MEC LOCATION API

– Hands on

TURIN | NOVEMBER 2020

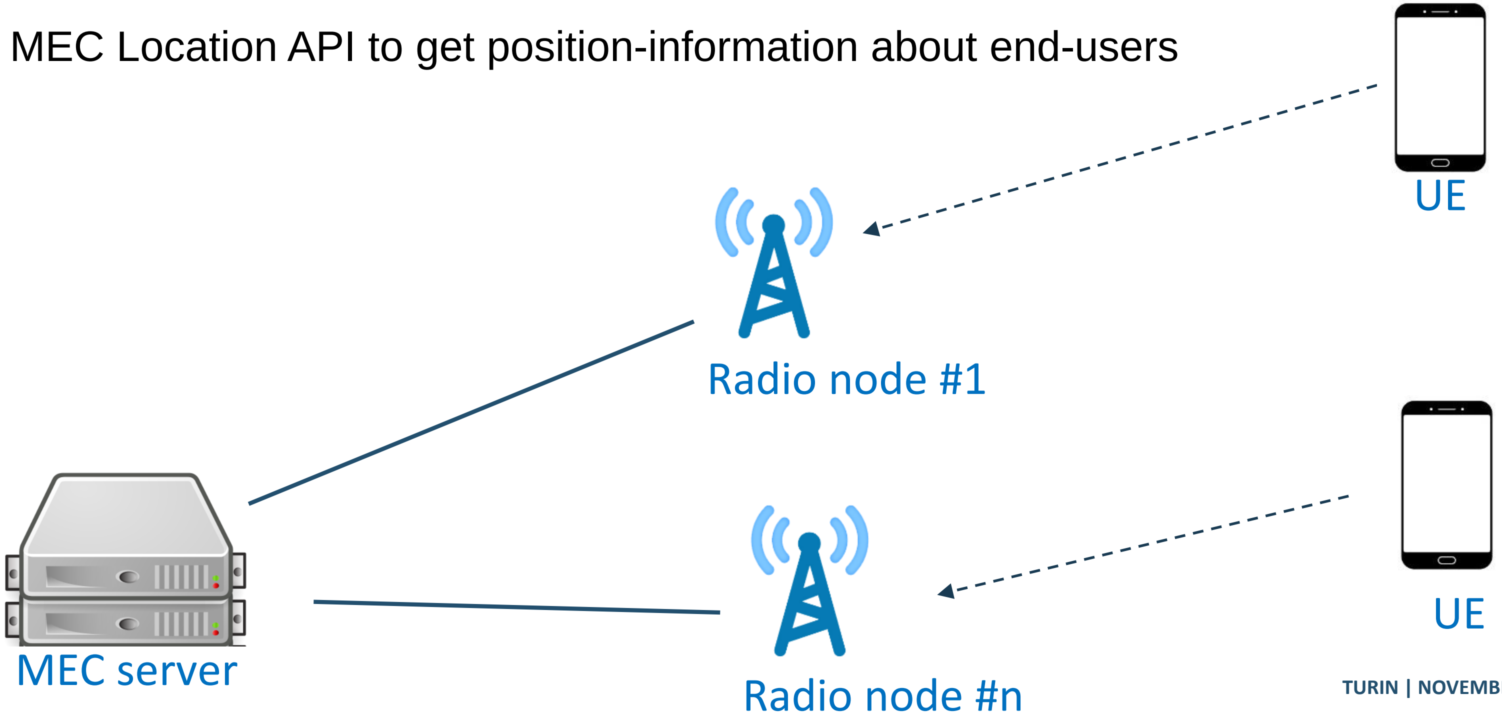
linksfoundation.com

OVERVIEW

- Overview of ETSI MEC Location API
- LINKS Foundation Solution
- MEC Location API Simulator
- Example of a Consumer Application
- Future Implementations

MEC LOCATION API

- Defined in ETSI GS MEC 013 V2.1.1 (2019-09) - Mobile Edge Computing (MEC); Location API
- MEC location API provides **real-time location information** of the user equipment served by the radio node(s) associated with the Mobile edge host
  - ❑ no need any software installed at the end-user side
  - ❑ information provided by the radio nodes
- MEC applications can **query** the MEC Location API to get position-information about end-users



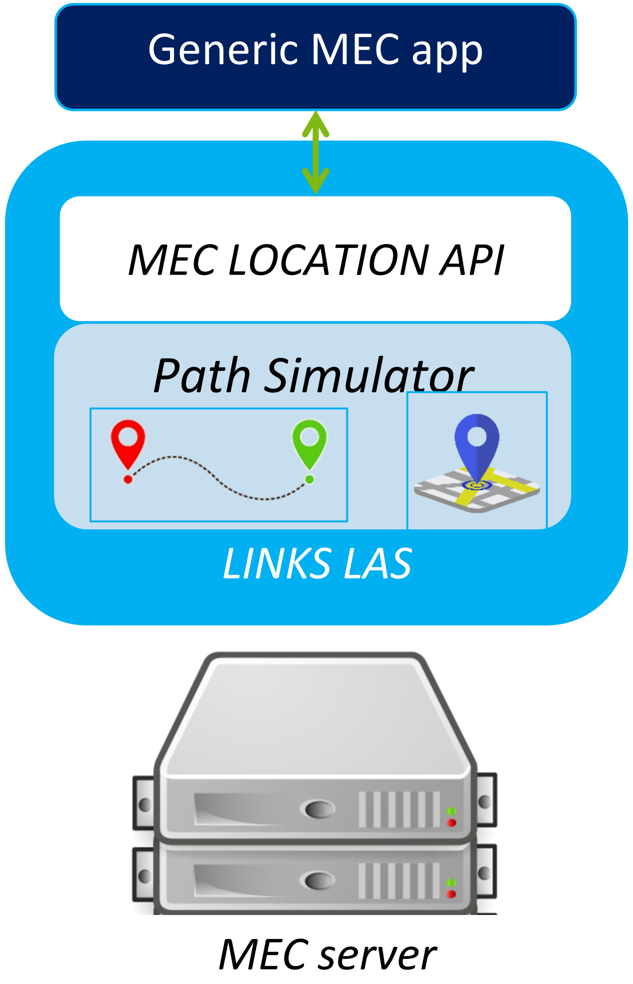
MEC LOCATION API

FEATURES

- **RESTful API**
- **Lookup procedures**
  - location reported only ***once for each location*** information request
  - MEC application requests information with GET method and Location API provides information in the body of the response
- **Subscribe procedures**
  - location reported **multiple times** for each location request, periodically or based on specific events, such as location change.
  - MEC application subscribes via POST method for receiving desired information and Location API provides updates of the information with POST messages
- Notifications of subscription received **until** the MEC application performs a Subscribe Cancellation procedure
  - MEC application cancel the subscription using DELETE method

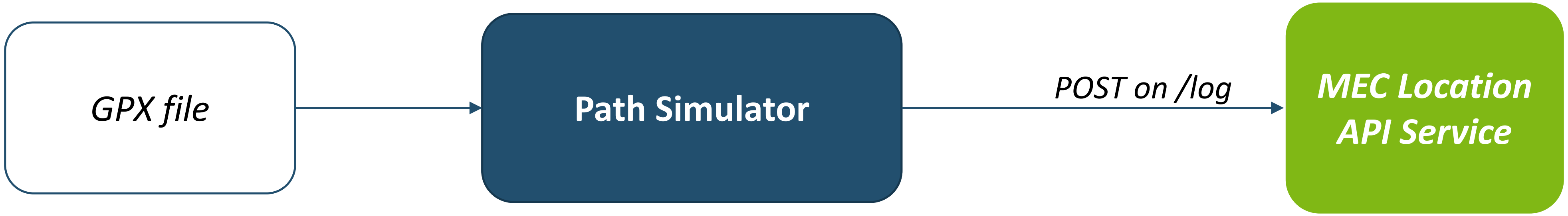
LINKS FOUNDATION SOLUTION

- Location information from radio nodes available only in **real production environments**
  - ❑ **difficult** to develop and test MEC applications exploiting Location API
- LINKS implemented a Location API Simulator (LAS) to **replicate** the behaviour of a real Location API
- Location API simulator implements the same functionalities of the Location API
  - ❑ no need to modify developed MEC applications when real Location API is used



MEC LOCATION API - SIMULATOR

- **Includes** movement UEs (User Equipment) simulator. The path simulator reads GPS traces and it simulates as UEs as wanted, related to a given path



MEC LOCATION API - PRODUCER

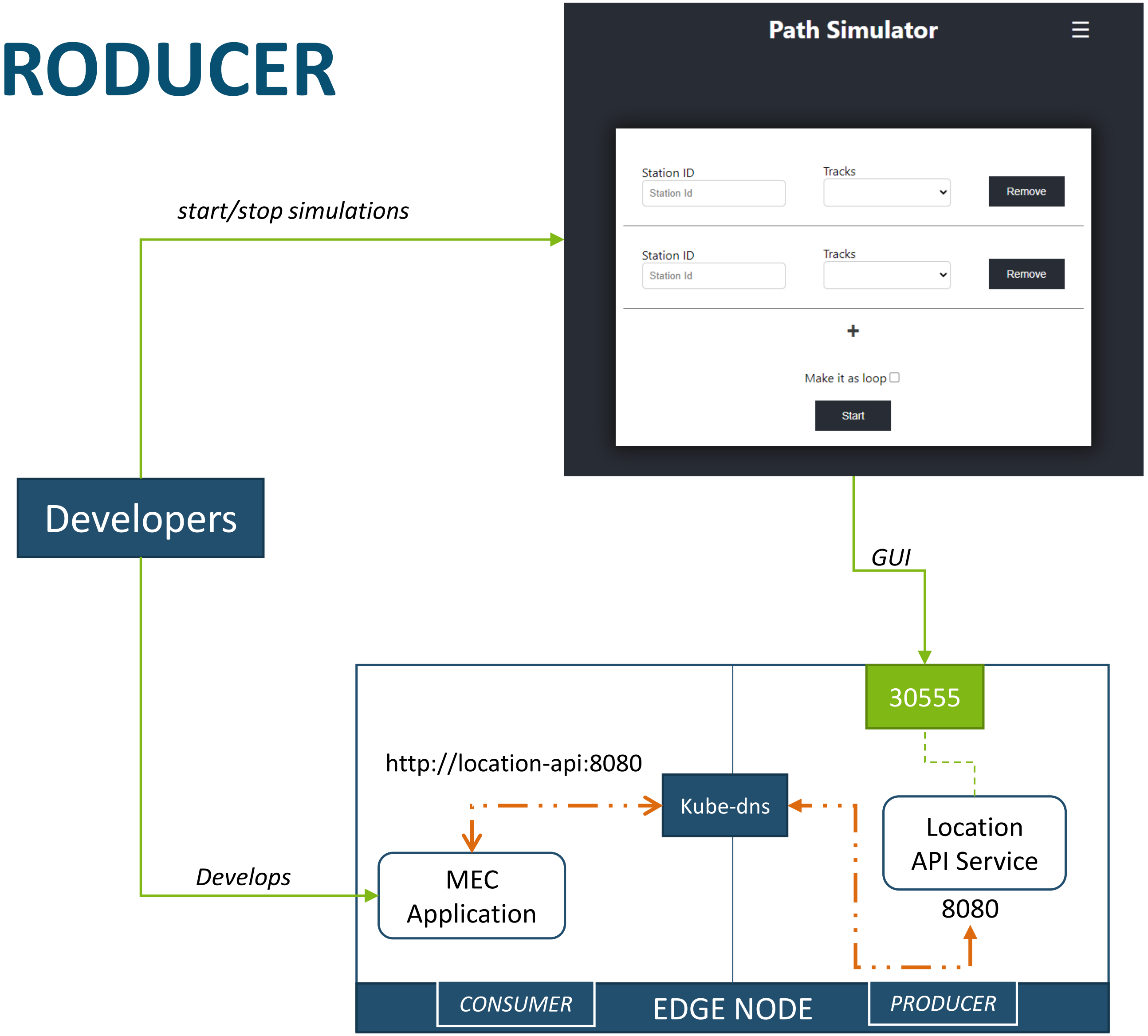
LOCATION API ON THE OPENNESS EDGE NODE

The environment follows the OpenNESS producer/consumer.

The **Location API Service** acts as a *producer* of the service, while **MEC Applications** as *consumers*

The environment includes:

- A GUI for simulations
- Location API Service



Develops

MEC APPLICATION PERSPECTIVE - CONSUMER

MEC LOCATION API AVAILABILITY

- MEC Location APIs available till now are the simplest one. They include:
  - **Lookup procedure**, by which is possible to GET information about all the simulated UEs or a specific one
    - List of UEs: $ curl http://location-api:8080/location/v2/users
    - A specific UE: $ curl http://location-api:8080/location/v2/users?address=acr:10.0.0.1“
  - **Periodic Tracking Subscription**, subscribing to a single or multiple UEs to receive information when they change their position.

MEC APPLICATION PERSPECTIVE - CONSUMER

PERIODIC TRACKING SUBSCRIPTION

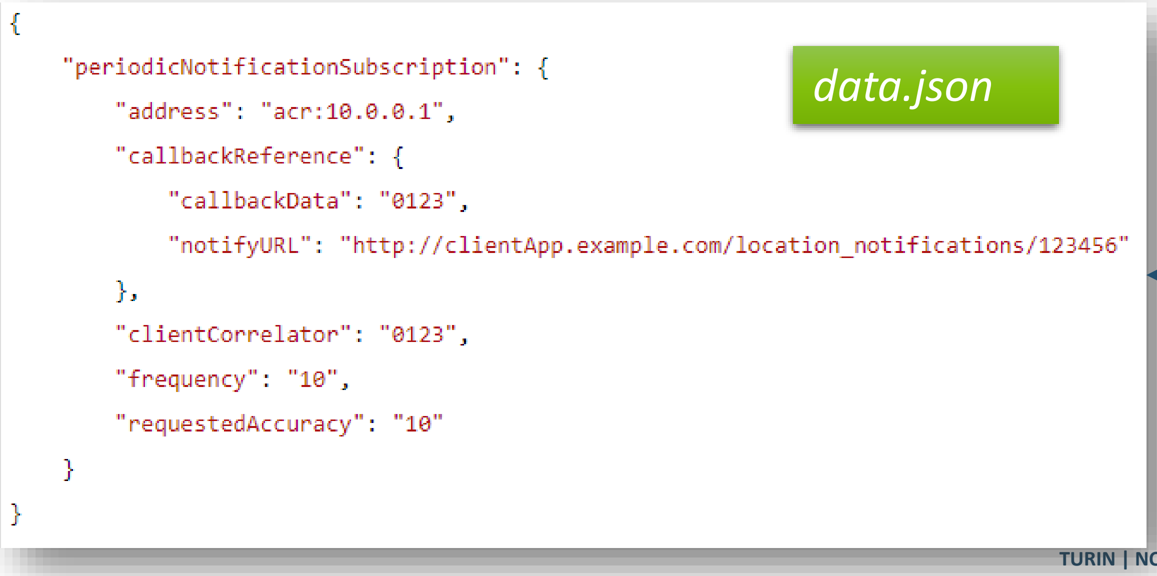
To perform such request, a MEC Application needs to send a POST with a well-formed JSON message.

For instance, a request could be:

- $ curl http://location-api:8080/location/v2/subscriptions/periodic -d @data.json

If the request is accepted, the response will include the generated {subscriptionId}. It will be useful *for* **deleting** the subscription later on. Indeed, the DELETE can be sent as follows:

- $ curl -X DELETE http://location-api:8080/location/v2/subscriptions/periodic/{subscriptionId}



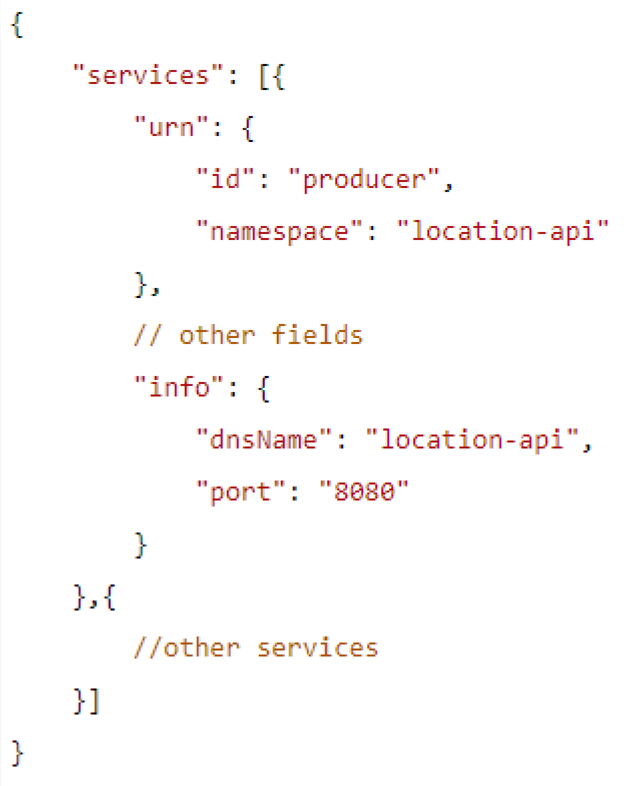
data.json

```
{
    "periodicNotificationSubscription": {
        "address": "acr:10.0.0.1",
        "callbackReference": {
            "callbackData": "0123",
            "notifyURL": "http://clientApp.example.com/location_notifications/123456"
        },
        "clientCorrelator": "0123",
        "frequency": "10",
        "requestedAccuracy": "10"
    }
}
```

MEC APPLICATION SAMPLE - CONSUMER

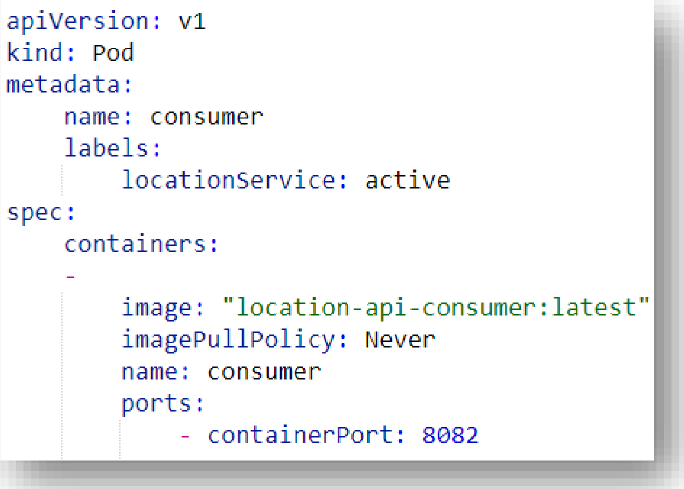
For developing your own MEC Application, you need to:

1. Create your own docker image, which should be able to register on OpenNESS EAA. After the authentication, it starts looking to the available services to the endpoint https://eaa.openness:443/services;
2. Create a well-formed YAML file in the **Controller Node**. Indeed, the Location API Service is accessible only if the consumer POD fits the networking request. To enable it, add the label *locationService: active,* so that it will be privileged to **directly** get and send information to the Location API Service.



```
{
    "services": [{
        "urn": {
            "id": "producer",
            "namespace": "location-api"
        },
        // other fields
        "info": {
            "dnsName": "location-api",
            "port": "8080"
        }
    },{
        //other services
    }]
}
```

(1) List of Services

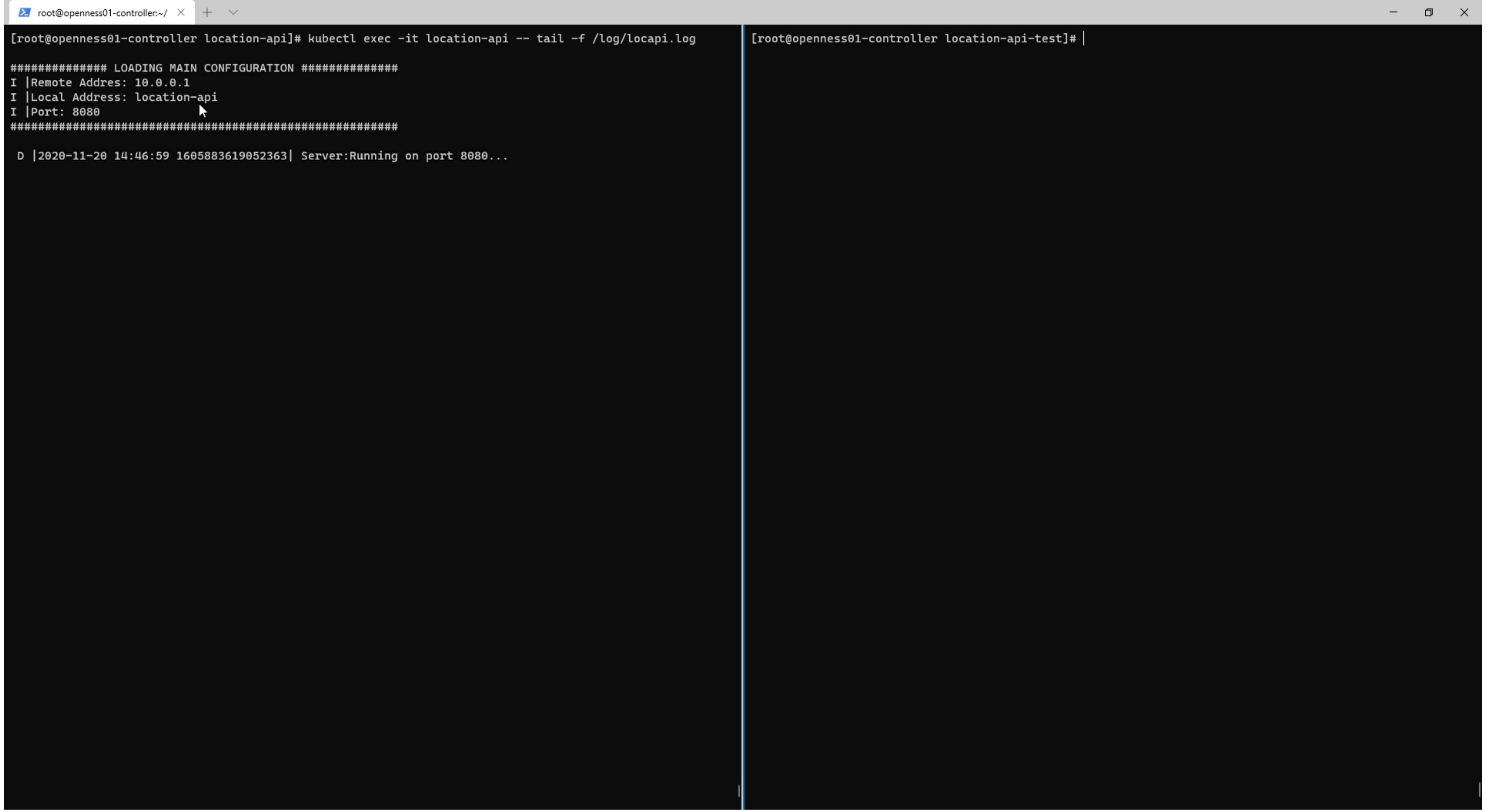


```
apiVersion: v1
kind: Pod
metadata:
    name: consumer
    labels:
        locationService: active
spec:
    containers:
    -
        image: "location-api-consumer:latest"
        imagePullPolicy: Never
        name: consumer
        ports:
            - containerPort: 8082
```

(2) YAML consumer app example

MEC LOCATION API

CONSUMER APPLICATION SAMPLE



```
[root@openness01-controller location-api]# kubectl exec -it location-api -- tail -f /log/locapi.log

############## LOADING MAIN CONFIGURATION ##############
I |Remote Addres: 10.0.0.1
I |Local Address: location-api
I |Port: 8080
########################################################

 D |2020-11-20 14:46:59 1605883619052363| Server:Running on port 8080...
```

```
[root@openness01-controller location-api-test]# 
```

MEC LOCATION API

FUTURE IMPLEMENATION

- The aim is to expand the number of APIs provided and enriching the service functionalities, which means to be fully compliant to ETSI GS MEC 013. Some planned implementations are:
  - **Distance Subscription and Lookup***:* The UE Distance Subscribe o Lookup is the procedure for applications acquiring up-to-data distance of a specific UE to a geographical location, or another UE;
  - **Area Subscription:** The Area subscription is the procedure for applications acquiring UE movement notifications in relation to a **geographic area**
- In addition, improve the Simulator capabilities, distinguishing the kind of UE simulate (car, bike, pedestrian) and their behaviour; Radio Node simulation over a simulated path.