# Lab3 - 函数的介绍

实验3中通过模板来创建函数，目前已经有了17个模板，实验使用python3

```
$ faas-cli template pull
Fetch templates from repository: https://github.com/openfaas/templates.git at
master
2020/01/09 18:57:41 Attempting to expand templates from
https://github.com/openfaas/templates.git
2020/01/09 18:57:51 Fetched 17 template(s) : [csharp csharp-armhf dockerfile go
go-armhf java12 java8 node node-arm64 node-armhf node12 php7 python python-armhf
python3 python3-armhf ruby] from https://github.com/openfaas/templates.git

$ faas-cli new --list
Languages available as templates:
- csharp
- csharp-armhf
- dockerfile
- go
- go-armhf
- java12
- java8
- node
- node-arm64
- node-armhf
- node12
- php7
- python
- python-armhf
- python3
- python3-armhf
- ruby
```

使用Python 3模板创建一个hello-openfaas函数。模板由--lang选项指定。--prefix选项指定的是函数名称的前缀，指定Docker Hub的帐户名，然后其将被发布到Docker Hub。

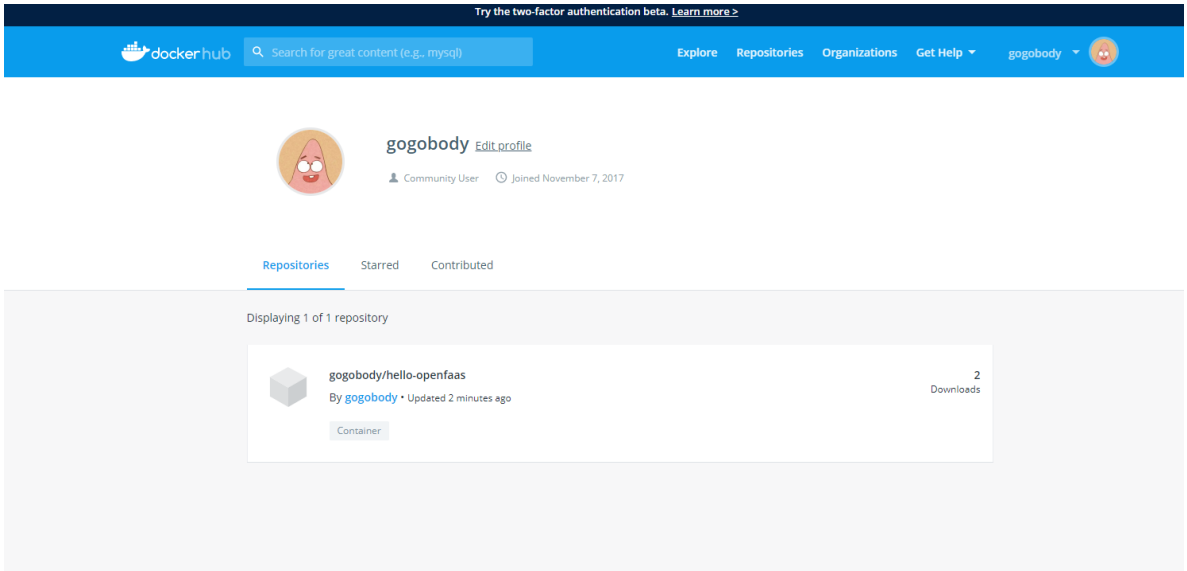
```
$  faas-cli new --lang python3 hello-openfaas --prefix="gogobody"
Folder: hello-openfaas created.
  ___                   _____           ____
 / _ \ _ __   ___ _ __ |  ___|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \| |_ / _` |/ _` \___ \
| |_| | |_) |  __/ | | |  _| (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|


Function created in folder: hello-openfaas
Stack file written: hello-openfaas.yml
```

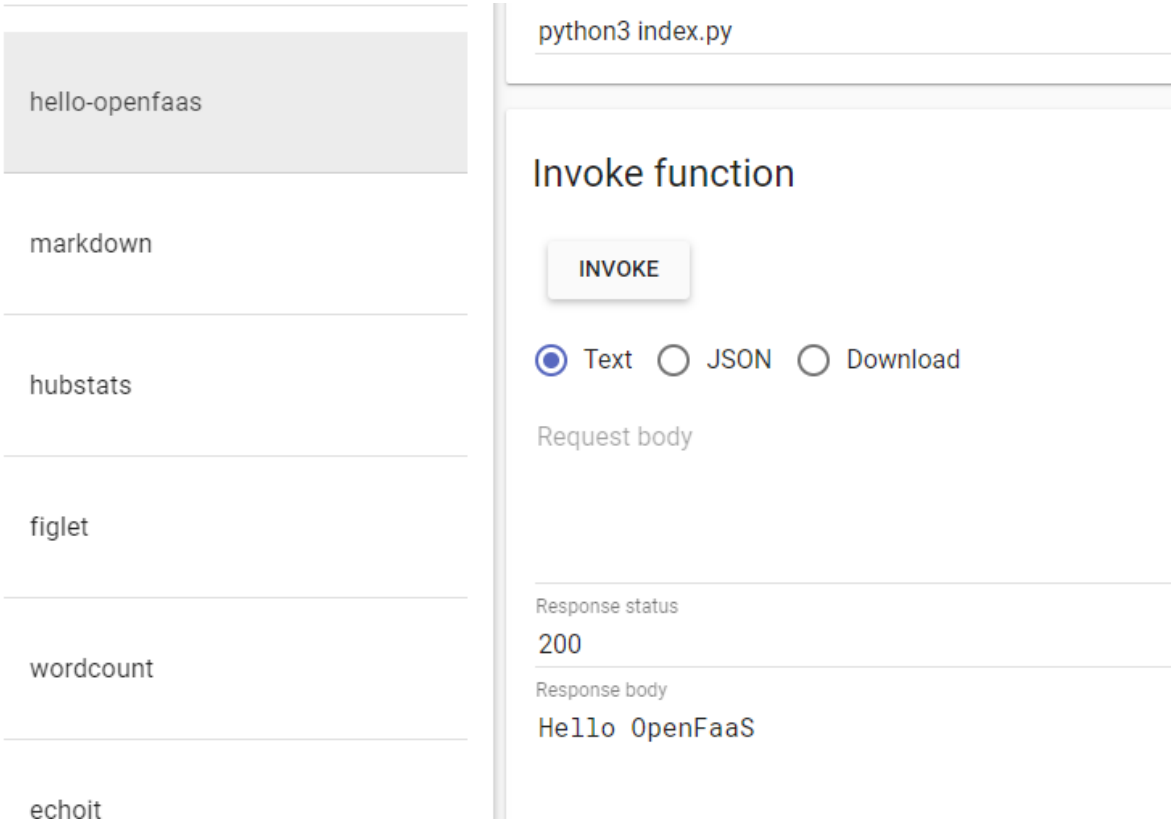
然后，设置将简单字符串“ Hello OpenFaaS”返回到handler.py，并部署hello-openfaas函数。

```
$ cd hello-openfaas
$ nano handler.py
def handle(req):
    """handle a request to the function
    Args:
        req (str): request body
    """

    return "Hello OpenFaaS"
```

流程为faas-cli build→faas-cli push→faas-cli deploy。函数执行可以通过faas-cli调用来执行。

```
$ faas-cli build -f ./hello-openfaas.yml

$ faas-cli push -f ./hello-openfaas.yml
[0] < Pushing hello-openfaas [gogobody/hello-openfaas:latest] done.
[0] Worker done.

$ faas-cli deploy -f ./hello-openfaas.yml
Deploying: hello-openfaas.

Deployed. 202 Accepted.
URL: http://127.0.0.1:8080/function/hello-openfaas

$ echo|faas-cli invoke hello-openfaas
Reading from STDIN - hit (Control + D) to stop.
Hello OpenFaaS
```






接下来实现一个叫 astronaut-finder 的函数，这个函数会从国际空间站上拉取一个随机的宇航员的名字。

重新build和deploy

```
$ echo| faas-cli invoke astronaut-finder
Oleg Kononenko is in space

$ echo| faas-cli invoke astronaut-finder
Anne McClain is in space
```

CLI 的 YAML 文件也可以把函数打组为一个 stack， 里像Docker Compose一样，尝试使用OpenFaaS来管理多个功能的设置。 faas-cli new命令具有--append选项，该选项允许将多个功能组合在同一个Stack文件中。

```
$ faas-cli new --lang python3 first
$ faas-cli new --lang python3 second --append=./first.yml
```

在引入自定义二进制文件时，faas-cli new命令的--lang选项不仅可以指定编程语言，还可以指定dockerfile。此功能允许将指定的Docker映像用作模板，这在定制Function时可能是必需的。

```
$ faas-cli new --lang dockerfile sorter --prefix="gogobody"
```