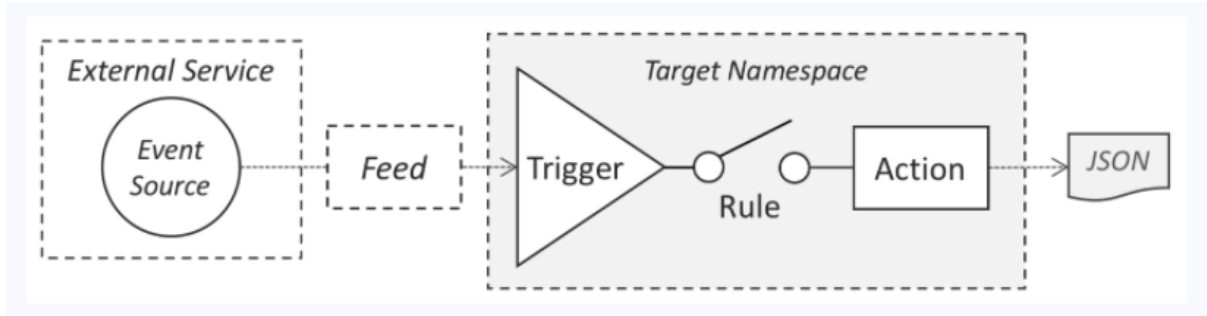


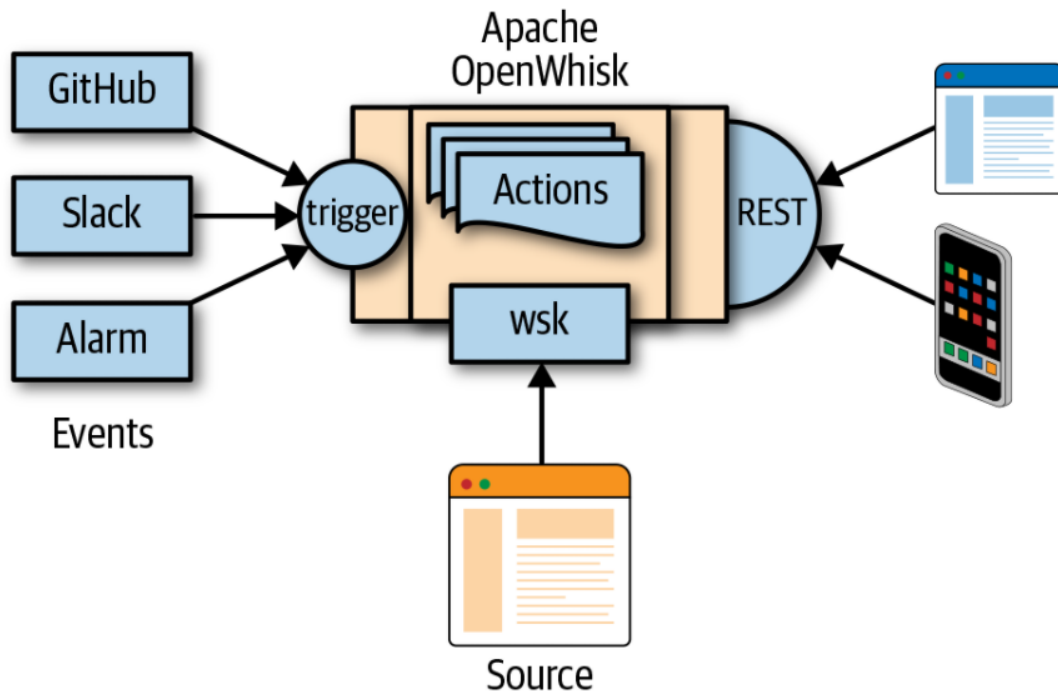
OpenWhisk Programlama Modeli



OpenWhisk'de olaylar(event) action adı verilen fonksiyonel yapıda serverless olarak çalışmayı sağlar. Eventler veri depoları(datastore), chatbotlar, mesaj kuyrukları(queue), mobil ya da web uygulama olabilir. Yani OpenWhisk event-driven'dir.

OpenWhisk herhangi bir programlama dilini destekler. Resmi olarak desteklenen diller ise: .Net, Go, Java, JavaScript, PHP, Python, Ruby, Swift'dir.

Resmi olarak desteklenmeyen diller ise Docker Runtime üzerinden çalıştırılabilir.



Actions

Action stateless fonksiyondur.(code snippets/kod parçacıkları) Yazılan fonksiyonun dili OpenWhisk üzerinde önemli değildir. OpenWhisk uygulamasının parçalar halinde oluşmasını sağlar.

Actionlar birbirleri ile bağlantılı olabilir, bir action'un outputu diğer bir action için input olabilir.

Namespacing: OpenWhisk varlıkları için adlandırma kurallarıdır.

Tetikleyiciler ve Kurallar(Triggers and Rules)

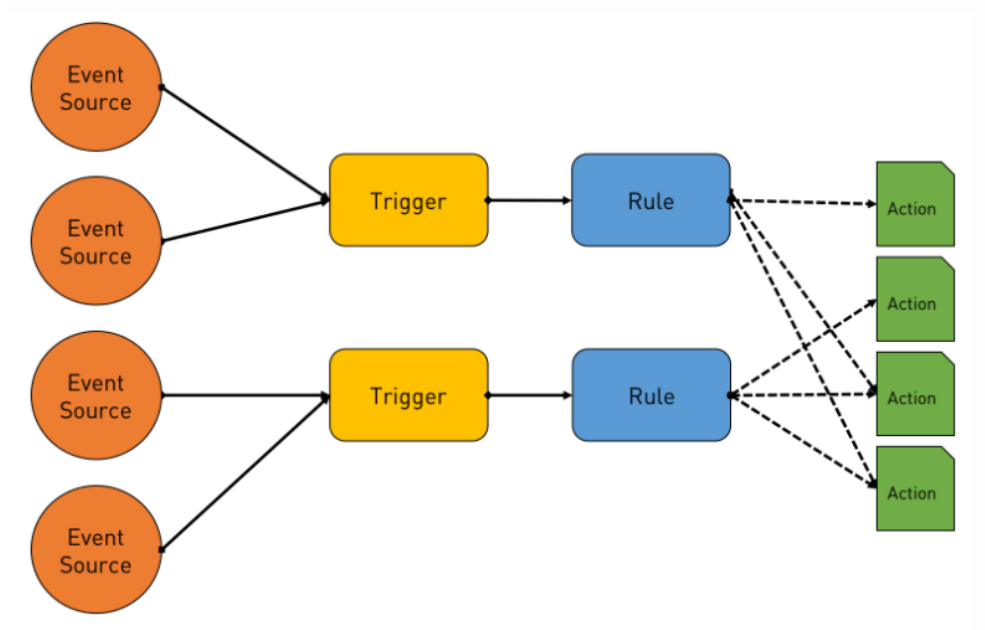
Trigger: Olay kaynaklarından(Event Sources) gönderilen sınıflar veya olay türleri için adlandırılmış kanallardır.

Triggerlar actionlara benzer ama birden çok actionu gerçekleştirmek için kurallar aracılığıyla kullanılır.

Rules: Trigger ve action arasında bağlantı kurar. Bir tetikleyici çağrıldığında ilgili actionlar çağrılır.

Event Sources: Verilerdeki değişimleri gösterir. Veritabanı değişimleri, cihaz sensörlerini ileten IoT frameworkleri ve web site etkileşimleri event sources'e örnektir.

OpenWhisk'de eylemler(action) olay kaynaklarına(event sources) bağlanmalıdır çünkü OpenWhisk programlama modeline göre otomasyonu sağlar ve parametler tetiklendikten sonra tanımın bozulmamasını sağlar.



OpenWhisk Geliştirme Ortamı

Local olarak ya da Bulut ortamında çalışabilir. Local olarak farklı deployment seçenekleri vardır. Bunlar Kubernetes, Docker Compose, Ansible, Vagrant ve Standalone direkt olarak kurulumdur. En uygun ortamı ise Docker'da etkinleştirilmiş Kubernetes de kurulumudur. Bulut ortamında ise IBM Cloud Functions ile kullanılmaktadır.

OpenWhisk Standalone Kurulum

OpenWhisk kurulumu için nodejs+npm, java, docker bilgisayarda kurulu olmalıdır.

-nodejs kurulumu

```
sudo apt install nodejs
```

```
elif@ubuntu:~$ sudo apt install nodejs
[sudo] password for elif:
Setting up npm (6.14.4+ds-1ubuntu2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
```

kurulumun kontrolü:

```
node --version
```

```
elif@ubuntu:~$ node --version  
v10.19.0
```

-npm kurulumu

```
sudo apt install npm
```

```
elif@ubuntu:~$ sudo apt install npm  
Reading package lists... Done
```

kurulumun kontrolü:

```
npm -v
```

```
Processing triggers for  
elif@ubuntu:~$ npm -v  
6.14.4
```

-java kurulumu

```
sudo apt install default-jdk
```

```
elif@ubuntu:~$ sudo apt install default-jdk  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done
```

```
Setting up libxcb1-dev:amd64 (1.14-2) ...  
Setting up libx11-dev:amd64 (2:1.6.9-2ubuntu1.2) ...  
Setting up default-jdk (2:1.11-72) ...  
Setting up libxt-dev:amd64 (1:1.1.5-1) ...  
elif@ubuntu:~$
```

kurulumun kontrolü:

```
java --version
```

```
Setting up libxt-dev:amd64 (1:1.1.5-1) ...  
elif@ubuntu:~$ java --version  
openjdk 11.0.11 2021-04-20  
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)  
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)  
elif@ubuntu:~$
```

-docker kurulumun kontrolü:

```
docker -v
```

```
elif@ubuntu:~$ docker -v  
Docker version 20.10.8, build 3967b7d
```

-openwhisk kurulumu

```
git clone https://github.com/apache/openwhisk.git
```

```
cd openwhisk
```

```
./gradlew core:standalone:bootRun
```

```
elif@ubuntu:~$ git clone https://github.com/apache/openwhisk.git
Cloning into 'openwhisk'...
remote: Enumerating objects: 53175, done.
remote: Counting objects: 100% (1145/1145), done.
remote: Compressing objects: 100% (548/548), done.
remote: Total 53175 (delta 399), reused 869 (delta 287), pack-reused 52030
Receiving objects: 100% (53175/53175), 66.91 MiB | 3.55 MiB/s, done.
Resolving deltas: 100% (28472/28472), done.
elif@ubuntu:~$ cd openwhisk
elif@ubuntu:~/openwhisk$ ./gradlew core:standalone:bootRun
Downloading https://services.gradle.org/distributions/gradle-5.5.1-all.zip
```

```
> Task :core:controller:compileScala
Pruning sources from previous analysis, due to incompatible CompileSetup.

> Task :tools:admin:compileScala
Pruning sources from previous analysis, due to incompatible CompileSetup.

> Task :core:standalone:compileScala
Pruning sources from previous analysis, due to incompatible CompileSetup.

> Task :core:standalone:bootRun
```

Git Commit: cf36299, Build Date: 2021-08-30T04:09:21+0300

Running pre flight checks ...

Unable to find image 'alpine:latest' locally

```
latest: Pulling from library/alpine
```

```
a0d0a0d46f8b: Pulling fs layer
```

a0d0a0d46f8b: Verifying Checksum

```
a0d0a0d46f8b: Download complete
```

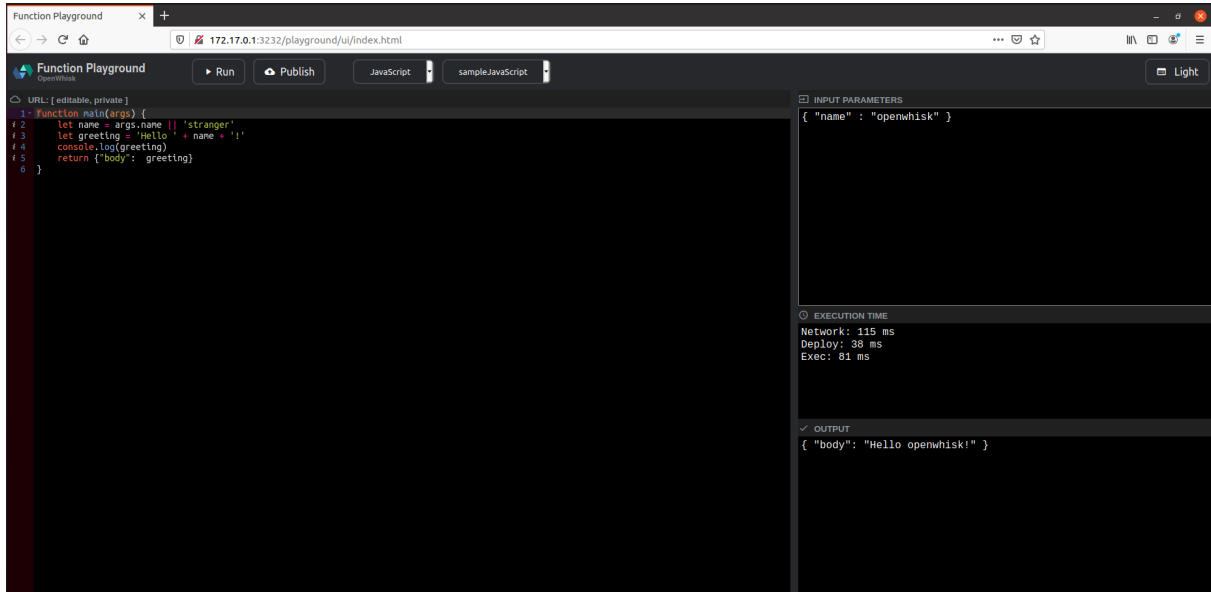
```
a0d0a0d46f8b: Pull complete
```

0' 1' 1' 056 1 000 010 15 000010 05011 070 051100 0110 05 001 01 001 0

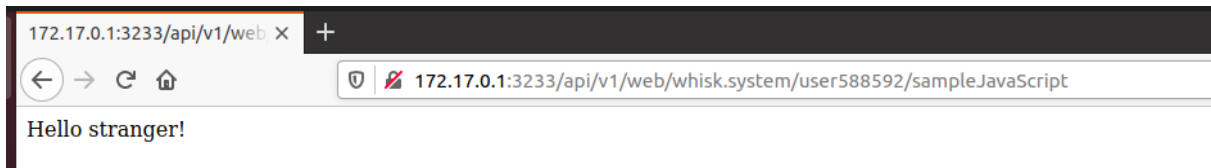
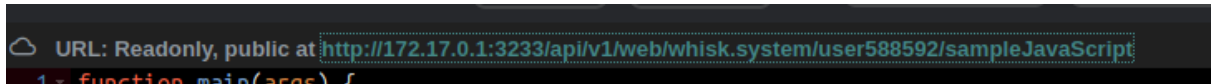
```
[ OK ] 'docker' cli found. (Docker version 20.10.8, build 3967b7d)
[ OK ] 'docker' version 20.10.8 is newer than minimum supported 18.3.0
[ OK ] 'docker' is running.[1m 19s]
[ OK ] 'wsk' cli found. (2021-04-01T23:49:54.523+0000)
[ WARN ] Configure wsk via below command to connect to this server as [quest]
```

-OpenWhisk arayüzünün açılması

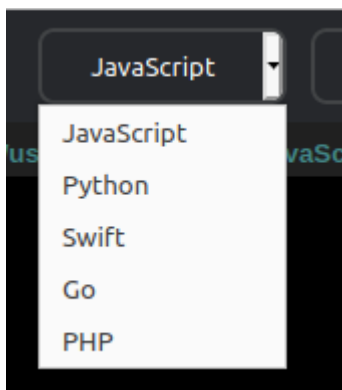
kurulum yapıldıktan sonra terminale `./gradlew core:standalone:bootRun` yazıldığında <http://172.17.0.1:3232> üzerinden arayüz açılır.



-fonksiyonu yayınlamak

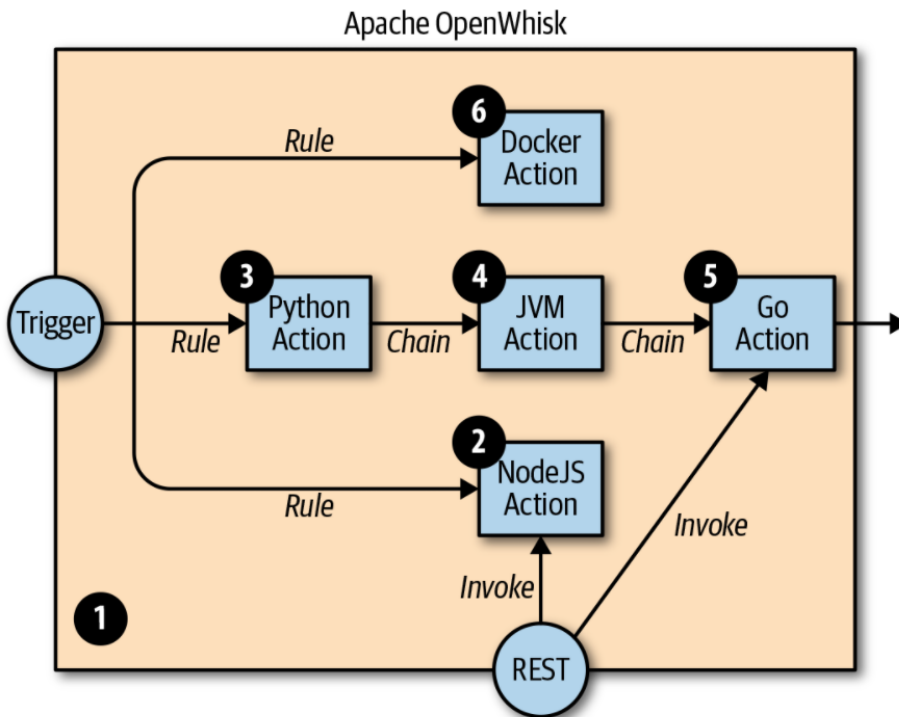


Arayüzde desteklenen diller:



-OpenWhisk CLI(wsk)

OpenWhisk varlıklarını oluşturmayı, çalıştırmayı ve yönetmeyi sağlar.




Kurulum: <https://github.com/apache/openwhisk-cli/releases> işletim sistemine uygun dağıtım seçilir ve çalıştırılabilir olan “wsk” dosyadan çıkartılır.

wsk'nin kolayca çalıştırılabilmesi için wsk'nin bulunduğu dosya PATH değişkenlerinin olduğu bir konuma taşınır. Böylece komut satırına wsk yazılarak erişilebilir hale gelir.

```
sudo mv <wsk'nin dosya konumu> <path konumu>
```

```
elif@ubuntu: ~/Desktop$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

```
elif@ubuntu:~/Desktop$ ls
wsk
elif@ubuntu:~/Desktop$ sudo mv wsk /usr/local/bin
elif@ubuntu:~/Desktop$ wsk --help
```


OpenStax

Usage:
wsk [command]

```
Available Commands:
  action    work with actions
  activation work with activations
  api       work with APIs
  help      Help about any command
  list      list entities in the current namespace
  namespace work with namespaces
  package   work with packages
  project   The OpenWhisk Project Management Tool
  property  work with whisk properties
  rule      work with rules
  sdk       work with the sdk
  trigger   work with triggers
```

-Konfigürasyon

wsk kullanabilmek API endpoint'i belirtmek ve kimlik doğrulama yapmak gereklidir.

<https://github.com/apache/openwhisk/blob/master/ansible/files/auth.guest>

```
wsk property set --apihost 'http://172.17.0.1:3233' --auth  
'23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123z03xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP'
```

```
.33.13:443: Connect: connection refused  
elif@ubuntu:/usr/local/bin$ wsk property set --apihost 'http://172.17.0.1:3233' --auth '23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123z03xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP'  
ok: whisk auth set. Run 'wsk property get --auth' to see the new value.  
ok: whisk API host set to http://172.17.0.1:3233
```

```
cat ~/.wskprops
```

```
elif@ubuntu:/usr/local/bin$ cat ~/.wskprops  
NAMESPACE=guest  
AUTH=23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123z03xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP  
APIHOST=http://172.17.0.1:3233
```

-özelliklerin doğrulanması

```
wsk property get
```

```
elif@ubuntu:/usr/local/bin$ wsk property get  
whisk API host      http://172.17.0.1:3233  
whisk auth          23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123z03xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP  
whisk namespace     guest  
client cert  
Client key  
whisk API version    v1  
whisk CLI version    2021-04-01T23:49:54.523+0000  
whisk API build      2021-08-30T01:09:21+0000  
whisk API build number cf36299
```

-host bağlantısının doğrulanması

```
wsk list -v
```

```
elif@ubuntu:/usr/local/bin$ wsk list -v  
REQUEST:  
[GET] http://172.17.0.1:3233/api/v1/namespaces/_/actions?limit=0&skip=0  
Req Headers  
{  
  "Authorization": [  
    "Basic MjNiYzQ2YjEtNzFmNi00ZWQ1LTJhNTQ0ODE2YWE0ZjhjNTAyOjEyM3pPM3haQ0xyTU42dJJC50sxZFhZRnBYbFBrY2NPRnFtMTJDZEFzTWdSVTRWck5aOWx5R1ZDR3VNREdJd1A="",  
  ],  
  "User-Agent": [  
    "OpenWhisk-CLI/1.0 (2021-04-01T23:49:54.523+0000) linux amd64"  
  ]  
}
```

OpenWhisk CLI'yi kullanmak

Actionları üretmek ve çağırmak

Bir action bir resimden yüzü tespit etmek ya da bir API çağrısına yanıt vermek gibi çeşitli amaçlarla kullanılabilir.

Nodejs Kullanımı

```
wsk action create helloJS hello.js
```

```
elif@ubuntu:~/Desktop$ wsk action create helloJS hello.js  
ok: created action helloJS  
elif@ubuntu:~/Desktop$
```

```
wsk action invoke helloJS --result --param name World
```

```
elif@ubuntu:~/Desktop$ wsk action invoke helloJS --result --param name World
{
  "greeting": "Hello World!"
}
```

inputlar key value şeklinde girilir.

```
elif@ubuntu:~/Desktop$ cat manifest.yml
packages:
  default:
    actions:
      helloJS:
        function: hello.js
```

-action deploy edilmesi:

```
wskdeploy -m manifest.yml
```

```
elif@ubuntu:~/Desktop$ wskdeploy -m manifest.yml
Success: Deployment completed successfully.
```

Go ile Kullanımı

```
cat hello.go
```

```
elif@ubuntu:~/Desktop$ cat hello.go
package main

import "fmt"

// Main function for the action
func Main(obj map[string]interface{}) map[string]interface{} {
    name, ok := obj["name"].(string)
    if !ok {
        name = "stranger"
    }
    fmt.Printf("name=%s\n", name)
    msg := make(map[string]interface{})
    msg["msg"] = "Hello, " + name + "!"
    return msg
}
```

```
wsk action create helloGo hello.go
```

```
elif@ubuntu:~/Desktop$ wsk action create helloGo hello.go
ok: created action helloGo
```



```
wsk action invoke helloGo --result --param name gopher
```

```
elif@ubuntu:~/Desktop$ wsk action invoke helloGo --result --param name gopher
{
  "msg": "Hello, gopher!"
}
```

```
GNU nano 4.8 manifest.yml
packages:
  default:
    actions:
      helloGo:
        function: hello.go
```

Trigger

```
wsk trigger create locationUpdate
```

```
elif@ubuntu:~$ wsk trigger create locationUpdate
ok: created trigger locationUpdate
elif@ubuntu:~$ wsk trigger list
triggers
/guest/locationUpdate private
elif@ubuntu:~$ wsk trigger fire locationUpdate --param name Bob --param place NYC
ok: triggered /locationUpdate with id
```

```
wsk action create hello hello.js
```

```
wsk action invoke --result hello --param name Bernard --param place Black
```

```
elif@ubuntu:~/Desktop$ wsk action create hello hello.js
ok: created action hello
elif@ubuntu:~/Desktop$ wsk action invoke --result hello --param name Bernard --param place Black
{
  "payload": "Hello, Bernard from Black"
}
```

```
elif@ubuntu:~/Desktop$ cat hello.js
function main(params) {
  return {payload: 'Hello, ' + params.name + ' from ' + params.place};
}
```

```
wsk rule create rule-whisk locationUpdate hello
```

```
elif@ubuntu:~/Desktop$ wsk rule create rule-whisk locationUpdate hello
ok: created rule rule-whisk
```

```
wsk trigger fire locationUpdate --param name Bob --param place NYC
```

```
elif@ubuntu:~/Desktop$ wsk trigger fire locationUpdate --param name Bob --param place NYC
ok: triggered /locationUpdate with id cf2f30210fae4356af30210faef356be
```

```
wsk activation list
```

```
2021-09-07 14:41:09 12c9978d6813482389978d6813182394 nodejs:14 cold 24ms success guest/hello:0.0.1
elif@ubuntu:~/Desktop$ wsk activation list
Datetime      Activation ID      Kind      Start Duration Status Entity
2021-09-07 14:42:13 cf2f30210fae4356af30210faef356be unknown warm 0s success guest/locationUpdate:0.0.1
2021-09-07 14:41:09 12c9978d6813482389978d6813182394 nodejs:14 cold 24ms success guest/hello:0.0.1
2021-09-07 13:54:15 446c1d4b39864deec1d4b39864deec3f go:1.15 cold 3.319s success guest/helloGo:0.0.1
```

```
wsk activation result cf2f30210fae4356af30210faef356be
```

```
elif@ubuntu:~/Desktop$ wsk activation result cf2f30210fae4356af30210faef356be
{
  "name": "Bob",
  "place": "NYC"
}
```

Whisk Deploy(wskdeploy)

OpenWhisk paketlerini, actionları, tetikleyicileri, kuralları ve API'leri bir komutla dağıtmaya yardımcı olur. wsk cli içinde alt komut olarak görülür. Standalone kurulum ile birlikte kullanılabilir.

Kurulum

wsk cli gibidir

<https://github.com/apache/openwhisk-wskdeploy/releases>

```
Usage:
  wskdeploy [flags]
  wskdeploy [command]

Available Commands:
  export      Export project assets from OpenWhisk
  help        Help about any command
  report      Provides a summary report of OpenWhisk assets being deployed/undeployed based on manifest/deployment YAML.
  sync        A tool to sync your OpenWhisk packages between client and server.
  undeploy    Undeploy OpenWhisk assets from server
  version     Print the version number of wskdeploy

Flags:
  --apihost HOST           whisk API HOST
  --apiversion VERSION     whisk API VERSION
  -u, --auth KEY           authorization KEY
  -c, --cert string        path of the .cert file
  --config string          config file (default is $HOME/.wskprops)
  -d, --deployment string  path to deployment file
  -h, --help               help for wskdeploy
  -k, --key string         path of the .key file
  --managed               allow project entities to be marked managed
  -m, --manifest string    path to manifest file
  -n, --namespace string   namespace
  --param KEY VALUE        parameter values in KEY VALUE format
  -P, --param-file FILE    FILE containing parameter values in JSON format
  --preview               show deployment/undeployment plan
  -p, --project string     path to serverless project (default ".")
  --projectname string     project name
  -s, --strict             allow user defined runtime version
  -v, --verbose            verbose output

Use "wskdeploy [command] --help" for more information about a command.
elif@ubuntu:~$
```

doğrulama:

```
Aunthentication:
wskdeploy reads credentials from $HOME/.wskprops by default
Overwrite default config file on CLI:
$ wskdeploy --config <config file> -m path/to/manifest.yaml
Or specify all three on CLI:
$ wskdeploy --apihost HOST --auth KEY --namespace NAMESPACE -m path/to/manifest.yaml

Usage:
```

OpenWhisk Entity İsimleri

entity URL yapısı “namespace/package/entity” şeklindedir.

package her entity için gerekli değildir. namespace bir kullanıcı gibidir.

namespace’in altında trigger, rule, package, action üretilebilir.

package’in altında ise trigger üretilemez ama diğer entityler üretilir.

OpenWhisk Package

Package: İlgili actionları bir araya getirir ve paylaşmayı sağlar. Ayrıca web uygulamaları tarafından kullanılacak URL(base URL) sağlarlar.

-paket oluşturmak

```
wsk package create sample -p email elif_sahingoz@hotmail.com
```

```
wsk API: built 2021-08-30T01:09:21+0000
elif@ubuntu:~/openwhisk$ wsk package create sample -p email elif_sahingoz@hotmail.com
ok: created package sample
elif@ubuntu:~/openwhisk$
```

-paketleri listelemek

```
wsk package list
```

```
elif@ubuntu:~/openwhisk$ wsk package list
packages
/guest/sample_whisk_v2      private
/guest/sample_whisk        private
/guest/sample               private
```

-paket bilgisini görüntülemek

```
wsk package get sample
```

```
elif@ubuntu:~/openwhisk$ wsk package get sample
ok: got package sample
{
  "namespace": "guest",
  "name": "sample",
  "version": "0.0.1",
  "publish": false,
  "parameters": [
    {
      "key": "email",
      "value": "elif_sahingoz@hotmail.com"
    }
  ],
  "binding": {},
  "updated": 1631264681710
}
```

-action oluşturmak

```
wsk action create basics/now now.js
```

```
elif@ubuntu:~/openwhisk$ wsk action create basics/now now.js
ok: created action basics/now
```

```
wsk action invoke basics/now
```

```
elif@ubuntu:~/openwhisk$ wsk action invoke basics/now
ok: invoked /_/basics/now with id 3034e1cd8c384aecb4e1cd8c38daec6a
```

OpenWhisk de actionlar default olarak asynchronous çalışır, bu yüzden fonkiyon çağrılmadan bir ID verilir.

-action çağırmak

iki yolu vardır

```
wsk activation result <id>
```

```
elif@ubuntu:~/openwhisk$ wsk activation result f9d0b80e44cf431c90b80e44cfd31cab
{
  "body": "Fri Sep 10 2021 09:15:13 GMT+0000 (Coordinated Universal Time)"
}
```

```
wsk action invoke basics/now -r
```

```
elif@ubuntu:~/openwhisk$ wsk action invoke basics/now -r
{
  "body": "Fri Sep 10 2021 09:18:29 GMT+0000 (Coordinated Universal Time)"
}
```

-web sayfası üzerinden göstermek

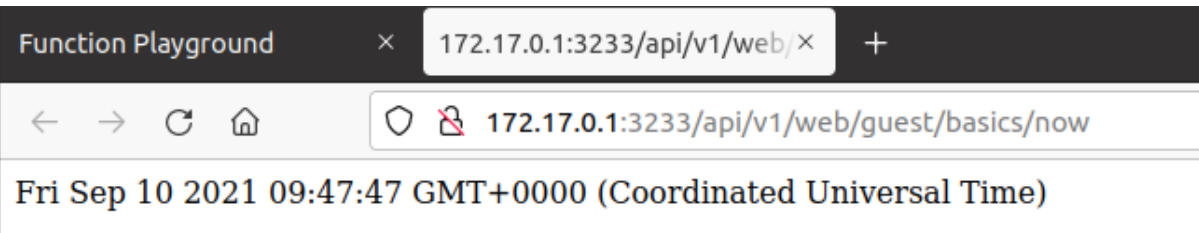
Not: web üzerinden girilmesi için ayrı bir flag kullanılır. --web true

```
wsk action update basics/now --web true
```

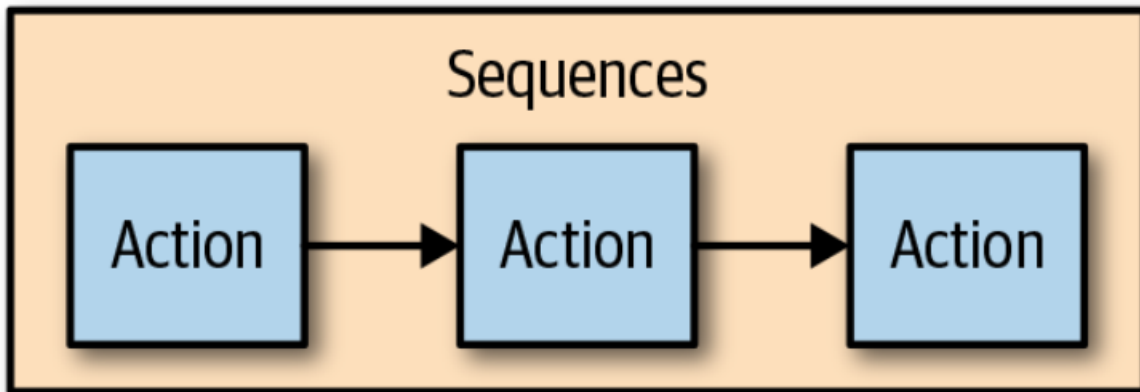
```
wsk action get basics/now --url
```

```
unknown flag: --web
elif@ubuntu:~/openwhisk$ wsk action update basics/now --web true
ok: updated action basics/now
elif@ubuntu:~/openwhisk$ wsk action get basics/now --url
ok: got action now
http://172.17.0.1:3233/api/v1/web/guest/basics/now
elif@ubuntu:~/openwhisk$
```

-web sayfası:



Actionların Zincirleme Dizileri(Chaining Sequences of Actions)



Cümleleri kelimelerine ayırıp sonra da sayan fonksiyonlar dizisinin birbirileri ile bağlanması örneği:

-split.js dosyası oluşturulur

```
function main(args) {  
  let words = args.text.split(' ')  
  return {  
    "words": words  
  }  
}
```

-action oluşturulması:

```
wsk action invoke basics/split \
```

```
-p text "openwhisk sequence example word split" -r \  
| tee save.json
```

```
elif@ubuntu:~/openwhisk$ wsk action invoke basics/split \  
> -p text "openwhisk sequence example word split" -r \  
> | tee save.json  
{  
  "words": [  
    "openwhisk",  
    "sequence",  
    "example",  
    "word",  
    "split"  
  ]  
}
```

count.js dosyası oluşturulur

```
function main(args) {  
  let words = args.words  
  let map = {}  
  let n = 0  
  for(word of words) {  
    n = map[word]  
    map[word] = n ? n+1 : 1  
  }  
  return map  
}
```

-json dosyasına göre çıktı alınır

```
wsk action update basics/count count.js
```

```
wsk action invoke basics/count -P save.json -r
```

```
elif@ubuntu:~/openwhisk$ nano count.js  
elif@ubuntu:~/openwhisk$ wsk action update basics/count count.js  
ok: updated action basics/count  
elif@ubuntu:~/openwhisk$ wsk action invoke basics/count -P save.json -r  
{  
  "example": 1,  
  "openwhisk": 1,  
  "sequence": 1,  
  "split": 1,  
  "word": 1  
}
```

-sequence oluşturmak

oluşturulan sequence ile json dosyasının kullanılmasına gerek kalmaz, split.js'in outputu count.js'in inputu olarak kullanılmıştır.

```
wsk action update basics/wordcount \
```

```
--sequence basics/split,basics/count
```

```
elif@ubuntu:~/openwhisk$ wsk action update basics/wordcount \  
> --sequence basics/split,basics/count  
ok: updated action basics/wordcount
```

```
wsk action invoke basics/wordcount -r -p text "can you can a can as a  
can as canner can can a can"
```

```
elif@ubuntu:~/openwhisk$ wsk action invoke basics/wordcount -r -p text "can you can a can as a cann  
er can can a can"  
{  
  "a": 3,  
  "as": 1,  
  "can": 6,  
  "canner": 1,  
  "you": 1  
}
```

OpenWhisk örnek

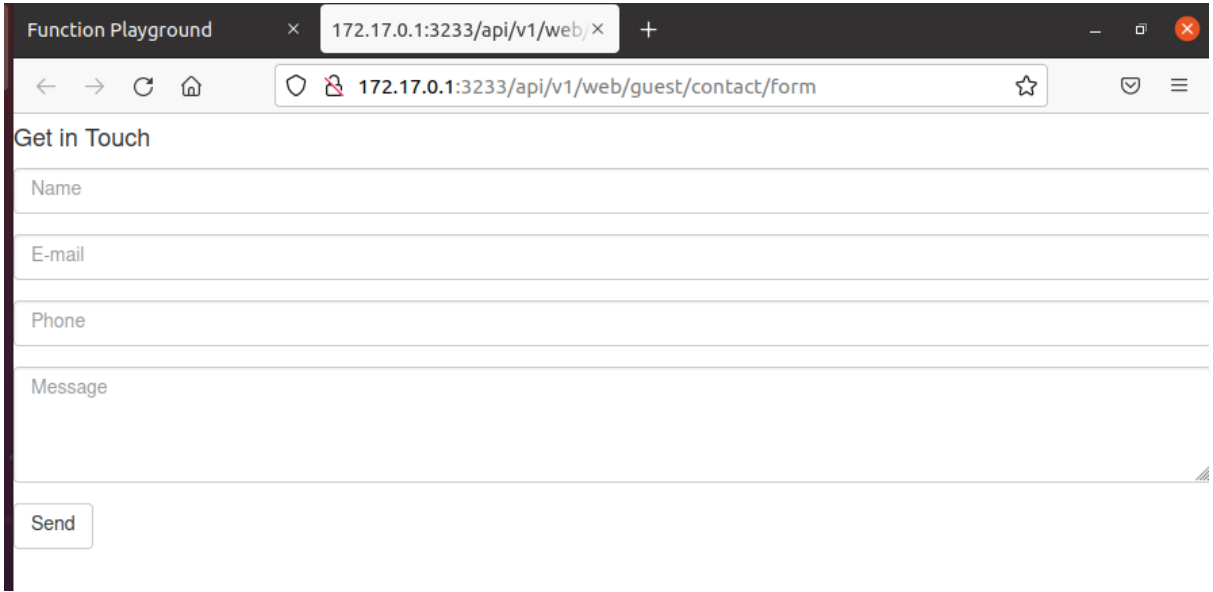
```
wsk action create contact/form action.js --web true
```

```
elif@ubuntu:~/openwhisk$ wsk action create contact/form action.js --web true
ok: created action contact/form
```

```
wsk action get contact/form --url
```

```
elif@ubuntu:~/openwhisk$ wsk action get contact/form --url
ok: got action form
http://172.17.0.1:3233/api/v1/web/guest/contact/form
```

-web sayfasının görüntülenmesi



The screenshot shows a web browser window with the title 'Function Playground'. The address bar displays '172.17.0.1:3233/api/v1/web/guest/contact/form'. The page content includes a form titled 'Get in Touch' with input fields for 'Name', 'E-mail', 'Phone', and a 'Message' text area. A 'Send' button is located at the bottom left of the form.

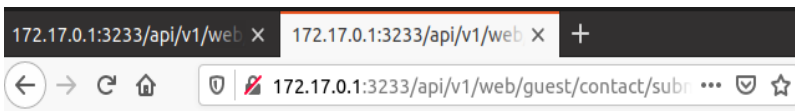
-submit için submit.js dosyası oluşturulur.

submit.js için:

<https://gist.githubusercontent.com/elifsz/a2ddc83a833b8f108ec78d3498829234/raw/25a1ac0bac9f9ed92cf4824becfcd30a44355965/submit.js>

```
elif@ubuntu:~/openwhisk$ wsk action create contact/submit submit.js --web true
ok: created action contact/submit
```

bilgiler eksik ya da yanlış girildiğinde:



The screenshot shows a web browser window with the title 'Function Playground'. The address bar displays '172.17.0.1:3233/api/v1/web/guest/contact/submit'. The page content includes a form titled 'Get in Touch' with input fields for 'Name', 'E-mail', 'Phone', and a 'Message' text area. A 'Send' button is located at the bottom left of the form. The form is currently empty, and the error message 'undefined' is visible at the top of the page.

Errors!

undefined

- No name provided
- Email missing or incorrect.
- Phone number missing or incorrect.

[Back](#)

bilgiler doğru girildiğinde:

Get in Touch

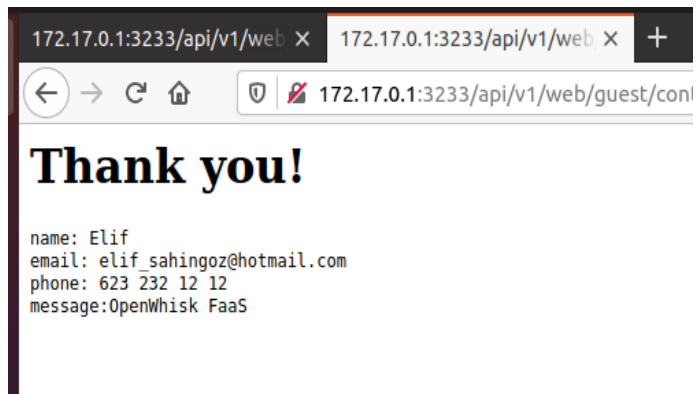
Elif

elif_sahingoz@hotmail.com

623 232 12 12

OpenWhisk FaaS

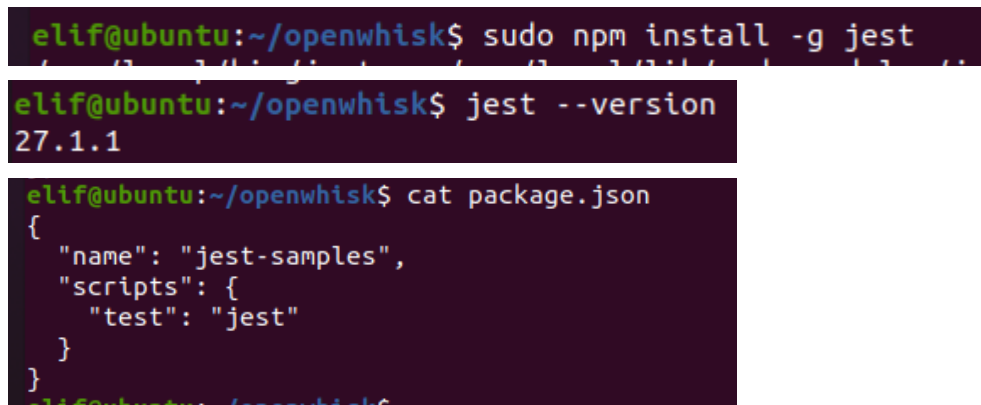
Send



OpenWhisk ve Unit Test

Unit test için jest tool'u kullanılacaktır.

```
sudo npm install -g jest
```




```
elif@ubuntu:~/openwhisk$ jest
PASS ./wordcount.test.js
  ✓ wordcount simple (47 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        2.635 s
Ran all test suites.
```

```
elif@ubuntu:~/openwhisk$ cat wordcount.js
function main(args) {
  let words = args.text.split(" ")
  let map = {}
  let n = 0
  for(word of words) {
    n = map[word]
    map[word] = n ? n+1 : 1
  }
  return map
}
module.exports.main = main
elif@ubuntu:~/openwhisk$ cat wordcount.test.js
const wordcount =
  require("./wordcount").main
test('wordcount simple', () => {
  res = wordcount({text: "a b a"})
  expect(res["a"]).toBe(2)
  expect(res["b"]).toBe(1)
})
elif@ubuntu:~/openwhisk$ cat package.json
{
  "name": "jest-samples",
  "scripts": {
    "test": "jest"
  }
}
```

OpenFaas vs OpenWhisk

-Resmi olarak desteklenen diller

OpenFaas: C#, Dockerfile, Go, Java8, Java, Node, Php, Python, Ruby, Rust, Powershell, Swift, lua53, Cobol, Perl'dür.

OpenWhisk: .Net, Go, Java, JavaScript, PHP, Python, Ruby, Swift'dir.

OpenWhisk'de docker sayesinde resmi olarak desteklenmeyen diller de yazılabilir.

-Github star sayıları

OpenFaaS: 20.3k

OpenWhisk: 5.4k

-StackOverflow'daki post sayıları

OpenFaas: 121

OpenWhisk: 551

-Katkıda bulunanlar

OpenFaas: 10

OpenWhisk: 33

-Kurumsal Destekçi

OpenFaas: VMWare

OpenWhisk: IBM(Cloud ortamında serverless fonksiyon yazma imkanı tanıyor)

-Yazıldığı programlama dili

OpenFaas: Go

OpenWhisk: Scala

-Her ikisi de docker container paket yönetiminde çalışmaktadır ve konfigürasyon için .yaml dosyası kullanırlar.

OpenFaas .yaml dosyası:

```
GNU nano 4.8
version: 1.0
provider:
  name: openfaas
  gateway: http://127.0.0.1:8080
functions:
  gohash:
    lang: go
    handler: ./gohash
    image: makinwavz/gohash
```

OpenWhisk .yaml dosyası:

```
GNU nano 4.8
packages:
  default:
    actions:
      helloGo:
        function: hello.go
```

OpenFaaS'da fonksiyonların çıktılarını görebilmek için hep .yaml dosyası kullanıldı ama OpenWhisk'de bu zorunlu değildir.

-Kullandığı Teknolojiler

OpenFaas: Alertmanager/Prometheus, Nats

OpenWhisk: CouchDB, Kafka, Nginx, Redis, Zookeeper

-Dokümantasyon

OpenFaas: Kendine ait dokümanı bulunmakta ve günceldir.

OpenWhisk: Kendine ait dokümanı vardır ama güncel değildir. Kurulum ya da konfigürasyon geliştiriciler için daha karmaşık olabilmektedir.

-CLI

OpenFaas: faas-cli

OpenWhisk: wsk. wsk'nin kurulumu faas-cli'ye daha manueeldir.

-Fonksiyon/action oluşturmak:

OpenFaas: `faas-cli new --lang go helloGo`

OpenWhisk: `wsk action create helloGo hello.go`

-Deploy etmek:

OpenFaas: `faas-cli build -f helloho.yml`

`faas-cli deploy -f helloho.yml`

ya da `faas-cli up -f helloho.yml` ile tek aşamada olabilir. Docker hesabına login olduğunda image oluşturur.

OpenWhisk: `wskdeploy -m manifest.yml`

-Invoke/fonksiyonun test edilmesi:

OpenFaas: `echo -n "test" | faas-cli invoke gohash`

OpenWhisk: `wsk action invoke helloGo --result --param name gopher` OpenWhisk'de input key value çifti olarak giriliyor.

-Başlatılması Süreleri

OpenFaas: log in olduğunda direkt başlamaktadır.

OpenWhisk: `./gradlew core:standalone:bootRun` komutu çalıştırıldıktan sonra başlamaktadır.

-Kurulum yapılabilen ortamlar

OpenWhisk IBM Cloud üzerinden kullanılabilirken OpenFaaS'da böyle bir durum yoktur.

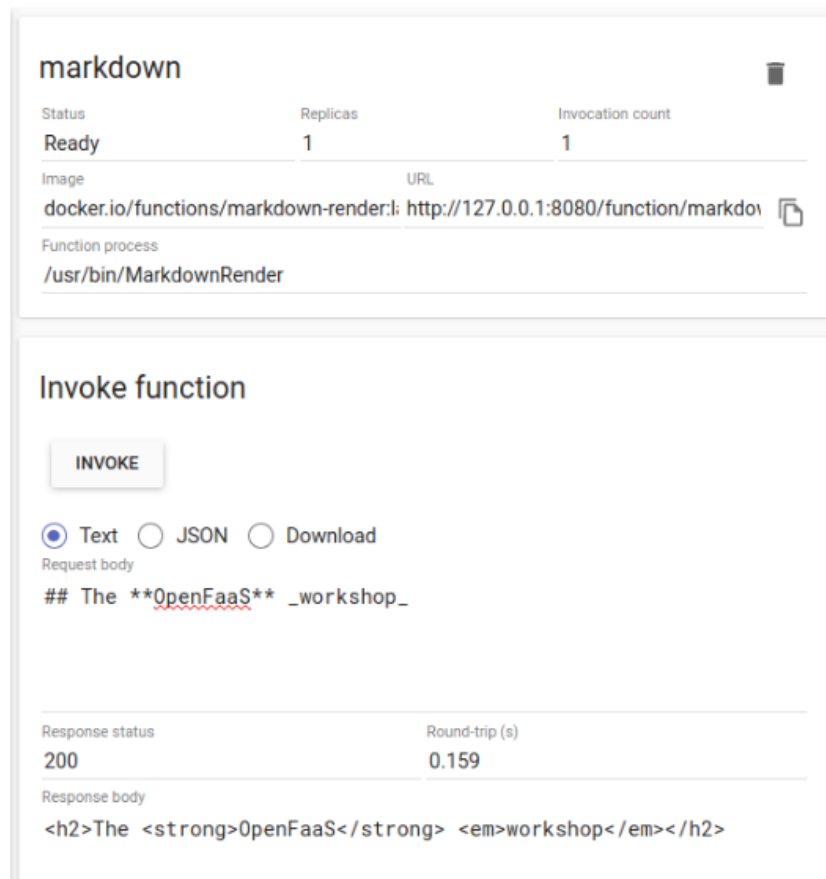
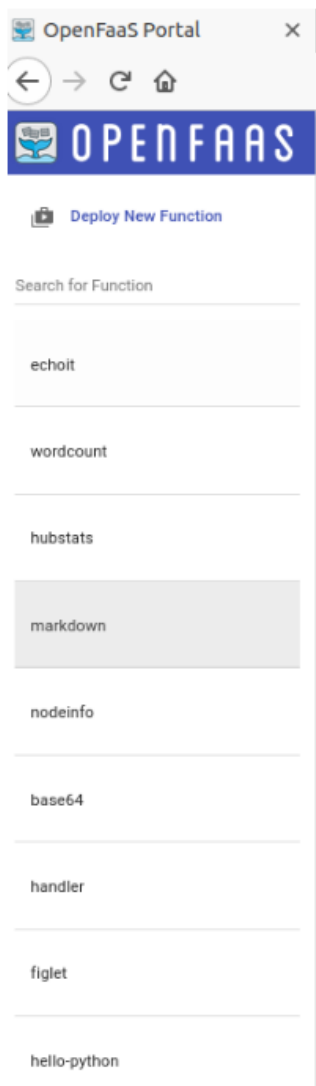
-Arayüz

OpenFaas: OpenWhisk'e göre fonksiyonlar daha kolay deploy edilir ve istatiksel olarak daha çok bilgi vermektedir.

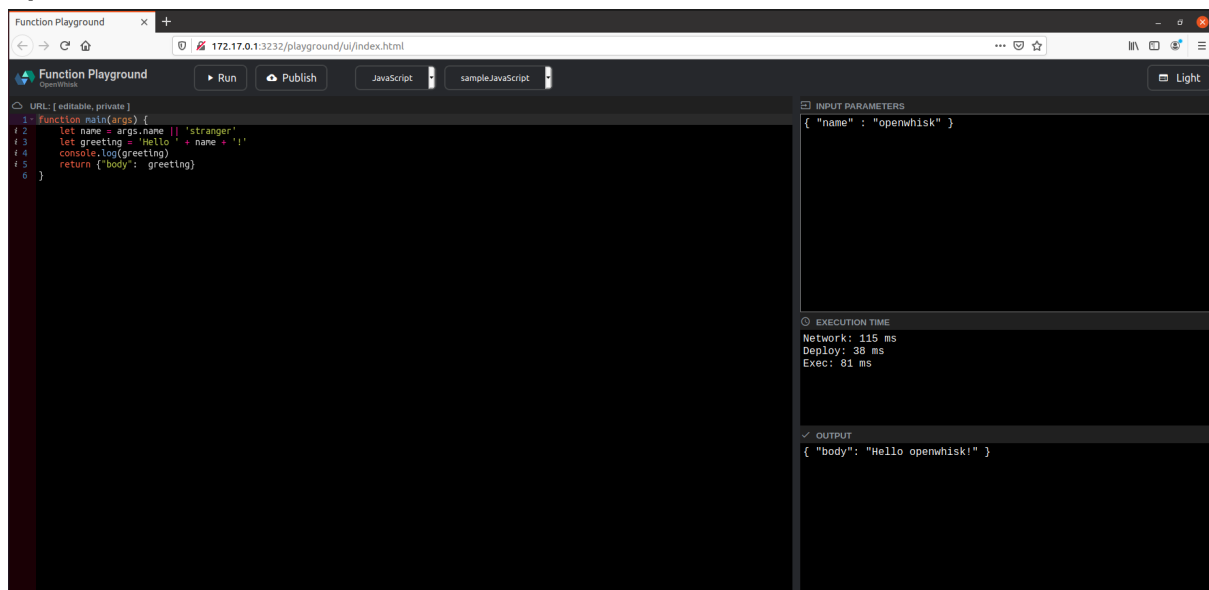
Kimlik doğrulaması OpenWhisk'e göre daha kolaydır.

Fonksiyonlara arayüzden erişebilmek için `--web true` flag'ı kullanılmalıdır. OpenFaaS'da bu işlemler otomatik olmaktadır.

OpenFaaS:



OpenWhisk:



Kaynakça

- “Documentation[online]”, <https://openwhisk.apache.org/documentation.html> [09/2021]
- Sciabarrà, Michele, "Learning Apache OpenWhisk", O'Reilly Media, Inc., July 2019
- Msv, JANAKIRAM, “An Architectural View of Apache OpenWhisk[online]”, <https://thenewstack.io/behind-scenes-apache-openwhisk-serverless-platform/> [09/2021]
- Butusov, MIKE, “Top 5 Serverless Platforms in 2021[online]”, <https://thenewstack.io/behind-scenes-apache-openwhisk-serverless-platform/> [09/2021]
- Nigam, RAJAT, “Apache OpenWhisk (Serverless for Kubernetes)[online]”, <https://faun.pub/apache-openwhisk-serverless-for-kubernetes-820f62534f24> [09/2021]
- “Getting started with IBM Cloud Functions[online]”, https://cloud.ibm.com/docs/openwhisk/openwhisk_webactions.html#openwhisk_webactions [09/2021]
- Glikson, ALEX, “Going ‘Serverless’ with OpenWhisk[online]”, <https://www.slideshare.net/AlexGlikson/going-serverless-with-openwhisk> [09/2021]