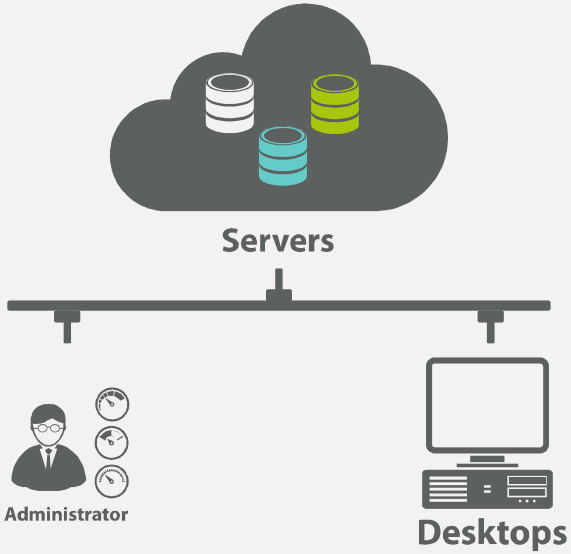
클라우드 기반 개발

AWS 서버리스 아키텍처 이해

MEGAZONE CLOUD



Index

Prologue

개요

Amazon Web Services 한국 블로그

Better Korea – AWS코리아 10주년 기념 백서를 공개합니다!

by AWS Korea | on 30 8월 2022 | in Customer Enablement, Events | Permalink | Share

올해는 AWS코리아 10주년이 되는 아주 특별한 해입니다. 지난 10년간 AWS코리아는 대한민국에서 가장 고객 중심적인 회사가 되자는 목표와 '클라우드 기술을 통해 더 나은 대한민국을 만들자(Build a Better Korea Powered by AWS)'는 미션을 가지고 많은 노력을 해왔습니다. 저희와 함께 대한민국의 고객들께서는 클라우드 기술이 혁신을 가속화하는데 얼마나 중요한 역할을 하는지 스스로 입증해 오셨습니다.

이러한 지속적인 투자와 서비스 확대를 통해서, 현재 AWS 서울 리전에서 지원하는 고객사는 수백 개에서 수 만개로 늘어났으며 파트너사 또한 수십 개에서 천여 개 이상으로 늘어났습니다. 하이테크/제조, 통신/미디어, 금융, 유통/소비재, 소프트웨어/인터넷, 게임, 여행/물류, 에너지 등 전 산업군에 소속되어 있는 대기업, 디지털네이티브, 중소기업, 스타트업에 이르기까지, 다양한 규모와 특징을 가진 고객사들이 AWS와 함께 클라우드에 기반한 디지털 전환을 실현해 나가고 있습니다.

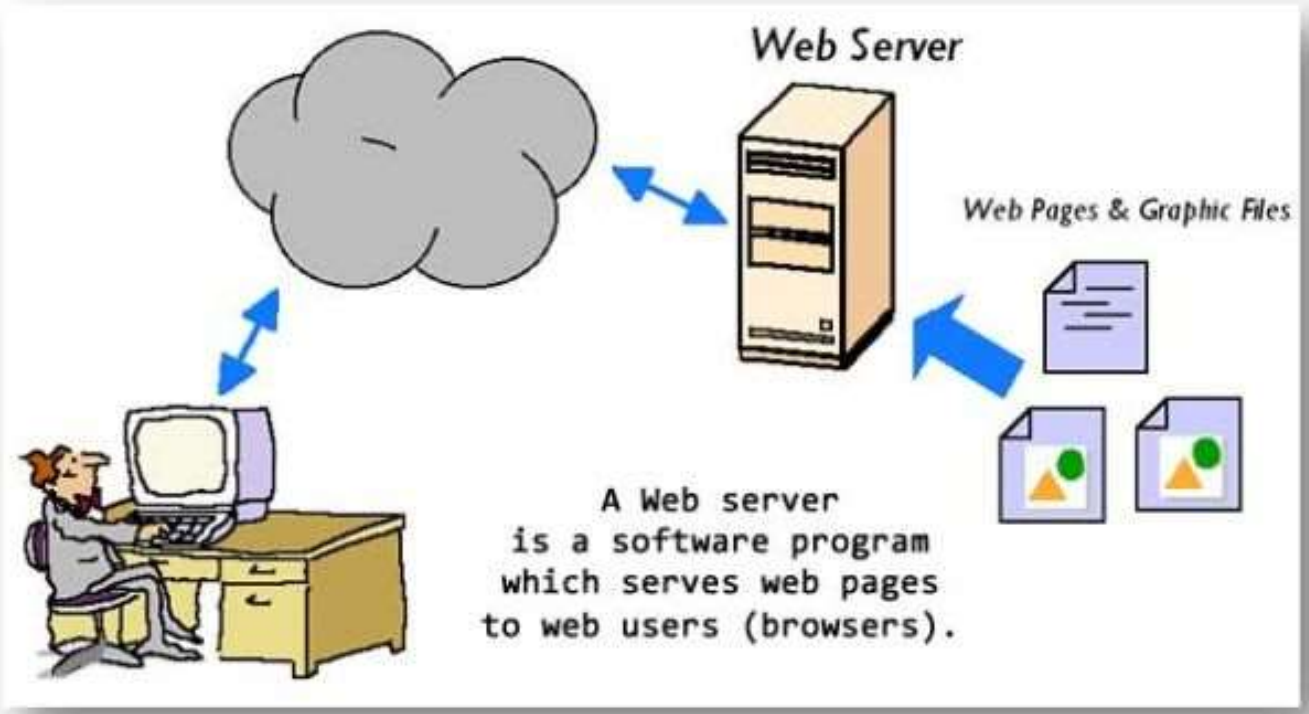
최근 팬데믹 상황에서 기업 뿐만 아니라 정부, 학교, 병원, 연구기관들이 그 어느때보다 중요한 역할을 해오고 있는데, 클라우드를 활용하여 신속히 대응하고 극복한 사례들이 늘어나고 있습니다. 교육계에서는 단순 대면교육 위주에서 비대면으로의 신속한 전환과 확산이 이루어졌고, 헬스케어 분야에서는 의료 연구, 진단, 백신 및 치료 개발 그리고 코로나 19에 대처하기 위한 다양한 노력을 기울이고 있습니다.

https://aws.amazon.com/ko/blogs/korea/better-korea-aws-korea-10th-anniversary-white-paper/

Serverless 개념

Serverless 개념

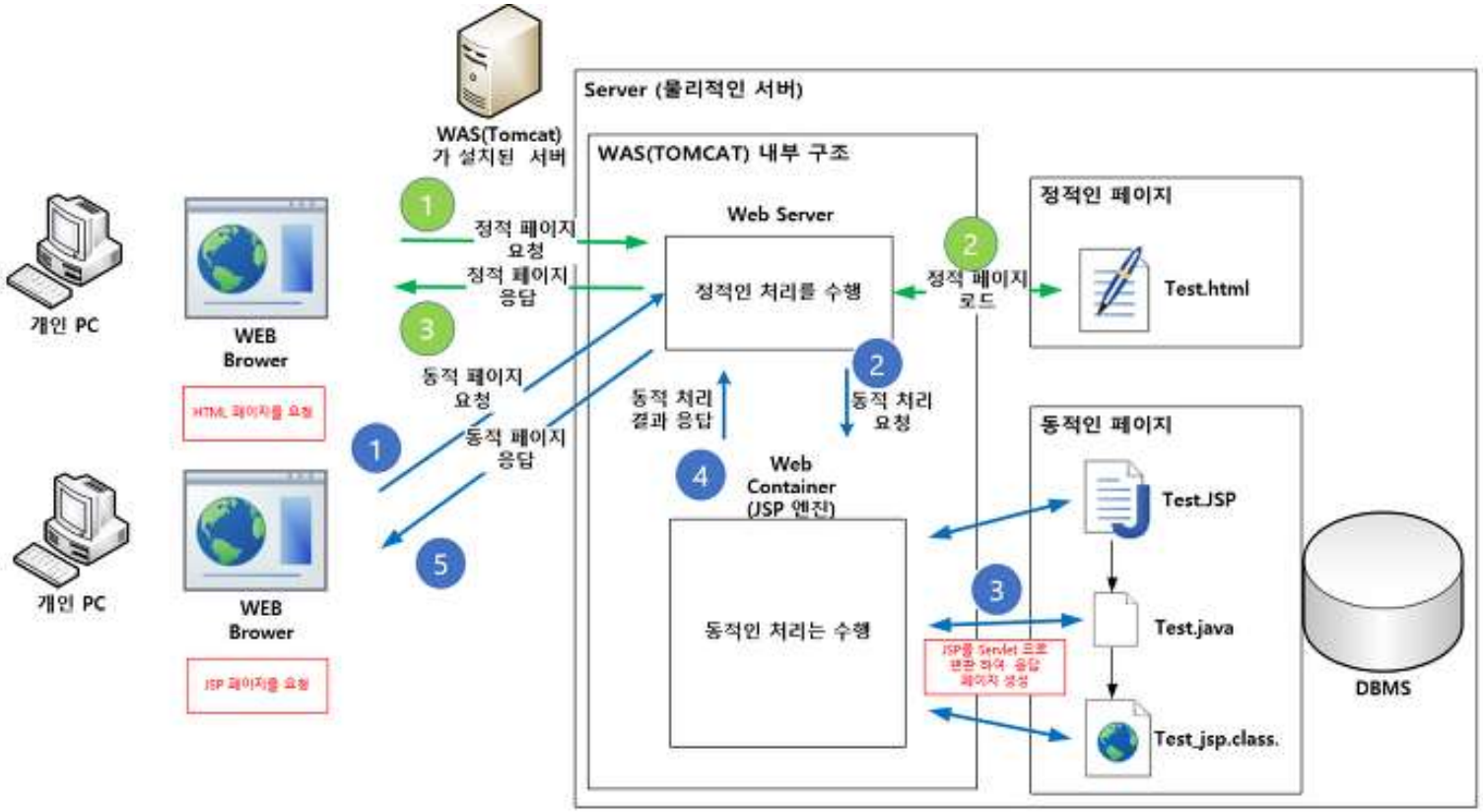
Server-Oriented



https://dyns.data.blog/2020/08/17/what-is-a-web-server/

Serverless 개념

Server-Oriented



https://doitnow-man.tistory.com/109

Server-Oriented

https://www.crn.com/news/data-center/google-unveils-new-750m-data-center-as-part-of-9-5b-goal

Serverless 개념

서버를 관리한다는 것은..

내 앱은 **서버 오류**를 어떻게 견뎌야 합니까?

언제 내 서버를 **확장**하기로 결정해야 합니까?

서버가 **손상**되었는지 어떻게 알 수 있습니까?

내 예산에 적합한 **서버 크기**는 무엇입니까?

내 서버의 **활용도**를 높이려면 어떻게 해야 합니까?

내 성능에 적합한 **서버 크기**는 무엇입니까?

내 서버에서 액세스를 **제어**하려면 어떻게 해야 합니까?

내 서버에서 **동적 구성 변경**을 구현하려면 어떻게 해야 합니까?

내 서버의 남은 **용량**은 얼마입니까?

얼마나 많은 **사용자**가 내 서버에 너무 많은 로드를 생성합니까?

몇 대의 서버에 대한 **예산**을 책정해야 합니까?

응용 프로그램은 서버 **하드웨어 오류**를 어떻게 처리합니까?

내 서버는 **어떤 OS**를 실행해야 합니까?

내 서버 운영 체제를 **패치** 상태로 유지하려면 어떻게 해야 합니까?

내 서버 이미지에 **어떤 패키지**를 구워야 합니까?

내 애플리케이션을 **최적화**하려면 OS 설정을 조정해야 합니까?

새 **코드**가 내 서버에 어떻게 배포됩니까?

서버 **확장**은 언제 결정해야 합니까?

Serverless 개념

개요

내 앱은 **서버 오류**를 어떻게 견뎌야 합니까?

언제 내 서버를 **확장**하기로 결정해야 합니까?

내 예산에 적합한 **서버 크기**는 무엇입니까?

내 서버의 **활용도**를 높이려면 어떻게 해야 합니까?

내 성능에 적합한 **서버 크기**는 무엇입니까?

서버가 **손상**되었는지 있습니까?

내 서버에서 액세스를 **제어**하려면 어떻게

하려면 어떻게 해야 합니까?

Server

내 서버의 남은 **용량**은 얼마입니까?

몇 대의 서버에 대한 **예산**을 책정해야 합니까?

내 서버에 너무 많은 로드를 생성합니까?

응용 프로그램은 서버 **하드웨어 오류**를 어떻게 처리합니까?

내 서버는 **어떤 OS**를 실행해야 합니까?

내 서버 운영 체제를 **패치** 상태로 유지하려면 어떻게 해야 합니까?

내 서버 이미지에 **어떤 패키지**를 구워야 합니까?

내 애플리케이션을 **최적화**하려면 OS 설정을 조정해야 합니까?

새 **코드**가 내 서버에 어떻게 배포됩니까?

서버 **확장**은 언제 결정해야 합니까?

What is Serverless?

https://stackify.com/function-as-a-service-serverless-architecture/

Serverless 개념

What is Serverless?

@Wikipedia

"Serverless" is a misnomer in the sense that servers are still used by cloud service providers to execute code for developers. However, developers of serverless applications are not concerned with capacity planning, configuration, management, maintenance, fault tolerance, or scaling of containers, VMs, or physical servers.

Serverless 개념

What is Serverless?

Serverless computing does not hold resources in volatile memory; computing is rather done in short bursts with the results persisted to storage. When an app is not in use, there are no computing resources allocated to the app. Pricing is based on the actual amount of resources consumed by an application. It can be a form of utility computing.

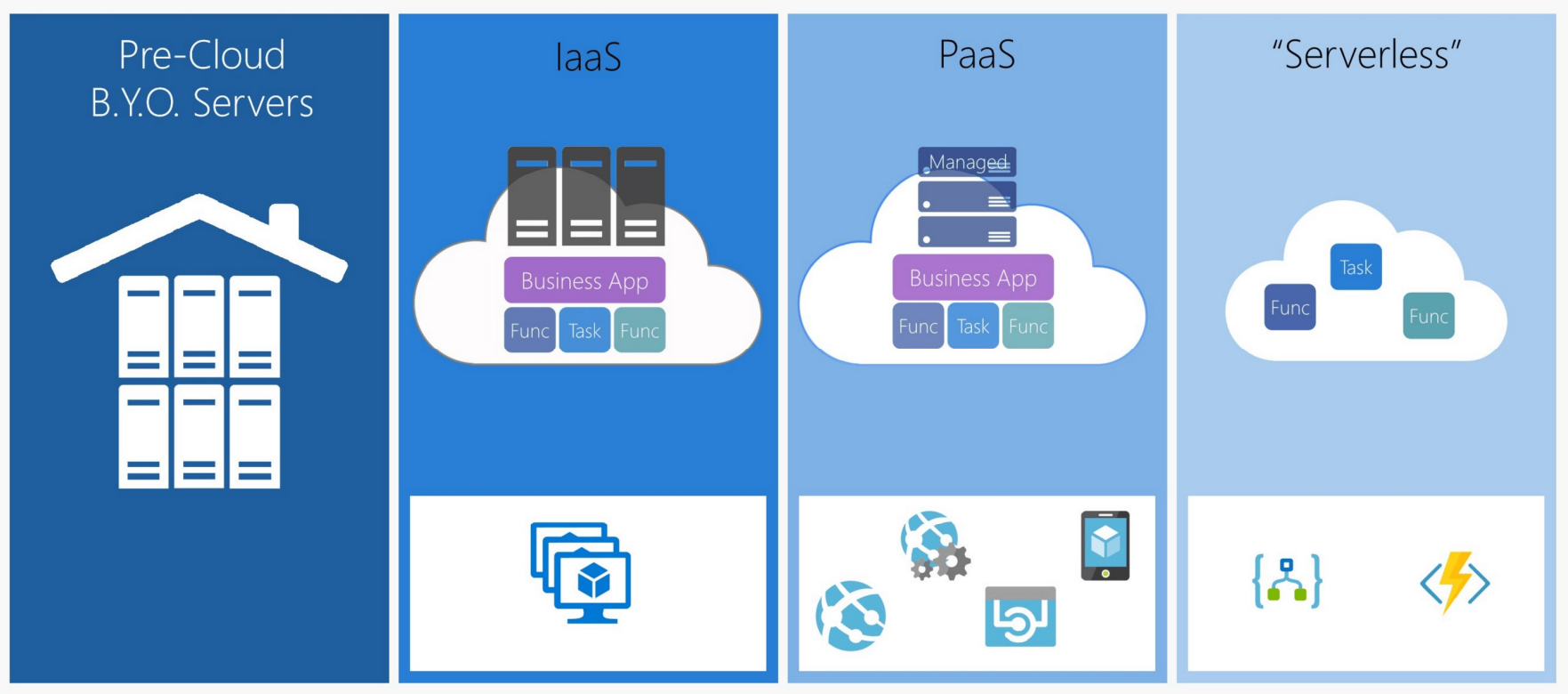
Serverless 개념

What is Serverless?

Serverless computing can simplify the process of deploying code into production. Serverless code can be used in conjunction with code deployed in traditional styles, such as microservices or monoliths. Alternatively, applications can be written to be purely serverless and use no provisioned servers at all. This should not be confused with computing or networking models that do not require an actual server to function, such as peer-to-peer (P2P).

Serverless 개념

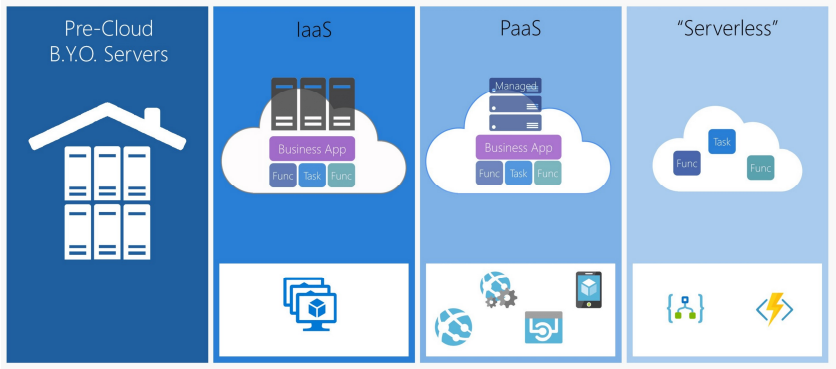
What is Serverless?



https://stackify.com/function-as-a-service-serverless-architecture/

Serverless 개념

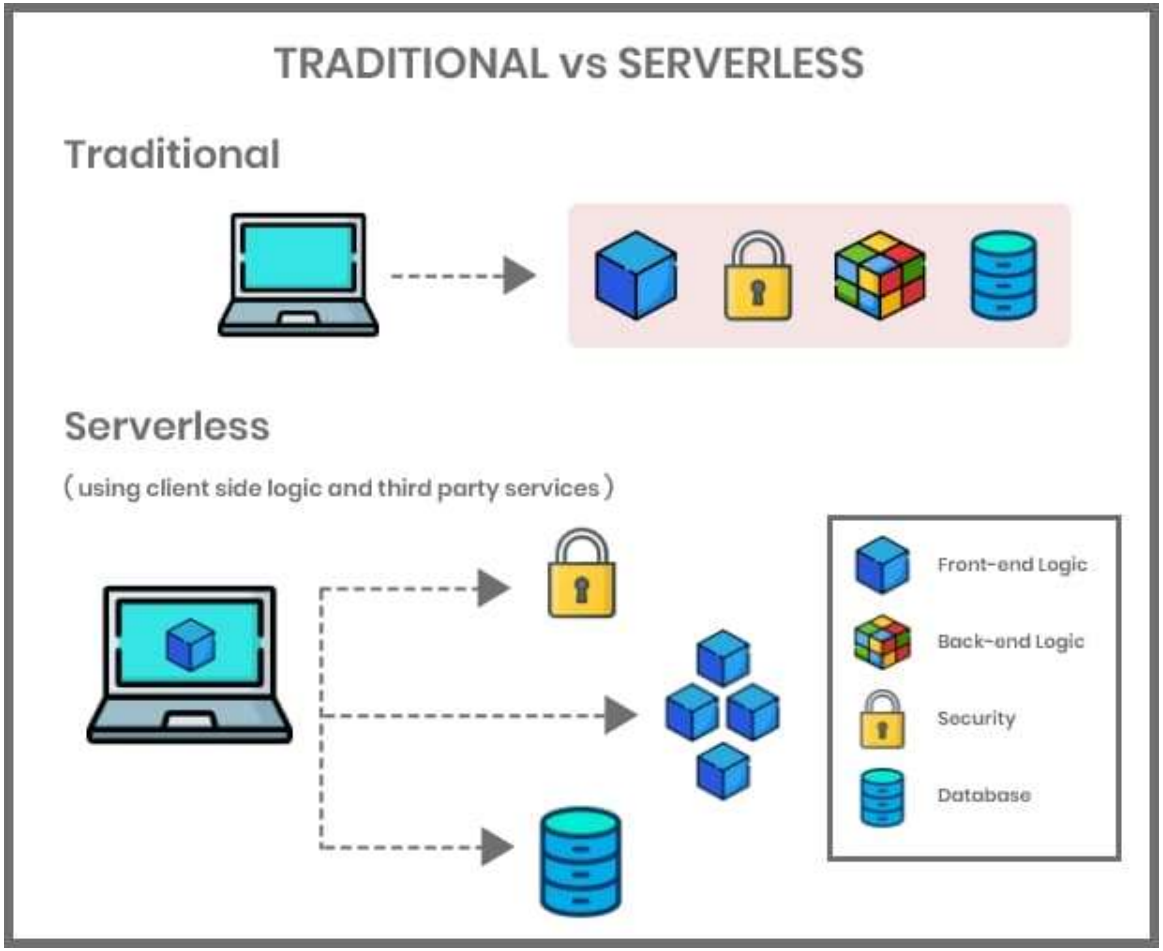
What is Serverless?



- Complete abstraction of servers away from the developer
- Billing based on consumption and executions, not server instance sizes
- Services that are event-driven and instantaneously scalable

Serverless 개념

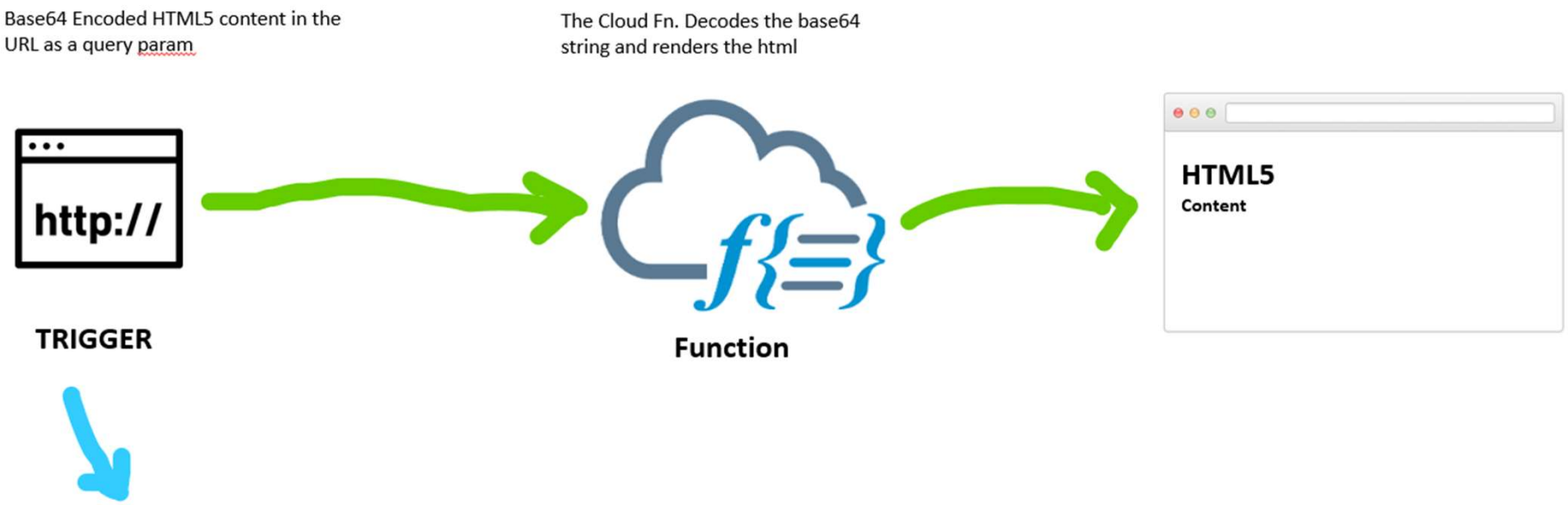
What is Serverless?



https://kinsta.com/blog/function-as-a-service/

Serverless 개념

What is Serverless?



https://<faas_http_trigger_url>/<function_name>/?q=<base64_encoded_html5_content>

Ex.

```
https://a46b334a-6c53-4e7d-944f-e8758537b4c5.ingress.live.faas-live.shoot.live.k8s-hana.ondemand.com/f1/?
q=PGg2Pg0KTGl0dGxlIFBvdGF0bw0KPC9oNj4NCjxoMT4NCkJpZyBQb3RhdG8NCjwvaDE+
```

https://www.validatek.com/technologies/function-service-faas

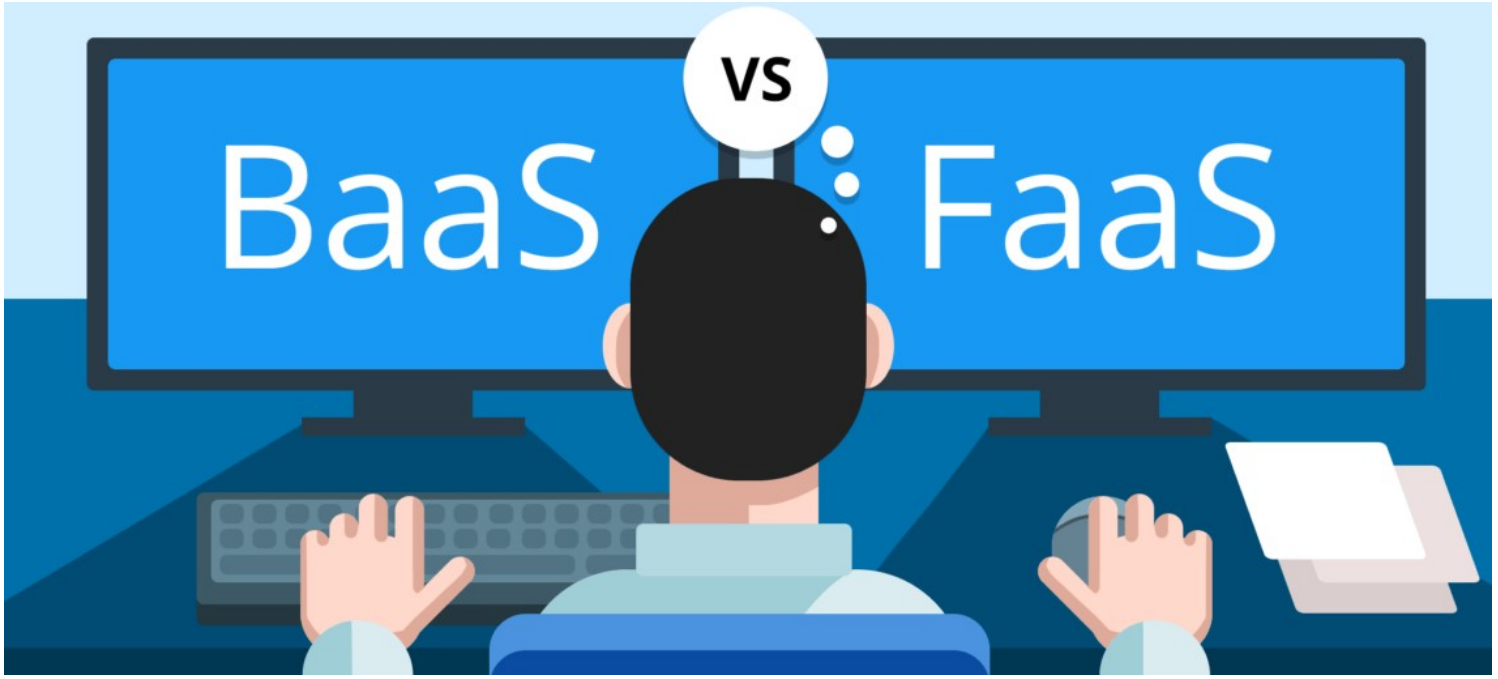
Serverless 개념

서버리스란?

https://github.com/cncf/landscape/issues/175

- 개발자가 서버를 관리할 필요없이 애플리케이션을 빌드하고 실행할 수 있도록 하는 클라우드 네이티브 개발 모델
- 서버를 관리할 필요가 없다 ➔ IaaS와 같은 모델처럼 트래픽에 따라 사용자가 직접 서버의 가용량을 증/감시킬 필요가 없다는 뜻.

서버리스 종류



https://blog.back4app.com/baas-vs-faas/

Serverless 개념

서버리스 종류

AWS Lambda

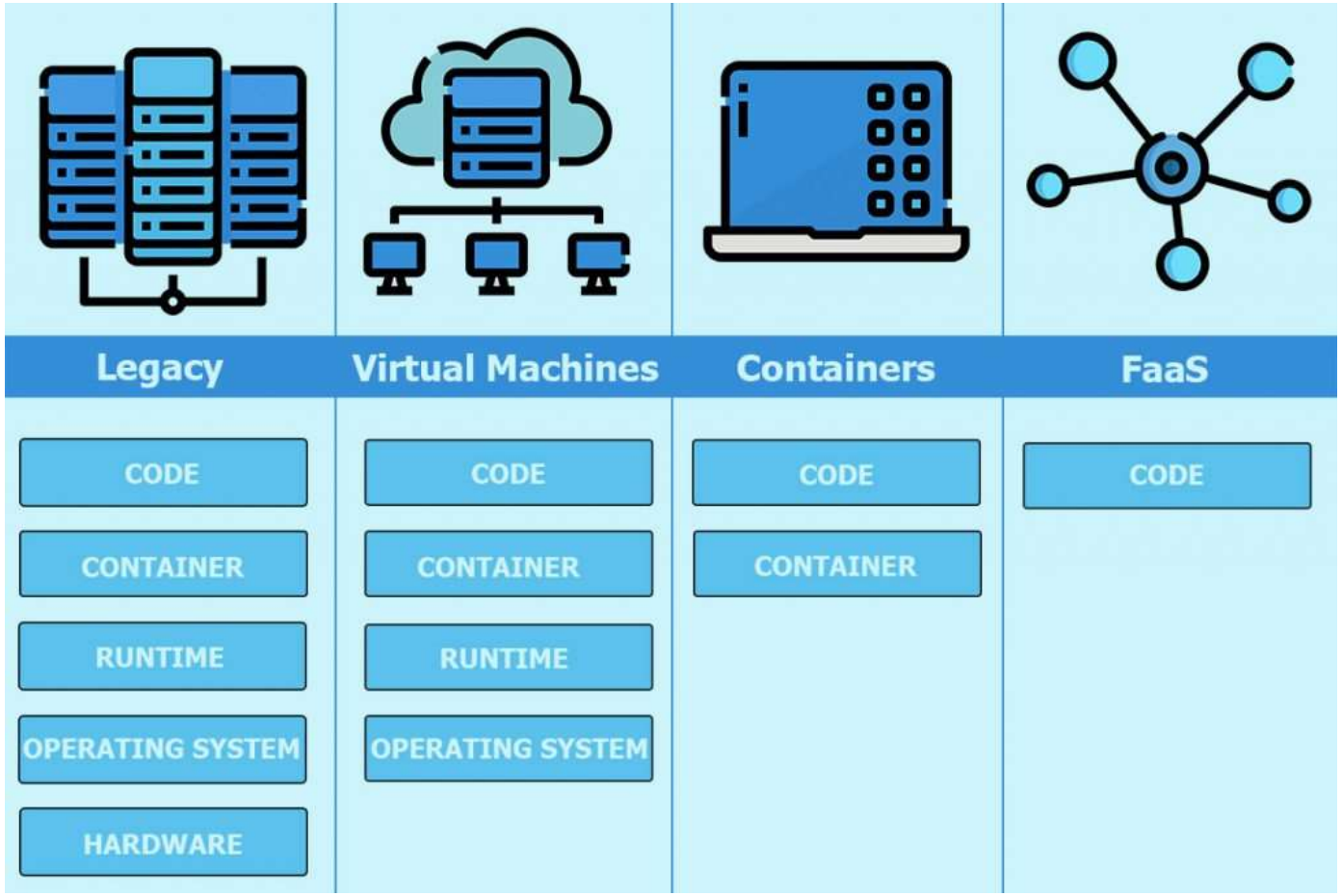
FaaS

BaaS

- 함수를 서비스로 제공(Function as a Service)
- 업로드한 코드를 함수 단위로 나누어 대기 상태
- 요청이 들어오면 서버가 대기 상태의 함수를 실행
- 비용은 함수 호출 횟수에 따라 청구
- AWS Lambda, MS Azure Function

Serverless 개념

서버리스 종류



https://www.webapper.com/case-for-functions-as-a-service/

서버리스 종류

FaaS

BaaS

- Backend as a Service
- 백엔드 개발에 필요한 여러 기능을 API로 제공하는 서비스
- SNS연동이나 DB와 같이 백엔드에 필요한 기능들을 사용자가 직접 구현할 필요없이 제공하는 API로 해당 기능을 구현
- Firebase

Serverless 개념

서버리스 장점

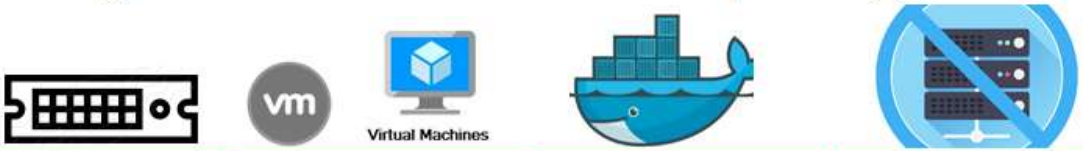
https://stackify.com/function-as-a-service-serverless-architecture/

- 가격
  - 실제 사용량에 대해서만 비용 청구
- 애플리케이션의 품질에 집중
  - 서버에 집중할 필요없이 애플리케이션 품질 향상에 좀 더 집중 가능
- 높은 가용성과 유연한 확장
  - 요청이 들어올 때만 실행되고
  - 동적으로 자원이 할당
  - 스케일링에 대한 고민 불필요.

Serverless 개념

서

Advantages of Serverless Computing



	Bare Metal	VM	Container	Serverless
Boot Time	~20 mins	~2 mins	2 secs	~0.0003 secs
App deployment lifecycle	Deploy in Weeks Live for years	Deploy in minutes Live for weeks	Deploy in Seconds Live for minutes/hours	Deploy in milliseconds Live for seconds
Development Complexity	Need to know: 1. Hardware 2. OS 3. Runtime Environm 4. Application code	Need to know: 1. OS 2. Runtime Environme 3. Application code	Need to know: 1. Runtime Environment 2. Application code	Need to know: 1. Application code
Investment	Buy/rent dedicated server	Rent a dedicated VM, on a shared server	Rent Containers, pay for the actual runtime	Pay for compute resouces used during runtime
Scaling	Takes months Should be approved by a panel of experts	Takes hours Should be approved by adminstators	Takes seconds Policy driven scaling	Takes milliseconds Scaling is event driven

https://3.bp.blogspot.com/-WfMUSEeaR84/WfFBaEVKdjI/AAAAAAAAMyM/_FlHqv5TaOIIi4fzlna90jHCqpO6CJB9ACLcBGAs/s320/Advantages%2Bof%2BServerless%2BComputing.png

서버리스 단점

https://github.com/cncf/landscape/issues/175

- Cold Start
  - 상시 대기가 아닌 요청이 들어올 때 시작
- 클라우드 제공 플랫폼에 더욱 종속적
  - 타 플랫폼 이전이 힘듦
- 긴 시간이 필요한 작업에 불리
  - 동영상 업로드, 데이터 백업 등.

Lab1. AWS Lambda Hello World

Serverless in AWS

AWS Lambda

https://aws.amazon.com/ko/lambda/

Serverless in AWS

AWS Serverless :

4 Main Benefits

- No Server Management
- Flexible Scaling
- Highly Available
- No Idle

Serverless in AWS

Best practices are built-in with AWS Serverless

- Microservices by design
- Fault tolerant by design
- Scalable by design
- Logging and metrics built-in

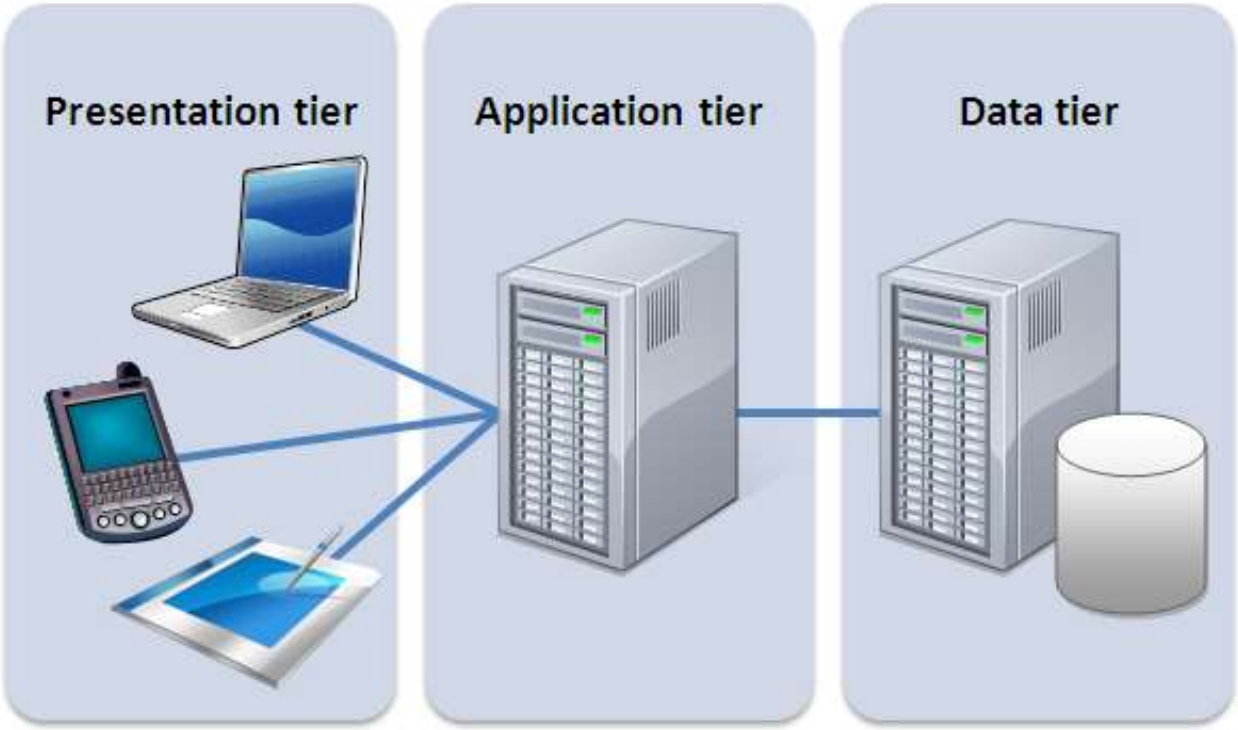
Best practices are built-in with AWS Serverless

No idle capacity by design

Find-grained throttling built-in

Retries and DLQ built-in

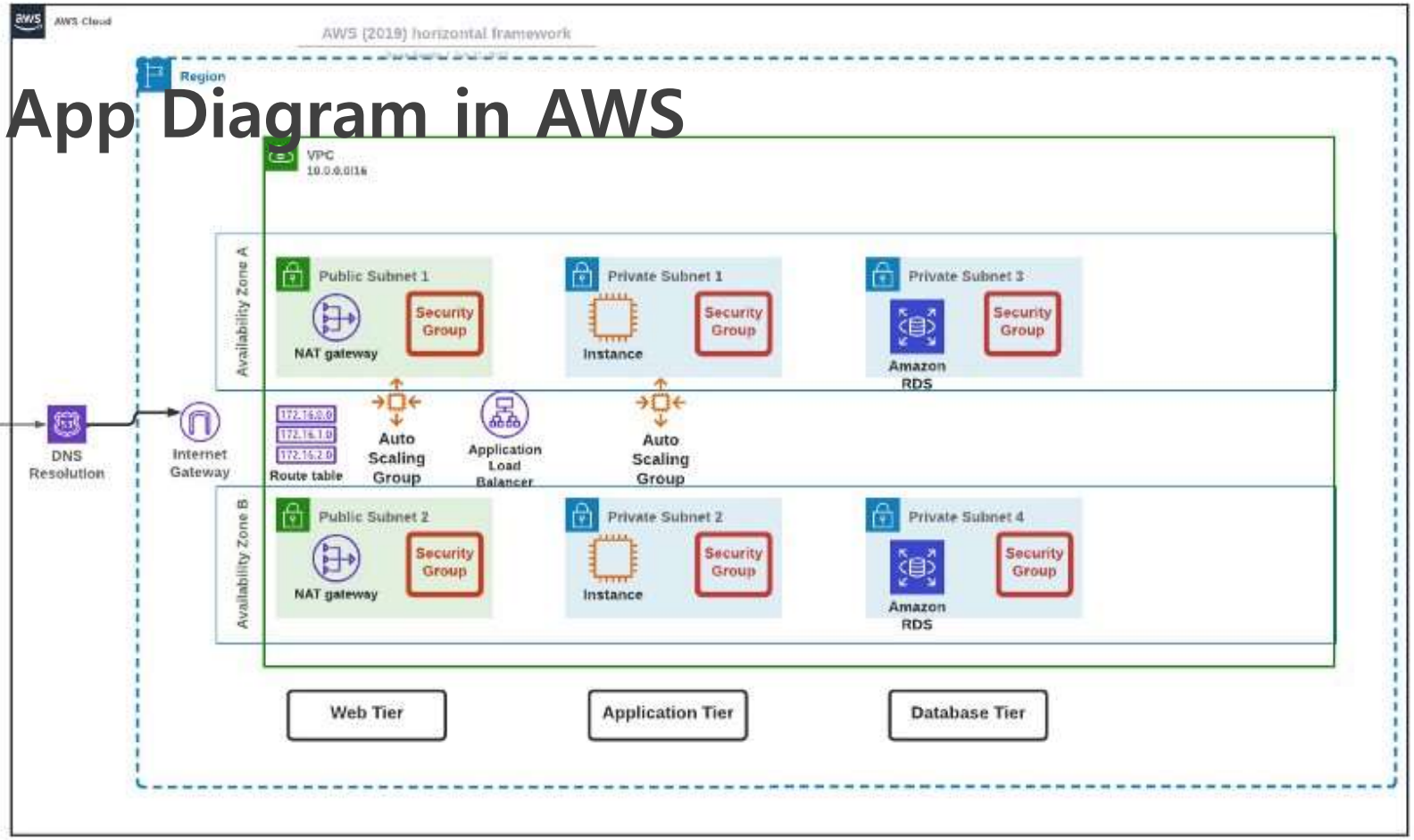
General 3-tier App Diagram



https://managementmania.com/en/three-tier-architecture

Serverless in AWS

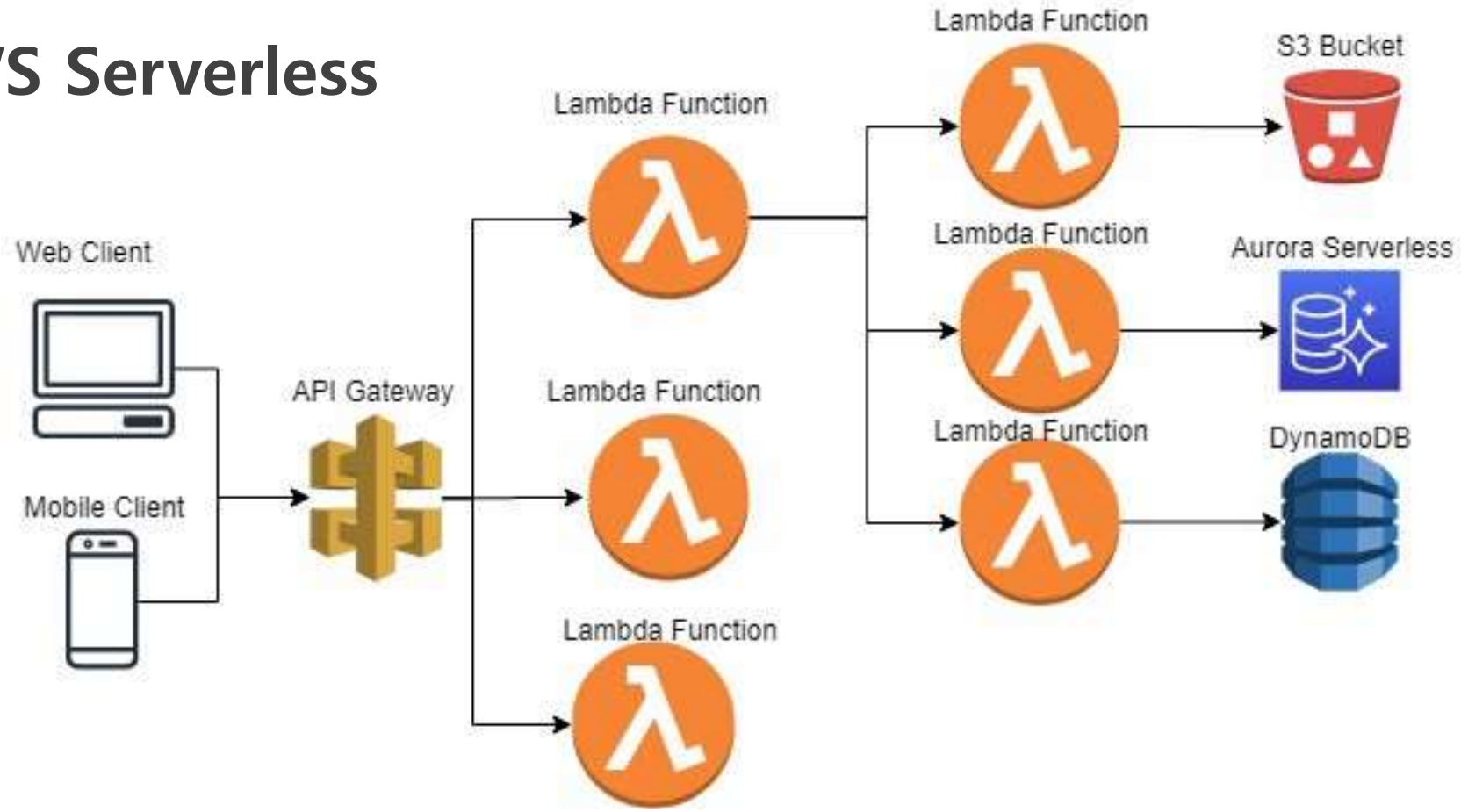
3-tier App Diagram in AWS



https://medium.com/@bryanJR/3-tier-architecture-on-aws-626173d15ba0

Serverless in AWS

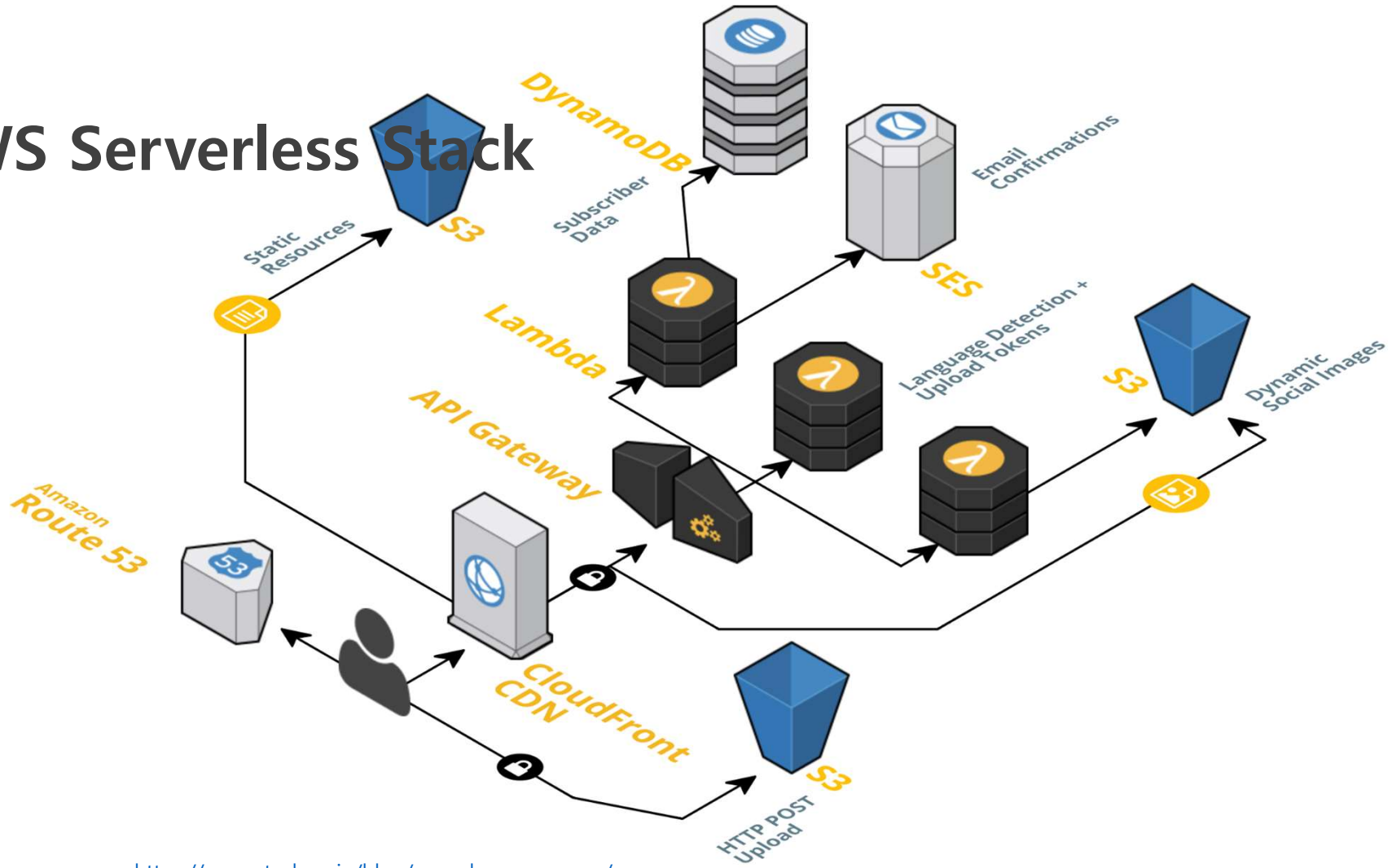
AWS Serverless



https://levelup.gitconnected.com/deploying-microservices-using-serverless-architecture-cf7d1570950

Serverless in AWS

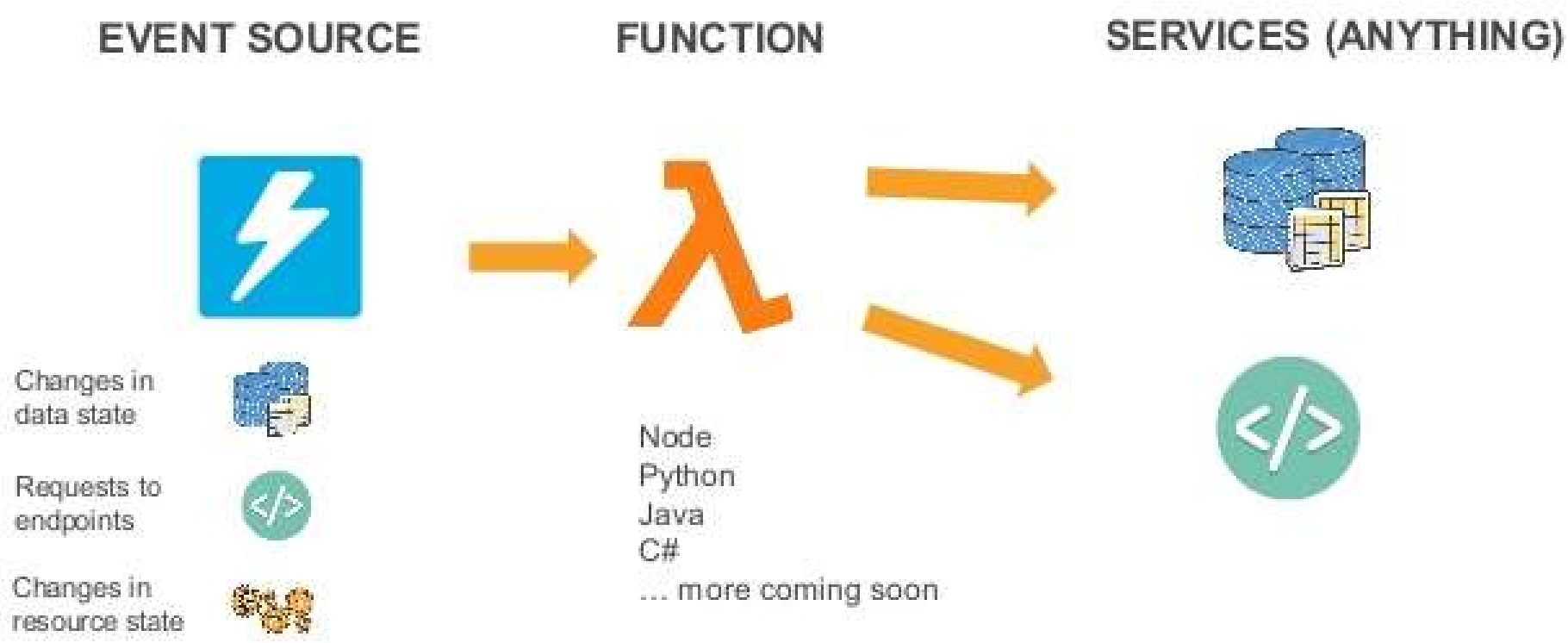
AWS Serverless Stack



https://www.stackery.io/blog/serverless-use-cases/

Serverless in AWS

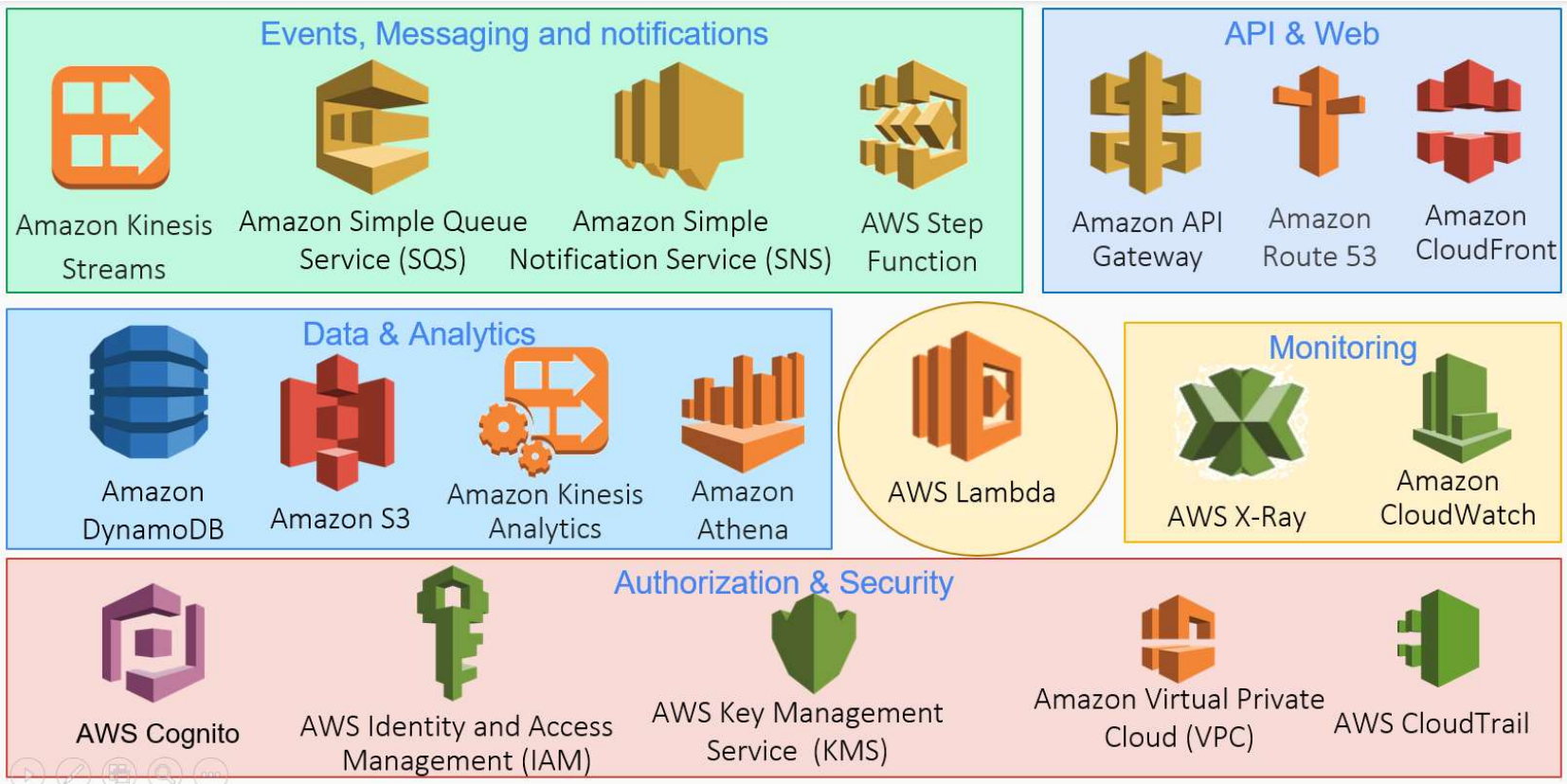
AWS Serverless Applications



https://vticloud.io/en/gioi-thieu-dich-vu-aws-lambda-va-cach-cau-hinh-cho-nguoi-moi-bat-dau/

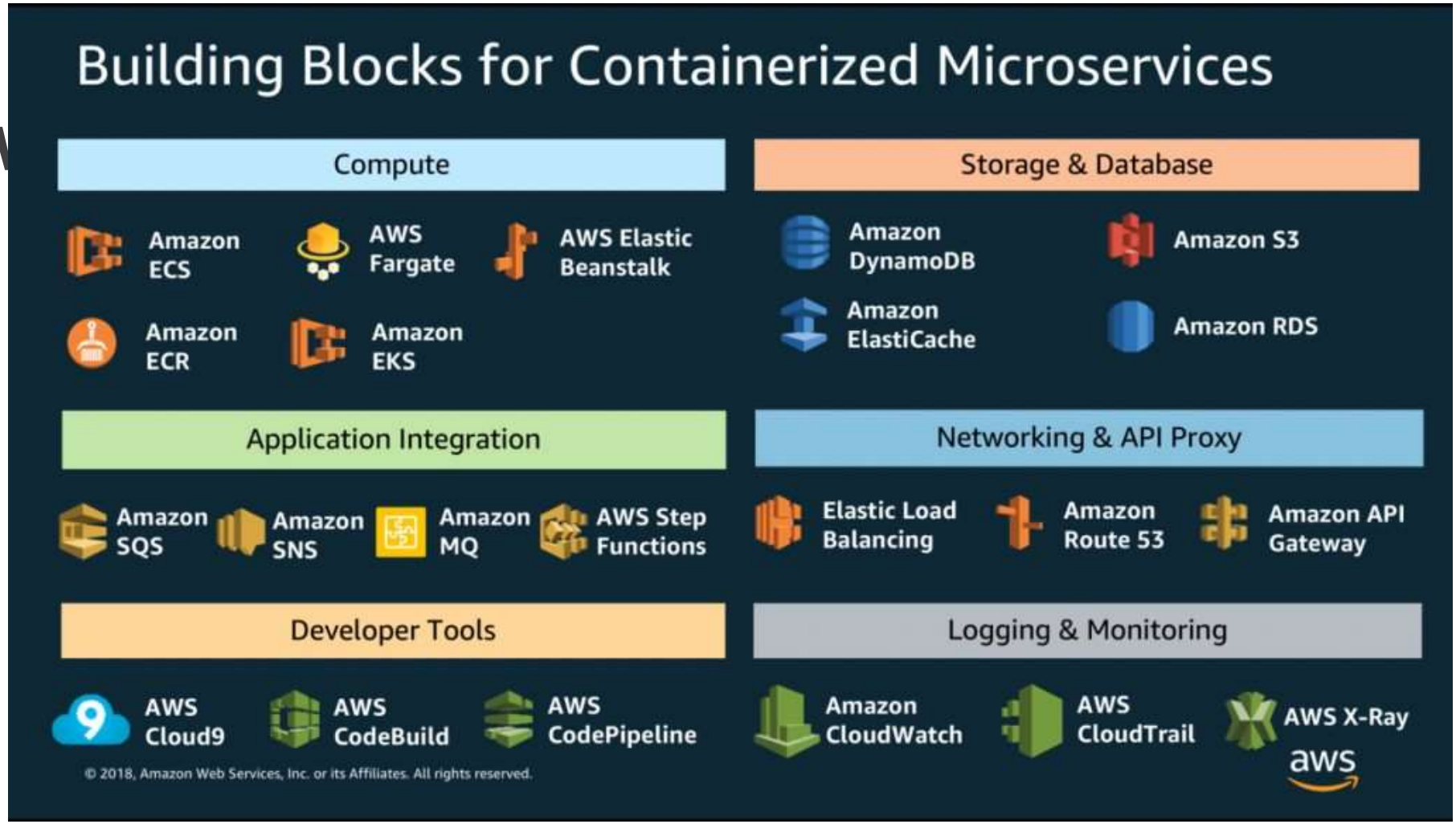
Serverless in AWS

A Mature Serverless Portfolio in AWS



https://www.mapyourown.com/blog/serverless-development-cloud

Serverless in AWS



https://aws.amazon.com/ko/blogs/opensource/microservices-on-aws-using-containers-serverless/

Serverless Use Cases

Serverless Use Cases

Common Use Cases

Web Applications

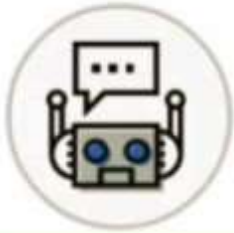
- Static websites
- Complex web apps
- Packages for Flask and Express

Backends

- Apps & services
- Mobile
- IoT

Data Processing

- Real time
- MapReduce
- Batch



Chatbots

- Powering chatbot logic

Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit

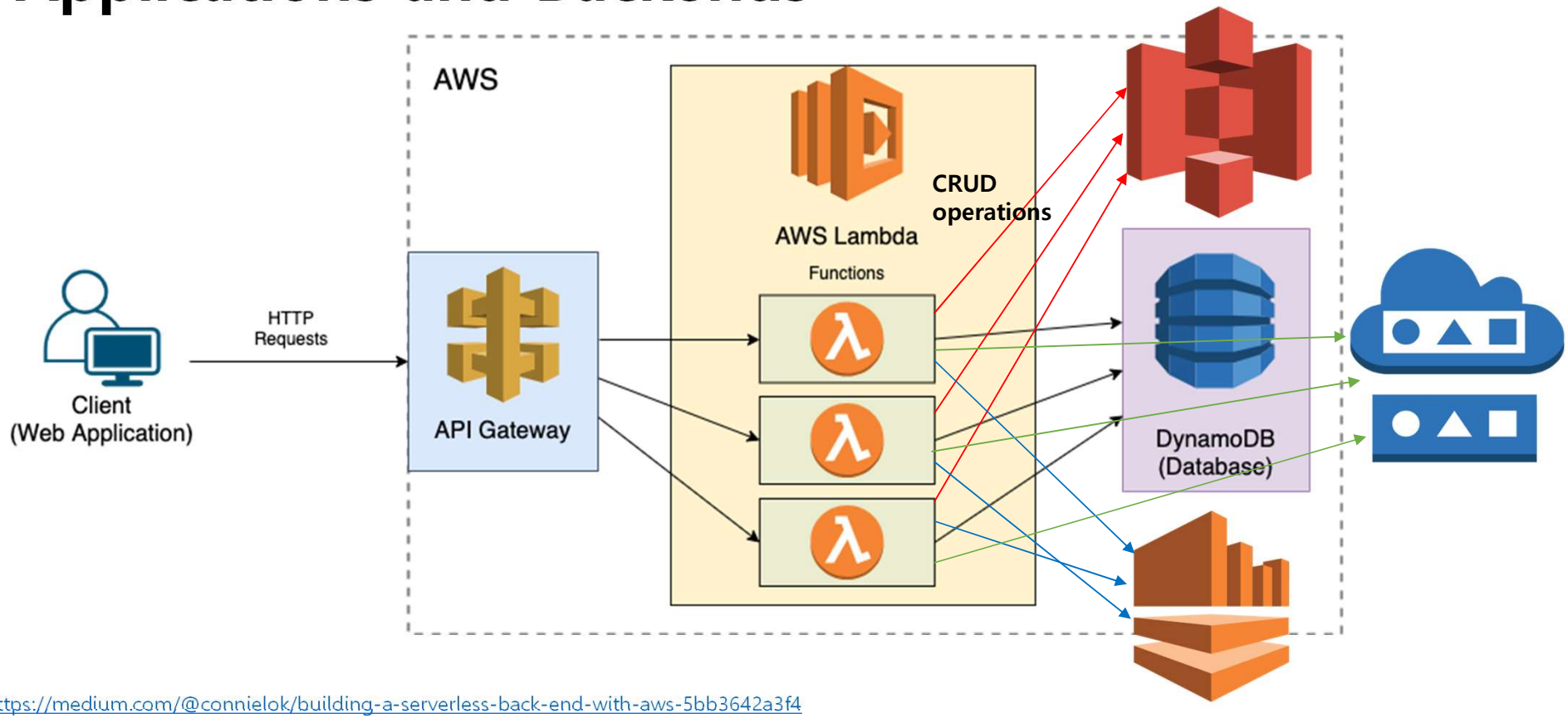
IT Automation

- Policy engines
- Extending AWS services
- Infrastructure management

https://kruschecompany.com/serverless-architecture-for-modern-apps-providers-and-caveats/

Serverless Use Cases

Web Applications and Backends



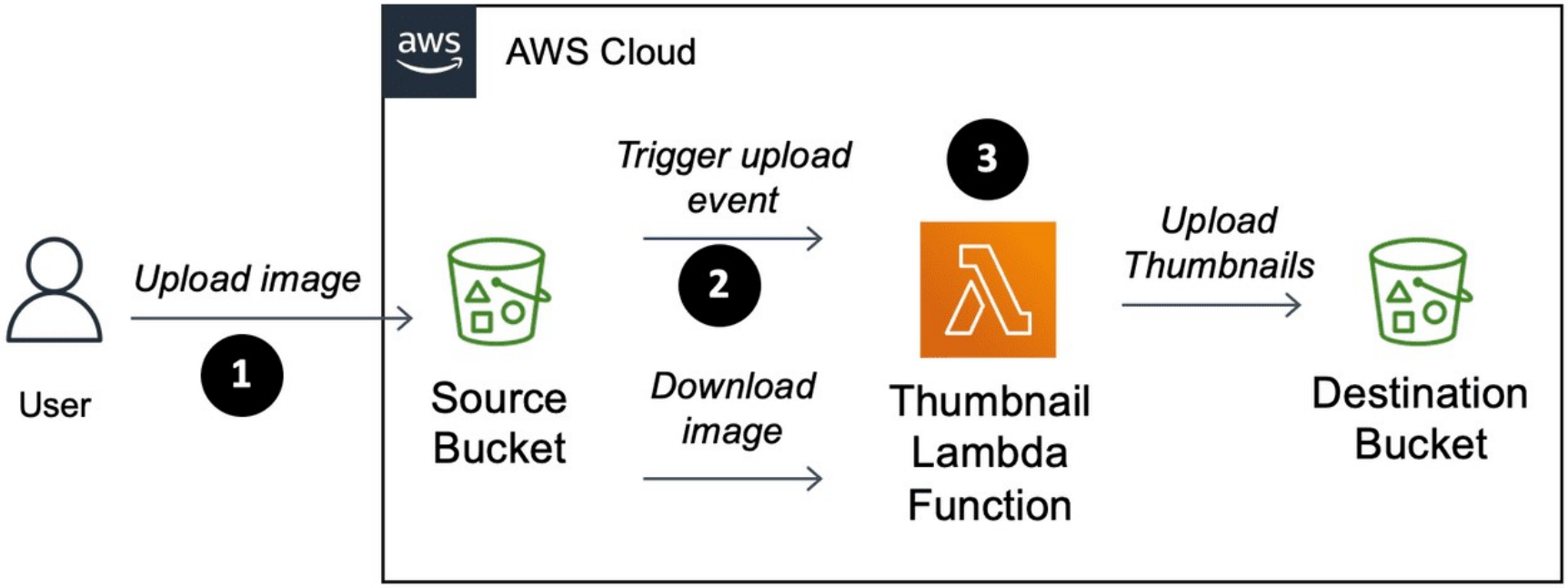
https://medium.com/@connielok/building-a-serverless-back-end-with-aws-5bb3642a3f4

Serverless Use Cases

Backend Order Processing

Serverless Use Cases

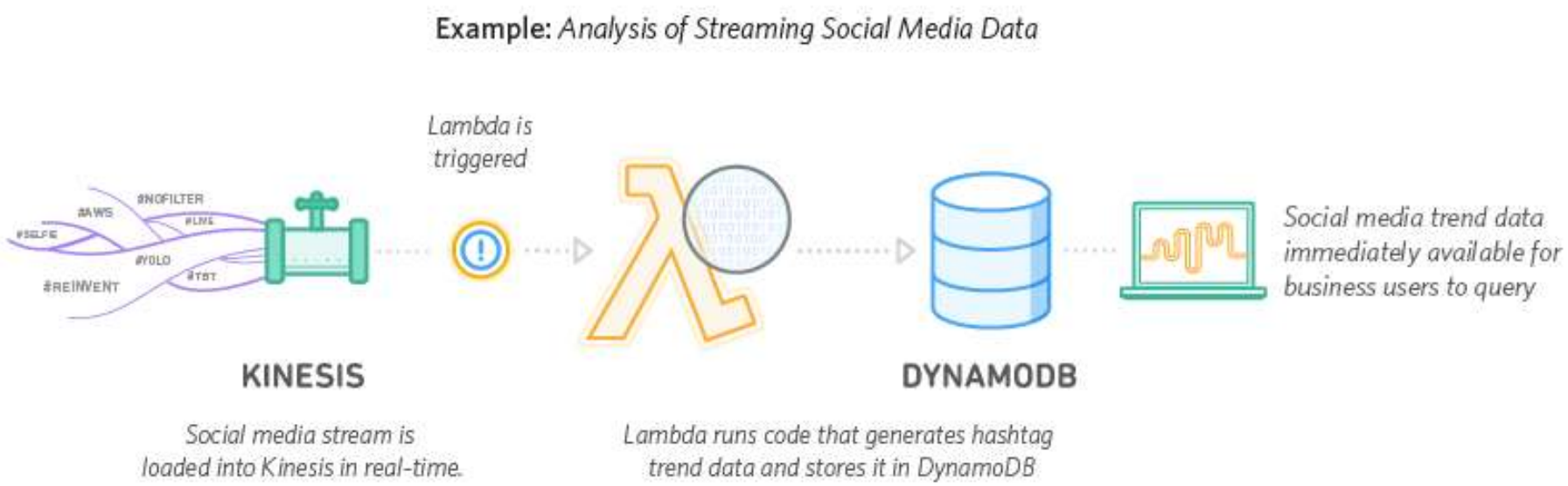
Image Thumbnail Generation



https://lukstei.com/2021-11-21-creating-a-serverless-thumbnail-service-with-aws-cdk/

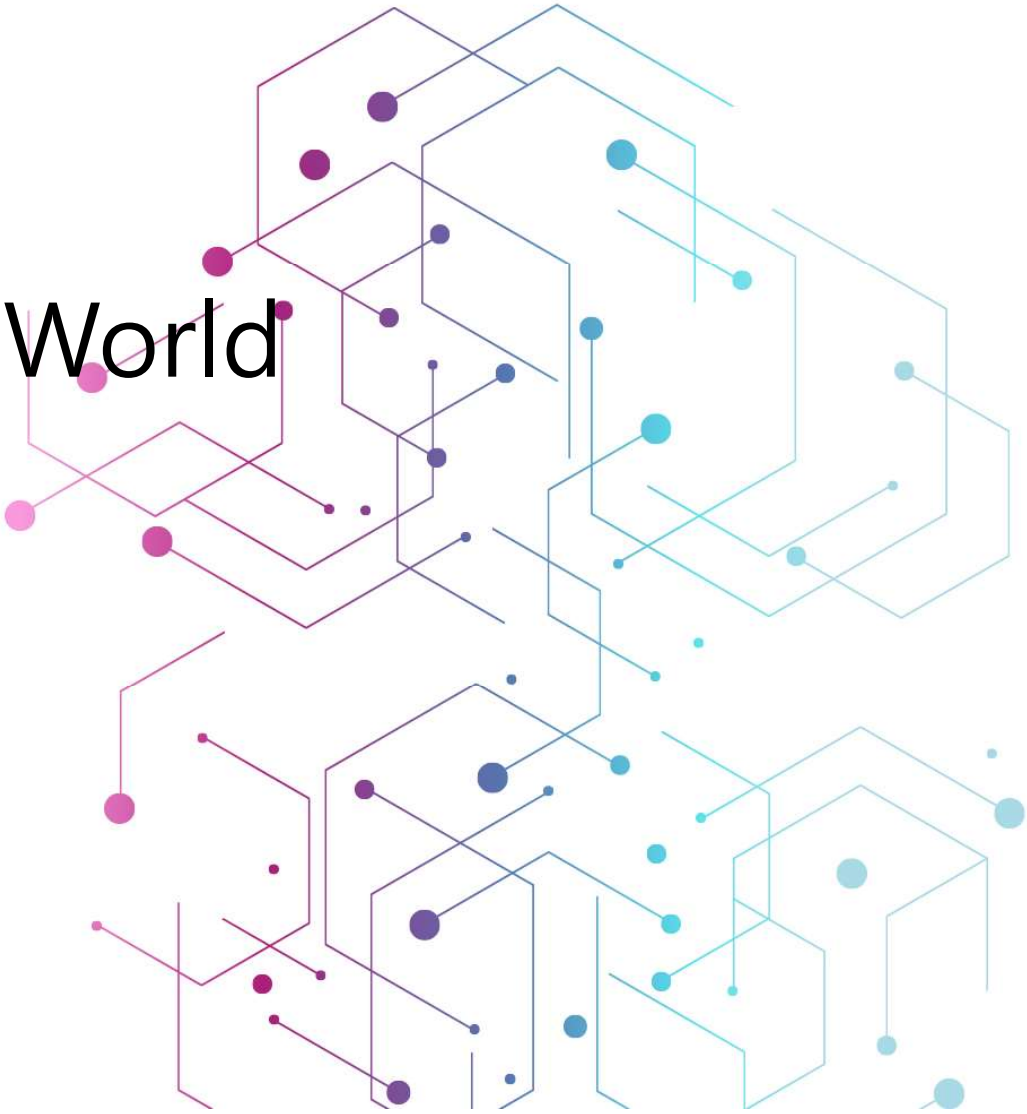
Serverless Use Cases

Data Processing



https://aws.amazon.com/ko/lambda/data-processing/

Lab2. Serverless Hello World



Epilogue

Epilogue

Additional Resources

- AWS Serverless Home
- AWS re:Invent 2017:Getting Started with Serverless Computing Using AWS Lambda
- AWS re:Invent 2017:Become a Serverless Black Belt:Optimizing Your Serverless Applications
- AWS re:Invent 2017:GPS:Real-Time Data Processing with AWS Lambda Quickly, at Scale
- Using AWS Lambda with Kinesis
- Lambda 시작하기