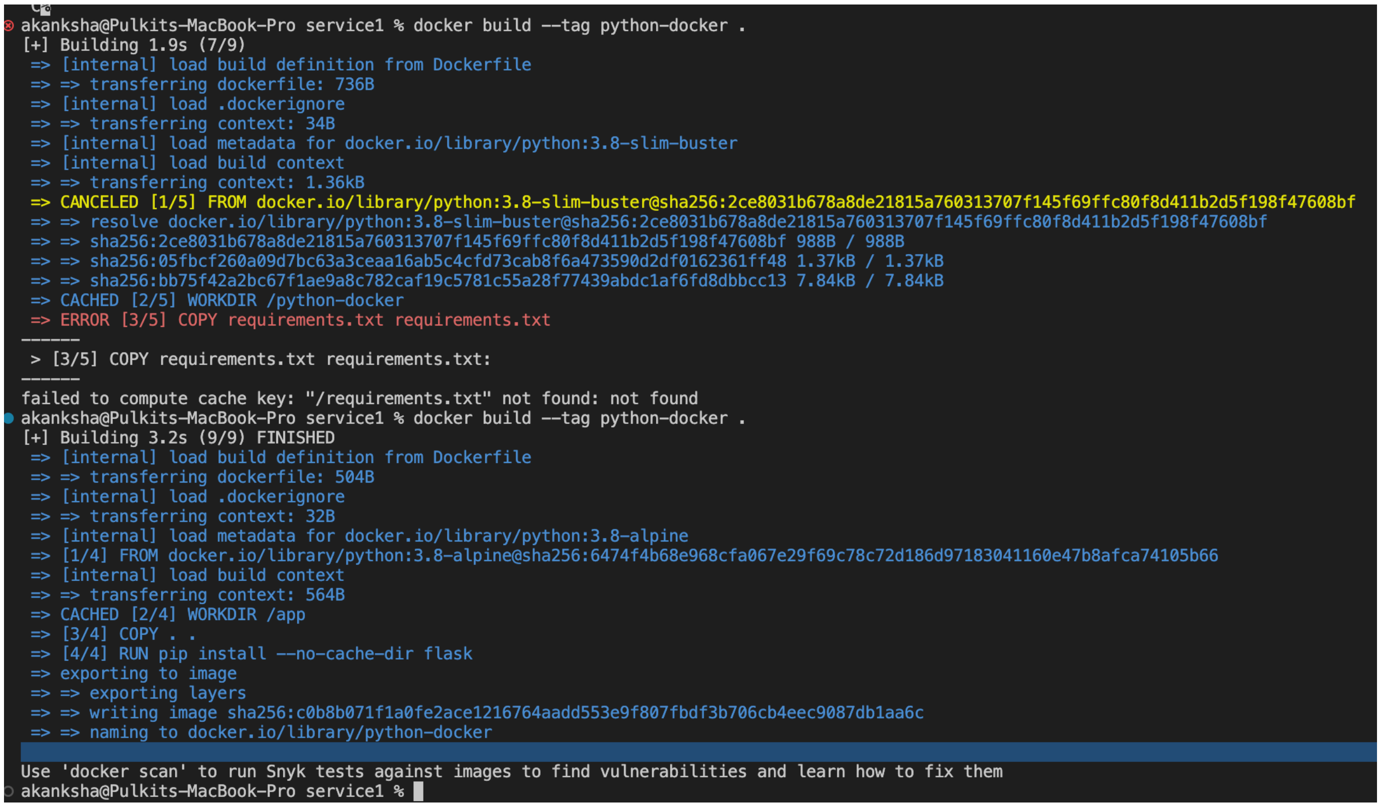
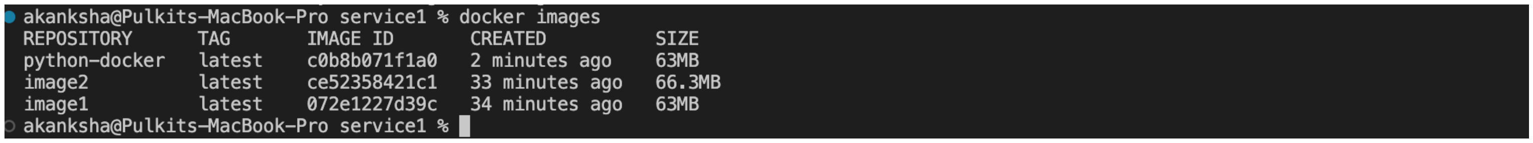
For Service 1:

After the python code, built the docker file:



```
akanksha@Pulkits-MacBook-Pro service1 % docker build --tag python-docker .
[+] Building 1.9s (7/9)
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 736B
 => [internal] load .dockerignore
 => => transferring context: 34B
 => [internal] load metadata for docker.io/library/python:3.8-slim-buster
 => [internal] load build context
 => => transferring context: 1.36kB
 => CANCELED [1/5] FROM docker.io/library/python:3.8-slim-buster@sha256:2ce8031b678a8de21815a760313707f145f69ffc80f8d411b2d5f198f47608bf
 => => resolve docker.io/library/python:3.8-slim-buster@sha256:2ce8031b678a8de21815a760313707f145f69ffc80f8d411b2d5f198f47608bf
 => => sha256:2ce8031b678a8de21815a760313707f145f69ffc80f8d411b2d5f198f47608bf 988B / 988B
 => => sha256:05fbcf260a09d7bc63a3ceaa16ab5c4cfd73cab8f6a473590d2df0162361ff48 1.37kB / 1.37kB
 => => sha256:bb75f42a2bc67f1ae9a8c782caf19c5781c55a28f77439abdc1af6fd8dbbcc13 7.84kB / 7.84kB
 => CACHED [2/5] WORKDIR /python-docker
 => ERROR [3/5] COPY requirements.txt requirements.txt
------
 > [3/5] COPY requirements.txt requirements.txt:
------
failed to compute cache key: "/requirements.txt" not found: not found
akanksha@Pulkits-MacBook-Pro service1 % docker build --tag python-docker .
[+] Building 3.2s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 504B
 => [internal] load .dockerignore
 => => transferring context: 32B
 => [internal] load metadata for docker.io/library/python:3.8-alpine
 => [1/4] FROM docker.io/library/python:3.8-alpine@sha256:6474f4b68e968cfa067e29f69c78c72d186d97183041160e47b8afca74105b66
 => [internal] load build context
 => => transferring context: 564B
 => CACHED [2/4] WORKDIR /app
 => [3/4] COPY . .
 => [4/4] RUN pip install --no-cache-dir flask
 => exporting to image
 => => exporting layers
 => => writing image sha256:c0b8b071f1a0fe2ace1216764aadd553e9f807fbdf3b706cb4eec9087db1aa6c
 => => naming to docker.io/library/python-docker

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
akanksha@Pulkits-MacBook-Pro service1 %
```

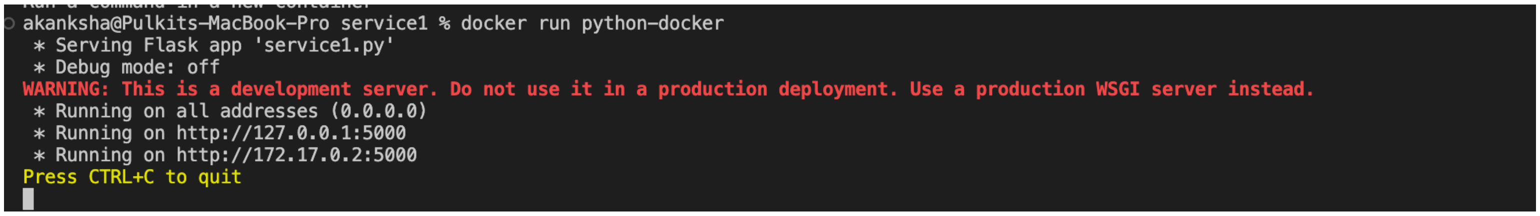
To view images from the command line, execute the following:



```
akanksha@Pulkits-MacBook-Pro service1 % docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
python-docker   latest    c0b8b071f1a0   2 minutes ago    63MB
image2          latest    ce52358421c1   33 minutes ago   66.3MB
image1          latest    072e1227d39c   34 minutes ago   63MB
akanksha@Pulkits-MacBook-Pro service1 %
```

We can see our image is there as python-docker.

Then execute the command:
Docker run python-docker

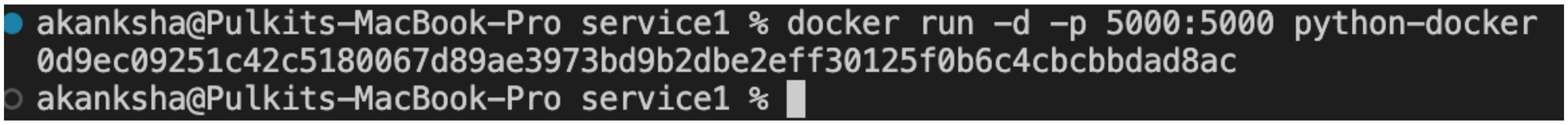


```
akanksha@Pulkits-MacBook-Pro service1 % docker run python-docker
 * Serving Flask app 'service1.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

While running the above command, you'll notice that on the command line it indicates that the application is running. But when you enter `http://localhost:5000/` on the browser, the greeting will be:
This site can't be reached localhost refused to connect.
Regardless of whether the container is running, it is doing so in isolation mode and cannot connect to localhost:5000.

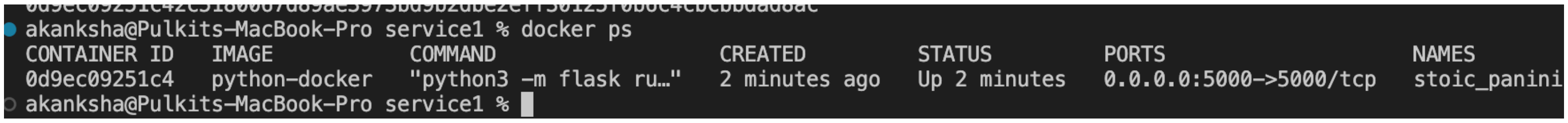
Then run in detached mode using

```
docker run -d -p 5000:5000 python-docker
```



```
akanksha@Pulkits-MacBook-Pro service1 % docker run -d -p 5000:5000 python-docker
0d9ec09251c42c5180067d89ae3973bd9b2dbe2eff30125f0b6c4cbcbbdad8ac
akanksha@Pulkits-MacBook-Pro service1 %
```

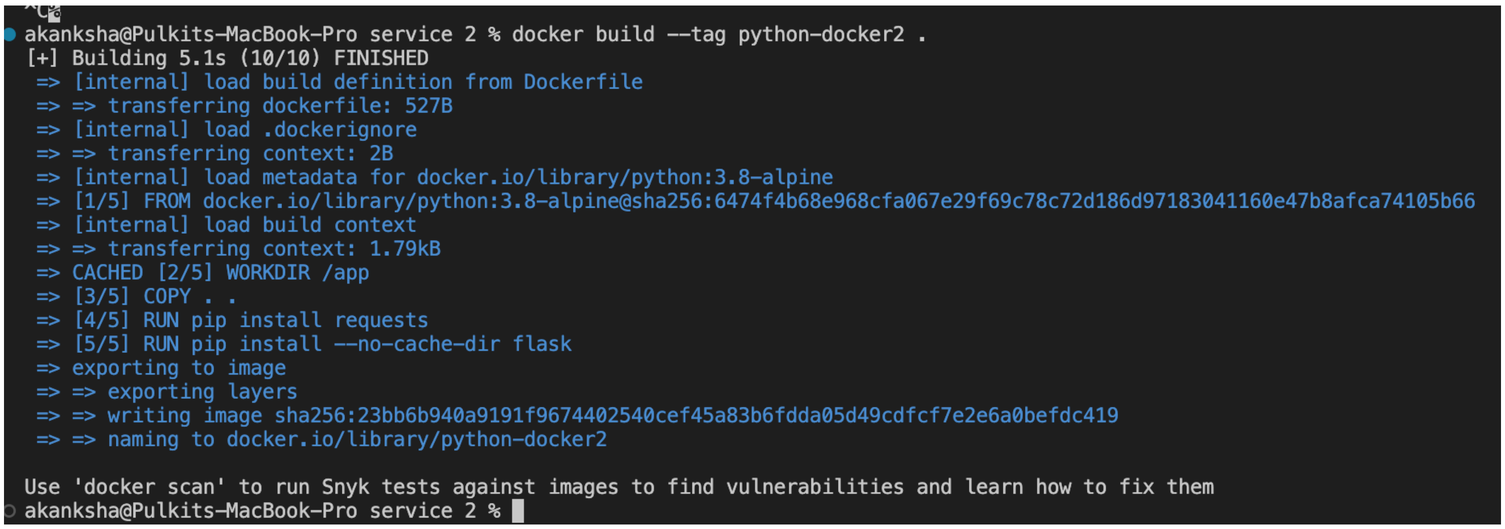
We can see that our image is running as python-docker:



```
akanksha@Pulkits-MacBook-Pro service1 % docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS         PORTS                    NAMES
0d9ec09251c4   python-docker   "python3 -m flask ru…"   2 minutes ago   Up 2 minutes   0.0.0.0:5000->5000/tcp   stoic_panini
akanksha@Pulkits-MacBook-Pro service1 %
```

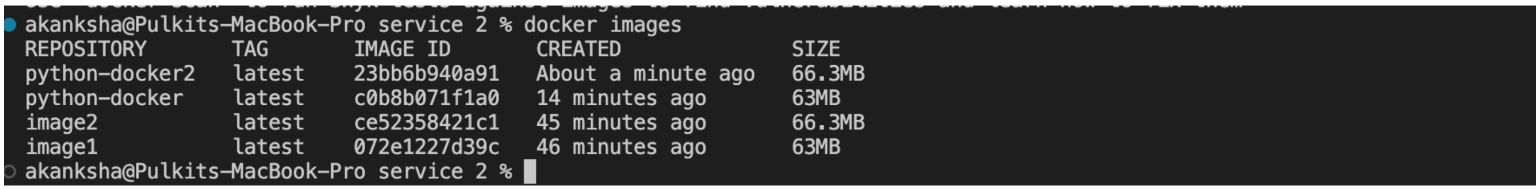
For Service2:

Step 1:



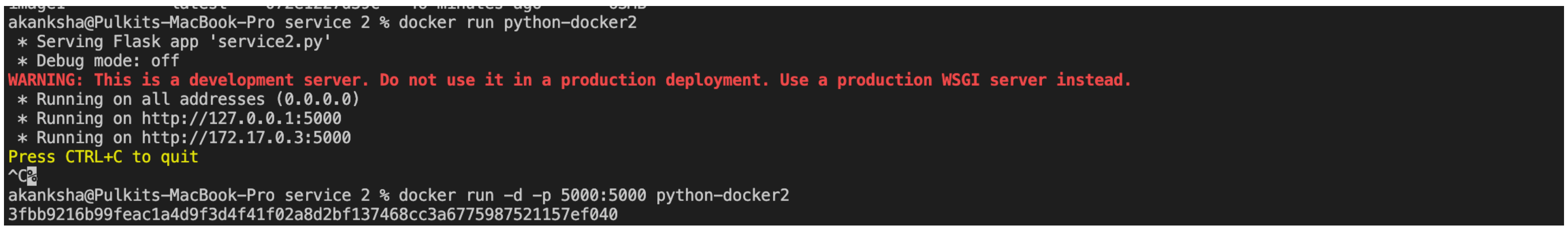
```
akanksha@Pulkits-MacBook-Pro service 2 % docker build --tag python-docker2 .
[+] Building 5.1s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 527B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.8-alpine
 => [1/5] FROM docker.io/library/python:3.8-alpine@sha256:6474f4b68e968cfa067e29f69c78c72d186d97183041160e47b8afca74105b66
 => [internal] load build context
 => => transferring context: 1.79kB
 => CACHED [2/5] WORKDIR /app
 => [3/5] COPY . .
 => [4/5] RUN pip install requests
 => [5/5] RUN pip install --no-cache-dir flask
 => exporting to image
 => => exporting layers
 => => writing image sha256:23bb6b940a9191f9674402540cef45a83b6fdda05d49cdfcf7e2e6a0befdc419
 => => naming to docker.io/library/python-docker2

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
akanksha@Pulkits-MacBook-Pro service 2 %
```

Step 2:



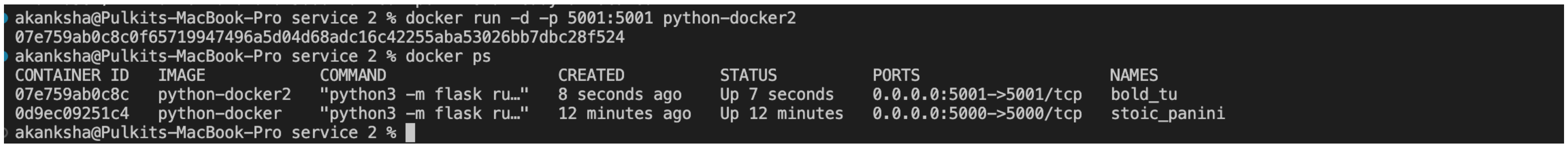
```
akanksha@Pulkits-MacBook-Pro service 2 % docker images
REPOSITORY       TAG       IMAGE ID       CREATED              SIZE
python-docker2   latest    23bb6b940a91   About a minute ago   66.3MB
python-docker    latest    c0b8b071f1a0   14 minutes ago       63MB
image2           latest    ce52358421c1   45 minutes ago       66.3MB
image1           latest    072e1227d39c   46 minutes ago       63MB
akanksha@Pulkits-MacBook-Pro service 2 %
```

Step 3:



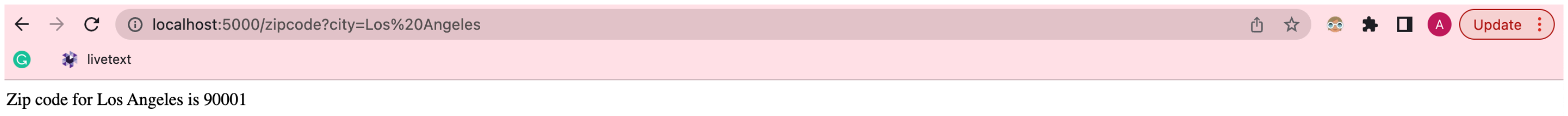
```
akanksha@Pulkits-MacBook-Pro service 2 % docker run python-docker2
 * Serving Flask app 'service2.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.3:5000
Press CTRL+C to quit
^C
akanksha@Pulkits-MacBook-Pro service 2 % docker run -d -p 5000:5000 python-docker2
3fbb9216b99feac1a4d9f3d4f41f02a8d2bf137468cc3a6775987521157ef040
```

Step 4:



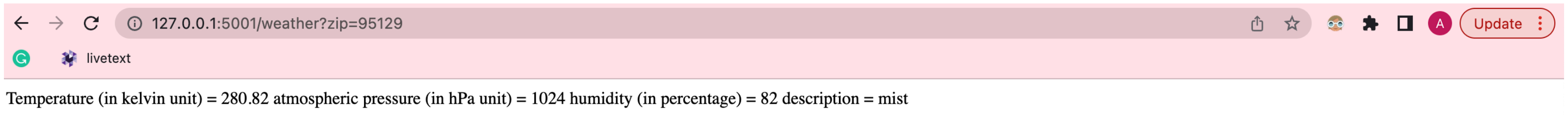
```
akanksha@Pulkits-MacBook-Pro service 2 % docker run -d -p 5001:5001 python-docker2
07e759ab0c8c0f65719947496a5d04d68adc16c42255aba53026bb7dbc28f524
akanksha@Pulkits-MacBook-Pro service 2 % docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED          STATUS          PORTS                    NAMES
07e759ab0c8c   python-docker2   "python3 -m flask ru…"   8 seconds ago    Up 7 seconds    0.0.0.0:5001->5001/tcp   bold_tu
0d9ec09251c4   python-docker    "python3 -m flask ru…"   12 minutes ago   Up 12 minutes   0.0.0.0:5000->5000/tcp   stoic_panini
akanksha@Pulkits-MacBook-Pro service 2 %
```

From Localhost for service 1:



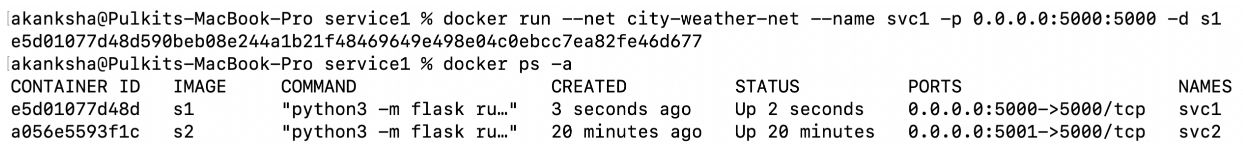
Zip code for Los Angeles is 90001

From localhost for service 2:



Temperature (in kelvin unit) = 280.82 atmospheric pressure (in hPa unit) = 1024 humidity (in percentage) = 82 description = mist

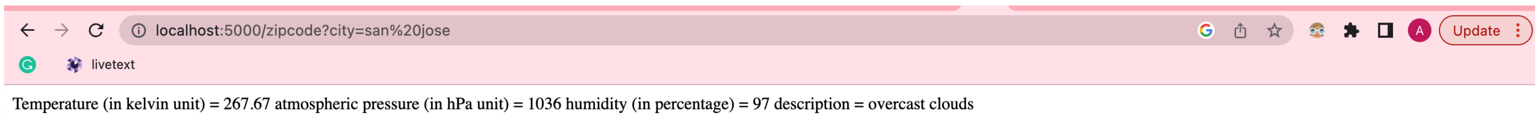
Setting up the connectivity between the 2 services:



```
akanksha@Pulkits-MacBook-Pro service1 % docker run --net city-weather-net --name svc1 -p 0.0.0.0:5000:5000 -d s1
e5d01077d48d590beb08e244a1b21f48469649e498e04c0ebcc7ea82fe46d677
akanksha@Pulkits-MacBook-Pro service1 % docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                    NAMES
e5d01077d48d   s1        "python3 -m flask ru…"   3 seconds ago    Up 2 seconds    0.0.0.0:5000->5000/tcp   svc1
a056e5593f1c   s2        "python3 -m flask ru…"   20 minutes ago   Up 20 minutes   0.0.0.0:5001->5000/tcp   svc2
```

```
[/app # wget -q -O - 172.17.0.2:5000/weather?zip=95129
 Temperature (in kelvin unit) = 280.82
 atmospheric pressure (in hPa unit) = 1024
 humidity (in percentage) = 82
 description = mist172.17.0.3 - - [09/Feb/2023 22:28:25] "GET /weather?zip=95129 HTTP/1.1" 200 -
[                                                                                           /app #
[/app #
[/app #
[/app #
[akanksha@Pulkits-MacBook-Pro service1 %
[akanksha@Pulkits-MacBook-Pro service1 %
[akanksha@Pulkits-MacBook-Pro service1 % docker exec -it c sh
[/app # wget -q -O - 172.17.0.2:5000/weather?zip=95129
 Temperature (in kelvin unit) = 280.82
 atmospheric pressure (in hPa unit) = 1024
 humidity (in percentage) = 82
 description = mist172.17.0.3 - - [09/Feb/2023 22:29:13] "GET /weather?zip=95129 HTTP/1.1" 200 -
[                                                                                           /app #
[/app #
[/app #
[/app # wget -q -O - 172.17.0.2:5000/weather?zip=95012
172.17.0.3 - - [09/Feb/2023 22:29:19] "GET /weather?zip=95012 HTTP/1.1" 200 -
                                                                           Temperature (in kelvin unit) = 280.13
 atmospheric pressure (in hPa unit) = 1024
 humidity (in percentage) = 96
[ description = light rain/app #
```

After setting the connectivity, it looks like this on localhost:



Temperature (in kelvin unit) = 267.67 atmospheric pressure (in hPa unit) = 1036 humidity (in percentage) = 97 description = overcast clouds

Curl:

```
[akanksha@Pulkits-MacBook-Pro service1 % curl -X GET "http://localhost:5000/zipcode?city=san%20jose"
 Temperature (in kelvin unit) = 267.67
 atmospheric pressure (in hPa unit) = 1036
 humidity (in percentage) = 97
[ description = overcast clouds%
```

So finally we got the weather by giving the name of the city via service 1 itself.