

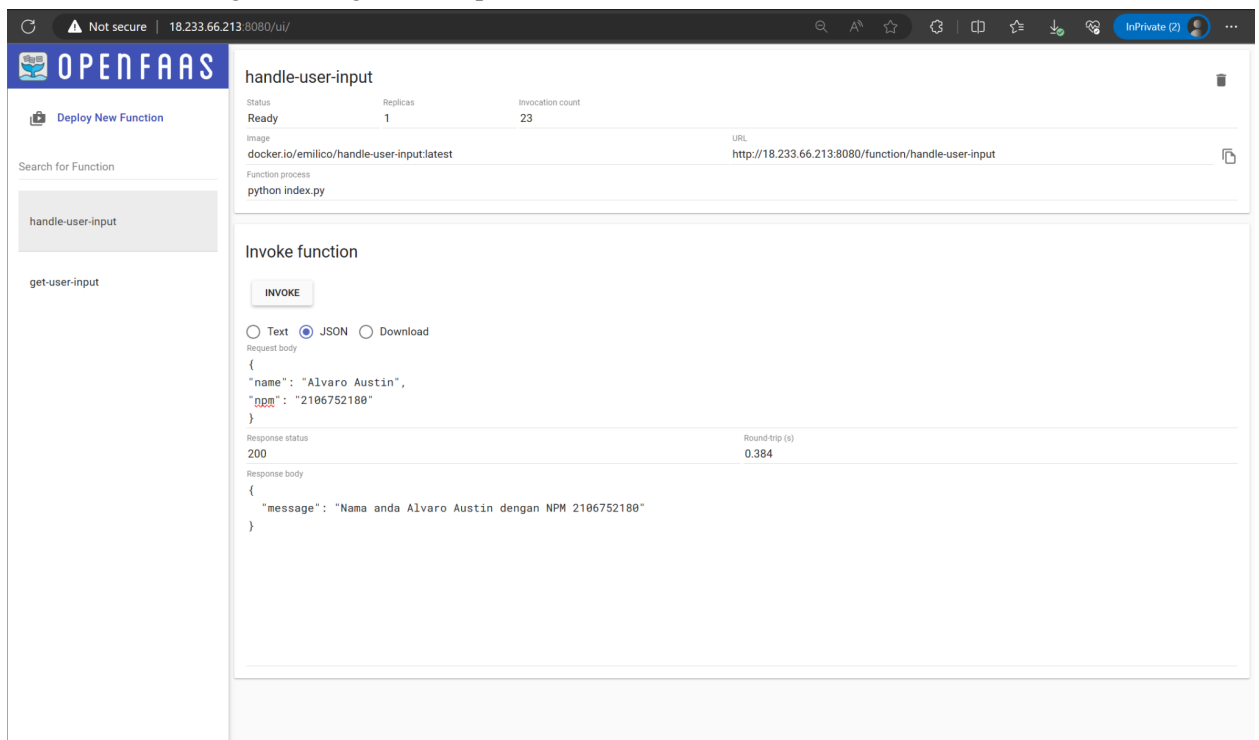
Pada UTS Praktek kali ini, saya ingin memperlihatkan hasil 2 buah serverless service yang saya telah buat dengan satu service memanggil service lainnya. Service saya terdiri dari 2 yaitu:

1. Get User Input (**get-user-input**) dimana service ini sebenarnya berguna untuk melakukan processing input dan mengeluarkan hasilnya.
2. Handle User Input (**handle-user-input**) dimana service ini berguna untuk melakukan return terhadap input yang sudah diprocess oleh service Get User Input sebelumnya.

Hasil jawaban saya perlihatkan dalam 2 bentuk, hasil **curl**, dan juga hasil **invoke** pada interface **UI openFAAS**. Saya juga memberikan 2 section yaitu jawaban dan proses. Hal ini saya lakukan, barangkali proses pengerjaan perlu didokumentasikan juga sehingga saya lakukan hal ini. Terima kasih sudah melihat keterangan ini 🙏. Mohon maaf apabila ada kesalahan dalam pengerjaannya.

Jawaban

1. Laman Service dengan nama get-user-input



The screenshot shows the OpenFaaS web interface in a browser. The address bar shows the URL `18.233.66.213:8080/ui/`. The page title is "handle-user-input". On the left sidebar, there is a "Deploy New Function" button and a search bar. Below the search bar, the function "handle-user-input" is listed, and "get-user-input" is selected. The main content area displays the details for the "handle-user-input" function:

- Status: Ready
- Replicas: 1
- Invocation count: 23
- Image: `docker.io/emilico/handle-user-input:latest`
- URL: `http://18.233.66.213:8080/function/handle-user-input`
- Function process: `python index.py`

Below the details, there is an "Invoke function" section with an "INVOKE" button. Underneath, there are radio buttons for "Text", "JSON" (selected), and "Download". The "Request body" is shown as a JSON object:

```
{
  "name": "Alvaro Austin",
  "npm": "2106752180"
}
```

The "Response status" is 200, and the "Round-trip (s)" is 0.384. The "Response body" is shown as a JSON object:

```
{
  "message": "Nama anda Alvaro Austin dengan NPM 2106752180"
}
```

2. Laman Service dengan nama handle-user-input (perhatikan urlnya)

The screenshot shows the OpenFaaS dashboard. On the left, there's a sidebar with the OpenFaaS logo, a 'Deploy New Function' button, and a search bar. Below the search bar, a list of functions is shown, with 'handle-user-input' selected. The main panel displays details for the 'handle-user-input' function. It shows the status as 'Ready', 1 replica, and 22 invocation counts. The image used is 'docker.io/emilico/handle-user-input:latest' and the function process is 'python index.py'. The URL is 'http://18.233.66.213:8080/function/handle-user-input'. Below this, there's an 'Invoke function' section with an 'INVOKE' button and radio buttons for 'Text' (selected), 'JSON', and 'Download'. A 'Request body' input field is present. At the bottom, the 'Response status' is '200 OK' and the 'Roundtrip (s)' is '0.606'. The 'Response body' is empty.

3. Kode dari service get-user-input

```
ubuntu@ip-172-31-51-226:~/get-user-input$ cat handler.py
import requests
import json

def handle(req):
    """handle a request to the function
    Args:
        req (str): request body
    """

    url = "http://18.233.66.213:8080/function/handle-user-input"
    data_json = json.loads(req)
    data = {
        "name": data_json.get("name"),
        "npm": data_json.get("npm"),
    }
    json_string = json.dumps(data)
    headers = {'Content-Type': 'application/json'}
    response = requests.post(url, data=json_string, headers=headers)

    return response.text
```

Note: Perlu diubah requirements.txtnya, sehingga:

```
ubuntu@ip-172-31-51-226:~/get-user-input$ cat requirements.txt
requests
ubuntu@ip-172-31-51-226:~/get-user-input$
```

4. Kode dari service handle-user-input

```
ubuntu@ip-172-31-51-226:~/handle-user-input$ cat handler.py
import json

def handle(req):
    """handle a request to the function
    Args:
        req (str): request body
    """
    data = json.loads(req)

    name = data.get("name")
    npm = data.get("npm")

    response = {
        "message": "Nama anda {} dengan NPM {}".format(name, npm)
    }

    return json.dumps(response)
```

5. Hasil dari curl pada EC2 instance:

```
ubuntu@ip-172-31-51-226:~$ curl -H 'Content-Type: application/json' -d '{"name": "Alvaro Austin", "npm": "2106752180"}' "http://127.0.0.1:8080/function/get-user-input"
{"message": "Nama anda Alvaro Austin dengan NPM 2106752180"}
ubuntu@ip-172-31-51-226:~$
```

6. Kode dari service Get User Input (**get-user-input**)

```
import requests
import json

def handle(req):
    """handle a request to the function
    Args:
        req (str): request body
    """

    url = "http://18.233.66.213:8080/function/handle-user-input"
    data_json = json.loads(req)
    data = {
        "name": data_json.get("name"),
        "npm": data_json.get("npm"),
    }
    json_string = json.dumps(data)
    headers = {'Content-Type': 'application/json'}
    response = requests.post(url, data=json_string, headers=headers)

    return response.text
```

7. Kode dari handle-user-input

```
import json

def handle(req):
    """handle a request to the function
    Args:
        req (str): request body
    """
    data = json.loads(req)

    name = data.get("name")
    npm = data.get("npm")

    response = {
        "message": "Nama anda {} dengan NPM {}".format(name, npm)
    }

    return json.dumps(response)
```

BONUS

Process:

1. Membuat EC2 Instance

The screenshot displays the AWS Management Console interface for an EC2 instance. The left sidebar shows the navigation menu with categories like EC2 Dashboard, Images, Elastic Block Store, Network & Security, and Load Balancing. The main content area is titled 'Instance summary for i-00dc2e3746df4bb4f' and shows various details:

- Instance ID:** i-00dc2e3746df4bb4f
- Public IPv4 address:** 18.233.66.213
- Instance state:** Running
- Private IP DNS name:** ip-172-31-51-226.ec2.internal
- Instance type:** t2.medium
- VPC ID:** vpc-087955dd08fc6f3ec
- Subnet ID:** subnet-07d33e56f91846a71
- Platform:** Ubuntu (Inferred)
- Platform details:** Linux/UNIX
- Stop protection:** Disabled
- AMI ID:** ami-053b0d53c279acc90
- AMI name:** ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230516
- Launch time:** Mon Oct 16 2023 11:12:27 GMT+0700 (Western Indonesia Time) (about 2 hours)
- Lifecycle:** normal
- Monitoring:** disabled
- Termination protection:** Disabled
- AMI location:** amazon/ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230516
- Stop-hibernate behavior:** Disabled

2. Instalasi Faas-cli dan docker

```
ubuntu@ip-172-31-51-226:~$ faas-cli

OpenFaaS

Manage your OpenFaaS functions from the command line

Usage:
  faas-cli [flags]
  faas-cli [command]

Available Commands:
  build      Builds OpenFaaS function containers
  completion Generates shell auto completion
  deploy     Deploy OpenFaaS functions
  describe   Describe an OpenFaaS function
  generate    Generate Kubernetes CRD YAML file
  help       Help about any command
  invoke     Invoke an OpenFaaS function
  list       List OpenFaaS functions
  local-run  Start a function with docker for local testing (experimental feature)
  login      Log in to OpenFaaS gateway
  logout     Log out from OpenFaaS gateway
  logs       Fetch logs for a functions
  namespace Manage OpenFaaS namespaces
  new        Create a new template in the current folder with the name given as name
  plugin     Manage plugins
  publish    Builds and pushes multi-arch OpenFaaS container images

i-00dc2e3746df4bb4f
PublicIPs: 18.233.66.213 PrivateIPs: 172.31.51.226
```

3. Membuat 2 service (get-user-input dan handle-user-input)

```
ubuntu@ip-172-31-51-226:~$ faas-cli new get-user-input --lang python
2023/10/16 04:24:04 No templates found in current directory.
2023/10/16 04:24:04 Attempting to expand templates from https://github.com/openfaas/templates.git
2023/10/16 04:24:05 Patched 17 template(s) : [bun catasp dockerfile go java11 java11-vert-x node node14 node16 node17 node18 php7 php8 python python3 python3-debian ruby] from https://github.com/openfaas/templates.git
Folder: get-user-input created.

OpenFaaS

Function created in folder: get-user-input
Stack file written: get-user-input.yml
ubuntu@ip-172-31-51-226:~$ faas-cli new handle-user-input --lang python -g "http://127.0.0.1:8081"
Folder: handle-user-input created.

OpenFaaS

Function created in folder: handle-user-input
Stack file written: handle-user-input.yml
ubuntu@ip-172-31-51-226:~$

ubuntu@ip-172-31-51-226:~$ ls
faasd  get-docker.sh  get-user-input  get-user-input.yml  handle-user-input  handle-user-input.yml  template
```

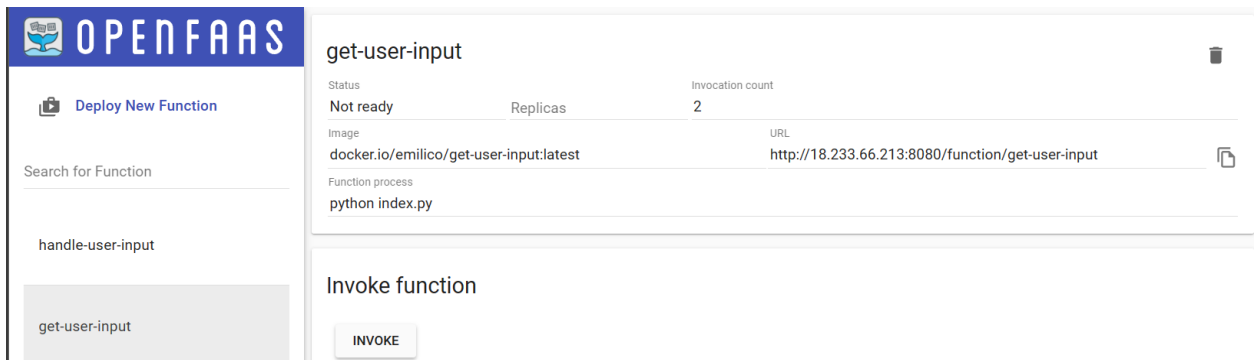
4. Melakukan up pada service get-user-input (selanjutnya juga pada handle-user-input)

```
[0] > Pushing get-user-input [emilico/get-user-input:latest]
The push refers to repository [docker.io/emilico/get-user-input]
d466a9724640: Layer already exists
55f5714c0960: Layer already exists
5f70bf18a086: Layer already exists
5881ceea100e: Layer already exists
44eca0deac20: Layer already exists
98d3aaaa99d7: Layer already exists
cfc3f96d2882: Layer already exists
b60e9c045f3e: Layer already exists
a7f9f7c5f9cf: Layer already exists
a0e50ae596ca: Layer already exists
97b24682868f: Layer already exists
0cf8dc1f6a73: Layer already exists
8cc4d49c2cb6: Layer already exists
a58c05b00d43: Layer already exists
83f4a7583b9c: Layer already exists
879c0d8666e3: Layer already exists
20a7b70bdf2f: Layer already exists
3fc750b41be7: Layer already exists
beee9f30bc1f: Layer already exists
latest: digest: sha256:8d1242bb588839ca15d0daac67dc92fc1a051718f8d85029a902839ff984d1d1 size: 4693
[0] < Pushing get-user-input [emilico/get-user-input:latest] done.
[0] Worker done.
Deploying: get-user-input.

Deployed. 200 OK.
URL: http://127.0.0.1:8080/function/get-user-input

ubuntu@ip-172-31-51-226:~$
```

5. Apabila kita membuka UI dari openFAAS:



The screenshot shows the OpenFaaS web interface. On the left, there's a sidebar with the 'OPENFAAS' logo and a 'Deploy New Function' button. Below that is a search bar with the text 'handle-user-input' and a list of functions including 'get-user-input'. The main panel displays details for the 'get-user-input' function. It shows a status of 'Not ready', 2 replicas, and an invocation count of 2. The image is 'docker.io/emilico/get-user-input:latest' and the URL is 'http://18.233.66.213:8080/function/get-user-input'. The function process is 'python index.py'. At the bottom, there's an 'Invoke function' section with an 'INVOKE' button.

Status	Replicas	Invocation count
Not ready	2	2

Image: docker.io/emilico/get-user-input:latest
URL: http://18.233.66.213:8080/function/get-user-input
Function process: python index.py

Invoke function
INVOKE

6. Kita dapat melihat urlnya, kita dapat menggunakan informasi tersebut pada kode handler kita.

```
ubuntu@ip-172-31-51-226:~/get-user-input$ cat handler.py
import requests
import json

def handle(req):
    """handle a request to the function
    Args:
        req (str): request body
    """

    url = "http://18.233.66.213:8080/function/handle-user-input"
    data_json = json.loads(req)
    data = {
        "name": data_json.get("name"),
        "npm": data_json.get("npm"),
    }
    json_string = json.dumps(data)
    headers = {'Content-Type': 'application/json'}
    response = requests.post(url, data=json_string, headers=headers)

    return response.text
```

7. Selanjutnya hanya disesuaikan saja sesuai dengan hasil yang diinginkan, saya menginginkan service 2 (handle-user-input) untuk melakukan handle input tersebut, sehingga menghasilkan message berupa "Nama saya Alvaro Austin dengan NPM 2106752180".

```
ubuntu@ip-172-31-51-226:~/handle-user-input$ cat handler.py
import json

def handle(req):
    """handle a request to the function
    Args:
        req (str): request body
    """
    data = json.loads(req)

    name = data.get("name")
    npm = data.get("npm")

    response = {
        "message": "Nama anda {} dengan NPM {}".format(name, npm)
    }

    return json.dumps(response)
```

8. Setelah itu curl/menggunakan interface web yang disediakan.