OPEN-WHISK DEPLOYMENT

Function as a service (FaaS)

Professor Maria Fazio

Project by Mujibullah Rabin 539269

University of Messina
Operating system

Content:

Function as a service and open whisk

- Function as a Service (FaaS) is a way to deploy a function in the cloud without having to worry about servers.
- So open whisk is an open-source platform that can implement function as a service. Simply we open whisk is a platform that throw this platform we will be able to deploy the function in the server.
  At the same time, we can also deploy it locally for learning purposes or for testing purposes.
  If we want to deploy it throw the server, we will need to have an account in the one of the clouds like IBM, AMAZON, AZURE or any other that can supports open whisk.
  If we want to deploy it locally, we need to have these programs in our machine.
    1. Java
    2. Nodejs
    3. Docker

  and we have also different methods to deploy an open whisk function locally

    1. Deploy by Kubernetes
    2. Docker composes
    3. Ansible
    4. Standalone
- Kubernetes is a container orchestration platform that automates the deployment, scaling, and management of containerized applications.

  And we can deploy the open whisk throw Kubernetes using the Helm.

  Helm is a package manager for Kubernetes that simplifies the management of Kubernetes applications.

- Docker compose containerizing all the requirements and application of the open whisk and running them throw the docker.
- Standalone a computer server that operates independently without relying on any other servers to function.

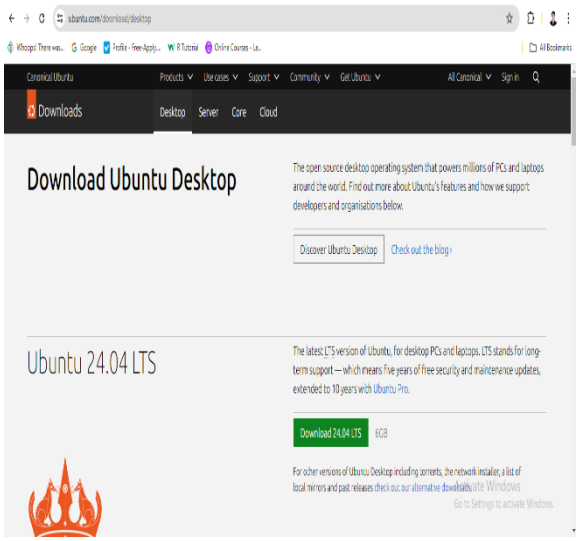
WSL in windows

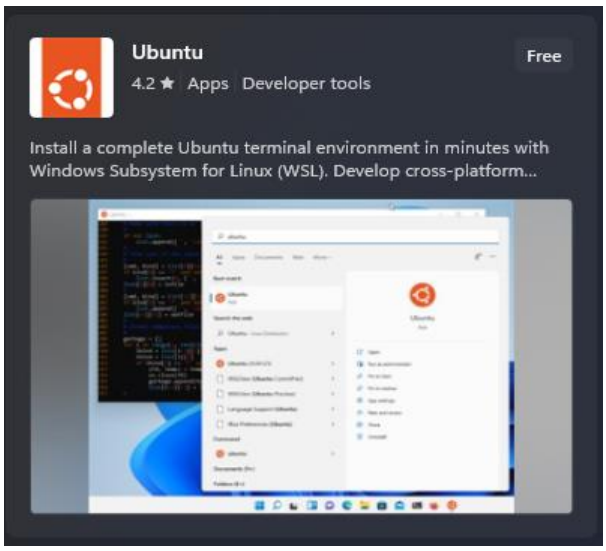
As I am using windows for the easy installing of the open whisk and its requirements batch would be better option and when we want to work with the cloud and server unix system is better choice to deploy easily. Linux and mac are based on unix system.

we can use virtual machine or ubuntu wsl, so we will have specific environment for that to run our commands and install whatever we want in that environment.

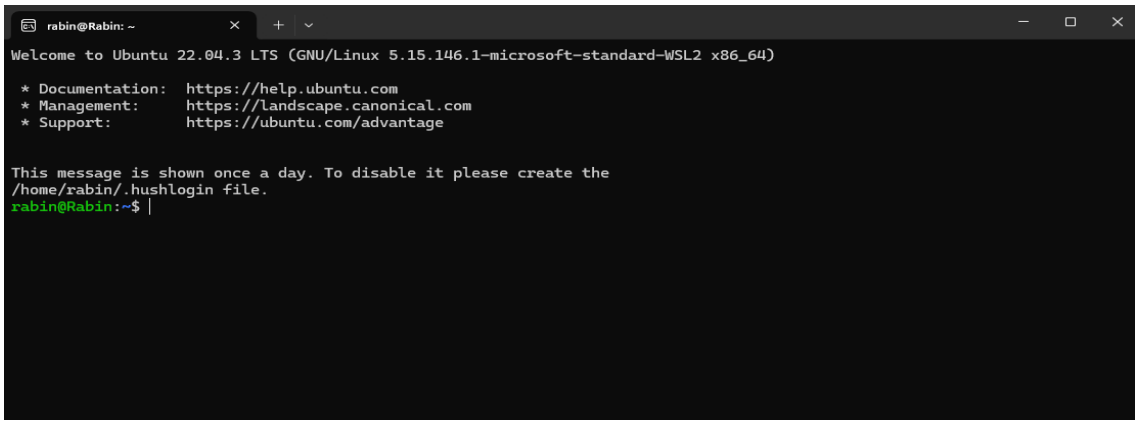
I have used ubuntu wsl that is Linux os, we can download the ubuntu wsl from google.

https://ubuntu.com/download/desktop or from the Microsoft store from windows and app store in the mac.





After installing the ubuntu we can directly open the batch shell or ubuntu shell from our pc where it will ask for a username and password for our Linux environment.

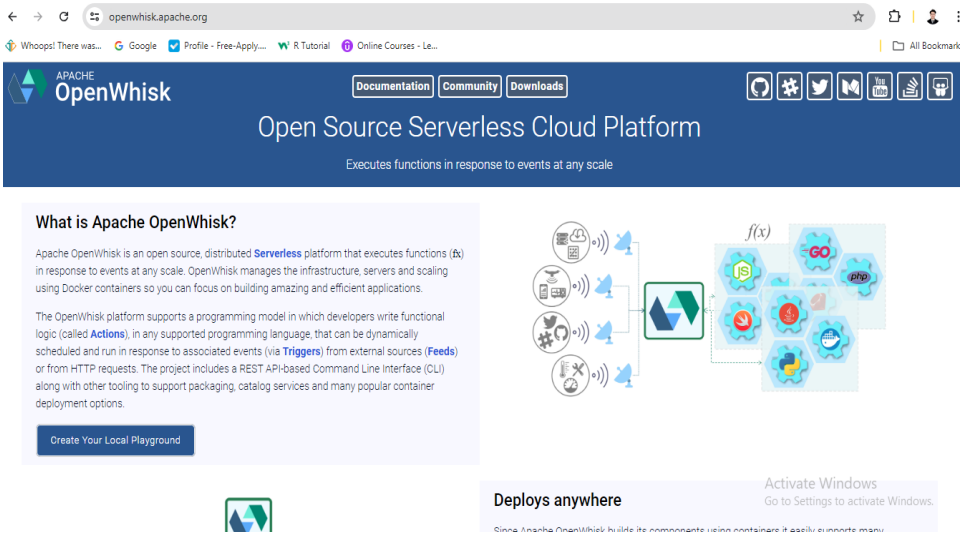


And from here we can start our batch commands and install whatever we want in the Linux environment.

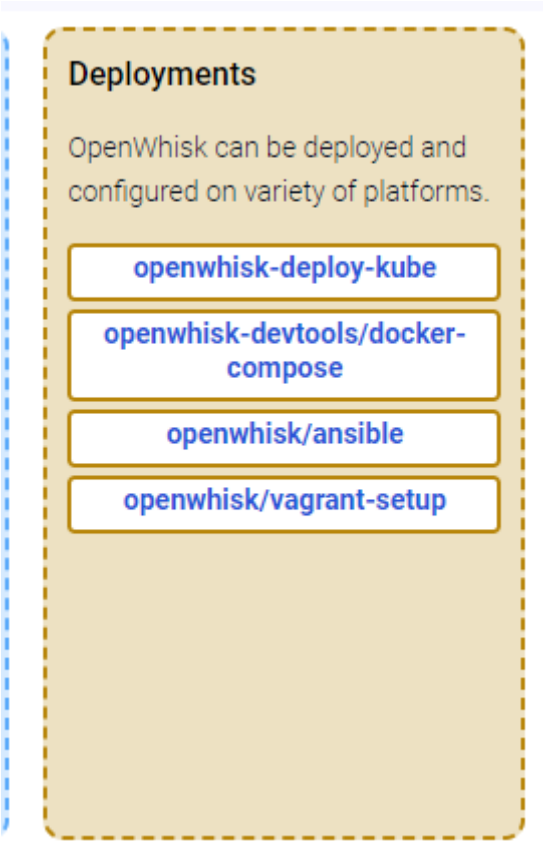
Let’s start deploying locally

First, we will go to the Apache open whisk official website where we can find all the instruction of deployment and creating function in the server or locally.

From this website https://openwhisk.apache.org/



When we go to create local playground we will see several options of deploying it locally



Each of them have their instruction in GitHub we can access.

Before starting our deployment locally, we must install

1. Open whisk
2. Open whisk cli
3. Node.js
4. Python (or preferred programming language)
5. Docker
6. Docker composes

Let's install the open whisk and open whisk cli

We can install it throw cloning to the GitHub account of the open whisk

git clone https://github.com/apache/openwhisk.git we use this command to clone the open whisk.

And it will start cloning.

```
rabin@Rabin:~$ git clone https://github.com/apache/openwhisk.git
Cloning into 'openwhisk'...
remote: Enumerating objects: 57396, done.
remote: Counting objects: 100% (1148/1148), done.
remote: Compressing objects: 100% (600/600), done.
Receiving objects:  15% (8610/57396), 4.46 MiB | 2.11 MiB/s
```

When the cloning is finished, we can directly access to the open whisk folder by using the "cd openwhisk" command

And to install the openwhisk client we use the homebrew that is package manager and throw the homebrew we will install the openwhisk client.

We can install the homebrew by its website https://brew.sh/ and copy the cloning code and install it.

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

```
rabin@Rabin:~$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
==> Checking for `sudo` access (which may request your password)...
==> This script will install:
/home/linuxbrew/.linuxbrew/bin/brew
/home/linuxbrew/.linuxbrew/share/doc/homebrew
/home/linuxbrew/.linuxbrew/share/man/man1/brew.1
/home/linuxbrew/.linuxbrew/share/zsh/site-functions/_brew
/home/linuxbrew/.linuxbrew/etc/bash_completion.d/brew
/home/linuxbrew/.linuxbrew/Homebrew

Press RETURN/ENTER to continue or any other key to abort:
```

The homebrew will be installed and will be update automatically by this command.

Then we can install the open whisk cli by: ‘brew install wsk’ command.

```
rabin@Rabin:~$ brew install wsk
==> Auto-updating Homebrew...
Adjust how often this is run with HOMEBREW_AUTO_UPDATE_SECS or disable with
HOMEBREW_NO_AUTO_UPDATE. Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
Warning: wsk 1.2.0 is already installed and up-to-date.
To reinstall 1.2.0, run:
  brew reinstall wsk
rabin@Rabin:~$
```

And we can ensure of the installing we can directly write wsk.

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

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

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
  -h, --help                  help for wsk
  -i, --insecure              bypass certificate checking
      --key string            client key
  -v, --verbose               verbose output
```

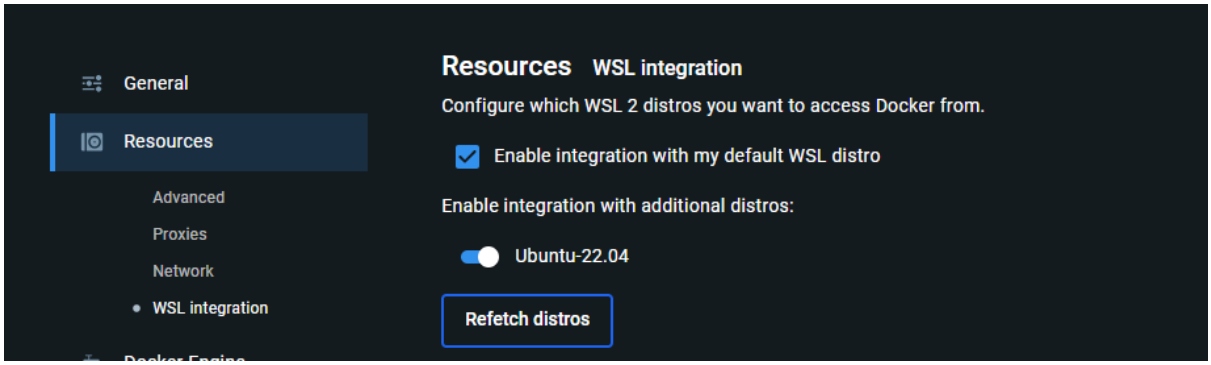
Where we also can see the instruction here.

Since I have installed python, node.js, java in my machine I can start deploying it locally.

Deploying open whisk using Kubernetes

For kubernetese we need docker where we are going to use the minikube because minikube is used for the linux.

We can install the docker inside of our ubuntu environment or we can connect the docker desktop to our wsl.



Now our ubuntu is connected with the docker.

And we have to install the helm by its official website https://helm.sh/docs/intro/install/

From Source (Linux, macOS)

Building Helm from source is slightly more work, but is the best way to go if you want to test the latest (pre-release) Helm version.

You must have a working Go environment.

```
$ git clone https://github.com/helm/helm.git
$ cd helm
$ make
```

$ git clone https://github.com/helm/helm.git

$ cd helm

$ make

By this command we can install helm and access to the helm directory.

```
rabin@Rabin:~$ git clone https://github.com/helm/helm.git
Cloning into 'helm'...
remote: Enumerating objects: 57295, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 57295 (delta 12), reused 14 (delta 4), pack-reused 57263
Receiving objects: 100% (57295/57295), 20.16 MiB | 1.30 MiB/s, done.
Resolving deltas: 100% (38269/38269), done.
Updating files: 100% (1367/1367), done.
rabin@Rabin:~$
```

Then we need a Kubernetes where for Linux minikube is a good option.

We can install the minikube from its official website.
https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download

According to the operating system we can download the minikube. And for linux this is the command.

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

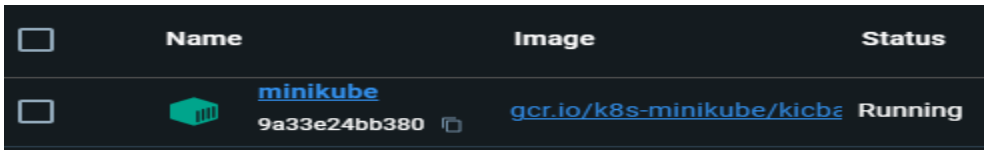
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64

```
rabin@Rabin:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
 12 91.1M   12 11.3M    0     0  1900k      0  0:00:49  0:00:06  0:00:43 2200k
```

Then we can start our cluster by using 'minikube start' command where this command for the first time pulls the image and create a container for the minikube, but when its created next time it will just start the cluster and container.

```
rabin@Rabin:~$ minikube start
😄  minikube v1.33.1 on Ubuntu 22.04 (amd64)
✨  Using the docker driver based on existing profile

❗  Requested memory allocation (1800MB) is less than the recommended minimum 1900MB. Deployments may fail.


❗  The requested memory allocation of 1800MiB does not leave room for system overhead (total system memory: 1862MiB).

💡  Suggestion: Start minikube with less memory allocated: 'minikube start --memory=1862mb'

👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.44 ...
🤷  docker "minikube" container is missing, will recreate.
🔥  Creating docker container (CPUs=2, Memory=1800MB) ...-
```

And here we can see the container and images have pulled in the docker.



As now we have installed all the requirements, we will continue to deploy the open whisk.

Here is the instruction of deployment by Kubernetes and helm.

https://github.com/apache/openwhisk-deploy-kube/blob/master/README.md#prerequisites-kubernetes-and-helm

now we have to make a cluster yaml file that is already mentioned in the website.

Single Worker Node Clusters

If your cluster has a single worker node, then you should configure OpenWhisk without node affinity. This is done by adding the following lines to your `mycluster.yaml`

```
affinity:
  enabled: false

toleration:
  enabled: false

invoker:
  options: "-Dwhisk.kubernetes.user-pod-node-affinity.enabled=false"
```

We will create a directory and inside that directory we will have all the file we would need, including this docker file.

Here we have created a directory owkub where we will save the yaml file here.

```
rabin@Rabin:~$ mkdir openwhisk kub
mkdir: cannot create directory 'openwhisk': File exists
rabin@Rabin:~$ mkdir owkub
rabin@Rabin:~$ cs
Command 'cs' not found, but can be installed with:
sudo apt install csound
rabin@Rabin:~$ cd owkub
rabin@Rabin:~/owkub$
```

We will create a yaml file using nano

```
rabin@Rabin:~/owkub$ nano mycluster.yaml
```

When we use the nano, it will directly take us a page where we can write and edit our code.

```
  GNU nano 6.2                                          mycluster.yaml *
affinity:
  enabled: false

toleration:
  enabled: false

invoker:
  options: "-Dwhisk.kubernetes.user-pod-node-affinity.enabled=false"
```

When we make the file, we save it.

Our cluster is done now we will deploy the openwhisk using the Helm.

Here are the commands to deploy openwhisk inside the minikube using the helm.

Deploying Released Charts from Helm Repository

The OpenWhisk project maintains a Helm repository at `https://openwhisk.apache.org/charts`. You may install officially released versions of OpenWhisk from this repository:

```
helm repo add openwhisk https://openwhisk.apache.org/charts
helm repo update
helm install owdev openwhisk/openwhisk -n openwhisk --create-namespace -f mycluster.yaml
```

Commands:

helm repo add openwhisk https://openwhisk.apache.org/charts

helm repo update

helm install owdev openwhisk/openwhisk -n openwhisk --create-namespace -f mycluster.yaml

```
rabin@Rabin:~/owkub$ helm repo add openwhisk https://openwhisk.apache.org/charts
helm repo update
helm install owdev openwhisk/openwhisk -n openwhisk --create-namespace -f mycluster.yaml
"openwhisk" already exists with the same configuration, skipping
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "openwhisk" chart repository
Update Complete. ⎈Happy Helming!⎈
Error: INSTALLATION FAILED: cannot re-use a name that is still in use
```

Here I got an error because I have already used the name previous time that is why it cannot installed it again, I can use another name, or I can use upgrade option to force it install or I can delete it and install again.

So, I preferred to uninstall it install it again:

Uninstallation command: helm uninstall owdev -n openwhisk

```
rabin@Rabin:~/owkub$ helm uninstall owdev -n openwhisk
release "owdev" uninstalled
rabin@Rabin:~/owkub$
```

And I install it again.

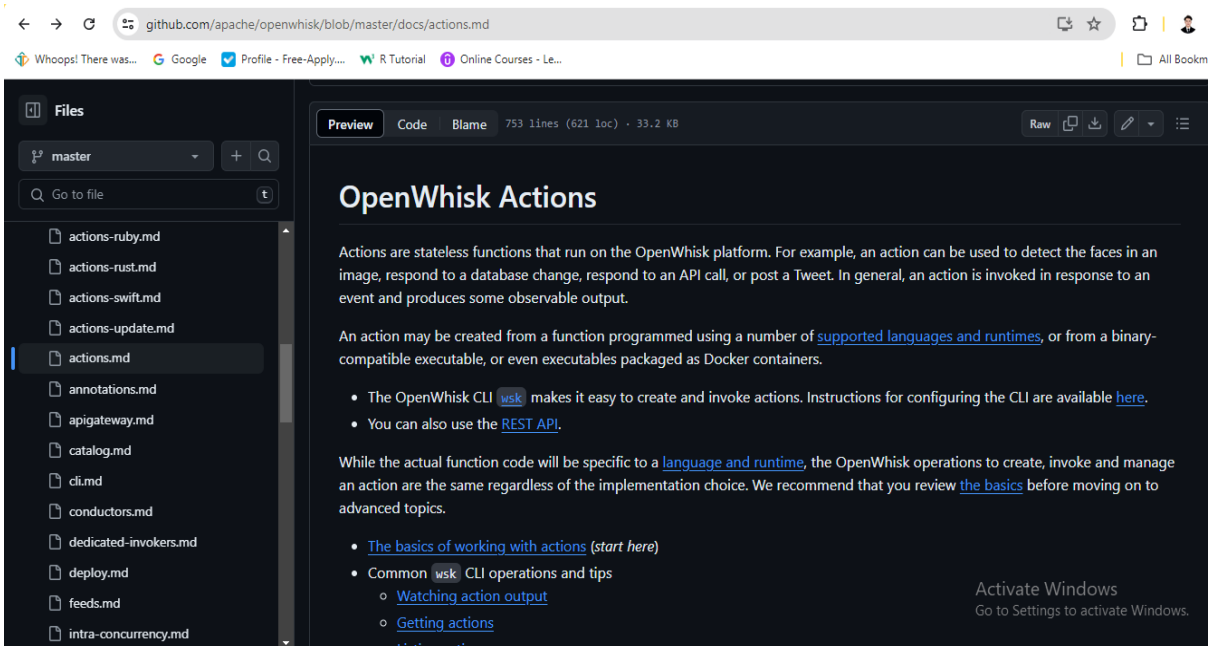
```
rabin@Rabin:~/owkub$ helm repo add openwhisk https://openwhisk.apache.org/charts
helm repo update
helm install owdev openwhisk/openwhisk -n openwhisk --create-namespace -f mycluster.yaml
"openwhisk" already exists with the same configuration, skipping
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "openwhisk" chart repository
Update Complete. ⎈Happy Helming!⎈
NAME: owdev
LAST DEPLOYED: Fri Jun 21 16:08:40 2024
NAMESPACE: openwhisk
STATUS: deployed
REVISION: 1
NOTES:
Apache OpenWhisk
Copyright 2016-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

To configure your wsk cli to connect to it, set the apihost property
using the command below:

  $ wsk property set --apihost :31001

Your release is named owdev.

To learn more about the release, try:

  $ helm status owdev
  $ helm get owdev

Once the 'owdev-install-packages' Pod is in the Completed state, your OpenWhisk deployment is ready to be used.

Once the deployment is ready, you can verify it using:
```

So here after installation got complete, we can see that we have deployed the open whisk inside the minikube (cluster) successfully.

It has created the API host, 31001 and now we must create an action.

To create an action, we can get the instruction form this GitHub account:

https://github.com/apache/openwhisk/blob/master/docs/actions.md



So, we already have the API host as it is 31001 and now, I need to know the minikube ip address and auth key.

In conclusion simply we have created the auth key and API host locally.

```
rabin@Rabin:~/owkub$ minikube ip
❗  Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 2.307423415s
💡  Restarting the docker service may improve performance.
192.168.148.2
rabin@Rabin:~/owkub$ kubectl get secrets -n openwhisk
NAME                          TYPE                  DATA   AGE
owdev-db.auth                 Opaque                2      13m
owdev-nginx                   kubernetes.io/tls     2      10d
owdev-whisk.auth              Opaque                2      13m
sh.helm.release.v1.owdev.v1   helm.sh/release.v1    1      13m
```

Here I have got my minikube address by using the ‘’minikube ip’’ command that the address is 192.168.148.2

And to get the auth key I use ‘kubectl’ that minikube command to access the cluster pods. By using this this command, I will get my authentication key.

kubectl get secret owdev-whisk.auth -n openwhisk -o jsonpath='{.data.guest}' | base64 –decode

now I will set the API host and the authentication key

API host setting command: wsk property set --apihost http://192.168.148.2:31001

Auth key setting command: wsk property set --auth 23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP

```
rabin@Rabin:~/owkub$ wsk property set --apihost http://192.168.148.2:31001
ok: whisk API host set to http://192.168.148.2:31001
rabin@Rabin:~/owkub$ wsk property set --auth 23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP
ok: whisk auth set. Run 'wsk property get --auth' to see the new value.
rabin@Rabin:~/owkub$
```

Here we have set the API host and auth key both are ok. Mean successfully we have set up both.

- Now I will create a simple function hello world to check if it will work or not:

I will create a python file.

```
  GNU nano 6.2                                                    hello.py *
def main(params):
    return {"message": "Hello, World!"}
```

Now I will use wsk (open whisk cli) to create a function.

Using this command: wsk action create hello hello.py --kind python:3.7

now I should be able deploy the hello world action.

But I got this error:

```
rabin@Rabin:~/owkub$ kubectl get secrets -n openwhisk
NAME                          TYPE                  DATA   AGE
owdev-db.auth                 Opaque                2      13m
owdev-nginx                   kubernetes.io/tls     2      10d
owdev-whisk.auth              Opaque                2      13m
sh.helm.release.v1.owdev.v1   helm.sh/release.v1    1      13m
rabin@Rabin:~/owkub$ kubectl get secret <secret-name> -n openwhisk -o jsonpath='{.data.guest}' | base64 --decode
-bash: secret-name: No such file or directory
rabin@Rabin:~/owkub$ kubectl get secret owdev-whisk.auth -n openwhisk -o jsonpath='{.data.guest}' | base64 --decode
23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwPrabin@Rabin:~/owkub$
rabin@Rabin:~/owkub$ wsk property set --apihost http://192.168.148.2:31001
ok: whisk API host set to http://192.168.148.2:31001
rabin@Rabin:~/owkub$ wsk property set --auth 23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP
ok: whisk auth set. Run 'wsk property get --auth' to see the new value.
rabin@Rabin:~/owkub$ nano hello.py
rabin@Rabin:~/owkub$ wsk action create hello hello.py --kind python:3.7
error: Unable to create action 'hello': Put "http://192.168.148.2:31001/api/v1/namespaces/_/actions/hello?overwrite=false": dial tcp 192.168.148.2:3
1001: i/o timeout
Run 'wsk --help' for usage.
rabin@Rabin:~/owkub$
```

The error indicates that it cannot create a function because it the port can not connect with the network.

So, we have to check the pods.

We can check the pods by using the minikube kubectl command:

Kubectl get pods -n openwhisk

```
rabin@Rabin:~/owkub$ wsk action create hello hello.py --kind python:3.7
error: Unable to create action 'hello': Put "http://192.168.148.2:31001/api/v1/namespaces/_/actions/hello?overwrite=false": dial tcp
192.168.148.2:31001: i/o timeout
Run 'wsk --help' for usage.
rabin@Rabin:~/owkub$ kubectl get pods -n openwhisk
NAME                                                READY   STATUS             RESTARTS        AGE
owdev-alarmprovider-6bf4cdd68c-qsd8n                0/1     Init:0/1           0               34m
owdev-apigateway-644688857f-z9z4s                   1/1     Running            0               34m
owdev-controller-0                                  0/1     CrashLoopBackOff   13 (2m1s ago)   34m
owdev-couchdb-555b6d8ff4-snqcj                      1/1     Running            0               34m
owdev-gen-certs-tvxb5                               0/1     Completed          0               34m
owdev-init-couchdb-jsmwt                            0/1     Completed          0               34m
owdev-install-packages-6gbmp                        0/1     Init:0/1           0               34m
owdev-invoker-0                                     0/1     Init:0/1           0               34m
owdev-kafka-0                                       1/1     Running            2 (27m ago)     34m
owdev-kafkaprovider-5785cbb579-7gp2p                0/1     Init:0/1           0               34m
owdev-nginx-6c7448668b-spqsm                        0/1     Init:0/1           0               34m
owdev-redis-848f56f555-56s8s                        1/1     Running            0               34m
owdev-wskadmin                                      1/1     Running            0               34m
owdev-zookeeper-0                                   1/1     Running            1 (32m ago)     34m
wskowdev-invoker-00-11-whisksystem-invokerhealthtestaction0   1/1     Running            4 (59m ago)     9d
wskowdev-invoker-00-8-prewarm-nodejs10              1/1     Running            4 (59m ago)     9d
wskowdev-invoker-00-9-prewarm-nodejs10              1/1     Running            4 (59m ago)     9d
rabin@Rabin:~/owkub$
```

Here we can see that there are some pods are not running and some have error.

And we will check the pod detail that is crashed:

Command: kubectl describe pod owdev-controller-0 -n openwhisk

Here we can see the pod have the problem connecting with the TCP connection.

```
    DownwardAPI:              true
QoS Class:                    BestEffort
Node-Selectors:               <none>
Tolerations:                  node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason     Age                    From               Message
  ----     ------     ----                   ----               -------
  Normal   Scheduled  39m                    default-scheduler  Successfully assigned openwhisk/owdev-controller-0 to minikube
  Normal   Pulled     38m                    kubelet            Container image "busybox:latest" already present on machine
  Normal   Created    38m                    kubelet            Created container wait-for-kafka
  Normal   Started    38m                    kubelet            Started container wait-for-kafka
  Normal   Pulled     36m                    kubelet            Container image "busybox:latest" already present on machine
  Normal   Created    36m                    kubelet            Created container wait-for-couchdb
  Normal   Started    36m                    kubelet            Started container wait-for-couchdb
  Normal   Pulling    36m                    kubelet            Pulling image "openwhisk/controller:1.0.0"
  Normal   Pulled     36m                    kubelet            Successfully pulled image "openwhisk/controller:1.0.0" in 6.495s (6.4
95s including waiting). Image size: 544112018 bytes.
  Normal   Killing    35m                    kubelet            Container controller failed liveness probe, will be restarted
  Normal   Pulled     35m                    kubelet            Container image "openwhisk/controller:1.0.0" already present on machi
ne
  Normal   Created    35m (x2 over 36m)      kubelet            Created container controller
  Normal   Started    35m (x2 over 36m)      kubelet            Started container controller
  Warning  Unhealthy  35m (x5 over 36m)      kubelet            Liveness probe failed: Get "http://10.244.0.186:8080/ping": dial tcp
10.244.0.186:8080: connect: connection refused
  Warning  Unhealthy  13m (x55 over 36m)     kubelet            Readiness probe failed: Get "http://10.244.0.186:8080/ping": dial tcp
 10.244.0.186:8080: connect: connection refused
  Warning  BackOff    3m21s (x113 over 32m)  kubelet            Back-off restarting failed container controller in pod owdev-controll
er-0_openwhisk(cf6c57b5-9fef-4e35-b30d-246fcb0ed5ea)
rabin@Rabin:~/owkub$
```

Here the pods are restarting and trying to fix but they cannot this restarting and resolving method that tried to solve the problem was:

Alias kubectl and alias kubectl tries to fix some pods when they don't run, or they got down and restart them try to recover those pods:

You can also make your life easier by adding the following to your shell config: (for more details see: kubectl)

```
alias kubectl="minikube kubectl --"
```

Initially, some services such as the storage-provisioner, may not yet be in a Running state. This is a normal condition during cluster bring-up, and will resolve itself momentarily. For additional insight into your cluster state, minikube bundles the Kubernetes Dashboard, allowing you to get easily acclimated to your new environment:

But still with this it didn't work, and the pods are restarting but they don't get fixed.

Deploying the open whisk using docker compose

After many attempts and trying many ways, I was not ale to fix the pods and create an action.

So, i will with another method of deploying it locally using docker compose.

And to get more instructions we can go to GitHub site that Is mentioned on the website.

https://github.com/apache/openwhisk-devtools/blob/master/docker-compose/README.md

first, we will clone the open whisk using the docker compose:

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

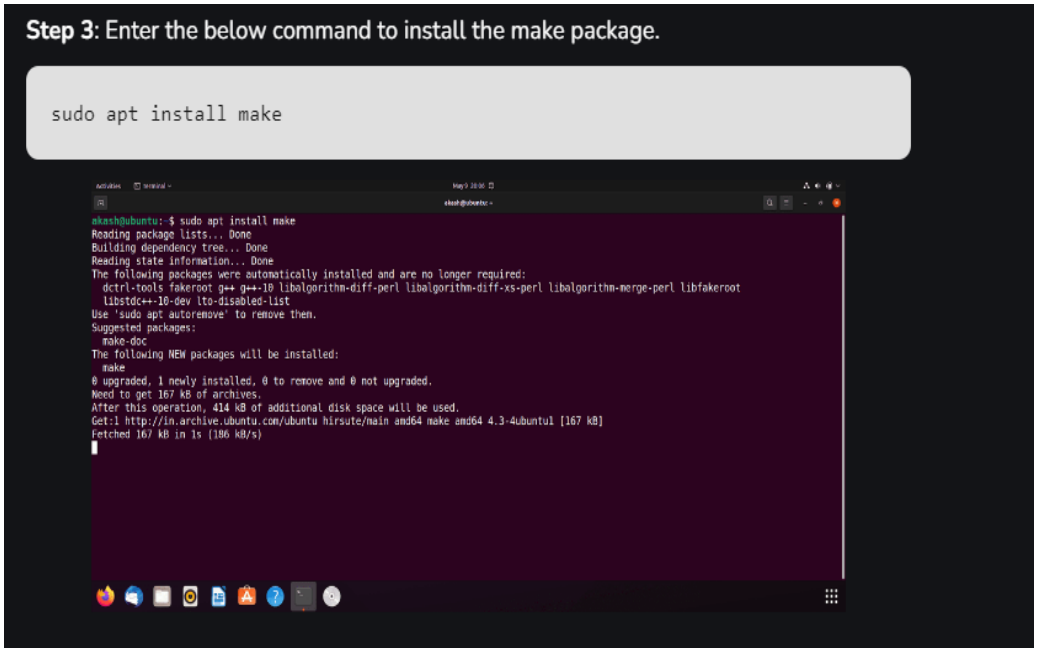
```
rabin@Rabin:~/owkub$ cd
rabin@Rabin:~$ git clone https://github.com/apache/openwhisk-devtools.git
Cloning into 'openwhisk-devtools'...
remote: Enumerating objects: 2010, done.
remote: Counting objects: 100% (232/232), done.
remote: Compressing objects: 100% (26/26), done.
Receiving objects: 100% (2010/2010), 74.14 MiB | 2.55 MiB/s, done.
remote: Total 2010 (delta 221), reused 206 (delta 206), pack-reused 1778
Resolving deltas: 100% (1035/1035), done.
Updating files: 100% (293/293), done.
rabin@Rabin:~$ |
```

And we can access to the directory use the ‘make quick start’ command where this command will install all the files and create the containers and images using docker compose.

```
rabin@Rabin:~/openwhisk-devtools/docker-compose$ make quick-start
Command 'make' not found, but can be installed with:
sudo apt install make        # version 4.3-4.1build1, or
sudo apt install make-guile  # version 4.3-4.1build1
rabin@Rabin:~/openwhisk-devtools/docker-compose$
```

Here I got an error about the it didn't recognize the make command, so we must install the make command

- We can install make by the instruction written in ubuntu directly
- Or by the make website get the command and install
  Make website: https://www.geeksforgeeks.org/how-to-install-make-on-ubuntu/



And here we have installed the make:

```
rabin@Rabin:~/openwhisk-devtools/docker-compose$ sudo apt install make
[sudo] password for rabin:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
0 upgraded, 1 newly installed, 0 to remove and 47 not upgraded.
Need to get 180 kB of archives.
After this operation, 426 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 make amd64 4.3-4.1build1 [180 kB]
Fetched 180 kB in 1s (342 kB/s)
Selecting previously unselected package make.
(Reading database ... 24209 files and directories currently installed.)
Preparing to unpack .../make_4.3-4.1build1_amd64.deb ...
Unpacking make (4.3-4.1build1) ...
Setting up make (4.3-4.1build1) ...
Processing triggers for man-db (2.10.2-1) ...
```

We must put it in the local user bin

The instruction is already mentioned in make website

Now we will start again the ‘make quick start command’

Now the make is installed it started downloading the open whisk in docker compose

```
rabin@Rabin:~/openwhisk-devtools/docker-compose$ make quick-start
/bin/bash: line 1: route: command not found
env: 'ifconfig': No such file or directory
/bin/bash: line 1: route: command not found
env: ‘ifconfig’: No such file or directory
Makefile:502: warning: overriding recipe for target 'install-package-alarms'
Makefile:495: warning: ignoring old recipe for target 'install-package-alarms'
Unpacking tarball.
downloading the CLI tool ...
downloading cli for linux
```

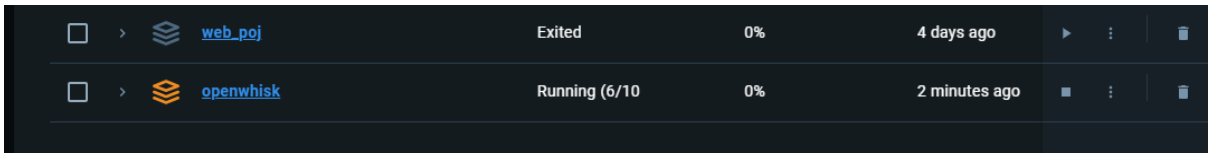
And it will start the pulling images and containers.

```
rabin@Rabin:~/openwhisk-devtools/docker-compose$ make quick-start
/bin/bash: line 1: route: command not found
env: 'ifconfig': No such file or directory
/bin/bash: line 1: route: command not found
env: ‘ifconfig’: No such file or directory
Makefile:502: warning: overriding recipe for target 'install-package-alarms'
Makefile:495: warning: ignoring old recipe for target 'install-package-alarms'
Unpacking tarball.
downloading the CLI tool ...
downloading cli for linux
nightly: Pulling from openwhisk/controller
a0d0a0d46f8b: Pull complete
fb5500646e2d: Pull complete
d32d1308a415: Downloading [========>                                          ]  35.62MB/203.3MB
836f0c831c2e: Download complete
ba15d8519760: Download complete
c3992e518918: Download complete
a72a3b3209f7: Download complete
3037337e54bd: Download complete
39ed04e208a7: Download complete
157dc08bb87f: Download complete
4b1aa0291dec: Download complete
7a31b7b65228: Download complete
4d90be49fc9a: Downloading [========================>                          ]  11.26MB/23.37MB
247026da2f7f: Download complete
5b4cdf3629e6: Downloading [>                                                  ]  1.619MB/135.4MB
88b2f38ae246: Waiting
09e64892c5d4: Pulling fs layer
9219e3cc5141: Pulling fs layer
```

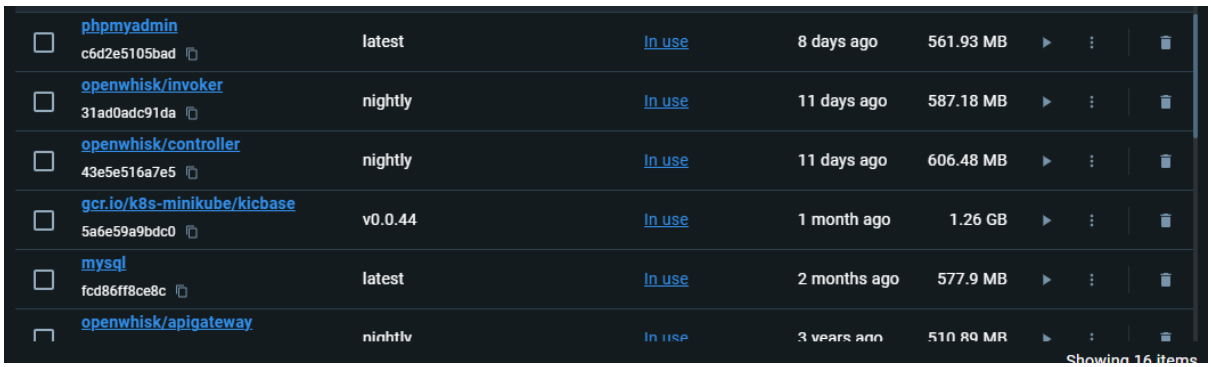
Here all the images are downloaded and installed without any problem:

```
 + kafka-topics-ui [      ] Pulling 80.4s
[+] Running 55/6##] 71.23MB / 71.7MB  Pulling 80.4s
 ✔zookeeper Pulled         326.3s .3s MB Pulling 326.3s
 ✔db Pulled                357.6s MB Pulling 357.6s ose/docker-compose.yml: `version` is obsolete
 ✔kafka-topics-ui Pulled   114.7s .7s .361MB Pulling 114.7s
 ✔redis Pulled             89.9s .9s Pulling 89.8s tes (healthy)
 ✔kafka Pulled             250.6s .6s Pulling 250.5s ch...
 ✔kafka-rest Pulled        210.3s .3s    Pulling 210.3s r 1
[+] Running 10/10nwhisk-devtools/docker-compose$
 ✔Container openwhisk-minio-1            Running0.0s
 ✔Container openwhisk-db-1               Created4.5s
 ✔Container openwhisk-zookeeper-1        Created4.7s
 ✔Container openwhisk-redis-1            Created4.4s
 ✔Container openwhisk-kafka-1            Created2.3s
 ✔Container openwhisk-kafka-rest-1       Created0.6s
 ✔Container openwhisk-controller-1       Created0.8s
 ✔Container openwhisk-invoker-1          Created0.8s
 ✔Container openwhisk-kafka-topics-ui-1  Created0.4s
 ✔Container openwhisk-apigateway-1       Created0.3s
zookeeper-1        | ZooKeeper JMX enabled by default
zookeeper-1        | Using config: /conf/zoo.cfg
kafka-1            | Error response from daemon: {"message":"client version 1.23 is too old. Minimum supported API version is 1.24, please upgrade y
our client to a newer version"}
```

As in my docker desktop all the images and containers are available now.



web_poj	Exited	0%	4 days ago	
openwhisk	Running (6/10	0%	2 minutes ago	



phpmyadmin c6d2e5105bad	latest	In use	8 days ago	561.93 MB	
openwhisk/invoker 31ad0adc91da	nightly	In use	11 days ago	587.18 MB	
openwhisk/controller 43e5e516a7e5	nightly	In use	11 days ago	606.48 MB	
gcr.io/k8s-minikube/kicbase 5a6e59a9bdc0	v0.0.44	In use	1 month ago	1.26 GB	
mysql fcd86ff8ce8c	latest	In use	2 months ago	577.9 MB	
openwhisk/apigateway	nightly	In use	3 years ago	510.89 MB	

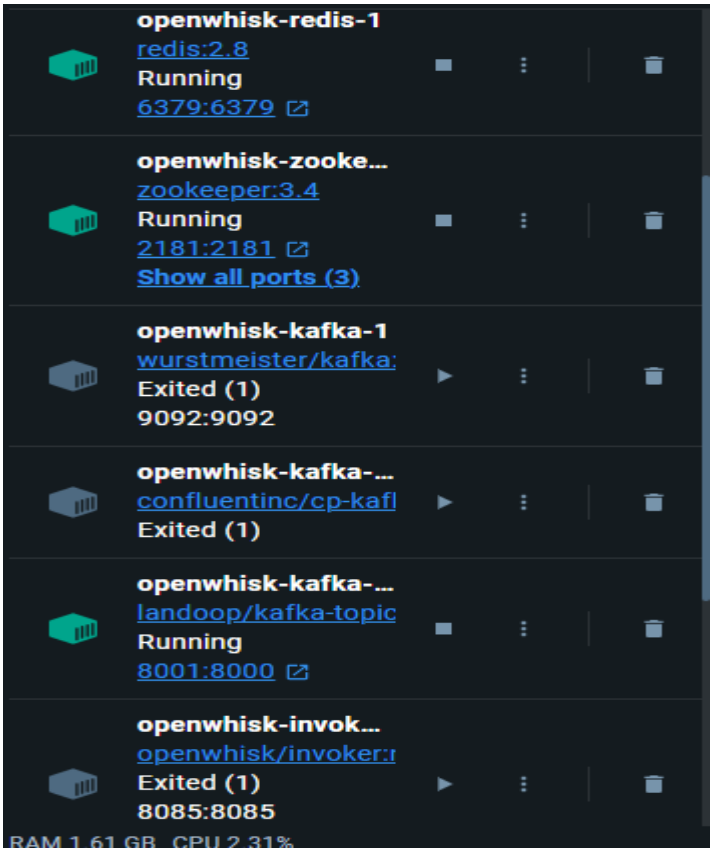
Showing 16 items

But there is still similar error as the Kubernetes deployment method, it cannot connect some pods cannot connect to the network, in this one some images don't run when I start them manually, they will stop again.

```
d_shards_from_disk,1,[{file,"src/mem3_shards.erl"},{line,370}]},{mem3_shards,load_shards_from_disk,2,[{file,"src/mem3_shards.erl"},{line,399}]},{mem
3_shards,for_docid,3,[{file,"src/mem3_shards.erl"},{line,86}]},{fabric_doc_open,go,3,[{file,"src/fabric_doc_open.erl"},{line,39}]},{chttpd_auth_cach
e,ensure_auth_ddoc_exists,2,[{file,"src/chttpd_auth_cache.erl"},{line,195}]},{chttpd_auth_cache,listen_for_changes,1,[{file,"src/chttpd_auth_cache.e
rl"},{line,142}]}]}
db-1              |
apigateway-1      | 2024/06/21 16:17:15 Executing sync cmd: rclone sync minio:api-gateway /etc/api-gateway/
apigateway-1      | 2024/06/21 16:17:16 done
db-1              | [notice] 2024-06-21T16:17:17.636752Z nonode@nohost <0.376.0> -------- chttpd_auth_cache changes listener died database_does_not
_exist at mem3_shards:load_shards_from_db/6(line:395) <= mem3_shards:load_shards_from_disk/1(line:370) <= mem3_shards:load_shards_from_disk/2(line:3
99) <= mem3_shards:for_docid/3(line:86) <= fabric_doc_open:go/3(line:39) <= chttpd_auth_cache:ensure_auth_ddoc_exists/2(line:195) <= chttpd_auth_cac
he:listen_for_changes/1(line:142)
db-1              | [error] 2024-06-21T16:17:17.636878Z nonode@nohost emulator -------- Error in process <0.2159.0> with exit value:
db-1              | {database_does_not_exist,[{mem3_shards,load_shards_from_db,"_users",[{file,"src/mem3_shards.erl"},{line,395}]},{mem3_shards,loa
d_shards_from_disk,1,[{file,"src/mem3_shards.erl"},{line,370}]},{mem3_shards,load_shards_from_disk,2,[{file,"src/mem3_shards.erl"},{line,399}]},{mem
3_shards,for_docid,3,[{file,"src/mem3_shards.erl"},{line,86}]},{fabric_doc_open,go,3,[{file,"src/fabric_doc_open.erl"},{line,39}]},{chttpd_auth_cach
e,ensure_auth_ddoc_exists,2,[{file,"src/chttpd_auth_cache.erl"},{line,195}]},{chttpd_auth_cache,listen_for_changes,1,[{file,"src/chttpd_auth_cache.e
rl"},{line,142}]}]}
db-1              |
```

Here we can see that its restarting and trying but it's not fixing so for now it's a kind of error where it will try to fix, and it will never resolved an **error loop occurred**

like in here we can see that the same pods were not running in Kubernetes deployment methods I have the same problem here. Same containers are not running, restarting them will not work in this deployment method it has auto resolve where in this one tries to resolve and restart the container, but it cannot so a loop happens.



Steps to resolve the problems and errors:

And as it was not able to connect with the network, I installed the network tools:

```
rabin@Rabin:~$ sudo apt-get update
sudo apt-get install net-tools
[sudo] password for rabin:
Sorry, try again.
[sudo] password for rabin:
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1087 kB]
Fetched 1344 kB in 2s (658 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  net-tools
0 upgraded, 1 newly installed, 0 to remove and 47 not upgraded.
Need to get 204 kB of archives.
After this operation, 819 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 net-tools amd64 1.60+git20181103.0eebece-1ubuntu5 [204 kB]
Fetched 204 kB in 3s (76.6 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 24227 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20181103.0eebece-1ubuntu5_amd64.deb ...
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Setting up net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Processing triggers for man-db (2.10.2-1) ...
rabin@Rabin:~$
```

Commands to install network tools:

sudo apt-get update

sudo apt-get install net-tool

and fix the docker compose version where one of the other errors was this one.

```
rabin@Rabin:~/openwhisk-devtools/docker-compose$ nano docker-compose.yml
rabin@Rabin:~/openwhisk-devtools/docker-compose$ nano docker-compose.yml
```

And inside of that docker compose I changed the version from 3 to 3.8

```
#
version: '3'
services:
  db:
    image: apache/couchdb:2.3
    ports:
      - "5984:5984"
    environment:
      COUCHDB_USER: whisk_admin
      COUCHDB_PASSWORD: some_passw0rd
    volumes:
      - ~/tmp/openwhisk/couchdb:/usr/local/var/lib/couchdb:rw

  # KAFKA SERVICES
  zookeeper:
```

And I updated the docker client version.

```
rabin@Rabin:~/openwhisk-devtools/docker-compose$ sudo apt-get update
sudo apt-get install \
  ca-certificates \
  curl \
  gnupg \
  lsb-release
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu4).
lsb-release set to manually installed.
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.16).
curl set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 47 not upgraded.
```

By this I have updated some containers and images and installed docker client version.

Where it updates some containers.

```
rabin@Rabin:~/openwhisk-devtools/docker-compose$ sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
rabin@Rabin:~/openwhisk-devtools/docker-compose$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
rabin@Rabin:~/openwhisk-devtools/docker-compose$ sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
```

Here are the commands

Upgrade the network tool:

sudo apt-get update sudo apt-get install net-tools

upgrade the docker cli:

sudo apt-get update sudo apt-get install docker-ce docker-ce-cli containerd.io

now after all these updates we will try again by accessing to the directory and using make quick start command.

After pulling the images once more now I got this error:

Before there were several errors which with these updates, I have done most of them solved except this error and now it's not looping.

```
2024/06/21 16:49:21 ERROR : Attempt 2/3 failed with 1 errors and: RequestError: send request failed
caused by: Put "http://minio.docker:9000/api-gateway": dial tcp: lookup minio.docker on 127.0.0.11:53: no such host
2024/06/21 16:53:47 ERROR : Attempt 3/3 failed with 1 errors and: RequestError: send request failed
caused by: Put "http://minio.docker:9000/api-gateway": dial tcp: lookup minio.docker on 127.0.0.11:53: no such host
2024/06/21 16:53:47 Failed to mkdir: RequestError: send request failed
caused by: Put "http://minio.docker:9000/api-gateway": dial tcp: lookup minio.docker on 127.0.0.11:53: no such host
make: *** [Makefile:253: setup] Error 1
rabin@Rabin:~/openwhisk-devtools/docker-compose$
```

This error is the same network problem and same error as the Kubernetes one.

After doing a lot of attempts this error cannot be solved and there is some problem with docker-compose file or there is a problem with open whisk network.

In conclusion:

This report provides an in-depth exploration of Function as a Service (FaaS) and the implementation of Apache Open Whisk. FaaS enables developers to deploy functions in the cloud without managing the underlying server infrastructure. OpenWhisk, an open-source platform, facilitates this by allowing functions to be deployed both in the cloud and locally for development and testing.

The report outlines various methods for deploying OpenWhisk locally, including using Kubernetes, Docker Compose, Ansible, and standalone setups. Detailed instructions are provided for setting up the necessary environment and tools, such as Java, Node.js, Docker, and Homebrew, to facilitate the deployment process.

The deployment process using Kubernetes involves installing Minikube and Helm to manage the Kubernetes applications. Despite successfully setting up the necessary components, issues with pod connectivity and network errors were encountered, leading to persistent problems that could not be resolved even with multiple troubleshooting attempts.

Similarly, deploying OpenWhisk using Docker Compose also faced challenges. Errors related to network connectivity and container management persisted, even after updating and installing necessary network tools and Docker client versions. These issues highlight the complexities and potential obstacles in setting up OpenWhisk in a local environment.

while the report provides comprehensive guidance on deploying OpenWhisk locally using various methods, it also underscores the technical challenges and potential issues that can arise. These challenges emphasize the need for robust troubleshooting and a deep understanding of the underlying infrastructure to successfully deploy and manage serverless functions using Open Whisk.

References and sources:

1. https://openwhisk.apache.org/ open whisk official website
2. https://github.com/apache/openwhisk/blob/master/docs/cli.md#openwhisk-cli open whisk cli
3. https://github.com/apache/openwhisk-deploy-kube deploying using Kubernetes
4. https://github.com/apache/openwhisk-devtools/blob/master/docker-compose/README.md docker compose deployment
5. https://minikube.sigs.k8s.io/docs/start/?arch=%2Flinux%2Fx86-64%2Fstable minikube official website
6. https://helm.sh/docs/intro/install/ helm official website
7. https://www.geeksforgeeks.org/how-to-install-make-on-ubuntu/ make installation and official website
8. https://nodejs.org/en/download/package-manager node .js official website and installation instructions
9. https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview ubuntu wsl installing website.