Persistent

Docker Example :
Multi-container application stack

A multi-container application stack

- In our sample application, we're going to build a series of images that will allow a Node container to serve our Node application, which will be linked to:
  - A Redis primary container to hold and cluster our state
  - Two Redis replica containers to cluster our state.
  - A logging container to capture our application logs.

- The reference for this multi container application stack is from the book
  - The Docker Book by James Turnball.

Node.js image

- Let's start with an image that installs Node.js, our Express application, and the associated prerequisites.
- Run the below commands.
  - $ mkdir nodejs
  - $ cd nodejs
  - $ mkdir -p nodeapp
  - $ vi Dockerfile

- The node.js application code is available at https://dockerbook.com/code/6/node/nodejs/
- Copy the server dot js file, and the package dot json file, from the above link.

Dockerfile file for Node.js application

```
FROM ubuntu:14.04

MAINTAINER James Turnbull <james@example.com>

ENV REFRESHED_AT 2014-06-01

RUN apt-get -yqq update

RUN apt-get -yqq install nodejs npm

RUN ln -s /usr/bin/nodejs /usr/bin/node

RUN mkdir -p /var/log/nodeapp

ADD nodeapp /opt/nodeapp/

WORKDIR /opt/nodeapp

RUN npm install

VOLUME [ "/var/log/nodeapp" ]

EXPOSE 3000

ENTRYPOINT [ "nodejs", "server.js"
```

Node.js application code – sever.js

- The code for server.js file is available at below location https://dockerbook.com/code/6/node/nodejs/nodeapp/
- The server.js file pulls in all the dependencies and starts an Express application
- The Express app is configured to store its session information in Redis and exposes a single endpoint that returns a status message as JSON.
- We've configured its connection to Redis to use a host called redis_primary with an option to override this with an environment variable if needed.
- Let’s build our image now
  - sudo docker build -t jamtur01/nodejs .

Redis base image

- Let's continue with our first Redis image: a base image that will install Redis.
- On top of this base image that we'll build our Redis primary and replica images.
- Run the below commands
  - $ mkdir redis_base
  - $ cd redis_base
  - $ vi Dockerfile

Dockerfile for Redis base image

- FROM ubuntu:14.04
- MAINTAINER James Turnbull <james@example.com>
- ENV REFRESHED_AT 2014-06-01
- RUN apt-get -yqq update
- RUN apt-get install -yqq software-properties-common python-software-properties
- RUN add-apt-repository ppa:chris-lea/redis-server
- RUN apt-get -yqq update
- RUN apt-get -yqq install redis-server redis-tools
- VOLUME ["/var/lib/redis", "/var/log/redis/"]
- EXPOSE 6379
- Build the primary Dokcer image for Redis –
  - sudo docker build -t jamtur01/redis

Dockerfile for Redis primary image

- Let's continue with our first Redis image: a Redis primary server.
- Let’s run below commands
  - $ mkdir redis_primary
  - $ cd redis_primary
  - $ vi Dockerfile

Dockerfile for Redis primary image…

- The contents of Dockerfile are as below

```
FROM jamtur01/redis

MAINTAINER James Turnbull <james@example.com>

ENV REFRESHED_AT 2014-06-01

ENTRYPOINT [ "redis-server", "--logfile /var/log/redis/redis-

server.log" ]
```

- Let's build our Redis primary image now.
  - $ sudo docker build -t jamtur01/redis_primary.

Redis replica image

- As a complement to our Redis primary image, we're going to create an image that runs a Redis replica to allow us to provide some redundancy to our Node.js
- Run the below commands
  - $ mkdir redis_replica
  - $ cd redis_replica
  - $ touch Dockerfile

Dockerfile for Redis replica image

- The contents of Redis replica image Dockerfile are as below

```
FROM jamtur01/redis

MAINTAINER James Turnbull <james@example.com>

ENV REFRESHED_AT 2014-06-01

ENTRYPOINT [ "redis-server", "--logfile /var/log/redis/redis-

replica.log", "--slaveof redis_primary 6379" ]
```

- Let's build our Redis replica image now.
  - sudo docker build -t jamtur01/redis_replica.

Creating our Redis back-end cluster

- Let's start by building the Redis primary container.
- Run the below command
  - sudo docker run -d -h redis-primary --name redis_primary amtur01/redis_primary

- Run the below command
  - sudo docker run -ti --rm --volumes-from redis_primary Ubuntu cat/var/log/redis/redis-server.log

Redis back-end cluster log files

```
ubuntu@ip-172-31-36-70:~$ sudo docker run -ti --rm --volumes-from redis_primary ubuntu cat /var/log/redis/redis-server.log
                _.-
            .-``  ''-.
        .-``  __  `.  ''-.              Redis 3.0.7 (00000000/0) 64 bit
    .-``  .-```.  ```\/    _.,_ ''-._
   (    '      ,       .-`  | `,    )     Running in standalone mode
   |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
   |    `-._   `._    /     _.-'    |     PID: 1
    `-._    `-._  `-./  _.-'    _.-'
   |`-._`-._    `-.__.-'    _.-'_.-'|
   |    `-._`-._        _.-'_.-'    |           http://redis.io
    `-._    `-._`-.__.-'_.-'    _.-'
   |`-._`-._    `-.__.-'    _.-'_.-'|
   |    `-._`-._        _.-'_.-'    |
    `-._    `-._`-.__.-'_.-'    _.-'
        `-._    `-.__.-'    _.-'
            `-._        _.-'
                `-.__.-'

1:M 26 Jun 06:00:02.122 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
1:M 26 Jun 06:00:02.122 # Server started, Redis version 3.0.7
1:M 26 Jun 06:00:02.122 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' t
o /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 26 Jun 06:00:02.122 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To
fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after
 a reboot. Redis must be restarted after THP is disabled.
1:M 26 Jun 06:00:02.122 * The server is now ready to accept connections on port 6379
ubuntu@ip-172-31-36-70:~$
```

Running Redis Replica container

- Let's create our first Redis replica.
  - sudo docker run -d -h redis-replica1 --name redis_replica1 --link redis_primary:redis_primary jamtur01/redis_replica

- Let’s check new container’s logs
  - sudo docker run -ti --rm --volumes-from redis_replica1 ubuntu cat /var/log/redis/redis-replica.log

Running one more Redis Replica container

- Let's create our second Redis replica container.
- Run the below command
  - sudo docker run -d -h redis-replica2 --name redis_replica2 --link redis_primary:redis_primary jamtur01/redis_replica

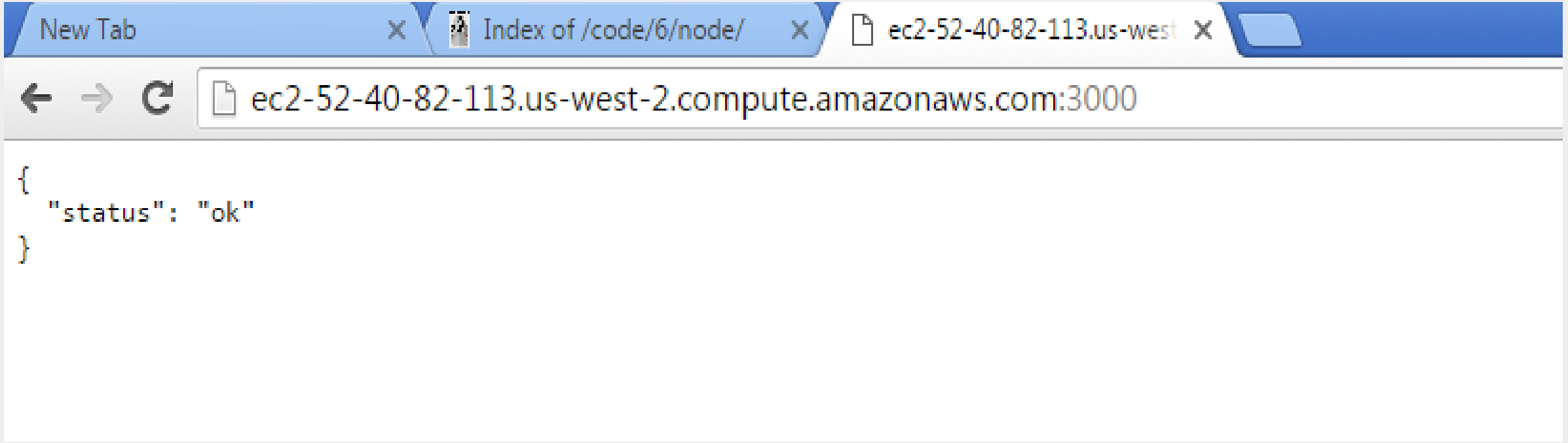
- Let’s check new container’s logs
  - sudo docker run -ti --rm --volumes-from redis_replica2 ubuntu cat /var/log/redis/redis-replica.log

Creating our Node container

- Now that we've got our Redis cluster running, we can launch a container for our Node.js application.
- Run the below command
  - sudo docker run -d –name nodeapp –p 3000:3000 --link redis_primary:redis_primary jamtur01/nodejs

- Let’s check new container’s logs
  - sudo docker logs nodeapp

Running Node container



Capturing our application logs

- We will capture the log output and put it into our logging servers.
- We are going to use Logstash to do so.
- Let’s start creating an image that installs Logstash.
  - $ mkdir logstash
  - $ cd logstash
  - $ touch Dockerfile

Dockerfile for Logstash

```
FROM ubuntu:14.04

MAINTAINER James Turnbull <james@example.com>

ENV REFRESHED_AT 2014-06-01

RUN apt-get -yqq update

RUN apt-get -yqq install wget

RUN wget -O - http://packages.elasticsearch.org/GPG-KEY-↩

elasticsearch | apt-key add -

RUN echo 'deb http://packages.elasticsearch.org/logstash/1.4/↩

debian stable main' > /etc/apt/sources.list.d/logstash.list

RUN apt-get -yqq update

RUN apt-get -yqq install logstash

ADD logstash.conf /etc/

WORKDIR /opt/logstash

ENTRYPOINT [ "bin/logstash" ]

CMD [ "--config=/etc/logstash.conf" ]
```

Contents of Logstash config file

```
input {
file {
type => "syslog"
path => ["/var/log/nodeapp/nodeapp.log",
"/var/log/redis/
redis-server.log"]
}
}
output {
stdout {
codec => rubydebug
}
```

Starting Logstash container

- Run the below command to start the Logstash container.
  - sudo docker run -d --name logstash --volumes-from redis_primary --volumes-from nodeapp jamtur01/logstash

- Let’s check new container’s logs
  - sudo docker logs -f logstash

```
{
      "message" => "123.201.208.142 - - [Sun, 26 Jun 2016 07:54:18 GMT] \"GET / HTTP/1.1\" 200 20 \"-\" \"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTM
, like Gecko) Chrome/51.0.2704.103 Safari/537.36\"",
     "@version" => "1",
   "@timestamp" => "2016-06-26T07:54:18.993Z",
         "type" => "syslog",
         "host" => "4d155885ccd3",
         "path" => "/var/log/nodeapp/nodeapp.log"
}
```

Reference Material : Websites & Blogs

- https://www.docker.com/
- https://training.docker.com/self-paced-training
- https://www.youtube.com/watch?v=Q5POuMHxW-0
- https://www.digitalocean.com/community/tutorials/how-to-run-nginx-in-a-docker-container-on-ubuntu-14-04

Docker up and Running by Karl Matthias and Sean kane

The Docker Book by James Turnball

Docker Interactive

Dattatray Kulkarni

dattatray_kulkarni@persistent.co.in

Asif Immanad

asif_immanad@persistent.co.in

Vice President

Shubhangi Kelkar

shubhangi_kelkar@persistent.co.in

Persistent

Thank you!

Persistent University