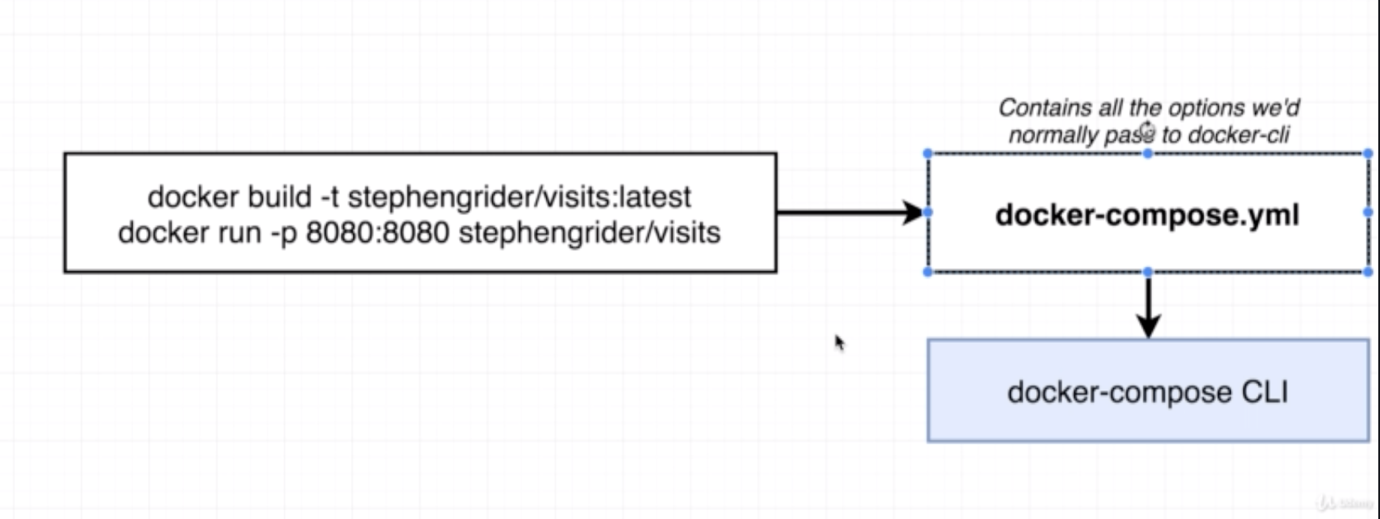
Description automatically generated

Docker compose is a separate cli, which is installed with docker.

Graphical user interface, text, application

Description automatically generated

* **Docker compose file:**



Graphical user interface, text, application

Description automatically generated

Docker compose file:

version: '3'

services:

redis-server:

image: redis

node-app:

build: .

ports:

- "8081:8081"

* **Docker compose commands:**

**Diagram

Description automatically generated**

* **Stopping Docker Compose Containers:**

With normal docker here are the command to start and stop the container

docker run -d redis

docker ps

docker stop id\_retrieved\_from\_ps

Graphical user interface, application

Description automatically generated

* **Container Maintenance with Compose:**
* **Automatic container restarts:**

Table

Description automatically generated

If any running container goes down we can restart the crashed container.

We have following four policy to bring back the crash instace.

Table

Description automatically generated

* **Container status with docker compose:**

With docker cli we use “docker ps” to check the status of the container, we can do the same with docker-compose also “docker-compose ps”

Note: When you run “docker-compose ps” the directory in which you run the command must have the “docker-compose.yml” file, else it will throw an error.