

## Deploy the Kong API Gateway

**Once successfully deployed, include the updated list of Docker Run Commands (or Docker Compose Yaml File) in the PDF.**

### Kong API set up

```
docker run -d --name kong-database \  
    -p 9042:9042 \  
    cassandra:2.2
```

```
-----  
docker run --rm \  
    --link kong-database:kong-database \  
    -e "KONG_DATABASE=cassandra" \  
    -e "KONG_PG_HOST=kong-database" \  
    -e "KONG_CASSANDRA_CONTACT_POINTS=kong-database" \  
    kong:0.9.9 kong migrations up
```

```
-----  
docker run -d --name kong \  
    --link kong-database:kong-database \  
    --link gumball:gumball \  
    -e "KONG_DATABASE=cassandra" \  
    -e "KONG_PG_HOST=kong-database" \  
    -e "KONG_CASSANDRA_CONTACT_POINTS=kong-database" \  
    -e "KONG_PROXY_ACCESS_LOG=/dev/stdout" \  
    -e "KONG_ADMIN_ACCESS_LOG=/dev/stdout" \  
    -e "KONG_PROXY_ERROR_LOG=/dev/stderr" \  
    -e "KONG_ADMIN_ERROR_LOG=/dev/stderr" \  
    -e "KONG_ADMIN_LISTEN=0.0.0.0:8001" \  
    -e "KONG_ADMIN_LISTEN_SSL=0.0.0.0:8444" \  
    -p 8000:8000 \  
    -p 8443:8443 \  
    -p 8001:8001 \  
    -p 8444:8444 \  
    kong:0.9.9
```

Docker build -t nodegumball .

```
docker run --name nodejs --link kong:kong -p 4001:4000 -d nodegumball
```

Also run the following two Docker Commands and take a Screenshot of their output to be added to PDF

docker-ps:

```
docker ps --all --format "table {{.ID}}\t{{.Names}}\t{{.Image}}\t{{.Status}}\t"
```

docker-ps-ports:

```
docker ps --all --format "table {{.Names}}\t{{.Ports}}\t"
```

```
mandipsinhgohil (master *) go
$ docker ps --all --format "table {{.ID}}\t{{.Names}}\t{{.Image}}\t{{.Status}}\t"
CONTAINER ID        NAMES               IMAGE                STATUS
0f7856ffcbecc       nodejs              nodegumball          Up 31 minutes
56a45b84fd49         kong                kong:0.9.9           Up 44 minutes
47b3bdd2c542         kong-database       cassandra:2.2        Up 44 minutes
3971d1f5866d         gumball             gumball              Up 3 hours
eb1e6aa96c1b         mongoddb            mongo:latest         Up 3 hours
60ecaabcf5f          rabbitmq            rabbitmq:3-management Up 3 hours
db195c6c261b         node                a6c287303bf8        Exited (1) 5 hours ago
3bfebd3af5a         mandiptud           a6c287303bf8        Exited (1) 5 hours ago
36351f0f925c         tender_chebyshev    327044743968        Exited (1) 9 hours ago
13c8aa96da9d         gracious_hawking    07ab453c463c        Exited (1) 9 hours ago
5d57de12a405         mysql               mysql:5.5             Exited (0) 31 hours ago
a8fc3865d7dd         starbucks           starbucks             Exited (137) 31 hours ago
9998aaba61a2         angry_bohr          nodejs                Exited (255) 3 days ago
de6927424960         gumballredis        mandipsinhgohil/gumball-redis Exited (255) 4 days ago
14cf3c3f12ea         gumball-redis       9235ef35e722        Exited (1) 4 days ago
91fef8791761         festive_stonebraker b34b9d60f1d8        Exited (1) 4 days ago
4126cb879cd7         redis               redis:4.0             Exited (255) 4 days ago
dee25bf22b69         optimistic_lewin    927cdedbd087        Exited (1) 2 weeks ago
ea4ea5fa9467         agitated_albattani  536d11fb8672        Exited (1) 2 weeks ago
950c666a3d3a         frosty_khorana      b911fd3838b8        Exited (255) 3 weeks ago
mandipsinhgohil (master *) go
$ █
```

```
mandipsinhgohil (master *) go
$ docker ps --all --format "table {{.Names}}\t{{.Ports}}\t"
NAMES                PORTS
nodejs               8080/tcp, 0.0.0.0:4001->4000/tcp
kong                 0.0.0.0:8000-8001->8000-8001/tcp, 7946/tcp, 0.0.0.0:8443-8444->8443-8444/tcp
kong-database        7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp
gumball              0.0.0.0:4000->3000/tcp
mongoddb             27017/tcp
rabbitmq             4369/tcp, 5671-5672/tcp, 15671/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp
node
mandiptud
tender_chebyshev
gracious_hawking
mysql
starbucks
angry_bohr           0.0.0.0:8080->8080/tcp
gumballredis         0.0.0.0:8080->3000/tcp
gumball-redis
festive_stonebraker
redis                6379/tcp
optimistic_lewin
agitated_albattani
frosty_khorana       9090/tcp
mandipsinhgohil (master *) go
$ █
```

**Configure Kong API Gateway to Go API as upstream server**  
**Add API Key Authentication Plugin to Kong to protect the Go API.**  
**Include all the REST API calls to Kong for your configuration in PDF**  
**(Or Screenshots evidence of Kong Configuration if using a Kong UI Admin Tool)**

GET Status

GET List API

POST Add API

GET List API

POST Plugin Add API key Auth

POST API client Add

POST API client Add Key

The screenshot displays a REST client interface with a GET request to `http://localhost:8001/status`. The response is a JSON object containing server and database statistics. The status bar indicates a 200 status code and a 66 ms response time.

**Request Headers:**

- cache-control: "no-cache"
- postman-token: "c12b803-1ea6-40b4-aa2b-29900d446381"
- user-agent: "PostmanRuntime/7.1.1"
- accept: "\*//\*"
- host: "localhost:8001"
- accept-encoding: "gzip, deflate"

**Response Headers:**

- date: "Sun, 01 Apr 2018 03:39:05 GMT"
- content-type: "application/json; charset=utf-8"
- transfer-encoding: "chunked"
- connection: "keep-alive"
- access-control-allow-origin: ""
- server: "kong/0.9.0"

**Response Body:**

```
{
  "server": {
    "connections_handled": 415,
    "connections_reading": 0,
    "connections_active": 1,
    "total_requests": 468,
    "connections_accepted": 415,
    "connections_writing": 1,
    "connections_waiting": 0
  },
  "database": {
    "oauth2_credentials": 0,
    "jwt_secrets": 0,
    "response_ratelimiting_metrics": 0,
    "keyauth_credentials": 0,
    "oauth2_authorization_codes": 0,
    "acls": 0,
    "apis": 1,
    "basicauth_credentials": 0,
    "consumers": 0,
    "ratelimiting_metrics": 0,
    "oauth2_tokens": 0,
    "nodes": 1,
    "hmacauth_credentials": 0,
    "plugins": 0
  }
}
```

GET http://localhost:8001/apis

08:39:06.579

PrettyRaw

200

25 ms

Request Headers:

cache-control: "no-cache"  
postman-token: "7b753a44-4f5c-4d50-9d50-20441f805717"  
user-agent: "PostmanRuntime/7.1.1"  
accept: "\*/"\*  
host: "localhost:8001"  
accept-encoding: "gzip, deflate"

Response Headers:

date: "Sun, 01 Apr 2018 03:39:08 GMT"  
content-type: "application/json; charset=utf-8"  
transfer-encoding: "chunked"  
connection: "keep-alive"  
access-control-allow-origin: ""  
server: "kong/0.9.9"

Response Body:

data:

0:

upstream\_url: "http://gumball:3000/"  
strip\_request\_path: true  
request\_path: "/goapi"  
id: "e3e7ff7b1-2644-4ae5-b134-bdece1d94985"  
created\_at: 1522552935000  
preserve\_host: true  
name: "goapi"

total: 1

POST http://localhost:8001/apis

08:39:09.047

PrettyRaw

409

17 ms

Request Headers:

content-type: "application/json"  
cache-control: "no-cache"  
postman-token: "4de9b8d1-f4cb-47a3-b7d4-eb8331a56f77"  
user-agent: "PostmanRuntime/7.1.1"  
accept: "\*/"\*  
host: "localhost:8001"  
accept-encoding: "gzip, deflate"  
content-length: 156

Request Body:

name: "goapi"  
request\_path: "/goapi"  
strip\_request\_path: true  
preserve\_host: true  
upstream\_url: "http://gumball:3000/"

Response Headers:

date: "Sun, 01 Apr 2018 03:39:10 GMT"  
content-type: "application/json; charset=utf-8"  
transfer-encoding: "chunked"  
connection: "keep-alive"  
access-control-allow-origin: ""  
server: "kong/0.9.9"

Response Body:

name: "already exists with value 'goapi'"  
request\_path: "already exists with value '/goapi'"

GET http://localhost:8001/apis

08:39:12.645

PrettyRaw

200

42 ms

Request Headers:

cache-control: "no-cache"  
postman-token: "6ffd9415-d1ac-4c19-8755-706b3ff5bb4a"  
user-agent: "PostmanRuntime/7.1.1"  
accept: "\*/"\*  
host: "localhost:8001"  
accept-encoding: "gzip, deflate"

Response Headers:

date: "Sun, 01 Apr 2018 03:39:14 GMT"  
content-type: "application/json; charset=utf-8"  
transfer-encoding: "chunked"  
connection: "keep-alive"  
access-control-allow-origin: ""  
server: "kong/0.9.9"

Response Body:

data:

0:

upstream\_url: "http://gumball:3000/"  
strip\_request\_path: true  
request\_path: "/goapi"  
id: "e3e7ff7b1-2644-4ae5-b134-bdece1d94985"  
created\_at: 1522552935000  
preserve\_host: true  
name: "goapi"

total: 1

▼ POST http://localhost:8001/apis/goapi/plugins

88:39:16.999

201

69 ms

PrettyRaw

▼ Request Headers:  
cache-control: "no-cache"  
postman-token: "b6237db0-97e1-4560-ac90-2712b0e5fa6f"  
user-agent: "PostmanRuntime/7.1.1"  
accept: "\*/\*"  
host: "localhost:8001"  
accept-encoding: "gzip, deflate"  
content-type: "multipart/form-data; boundary=-----511207490240114785643603"  
content-length: 167

▼ Request Body:  
name: "key-auth"

▼ Response Headers:  
date: "Sun, 01 Apr 2018 03:39:18 GMT"  
content-type: "application/json; charset=utf-8"  
transfer-encoding: "chunked"  
connection: "keep-alive"  
access-control-allow-origin: ""  
server: "kong/0.9.9"

▼ Response Body:  
api\_id: "e3e7f7b1-2644-4ae5-b134-bdece1d94985"  
id: "5fcef6e9c-962b-4903-8ccb-b10bb38a895d"  
created\_at: 1522553958000  
enabled: true  
name: "key-auth"  
▼ config:  
  ▼ key\_names:  
    0: "apikey"  
  hide\_credentials: false

▼ POST http://localhost:8001/consumers/

88:39:23.548

201

33 ms

PrettyRaw

▼ Request Headers:  
cache-control: "no-cache"  
postman-token: "c12ff9c2-2f20-4112-a003-151cc61d3568"  
user-agent: "PostmanRuntime/7.1.1"  
accept: "\*/\*"  
host: "localhost:8001"  
accept-encoding: "gzip, deflate"  
content-type: "multipart/form-data; boundary=-----690660548411064311844901"  
content-length: 172

▼ Request Body:  
username: "apiclient"

▼ Response Headers:  
date: "Sun, 01 Apr 2018 03:39:25 GMT"  
content-type: "application/json; charset=utf-8"  
transfer-encoding: "chunked"  
connection: "keep-alive"  
access-control-allow-origin: ""  
server: "kong/0.9.9"

▼ Response Body:  
username: "apiclient"  
created\_at: 1522553965000  
id: "68a79935-5332-4d08-93f5-4215e6c7849b"

▼ POST http://localhost:8001/consumers/apiclient/key-auth

88:39:27.821

201

37 ms

PrettyRaw

▼ Request Headers:  
content-type: "text/plain"  
cache-control: "no-cache"  
postman-token: "07a2f9c2-f1eb-47a0-9bbb-3775c115611c"  
user-agent: "PostmanRuntime/7.1.1"  
accept: "\*/\*"  
host: "localhost:8001"  
accept-encoding: "gzip, deflate"  
content-length: ""

▼ Response Headers:  
date: "Sun, 01 Apr 2018 03:39:29 GMT"  
content-type: "application/json; charset=utf-8"  
transfer-encoding: "chunked"  
connection: "keep-alive"  
access-control-allow-origin: ""  
server: "kong/0.9.9"

▼ Response Body:  
key: "8cf694fe37d143c7b25048955e1b2f49"  
consumer\_id: "68a79935-5332-4d08-93f5-4215e6c7849b"  
created\_at: 1522553969000  
id: "d49cc709-6f00-44b5-86fb-c32da4f0ad09"

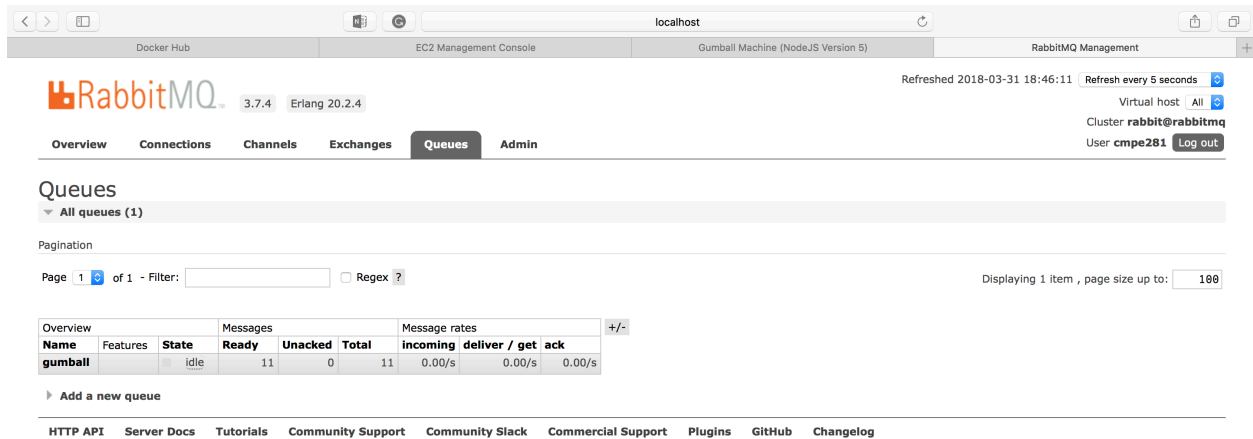
## Include snippets of Node.js changes in the PDF document

Project	app.js	server.go	package.json
nodejs	11		
images	12 NodeJS-Enabled Standing Gumball		
node_modules	13 Model# M102988		
views	14 Serial# 1234998871109		
app.js	15		
Dockerfile	16 **/		
package-lock.json	17		
package.json	18 //var machine = "http://localhost:3000/gumball";		
	19 //var endpoint = "http://localhost:3000/order";		
	20		
	21 //var machine = "http://gumball:3000/gumball";		
	22 //var endpoint = "http://gumball:3000/order";		
	23		
	24 var machine = "http://kong:8000/goapi/gumball";		
	25 var endpoint = "http://kong:8000/goapi/order";		
	26		
	27 // added in v3: handlebars		
	28 // https://www.npmjs.org/package/express3-handlebars		
	29 // npm install express3-handlebars		
	30		
	31 // added in v2: crypto		
	32 // crypto functions: http://nodejs.org/api/crypto.html		
	33		
	34		

Take a screenshot of the Node.js App after placing a few Gumball Orders (note, you'll have to insert a quarter for each order)



Take a screenshot of the RabbitMQ "gumball" queue showing the count of messages (1 for each Order)



The screenshot shows the RabbitMQ Management interface in a web browser. The top navigation bar includes links for Overview, Connections, Channels, Exchanges, Queues (selected), and Admin. The main content area is titled 'Queues' and shows a list of queues. The 'gumball' queue is selected, and its details are displayed in a table. The table shows that the queue has 1 message (Ready: 1, Unacked: 0, Total: 1). The message rates are also shown: incoming 0.00/s, deliver 0.00/s, and get ack 0.00/s.

Overview			Messages			Message rates			
Name	Features	State	Ready	Unacked	Total	incoming	deliver	get ack	
gumball		idle	11	0	11	0.00/s	0.00/s	0.00/s	+/-

Take a screenshot of the output of the command.

```
curl -X POST \
http://dockerhost:8000/goapi/order \
-H 'apikey: 8cf694fe37d143c7b25648955e1b2f49' \
-H 'content-type: application/json'

mandipsinhgohil (master *) go
$ curl -X POST \
> http://localhost:8000/goapi/order \
> -H 'apikey: 8cf694fe37d143c7b25648955e1b2f49' \
[> -H 'content-type: application/json'
{
  "Id": "7fa14e27-d858-44c6-946a-fe33221652c6",
  "OrderStatus": "Order Placed"
}
```