Inter-container Communication in Docker

Introduction : A lot of environment have instances where container are used to provide a single service such as Database as a Mongo container or web server setup in some other container or proxy server in some container and the key here is how one container talk to another container or one container make a request to another container.

Implementation Details: Docker containers are self-contained, isolated environments. However, they are often only useful if they can talk to each other. Here we have created two Ubuntu container and in one container we have install and setup the Nginx web server and in another container we have install Nodejs web server. Nginx web server is configured on port 80 and we have written a simple java script code to configure the Nodejs web server on port 3000.

Steps:

1. **Setup Two Docker Containers on Host**

   Run this command twice to download the public Ubuntu image and stand up two instances of that image in the background.

   Docker run -it –d ubuntu

2. **Verify image is downloaded**

   Docker images

3. **Verify container are up and running**

   Docker ps –a

4. **Setup NGINX in one of the containers**

   Docker exec –it container_id bash

5. **Update the Package Manager**

   Before downloading any new software from the package manager we want to update the package manager to ensure everything is up to date.

   apt-get update

6. **Install NGINX**

   Install NGINX web server software. This will allow us to stand up a demo website on port 80.

   apt-get install nginx

7. **Stand Up nginx web server**

   Service nginx reload

   Service nginx restart

8. **Install Curl**

   So that we can verify our web server is up and running we need a utility to request the origin content on port 80 (localhost: 80). So we need to download the CURL utility.

   apt-get install curl

9. **Verify the web server is running**

This should return the "Welcome to NGINX" html page.

Curl localhost

10. **Log into the other Container**

Docker exec –it container_id bash

11. **Update the Package Manager**

Before downloading any new software from the package manager we want to update the package manager to ensure everything is up to date.

apt-get update

12. **Install Node**

apt-get install nodejs

apt-get install npm

13. **Switch the Node Alias**

Sudo update-alternative –install /usr/bin/node node /usr/bin/nodejs

14. **Verify Node is installed**

node –v

15. **Express**

npm install express –save

16. **Install Curl**

apt-get install curl

17. **Install VIM text editor**

apt-get install vim

18. **Create Node.js web server on port 3000**

vim app.js

19. **Copy the code below as app.js**

```
var express = require('express');
var app = express();
app.get('/', function(req,res){
res.send('Hello World');});
app.listen(3000, function() {
console.log ('Example app listening on port 3000!');
});
```

20. **Save and Exit file**
    :x+ enter

21. **Install Forever for Node.js**
    Npm install forever –g

22. **Stand Up Node server as Daemon**
    Forever start app.js

23. **Verify the Node server is working**
    Curl localhost: 3000

24. **Log out of Container**
    exit + enter

25. **Get IP Address of Container**

    Docker inspect container_id

26. **Verify Node.js Container Service from Host**
    Curl ip_address:3000

27. **Verify NGINX Container Service from Host**
    Curl ip_address:80

28. **Communication Between Docker Containers**

    Log onto Node.js container

       Docker exec –it container_id bash

    Curl NGINX web server

       This should return the "Welcome to NGINX" html page.

       Curl nginx_docker_ip_address