**K3S with OpenFaaS**

**Requirements:**

**1. Raspberry pi (3 No. )**

**2. USB hub for power supply**

**3. Continuous wifi supply**

**Steps:**

**1. Creating the K3S cluster**

**2. Creating a OpenFaaS service in the cluster**

**3. Building and deploying functions in OpenFaaS**

**4. Creating tunnel through ngrok or inlets TLS for remote access of the cluster**

**Creating the K3S cluster**

**1. Add “**cgroup\_enable=cpuset cgroup\_memory=1 cgroup\_enable=memory**” at the end of the file** /boot/cmdline.txt

**Now reboot the pi**

**2. Among all the raspberrypis , pick any one of the node as server and run the script “k3s.sh”**

**3. Now collect the TOKEN at** /var/lib/rancher/k3s/server/node-token

**The token looks like “**K1089729d4ab5e51a44b1871768c7c04ad80bc6319d7bef5d94c7caaf9b0bd29efc::node:1fcdc14840494f3ebdcad635c7b7a9b7**”**

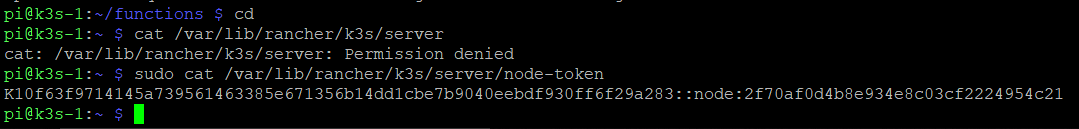
**The Server is up and running by this time**

**Joining the worker nodes**

**On the worker nodes do the following:**

$ export K3S\_URL="https://<**SERVER\_NODE\_IP**>:6443"

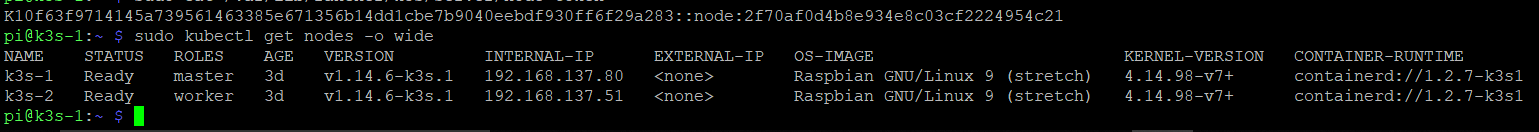
$ export K3S\_TOKEN="K1089729d4ab5e51a44b1871768c7c04ad80bc6319d7bef5d94c7caaf9b0bd29efc::node:1fcdc14840494f3ebdcad635c7b7a9b7"



**Now run the script “k3s.sh” on the worker nodes after exporting the environment variables**

**The cluster is up and ready with master and worker nodes**

**List the nodes**

****

$ kubectl get node -o wide

**Deploying a openFaaS microservice**

Save: openfaas-figlet-svc.yaml.

apiVersion: v1

kind: Service

metadata:

name: openfaas-figlet

labels:

app: openfaas-figlet

spec:

type: NodePort

ports:

- port: 8080

protocol: TCP

targetPort: 8080

nodePort: 31111

selector:

app: openfaas-figlet

Save: openfaas-figlet-dep.yaml.

apiVersion: apps/v1

kind: Deployment

metadata:

name: openfaas-figlet

labels:

app: openfaas-figlet

spec:

replicas: 1

selector:

matchLabels:

app: openfaas-figlet

template:

metadata:

labels:

app: openfaas-figlet

spec:

containers:

- name: openfaas-figlet

image: functions/figlet:latest-armhf

imagePullPolicy: Always

ports:

- containerPort: 31112

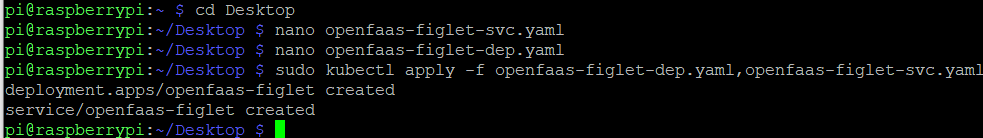
protocol: TCP

* Now apply the configuration:

$ kubectl apply -f openfaas-figlet-dep.yaml,openfaas-figlet-svc.yaml

deployment.apps/openfaas-figlet created

service/openfaas-figlet created



**The cluster is up running with openFaaS service.**

**Deploying functions to the OpenFaaS**

We have selected python environment as a function, follow the below steps to create a “hello-python” function

Create a new folder for your work:

$ mkdir -p ~/functions && \

cd ~/functions

Now let's create a new Python function using the CLI:

$ faas-cli new --lang python3-armhf hello-python

This creates three files for you:

hello-python/handler.py

hello-python/requirements.txt

hello-python.yml

Now edit the handler.py file:

def handle(req):

print("The cluster is up and running python" + req)

Checkout the YAML file “hello-python.yml”:

provider:

name: faas

gateway: http://127.0.0.1:31112

functions:

hello-python:

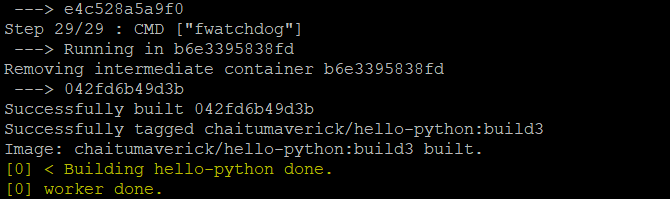
lang: python

handler: ./hello-python

image: chaitumaverick/hello-python:latest

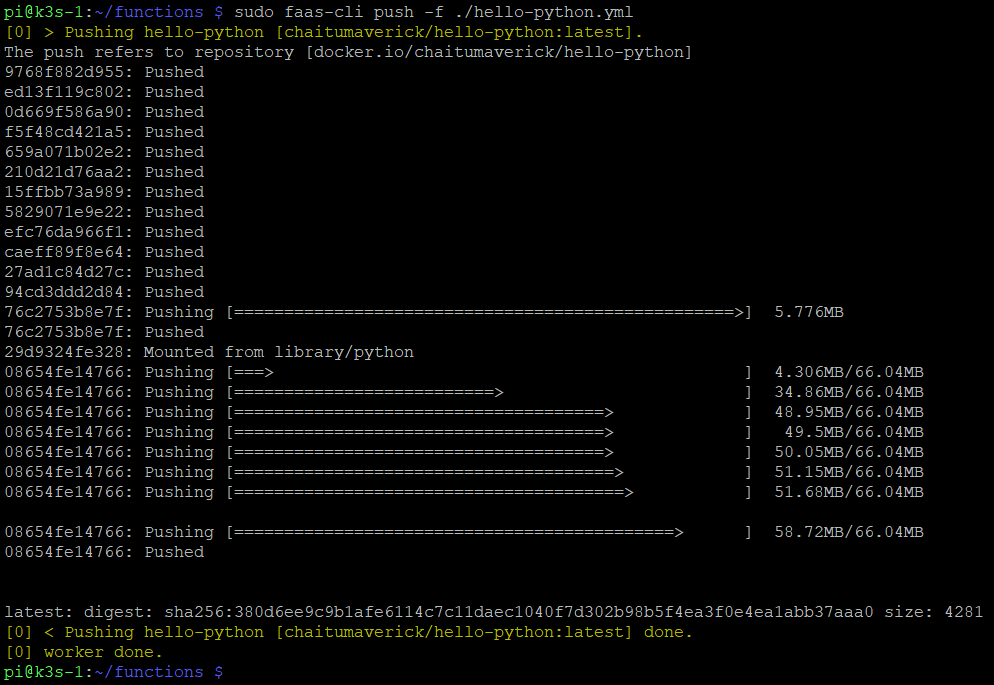
let's build the function.

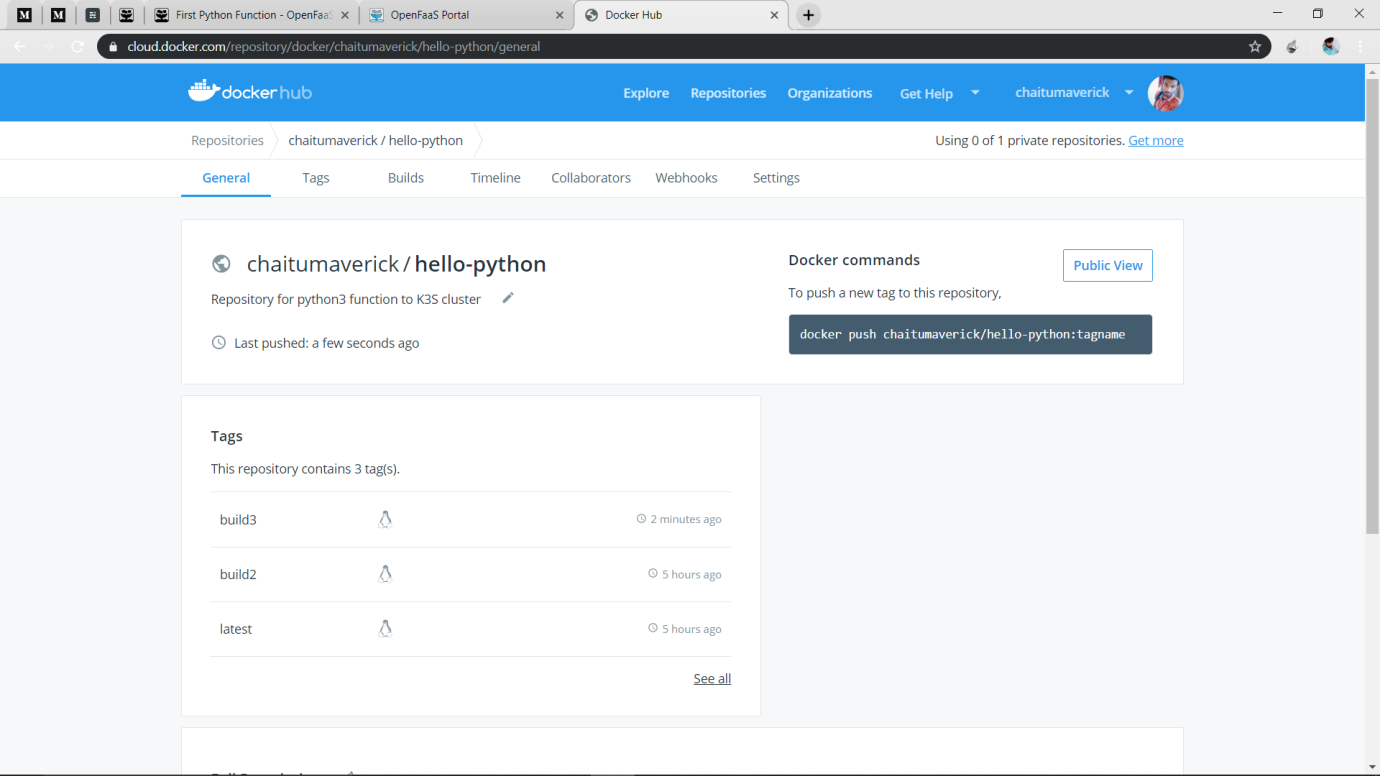
$ faas-cli build -f ./hello-python.yml



Now try to upload the function to a remote registry docker (if needed):

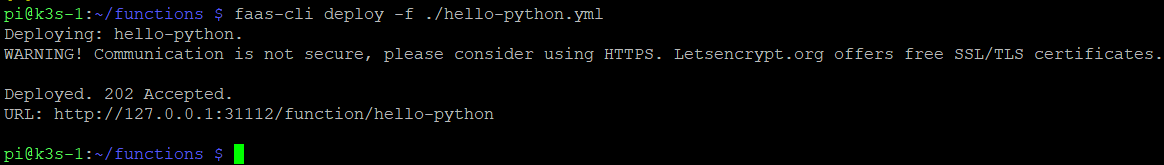
$ faas-cli push -f ./hello-python.yml

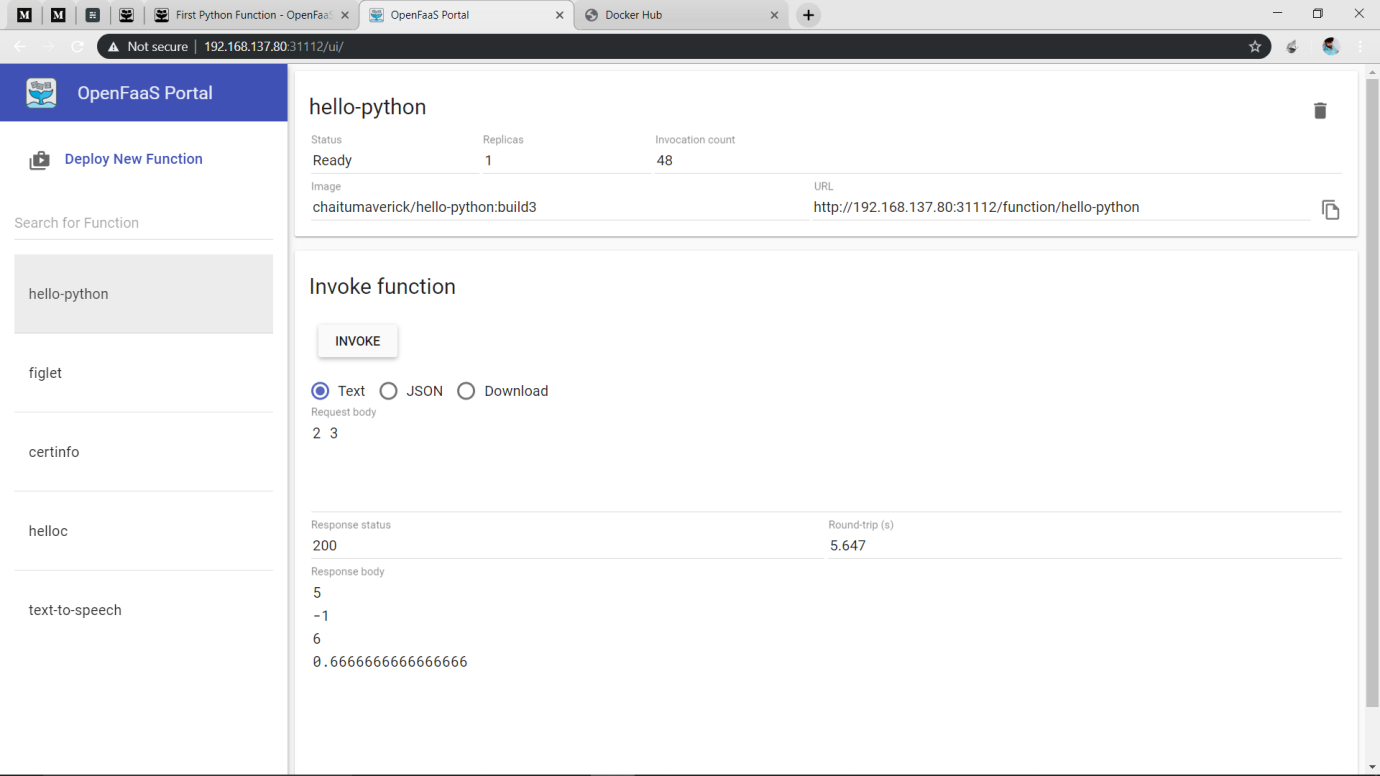




Now deploy the function:

$ faas-cli deploy -f ./hello-python.yml



**Now we can view the function in the browser as:**