



Bài 12. Một số dịch vụ cao cấp của AWS



Nội dung

- Serverless và dịch vụ Lambda
- Microservice và dịch vụ Kuberness



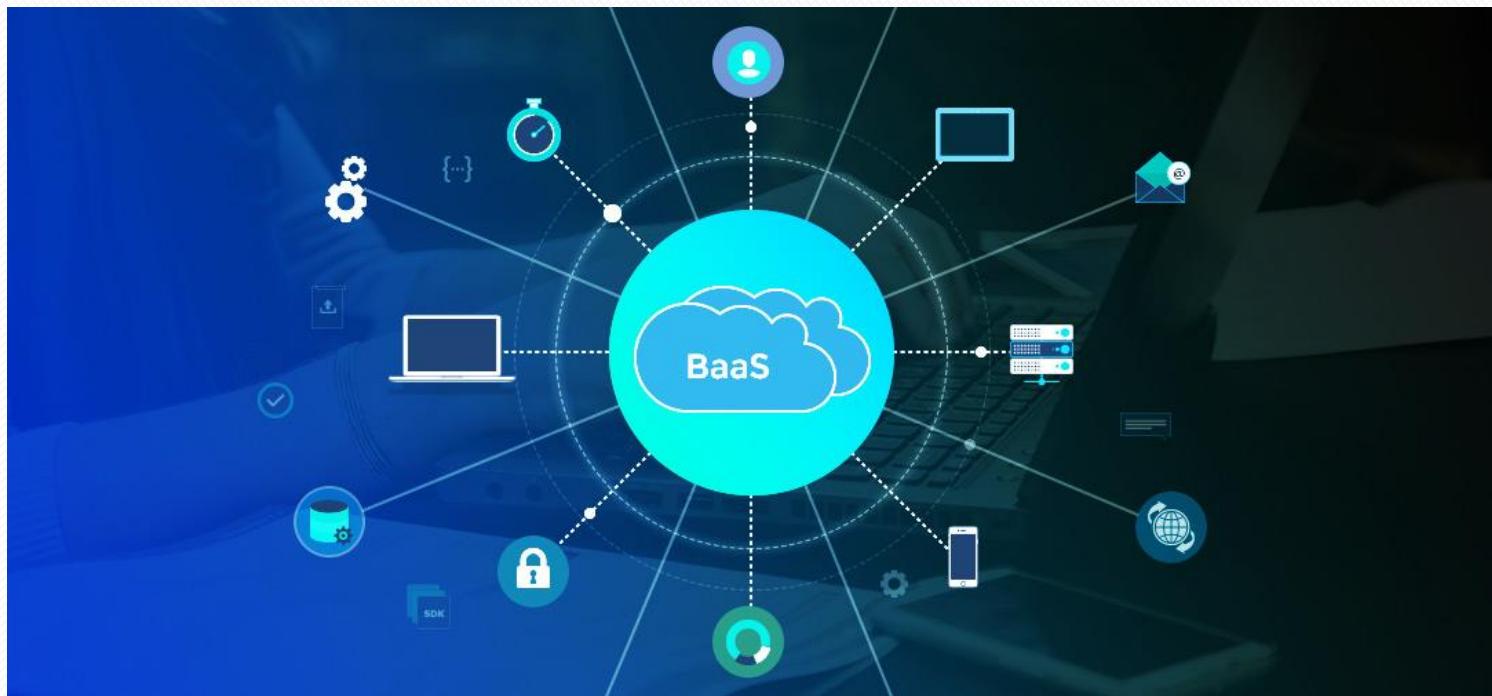
Serverless là gì?

- **Serverless** là môi trường, nền tảng thực thi ứng dụng và dịch vụ mà không phải quan tâm đến máy chủ.
- Điều quan trọng và khác biệt nhất trong serverless là bạn **chỉ trả tiền khi và chỉ những phần bạn sử dụng**.



● BaaS – Backend as a Service:

- “Ở mô hình này, phần lớn code logic của chúng ta sẽ chuyển về xử lý ở phía Frontend. Còn Backend thì sử dụng các API có sẵn của bên thứ ba.”



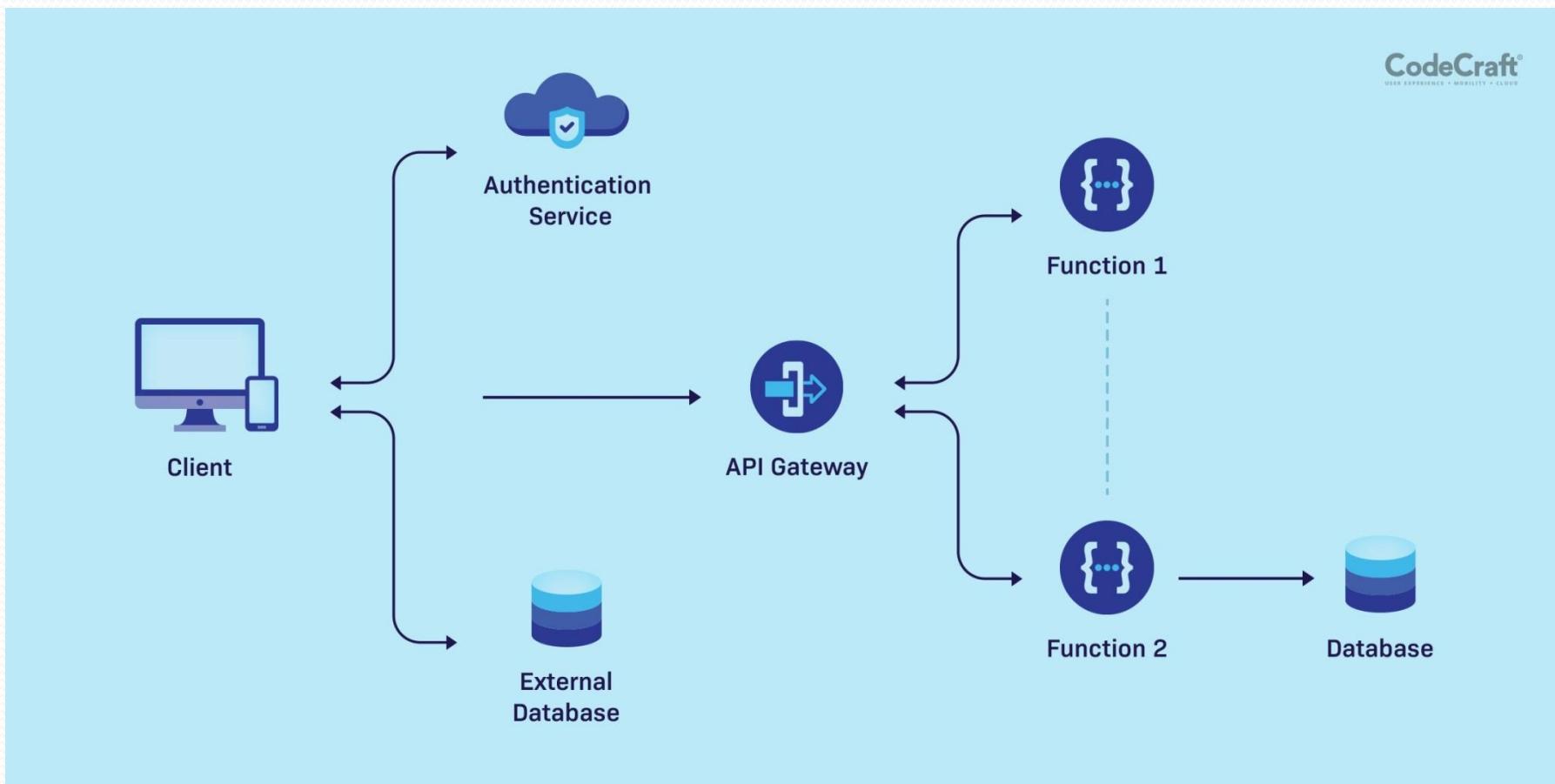


● FaaS – Function as a Service:

- Tự viết các API cho mục đích của mình, và triển khai chúng lên Server.
- Nhưng thay vì triển khai theo mô hình client-server thông thường là phải thuê Server rồi deploy code lên đó, thì chúng ta lại **deploy code dưới dạng các Function (Function as a Service)** và các function này có thể gọi dưới dạng RestAPI.
- Với mô hình **FaaS** này, chúng ta sẽ chỉ cần viết code thôi, không cần quan tâm việc server và code được lưu trữ ở đâu, các nhà cung cấp Cloud Computing bên thứ ba sẽ tự quản lý phần này.

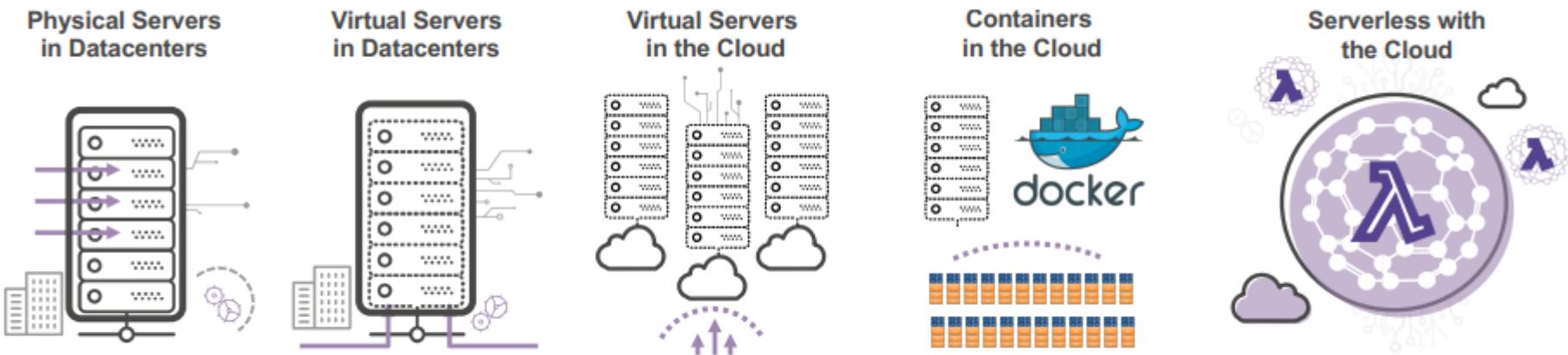


Kiến trúc serverless





Quá trình phát triển





Ưu, nhược điểm

- **Ưu điểm:**
 - Người phát triển ứng dụng không lo việc quản lý và vận hành máy chủ.
 - Tiết kiệm chi phí thuê máy chủ.
 - Dễ dàng mở rộng.
 - Độ sẵn sàng cao.
- **Nhược điểm:**
 - Việc sử dụng ứng dụng có độ trễ nhất định
 - Việc giám sát và sửa lỗi ứng dụng khó khăn
 - Giới hạn về bộ nhớ và thời gian của nhà cung cấp dịch vụ đối với ứng dụng.
 - Phụ thuộc nhà cung cấp dịch vụ.
 - Chi phí ngầm.
 - Tốn thời gian cho việc tìm hiểu các công cụ liên quan.



Một số dịch vụ



AWS Lambda



Azure Function



Google Cloud Functions



Tính năng	AWS Lambda	Google Cloud	Azure Functions
Khả năng mở rộng	Tự động	Tự động	Bằng tay hoặc theo plan đặt trước
Số Function tối đa	Không giới hạn	1000 trên 1 project	Không giới hạn
Xử lý đồng thời	1000 trên 1 account 1 region (soft limit)	Không giới hạn	Không giới hạn
Thời gian xử lý tối đa	300 sec (5 min)	540 seconds (9 minutes)	300 sec (5 min)
Ngôn ngữ	JavaScript, Java, C#, and Python	Only JavaScript	C#, JavaScript, F#, Python, Batch, PHP, PowerShell



AWS Lambda

- AWS Lambda là dịch vụ của AWS cho phép chạy mã mà không cần cung cấp hay quản lý máy chủ.
- Với Lambda, bạn có thể chạy mã cho gần như toàn bộ các loại ứng dụng hay dịch vụ backend – tất cả đều không cần quản trị.
- Chỉ cần tải đoạn mã của bạn lên và Lambda sẽ lo hết những gì cần làm để chạy và mở rộng mã của bạn với mức độ có sẵn cao.
- Bạn có thể thiết lập để mã của bạn tự động kích hoạt từ các dịch vụ AWS khác, hoặc gọi trực tiếp từ bất cứ ứng dụng web hay di động nào.



Cách thức hoạt động



Upload your code to AWS Lambda or write code in Lambda's code editor



Set up your code to trigger from other AWS services, HTTP endpoints, or in-app activity



AWS Lambda

Lambda runs your code only when triggered, using only the compute resources needed



Just pay for the compute time you use



Ví dụ

The Seattle Times



Photograph is taken

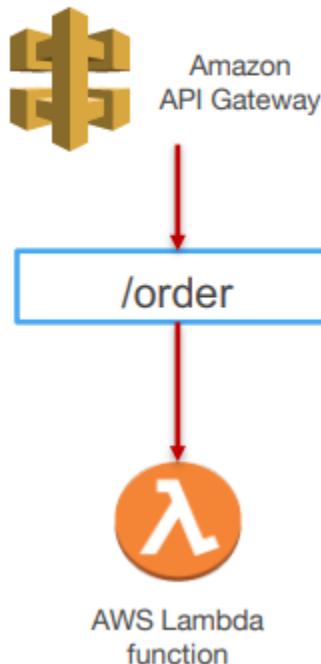


Photo is resized into web, mobile, and tablet sizes



Mô hình thực thi

Synchronous (push)



Asynchronous (event)



Stream-based





Hoạt động của Lambda

- Các thành phần chính: Lambda Function và Event Source
 - Event Source đưa ra các sự kiện
 - Lambda function: upload code lên Lambda.
 - Tính toán tài nguyên cho hệ thống là con số ước lượng mà bạn đưa ra mà function của mình cần dùng.
- Gọi Lambda function tự động: sử dụng Event Sources



Tạo ứng dụng Lambda

- Chọn mục Lambda trong nhóm dịch vụ Compute

The screenshot shows the AWS Lambda service in the AWS Management Console. The left sidebar has 'AWS Lambda' selected under 'Functions'. The main area displays a table of functions:

Function name	Description	Runtime	Code size	Last Modified
shorten-url		Node.js 8.10	2.1 MB	2 days ago
test		Node.js 8.10	674.7 kB	2 days ago

At the bottom, there are links for 'Feedback', 'English (US)', and legal notices: '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.



- Chọn vào Create function để tạo 1 Lambda function mới:

Author from scratch

Start with a simple "hello world" example.

Blueprints

Choose a preconfigured template as a starting point for your Lambda function.

Serverless Application Repository

Find and deploy serverless apps published by developers, companies, and partners on AWS.

Author from scratch [Info](#)

Name

Runtime

Role
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name
Enter a name for your new role.

(i) This new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.

Policy templates
Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

[Cancel](#)



- Chọn *Author from scratch* để tạo 1 function rỗng
- Thiết lập các thông số:
 - *Name* là tên Lambda function,
 - *Runtime* là môi trường chạy code (Nodejs,...)
 - *Role* là vai trò tạo khởi tạo hàm,
 - *Policy templates* là tùy chọn sử dụng thêm các AWS mà Amazon cung cấp, sử dụng API Gateway.



● Cửa sổ soạn thảo code

Code entry type: Edit code inline Runtime: Node.js 8.10 Handler: index.handler

File Edit Find View Goto Tools Window

Environment myFunction index.js

```
index.js
1 exports.handler = async (event) => {
2     // TODO implement
3     return 'Hello from Lambda!';
4 };
5
```

1:1 JavaScript Spaces: 4



- Soạn code hoặc upload code đã có trên máy cá nhân.

The screenshot shows a code editor interface with the following details:

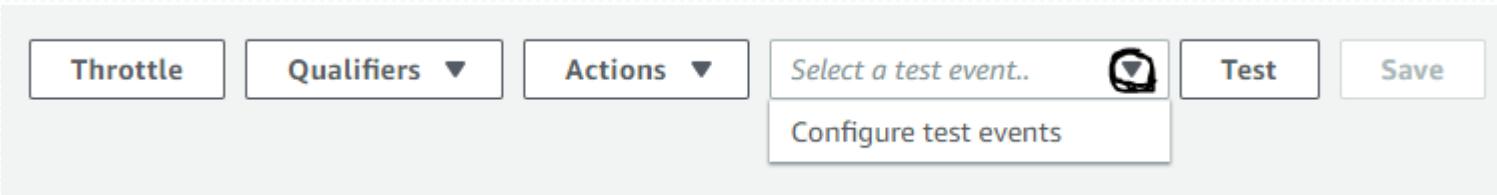
- EXPLORER** sidebar:
 - OPEN EDITORS**: .gitignore, **index.js**
 - SHORTEN-URL-SERVICE**: node_modules, .gitignore, config.json, **index.js**, package-lock.json, package.json, shorten_url.zip
- index.js** file tab is active.
- Content of index.js:**

```
1 const BitlyClient = require('bitly');
2 const config = require('./config.json');
3 let bitly = new BitlyClient.BitlyClient(config.token);
4
5
6 exports.handler = (event, context, callback) => {
7     bitly.shorten(event.url)
8         .then(shortUrl => {
9             callback(null, shortUrl);
10        })
11         .catch(err => {
12             callback(err)
13        })
14    }
15
```



Test Lambda function

- Chọn **Select a test event..** -> **Configure test events**





Configure test event

X

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

- Create new test event
- Edit saved test events

Event template

Hello World



Event name

MyEventName

```
1  [{}  
2    "key3": "value3",  
3    "key2": "value2",  
4    "key1": "value1"  
5 ]|
```



- Vì function của chúng ta nhận sự kiện `event.url` nên ta sẽ viết test như sau.

Event name

testLongUrl

```
1 ▾ {  
2     "url": "http://facebook.com"  
3 }
```



- Sau đó Save lại và nhấn chọn **Test**, nó sẽ trả về kết quả là link đã rút gọn nhờ bitly:

Execution result: succeeded ([logs](#))

X

▼ Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

```
"http://bit.ly/2KKbmbK"
```

Summary

Code SHA-256	QGVmTuJlyDx9sHTc9lwFqnvHpeFyogynhzUPahT7JKQ=	Request ID	0a4f3fd4-8ee9-11e8-82a3-652321f2885e
Duration	197.66 ms	Billed duration	200 ms
Resources configured	1024 MB	Max memory used	66 MB

Log output

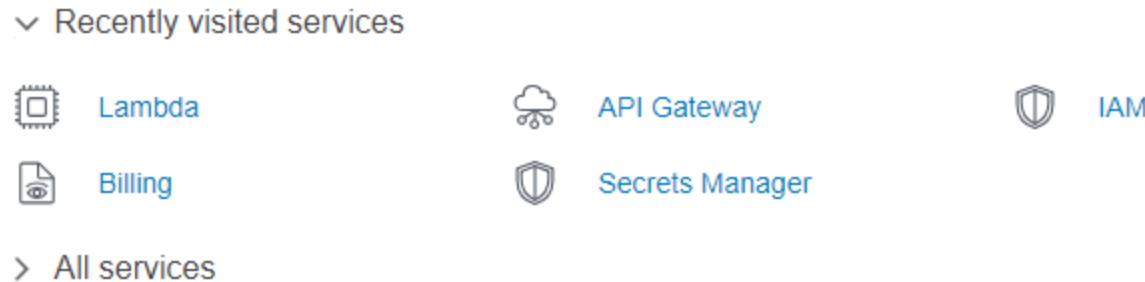
The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: 0a4f3fd4-8ee9-11e8-82a3-652321f2885e Version: $LATEST
END RequestId: 0a4f3fd4-8ee9-11e8-82a3-652321f2885e
REPORT RequestId: 0a4f3fd4-8ee9-11e8-82a3-652321f2885e Duration: 197.66 ms      Billed Duration: 200 ms      Memory Size: 1024 MB      Max Memory Used: 66 MB
```



Tạo API endpoint cho Lambda function

- Vào trang *API Gateway* ở trang chủ của AWS





Amazon API Gateway APIs

APIs

- shorten-url-API
- Usage Plans
- API Keys
- Custom Domain Names
- Client Certificates
- VPC Links
- Settings

+ Create API

shorten-url-API 

Created on 7/23/2018

Created by AWS Lambda

Endpoint Configuration

Endpoint Type  Regional



- Chọn **Create API** và tiến hành set name cho nó:

Create new API

In Amazon API Gateway, an API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

New API Clone from existing API Import from Swagger Example API

Settings

Choose a friendly name and description for your API.

API name*	<input type="text" value="minhphuong"/>
Description	<input type="text" value=""/>
Endpoint Type	<input type="text" value="Regional"/> <small>ⓘ</small>



● Add resource:

New Child Resource

Use this page to create a new child resource for your resource. 

Configure as proxy resource



Resource Name*



Resource Path*



You can add path parameters using brackets. For example, the resource path `{username}` represents a path parameter called 'username'. Configuring `/{proxy+}` as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to `/foo`. To handle requests to `/`, add a new ANY method on the `/` resource.

Enable API Gateway CORS



* Required

Cancel

Create Resource



- Add methods cho resource :

The screenshot shows a user interface for managing API resources. The top navigation bar includes 'Resources' and 'Actions'. Below this, the URL path is shown as '/shorten'. The main content area is titled '/shortenlink Methods' and contains two sections: 'RESOURCE ACTIONS' and 'API ACTIONS'. Under 'RESOURCE ACTIONS', the options are 'Create Method', 'Create Resource', 'Enable CORS', 'Edit Resource Documentation', and 'Delete Resource' (highlighted in red). Under 'API ACTIONS', the options are 'Deploy API', 'Import API', 'Edit API Documentation', and 'Delete API' (highlighted in red).

Actions	
/shorten Methods	
RESOURCE ACTIONS	
Create Method	
Create Resource	
Enable CORS	
Edit Resource Documentation	
Delete Resource	
API ACTIONS	
Deploy API	
Import API	
Edit API Documentation	
Delete API	



- Chọn *Create Method* để thêm phương thức request

Choose the integration point for your new method.

Integration type Lambda Function ⓘ

- HTTP ⓘ
- Mock ⓘ
- AWS Service ⓘ
- VPC Link ⓘ

Use Lambda Proxy integration ⓘ

Lambda Region ⓘ

Lambda Function ⓘ

Use Default Timeout ⓘ

Save



- Trong bảng *Action* chọn *Deploy API*, chọn *New stage* rồi nhập tên stage bất kì trong ô *Stage name* rồi nhấn *Deploy*:

Deploy API ● ×

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage	<input style="width: 150px; height: 25px; border: 1px solid #ccc; padding: 2px 5px;" type="button" value="New Stage"/>
Stage name*	<input style="width: 450px; height: 30px; border: 1px solid #ccc; padding: 2px 5px;" type="text" value="shortenlink"/>
Stage description	<input style="width: 450px; height: 40px; border: 1px solid #ccc; padding: 2px 5px;" type="text"/>
Deployment description	<input style="width: 450px; height: 40px; border: 1px solid #ccc; padding: 2px 5px;" type="text"/>

Cancel Deploy



- Sau đó bạn sẽ nhận được 1 trang quản lý api của mình.
- Vào Endpoint để lấy URL

shortenlink Stage Editor Delete Stage

Invoke URL: [REDACTED]

Settings Logs Stage Variables SDK Generation Export Deployment History Documentation History Canary

Configure the metering and caching settings for the **shortenlink** stage.

Cache Settings

Enable API cache

Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. [Read more about API Gateway throttling](#)

Enable throttling

Rate: requests per second

Burst: requests

Client Certificate

Select the client certificate that API Gateway will use to call your integration endpoints in this stage.

Certificate:

Tags

You can tag your API stages with a key-value pair. This is useful for tracking cost allocation among your AWS resources. [Read more about AWS Tagging](#)

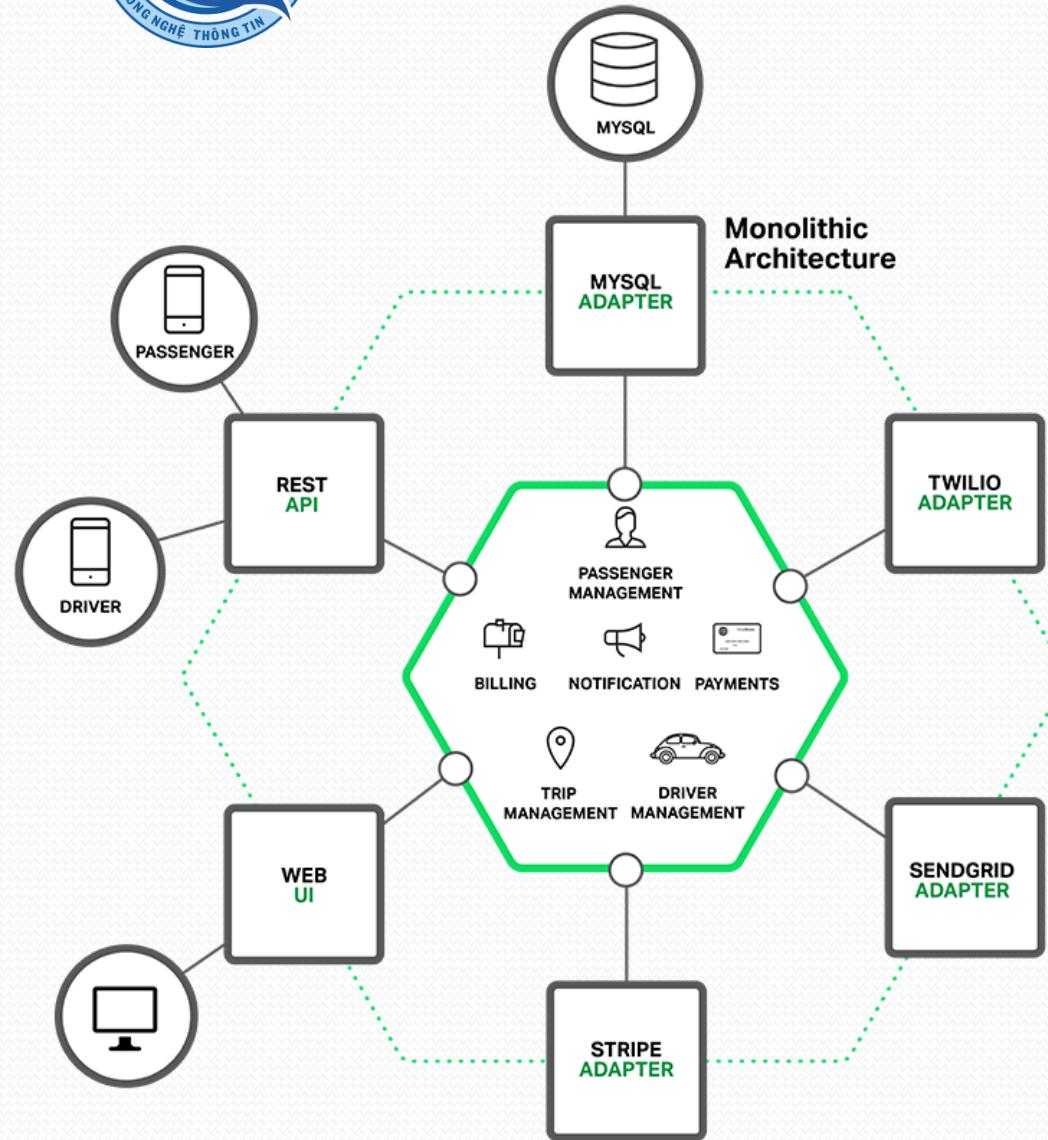
Key	Value
Add Stage Tag	



Microservices và dịch vụ Kubernetes

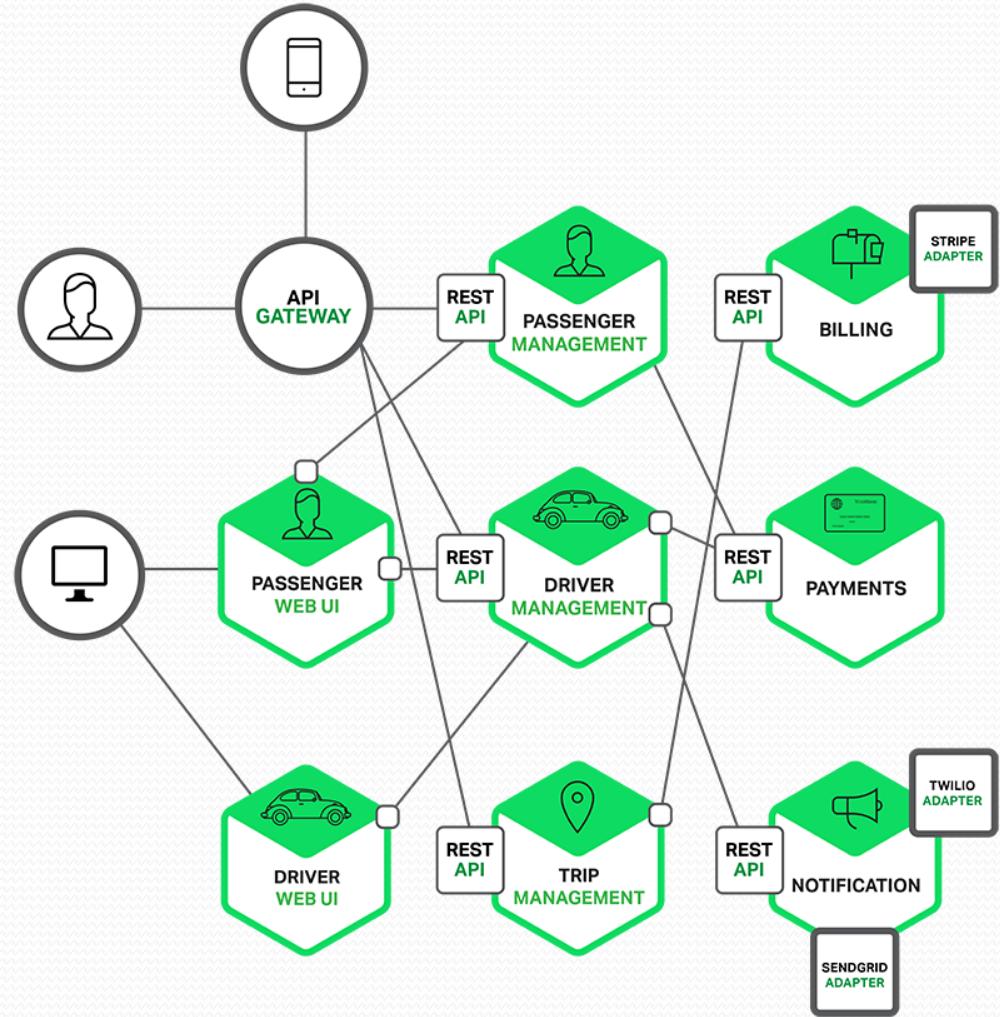


Kiến trúc một khối (Monolithic Architecture)





- **Kiến trúc dịch vụ nhỏ
(Microservices Architecture)**





Ưu nhược điểm của Microservices

- **Ưu điểm**
- Cho phép dễ dàng triển khai các ứng dụng lớn, phức tạp:
 - Cải thiện khả năng bảo trì - mỗi service tương đối nhỏ do đó dễ hiểu và thay đổi hơn
 - Khả năng testing dễ dàng hơn - các services nhỏ hơn và nhanh hơn để test
 - Khả năng triển khai tốt hơn - các services có thể được triển khai độc lập
 - Cho phép các services được phát triển bởi những team khác nhau.
 - Giảm thiểu rủi ro: Nếu có lỗi trong một service thì chỉ có service đó bị ảnh hưởng.
 - Dễ dàng thay đổi sử dụng các công nghệ mới

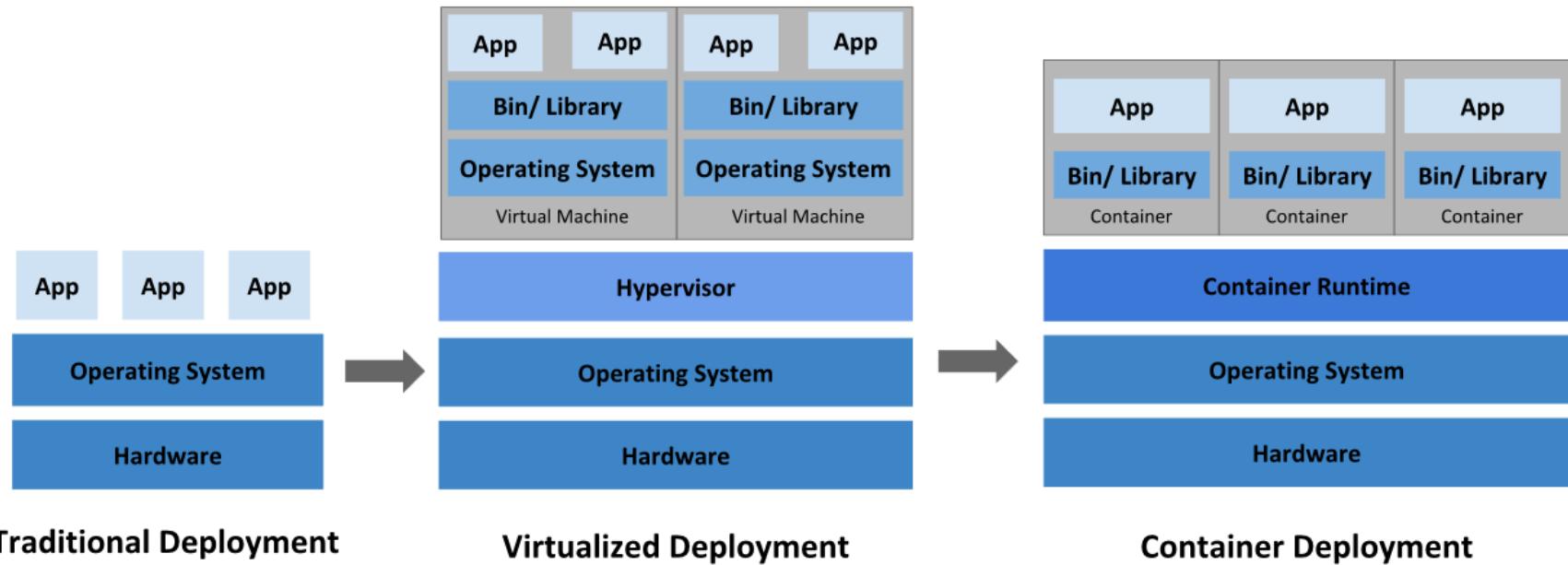


- **Nhược điểm**
- Các nhà phát triển phải đối phó với sự phức tạp của việc tạo ra một hệ thống phân tán:
 - Cần cài đặt việc truyền thông giữa các inter-services
 - Xử lý lỗi rất phức tạp vì một luồng xử lý cần đi qua nhiều services
 - Việc thực hiện các requests trải rộng trên nhiều services khó khăn hơn
 - Khó khăn trong việc đảm bảo toàn vẹn CSDL nếu triển khai theo kiến trúc cơ sở dữ liệu phân vùng
 - Triển khai và quản lý microservices nếu làm thủ công theo cách đã làm với ứng dụng một khối phức tạp hơn rất nhiều
 - Phải xử lý sự cố khi kết nối chậm, lỗi khi thông điệp không gửi được hoặc thông điệp gửi đến nhiều đích đến vào các thời điểm khác nhau



Kubernetes

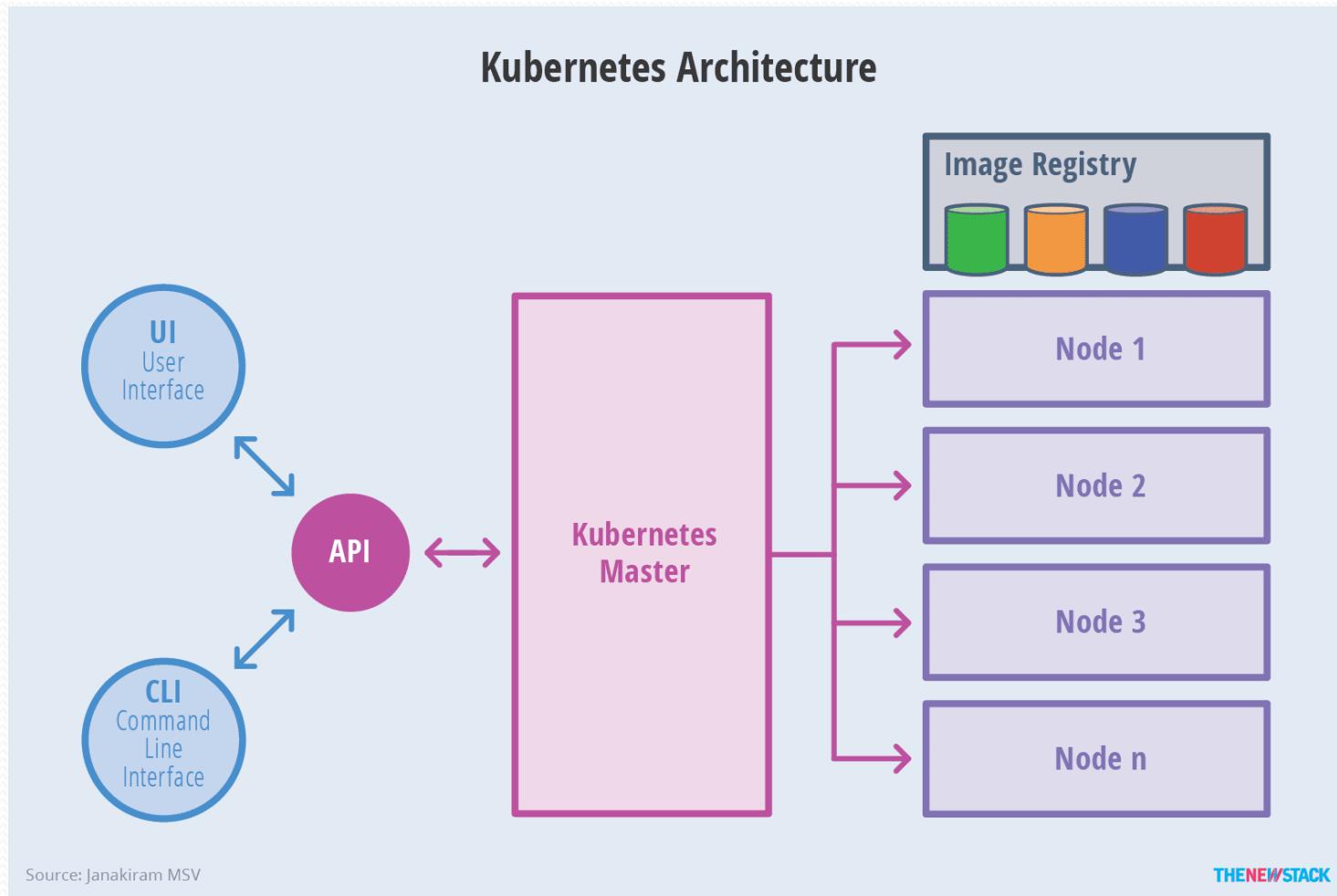
- Quá trình phát triển mô hình phát triển ứng dụng



- Kubernetes là một mã nguồn mở được dùng để tự động triển khai hệ thống, scaling, quản lý các container.



Kiến trúc Kubernetes

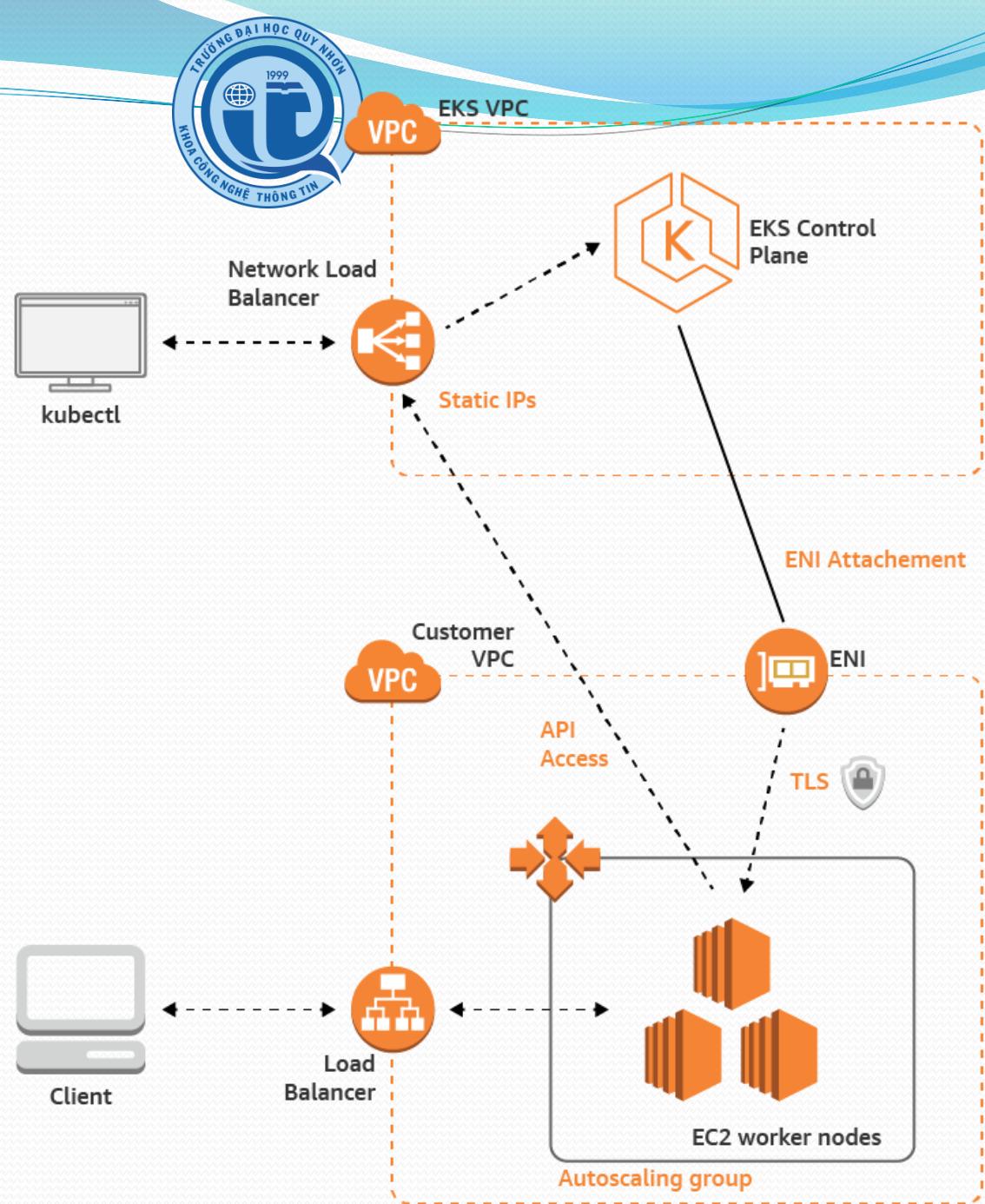




Amazon Elastic Kubernetes Service

- Amazon Elastic Kubernetes Service (Amazon EKS) là một dịch vụ Kubernetes được quản lý hoàn toàn.
- AWS EKS cho phép bạn deploy, run Kubernetes trên AWS.
- AWS EKS sẽ deploy Kubernetes master, rồi add các worker node vào cluster.
- Sau đó bạn có thể dùng Kubernetes command để kết nối và thao tác với Kubernetes.

- Mô hình ứng dụng triển khai trên Kubernetes





Tổng kết

- Serverless và Kubernetes là những dịch vụ cho các ứng dụng lớn, phức tạp.
- Điện toán đám mây là môi trường thuận lợi cho việc phát triển những dịch vụ này.