TSR – 19th January 2017. EXERCISE 6

Given these three programs:

```
// Client.js
const zmq = require('zmq');
const req = zmq.socket('req');

var args = process.argv.slice(2);

if (args.length < 3) {
  console.error('Three arguments are needed:');
  console.error('  - IP address of the broker.');
  console.error('  - Pattern to be looked for.');
  console.error('  - String to be processed.');
  process.exit(1);
}

req.connect('tcp://'+args[0]+':8000');

req.send([args[1],args[2]]);

req.on('message', function(m) {
  console.log('The string "%s" has been '+
    'found %d times.', args[1], m+");
  process.exit(0);
});
```

```
// Broker.js
const zmq = require('zmq');
const rou = zmq.socket('router');
const dea = zmq.socket('dealer');

try {
  rou.bindSync('tcp://*:8000');
  dea.bindSync('tcp://*:8001');
} catch (e) {
  if (e) {
    console.error('Error "%s" binding '+
      'the broker sockets.', e);
    console.error('Exiting...');
    process.exit(1);
  }
}

rou.on('message', function() {
  var segs = Array.apply(null, arguments);
  dea.send(segs);
});

dea.on('message', function() {
  var segs = Array.apply(null, arguments);
  rou.send(segs);
});
```

```
// Worker.js
const zmq = require('zmq');
const rep = zmq.socket('rep');

var args = process.argv.slice(2);

if (args.length < 1) {
  console.error('One argument is needed:');
  console.error('  - IP address of the broker.');
  process.exit(1);
}

rep.connect('tcp://'+args[0]+':8001');

rep.on('message', function(pattern,str) {
  var str2 = str+";
  var count = 0;
  for (var i=0; i<str2.length; i++) {
    if (pattern ==
    str2.substr(i,pattern.length))
      count++;
  }
  rep.send(count);
});
```

The worker program implements a service that counts how many times a given short string (or "pattern") is contained in a large string. To this end, the client program sends both strings to the broker. Each worker is able to process each request independently on the others.

Please, answer the following questions related to the deployment of those programs. To this end, let us assume that the host machine has a local Docker image based on "fedora:latest" with the **node** and **npm** commands, the ZeroMQ library, and the **zmq** NodeJS module properly installed. The name of that image is "**zmq-devel**":

1. Propose a hierarchy of folders **and their contents** in order to hold those files and the Dockerfiles being needed for creating three Docker images, one per program (1 point).

   Although there are other alternatives, the solutions suggested in Seminar 4 for this kind of question relied on placing the source files of each component in a separate folder. Therefore, a possible hierarchy could be:
   - Common ancestor folder.
     - "Client" folder:
       - "Client.js" file.
       - "Dockerfile" (for the client component).
     - "Broker" folder:
       - "Broker.js" file.
       - "Dockerfile" (for the broker component).
     - "Worker" folder:
       - "Worker.js" file.
       - "Dockerfile" (for the worker component).

2. Write the Dockerfile for generating the broker image (3 points).

```
FROM zmq-devel
RUN mkdir /d1
COPY ./Broker.js /d1/Broker.js
EXPOSE 8000 8001
WORKDIR /d1
CMD node Broker.js
```

Note that there are many variants of this Dockerfile that may also work correctly. For instance, the WORKDIR command is not necessary if the CMD command specifies the full pathname of the JavaScript program to be run.

The mandatory elements are:

- To include a FROM command in the first line, referring to the "zmq-devel" image.
- To COPY or ADD the "Broker.js" file from the host to any directory in the container.
- To include a EXPOSE command for specifying that the ports to be used are 8000 and 8001.
- To include a CMD or ENTRYPOINT command for running the Broker.js program using the "node" interpreter.

3. Give the appropriate docker command for creating that broker image whose name should be "**broker**" (1 point).

The following command must be run in the "Broker" folder, where the Dockerfile written in question 2 should be placed:
docker build –t broker .

4. Write a command line for starting a broker container and describe how its IP address could be obtained (2 point).

The command to be used is:
docker run broker
and it may be run in any folder, once the "docker build" command requested in the previous question has been run.
While the **broker** container is running, we may use **docker ps** in order to find out the ID or name of that broker container. Once the ID is known (let us assume that its prefix is "b875…"), we may find its IP address using…
docker inspect b875 | grep IPAddress
or simply using the **docker inspect b875** command and looking for any IP address line in its output.

5. Let us assume that the address obtained in the previous question was 172.17.0.2. Now, we need to create the worker image. Write a Dockerfile to this end (2 point).

FROM zmq-devel
RUN mkdir /d1
COPY ./Worker.js /d1/Worker.js
WORKDIR /d1
CMD node Worker.js 172.17.0.2

That is the basic solution for this question. A more realistic one could have used an environment variable for receiving the IP address of the broker. For instance, using this as the last line of the Dockerfile:
CMD node Worker.js $BROKER_IP
In that case, the value for that environment variable could have been passed in the subsequent **docker run** command being needed for running an instance of this component in a container. Something like this:
docker run –e BROKER_IP=172.17.0.2 worker
…assuming the we have already created a **worker** image as requested in question 6.

6. Give the appropriate docker command to create such image that should be called "**worker**" (1 point).

The following command must be run in the "Worker" folder, where the Dockerfile written in question 5 should be placed:
docker build –t worker .