OpenWhisk running on IBM Cloud Private

This repo setup the Apache OpenWhisk on IBM Cloud Private (ICp) with a sample application monitoring the foreign currency exchange rate to demonstrate the serverless functionality.

Prerequisites and assumptions

- You have ICp deployed with dynamic storage provisioning.

  The deployment will use default storage class fulfill the Physical Volume Claims (PVC)

- You have kubectl commandline tool installed and configured

- You have Helm commandline tool installed and configured (which depeneds on the kubectl tool)

To facilitate the deployment and configuration the following tools are used.

- Ruby - windows version download here
- Install the following ruby gems

```
gem install dotenv
gem install sshkit_addon
```

- Optional cmder installed for Windows
- On the ICp master sever, nodejs and npm is installed to build the OpenWhisk actions

Part I. Deploy OpenWhisk onto ICp

Thanks to the effort of Xingzhou who has developed a Helm chart for OpenWhisk. Based on his work, we made a couple of changes

- Make the couchdb persistent on PV.

  Instead of reinitialize the DB every restart, we keep the changes persistent on physical volume by setting a initialized flag.

- Make https as default for the nginx

- Fixed some typo

The steps to deploy the OpenWhisk helm chart on ICp is listed as below (assuming you have already clone this repo)

1. Create openwhisk namespace

```
kubectl create namespace openwhisk

kubectl create clusterrolebinding tiller-cluster-admin --clusterrole=cluster-admin --
serviceaccount=kube-system:default
```

2. Update values.yaml (*optional*)

   Edit the openwhisk_chart/values.yaml file cloned from the repo. You can update the password or the default db prefix

3. Deploy the chart

   Optionally you can validate your settings with a dry-run of install

```
cd icp-openwhisk-currency-monitor

helm install --dry-run --debug openwhisk_chart --namespace=openwhisk
```

After that you perform the actual deployment.

```
cd icp-openwhisk-currency-monitor

helm install openwhisk_chart --namespace=openwhisk
```

4. Wait for the pods to be fully up.
5. while waiting, you can install the openwhisk client.

```
cd /tmp

curl -LO https://github.com/apache/incubator-openwhisk-cli/releases/download/latest/OpenWhisk_CLI-
latest-linux-amd64.zip

sudo apt install unzip -y
unzip OpenWhisk_CLI-latest-linux-amd64.zip
sudo mv wsk /usr/local/bin
```

6. Config the wsk client

   Identify the NodePort of the openwhisk nginx https service

```
kubectl get svc -n openwhisk | grep nginx
```

Note down the port (3xxxx) that is mapped to port 443. Assign the apihost with the master node's IP. See the sample command below:

```
export whisk_default_auth=23bc46b1-71f6-4ed5-8c54-
816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP

wsk property set --auth ${whisk_default_auth} --apihost https://192.168.64.238:32396
```

7. List openwhisk components

   Validate from the Kubernetes dashboard that all the openwhisk pods are running. Then list the components of the openwhisk (all should be empty)

```
wsk -l list
```

Couchdb Docker Image

The docker image is built and pushed to docker hub as "zhiminwen/ow-couchdb:0.1.0".

Optionally, you can build the docker image with the Dockerfile from the directory of couchdb_for_openwhisk in the repo.

Part II. Build OpenWhisk Functions

We will build some serverless functions using NodeJS to get the foreign currency exchange rate. The data that we are retrieving is from the MAS API. A sample data can be viewed in json format.

A list of functions

Package/Function	script	Description
Indicator Function		
FCEX-Mon/lastNValue	lastNValue.js	Get the last N days value of a foreign currency exchange rate to Singapore dollars from MAS API service. All the rest indicators are calling this to provide data
FCEX-Mon/lastValue	lastValue.js	latest exchange rate
FCEX-Mon/SMA	sma.js	Simple Moving Average
FCEX-Mon/EMA	ema.js	Exponential Moving Average
FCEX-Mon/MACD	macd.js	Moving Average Convergence Divergence
Monitoring Function		
FCEX-Mon/Send2Frontend	send2frontend.js	Send back message through websocket
FXEX-Mon/currencyMonitor	monitor.js	Evaluate the indicator formula. Invoke trading action when its true. Send back result for frontend users by calling the send2frontend
Trading Function		
FCEX-Act/Notify	notify.js	Simulate Notify action
FCEX-Act/Sell	sell.js	Simulate Sell action
FCEX-Act/Purchase	purchase.js	Simulate Purchase action

Install OpenWhisk Functions

The following rake tasks created to automate the deployment of the functions. (*It works based on the assumption that you have cloned the repo to your local laptop and you will upload this to the ICp servers using ssh*)

Update the master_ip and password two lines with your actual settings

```
password = "password"
master_ip = %Q(192.168.xx.xx)
```

Run the task in order one by one

```
cd openwhisk_actions
λ rake -f build.rb -T
rake a:01_create_package           # create package
rake a:02_install_lastNValue       # install lastNValue
rake a:03_install_lastValue        # install lastValue
rake a:04_install_SMA              # install SMA
rake a:05_install_EMA              # install EMA
rake a:06_install_MACD             # install MACD
rake a:07_install_currencyMonitor  # install currencyMonitor
rake a:08_install_Send2Frontend    # install Send2Frontend
rake a:09_install_actions          # install purchase and sell actions
rake a:10_test_action              # test action
```

For example

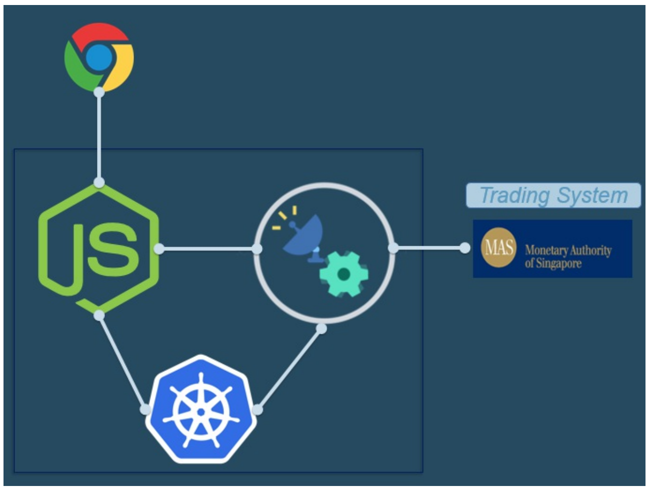
```
rake -f build.rb a:01_create_package
rake -f build.rb a:02_install_lastNValue
...
```

You can modify the related js file and redeploy the openwhisk action by calling the respective rake task.

Part III. Foreign Currency Monitor Application

Introduction

To monitor the foreign currency exchange rate, we build a React web front end together with a NodeJS backend serving the request from the browser. A high level system architecture diagram is shown as below.



The detail architecture and the dataflow is listed as below.

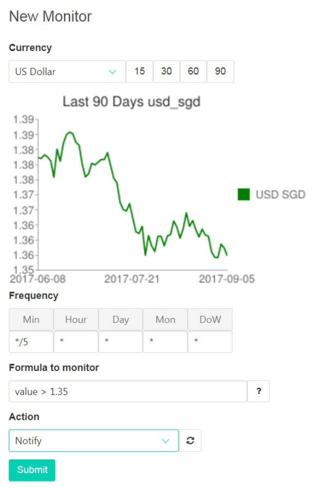
- A NodeJS Express App Server is running as a K8S pod. It hosts a React SPA frontend for user to monitor the exchange rate.
- The NodeJS App Server also interact with Openwhisk to get data for the frontend.

  The Openwhisk functions in turns talk to MAS API to get the data requested from App Server.
- When a user define a schedule, the backend inform Kubernetes to create a cron Job with all the defined info from user, such as,
  - Foreign currency name
  - The frequency to check
  - Indicator formula
  - Action to perform if formula evaluated as true
- The Kubernetes' cron job trigger the OpenWhisk monitor function.

  If the monitor function evaluate the technical indicators as true, it will trigger the user defined trading actions which is implemented as OpenWhisk functions.
- The monitor will send back the result to NodeJS app server through web socket.
- The app server further inform the user through web socket (*socket.io*) on the frontend
- User is able to examine the result or take further action

Some sample screen capture shows the flow.

- Build a monitor



New Monitor

Currency

US Dollar | 15 | 30 | 60 | 90

Frequency

Min	Hour	Day	Mon	DoW
*/5	*	*	*	*

Formula to monitor

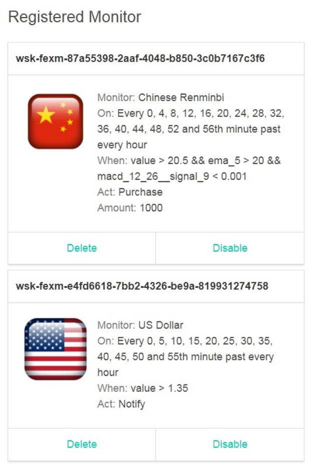
value > 1.35 ?

Action

Notify

Submit

- Registered minitor



Registered Monitor

wsk-fexm-87a55398-2aaf-4048-b850-3c0b7167c3f6

Monitor: Chinese Renminbi
On: Every 0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52 and 56th minute past every hour
When: value > 20.5 && ema_5 > 20 && macd_12_26__signal_9 < 0.001
Act: Purchase
Amount: 1000

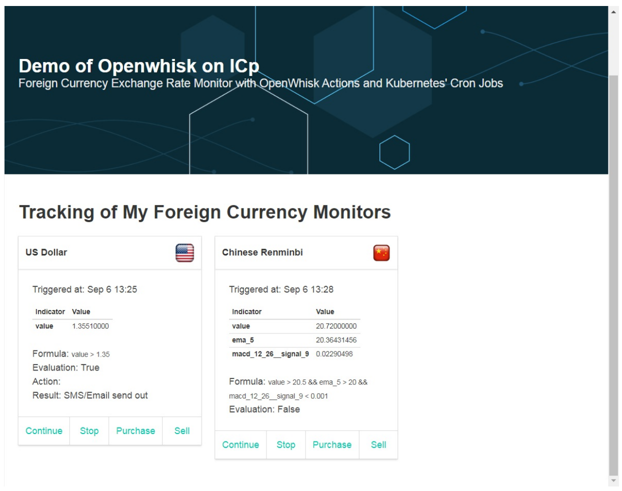
Delete | Disable

wsk-fexm-e4fd6618-7bb2-4326-be9a-819931274758

Monitor: US Dollar
On: Every 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 and 55th minute past every hour
When: value > 1.35
Act: Notify

Delete | Disable

- Tracking



Demo of Openwhisk on ICp
Foreign Currency Exchange Rate Monitor with OpenWhisk Actions and Kubernetes' Cron Jobs

Tracking of My Foreign Currency Monitors

US Dollar

Triggered at: Sep 6 13:25

Indicator	Value
value	1.35510000

Formula: value > 1.35
Evaluation: True
Action:
Result: SMS/Email send out

Continue | Stop | Purchase | Sell

Chinese Renminbi

Triggered at: Sep 6 13:28

Indicator	Value
value	20.72000000
ema_5	20.36431456
macd_12_26__signal_9	0.02290498

Formula: value > 20.5 && ema_5 > 20 && macd_12_26__signal_9 < 0.001
Evaluation: False

Continue | Stop | Purchase | Sell

Deployment

First you will need to update the .env file to fit in your environment.

```
WHISKER_API_HOST=192.168.64.238:32396
WHISKER_KEY=23bc46b1-71f6-4ed5-8c54-
816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP

FCEX_ACTION_PACKAGE=FCEX-Act
```

```
FCEX_MONITOR_PACKAGE=FCEX-Mon
FCEX_MONITOR_ACTION=FCEX-Mon/currencyMonitor

K8S_API_SERVER_URL=https://192.168.64.238:8001
K8S_TOKEN=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZX
JuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9
zZWNyZXQubmFtZSI6ImRlZmF1bHQtdG9rZW4tc3IwdHQiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNj
b3VudC5uYW1lIjoiZGVmYXVsdCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImI4M
GQzMWQ1LThjNzUtMTFlNy1hOGIxLTAwNTA1NmI3YmZmOCIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZm
F1bHQifQ.j3nnFyviJa4KOvAxYdgsjPTeV2Zb4W3XFI6kFimpwDJDBMTjq64P0T-
xHvBsUTUT52sd8mCW742npworqXArU291NJwSSzUOGrzI0kJozBikBX_JCAtB_iVvpsbTTC2et8jzt5pu2I7iyFiy-
zetlNMmxPLONCgdBNGJMuamgwT9XwoERt1-
mXVJE76viWH0sFaOco68k8evZhoBxzu2ImLGhfLsRP1hc7hBlT7faVwBUFKeVGtwcxdwjVZLdRWJJdzOeXlp7LbZi4yW5cpdCqMId
dw8CsX7O1NrylvQGWO6_V8XIpmiT7oGm_LSwBkzf7CcKIyjjXG37-dbblxeDw

K8S_JOB_IMAGE=zhiminwen/wsk-client:0.1.0

FRONTEND_SOCKET_URL=http://192.168.64.238:31720
FRONTEND_SOCKET_TOPIC=monitor-tracking-data
```

The token is a service account token to access K8.

A rakefile is created to automate the building. You will need update with your environment settings accordingly (example: ip address, user and password, ICp private registry access). ((*It works based on the assumption that you have cloned the repo to your local laptop and you will upload this to the ICp servers using ssh*)

Follow the following task in order to upload, build the docker image, push to ICp registry, deploy into ICp Kubernetes.

```
cd monitor_app

λ rake -T
rake 01_upload              # upload
rake 02_build               # build
rake 03_push                # push to icp
rake 04_deploy_k8s          # deploy to K8s
rake 05_scale_down          # scale down
rake 06_remove_old_image    # remove old image
rake 07_restart             # restart image
```

Docker image for OpenWhisk client

The Monitor App launch cronjobs with a docker container. It has a OpenWhisk client so that the client can trigger the OpenWhisk functions. The image is published as zhiminwen/wsk-client:0.1.0 in docker hub.

The Dockerfile to build the image is in the folder of wsk_client

Clean up

You can remove the jobs that has been executed from K8s by running the command

```
kubectl delete jobs $(kubectl get jobs | grep wsk-fexm | awk -e '{print $1}')
```