

# OpenFunction 202

Node.js Async Function Quickstart

Press Space for next page →




# Haili Zhang

KubeSphere Ambassador, CNCF OpenFunction Maintainer.


Cloud Platform Director of UISEE© Technology.

Cloud Native focuses: Kubernetes, DevOps, Observability, Service Mesh, Serverless.



 [webup](#)

 [zhanghaili0610](#)

 [haili.zhang@uisee.com](mailto:haili.zhang@uisee.com)

 [ServiceUP · 语雀](#)

# Prerequisites

Use `ofn` <sup>[1]</sup> <sup>[2]</sup> CLI tool to deploy OpenFunction.

- Install OpenFunction with Async Runtime only <sup>[3]</sup>

```
$ ofn install --async
```

- Install OpenFunction with Async Runtime and function build framework

```
$ ofn install --async --shipwright
```

- 
1. Add `--region-cn` option in case you have limited access to gcr.io or github.com
  2. Use `--dry-run` to peek the components and their versions to be installed by the current command
  3. Please refer [this](#) to learn how to build function image at local  
-----

# Your First Async Function

A brief walkthrough

# Async

```
function (ctx, data) {}
```

# (HTTP) Sync

```
function (req, res) {}
```

# Async

```
function (ctx, data) {}
```

## PARAMETERS

- `ctx`: OpenFunction context object
  - `ctx.send(payload, output?)`: Send `payload` to all or one specific `output` of Dapr Output Binding or Pub Broker
- `data`: Data recieved from Dapr Input Binding or Sub Broker

## NOTICE

- `ctx.send` CAN be invoked where necessary, when you have certain outgoing data to send

# (HTTP) Sync

```
function (req, res) {}
```

# Async

```
function (ctx, data) {}
```

## PARAMETERS

- `ctx`: OpenFunction context object
  - `ctx.send(payload, output?)`: Send `payload` to all or one specific `output` of Dapr Output Binding or Pub Broker
- `data`: Data recieved from Dapr Input Binding or Sub Broker

## NOTICE

- `ctx.send` CAN be invoked where necessary, when you have certain outgoing data to send

# (HTTP) Sync

```
function (req, res) {}
```

## PARAMETERS

- `req`: Express standard request object
- `res`: Express standard response object
  - `res.send(body)`: Use this method to send HTTP response in most common cases

## NOTICE

- Response process SHOULD be explictly ended with `res.send()`, `res.json()`, `res.end()` and alike methods

A sample ``foward`` function



# A sample `forward`` function

## INDEX.MJS

```
// Async function
export const forward = (ctx, data) => {
  console.log('Data received: %o', data);
  ctx.send(data);
};

// HTTP sync function
export const tryKnative = (req, res) => {
  res.send(`Hello, ${req.query.u || 'World'}!`);
};
```

## PACKAGE.JSON

```
{
  "main": "index.mjs",
  "scripts": {
    "start": "functions-framework --target=tryKnative"
  },
  "dependencies": {
    "@openfunction/functions-framework": "^0.4.1"
  }
}
```

# A sample `forward`` function

## INDEX.MJS

```
// Async function
export const forward = (ctx, data) => {
  console.log('Data received: %o', data);
  ctx.send(data);
};

// HTTP sync function
export const tryKnative = (req, res) => {
  res.send(`Hello, ${req.query.u || 'World'}!`);
};
```

## PACKAGE.JSON

```
{
  "main": "index.mjs",
  "scripts": {
    "start": "functions-framework --target=tryKnative"
  },
  "dependencies": {
    "@openfunction/functions-framework": "^0.4.1"
  }
}
```

## NOTICE

- Several async and sync functions CAN be placed in ONE SINGLE JavaScript file
  - Target function CAN be assigned when applying Function CR manifest
- In `package.json``, `scripts`` and `dependencies`` sections could be omitted
  - @openfunction/openfunction-framework lib would be automatically added during build
  - `start`` script is highly recommended for local development

# Build Function Image via Pack Optional

Local build is recommended if your Kubernetes nodes have limited access to GitHub or Docker Hub.

1. Install Cloud Native Buildpacks project's Pack CLI tool
2. Use `pack` tool to build your function image at local <sup>[1]</sup>

```
pack build -B openfunction/builder-node:v2-16.13 \    # Builder image, `16` is the latest version
-e FUNC_NAME=tryKnative \                          # Default entry point function
-p src \                                             # Path of source files to be built
<image-repo>/<image-name>:<tag>
```

3. Push function image to target container repository (e.g. Docker Hub)

```
docker push <image-repo>/<image-name>:<tag>
```

---

1. `pack` tool would download builder image during the build process



# Lab: MQTT Forwarder

---

Use `forwarder` async function to bridge MQTT topic channels

# Set up MQTT Broker Optional

In this lab, we will use EMQX as the broker infrastructure. Learn full steps.

## 1. Add EMQX Helm Chart repository

```
helm repo add emqx https://repos.emqx.io/charts
helm repo update
```

## 2. Search available charts of EMQX

```
helm search repo emqx
```

NAME	CHART VERSION	APP VERSION	DESCRIPTION
emqx/emqx	4.4.3	4.4.3	A Helm chart for EMQX
emqx/emqx-ee	4.4.3	4.4.3	A Helm chart for EMQ X

## 3. Deploy single replica of EMQX, and expose NodePort service

```
helm install emqx emqx/emqx --set replicaCount=1 --set service.type=NodePort
```



```
apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-async-bindings
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<tag>'
  serving:
    # default to knative
    runtime: async
    annotations:
      # default to "grpc"
      daprio.io/app-protocol: http
  template:
    containers:
      - name: function
  params:
    # default to FUNC_NAME value
    FUNCTION_TARGET: tryAsync
  inputs:
    - name: mqtt-input
      component: mqtt-in
  outputs:
    - name: mqtt-output
      component: mqtt-out
      operation: create
```







```
apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-async-bindings
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<tag>'
  serving:
    # default to knative
    runtime: async
    annotations:
      # default to "grpc"
      daprio.io/app-protocol: http
  template:
    containers:
      - name: function
  params:
    # default to FUNC_NAME value
    FUNCTION_TARGET: tryAsync
  inputs:
    - name: mqtt-input
      component: mqtt-in
  outputs:
    - name: mqtt-output
      component: mqtt-out
      operation: create
```

```
apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-async-bindings
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<tag>'
  serving:
    # default to knative
    runtime: async
    annotations:
      # default to "grpc"
      daprio.io/app-protocol: http
  template:
    containers:
      - name: function
  params:
    # default to FUNC_NAME value
    FUNCTION_TARGET: tryAsync
  inputs:
    - name: mqtt-input
      component: mqtt-in
  outputs:
    - name: mqtt-output
      component: mqtt-out
      operation: create
```

```

apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-async-bindings
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<tag>'
  serving:
    # default to knative
    runtime: async
    annotations:
      # default to "grpc"
      daprio/app-protocol: http
  template:
    containers:
      - name: function
  params:
    # default to FUNC_NAME value
    FUNCTION_TARGET: tryAsync
  inputs:
    - name: mqtt-input
      component: mqtt-in
  outputs:
    - name: mqtt-output
      component: mqtt-out
      operation: create

```

```

bindings:
  mqtt-in:
    type: bindings.mqtt
    version: v1
    metadata:
      - name: consumerID
        value: '{uuid}'
      - name: url
        value: tcp://admin:public@emqx:1883
      - name: topic
        value: in
  mqtt-out:
    type: bindings.mqtt
    version: v1
    metadata:
      - name: consumerID
        value: '{uuid}'
      - name: url
        value: tcp://admin:public@emqx:1883
      - name: topic
        value: out

```

- Dapr Component - Bindings - MQTT
- OpenFunction - Function CRD - DaprIO
- Check full sample codes

# Lab 1: MQTT Input and Output Binding

- Apply function manifest, and check running states

```
$ kubectl apply -f async-bindings.yaml
function.core.openfunction.io/sample-node-async-bindings created

$ kubectl get fn
NAME                                BUILDSTATE  SERVINGSTATE  BUILDER  SERVING  URL  AGE
sample-node-async-bindings         Skipped     Running         
serving-8f7xc                      serving-8f7xc  140m

$ kubectl get po
NAME                                READY  STATUS  RESTARTS  AGE
serving-8f7xc-deployment-v200-l78xc-564c6b5bf7-vksg7  2/2    Running  0         141m
```

- Furthermore, check whether `function` container output correct logs`

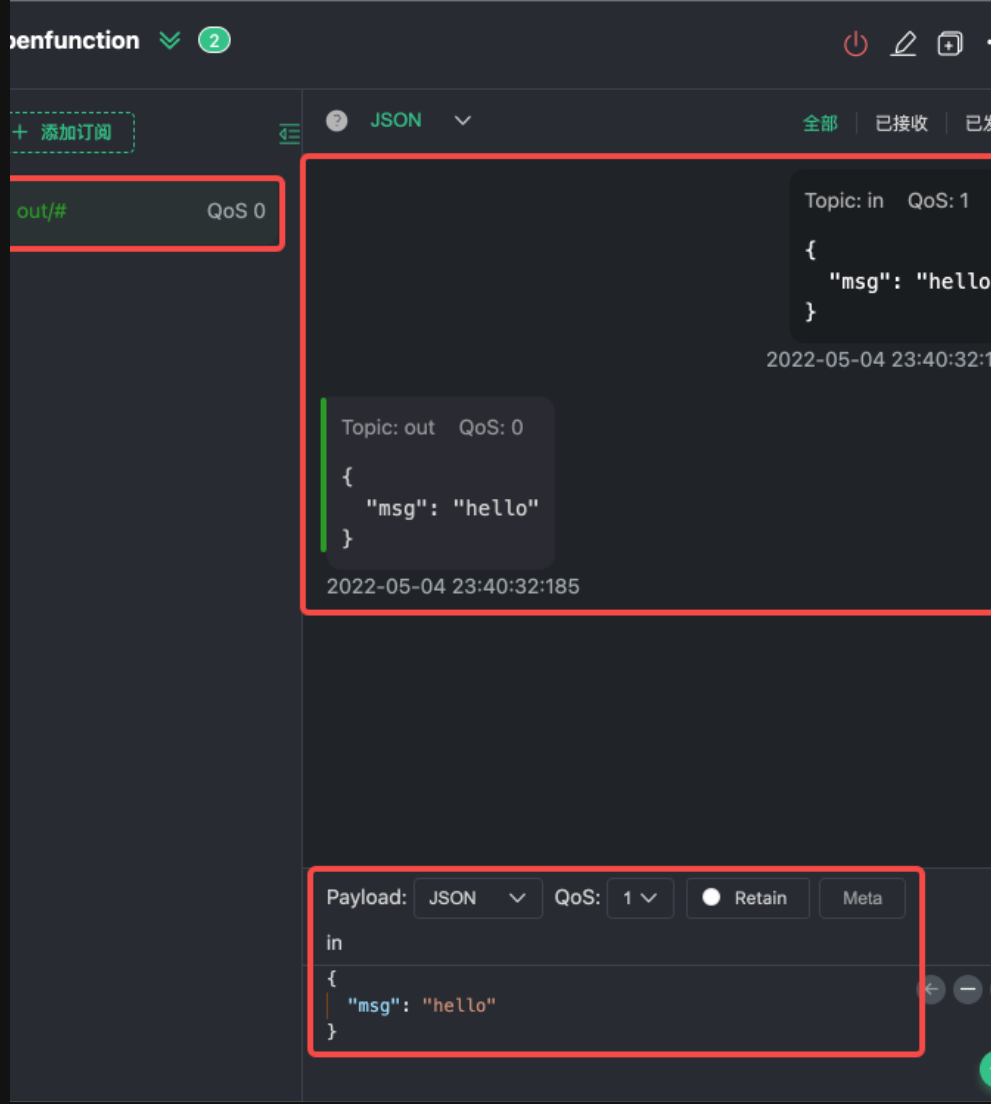
```
$ kubectl logs -c function serving-8f7xc-deployment-v200-l78xc-564c6b5bf7-vksg7
.. .
[Dapr-JS] Listening on 8080
[Dapr-JS] Letting Dapr pick-up the server (Maximum 60s wait time)
[Dapr-JS] - Waiting till Dapr Started (#0)
[Dapr-JS] Server Started
```

# Lab 1: Trigger Event

See also: [MQTT X desktop client](#)

- Connect EQMX server via NodePort mapped to `tcp:1883`
- Publish `{"msg": "hello"}` to `in` topic
  - Payload received in `out` topic 📌 ➡
- Check the `function` container log

```
$ kubectl logs -c function serving-8f7xc-deploym
...
[Dapr-JS] Listening on 8080
[Dapr-JS] Letting Dapr pick-up the server (Maxim
[Dapr-JS] - Waiting till Dapr Started (#0)
[Dapr-JS] Server Started
Data received: { msg: 'hello' }
```



```
apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-async-pubsub
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<tag>'
  serving:
    # default to knative
    runtime: async
    annotations:
      # default to "grpc"
      dapr.io/app-protocol: http
    template: ...
  params:
    # default to FUNC_NAME value
    FUNCTION_TARGET: tryAsync
  inputs:
    - name: mqtt-sub
      component: mqtt-pubsub
      topic: sub
  outputs:
    - name: mqtt-pub
      component: mqtt-pubsub
      topic: pub
```









```
apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-async-pubsub
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<tag>'
  serving:
    # default to knative
    runtime: async
    annotations:
      # default to "grpc"
      daprio.io/app-protocol: http
    template: ...
  params:
    # default to FUNC_NAME value
    FUNCTION_TARGET: tryAsync
  inputs:
    - name: mqtt-sub
      component: mqtt-pubsub
      topic: sub
  outputs:
    - name: mqtt-pub
      component: mqtt-pubsub
      topic: pub
```

```

apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-async-pubsub
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<tag>'
  serving:
    # default to knative
    runtime: async
    annotations:
      # default to "grpc"
      daprio.io/app-protocol: http
  template: ...
  params:
    # default to FUNC_NAME value
    FUNCTION_TARGET: tryAsync
  inputs:
    - name: mqtt-sub
      component: mqtt-pubsub
      topic: sub
  outputs:
    - name: mqtt-pub
      component: mqtt-pubsub
      topic: pub

```

```

pubsub:
  mqtt-pubsub:
    type: pubsub.mqtt
    version: v1
    metadata:
      - name: consumerID
        value: '{uuid}'
      - name: url
        value: tcp://admin:public@emqx:1883
      - name: qos
        value: 1

```

- Dapr Component - Pub/Sub Brokers - MQTT
- OpenFunction - Function CRD - DaprIO
  - `topic` field is required for pubsub component
- Check full sample codes

# Lab 2: MQTT Pub and Sub

- Apply function manifest, and check running states

```
$ kubectl apply -f async-pubsub.yaml
function.core.openfunction.io/sample-node-async-pubsub created

$ kubectl get fn
NAME                                BUILDSTATE    SERVINGSTATE    BUILDER    SERVING    URL    AGE
sample-node-async-pubsub           Skipped       Running                    serving-2qfkl    140m

$ kubectl get po
NAME                                READY    STATUS    RESTARTS    AGE
serving-2qfkl-deployment-v200-6cshf-57c8b5b8dd-ztmbf  2/2      Running    0           141m
```

- Furthermore, check whether `function` container output correct logs`

```
$ kubectl logs -c function serving-2qfkl-deployment-v200-6cshf-57c8b5b8dd-ztmbf
.. .
[Dapr-JS] Listening on 8080
[Dapr-JS] Letting Dapr pick-up the server (Maximum 60s wait time)
[Dapr-JS] - Waiting till Dapr Started (#0)
[Dapr API][PubSub] Registered 1 PubSub Subscriptions
[Dapr-JS] Server Started
```

# Lab 2: Trigger Event

- Connect EQMX server via NodePort mapped to `tcp:1883`
- Publish a CloudEvents event to `pub` topic
  - CloudEvents payload got in `sub` 🙌
  - Pure data recieved in async function 🙌
- Check the `function` container log

```
$ kubectl logs -c function serving-2qfkl-deploym
...
[Dapr-JS] Listening on 8080
[Dapr-JS] Letting Dapr pick-up the server (Maxim
[Dapr-JS] - Waiting till Dapr Started (#0)
[Dapr API][PubSub] Registered 1 PubSub Subscript
[Dapr-JS] Server Started
Data received: { orderId: '100' }
```

The screenshot shows the openfunction dashboard interface. At the top, there's a header with the 'openfunction' logo and a notification badge. Below the header, there's a section for subscriptions. A subscription named 'pub/#' is highlighted with a red box. To the right of this, a large red box contains the JSON payload for a CloudEvent. Below this, there's a section for the subscription's configuration, including 'Payload: JSON', 'QoS: 1', and 'Retain'. At the bottom, another red box shows the JSON payload for the 'sub' topic, which is a CloudEvent containing the order ID '100'.

openfunction 2

+ 添加订阅

out/# QoS 0

pub/# QoS 0

Topic: pub QoS: 0

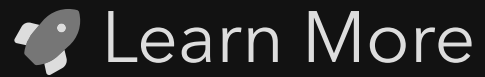
```
{
  "traceid": "00-65eb31f05c0e628b67aa591d8e65b441-7b74b1b1f22c6ee2-00",
  "specversion": "1.0",
  "datacontenttype": "application/json",
  "source": "sample-node-async-pubsub-openfunction",
  "type": "com.dapr.event.sent",
  "topic": "pub",
  "pubsubname": "serving-2qfkl-component-mqtt-pubsub-zwddr",
  "data": {
    "orderId": "100"
  },
  "id": "8a08f27e-b4e3-4cc6-a31f-0afbdb697fd4"
}
```

2022-08-05 13:33:55:446

Payload: JSON QoS: 1 Retain Meta

sub

```
{
  "specversion": "1.0",
  "type": "com.dapr.cloudevent.sent",
  "source": "testcloudeventspubsub",
  "subject": "Cloud Events Test",
  "id": "someCloudEventId",
  "time": "2021-08-02T09:00:00Z",
  "datacontenttype": "application/cloudevents+json",
  "data": {
    "orderId": "100"
  }
}
```



OpenFunction · Node.js Function Framework