

به نام خالق ابرها

Distributed Systems

Serverless FaaS Platform

© **Mosafer**

<https://github.com/mohsafer>

ابتدا نیازمندی های اصلی مورد نظر از جمله Docker و Minikube و faas-cli و سپس سرویس های نظیر dashboard, Gateway , prometheus را راه اندازی می نمایم

برای نصب داکر از اسکریپت استاندارد

```
$ curl -fsSL https://get.docker.com -o install-docker.sh
```

```
$ sudo sh install-docker.sh
```

نصب minikube

```
curl -LO
```

```
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

```
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

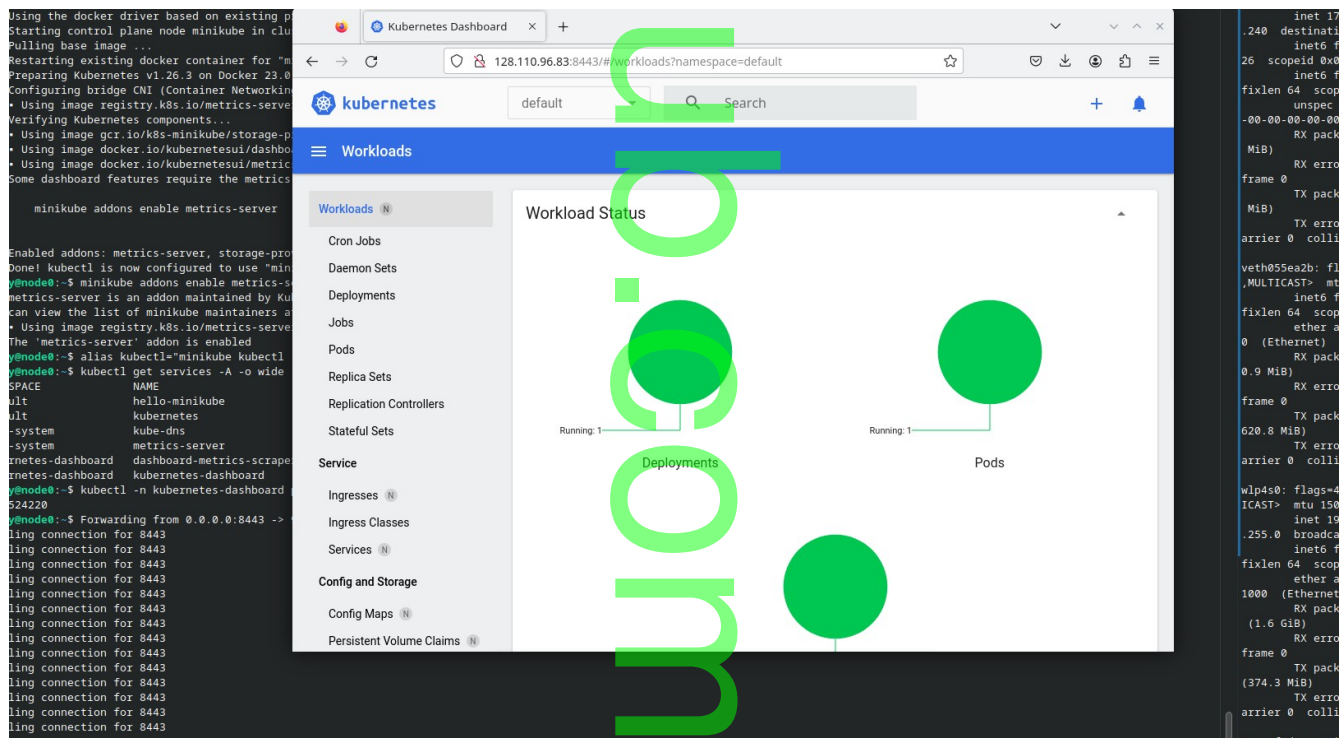
```
mosafer@node0:~$ kubectl get nodes -A -o wide
NAME          STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
minikube      Ready     control-plane  5h37m   v1.26.3   192.168.49.2   <none>        Ubuntu 20.04.5 LTS   5.15.0-69-generic   docker://23.0.2

mosafer@node0:~$ 
mosafer@node0:~$ 
mosafer@node0:~$ kubectl get services -A -o wide
NAMESPACE     NAME                                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE   SELECTOR
default       hello-minikube                     NodePort    10.100.220.234 <none>        8080:31779/TCP   5h34m app=hello-minikube
default       kubernetes                         ClusterIP    10.96.0.1     <none>        443/TCP          5h37m <none>
kube-system   kube-dns                          ClusterIP    10.96.0.10    <none>        53/UDP,53/TCP,9153/TCP 5h37m k8s-app=kube-dns
kube-system   kube-state-metrics                 ClusterIP    None          <none>        8080/TCP,8081/TCP   5h16m app.kubernetes.io/n
kube-system   metrics-server                     ClusterIP    10.111.203.84 <none>        443/TCP          5h22m k8s-app=metrics-ser
kubernetes-dashboard dashboard-metrics-scraper           ClusterIP    10.104.198.178 <none>        8000/TCP          5h25m k8s-app=dashboard-m
kubernetes-dashboard kubernetes-dashboard               ClusterIP    10.109.168.254 <none>        80/TCP           5h25m k8s-app=kubernetes-
monitoring     node-exporter                      ClusterIP    10.189.84.113 <none>        9100/TCP         4h45m app.kubernetes.io/c
monitoring     prometheus-service                 NodePort    10.103.245.121 <none>        8080:30000/TCP   4h45m app=prometheus-serv
openfaas       alertmanager                       ClusterIP    10.102.142.233 <none>        9093/TCP         4h13m app=alertmanager
openfaas       gateway                            ClusterIP    10.102.135.186 <none>        8080/TCP         4h13m app=gateway
openfaas       gateway-external                   NodePort    10.105.40.198   <none>        8080:31112/TCP   4h13m app=gateway
openfaas       nats                               ClusterIP    10.99.197.155   <none>        4222/TCP         4h13m app=nats
openfaas       prometheus                         ClusterIP    10.98.227.192   <none>        9090/TCP         4h13m app=prometheus
```

لازم به ذکر است چندین مرتبه برای نصب به صورت لوکال اقدام شد که به علت مشکلات و محدودیت IPها کلاستر کوبر روی فضای ابری CloudLab راه اندازی شد

برای مانیتور اجزای کلاستر داشبورد کلاستر به سمت اینترنت اکسپوز شد

```
kubectl -n kubernetes-dashboard port-forward --address 0.0.0.0
svc/kubernetes-dashboard 8443:80 &
```



جهت پیاده سازی OpenFaaS به minikube

```
fass-cli نصب
curl -sL cli.openfaas.com | sudo sh
```

```
kubectl apply -f https://raw.githubusercontent.com/openfaas/faas-netes/master/namespaces.yml
```

```
helm repo add openfaas https://openfaas.github.io/faas-netes/
```

```
export PASSWORD=$(head -c 12 /dev/urandom | shasum | cut -d' ' -f1)
echo $PASSWORD before
kubectl -n openfaas create secret generic basic-auth --from-literal=basic-auth-user=admin --from-literal=basic-auth-password="$PASSWORD"
```



```
--set functionNamespace=openfaas-fn > openfaas.yaml
```

```
kubectl apply -f namespaces.yaml,openfaas.yaml
```

```
faas-cli store deploy cows
kubectl get -n openfaas-fn service/cows -o yaml
kubectl get -n openfaas-fn deploy/cows -o yaml
```

فراخوانی فانکشن Cow

The screenshot displays the OpenFaaS Portal interface in a web browser. The main panel shows the 'COWS' function details, including its status (Ready), replicas (2), invocation count (3), image (ghcr.io/openfaas/cows:latest), and URL (http://128.110.96.36:31112/function/cows). Below this, the 'Invoke function' section is active, showing the 'Text' input method selected. The 'Request body' field is empty, and the 'Response status' is 200 with a round-trip time of 0.728s. The 'Response body' displays a ASCII art cow and the text 'Cow Slug'. An inset window shows the terminal output of the 'faas-cli store deploy cows' command, which includes the function's definition and the message 'Cow attempting to fly off a tree.'

```
mosafer@node0:~$ Handling connection for 31112
kubectl get -n openfaas-fn service/cows -o yaml
apiVersion: v1
kind: Service
```

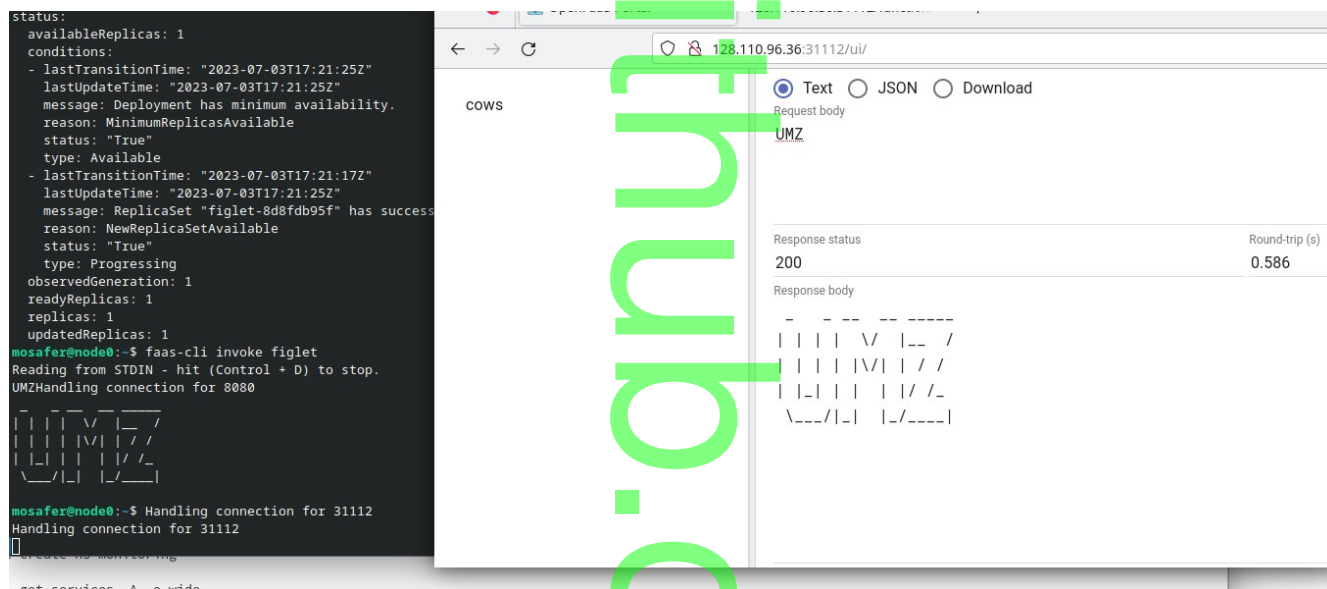
```
metadata:
  annotations:
    com.openfaas.git-repo-url: https://github.com/openfaas/store-
functions
  prometheus.io.scrape: "false"
  creationTimestamp: "2023-07-03T16:35:11Z"
  name: cows
  namespace: openfaas-fn
  resourceVersion: "19639"
  uid: 561a7038-626f-49d3-b78b-ea05d6771ac6
spec:
  clusterIP: 10.103.139.59
  clusterIPs:
    - 10.103.139.59
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  ports:
    - name: http
      port: 8080
      protocol: TCP
      targetPort: 8080
  selector:
    faas_function: cows
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
mosafer@node0:~$ Handling connection for 8443
Handling connection for 8443
Handling connection for 8443
Handling connection for 8443
Handling connection for 8443
Handling connection for 8443
^C
mosafer@node0:~$ kubectl get -n openfaas-fn deploy/cows -o yaml
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  annotations:
    com.openfaas.git-repo-url: https://github.com/openfaas/store-
functions
    deployment.kubernetes.io/revision: "1"
    prometheus.io.scrape: "false"
  creationTimestamp: "2023-07-03T16:35:11Z"
  generation: 1
  labels:
    faas_function: cows
  name: cows
  namespace: openfaas-fn
  resourceVersion: "19695"
  uid: cf2dc451-b497-428f-8220-bf1bc2f5a404
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      faas_function: cows
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      annotations:
        com.openfaas.git-repo-url: https://github.com/openfaas/store-
functions
        prometheus.io.scrape: "false"
      creationTimestamp: null
      labels:
        com.openfaas.git-branch: master
        com.openfaas.git-owner: openfaas
        com.openfaas.git-repo: store-functions
        com.openfaas.git-sha: f79e2c86e8d67f747d1e449ba6ca63eb5858e5bb
        com.openfaas.scale.min: "2"
```

```
    faas_function: cows
  name: cows
spec:
  containers:
  - image: ghcr.io/openfaas/cows:latest
    imagePullPolicy: Always
    livenessProbe:
      failureThreshold: 3
      httpGet:
        path: /_/health
        port: 8080
        scheme: HTTP
      initialDelaySeconds: 2
      periodSeconds: 2
      successThreshold: 1
      timeoutSeconds: 1
    name: cows
    ports:
    - containerPort: 8080
      name: http
      protocol: TCP
    readinessProbe:
      failureThreshold: 3
      httpGet:
        path: /_/health
        port: 8080
        scheme: HTTP
      initialDelaySeconds: 2
      periodSeconds: 2
      successThreshold: 1
      timeoutSeconds: 1
    resources: {}
    securityContext:
      readOnlyRootFilesystem: true
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /tmp
      name: temp
```



```
dnsPolicy: ClusterFirst
restartPolicy: Always
schedulerName: default-scheduler
securityContext: {}
terminationGracePeriodSeconds: 30
volumes:
- emptyDir: {}
  name: temp
status:
availableReplicas: 2
conditions:
- lastTransitionTime: "2023-07-03T16:35:30Z"
  lastUpdateTime: "2023-07-03T16:35:30Z"
  message: Deployment has minimum availability.
  reason: MinimumReplicasAvailable
  status: "True"
  type: Available
- lastTransitionTime: "2023-07-03T16:35:11Z"
  lastUpdateTime: "2023-07-03T16:35:30Z"
  message: ReplicaSet "cows-7575b9fb65" has successfully progressed.
  reason: NewReplicaSetAvailable
  status: "True"
  type: Progressing
observedGeneration: 1
readyReplicas: 2
replicas: 2
updatedReplicas: 2
```

```
curl -X POST http://128.110.96.36:31112/function/nslookup -d
'developer.cisco.com'
```

ساختن Costume Function در CLI

```
faas-cli template store pull python3
```

```
faas-cli new --lang python3 hello-python
```

```
mosafer@node0:~/workspace/OpenFaaS/hello-python$ cat handler.py
```

```
def handle(req):
    """handle a request to the function
    Args:
        req (str): request body
    """
    print("I am a great magician!, I think you said: " + req)
```

عملیات BUILD

```
faas-cli build -f hello-python.yml
```

```
#19 [stage-1 13/18] WORKDIR /home/app/function/
#19 DONE 1.0s

#20 [stage-1 14/18] COPY function/requirements.txt
#20 DONE 0.6s

#21 [stage-1 15/18] RUN pip install -r requirements.txt --target=/home/app/python
#21 DONE 3.6s

#22 [stage-1 16/18] WORKDIR /home/app/
#22 DONE 1.0s

#23 [stage-1 17/18] COPY function function
#23 DONE 0.6s

#24 [stage-1 18/18] RUN chown -R app:app ./ && chmod -R 777 /home/app/python
#24 DONE 1.4s

#25 exporting to image
#25 exporting layers
#25 exporting layers 2.7s done
#25 writing image sha256:ffc3e695cf02a13d3edc73adb08081a2b4151343eadf5fb82251d09fc7cc1284 0.0s done
#25 naming to localhost:5000/hello-python:latest 0.1s done
#25 DONE 2.8s
Image: localhost:5000/hello-python:latest built.
[0] < Building hello-python done in 39.18s.
[0] Worker done.

Total build time: 39.18s
mosafer@node0:~/workspace/OpenFaaS$
```

```
docker run -d -p 5000:5000 --restart=always --name registry
registry:2
```

عملیات PUSH, DEPLOY

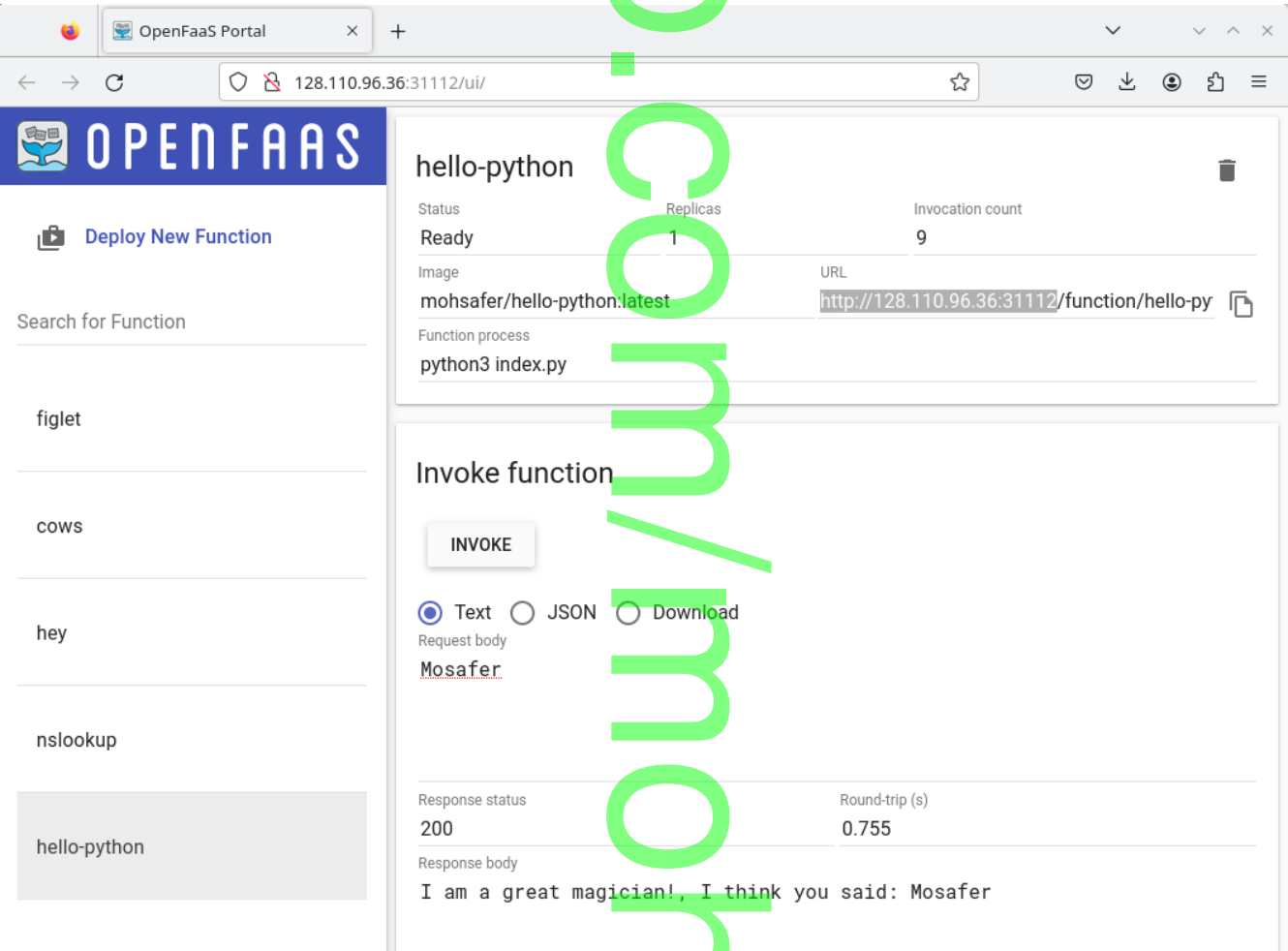
```
faas-cli push -f hello-python.yml
```

```
Digest: sha256:9977826e0d1d0eccc7af97017ae41f2dbe13f2c61e4c886ec28f0 added8c4078aa
Status: Downloaded newer image for registry:2
71296432d064ca486ea8274804b6a8a3e5d7815048ab9f4fdd4ae141d162ed89
mosafer@node0:~/workspace/OpenFaaS$ faas-cli push -f hello-python.yml
[0] > Pushing hello-python [localhost:5000/hello-python:latest]
The push refers to repository [localhost:5000/hello-python]
59118fac44b2: Pushed
59e93b1aa730: Pushed
5f70bf18a086: Pushed
0d0b98ca7899: Pushed
ad6967124b44: Pushed
fad44b659e9c: Pushed
f0e1279d0b2e: Pushed
d4312835e6f4: Pushed
dc69169959a3: Pushed
cd6a5ecb1144: Pushed
a7c7e1e89608: Pushed
ef6bcb11636a: Pushed
d1f4802bfc4b: Pushed
5eccfa029d2b: Pushed
e30a74f89cdc: Pushed
776df4582527: Pushed
42568fdeffd1: Pushed
9ec06db39a30: Pushed
cd18acbc1cce: Pushed
78a822fe2a2d: Pushed
latest: digest: sha256:58fe6c91630fb5d9b112ccf337ba942256b854ff054ae2bdec63a07e2a236668 size: 4900
[0] < Pushing hello-python [localhost:5000/hello-python:latest] done.
[0] Worker done.
mosafer@node0:~/workspace/OpenFaaS$
```

```
faas-cli deploy -f hello-python.yml
```

```
mosafer@node0:~/workspace/OpenFaaS$ cat hello-python.yml
version: 1.0
provider:
  name: openfaas
  gateway: http://127.0.0.1:8080
functions:
  hello-python:
    lang: python3
    handler: ./hello-python
    image: mosafer/hello-python:latest
```

```
curl -X POST http://128.110.96.36:31112/function/hello-python -d "Mosafer"
I am a great magician!, I think you said: Mosafer
[neo@fedora ~]$
```



```
mosafer@node0:~$ kubectl get services -A
NAMESPACE          NAME          TYPE          AGE
CLUSTER-IP          EXTERNAL-IP   PORT(S)

```

default	hello-minikube	NodePort
10.100.220.234	<none> 8080:31779/TCP	10h
default	kubernetes	ClusterIP
10.96.0.1	<none> 443/TCP	11h
kube-system	kube-dns	ClusterIP
10.96.0.10	<none> 53/UDP,53/TCP,9153/TCP	11h
kube-system	kube-state-metrics	ClusterIP
<none>	8080/TCP,8081/TCP	10h
kube-system	metrics-server	ClusterIP
10.111.203.84	<none> 443/TCP	10h
kubernetes-dashboard	dashboard-metrics-scraper	ClusterIP
10.104.198.178	<none> 8000/TCP	10h
kubernetes-dashboard	kubernetes-dashboard	ClusterIP
10.109.168.254	<none> 80/TCP	10h
monitoring	node-exporter	ClusterIP
10.109.84.113	<none> 9100/TCP	10h
monitoring	prometheus-service	NodePort
10.103.245.121	<none> 8080:30000/TCP	10h
openfaas-fn	cows	ClusterIP
10.103.139.59	<none> 8080/TCP	4h33m
openfaas-fn	figlet	ClusterIP
10.106.213.48	<none> 8080/TCP	3h47m
openfaas-fn	hello-python	ClusterIP
10.100.2.7	<none> 8080/TCP	28m
openfaas-fn	hey	ClusterIP
10.105.44.183	<none> 8080/TCP	4h6m
openfaas-fn	nslookup	ClusterIP
10.109.216.69	<none> 8080/TCP	3h38m
openfaas	alertmanager	ClusterIP
10.102.142.233	<none> 9093/TCP	9h
openfaas	gateway	ClusterIP
10.102.135.186	<none> 8080/TCP	9h
openfaas	gateway-external	NodePort
10.105.40.198	<none> 8080:31112/TCP	9h
openfaas	nats	ClusterIP
10.99.197.155	<none> 4222/TCP	9h
openfaas	prometheus	ClusterIP
10.98.227.192	<none> 9090/TCP	9h
openfaas	prometheus-ui	NodePort

10.97.159.22

<none>

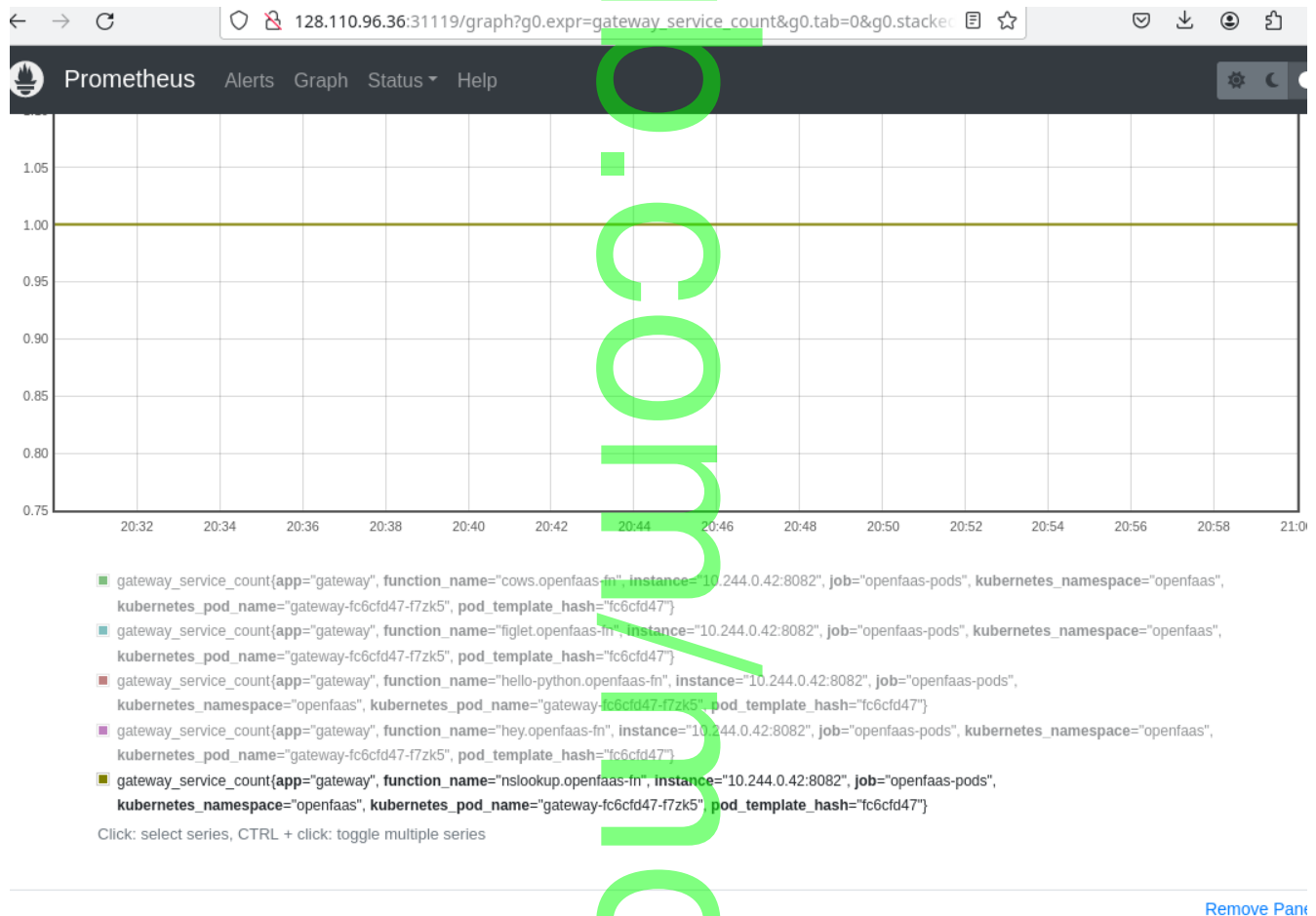
9090:31612/TCP

14m

اکسپوز پرومیتئوس

```
kubectl expose deployment prometheus -n openfaas --type=NodePort --name=prometheus-ui
```

جمع آوری داده ها از طریق مانیفست هایی نظیر service-metric.yaml



```
kubectl expose deployment prometheus -n openfaas --type=NodePort --name=prometheus-ui
```

```
minikube kubectl -- --namespace monitoring port-forward --address 0.0.0.0 svc/prometheus-service 3000:8080 &
```



```
kubectl port-forward -n openfaas svc/grafana 3000:3000 &
```

```
mosafer@node0:~/workspace/OpenFaaS/hello-python$ kubectl get deploy
```

```
-A
```

NAME	READY	UP-TO-DATE
hello-minikube	1/1	1
coredns	1/1	1
kube-state-metrics	1/1	1
metrics-server	1/1	1
dashboard-metrics-scraper	1/1	1
kubernetes-dashboard	1/1	1
prometheus-deployment	1/1	1
cows	2/2	2
figlet	1/1	1
hello-python	1/1	1
hey	1/1	1
nslookup	1/1	1
alertmanager	1/1	1
gateway	1/1	1
nats	1/1	1
prometheus	1/1	1

worker

ing: Alert rules - G: X [Explore -](#)

explore?left=("datasource":"Promet

_memory_usage_bytes) > 1

Time series Step: auto Type: Both

Inspector



01:30 01:35 01:40 0

Ns10

ت ولی نه چندان

18



The screenshot shows a network traffic analysis tool interface. At the top, there's a search bar with the text 'alert rules - G...'. Below it, a packet capture is displayed. The packet is of type 'NS' (Name Service) and contains the text 'ولی نه چند' (Veli ne chand). The packet size is 18 bytes. The interface also shows a timeline at the bottom with a play button and a progress bar.

The screenshot shows a network traffic analysis tool interface. At the top, there's a search bar with the text 'alert rules - G...'. Below it, a packet capture is displayed. The packet is of type 'NS' (Name Service) and contains the text 'ولی نه چند' (Veli ne chand). The packet size is 18 bytes. The interface also shows a timeline at the bottom with a play button and a progress bar.

The screenshot shows a network traffic analysis tool interface. At the top, there's a search bar with the text 'alert rules - G...'. Below it, a packet capture is displayed. The packet is of type 'NS' (Name Service) and contains the text 'ولی نه چند' (Veli ne chand). The packet size is 18 bytes. The interface also shows a timeline at the bottom with a play button and a progress bar.

[illegible]

0.059 [16]	■■■■■■■■■■
0.068 [8]	■■■■■
0.077 [10]	■■■■■■■
0.085 [1]	■
0.094 [4]	■■■
0.102 [1]	■

اما با بالا رفتن نرخ درخواست ها فانکشن از سرویس دهی خارج گردید

```
hey http://128.110.96.36:31112/function/nslookup -d
'developer.cisco.com' -m POST -n 10000
```

The screenshot shows the OpenFaaS dashboard in a web browser. The address bar shows the URL `128.110.96.36:30000`. The dashboard has a sidebar with the OpenFaaS logo and a list of functions: `nslookup`, `hello-python`, `figlet`, `cows`, and `hey`. The `nslookup` function is selected and its details are shown on the right.

nslookup function details:

Status	Replicas	Invocation count
Ready	1	1402

Image: `ghcr.io/openfaas/nslookup:latest` | URL: `http://128.110.96.36:31112/function/nslookup`

Invoke function:

INVOKE

☒ Text ☐ JSON ☐ Download

Request body

Response details:

Response status	Round-trip (s)
500	0.58

Response body:

```
Internal Server Error
exit status 123
```

