Expose Docker Container services on the Internet using the ngrok docker image

January 6, 2019

The challenge: you are running a service, API or web application in a Docker container, locally on your laptop or in a cloud based VM or container platform. You would like to provide access to external consumers – yourself on your smart phone, a piece of code running in a cloud environment, a colleague on your local network or on the other side of the world. The question is: how to get requests from these external parties to the application that does not and cannot [easily] expose and external endpoint.

In this article, we will look at how ngrok – a tool and a cloud service – makes this happen. It generates a public URL and ensures that all requests sent to that URL are forwarded to a local agent (running in its own, stand alone Docker container) that can then pass them on to the local service. See https://technology.amis.nl/2016/12/07/publicly-exposing-a-local-service-to-nearby-and-far-away-consumer-on-the-internet-using-ngrok/ for an introduction to ngrok.

Tutorial 1: Expose a local docker container on the Internet with `ngrok`-generated public URL

Assuming you are working on a Docker host – a system that can run Docker containers, here are some steps to try out:

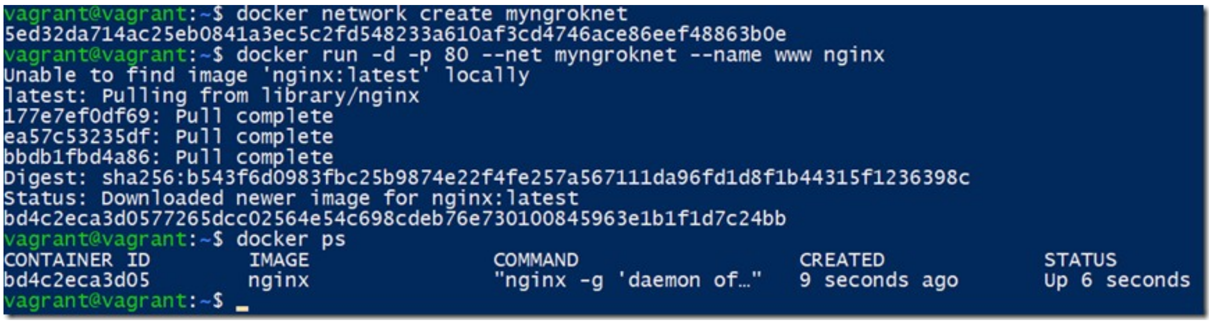
First steps with ngrok and Docker

Define a logical network `myngroknet` to link two or more containers together:

```
docker network create myngroknet
```

Run a Docker Container called `www` based on the `nginx` image and associate it with the `mynrgoknet` network:

```
docker run -d -p 80 --net myngroknet --name www nginx
```



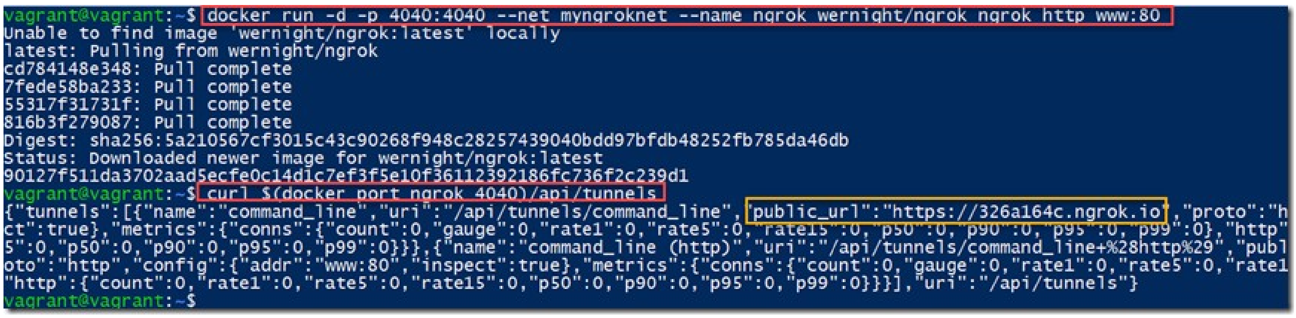
```
vagrant@vagrant:~$ docker network create myngroknet
5ed32da714ac25eb0841a3ec5c2fd548233a610af3cd4746ace86eef48863b0e
vagrant@vagrant:~$ docker run -d -p 80 --net myngroknet --name www nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
177e7ef0df69: Pull complete
ea57c53235df: Pull complete
bbdb1fbd4a86: Pull complete
Digest: sha256:b543f6d0983fbc25b9874e22f4fe257a567111da96fd1d8f1b44315f1236398c
Status: Downloaded newer image for nginx:latest
bd4c2eca3d0577265dcc02564e54c698cdeb76e730100845963e1b1f1d7c24bb
vagrant@vagrant:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
bd4c2eca3d05        nginx               "nginx -g 'daemon of…"   9 seconds ago       Up 6 seconds
vagrant@vagrant:~$ _
```

Run a container called `ngrok` based on the `ngrok container image.` Associate the container with the `myngroknet` network; this enables the container to access container `www` using its container name as hostname (for example `http://www`). Expose port 4040 – where the ngrok inspection interface is accessed. Specify that `ngrok` should open a tunnel (expose a public url) for HTTP requests to port 80 on container `www`:

```
docker run -d -p 4040:4040 --net myngroknet --name ngrok wernight/
ngrok ngrok http www:80
```

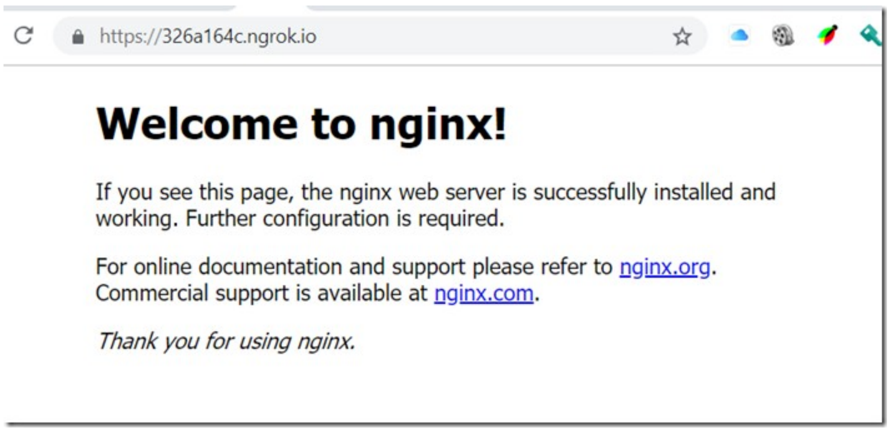
Access `ngrok` monitor: access port 4040 on the Docker host.

Either from the `ngrok` monitor of from the command line in the Docker host using `curl $(docker port ngrok 4040)/api/tunnels` get the public url that has been assigned to the ngrok session.



```
vagrant@vagrant:~$ docker run -d -p 4040:4040 --net myngroknet --name ngrok wernight/ngrok ngrok http www:80
Unable to find image 'wernight/ngrok:latest' locally
latest: Pulling from wernight/ngrok
cd784148e348: Pull complete
7fede58ba233: Pull complete
55317f31731f: Pull complete
816b3f279087: Pull complete
Digest: sha256:5a210567cf3015c43c90268f948c28257439040bdd97bfdb48252fb785da46db
Status: Downloaded newer image for wernight/ngrok:latest
90127f511da3702aad5ecfe0c14d1c7ef3f5e10f36112392186fc736f2c239d1
vagrant@vagrant:~$ curl $(docker port ngrok 4040)/api/tunnels
{"tunnels":[{"name":"command_line","uri":"/api/tunnels/command_line","public_url":"https://326a164c.ngrok.io","proto":"h
ct":true},"metrics":{"conns":{"count":0,"gauge":0,"rate1":0,"rate5":0,"rate15":0,"p50":0,"p90":0,"p95":0,"p99":0},"http"
5":0,"p50":0,"p90":0,"p95":0,"p99":0}}},{"name":"command_line (http)","uri":"/api/tunnels/command_line+%28http%29","publ
oto":"http","config":{"addr":"www:80","inspect":true},"metrics":{"conns":{"count":0,"gauge":0,"rate1":0,"rate5":0,"rate1
"http":{"count":0,"rate1":0,"rate5":0,"rate15":0,"p50":0,"p90":0,"p95":0,"p99":0}}}],"uri":"/api/tunnels"}
vagrant@vagrant:~$
```

Access that URL from any browser on any machine anywhere in the world. The request from the browser should be handled by the Docker Container, in this case the `www` container running `nginx`.



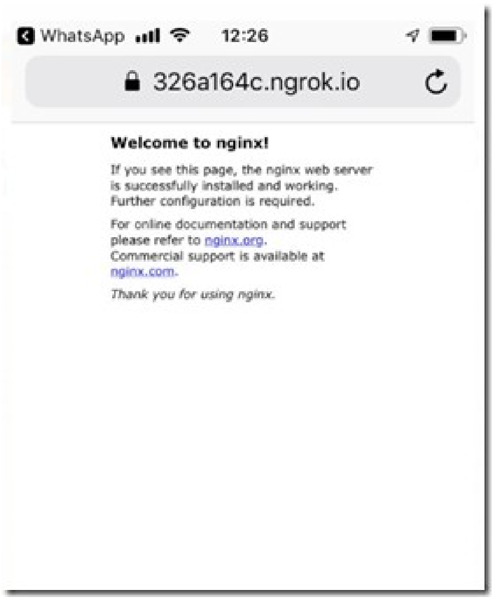
https://326a164c.ngrok.io

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.


WhatsApp 12:26

326a164c.ngrok.io

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Tutorial 2: Expose a local Node Application on the Internet with `ngrok`-generated public URL

On the Docker host – for example the Ubuntu Linux VM created with Vagrant as described here – clone the code-cafe GitHub repository:

```
git clone https://github.com/AMIS-Services/code-cafe
```

Then navigate to the directory that contains the Node application that we will expose on the internet:

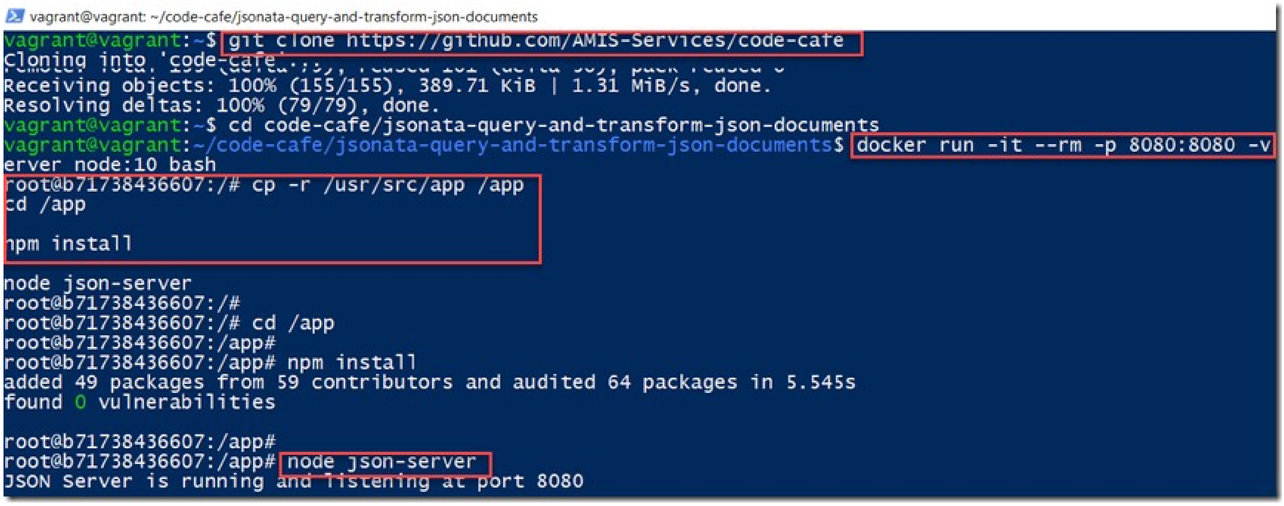
```
cd code-cafe/jsonata-query-and-transform-json-documents
```

And run a Docker container with a Node runtime called `json-server`; the current directory (`$PWD`) is mapped into the container at `/usr/src/app`. The container is associated with the `myngroknet` network that makes it accessible later on to the container running `ngrok`.

```
docker run -it --rm -p 8080:8080 -v "$PWD":/usr/src/app --net
myngroknet --name json-server node:10 bash
```

Once the container is started, you will find yourself in a shell in the container. Perform the following steps to copy the sources, install dependencies and run the Node application:

```
cp -r /usr/src/app /app
cd /app
npm install
node json-server
```



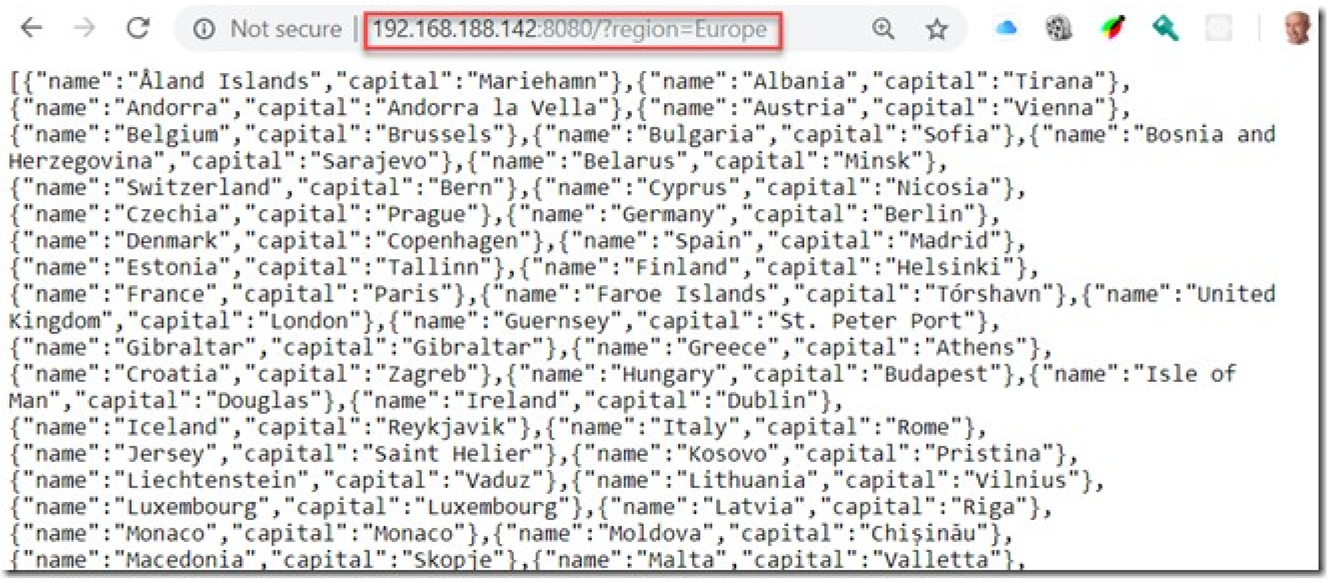
```
vagrant@vagrant: ~/code-cafe/jsonata-query-and-transform-json-documents
vagrant@vagrant:~$ git clone https://github.com/AMIS-Services/code-cafe
Cloning into 'code-cafe'...
Receiving objects: 100% (155/155), 389.71 KiB | 1.31 MiB/s, done.
Resolving deltas: 100% (79/79), done.
vagrant@vagrant:~$ cd code-cafe/jsonata-query-and-transform-json-documents
vagrant@vagrant:~/code-cafe/jsonata-query-and-transform-json-documents$ docker run -it --rm -p 8080:8080 -v
erver node:10 bash
root@b71738436607:/# cp -r /usr/src/app /app
cd /app

npm install

node json-server
root@b71738436607:/#
root@b71738436607:/# cd /app
root@b71738436607:/app#
root@b71738436607:/app# npm install
added 49 packages from 59 contributors and audited 64 packages in 5.545s
found 0 vulnerabilities

root@b71738436607:/app#
root@b71738436607:/app# node json-server
JSON Server is running and listening at port 8080
```

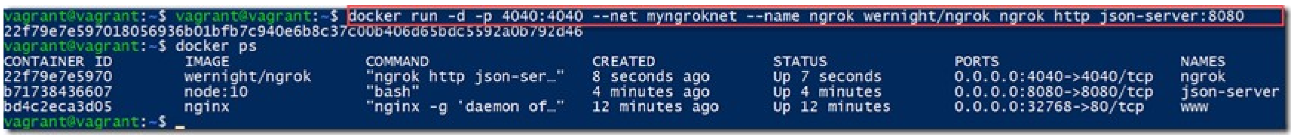
The Node application is up and listening at port 8080. You can verify this from the Docker Host `http://localhost:8080/?region=Europe` (or possibly the Windows host: `http://&lt;vagrant VM IP&gt;:8080/?region=Europe`.



```
[{"name":"Åland Islands","capital":"Mariehamn"},{"name":"Albania","capital":"Tirana"},
{"name":"Andorra","capital":"Andorra la Vella"},{"name":"Austria","capital":"Vienna"},
{"name":"Belgium","capital":"Brussels"},{"name":"Bulgaria","capital":"Sofia"},{"name":"Bosnia and
Herzegovina","capital":"Sarajevo"},{"name":"Belarus","capital":"Minsk"},
{"name":"Switzerland","capital":"Bern"},{"name":"Cyprus","capital":"Nicosia"},
{"name":"Czechia","capital":"Prague"},{"name":"Germany","capital":"Berlin"},
{"name":"Denmark","capital":"Copenhagen"},{"name":"Spain","capital":"Madrid"},
{"name":"Estonia","capital":"Tallinn"},{"name":"Finland","capital":"Helsinki"},
{"name":"France","capital":"Paris"},{"name":"Faroe Islands","capital":"Tórshavn"},{"name":"United
Kingdom","capital":"London"},{"name":"Guernsey","capital":"St. Peter Port"},
{"name":"Gibraltar","capital":"Gibraltar"},{"name":"Greece","capital":"Athens"},
{"name":"Croatia","capital":"Zagreb"},{"name":"Hungary","capital":"Budapest"},{"name":"Isle of
Man","capital":"Douglas"},{"name":"Ireland","capital":"Dublin"},
{"name":"Iceland","capital":"Reykjavik"},{"name":"Italy","capital":"Rome"},
{"name":"Jersey","capital":"Saint Helier"},{"name":"Kosovo","capital":"Pristina"},
{"name":"Liechtenstein","capital":"Vaduz"},{"name":"Lithuania","capital":"Vilnius"},
{"name":"Luxembourg","capital":"Luxembourg"},{"name":"Latvia","capital":"Riga"},
{"name":"Monaco","capital":"Monaco"},{"name":"Moldova","capital":"Chișinău"},
{"name":"Macedonia","capital":"Skopje"},{"name":"Malta","capital":"Valletta"},
```

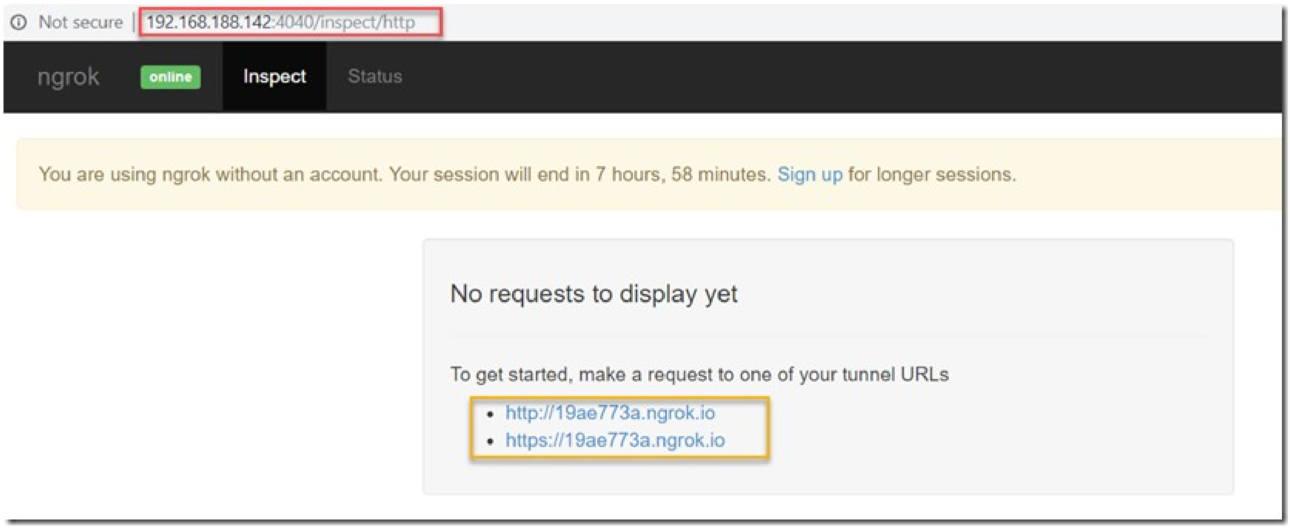
Run the `ngrok` Docker Container to create a tunnel from the a newly assigned public URL to port 8080 on the `json-server` container (at which the Node application is handling requests).

```
docker run -d -p 4040:4040 --net myngroknet --name ngrok wernight/
ngrok ngrok http json-server:8080
```



```
vagrant@vagrant:~$ vagrant@vagrant:~$ docker run -d -p 4040:4040 --net myngroknet --name ngrok wernight/ngrok ngrok http json-server:8080
22f79e7e597018056936b01bfb7c940e6b8c37c00b406d65bdc5592a0b792d46
vagrant@vagrant:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
22f79e7e5970        wernight/ngrok      "ngrok http json-ser…"   8 seconds ago       Up 7 seconds        0.0.0.0:4040->4040/tcp   ngrok
b71738436607        node:10             "bash"                   4 minutes ago       Up 4 minutes        0.0.0.0:8080->8080/tcp   json-server
bd4c2eca3d05        nginx               "nginx -g 'daemon of…"   12 minutes ago      Up 12 minutes       0.0.0.0:32768->80/tcp    www
vagrant@vagrant:~$ _
```

Inspect ngrok at port 4040 and learn about the public url – or use `curl $(docker port ngrok 4040)/api/tunnels` to get that url.



Not secure | 192.168.188.142:4040/inspect/http

ngrok online Inspect Status

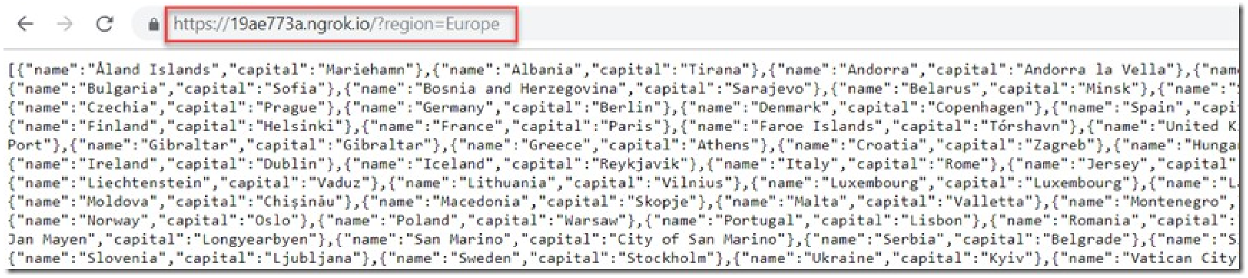
You are using ngrok without an account. Your session will end in 7 hours, 58 minutes. Sign up for longer sessions.

No requests to display yet

To get started, make a request to one of your tunnel URLs

- http://19ae773a.ngrok.io
- https://19ae773a.ngrok.io

Access the Node application from any client anywhere in the world (for example your mobile device) at the url: `http://&lt;assigned ngrok id&gt;.ngrok.io/?region=Europe`

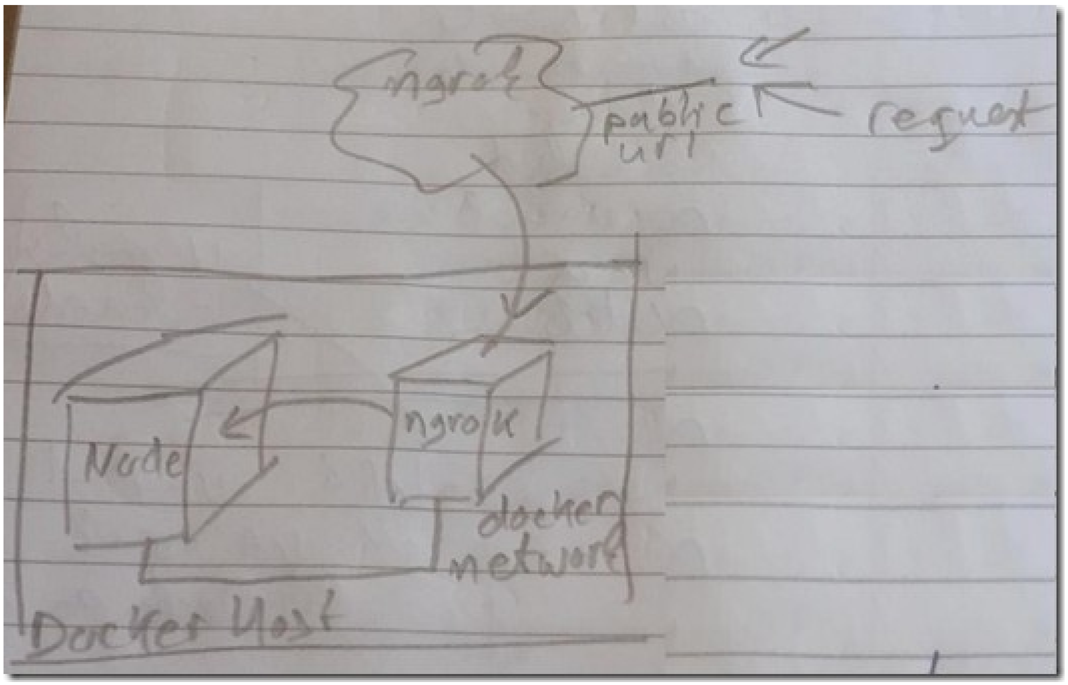


https://19ae773a.ngrok.io/?region=Europe

```
[{"name":"Åland Islands","capital":"Mariehamn"},{"name":"Albania","capital":"Tirana"},{"name":"Andorra","capital":"Andorra la Vella"},{"nam
{"name":"Bulgaria","capital":"Sofia"},{"name":"Bosnia and Herzegovina","capital":"Sarajevo"},{"name":"Belarus","capital":"Minsk"},{"name":"
{"name":"Czechia","capital":"Prague"},{"name":"Germany","capital":"Berlin"},{"name":"Denmark","capital":"Copenhagen"},{"name":"Spain","capi
{"name":"Finland","capital":"Helsinki"},{"name":"France","capital":"Paris"},{"name":"Faroe Islands","capital":"Tórshavn"},{"name":"United K
Port"},{"name":"Gibraltar","capital":"Gibraltar"},{"name":"Greece","capital":"Athens"},{"name":"Croatia","capital":"Zagreb"},{"name":"Hunga
{"name":"Ireland","capital":"Dublin"},{"name":"Iceland","capital":"Reykjavik"},{"name":"Italy","capital":"Rome"},{"name":"Jersey","capital"
{"name":"Liechtenstein","capital":"Vaduz"},{"name":"Lithuania","capital":"Vilnius"},{"name":"Luxembourg","capital":"Luxembourg"},{"name":"L
{"name":"Moldova","capital":"Chișinău"},{"name":"Macedonia","capital":"Skopje"},{"name":"Malta","capital":"Valletta"},{"name":"Montenegro",
{"name":"Norway","capital":"Oslo"},{"name":"Poland","capital":"Warsaw"},{"name":"Portugal","capital":"Lisbon"},{"name":"Romania","capital":
Jan Mayen","capital":"Longyearbyen"},{"name":"San Marino","capital":"City of San Marino"},{"name":"Serbia","capital":"Belgrade"},{"name":"S
{"name":"Slovenia","capital":"Ljubljana"},{"name":"Sweden","capital":"Stockholm"},{"name":"Ukraine","capital":"Kyiv"},{"name":"Vatican City
```

So what do we have running now?

Two containers on a Docker host – which can be a local one or a cloud based host. One container with a Node JS runtime and a custom Node application (cloned from GitHub) and the other with the `ngrok client`. The containers are linked through a Docker network. The `ngrok client` is connected to the `ngrok` public site – requests sent to a specific URL on that site are relayed to the `ngrok client` that in turn sends these requests to the companion container that it acts as a side car for.



Alternative Offerings

Ngrok is not the only option for exposing local services through a public url. Two alternatives are briefly introduced below.

Localtunnel

localtunnel exposes your localhost to the world for easy testing and sharing! No need to mess with DNS or deploy just to have others test out your changes.

Check out: https://github.com/localtunnel/localtunnel

Localtunnel is available in a Docker Container, very similar to the ngrok solution discussed overhead:

https://hub.docker.com/r/efrecon/localtunnel/ .

Note: localtunnel can use localtunnel.me as its server – or you can run your own server to handle all requests (see: https://github.com/localtunnel/server)

Vagrant Share

Vagrant Share allows you to share your Vagrant environment with anyone in the world, enabling collaboration directly in your Vagrant environment in almost any network environment with just a single command: vagrant share.

See https://www.vagrantup.com/docs/share/ for details
and http://www.gizmola.com/blog/archives/121-Vagrant-Share-and-Ngrok.html
for more background.