**VIETNAM NATIONAL UNIVERSITY HCM UNIVERSITY OF INFORMATION TECHNOLOGY**

**FACULTY OF COMPUTER NETWORKS AND COMMUNICATIONS**



**Ảnh có chứa màu trắng, thiết kế

Mô tả được tạo tự độngẢnh có chứa hình mẫu, Đồ họa, biểu tượng, Decal ô tô

Mô tả được tạo tự động**

**BÁO CÁO ĐỒ ÁN**

**NGHIÊN CỨU, TRIỂN KHAI APACHE OPENWHISK**

**GVHD: ThS. BÙI THANH BÌNH**

**Lớp: NT533.N21**

Nhóm 10

Diệc Thành – 20521909

Mai Phước Sang – 20520735

Trần Triệu Thiên – 20521954

*Thành phố Hồ Chí Minh, ngày 05 tháng 05 năm 2023*

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

*……., ngày……...tháng……năm 2023*

**Người nhận xét**

*(Ký tên và ghi rõ họ tên****)***

NỘI DUNG

[NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN 1](#_Toc135049287)

[A. DANH MỤC CÁC BẢNG, HÌNH VÀ BIỂU ĐỒ 3](#_Toc135049288)

[B. CƠ SỞ LÝ THUYẾT 3](#_Toc135049289)

[I) Khái niệm căn bản 3](#_Toc135049290)

[1) FaaS 3](#_Toc135049291)

[2) Apache Openwhisk 4](#_Toc135049292)

[II) Cơ chế và mô hình 5](#_Toc135049293)

[1) Cơ chế 5](#_Toc135049294)

[2) Mô hình 6](#_Toc135049295)

[3) Thực thi 8](#_Toc135049296)

[C. THAM KHẢO, TRÍCH DẪN 9](#_Toc135049297)

# DANH MỤC CÁC BẢNG, HÌNH VÀ BIỂU ĐỒ

|  |  |  |
| --- | --- | --- |
| **Hình** | **Tên bảng / biểu đồ / hình** | **Trang** |
| 1 | Sơ đồ cơ chế của Apache OpenWhisk | 5 |
| 2 | Mô hình của Apache OpenWhisk | 6 |
| 3 | Dịch vụ cần thiết của Apache OpenWhisk | 8 |

# CƠ SỞ LÝ THUYẾT

## Khái niệm căn bản

### FaaS

#### Định nghĩa

* Function as a Service, hay còn được gọi là FaaS, là một cách serverless để chạy function trong môi trường đám mây. Với sự giúp đỡ của FaaS, các developer có thể tập trung function mà không cần quan tâm đến phía Server như deploy, duy trì cơ sở hạ tầng, hệ điều hành hay bảo mật. Thay vào đó, nền tảng serverless sẽ tự động quản lý các tài nguyên phía máy chủ, chỉ chạy function khi được gọi và dừng khi hoàn thành. FaaS được sử dụng chủ yếu trong bối cảnh tính toán sự kiện, trong đó các hàm được kích hoạt bởi một sự kiện cụ thể như message queue, các HTTP request, vv.
* Nhiều nền tảng hỗ trợ FaaS đã xuất hiện trên thị trường, ví dụ như AWS Lambda, Azure Functions, Google Cloud Functions và nhiều hơn nữa. Những nền tảng này hỗ trợ nhiều ngôn ngữ lập trình, cung cấp các tiện ích để phát triển, thử nghiệm và triển khai các function.[1]

#### Ưu điểm

* Tốc độ – Improved developer velocity: Với FaaS, lập trình viên có thể có thêm thời gian để viết về logic của ứng dụng, bớt thời gian lo lắng về servers và deploys. Điều đó cũng đồng nghĩa với việc development nhanh hơn.
* Khả năng mở rộng – Scability: Dev không còn phải lo vấn đề về quá tải khi traffic cao/ Server sẽ cung cấp và handle tất cả các vấn đề liên quan với scaling.
* Giá cả – Cost efficiency: Khách hàng chỉ trả tiền cho thời gian mà họ sử dụng, không phải phung phí tiền cho các tài nguyên cloud khác.[2]

#### Nhược điểm

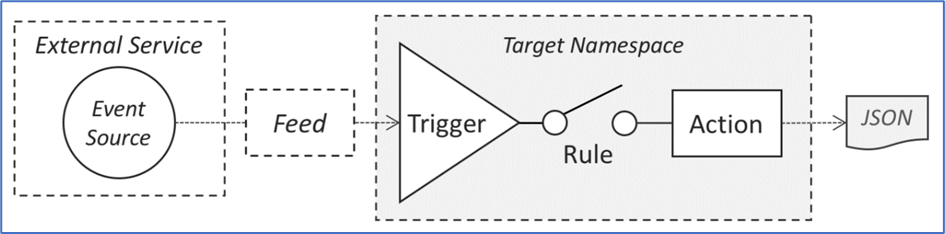
* Giới hạn chức năng: Các function trong FaaS là các đoạn mã nhỏ được sử dụng để thực hiện các nhiệm vụ đơn giản (Function do only one thing). Nếu bạn muốn viết các hàm phức tạp, FaaS có thể không phải là lựa chọn tốt nhất của bạn.
* Khó khăn trong quản lý chức năng: Vì bạn sẽ phát triển các đoạn mã chức năng nhỏ, việc quản lý một số lượng lớn các chức năng có thể gặp khó khăn khi doanh nghiệp của bạn tiếp tục phát triển.
* Hiệu năng thấp: vì mỗi request cần phải có thời gian để start.[2]

### Apache Openwhisk

* OpenWhisk là 1 nền tảng Serverless phân tán mã nguồn mở, thực thi các chức năng phản hồi các Event ở mọi quy mô, được sử dụng bởi IBM Cloud.
* OpenWhisk quản lý cơ sở hạ tầng, các máy chủ và mở rộng quy mô bằng các Docker container để người dùng có thể tập trung xây dựng các ứng dụng hiệu quả.
* Nền tảng OpenWhisk hỗ trợ 1 mô hình lập trình trong đó các nhà phát triển viết logic mang chức năng (được gọi là Actions), bằng bất kỳ ngôn ngữ lập trình nào được hỗ trợ, có thể được lên lịch động và chạy để phản hồi các Event (thông qua các Trigger) từ các nguồn bên ngoài (Feeds) hoặc từ các yêu cầu HTTP.
* Một dự án bao gồm một Giao diện dòng lệnh (CLI – Command Line Interface) dựa trên REST API cùng với các công cụ khác hỗ trợ đóng gói, các dịch vụ danh mục và nhiều tùy chọn triển khai container phổ biến.[3]

## Cơ chế và mô hình

### Cơ chế



Hình 1. Sơ đồ cơ chế của Apache OpenWhisk

* Event Source: các sự kiện được tạo ra từ những dịch vụ thường chỉ ra những thay đổi trong dữ liệu hoặc tự mang dữ liệu. 1 số ví dụ về nguồn sự kiện phổ biến:
  + Tin nhắn đến trên hàng đợi tin nhắn
  + Thay đổi trong cơ sở dữ liệu
  + Thay đổi trong Document Store
  + Tương tác trang web hoặc ứng dụng web
  + Các dịch vụ API đang được gọi
  + Các framework IoT chuyển tiếp dữ liệu cảm biến thiết bị
* Actions: là các chức năng phi trạng thái (đoạn mã) chạy trên nền tảng OpenWhisk. Các Action đóng gói logic ứng dụng để được thực thi đáp ứng các Event. Các Action có thể được gọi thủ công bằng OpenWhisk REST API, OpenWhisk CLI, các API đơn giản và do người dùng tạo hoặc được tạo tự động thông qua các Trigger.
* Triggers: là các kênh được đặt tên cho các lớp hoặc loại sự kiện được gửi từ Nguồn sự kiện (Event Source).
* Rules: được sử dụng để liên kết 1 trigger với 1 action. Sau khi loại liên kết này được tạo, mỗi khi 1 event kích hoạt thì hành động tương ứng sẽ được gọi. [4]
* Cơ chế hoạt động: OpenWhisk dựa trên kiến trúc hướng sự kiện trong đó hầu hết các Action được thực thi khi các event diễn ra. Bản thân Trigger được kích hoạt với 1 từ điển các cặp khóa-giá trị (các tham số) đến từ Nguồn sự kiện và cho phép định cấu hình các giá trị mặc định tùy chọn, nhờ đó giúp đảm bảo dữ liệu tương thích với mọi Action được liên kết. Các Rule cho phép cùng 1 Trigger được liên kết với nhiều Action, cũng như cho phép bật hoặc tắt tự động hóa cụ thể 1 cách linh hoạt mà không phá hủy định nghĩa Trigger.

### Mô hình

A picture containing text, screenshot, diagram, design

Description automatically generated

Hình 2. Mô hình của Apache OpenWhisk

* Apache OpenWhisk, là một nền tảng serverless cloud mã nguồn mở. Nó hoạt động bằng cách thực thi các function (hay action) đáp lại các event. Các event có thể xuất phát từ nhiều nguồn khác nhau, bao gồm hẹn giờ, cơ sở dữ liệu, message queue hoặc các trang web như Slack hoặc GitHub.
* OpenWhisk chấp nhận source code làm đầu vào để thực thi một lệnh đơn với giao diện dòng lệnh (CLI), sau đó vận chuyển các dịch vụ thông qua web tới nhiều consumer, chẳng hạn như các trang web khác, các ứng dụng di động hoặc các dịch vụ dựa trên REST APIs. [5]
* Các thành phần:

#### Function và Event

* Trong OpenWhisk, các function được sử dụng để hoàn thành các tác vụ. Một hàm thường là một đoạn mã nhận một số input và phản hồi lại với output. Điều quan trọng cần lưu ý ở đây là sự stateless của function.
* Nếu thiết kế 1 fuction là statefull, nó giới hạn khả năng scaling của hệ thống vì sẽ cần 1 storage cho CSDL. Quan trọng hơn, hệ thống sẽ cần một công cụ để đồng bộ trạng thái giữa các lần kích hoạt làm chi phí đắt đỏ hơn.
* Môi trường OpenWhisk quản lý cơ sở hạ tầng,và waiting cho 1 event xảy ra, khi đó function sẽ được gọi. Xử lý event là hoạt động quan trọng nhất mà môi trường serverless quản lý. Các function được viết để xử lý và phản hồi đúng khi event xảy ra - ví dụ, yêu cầu từ người dùng hoặc sự xuất hiện của dữ liệu mới.

[5]

#### Action và cấu tạo của Action

* Một action là một đoạn mã, được viết bằng một trong các ngôn ngữ lập trình được hỗ trợ. Khi gọi , action sẽ nhận một số parameter dưới dạng input.
* Để tiêu chuẩn hoá việc truyền parameter giữa nhiều ngôn ngữ lập trình, OpenWhisk sử dụng định dạng JavaScript Object Notation (JSON) vì nó khá đơn giản và có thư viện để mã hóa và giải mã định dạng này có sẵn cho hầu hết các ngôn ngữ lập trình.
* Các parameter được truyền cho các action trong các đối tượng JSON ,được xem như 1 string sẽ được action nhận vào khi bắt đầu và xử lý. Kết thúc quá trình xử lý, mỗi action phải trả vềmột kết quả, dưới dạng giá trị JSON.

[5]

#### Dịch vụ cần thiết

* OpenWhisk được xây dựng trên nền tảng các dự án mã nguồn mở phổ biến:
  + - * Nginx: một proxy reverse và là một web server hiệu suất cao.
      * CouchDB: cơ sở dữ liệu NoSQL document-oriented có khả năng mở rộng dễ dàng.
      * Kafka: hệ thống message publish/subscribe phân tán, hiệu suất cao.
* Ngoài ra, OpenWhisk còn bao gồm các thành phần chính sau:
  + - * Controller: quản lý các thực thể, xử lý các chuỗi trigger và định hướng các action.
      * Invoker: khởi chạy các container để thực thi các action.
      * Action Containers: thực hiện các action.

[5]

Ảnh có chứa văn bản, biểu đồ, Song song, hàng

Mô tả được tạo tự động

Hình 3. Dịch vụ cần thiết của Apache OpenWhisk

### Thực thi

Để thực thi một hành động trong OpenWhisk, các bước sau được thực hiện nội bộ:

* Người dùng gửi request đến OpenWhisk để kích hoạt một action. Action bao gồm tên action, các tham số và bất kỳ thông tin xác thực nào cần thiết.
* Controller của OpenWhisk nhận request và xác thực thông tin xác thực của người dùng. Sau đó, nó kiểm tra cơ sở dữ liệu của hệ thống (CouchDB) để tìm action đã chỉ định.
* Sau khi Controller đã tìm thấy action, nó kiểm tra xem liệu nó có cần được khởi tạo không. action chưa được khởi tạo, Controller tạo một instance cho action và khởi tạo nó với các dependency cần thiết.
* Nếu action đã được khởi tạo, Controller gửi yêu cầu đến Invoker để thực thi action.
* Action nhận yêu cầu và xử lý nó theo code đã được lập trình. Action đọc parameter được truyền dưới dạng JSON và tạo ra một response trong định dạng JSON.
* Sau khi hoàn tất việc thực thi, container trả lại kết quả cho Invoker.
* Invoker trả về kết quả cho Controller.
* Controller trả về kết quả cho người dùng.
* Tổng thể, quá trình thực thi là sự kết hợp của nhiều lớp và thành phần hoạt động cùng nhau một cách trơn tru để cung cấp cho end user một trải nghiệm đơn giản.

# THAM KHẢO, TRÍCH DẪN

[1] “Serverless vs. Function-as-a-Service (FaaS): Which One to Choose?” https://kubesphere.io/blogs/serverless-vs-faas/ (accessed May 12, 2023).

[2] “Tìm hiểu về Serverless, FaaS, deploy functions lên Firebase Cloud Functions,” Sep. 18, 2019. https://viblo.asia/p/tim-hieu-ve-serverless-faas-deploy-functions-len-firebase-cloud-functions-yMnKMw0El7P (accessed May 12, 2023).

[3] “Documentation.” https://openwhisk.apache.org/documentation.html (accessed May 12, 2023).

[4] “Build a serverless app with Node.js and OpenWhisk - LogRocket Blog.” https://blog.logrocket.com/build-serverless-app-node-js-openwhisk/ (accessed May 12, 2023).

[5] “1. Serverless and OpenWhisk Architecture - Learning Apache OpenWhisk [Book].” https://www.oreilly.com/library/view/learning-apache-openwhisk/9781492046158/ch01.html (accessed May 12, 2023).