

AWS Serverless Computing

Jae Hyeon Kim

— 목차

1. Serverless Computing
2. AWS Lambda
3. AWS Fargate

Serverless Computing

— Serverless

- 서버리스(serverless)란 개발자가 서버를 관리할 필요 없이 애플리케이션을 빌드하고 실행할 수 있도록 하는 클라우드 네이티브 개발 모델
- 서버리스 모델은 서버가 추상화 되어있다. 클라우드 제공업체가 서버 인프라에 대한 프로비저닝, 유지 관리, 스케일링 등의 일상적인 작업을 처리하며, 개발자는 배포를 위해 코드를 컨테이너에 패키징만 하면 된다.

— Serverless Architecture

- 서버리스는 클라우드 제공업체가 클라우드 인프라와 애플리케이션의 스케일링을 모두 관리한다는 점에서 다른 클라우드 컴퓨팅 모델과 차이를 보인다. 서버리스 애플리케이션은 호출 시 온디맨드로 자동 시작되는 컨테이너에 배포된다.

— Serverless Architecture

- IaaS 클라우드 컴퓨팅 모델에서 사용자는 용량 단위를 사전 구매하게 된다. 즉, 애플리케이션을 구동하기 위해 퍼블릭 클라우드 공급업체에서 상시 가동 중인 서버 구성 요소에 대한 비용을 지불해야 한다.
- 애플리케이션을 구동하기 위해 필요한 클라우드 인프라는 애플리케이션이 사용되지 않을 때에도 활성화된 상태이다.

— Serverless Architecture

- 서버리스 아키텍처에서는 애플리케이션이 필요한 경우에만 시작된다. 이벤트가 구동을 위한 애플리케이션 코드를 트리거하면 퍼블릭 클라우드 공급업체가 신속하게 해당 코드에 대한 리소스를 할당한다. 코드 실행이 종료되면 비용도 청구되지 않는다.
- 비용과 효율성이라는 이점 이외에도, 서버리스는 애플리케이션 스케일링 및 서버 프로비저닝과 같은 일상적이고 사소한 태스크에서 개발자의 부담을 덜어준다.

AWS Lambda

— AWS Lambda

- AWS Lambda는 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있게 해주는 컴퓨팅 서비스
- Lambda는 고가용성 컴퓨팅 인프라에서 코드를 실행하고 서버와 운영체제 유지 관리, 용량 프로비저닝 및 자동 조정, 코드 및 보안 패치 배포, 로깅 등 모든 컴퓨팅 리소스 관리를 수행한다.
- Lambda를 사용하면 거의 모든 유형의 애플리케이션 또는 백엔드 서비스에 대한 코드를 실행할 수 있다.

— AWS Lambda 특징

- 100% 서버리스
- 함수 단위의 실행 환경
- 다양한 프로그래밍 언어 및 컨테이너 지원
- 자동 스케일링, 높은 가용성
- 호출 기반의 비용
- 다양한 코드 작성 및 배포 모델

— AWS Lambda SLA

- 서비스 약속

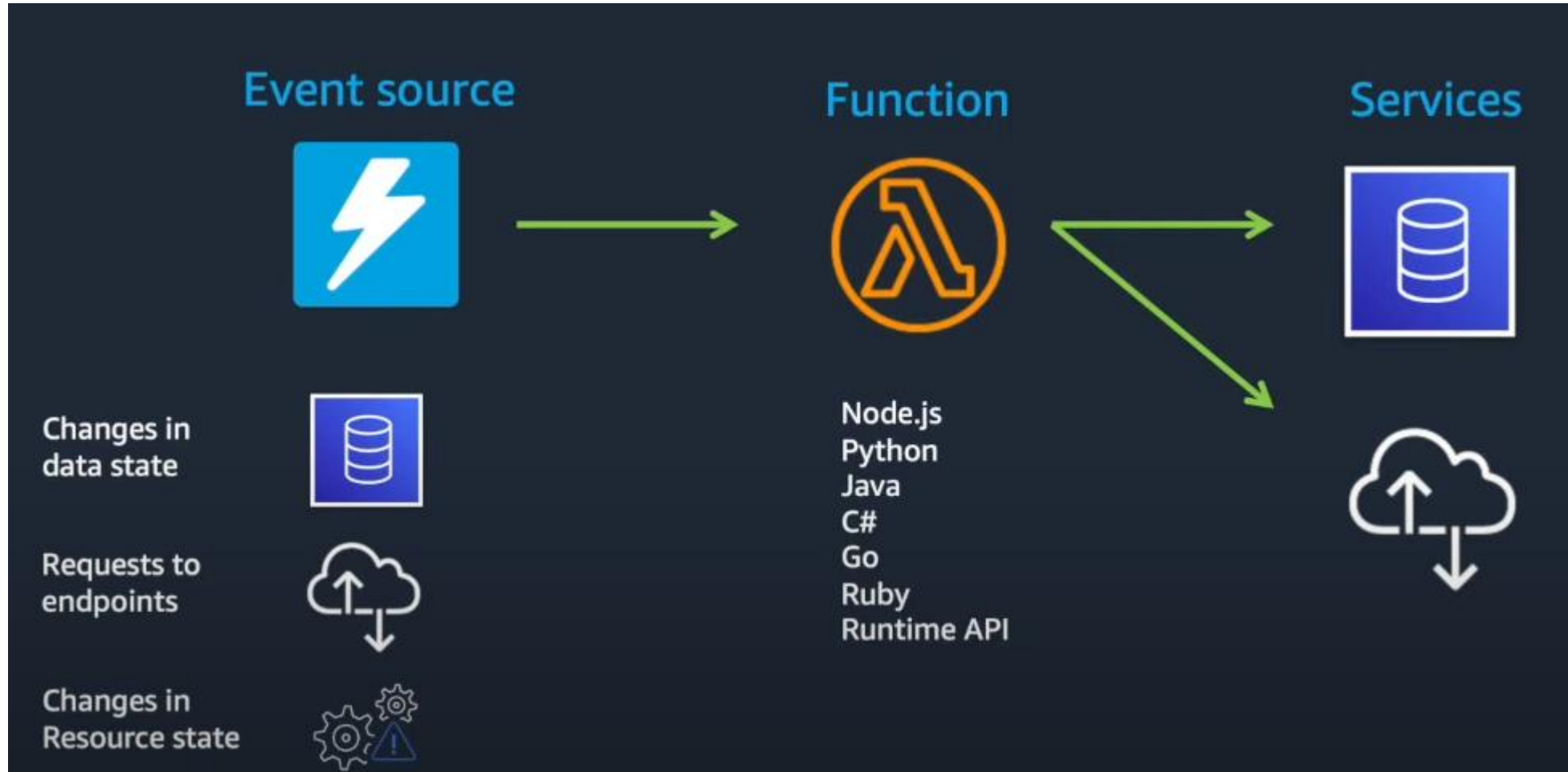
- AWS는 월별 청구 주기 동안 각 AWS 리전에 최소 99.95%의 월간 가동률로 Lambda를 사용할 수 있도록 약속

- 서비스 크레딧

- 서비스 크레딧은 월간 가동 시간 백분율이 아래 표에 명시된 범위에 속하는 월별 청구 주기 동안 영향을 받는 AWS 리전에서 사용자가 Lambda에 대해 지불한 총 요금의 백분율로 계산

월간 가동 시간 비율	서비스 크레딧 비율
99.95% 미만 99.0% 이상	10%
99.0% 미만 95.0% 이상	25%
95.0% 미만	100%

— AWS Lambda Application



— AWS Lambda 기능

- 동시성 및 크기 조정 컨트롤
- 컨테이너 이미지로 정의된 함수
- 코드 서명
- Lambda 익스텐션
- 함수 블루프린트
- 데이터베이스 액세스
- 파일 시스템 액세스

— Lambda 함수 크기 조정

- 함수를 최초로 호출하면 AWS Lambda는 함수의 인스턴스를 생성하고 핸들러 메서드를 실행하여 이벤트를 처리한다.
- 함수가 응답을 반환하면 활성 상태를 유지하고 추가 이벤트를 처리하기 위해 대기한다.
- 첫 번째 이벤트가 처리되는 동안 함수를 다시 호출하면 Lambda는 다른 인스턴스를 초기화하고 함수는 두 이벤트를 동시에 처리한다.
- 이벤트가 추가로 수신되면 Lambda는 이 이벤트를 사용 가능한 인스턴스로 라우팅하고 필요 시 새 인스턴스를 생성한다.

— Lambda 함수 크기 조정

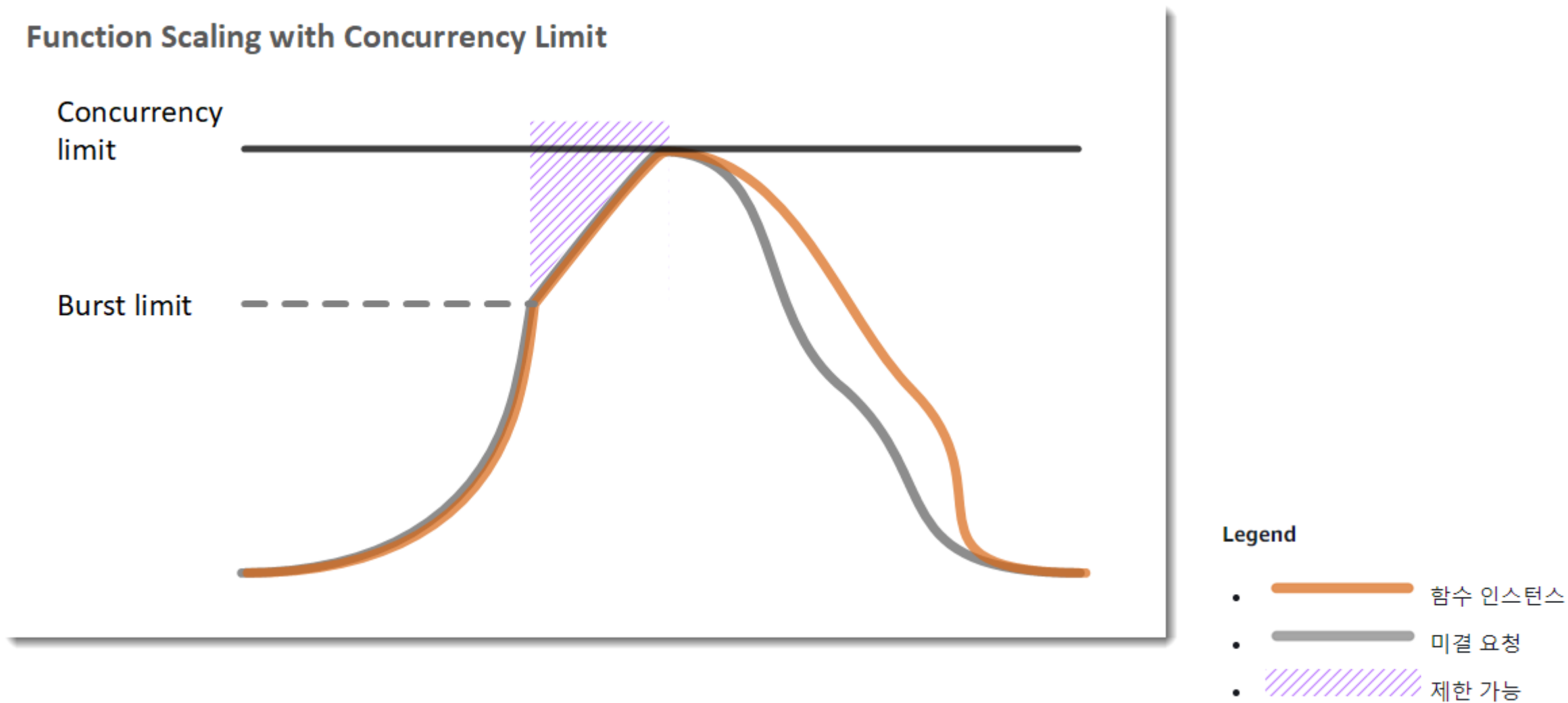
- 함수의 동시성은 특정 시각에 요청을 처리하는 인스턴스의 수
- 초기 트래픽 버스트의 경우, 리전에서 함수의 누적 동시성은 500~3000의 최초 레벨에 도달할 수 있으며, 이는 리전에 따라 달라진다.

버스트 동시성 할당량

- - 3,000 미국 서부(오리건), 미국 동부(버지니아 북부), 유럽(아일랜드)
 - - 1,000 아시아 태평양(도쿄), 유럽(프랑크푸르트), 미국 동부(오하이오)
 - 500 - 기타 리전
- 최초 버스트 이후 함수의 동시성은 매분 500개의 추가 인스턴스까지 확장될 수 있다.

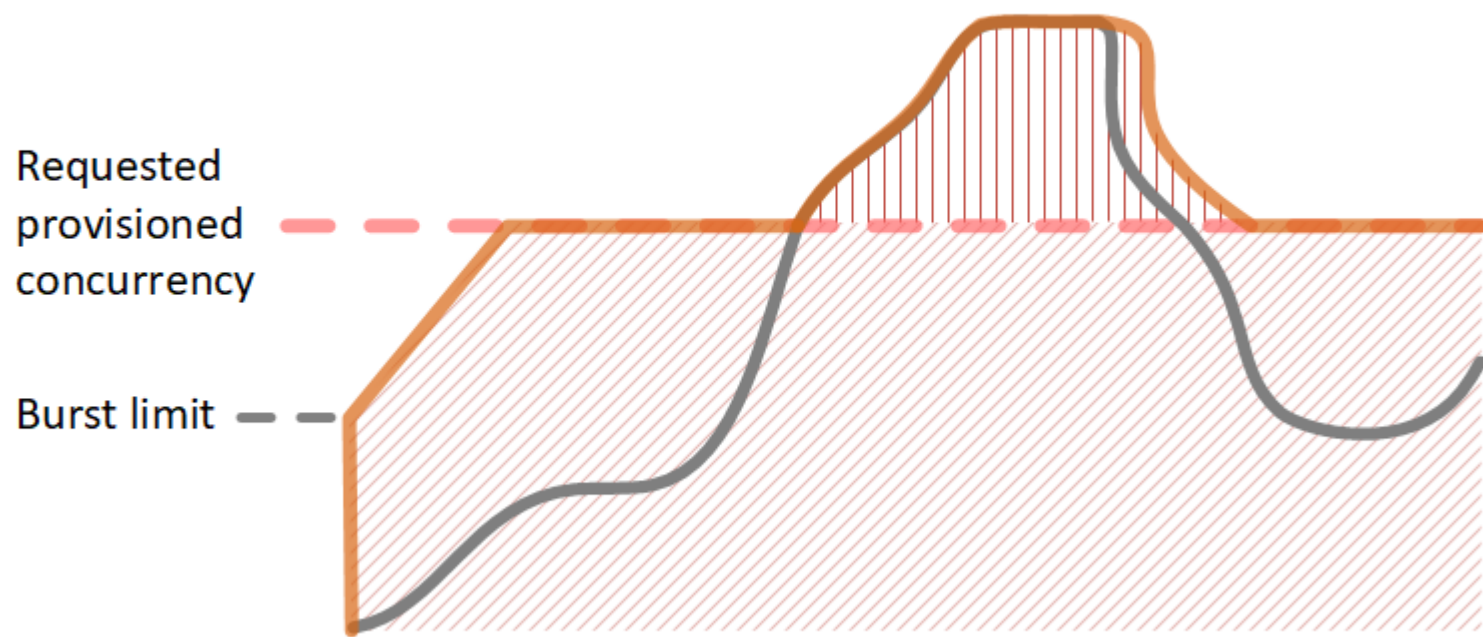
— Lambda 함수 크기 조정

Function Scaling with Concurrency Limit







— Lambda 함수 크기 조정

Function Scaling with Provisioned Concurrency



Legend

-  함수 인스턴스
-  미결 요청
-  프로비저닝된 동시성
-  표준 동시성

— AWS Lambda 함수

Handler() function

Function to be executed upon invocation

Event object

Data sent during Lambda function Invocation

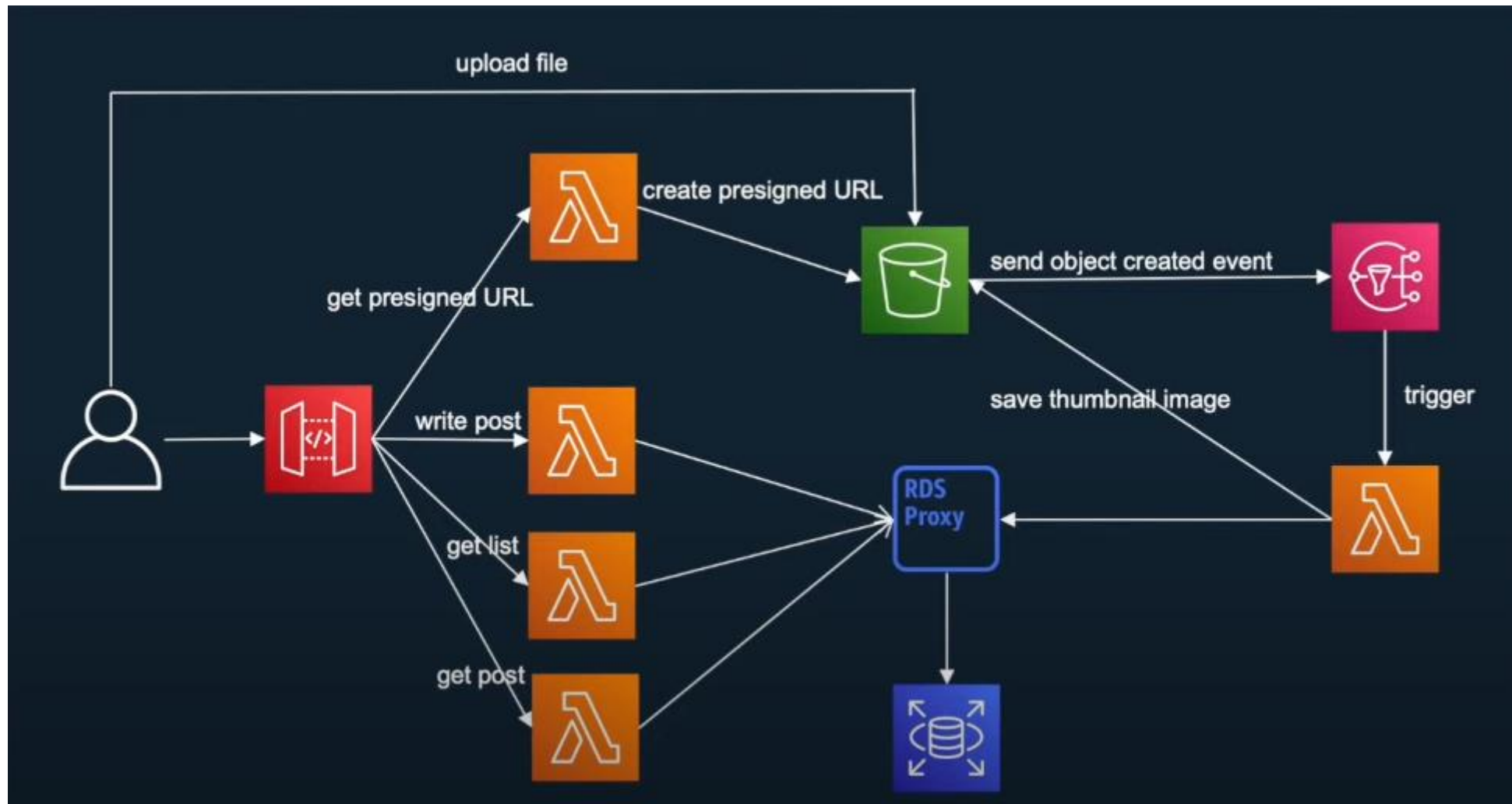
Context object

Methods available to interact with runtime information (request ID, log group, more)

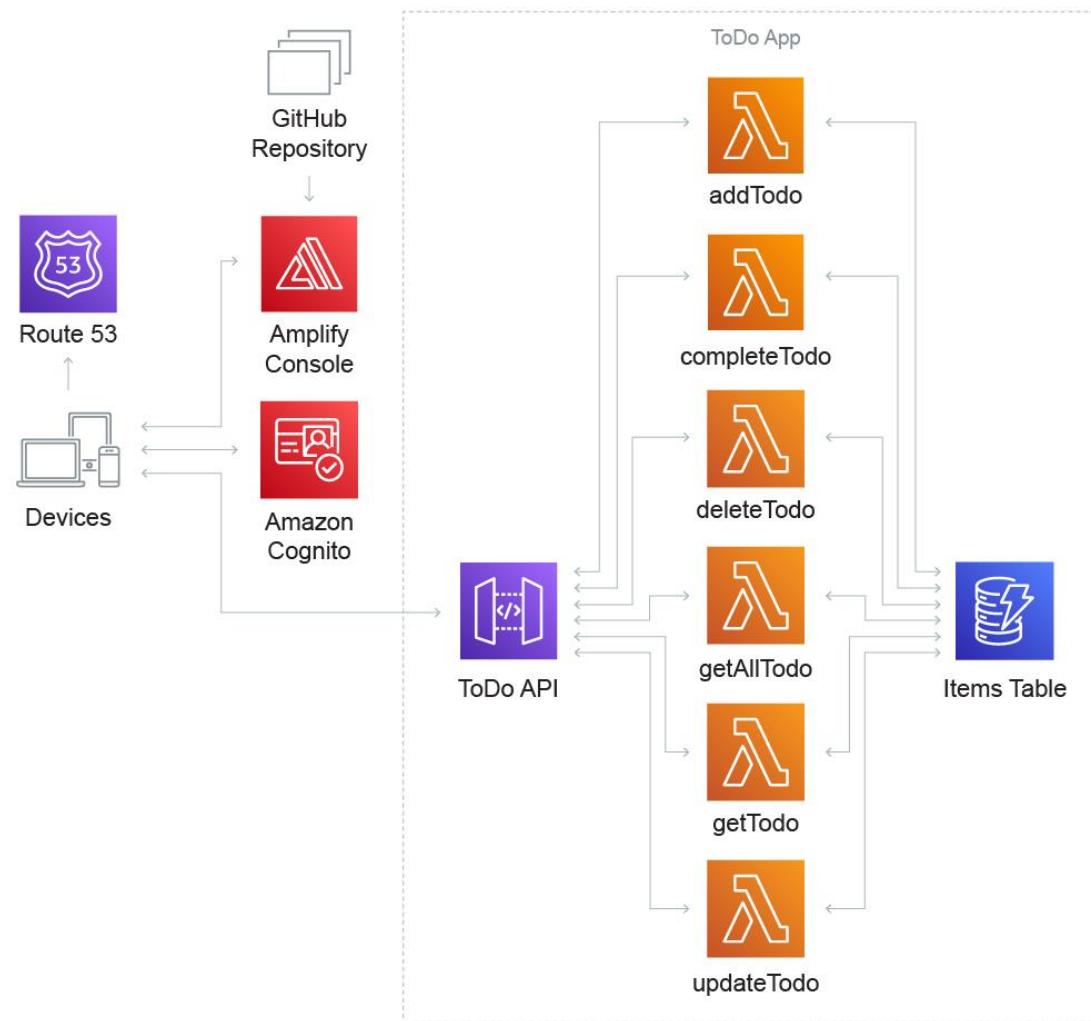
```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello world!')
    }
```

— AWS Lambda 사용 예



— AWS Lambda 사용 예



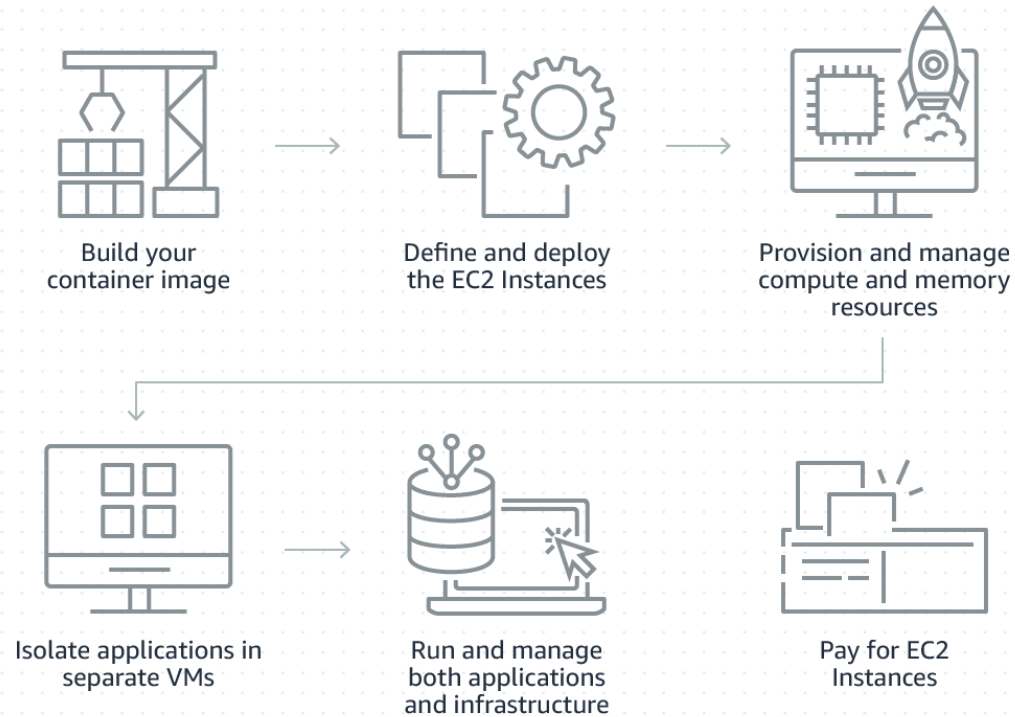
AWS Fargate

— AWS Fargate

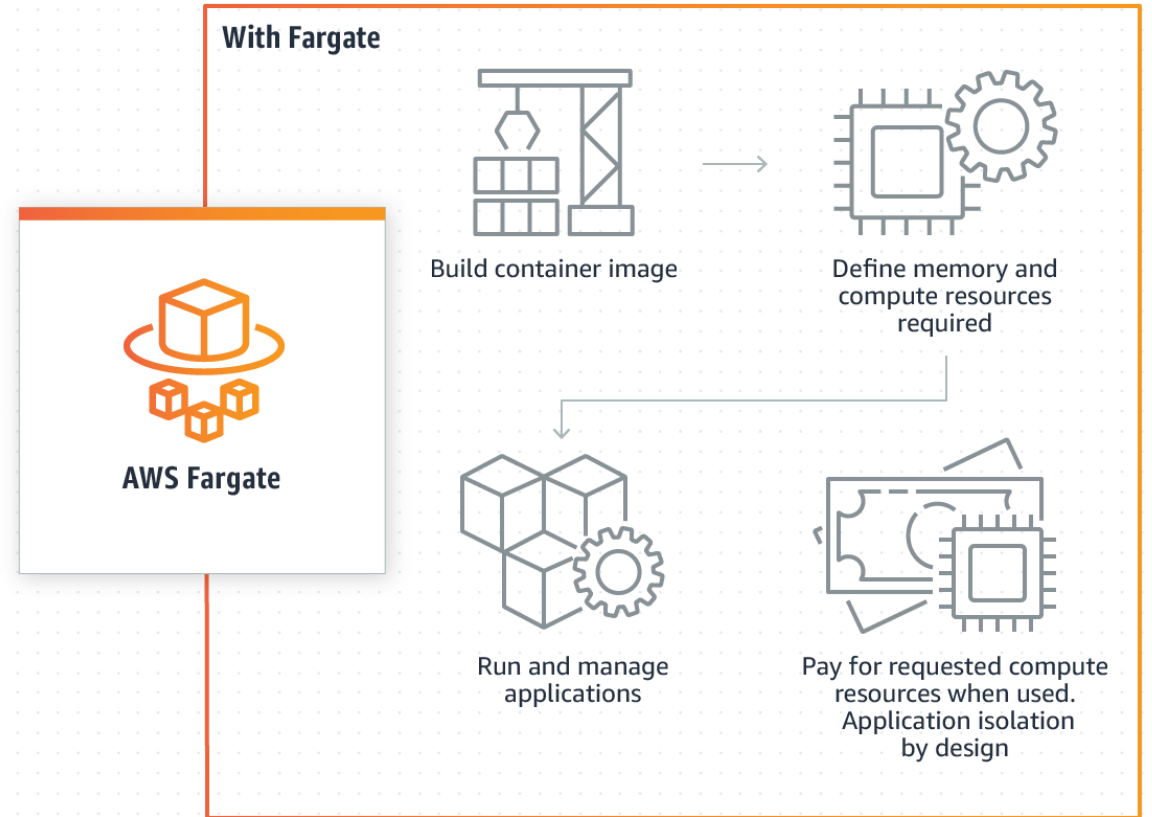
- AWS Fargate는 Amazon EC2 인스턴스의 서버나 클러스터를 관리할 필요 없이 컨테이너를 실행하기 위해 Amazon ECS에 사용할 수 있는 기술
- Fargate를 사용하면 더 이상 컨테이너를 실행하기 위해 가상 머신의 클러스터를 프로비저닝, 구성 또는 조정할 필요가 없다. 따라서 서버 유형을 선택하거나, 클러스터를 조정할 시점을 결정하거나, 클러스터 패킹을 최적화할 필요가 없다
- AWS Fargate는 서버를 관리하지 않고도 애플리케이션 구축에 초점을 맞출 수 있도록 지원하는 종량제 서버리스 컴퓨팅 엔진
- AWS Fargate는 ECS 및 EKS 모두와 호환된다.

— AWS Fargate

Without Fargate



With Fargate



— AWS Fargate 사용 사례

- 웹 앱, API 및 마이크로서비스
- 컨테이너 워크로드 실행 및 크기 조정
- AI 및 기계 학습 훈련 애플리케이션 지원
- 비용 최적화