Arbeta med Docker Composer på Raspberry Pi Model 2

Dokumentet beskriver hur du kan arbeta med Docker Composer på Raspberry Pi Model 2. Vi kommer att arbeta med tre stycken containers med enkel webbapplikation i Node.js och en container med `haproxy`.

Den enkla webbapplikationen är variant på den klassiska ”Hello World”.

Arbetsuppgift 1: Skapa katalogstruktur för projektet.

I din hemmakatalog.

Steg 1: Skriv in följande kommando: **`sudo mkdir node-hello`**, klicka därefter på Enter.

Steg 2: Förflytta dig till denna katalog, genom att skriva in kommandot: **`cd node-hello`**, klicka därefter på Enter.

Steg 3: Skapa katalog med namnet src, genom att skriva in kommandot: **`sudo mkdir src`**, klicka därefter på Enter.

Steg 4: Förflytta dig till denna katalog, genom att skriva in kommandot: cd src, klicka därefter på Enter.

Arbetsuppgift 2: Skapa applikation.

Steg 1: Skriv in kommandot: **`sudo nano index.js`**, klicka därefter på Enter.

Steg 2: Skriv in följande kod, i nano:

```
var express = require('express');
var os = require("os");

var app = express();
var hostname = os.hostname();

app.get('/', function (req, res) {
  res.send('<html><body>Hello World from Node.js cont
ainer ' + hostname + '</body></html>');
});

app.listen(80);
console.log('Running on http://localhost');
```

Observera! Raden som börjar med `res.send`, skall skrivas på samma rad, radformateringen här gör att den hamnar på flera rader!

Klicka på ctrl+o för att spara, klicka på Enter för att bekräfta filnamnet. Klicka på ctrl+x för att avsluta nano.

Arbetsuppgift 3: Skapa `src/package.json`, fil som innehåller beroende som behövs för att starta din applikation.

Steg 1: Skriv in kommandot: **`sudo nano package.json`**, klicka därefter på Enter.

Steg 2: Skriv in följande kod, i nano:

```
{
  "name": "node-hello-world",
  "private": true,
  "version": "0.0.1",
  "description": "Node.js Hello world app on docker",
  "author": "linuxkurser.nu",
  "dependencies": {
    "express": "4.12.0"
  }
}
```

Klicka på ctrl+o för att spara, klicka på Enter för att bekräfta filnamnet. Klicka på ctrl+x för att avsluta nano.

Arbetsuppgift 4: Skapa Dockerfile, som skall användas för att skapa Docker container.

Förflytta dig upp ett steg i din katalogstruktur.

Steg 1: Skriv in kommandot: **`cd ..`**, klicka därefter på Enter.

Skapa Dockerfile.

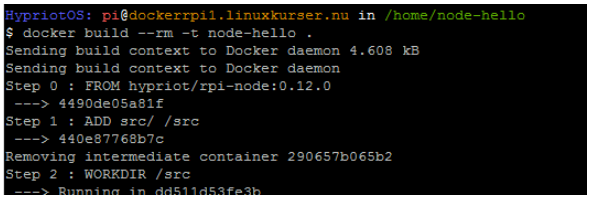
Steg 2: Skriv in kommandot: **`sudo nano Dockerfile`**, klicka därefter på Enter.

Steg 3: Skriv in följande i nano:

```
# Uses Hypriot Raspberry Pi container with Node.js
FROM hypriot/rpi-node:0.12.0

# make the src folder available in the docker image
ADD src/ /src
WORKDIR /src

# install the dependencies from the package.json file
RUN npm install

# make port 80 available outside of the image
EXPOSE 80

# start node with the index.js file of our hello-world application
CMD ["node", "index.js"]
```

Observera! Raden som börjar med `# install`, skall skrivas på samma rad, radformateringen här gör att den hamnar på flera rader!

Klicka på ctrl+o för att spara, klicka på Enter för att bekräfta filnamnet. Klicka på ctrl+x för att avsluta nano.



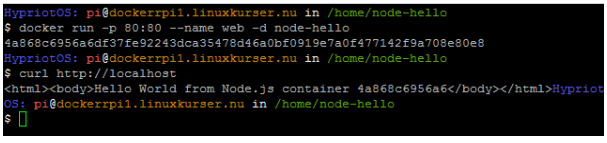
Arbetsuppgift 5: Skapa Docker container.

Steg 1: Skriv in kommandot: **`docker build -t node-hello .`**, klicka därefter på Enter.

Arbetsuppgift 6: Kör din container och test din applikation.

Steg 1: Skriv in kommandot: **`docker run -p 80:80 --name web -d node-hello`**, klicka därefter på Enter.

Steg 2: Skriv in kommandot: **`curl http://localhost`**, klicka därefter på Enter.



Din applikation skall svara med:
`<html><body>Hello World`

```
from Node.js container xxxxxxxxxxxx </body></html>
```

Eller i browser:

Steg 3: Stoppa containern, genom att skriva in kommandot: **`docker stop web`**, klicka därefter på Enter.

Köra flera container med hjälp av Docker Composer.

Vi kommer att använda Docker Composer för att skapa en farm av webb servrar som finns bakom HAProxy lastbalanserare.

Arbetsuppgift 1: Installera Docker Composer på Rasberry Pi Model 2.

Teamet Hypriot har byggt binary för att köras på Rasberry Pi.

Steg 1: Skriv in följande kommando: **`sudo sh -c "curl -L https://github.com/hypriot/compose/releases/download/1.1.0-raspbian/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose; chmod +x /usr/local/bin/docker-compose"`**

```
HypriotOS: pi@dockerrpi1.linuxkurser.nu in ~
$ sudo sh -c "curl -L https://github.com/hypriot/compose/releases/download/1.1.0
-raspbian/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose;
chmod +x /usr/local/bin/docker-compose"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   401    0   401    0     0    528      0 --:--:-- --:--:-- --:--:--   768
100 4711k  100 4711k    0     0   835k      0  0:00:05  0:00:05 --:--:--  998k
HypriotOS: pi@dockerrpi1.linuxkurser.nu in ~
$
```

Arbetsuppgift 2: Skapa `docker-compose.yml`.

Denna fil används för att sätta upp tre containers, webb1, webb2 och webb3, dessa är baserade på applikationsimage som vi skapade tidigare, samt en

container med haproxy. För haproxy används Hypriot:s images `hypriot/rpi-haproxy`.

Steg 1: Skriv in kommandot: **`sudo nano docker-compose.yml`**, klicka därefter på Enter.

Skriv in följande:

```
webb1:
      build: .
      expose:
          - 80
webb2:
      build: .
      expose:
          - 80
webb1:
      build: .
      expose:
          - 80
webb3:
      build: .
      expose:
          - 80
haproxy:
      image: hypriot/rpi-haproxy
      volumes:
         - haproxy:/haproxy-override
```

```
    links:
      - webb1
      - webb2
      - webb3
    ports:
       - "80:80"
       - "70:70"
     expose:
       - "80"
       - "70"
```

Klicka på ctrl+o för att spara, klicka på Enter för att bekräfta filnamnet. Klicka på ctrl+x för att avsluta nano.

Arbetsuppgift 3: Skapa konfigurationsfil för haproxy.

Skapa katalog med namnet haproxy.

Steg 1: Skriv in kommandot: sudo mkdir haproxy, klicka därefter på Enter.

Steg 2: Skriv in kommandot: **`sudo nano haproxy/haproxy.cfg`**, klicka därefter på Enter.

Skriv in följande:

```
global
    log 127.0.0.1 local0
    log 127.0.0.1 local1 notice
defaults
    log global
    mode http
    option httplog
```


```
    option dontlognull
    timeout connect 5000
    timeout client 10000
    timeout server 10000
listen stats :70
    stats enable
    stats uri /
frontend balancer
    bind 0.0.0.0:80
    mode http
    default_backend aj_backends
backend aj_backends
    mode http
    option forwardfor
    # http-request set-header X-Forwarded-Port
&[dst_port]
    balance roundrobin
    server webb1:80 check
    server webb2:80 check
    server webb3:80 check
    # option httpchk OPTIONS * http/1.1\r\nHost:\
localhost
    option httpchk GET /
    http-check expect status 200
```

Arbetsuppgift 4: Skapa containers med Docker-compose.

Steg 1: Skriv in följande kommando: **`docker-compose -d`**, klicka därefter på Enter.

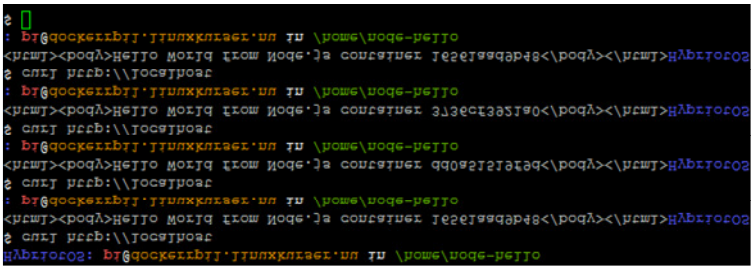
```
HypriotOS: pi@dockerrpi1.linuxkurser.nu in /home/node-hello
$ docker-compose up -d
Creating nodehello_webb1_1...
Creating nodehello_webb3_1...
Creating nodehello_webb2_1...
Creating nodehello_haproxy_1...
HypriotOS: pi@dockerrpi1.linuxkurser.nu in /home/node-hello
$
```

Kontrollera att dina containers är uppstartade.

Steg 2: Skriv in följande kommando: **`docker-compose ps`**, klicka därefter på Enter.

```
HypriotOS: pi@dockerrpi1.linuxkurser.nu in /home/node-hello
$ docker-compose ps
      Name                Command              State                Ports
-------------------------------------------------------------------------------
nodehello_haproxy   bash /haproxy-      Up                  443/tcp, 0.0.0.0:
_1                  start                                   70->70/tcp, 0.0.0
                                                            .0:80->80/tcp
nodehello_webb1_1   node index.js       Up                  80/tcp
nodehello_webb2_1   node index.js       Up                  80/tcp
nodehello_webb3_1   node index.js       Up                  80/tcp
HypriotOS: pi@dockerrpi1.linuxkurser.nu in /home/node-hello
$
```

Arbetsuppgift 5: Testa din webbapplikation.



Steg 1: Skriv in kommandot: **`curl http://localhost`**, klicka därefter på Enter. Du skall få svar från en av dina containers.

Steg 2: Upprepa kommandot tre gånger, sista gången kommer du att få svar från den första containern. `haproxy` arbetar med enkel lastbalansering, baserad på roundrobin.

Arbetsuppgift 6: Stoppa dina containers.

Steg 1: Skriv in kommandot: **`docker-compose stop`**, klicka därefter på Enter.

Dina ingående containers kommer att stoppas.