

NGHIÊN CỨU VÀ XÂY DỰNG ỨNG DỤNG QUẢN LÝ TASK VỚI MÔ HÌNH SERVERLESS SERVICE

GVHD: ThS. Đinh Nguyễn Anh Dũng



Nội dung

1. Giới thiệu đề tài
2. Tổng quan
3. Cơ sở lý thuyết

4. Nguyên lý hoạt động
5. Kết quả đạt được
6. Demo ứng dụng



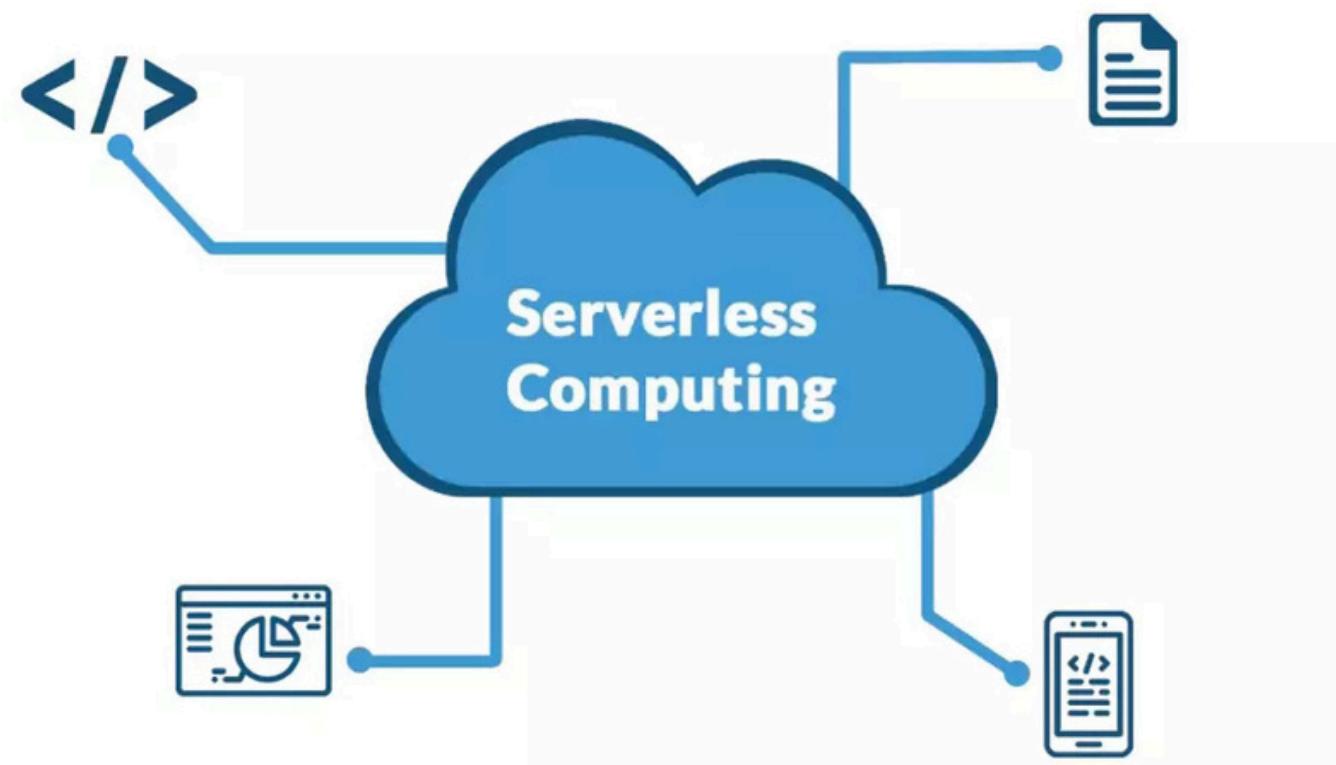
Giới thiệu

Giới thiệu đề tài

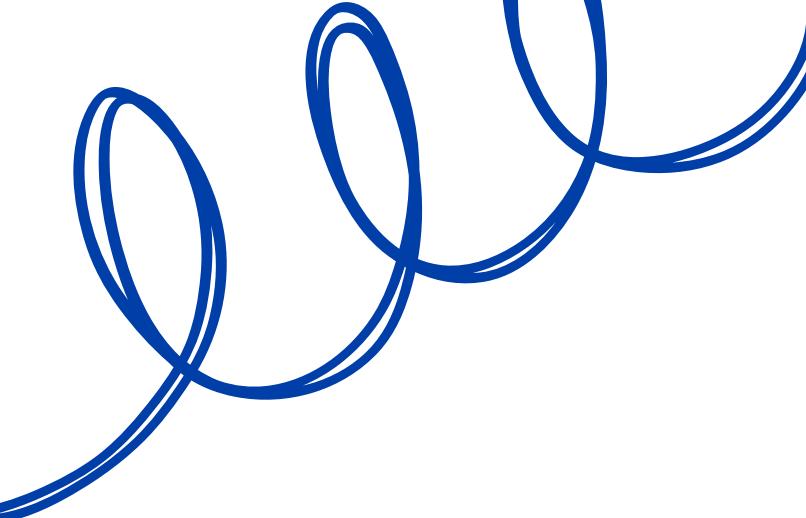
Việc nghiên cứu mô hình Serverless thông qua Firebase mang ý nghĩa quan trọng, đặc biệt trong bối cảnh nhu cầu cầu xây dựng các ứng dụng hiện đại, hiệu quả và dễ bảo trì ngày càng tăng. Thông qua đề tài nghiên cứu này, nhóm chúng em sẽ không chỉ tìm hiểu lý thuyết về mô hình Serverless mà còn áp dụng thực tế bằng cách xây dựng một ứng dụng quản lý công việc dựa trên các dịch vụ của Firebase. Ứng dụng này sẽ giúp giải quyết bài toán về quản lý công việc, bao gồm tạo nhiệm vụ, phân công công việc, theo dõi tiến độ.

Tổng quan

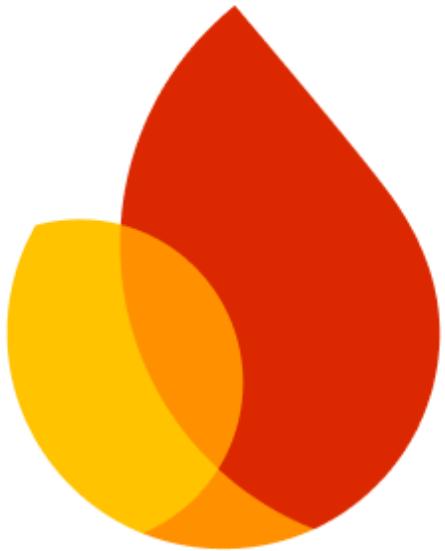
Serverless



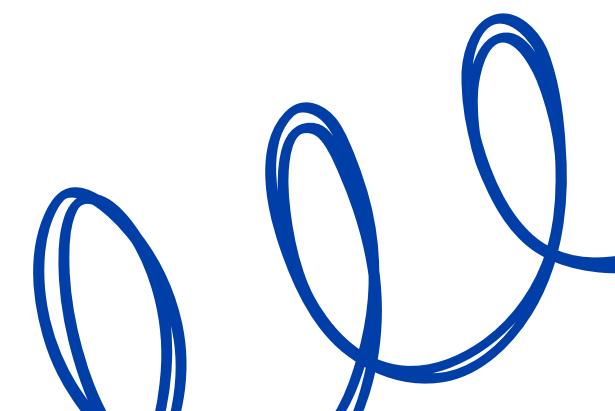
- Serverless là một mô hình kiến trúc điện toán đám mây hiện đại, trong đó các nhà phát triển tập trung hoàn toàn vào việc viết mã và triển khai ứng dụng mà không cần quản lý cơ sở hạ tầng vật lý hoặc máy chủ.
- Thay vào đó, nhà cung cấp dịch vụ đám mây (như AWS, Google Cloud, Azure) đảm nhận mọi công việc liên quan đến vận hành, mở rộng, và bảo trì hạ tầng.



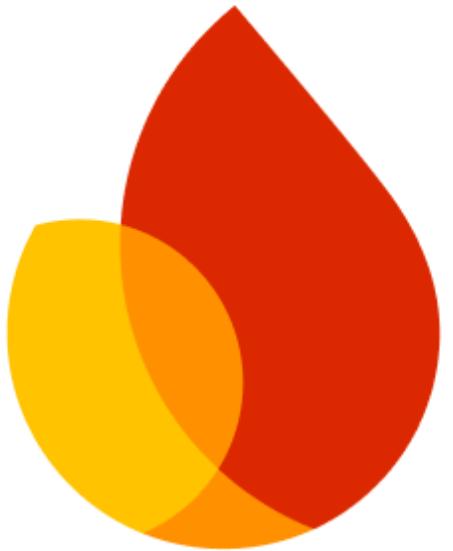
Firebase



Firebase

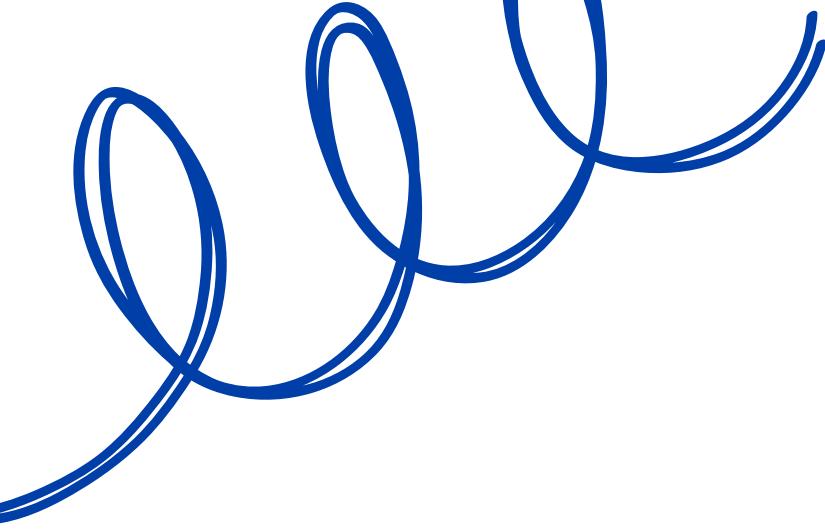
- **Firebase là dịch vụ Serverless của google**
 - **Tích hợp nhiều dịch vụ để xây dựng và vận hành ứng dụng nhanh chóng.**
 - **Có gói miễn phí khi người dùng mới sử dụng và dễ tiếp cận**
 - **Hỗ trợ vừa FaaS (Function as Service) và BaaS (Backend as Service)**
- 

Firebase



Firebase

- **Có các dịch vụ BaaS giúp dễ dàng tạo ứng dụng gồm: Firebase Firestore, Firebase Authentication**
- **Kết hợp với dịch vụ FaaS giúp xử lý các API thông qua Firebase Function**



AWS Lambda



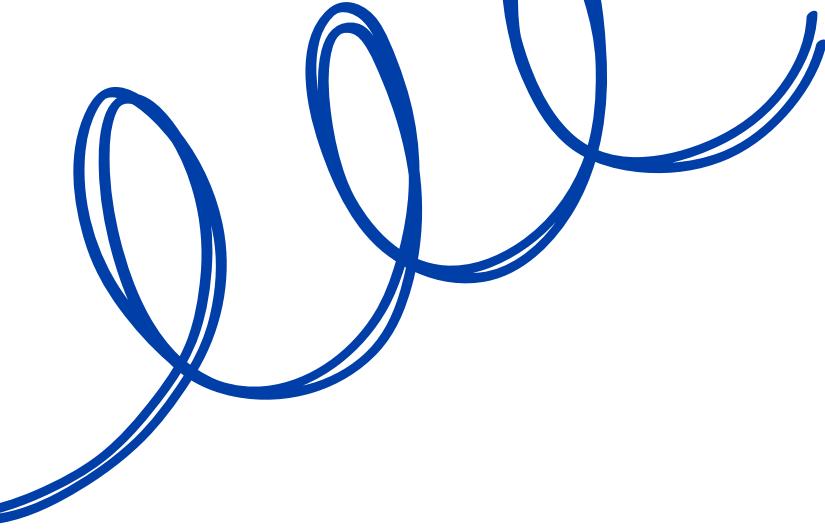
aws Lambda

- **Là dịch vụ serverless cho phép chạy mã không cần quản lý máy chủ**
- **Hỗ trợ nhiều ngôn ngữ: Node.js, Python, Java, Go**
- **Tính phí theo số lượng request và thời gian thực thi**
- **Tự động mở rộng để xử lý nhiều request đồng thời**

Google Cloud Functions

- Là dịch vụ serverless của Google Cloud Platform
- Hỗ trợ các ngôn ngữ: JavaScript, Python, Go
- Tích hợp tốt với các dịch vụ Google Cloud khác như Storage và Pub/Sub
- Phù hợp xử lý HTTP requests và các sự kiện đám mây
- Tự động mở rộng theo nhu cầu sử dụng





Azure Function



Azure Functions

- Là dịch vụ serverless của Microsoft Azure
- Hỗ trợ nhiều ngôn ngữ: C#, Java, Python, JavaScript
- Tích hợp tốt với các dịch vụ Azure như Storage, Event Grid, Service Bus
- Linh hoạt trong việc mở rộng từ tác vụ đơn giản đến phức tạp
- Tính phí theo số lượng request và thời gian chạy

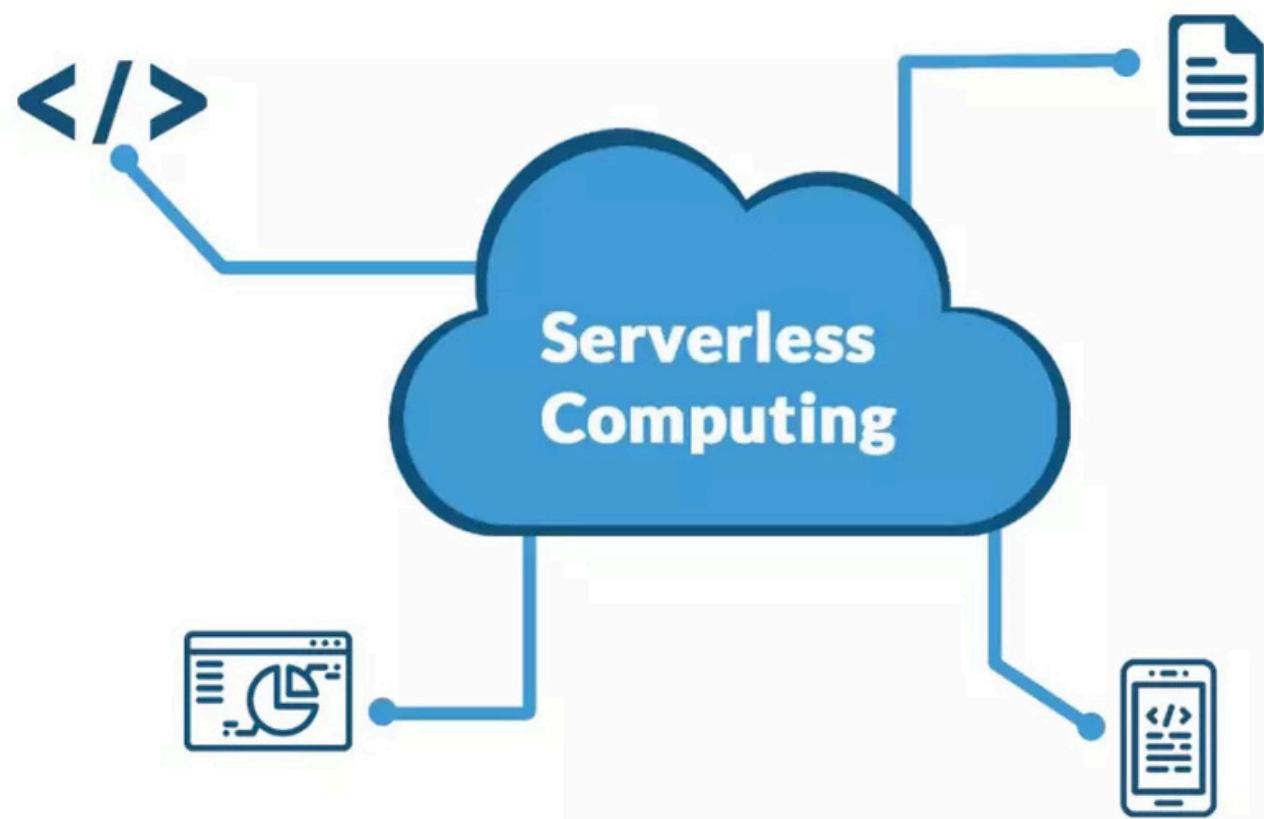
Firebase Cloud Functions

- Là dịch vụ serverless tích hợp trong nền tảng Firebase
- Tích hợp tốt với các dịch vụ Firebase như Realtime Database, Authentication, Cloud Messaging
- Phù hợp cho ứng dụng web và di động dùng Firebase
- Triển khai nhanh chóng và dễ dàng



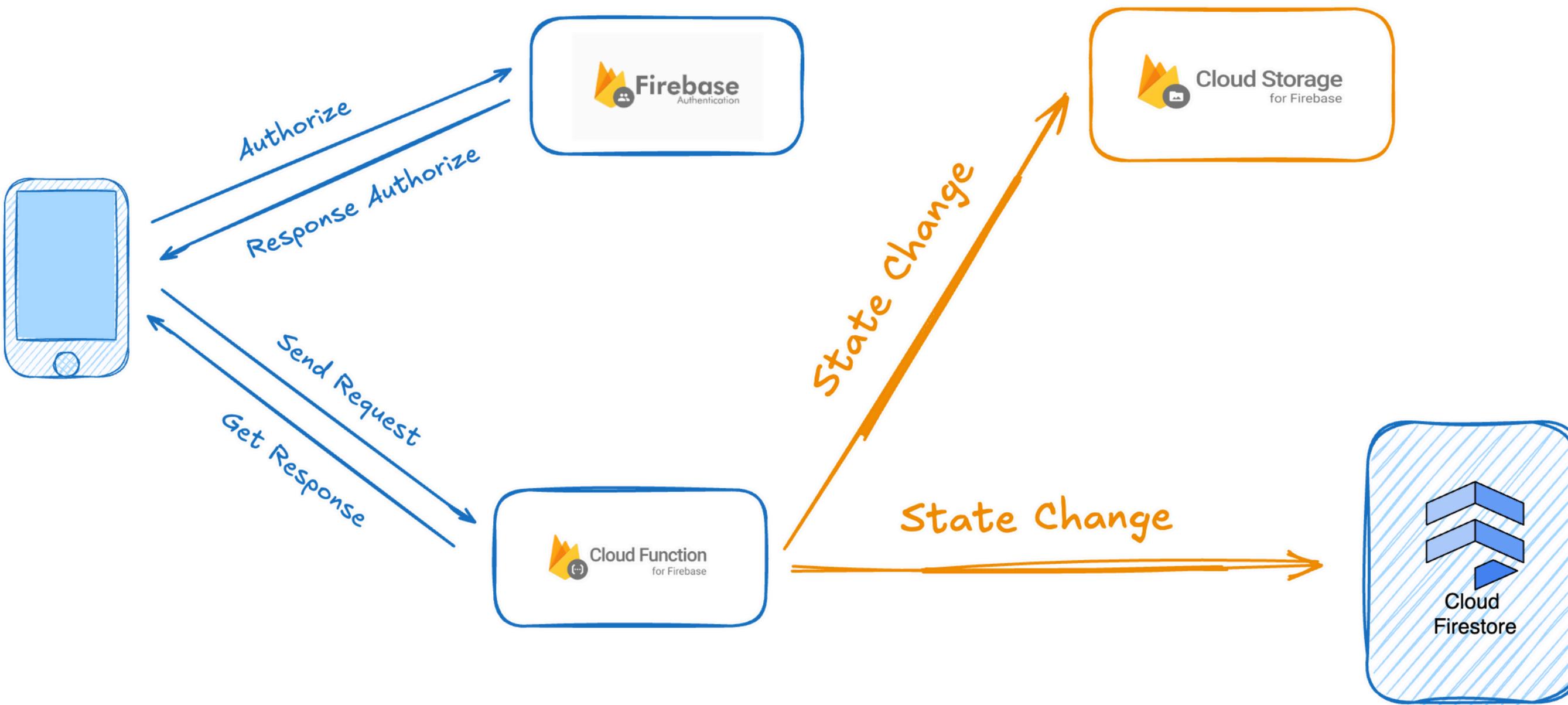
Cơ sở lý thuyết

Kiến trúc serverless



- Mặc dù tên gọi là **serverless**, điều này không có nghĩa là không có máy chủ để vận hành hệ thống.
- Thay vào đó, **serverless** đề cập đến việc các máy chủ đã được quản lý hoàn toàn bởi các nhà cung cấp dịch vụ đám mây.
- **Kiến trúc serverless** xây dựng dựa trên các function nhỏ và được gọi đến khi được kích hoạt event

Kiến trúc serverless

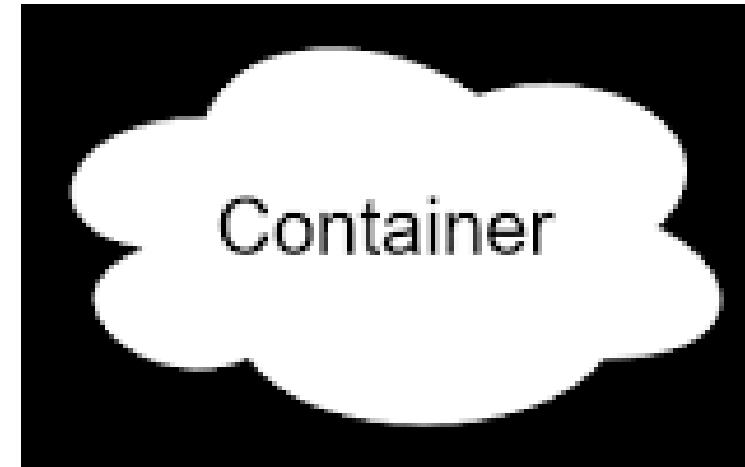




Serveless và kiến trúc khác



Micro - service



Container Architecture



So với Micro-service

Microservice và serverless đều có điểm chung là các thành phần hoạt động độc lập: trong microservice, các service không ảnh hưởng lẫn nhau khi gặp sự cố, tương tự, các function trong serverless cũng triển khai và tồn tại riêng lẻ.



So với Micro-service

Tiêu chí	Microservices	Serverless
Cách triển khai	Triển khai trên các máy chủ riêng lẻ	Triển khai lên nền tảng cloud computing Faas
Quản lý hạ tầng	Quản lý container các service	Nền tảng cloud Computing đảm nhiệm
Khả năng mở rộng	Mở rộng theo dịch vụ (manual hoặc auto-scaling).	Mở rộng tự động theo từng hàm dựa trên số lượng yêu cầu.
Thời gian phát triển	Cần nhiều thời gian để thiết lập, triển khai và tối ưu.	Tập trung vào logic ứng dụng, giảm thời gian thiết lập hạ tầng.
Trường hợp sử dụng phổ biến	Ứng dụng lớn, có các phần độc lập và yêu cầu tương tác phức tạp.	Ứng dụng nhỏ, quy mô vừa, yêu cầu xử lý nhanh và không liên tục.



So với Container Architecture

Tiêu chí	Container Architecture	Serverless Architecture
Quản lý môi trường	Lập trình viên tự duy trì hệ điều hành, runtime và container.	Dịch vụ cloud computing đảm nhận toàn bộ việc bảo trì.
Mở rộng	Yêu cầu sử dụng nền tảng như Kubernetes để mở rộng.	Tự động mở rộng dựa trên số lượng yêu cầu.
Ứng dụng phù hợp	Ứng dụng có lưu lượng truy cập cao liên tục hoặc yêu cầu kiểm soát chi tiết.	Ứng dụng dựa trên sự kiện, tác vụ ngắn hạn hoặc không thường xuyên.

Nguyên lý hoạt động

Tạo và deploy Firebase Function

B1: Tạo Project Firebase CLI

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

D:\Data\UIT\mk7\se400\Project\TestRequest\firebase_function> npx firebase init functions

D:\Data\UIT\mk7\se400\Project\TestRequest\firebase_function

? Are you ready to proceed? Yes

*** Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running `firebase use --add`,
but for now we'll just set up a default project.

? Please select an option: Use an existing project
? Select a default Firebase project for this directory: rhythm-party (Rhythm Party)
1 Using project rhythm-party (Rhythm Party)

*** Functions Setup

Let's create a new codebase for your functions.
A directory corresponding to the codebase will be created in your project
with sample code pre-configured.

See <https://firebase.google.com/docs/functions/organize-functions> for
more information on organizing your functions using codebases.

Functions can be deployed with `firebase deploy`.

? What language would you like to use to write Cloud Functions? TypeScript
? Do you want to use ESLint to catch probable bugs and enforce style? Yes
+ Wrote functions/package.json
+ Wrote functions/.eslintrc.js
+ Wrote functions/tsconfig.json
+ Wrote functions/tsconfig.dev.json
+ Wrote functions/src/index.ts
+ Wrote functions/.gitignore
? Do you want to install dependencies with npm now? Yes

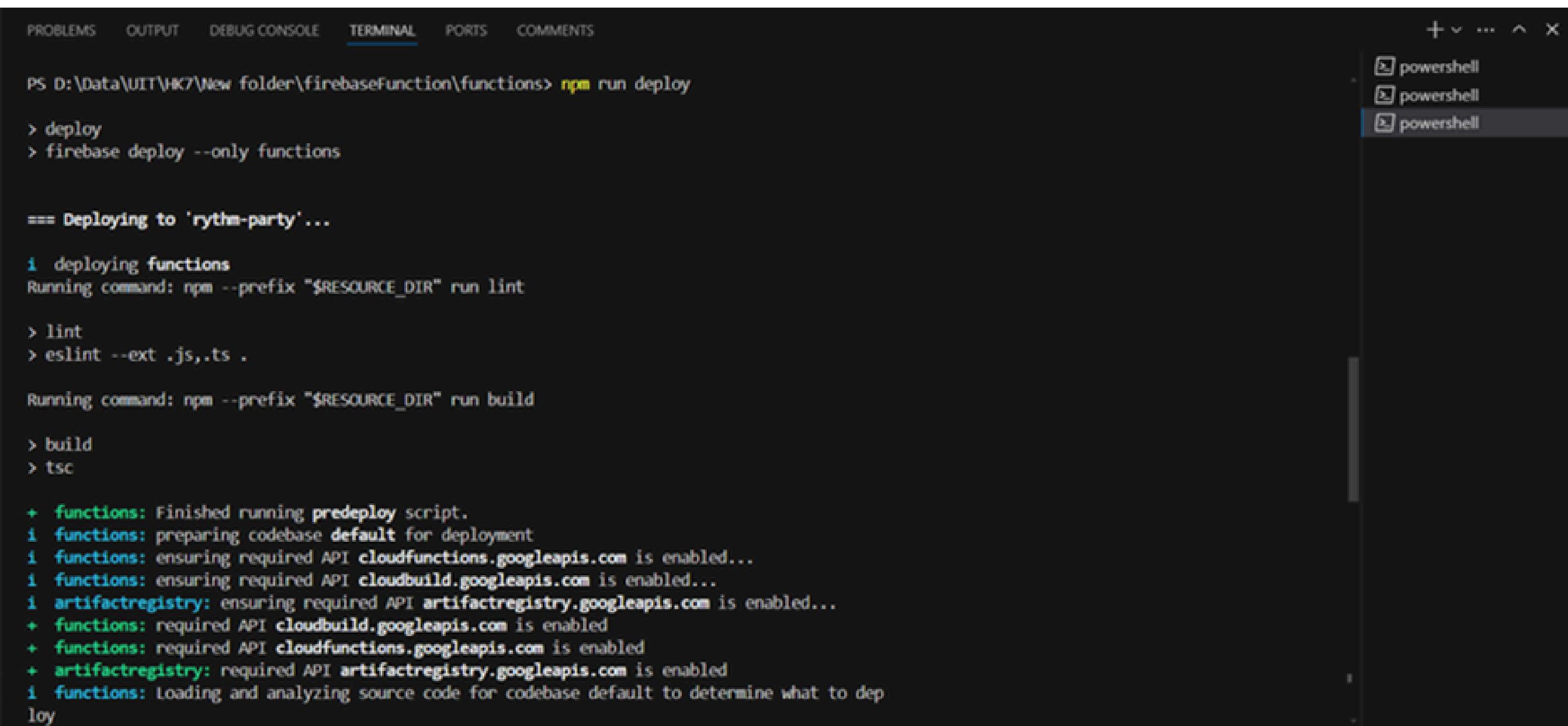
Tạo và deploy Firebase Function

B2. Viết 1 function trong Firebase function

```
1
2 import {onRequest} from "firebase-functions/v2/https";
3 import * as logger from "firebase-functions/logger";
4
5 // Start writing functions
6 // https://firebase.google.com/docs/functions/typescript
7
8 export const helloWorld = onRequest((request, response) => {
9   logger.info("Hello logs!", {structuredData: true});
10  response.send("Hello from Firebase!");
11});
```

Tạo và deploy Firebase Function

B3. Deploy function lên trên hệ thống server



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS + v ... ^ X
PS D:\Data\UIT\HK7>New folder\firebaseFunction\functions> npm run deploy
> deploy
> firebase deploy --only functions

== Deploying to 'rythm-party'...

i  deploying functions
Running command: npm --prefix "$RESOURCE_DIR" run lint

> lint
> eslint --ext .js,.ts .

Running command: npm --prefix "$RESOURCE_DIR" run build

> build
> tsc

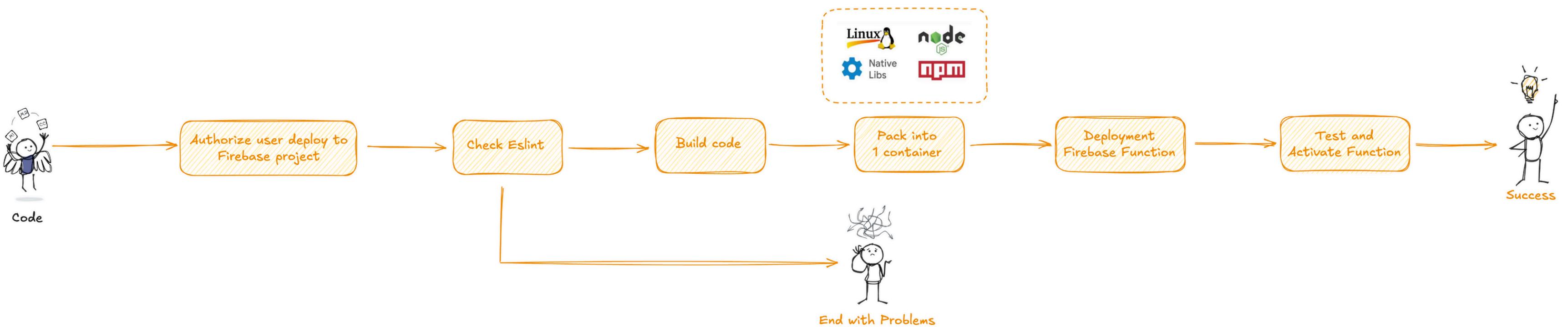
+ functions: Finished running predeploy script.
i  functions: preparing codebase default for deployment
i  functions: ensuring required API cloudfunctions.googleapis.com is enabled...
i  functions: ensuring required API cloudbuild.googleapis.com is enabled...
i  artifactregistry: ensuring required API artifactregistry.googleapis.com is enabled...
+ functions: required API cloudbuild.googleapis.com is enabled
+ functions: required API cloudfunctions.googleapis.com is enabled
+ artifactregistry: required API artifactregistry.googleapis.com is enabled
i  functions: Loading and analyzing source code for codebase default to determine what to dep
loy
```

Tạo và deploy Firebase Function

B4. Kiểm tra function trên hệ thống

Function	Trigger	Version	Requests (24 hrs)	Min / Max Instances	Timeout
heavyFunction us-central1	HTTP Request https://heavyfunction-c02kgvg15a-uc.a.run.app	v2	0	0 / 3	1m
helloWorld us-central1	HTTP Request https://helloworld-c02kgvg15a-uc.a.run.app	v2	0	0 / 3	1m

Quy trình deploy

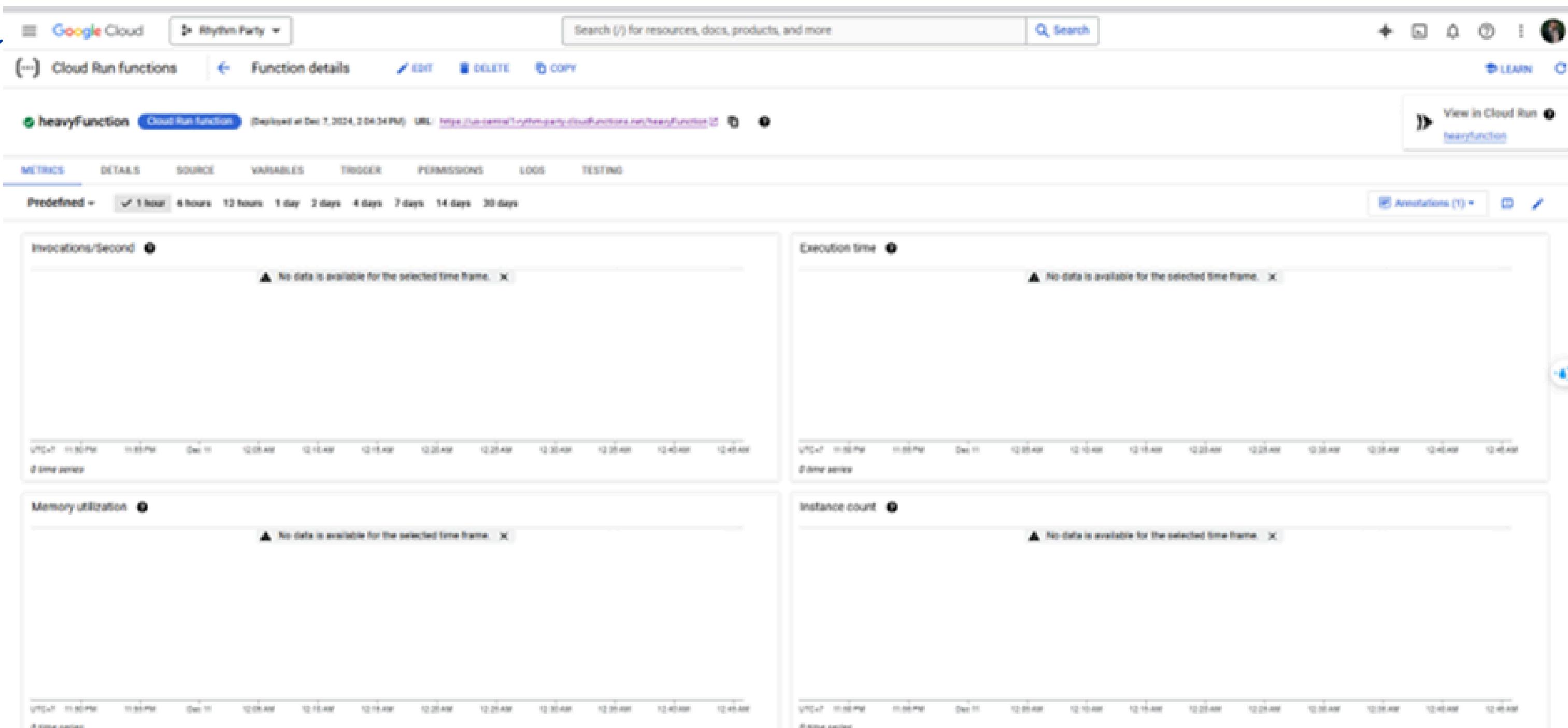


Quản lý Firebase Function

The screenshot shows the Google Cloud Functions interface for a project named "project-management". The top navigation bar includes "Google Cloud", the project name, a search bar, and user account information. Below the navigation is a toolbar with "Cloud Run functions", "Functions", "CREATE FUNCTION", and "REFRESH" buttons, along with "LEARN" and "RELEASE NOTES" links. A note at the top states: "We will be integrating Cloud Functions into Cloud Run UI in the upcoming months." The main content area displays a table of functions:

Environment	Name	Last Deployed	Region	Recommendation	Trigger	Runtime	Memory allocated	Executed function	Actions
Cloud Run function	CheckOtpCode	Jan 1, 2025, 4:38:37 PM	us-central1		HTTP	Node.js 18	256 MB	CheckOtpCode	⋮
Cloud Run function	CreatePasswordUser	Jan 1, 2025, 4:38:34 PM	us-central1		HTTP	Node.js 18	256 MB	CreatePasswordUser	⋮
Cloud Run function	CreateUserAccount	Jan 1, 2025, 4:38:36 PM	us-central1		HTTP	Node.js 18	256 MB	CreateUserAccount	⋮
Cloud Run function	CreateUserDetail	Jan 1, 2025, 4:38:36 PM	us-central1		HTTP	Node.js 18	256 MB	CreateUserDetail	⋮
Cloud Run function (1st gen)	CreateUserReaction	Jan 1, 2025, 4:38:34 PM	us-central1		Firebase Authentication (Preview)	Node.js 18	256 MB	CreateUserReaction	⋮
Cloud Run function	SendMailToUser	Jan 1, 2025, 4:38:34 PM	us-central1		HTTP	Node.js 18	256 MB	SendMailToUser	⋮
Cloud Run function	SendOtpCodeToUser	Jan 1, 2025, 4:38:36 PM	us-central1		HTTP	Node.js 18	256 MB	SendOtpCodeToUser	⋮
Cloud Run function	Test	Jan 1, 2025, 4:38:36 PM	us-central1		HTTP	Node.js 18	256 MB	Test	⋮

Quản lý Firebase Function



Cấu hình Function

[...] Cloud Run functions

Edit function

Runtime, build, connections and security settings ^

RUNTIME BUILD CONNECTIONS SECURITY AND >

Memory allocated * 256 MiB CPU * 1

Timeout * 60 seconds ?

Concurrency

Maximum concurrent requests per instance 80 ?

Autoscaling ?

Minimum number of instances 0 Maximum number of instances 3

Runtime service account ?

Service account Default compute service account

By default Cloud Functions uses the automatically created Default Compute Engine Service Account. [Learn more about service accounts.](#) ↗

NEXT

CANCEL

Container trong Firebase

- Khi deploy một Firebase Function, source code sẽ được đóng gói và lưu trữ trong môi trường của Firebase.
- Mỗi function được đóng gói vào một container riêng biệt, đảm bảo tính cô lập và bảo mật.
- Các dependencies (như thư viện, runtime, hoặc môi trường cần thiết) cũng được bao gồm trong container.

Instance trong Firebase

- Instance giống như 1 máy ảo chạy độc lập có các thông số tài nguyên CPU, RAM riêng biệt.
- Được khởi tạo với một runtime riêng biệt
- Không chia sẻ bộ nhớ, biến toàn cục, hoặc dữ liệu in-memory với các instance khác.

Cơ chế hoạt động của Instance

Khởi tạo instance:

- Nếu chưa có instance (hoặc sau một thời gian dài không sử dụng), Firebase sẽ khởi chạy một container mới từ image của function.
- Được gọi là "cold start" và thường mất thêm thời gian để khởi động

Cơ chế hoạt động của Instance

Tái Sử Dụng Instance:

- Sau khi một instance được khởi tạo, Firebase có thể tái sử dụng nó để xử lý các yêu cầu tiếp theo nhằm tối ưu hóa tài nguyên và giảm thời gian phản hồi.
- Giảm thời gian khởi tạo khi dùng lại những instance cũ

Cơ chế hoạt động của Instance

Tự Động Mở Rộng:

- **Firebase Functions tự động mở rộng (scale-out) bằng cách tạo thêm các instance mới khi lưu lượng tăng cao, tuy nhiên nó chỉ tạo đến mức tối đa được setup trong options.**
- **Khi 1 instance không thể xử lý thêm request thì Firebase sẽ cung cấp thêm 1 instance mới**

Cơ chế hoạt động của Instance

Tự động thu hẹp:

- Khi lưu lượng giảm, Firebase sẽ tự động thu hẹp số lượng instance (scale-in) để tiết kiệm tài nguyên
- Các instance không hoạt động trong một khoảng thời gian (khoảng 15 phút) sẽ bị xóa.
- Giúp tối ưu hóa chi phí vì bạn chỉ trả tiền cho tài nguyên được sử dụng.

Cold start là gì ?

- **Cold start là hiện tượng xảy ra khi một hàm serverless được khởi chạy lần đầu tiên hoặc sau một khoảng thời gian không hoạt động.**
- **Trong giai đoạn này, môi trường thực thi (runtime environment) cần được khởi tạo, dẫn đến thời gian phản hồi lâu hơn so với các lần gọi hàm sau đó (warm invocation).**
-

Quy trình khởi tạo Cold start

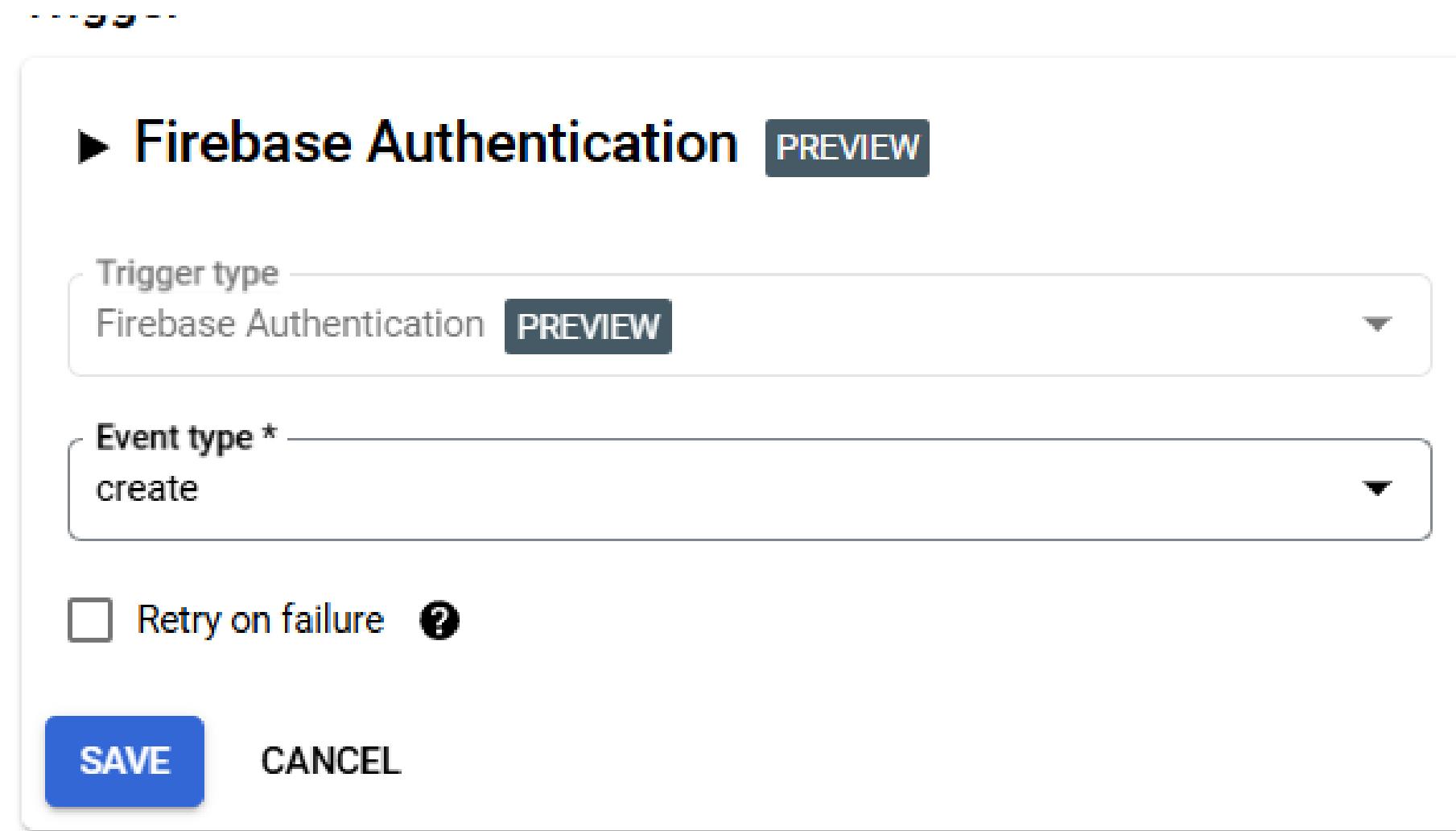
- **Cấp phát tài nguyên (container, bộ nhớ, CPU).**
- **Tải mã nguồn và các dependency (thư viện) cần thiết.**
- **Khởi tạo runtime (Node.js, Python, v.v.).**
- **Chạy các mã khởi tạo (initialization code) trong hàm.**

Khắc phục Cold start

Để Min Instance là 1 để luôn đảm bảo 1 có 1 instance tồn tại bên trong hệ thống. Tuy nhiên việc để 1 instance sẽ có thể ảnh hưởng đến chi phí không cần thiết

Retry function

Khi function lỗi thì tùy loại function mà ta có thể cho function đó retry khi gặp lỗi hay không.



Retry function

Tuy nhiên việc retry vậy sẽ loop vô tận nếu gặp lỗi liên tục nên ta sẽ có cách khắc phục là với mỗi function sẽ tồn tại 1 EventId và lưu eventId đó vô csdl và chỉ cho lặp lại 1 số lần thôi

Cơ Chế Mở Rộng Của Firebase Functions Khi Quá Tải

- Khi số lượng yêu cầu (request) vượt quá khả năng xử lý của một instance, Firebase sẽ tự động tạo thêm các instance mới để đáp ứng.
- Nếu đã đạt giới hạn tối đa về số lượng instance và các request tiếp theo không được xử lý.

Cơ Chế Mở Rộng Của Firebase

Functions Khi Quá Tải

- Các request sẽ được đưa vô hàng đợi nếu đã đạt giới hạn tối đa về số lượng instance và các request tiếp theo không được xử lý.
- Nếu một request trong hàng đợi không được xử lý kịp thời và vượt quá thời gian timeout của function (thường là 60 giây đối với HTTP-triggered functions), nó sẽ thất bại và trả lỗi về phía client

Cơ Chế Mở Rộng Của Firebase

Functions Khi Quá Tải

The screenshot shows the Google Cloud Functions logs for a Cloud Run function named 'heavyFunction'. The logs are filtered for 'Default' severity and show numerous errors (indicated by a yellow exclamation mark icon) from December 7, 2024, at 14:16:19.273 ICT. Each log entry details a GET request to the function URL, showing a status code of 429 (indicating a rate limit error), a response size of 14 B, and a duration of 0 ms. The logs also mention the source IP k6/0.55.0 and the URL https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction.

Severity	Timestamp	Summary
i	2024-12-07 14:16:19.273 ICT	GET 200 223 B 1.5 s k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:19.284 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
i	2024-12-07 14:16:20.464 ICT	GET 200 222 B 1.5 s k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
i	2024-12-07 14:16:20.524 ICT	GET 200 223 B 1.5 s k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
i	2024-12-07 14:16:20.739 ICT	GET 200 223 B 1.5 s k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:20.739 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:20.740 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:20.924 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
i	2024-12-07 14:16:20.925 ICT	GET 200 223 B 1.5 s k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:21.771 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:21.950 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:21.985 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:21.985 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:22.053 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction
!	2024-12-07 14:16:22.158 ICT	GET 429 14 B 0 ms k6/0.55.0 (https://k6.io/) https://us-central1-rhythm-party.cloudfunctions.net/heavyFunction

CI/CD trong Firebase Function với Git Action

```
1 name: CI
2
3 on:
4   push:
5     branches: ["master"]
6   pull_request:
7     branches: ["master"]
8   workflow_dispatch:
9
10 jobs:
11   build_and_deploy:
12     runs-on: ubuntu-latest
13     steps:
14       - name: Checkout code
15         uses: actions/checkout@v2
16
17       - name: Install Firebase dependencies
18         working-directory: BE
19         run: npm install
```

```
1   - name: Install dependencies
2     working-directory: BE/functions
3     run: npm install
4
5   - name: Deploy to Firebase
6     working-directory: BE/functions
7     run: npm run deploy
8     env:
9       FIREBASE_TOKEN: ${{ secrets.FIREBASE_TOKEN }}
10      SERVICE_ACCOUNT_KEY: ${{ secrets.SERVICE_ACCOUNT_KEY }}
11
```



Kết quả đạt được



01

Tìm hiểu sâu về Serverless và các nguyên lý

02

Nhóm có khả năng tìm hiểu và áp dụng công nghệ mới

03

Áp dụng vào ứng dụng demo

Demo

**Cảm ơn Thầy và các bạn
đã lắng nghe!**