Adventures in Apache OpenWhisk

Rob Allen

January 2019

Slides: https://akrabat.com/5507

Apache OpenWhisk

Apache OpenWhisk

- OpenSource
- http://openwhisk.org
- Incubator project working towards graduation
- Many contributors from multiple companies

Apache OpenWhisk

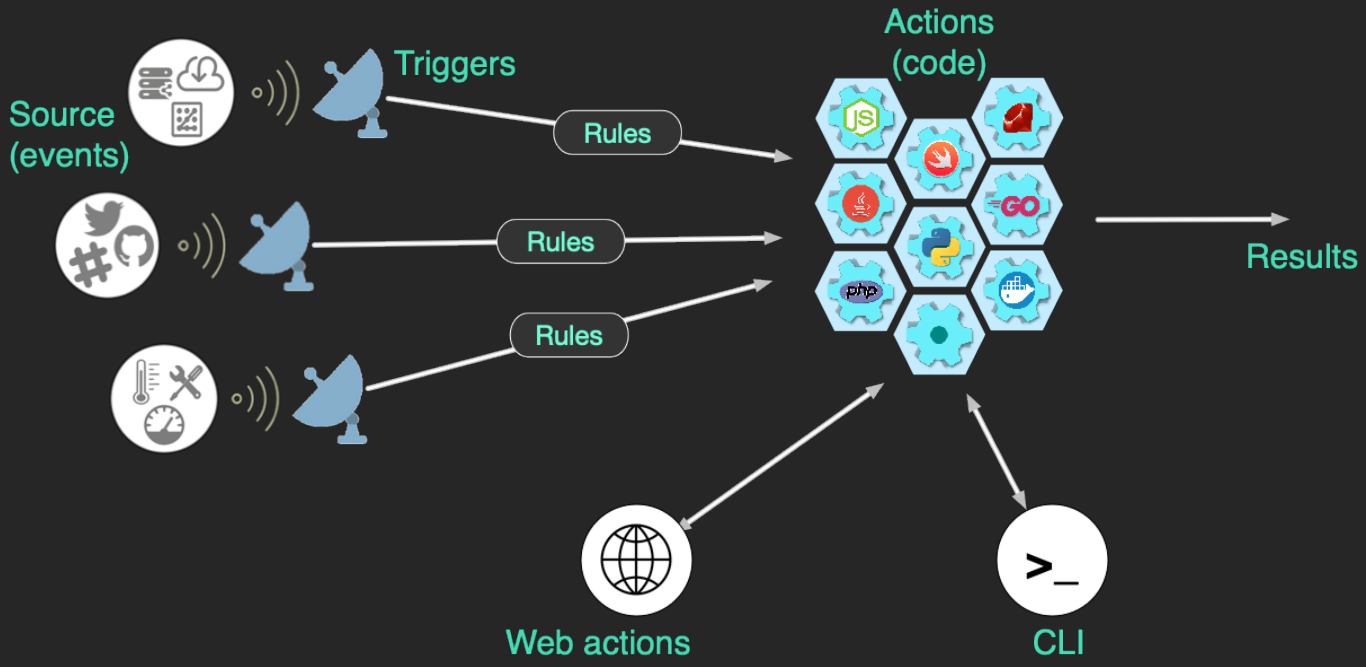
- Self-hosted. Deploy to:
  - Kubernetes
  - Mesos
  - OpenShift
  - Dev: Vagrant & Docker Compose
- Multiple public providers:
  - IBM
  - RedHat
  - Adobe (for Adobe Cloud Platform APIs)

Supported languages

- .NET Core
- Go
- NodeJS
- Python
- Swift
- Ballerina
- Java
- PHP
- Ruby

Also, your own Docker container can be deployed & invoked

Concepts


Source
(events)
Triggers
Rules
Rules
Rules
Actions
(code)
Results
Web actions
CLI
>_

Event providers (feeds)

- Alarms (scheduled tasks)
- CouchDB
- GitHub
- Kafka
- Push notifications
- RSS

Getting started with OpenWhisk

Rob Allen ~ @akrabat

Hello World

```
def main(args):
  name = args.get("name", "World")
  message = 'Hello {}!'.format(name)
  return {'body': message}
```

Hello World


```
def main(args):
  name = args.get("name", "World")
  message = 'Hello {}!'.format(name)
  return {'body': message}
```

Hello World

```
def main(args):
  name = args.get("name", "World")
  message = 'Hello {}!'.format(name)
  return {'body': message}
```

Hello World

```
def main(args):
  name = args.get("name", "World")
  message = 'Hello {}!'.format(name)
  return {'body': message}
```

Hello World

```
def main(args):
  name = args.get("name", "World")
  message = 'Hello {}!'.format(name)
  return {'body': message}
```

Deploy to OpenWhisk

```
$ zip -q hello.zip hello.py
```

Deploy to OpenWhisk

```
$ zip -q hello.zip hello.py
$ wsk action update --kind python:3.7 hello hello.zip
ok: updated action hello
```

Run it

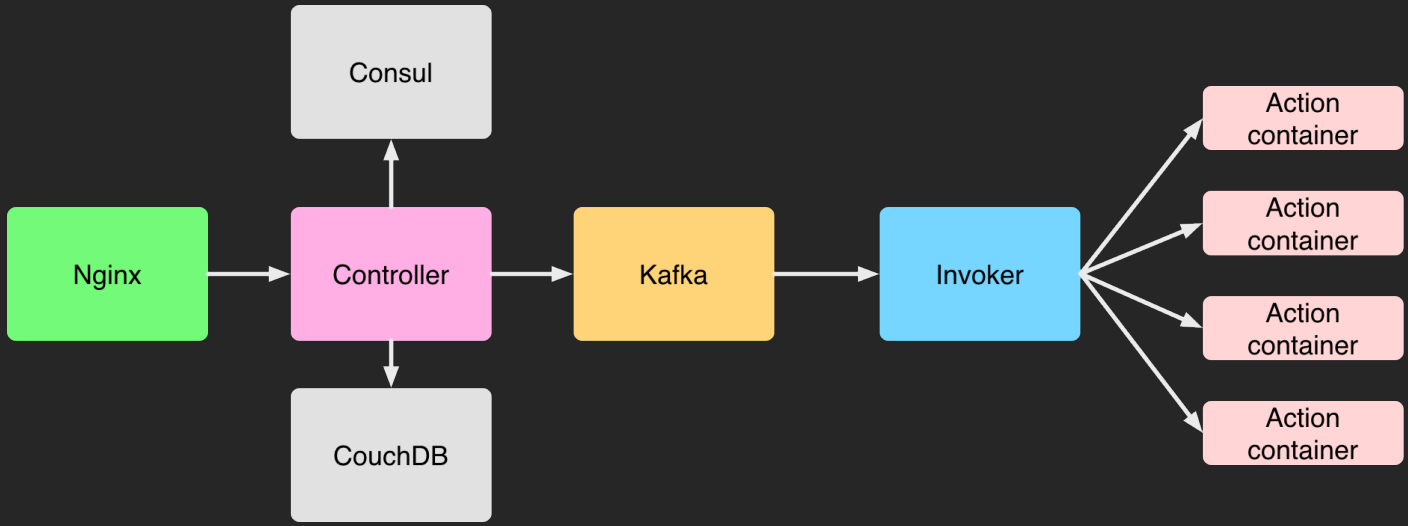
```
$ wsk action invoke hello --result --param name Rob
```

Deploying your action

```
$ wsk action invoke hello --result --param name Rob
{
    "body": "Hello Rob!"
}
```

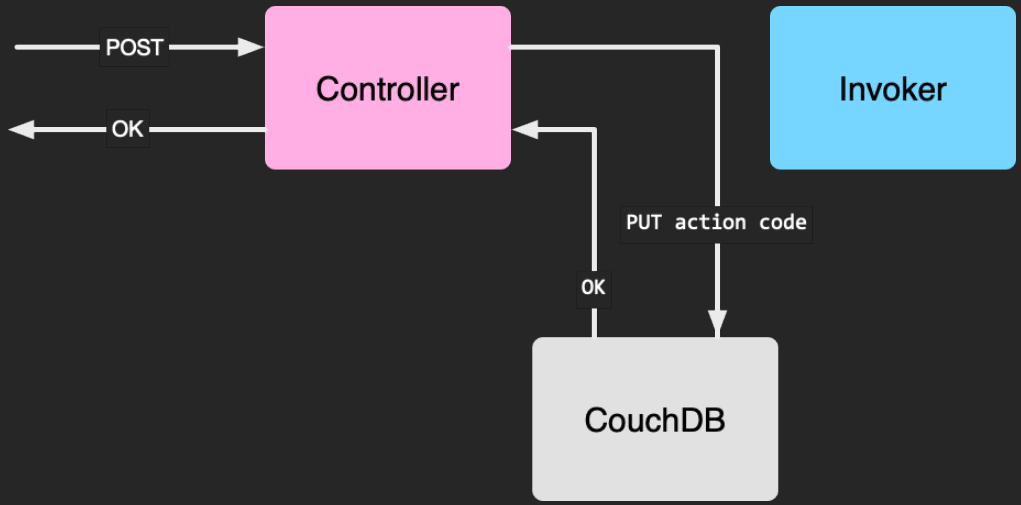
Segue: How did it do this?!

OpenWhisk's architecture



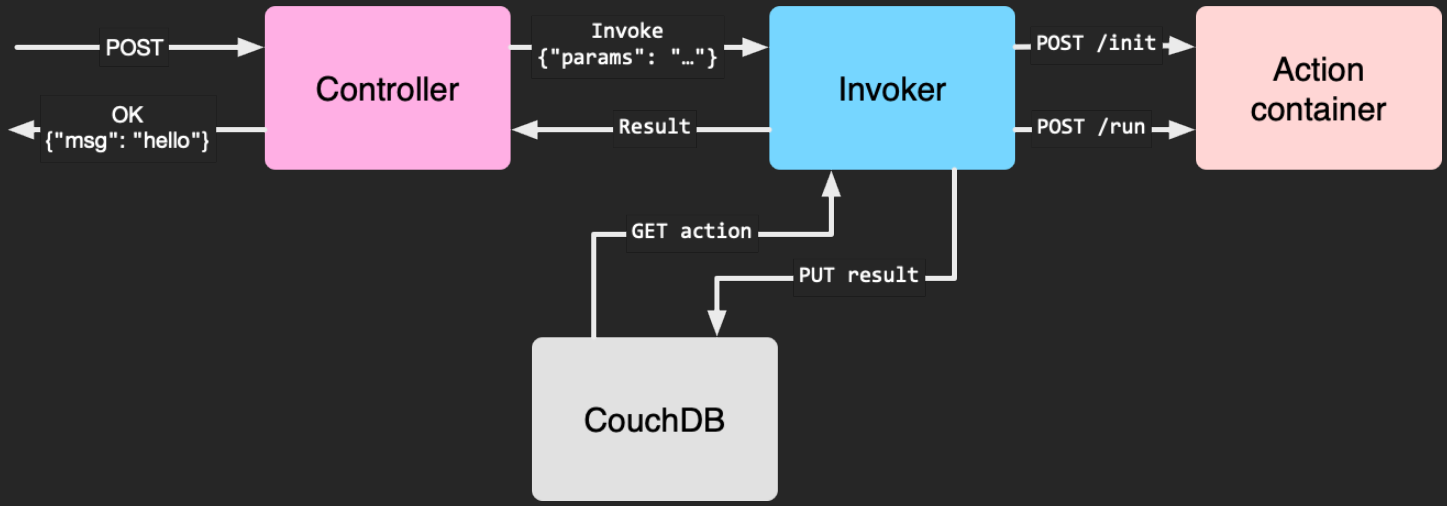
Create an action

```
$ wsk action create hello hello.py
```



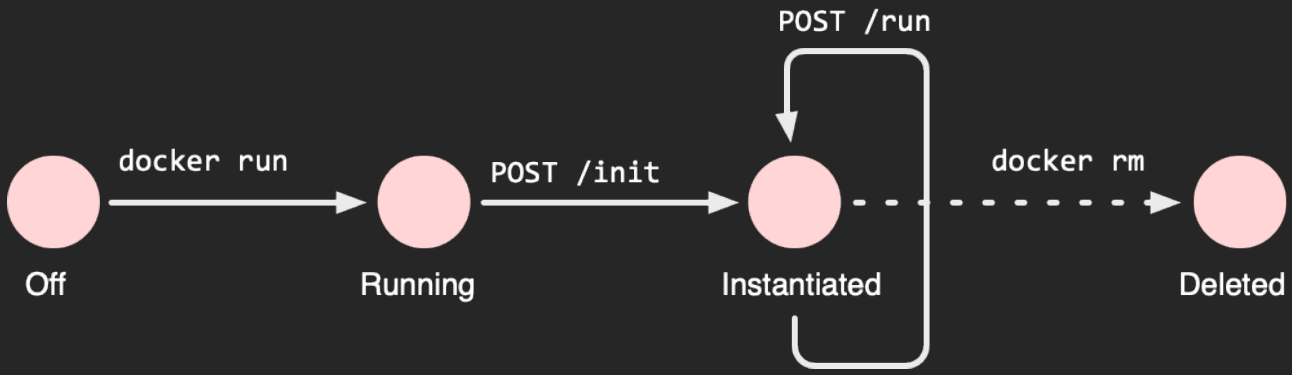
Invoke an action

```
$ wsk action invoke hello -r
```



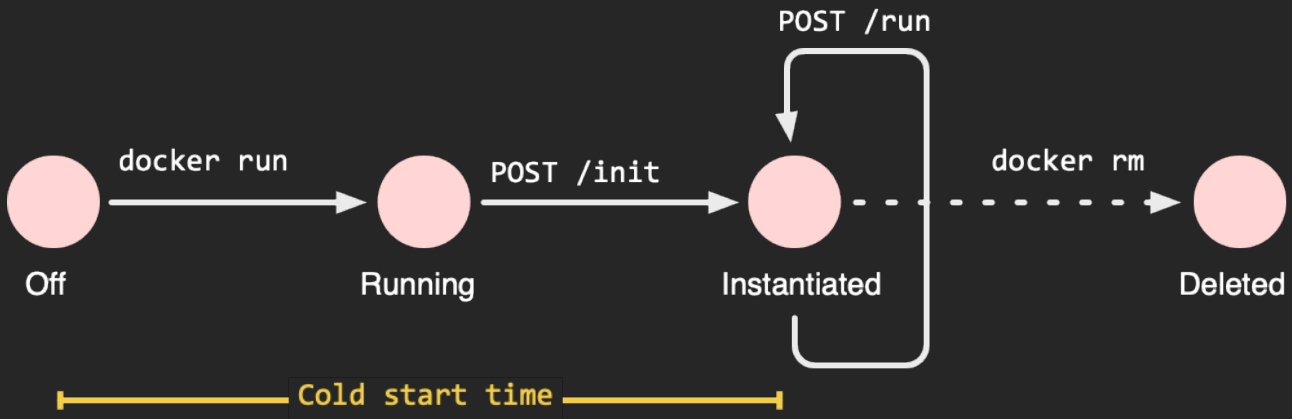
Action container lifecycle

- Hosts the user-written code
- Controlled via two end points: `/init` & `/run`



Action container lifecycle

- Hosts the user-written code
- Controlled via two end points: `/init` & `/run`



End Segue

Web-enabled actions

Access your action via HTTP: Perfect for web hooks

Add the `--web` flag:

```
$ wsk action update --kind python:3.7 --web true \
  demo/hello hello.zip
```

Web-enabled actions

Get URL and curl it:

```
$ wsk action get --url demo/hello
ok: got action hello
https://192.168.1.17/api/v1/web/guest/demo/hello

$ curl https://192…17/api/v1/web/guest/demo/hello?name=Rob
{
    "body": "Hello World!"
}
```

API Gateway

When you want to do more with HTTP endpoints

- Route endpoint methods to actions (Open API Spec support)
- Custom domains
- Rate limiting
- Security (API keys, OAuth, CORS)
- Analytics

API Gateway

API Gateway


```
$ wsk api create /myapp /hello GET hello
```

API Gateway


```
$ wsk api create /myapp /hello GET hello
ok: created API /myapp/hello GET for action /_/hello
```

API Gateway


```
$ curl https://example.com/myapp/hello?name=Rob
```

API Gateway


```
$ curl https://example.com/myapp/hello?name=Rob
{
  "message": "Hello Rob!"
}
```

Packages

- Group actions together
- Set parameters used by all actions

```
$ wsk package update demo
$ wsk action update --kind python:3.7 demo/hello hello.zip
```

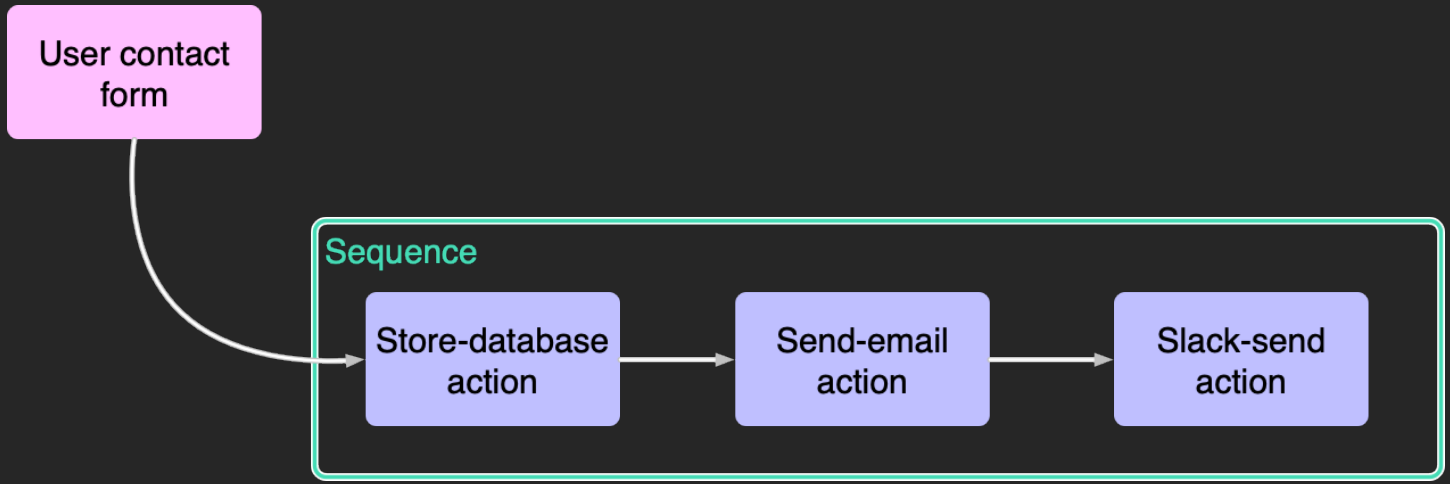
Built-in packages

- `alarms`
- `github`
- `kafka`
- `pushnotifications`
- `utils`
- `weather`
- `cloudant`
- `jira`
- `pushnotifications`
- `slack`
- `watson`
- `websocket`

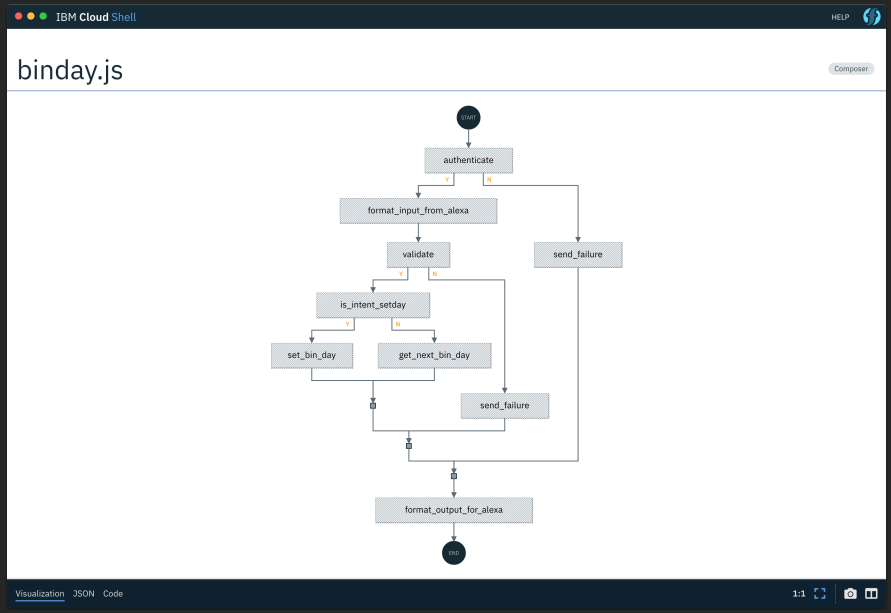
- & any other publicly shared package

Sequences

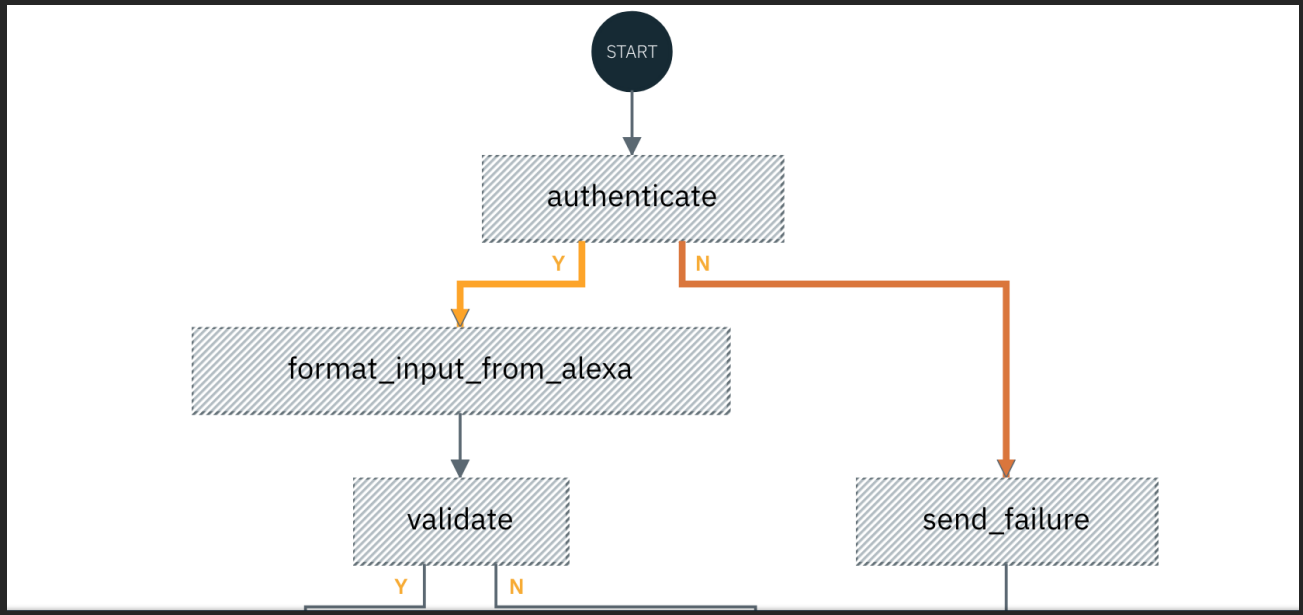
Invoke a set of actions in turn



Composer: Logic for your actions



Composer: Logic for your actions



Composer: It's just code

```
composer.sequence(
    composer.if(
        'binday/authenticate',
        composer.sequence(
          'format_input_from_alexa',
          composer.if(
              'binday/validate',
              'binday/get_next_bin_day'
              'binday/send_failure'
          ),
        ),
        'binday/send_failure'
    ),
    'binday/format_output_for_alexa'
)
```

Deployment: Serverless Framework

Set up:

```
$ npm install --global serverless serverless-openwhisk
$ serverless create --template openwhisk-php --path app
$ cd app
$ npm install
```

serverless.yml

```
service: ow-todo-backend

provider:
  name: openwhisk
  runtime: php

plugins:
  - serverless-openwhisk
```

serverless.yml

```yaml
functions:
  list-todos:
    handler: "src/actions/listTodos.main"
    name: "todo-backend/list-todos"
    events:
      - http:
          path: /todos
          method: get
  add-todo:
    handler: src/actions/addTodo.main
    # etc
```

Deploy

```
$ serverless deploy
Serverless: Packaging service...
Serverless: Compiling Functions...
Serverless: Compiling Packages...
Serverless: Compiling API Gateway definitions...
Serverless: Compiling Rules...
Serverless: Compiling Triggers & Feeds...
Serverless: Compiling Service Bindings...
Serverless: Deploying Packages...
Serverless: Deploying Functions...
Serverless: Deploying API Gateway definitions...
[...]
```

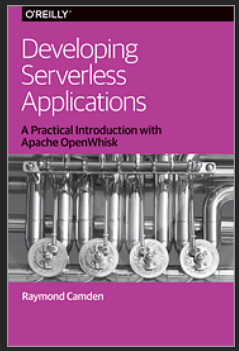
Resources

- http://www.openwhisk.org
- https://github.com/apache/incubator-openwhisk-workshop
- https://serverless.com/framework/docs/providers/openwhisk

Developing Serverless Applications

by Raymond Camden

Free via https://akrab.at/openwhiskbook



Thank you!