

گزارش پیاده‌سازی سرویس 1

هدف این سرویس پر کردن داده‌های ناموجود در یک سری زمانی است. به این صورت که سرویس پیاده‌سازی شده یک json را که شامل بخش‌های زیر است، به عنوان ورودی دریافت می‌کند:

```
7  {"data": {
8    "time": {
9      "0": 1577836800000,
10     "1": 1577923200000,
11     "2": 1578896000000
12   },
13   "vol": {
14     "0": 20,
15     "1": 40
16     "2": 100
17   }
18 }
19 "confing": {
20   "type": "miladi",
21   "time": "daily",
22   "interpolation": "linear"
23 }
24
25
26
27 }
28 |
```

افزون‌براین، برای پیاده‌سازی این سرویس، به یک فایل پایتون app، یک فایل پایتون interpolation_methods، یک فایل پایتون common، یک dockerfile، یک فایل متن requirements و یک فایل yml که در این‌جا آن را fandogh نام‌گذاری کردیم، نیاز است.

فایل json خروجی نیز به‌صورت زیر خواهد بود:

```
{
  "data": {
    "time": {
      "0": 1577836800000,
      "1": 1577923200000,
      "2": 1578009600000,
      "3": 1578096000000
    },
    "vol": {
      "0": 20,
      "1": 40,
      "2": 70,
      "3": 100
    }
  }
}
```

توضیحاتی درخصوص هر یک از بخش‌ها

:app1

```
3 from flask import Flask, request
4 from utils.common import response_message, read_json_time_series
5 from utils.interpolation_methods import linear_interpolation
6 app = Flask(__name__)
7
8 @app.route('/', methods=['GET', 'POST'])
9 def isup():
10     return response_message('API is active')
11
12 @app.route('/service1', methods=['GET', 'POST'])
13 def interpolation():
14     req = request.get_json()
15     data = read_json_time_series(req['data'])
16     config = req['config']
17
18     if config['type'] == 'miladi':
19         result = linear_interpolation(data, config)
20         result = result.ro_json()
21     return response_message(dict({"data": result}))
22
23 if __name__ == '__main__':
24     app.run(host='0.0.0.0', port=80)
```

در تابع interpolation نوشته‌شده، یک ریکوئست ورودی را به‌صورت json دریافت می‌شود.

Response_message را “API is active” تعریف کرده‌ایم.

در تابع interpolation، req، data و config تعریف شده‌اند و اگر config میلادی باشد، نتیجه دیتا را برمی‌گرداند.

:input

(عکس در صفحه 1)

شامل یک بخش دیتا و یک بخش از config‌ها (که در آن type، میلادی، time، روزانه و interpolation به‌صورت خطی در نظر گرفته شده است).

:common

```
3 import gzip
4 from flask import make_response, json
5 import pandas as pd
6 import numpy as np
7 import json
8
9 def response_message(data=None, status=200):
10     if status in range(200, 400):
11         content = gzip.compress(json.dumps(data, ensure_ascii=False, indent=3, default=convert,
12                                         sort_keys=False).encode('utf8'), 5)
13     else:
14         content = gzip.compress(
15             json.dumps({'message': data, 'status': 'error'}, ensure_ascii=False, indent=3).encode('utf-8'), 5)
16     response = make_response(content, status)
17     response.headers['Content-Length'] = len(content)
18     response.headers['Content-Encoding'] = 'gzip'
19     response.headers['Content-Type'] = 'application/json; charset=utf-8'
20
21 def convert(o):
22     if isinstance(o, np.bool_):
23         if o:
24             return True
25         else:
26             return False
27     if pd.isna(o):
28         return None
29
30 def read_json_time_series(dict_data):
31     j_data = json.dumps(dict_data)
32     data = pd.read_json(j_data)
33     data.time = pd.to_datetime(data.time, unit='ms')
34     return data
```

شامل توابع response_message، convert و read_json_time_series (که ورودی را به یک pandas dataframe تبدیل می‌کند) است.

:interpolation_methods

```
3 def linear_interpolation(data, config):
4     if config['time'] == 'daily':
5         data = data.set_index('time')
6         data = data.resample('D')
7         data = data.interpolate(method=config['interpolation'])
8         data.reset_index(inplace=True)
9     elif config['time'] == 'monthly':
10         data = data.set_index('time')
11         data = data.resample('M')
12         data = data.index(inplace=True)
13     else:
14         data = None
15     return data
```

همان‌طور که در عکس کد مشاهده می‌شود، این متود دیتا را می‌گیرد؛ time را ایندکس می‌کند. با resample تاریخ را به‌صورت روزانه درمی‌آورد، interpolation را انجام می‌دهد و در مرحله بعد دیتا را reset index می‌کند. و در خط آخر (15) دیتا را برمی‌گرداند.

:Dockerfile

```
3 FROM python:3.10-slim-buster
4
5 WORKDIR /app1
6
7 COPY . /app1
8
9 RUN pip install -r requirements.txt
10
11 ENTRYPOINT ["python"]
12
13 CMD ["app.py"]
14
15
```

که شامل نسخه استفاده شده، نام دایرکتوری، کپی کردن محتوا در دایرکتوری app، نصب پکیج های مورد نیاز (که در فایل requirement.txt قرار داده شده است) و در نهایت کد app را اجرا می کند.

Requirement.txt:

certifi==2022.5.18.1

click==8.1.3

colorama==0.4.4

Flask==2.1.2

itsdangerous==2.1.2

Jinja2==3.1.2

MarkupSafe==2.1.1

numpy==1.22.4

pandas==1.4.2

python-dateutil==2.8.2

pytz==2022.1

six==1.16.0

Werkzeug==2.1.2

wincertstore==0.2

:yaml

شامل بخش‌های kind، name و spec.

```
3 kind: ExternalService
4 name: fandogh-service
5 spec:
6   image: fandoghi-service:0.1
7   port: 80
8   static_path: static
```

پایاده‌سازی بر روی بستر ابری:

- ساخت یک environment با کامند (در anaconda):

conda create --name myenv

```
C:\WINDOWS\system32\cmd.exe
(base) C:\Users\Aysan>conda create -n myenv scipy
WARNING: A conda environment already exists at 'C:\Users\Aysan\conda\envs\myenv'
Remove existing environment (y/[n])? y

Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.8.2
  latest version: 4.13.0

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\Aysan\conda\envs\myenv
  added / updated specs:
    - scipy

The following packages will be downloaded:

package | build | size
-----|-----|-----
ca-certificates-2022.4.26 | haa95532_0 | 124 KB
certifi-2022.6.15 | py310haa95532_0 | 153 KB
intel-openmp-2021.4.0 | haa95532_3556 | 2.2 MB
libffi-1.4.2 | hd77b12b_4 | 107 KB
mkl-2021.4.0 | haa95532_640 | 114.9 MB
mkl-service-2.4.0 | py310h2bbff1b_0 | 48 KB
mkl_fft-1.3.1 | py310ha064dea_0 | 136 KB
mkl_random-1.2.2 | py310hded8f06_0 | 221 KB
numpy-1.22.3 | py310h62d095c_0 | 25 KB
numpy-base-1.22.3 | py310h206c741_0 | 4.9 MB
openssl-1.1.1p | h2bbff1b_0 | 4.8 MB
pip-21.2.4 | py310haa95532_0 | 1.9 MB
python-3.10.4 | hba2f4b1_0 | 15.9 MB
scipy-1.7.2 | py310h62d095c_0 | 14.0 MB
setuptools-61.2.0 | py310haa95532_0 | 1.0 MB
six-1.16.0 | pyhd3eb1b0_1 | 10 KB
sqlite-3.38.5 | h2bbff1b_0 | 798 KB
tk-8.6.12 | h2bbff1b_0 | 3.1 MB
tzdata-2022a | hda174b7_0 | 109 KB
vc-14.2 | h21ff451_1 | 8 KB
vs2015_runtime-14.27.29016 | h5e58377_2 | 1007 KB
```

- نصب CLI فندوق:

pip install fandogh_cli --upgrade

- در ادامه با دستور fandogh login در کامند وارد حساب فندوق می‌شویم.

```
C:\WINDOWS\system32\cmd.exe
(myenv) C:\Users\Aysan>fandogh login
username: aysan1998
password:
would you like to let Fandogh CLI to send context information in case any unhandled error happens? [y/N]: N
[ASCII ART]
logged in successfully

Welcome to Fandogh Cloud, you can start using our PaaS as quickly as a link click, Try it below:
https://docs.fandogh.cloud/docs/getting-started.html

(myenv) C:\Users\Aysan>fandogh image init --name=fandoghi_service
Image created successfully

(myenv) C:\Users\Aysan>fandogh image publish --version 0.1
Traceback (most recent call last):
  File "C:\Users\Aysan\.conda\envs\myenv\lib\runpy.py", line 196, in _run_module_as_main
    return _run_code(code, main_globals, None,
  File "C:\Users\Aysan\.conda\envs\myenv\lib\runpy.py", line 86, in _run_code
    exec(code, run_globals)
  File "C:\Users\Aysan\.conda\envs\myenv\Scripts\fandogh.exe\_main_.py", line 7, in <module>
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\click\core.py", line 1130, in __call__
    return self.main(*args, **kwargs)
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\click\core.py", line 1055, in main
    rv = self.invoke(ctx)
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\click\core.py", line 1657, in invoke
    return _process_result(sub_ctx.command.invoke(sub_ctx))
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\click\core.py", line 1657, in invoke
    return _process_result(sub_ctx.command.invoke(sub_ctx))
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\fandogh_cli\base_commands.py", line 57, in invoke
    raise exp
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\fandogh_cli\base_commands.py", line 27, in invoke
    return super(FandoghCommand, self).invoke(ctx)
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\click\core.py", line 1404, in invoke
    return ctx.invoke(self.callback, **ctx.params)
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\click\core.py", line 768, in invoke
    return _callback(*args, **kwargs)
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\fandogh_cli\image_commands.py", line 142, in publish
    workspace = Workspace()
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\fandogh_cli\workspace.py", line 23, in __init__
    self._create_tar_file()
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\fandogh_cli\workspace.py", line 29, in _create_tar_file
    self.tardir(self.path, tarf)
  File "C:\Users\Aysan\.conda\envs\myenv\lib\site-packages\fandogh_cli\workspace.py", line 80, in tardir
    tarh.add(os.path.join(self.context, file_path), arcname=file_path)
  File "C:\Users\Aysan\.conda\envs\myenv\lib\tar.py", line 1991, in add
    with bltn_open(name, "rb") as f:
```

- سپس از دستور `fandogh image publish --version 0.1` برای ساخت image استفاده می‌کنیم.

(تصویر بالا)

- همچنین پس از `publish` با دستور `fandogh image publish --version 0.1` با استفاده از دستور `fandogh service apply -f fandogh.yml`، بعد از این‌که داکر image را ساختیم، مشخصات فایل yml را به آن می‌دهیم تا سرویس را اجرا کند.

ایمیج‌ها

نام ایمیج	آخرین ورژن	آخرین آپدیت
fandoghi_service		۱۴۰۱/۰۴/۱۰

تعداد ایمیج‌ها: ۱

لیست ایمیج‌ها