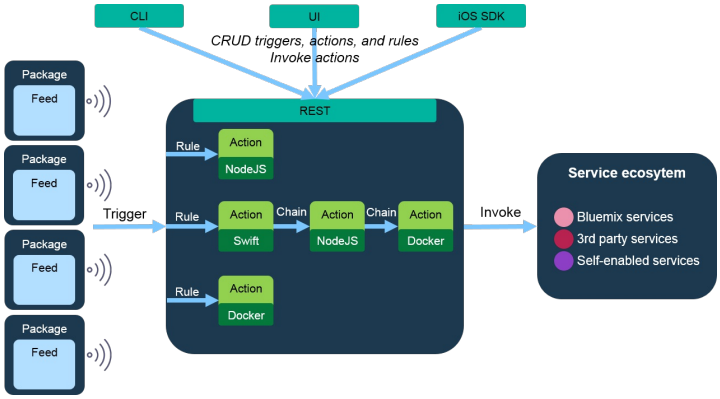
Microservices

Lab 9 – FaaS

Apache OpenWhisk is a serverless, open source cloud platform that executes functions in response to events at any scale. FaaS platforms allows users to optimize cost and outsource infrastructure and orchestration operations. Users instead focus on control and data flow, resource usage and other higher level concerns.

Components of OpenWhisk include:

- actions - stateless code snippets
- triggers - named channels for a class of events
- rules - to associate triggers with actions
- namespaces - to colocate resources
- packages - a way of organizing actions and feeds (cannot be nested)



Prerequisites

OpenWhisk uses Docker to isolate function execution, if you do not have Docker installed you will need to install it.

```
user@ubuntu:~$ wget -O - https://get.docker.com | sh

...

user@ubuntu:~$ sudo usermod -aG docker user

...

user@ubuntu:~$ reboot

...
```

OpenWhisk will try to communicate with Docker via TCP. By default dockerd listens on a domain socket: `/var/run/docker.sock`. We will need to update the SystemD configuration file for Docker so that it also listens on port 4243.

Modify the Docker systemd configuration in `/lib/systemd/system/docker.service` as follows.

First, we add the `-H tcp://0.0.0.0:4243` option to ExecStart. After editing it should look as follows, leave the remaining lines alone.

```
user@ubuntu:~$ sudo vim /lib/systemd/system/docker.service
user@ubuntu:~$ grep ExecStart /lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:4243

user@ubuntu:~$
```

Rebuild the system service dependency tree.

```
user@ubuntu:~$ sudo systemctl daemon-reload
user@ubuntu:~$
```

Cause Docker to use the new socket configuration.

```
user@ubuntu:~$ sudo systemctl restart docker
user@ubuntu:~$
```

Confirm that we can access the Docker Remote API via port 4243.

```
user@ubuntu:~$ docker -H 0.0.0.0:4243 version
Client:
 Version:      17.06.0-ce
 API version:  1.30
 Go version:   go1.8.3
 Git commit:   02c1d87
 Built:        Fri Jun 23 21:23:31 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.06.0-ce
 API version:  1.30 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   02c1d87
 Built:        Fri Jun 23 21:19:04 2017
 OS/Arch:      linux/amd64
 Experimental: false
user@ubuntu:~$
```

1. Install OpenWhisk

Not only is it early days for FaaS, its even earlier for OpenWhisk. We will use the latest source code to base our deployment on.

```
user@ubuntu:~$ git clone https://github.com/apache/incubator-openwhisk openwhisk
Cloning into 'openwhisk'...
remote: Counting objects: 22901, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 22901 (delta 0), reused 1 (delta 0), pack-reused 22898
Receiving objects: 100% (22901/22901), 54.64 MiB | 7.40 MiB/s, done.
Resolving deltas: 100% (12461/12461), done.
Checking connectivity... done.

user@ubuntu:~$ cd ~/openwhisk/

user@ubuntu:~/openwhisk$
```

Since we are using an Ubuntu VM, we will use the included build script for Ubuntu.

```
user@ubuntu:~/openwhisk$ cd tools/ubuntu-setup/
user@ubuntu:~/openwhisk/tools/ubuntu-setup$
```

Because we have already installed Docker, we need to comment the part where it is installed. We do this by editing `all.sh` in the directory `~/openwhisk/tools/ubuntu-setup/` .

Comment out `docker.sh` , leave all other settings alone. Currently, the built in script (`docker.sh`) does not work with our Ubuntu VM.

```
user@ubuntu:~/openwhisk/tools/ubuntu-setup$ vim all.sh

...
#echo "*** installing docker"
#u_release="$(lsb_release -rs)"
#if [ "${u_release%%.*}" -lt "16" ]; then
#    /bin/bash "$SCRIPTDIR/docker.sh"
#else
#    echo "--- WARNING ----------------------------------------------"
#    echo "Using EXPERIMENTAL Docker CE script on Xenial or later Ubuntu"
#    echo "--- WARNING ----------------------------------------------"
#    /bin/bash "$SCRIPTDIR/docker-xenial.sh"
#fi

...
```

Now we are set to build. This can take a while, grab a coffee!

```
user@ubuntu:~/openwhisk/tools/ubuntu-setup$ ./all.sh
...
user@ubuntu:~/openwhisk/tools/ubuntu-setup$
```

If all goes well you will return to the prompt with no errors. We are now ready to use OpenWhisk. We will use the cli tool called `wskdev` to set up our FaaS infrastructure.

Moving back to our project home directory.

```
user@ubuntu:~/openwhisk/tools/ubuntu-setup$ cd ~/openwhisk/
user@ubuntu:~/openwhisk$
```

2. Deploy OpenWhisk

We will launch the FaaS deployment via the `wskdev fresh` subcommand.

```
user@ubuntu:~/openwhisk$ sudo ./bin/wskdev fresh
...
user@ubuntu:~/openwhisk$
```

This will take 10-15 minutes. If all goes well, we will now be able to launch functions!

3. Using wsk

```
user@ubuntu:~/openwhisk$ ./bin/wsk -h


       ____      ___                   _   _ _     _     _
      /\   \    / _ \ _ __   ___ _ __ | |  | | |__ (_)___| | __
 /\  /__\   \  | | | | '_ \ / _ \ '_ \| |  | | '_ \| / __| |/ /
/  \____ \  /  | |_| | |_) |  __/ | | | |/\| | | | | \__ \   <
\   \  /  \/    \___/| .__/ \___|_| |_|__/\__|_| |_|_|___/_|\_\
 \___\/ tm           |_|

Usage:
  wsk [command]

Available Commands:
  action           work with actions
  activation       work with activations
  package          work with packages
  rule             work with rules
  trigger          work with triggers
  sdk              work with the sdk
  property         work with whisk properties
  namespace        work with namespaces
  list             list entities in the current namespace
  api-experimental work with APIs (experimental)
  api              work with APIs

Flags:
      --apihost HOST         whisk API HOST
      --apiversion VERSION   whisk API VERSION
  -u, --auth KEY             authorization KEY
      --cert string          client cert
  -d, --debug                debug level output
  -i, --insecure             bypass certificate checking
      --key string           client key
  -v, --verbose              verbose output

Use "wsk [command] --help" for more information about a command.

user@ubuntu:~/openwhisk$
```

```
user@ubuntu:~/openwhisk$ ./bin/wsk namespace list

error: The API host is not valid: An API host must be provided.
Run 'wsk --help' for usage.

user@ubuntu:~/openwhisk$
```

```
user@ubuntu:~/openwhisk$ ./bin/wsk property set --apihost 172.17.0.1

ok: whisk API host set to 172.17.0.1

user@ubuntu:~/openwhisk$
```

```
user@ubuntu:~/openwhisk$ ./bin/wsk namespace list

error: Unable to obtain the list of available namespaces: Unable to create HTTP request for GET: Unable to add the
```

```
HTTP authentication header: Authorization key is not configured (--auth is required)
Run 'wsk --help' for usage.

user@ubuntu:~/openwhisk$
```

```
user@ubuntu:~/openwhisk$ ./bin/wsk property set --auth $(cat ansible/files/auth.guest)

ok: whisk auth set. Run 'wsk property get --auth' to see the new value.

user@ubuntu:~/openwhisk$
```

```
user@ubuntu:~/openwhisk$ ./bin/wsk property get --auth

whisk auth              23bc46b1-71f6-4ed5-8c54-
816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP

user@ubuntu:~/openwhisk$
```

```
user@ubuntu:~/openwhisk$ ./bin/wsk namespace list

error: Unable to obtain the list of available namespaces: Get https://172.17.0.1/api/v1/namespaces: x509: cannot
validate certificate for 172.17.0.1 because it doesn't contain any IP SANs

user@ubuntu:~/openwhisk$
```

```
user@ubuntu:~/openwhisk$ ./bin/wsk -i namespace list

namespaces
guest

user@ubuntu:~/openwhisk$
```

```
user@ubuntu:~/openwhisk$ ./bin/wsk -i action invoke /whisk.system/utils/echo -p message hello --result

{
    "message": "hello"
}

user@ubuntu:~/openwhisk$
```

4. Hello World

. Create the code.

```
user@ubuntu:~/openwhisk$ vi action.js
user@ubuntu:~/openwhisk$ cat action.js

function main() {
    console.log('Hello World');
    return { hello: 'world' };
}

user@ubuntu:~/openwhisk$
```

. Create the action.

```
user@ubuntu:~/openwhisk$ ./bin/wsk -i action create myAction action.js

ok: created action myAction

user@ubuntu:~/openwhisk$
```

. Invoke the action.

```
user@ubuntu:~/openwhisk$ ./bin/wsk -i action invoke myAction

ok: invoked /_/myAction with id c541fcb8b7d0409183cccc754dd43316

user@ubuntu:~/openwhisk$
```

4. Retrieve the result.

```
user@ubuntu:~/openwhisk$ ./bin/wsk -i activation get c541fcb8b7d0409183cccc754dd43316

ok: got activation c541fcb8b7d0409183cccc754dd43316
{
    "namespace": "guest",
    "name": "myAction",
    "version": "0.0.1",
    "subject": "guest",
    "activationId": "c541fcb8b7d0409183cccc754dd43316",
    "start": 1501627113632,
    "end": 1501627121984,
    "duration": 8352,
    "response": {
        "status": "success",
        "statusCode": 0,
        "success": true,
        "result": {
            "hello": "world"
        }
    },
    "logs": [
        "2017-08-01T22:38:41.991069723Z stdout: Hello World"
    ],
    "annotations": [
        {
            "key": "limits",
            "value": {
                "logs": 10,
                "memory": 256,
                "timeout": 60000
            }
        },
        {
            "key": "path",
            "value": "guest/myAction"
        }
    ],
    "publish": false
}

user@ubuntu:~/openwhisk$
```

What just happened? In general, we did the following:

1. Create a function
2. Uploaded the function
3. Executed the function
4. Retrieved the function results

If you are looking for details on the internals, have a look at the official documentation here https://github.com/apache/incubator-openwhisk/blob/master/docs/about.md#the-internal-flow-of-processing

Aside from running functions, we can do other CRUD activities like list the available actions.

```
user@ubuntu:~/openwhisk$ ./bin/wsk -i activation list
activations
c541fcb8b7d0409183cccc754dd43316 myAction
9db3a52e6cc94f149911bc92213c76ee echo
user@ubuntu:~/openwhisk$
```

While we previously demoed a simple asynchronous, non-parametric function, we have the ability to do synchronous and parametric (or any mixture of those).

- Based on what we have discussed in the past two days, what concerns you about FaaS, what benefits do you see?

5. Cleanup

When we ran the `wskdev fresh` command to start OpenWhisk, we can see that it started several containers.

To clean up our OpenWhisk deployment, we can simply stop all of the running containers with the following command:

```
user@ubuntu:~$ docker container rm $(docker container stop $(docker container ls -q))

...
```

```
user@ubuntu:~$
```

The command passes the results of `docker container ls -q`, which is a complete listing of our running containers, through the `docker container stop` command. The resulting list of stopped container hashes are then passed through the `docker container rm` command, removing the containers.

Congratulations, you have completed the lab!