Hanbit RealTime 137

서버리스 아키텍처를 향한 첫 발걸음

처음 시작하는 AWS 람다

매튜 풀러 지음 문경식, 신원석, 오성근, 조영준 옮김

한빛미디어 Hanbit Media, Inc.

Hanbit RealTime 137

서버리스 아키텍처를 향한 첫 발걸음

처음 시작하는 AWS 람다

매튜 풀러 지음 **문경식, 신원석, 오성근, 조영준** 옮김

이 도서는
AWS Lambda:
A Guide to Serverless Microservices
(Amazon Digital Services LLC)의
번역서입니다

한빛미디어 Hanbit Media, Inc.

표지 사진 **김미진**

이 책의 표지는 김미진 님이 보내주신 풍경사진을 담았습니다.
리얼타임은 독자의 시선을 담은 풍경사진을 책 표지로 보여주고자 합니다.

사진 보내기 **ebookwriter@hanbit.co.kr**

처음 시작하는 AWS 람다 서버리스 아키텍처를 향한 첫 발걸음

전자책 발행 2016년 9월 1일

지은이 매튜 풀러 / **옮긴이** 문경식, 신원석, 오성근, 조영준 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 9월 30일 제10-1779호
ISBN 978-89-6848-829-0 95000 / **정가** 14,000원

총괄 전태호 / **책임편집** 김창수 / **기획·편집** 이상복
디자인 표지/내지 강은영, 조판 이경숙
마케팅 박상용, 송경석, 변지영 / **영업** 김형진, 김진불, 조유미 / **제작** 박성우, 김정우

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

지은이_ **매튜 풀러** Matthew Fuller

로체스터 공과대학교에서 응용 네트워킹 및 시스템 관리를 전공했다. 키바트Kibart, AAI, 모질라 등에서 인턴으로 웹 개발, 정보 보안 및 분석 업무를 수행했고, 에이비어리Aviary를 거쳐 어도비에서 데브옵스 및 보안 엔지니어로 일하고 있다. 웹 보안 및 AWS에 대한 관심이 높아 『Chromecast Web Development』 등의 전자책을 썼다. 트위터 @matthewdfuller

옮긴이_ **문경식, 신원석, 오성근, 조영준**

옮긴이들(가나다순)은 SK텔레콤 수도권 네트워크 본부에서 길게는 10년 이상 네트워크 시스템 관리/운용 및 관련 업무를 담당했다. 2G/3G/LTE 통신 인프라 및 다양한 연동 시스템을 운용해왔으며, 현재는 보라매NOC팀에서 가상화 기반 통신 인프라 구축/운용 및 다양한 고객 서비스를 제공하는 업무를 수행 중이다.

차세대 통신 시스템이 클라우드 솔루션을 활용함에 따라 아마존 웹 서비스, VMware, 오픈스택 등에 관심이 많으며, AWS 람다 또한 같은 이유에서 통신 및 사물인터넷 등에서 크게 도움이 될 수 있을 것이라 기대하고 있다.

· 문경식 kyungsik.moon@sk.com

· 신원석 shinws@sk.com

· 오성근 s.k.oh@sk.com

· 조영준 choga88@sk.com

옮긴이의 말

AWS 람다는 IT 인프라 운용자, 개발자 모두에게 놀라운 서비스이다. 서버와 운영체제, 미들웨어 없이도 코드를 실행할 수 있는 서비스이기 때문이다. 역자들은 현재 통신망 인프라를 운용하는 업무를 수행하고 있는데, 아마존의 람다는 이런 인프라가 없이도 즉각적으로 서비스 코드가 원하는 이벤트와 환경에서 간단히 실행될 수 있게끔 지원하여 IT뿐만 아니라 통신 환경에도 새로운 아이디어를 제시한다.

이 책의 원서는 람다를 처음 접해보는 사람들도 쉽게 시작할 수 있게 도와주는 입문서다. 책을 끝까지 읽게 되면 람다를 이용해 코드를 짜보고 싶다는 생각이 들 것이다. 열심히 부딪히면서 많이 사용해보고 같은 느낌을 공유했으면 한다.

책을 읽다 보면 오탈자나 어색한 우리말 표현이 있을 수도 있다. 독자의 마음으로 일일이 실습하고 검토하면서 이해하기 쉽게 쓰려 노력했지만 밤늦은 시간, 주말 시간을 활용하여 번역을 진행하다 보니 놓친 부분이 있을 수도 있을 것이다. 불편한 점이 있다면 언제든지 출판사 또는 역자에게 요청해주시면 바로 반영토록 노력하겠다.

이 책은 한빛미디어 이상복 편집자님의 적극적인 지원 덕분에 독자들이 읽기 편한 책으로 재탄생하게 되었다. 그 밖에도 이 책의 탄생에 직간접적으로 노력해주신 많은 분들께 감사드린다.

마지막으로 국내의 열악한 IT 환경에서도 고군분투하고 있을 많은 독자 분들에게 이 책이 신선한 아이디어와 좋은 서비스로 다가가기를 바란다.

차례

chapter 1

서론

소프트웨어 개발자, 개발 및 운용 엔지니어, 시스템 관리자, 그리고 그 밖의 IT 업종의 종사자는 소프트웨어를 어디에서 어떻게 실행해야 할지를 사전에 충분히 고려해야 한다는 것을 잘 알고 있다. 전통적으로, 기업은 서버가 필요하고 물리적 서버들은 데이터센터에 줄지어 있고, 이를 관리하는 관리자 또는 엔지니어 팀이 필요하다. 대형 업체라면 하드웨어, 보안, 네트워크, 업그레이드, 그리고 서버에 필요한 모든 측면을 다루는 인력을 쉽게 고용할 수 있을 것이다.

이 책의 독자라면, 이러한 산업 트렌드가 나아가는 방향인 클라우드에 익숙할 것이다. 최근 몇 년간 다소 빠르게 기존 사업의 많은 네트워크 트래픽이 클라우드 공급업체로 옮겨갔으며, 한편으로 새로운 사업은 클라우드 기반의 인프라 위에 구축되었다. 구글Google, 디지털오션DigitalOcean, 랙스페이스Rackspace, 아마존Amazon 같은 클라우드 공급업체는 하드웨어 관리 같은 전통적인 업무 부하를 줄일 수 있는 방식을 제공해 많은 인기를 누렸다. 일부 대형 기업은 망설이고 있지만, 대부분의 산업 전문가는 클라우드 기반의 환경(또는 적어도 하이브리드 환경)이 미래의 방식이라는 데 동의한다.

특정 제조사에 종속되는 장단점은 논외로 하고, 이러한 클라우드 호스팅 공급업체들은 기존 문제를 해결하는 클라우드 기반 솔루션을 제공하기 위해 지속적으로 노력했다. 이들은 스토리지 솔루션, 네트워크 용량, 직접적인 데이터 센터 연결, 컴퓨팅 성능 등을 개발하고 지속적으로 향상함으로써, 이런 공용public 클라우드 환경에 의존하는 수많은 기업들의 요구에 부응하였다. 해가 갈수록 이들은 더 많은 컴

퓨팅 서비스를 제공하고 있다.

이런 클라우드 공급업체 중에 가장 큰 기업은 아마존이다. 아마존 웹 서비스Amazon Web Services(AWS)는 다른 경쟁 업체들을 모두 합친 규모보다도 더 크고, 놀라운 속도로 성장하고 있다. AWS의 45개 이상의 제품들은 스토리지, 네트워킹, 컴퓨팅, 기계학습, 모바일 디바이스 테스트에 관한 솔루션을 제공한다. AWS 제품 중 아주 오래된 제품 중 하나가 일래스틱 컴퓨트 클라우드Elastic Compute Cloud, 즉 EC2이다. 이 제품은 개발자가 요구할 때 가상 서버 인스턴스를 시작할 수 있게 해준다. 메모리 크기, 디스크 공간, CPU 크기 같은 매개변수를 정한 후에, 관리자는 하나의 가상 인스턴스 또는 한 번에 수천 개의 인스턴스를 생성할 수 있다. 관리자와 개발자는 코드 배포, 애플리케이션 동작, 인스턴스의 정상 상태 유지에 대한 책임이 있다(메모리나 디스크 공간 등을 과도하게 사용하지 않도록).

EC2는 예나 지금이나 AWS에서 아주 인기 있는 핵심 제품 중 하나이다. 리눅스 루트나 윈도우 관리자 권한을 통해 개발자는 상상할 수 있는 모든 종류의 코드를 자유롭게 실행할 수 있다. 수백만 개의 애플리케이션이 EC2 위에서 개발되며, 히로쿠Heroku[01]처럼 전체 사업의 기반 인프라로 EC2를 사용하는 기업도 있다. 사실 다른 아무것도 사용하지 않고 EC2만으로도 대부분 조직은 스토리지, 데이터베이스, 라우팅 등 필요한 모든 업무를 처리할 수 있다. AWS EC2의 가용성과 성능은 타의 추종을 불허한다.

01 역자주: 유명한 PaaS 서비스로, 클라우드에서 애플리케이션을 생성하는 데 사용된다.

chapter 2

다양한 종류의 업무

EC2는 웹사이트를 운영하거나, 데이터베이스를 관리하거나, 트래픽 밸런싱을 원하는 기업들 사이에서 매우 빠르게 인기를 얻었다. AWS 사업이 성장함에 따라서, EC2 플랫폼으로 운영하는 웹사이트 숫자 또한 증가하게 되었다. 넷플릭스Netflix와 인스타그램Instagram 같은 거대한 기업도 사이트와 작업의 대부분을 AWS 기반의 서비스로 운용하고 있다. EC2 호스트는 수십만 개의 도메인들도 처리할 수 있다.

2012년 말 또는 2013년 초 즈음에, 아마존의 개발자와 관리자는 EC2 대부분이 웹과 관련된 것에 사용되고 있지만 반면 완전히 다른 범주인 이벤트 기반event-driven 처리에도 사용되고 있음을 깨달았다. 많은 기업은 인프라의 다른 부분에서 이벤트에 반응하는 EC2에 의존했다. 예를 들어 웹 서비스에서 사용자에 의해 저장된 이미지는 EC2 인스턴스에 자동으로 섬네일thumbnail로 구성된다. 전달받은 새 로그 파일에 대해 EC2 인스턴스가 데이터를 처리하고, 데이터를 의미 있게 만든다. 데이터베이스는 변경된 사항들을 처리하고 다른 곳에 업데이트를 적용하기 위해 구동되는 EC2 인스턴스를 업데이트한다. 본질적으로, 가장 많이 사용하는 추가적인 AWS 서비스인 EC2는 많이 변경되는 부분을 포함하는 이벤트 체인으로 사용된다.

이런 요구사항에 부응하여, AWS는 라스베이거스에서 열린 2014 Re:Invent 콘퍼런스 기조연설에서 새 제품 AWS 람다AWS Lambda를 발표했다. 아마존에 따르면 AWS 람다는 '새로운 정보에 빨리 대응하는 애플리케이션을 손쉽게 만들 수 있도

록 이벤트에 반응하는 코드들을 실행하고 자동으로 컴퓨트 리소스를 관리해주는 컴퓨트 서비스'이다.[01]

기존에 EC2에서 이벤트를 처리하고, 이미지를 조정하고, 데이터베이스 업데이트에 반응하고, 사용자의 클릭이나 웹사이트의 업데이트에 대한 응답을 처리하던 많은 조직에게 람다의 발표는 엄청난 사건이었다. EC2 모델에서는, 개발자 및 운용 스태프가 반드시 코드를 디자인하고, 리소스를 배포하고, 이벤트 훅을 설정하고, 리소스들을 관리하고, 모니터링을 구현하고, 서비스 중단 시간에 대응하고, 규모 변경을 계획하고, 적절한 규모 정책으로 급변하는 트래픽에 반응해야 했다. 람다를 이용하면 개발자는 단지 코드를 디자인하고, 몇 가지 옵션을 설정하고, 애플리케이션을 공개하기만 하면 된다.

이런 장점들에도 불구하고, 람다는 이벤트에 기반하지 않은 업무에 대해서는 아직 설계된 것이 없다. 예를 들어 람다 기능은 (2015년 7월 발표된 API 게이트웨이와 결합되지 않는 한) 직접 HTTP 요청에 반응할 수 없고, 백그라운드에서 장시간 동작하는 프로세스도 실행할 수 없다. 람다는 작은 타임아웃(현재 300초)과 적당한 메모리 할당량(이 책을 쓰는 시점에서 1.5GB까지)만을 지원하며, EC2를 완전히 대체하도록 설계되지는 않았다. 하지만 람다는 특정 서비스를 위한 일부 기능을 대체할 수 있고, 클라우드 컴퓨팅의 완전히 새로운 형태를 지원할 수 있다.

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

chapter 3

이 책에 대하여

이 책은 AWS 람다 서비스를 아직 사용해보지 않은 AWS 사용자의 눈높이에 맞출 것이다. 독자가 AWS 용어와 개념에 친숙하고, 이전에 AWS에서 프로젝트를 시작한 경험이 있다면 쉽게 책을 읽을 수 있을 것이다. 공인된 AWS 전문가일 필요는 없지만, 직접 몇 번 해본 경험이 AWS 람다 서비스에 다가가는 데 도움이 될 것이다. 이를 염두에 두고, 독자가 람다를 이전에 사용해본 적이 없거나 또는 매우 적은 지식만 가지고 있다고 가정하고 책을 시작할 것이다. 초보자 중심의 방식으로 기능과 용어에 대해 최선을 다해 설명할 것이다.

먼저 이 책이 다루지 않는 것에 대해서 얘기해보겠다. 이 책은 공식적인 가이드는 아니다. 책의 모든 내용은 필자가 AWS 람다 출시 후 지난 몇 달간 수많은 프로젝트를 해보며 얻은 지식과 경험을 기반으로 한다. 수많은 사용자와 전문가에게 이 책에 대해 조언을 받아 정확성을 확보하긴 했지만, 이 책이 AWS가 공개한 공식 문서를 대체할 수 있는 것은 아니다. 대신 AWS 공식 문서의 기술적 세부사항들을 이해하기 쉬운 방식으로 바꾸고, 독자의 프로젝트에 적용할 수 있는 가이드 역할을 할 것이다.

이 책은 가장 최신 버전과 모든 기능을 다루기 위해 노력했다. 하지만 람다는 완전히 새로운 서비스이고 언제든지 내용이 바뀔 수 있다. 따라서 책에 쓴 모든 것에 대해 정확성을 보장할 수는 없지만, 대신 모든 코드와 스크린샷은 실제 확인해본 것들이고 책이 출판될 때까지는 유효했다. 필자는 개인 프로젝트 및 업무 프로젝트에는 물론, 이 책에서 쓴 예제를 개발하기 위해서도 람다 관련 업무를 밀접하게

수행하였다.

이 책의 구조는 람다의 경험을 향상시키는 가이드로서의 역할을 할 것이다. 예제 코드와 함수 예제를 이용해서, 간단한 람다 작업부터 더 복잡한 통합 작업까지 람다를 사용해서 해볼 것이다. 연습하는 동안, 다른 AWS 서비스도 추가로 사용할 것이다(구체적으로는 IAM과 S3). 집에서 실습하는 독자라면, 대부분의 AWS 서비스들이 1년 동안만 무료로 제공된다는 것을 잊지 말아야 한다. 만약 1년을 넘어가거나 이미 AWS을 사용해왔다면 비용을 내야 하고, 더 이상 프리 티어free tier, 즉 무료 단계를 이용할 수 없다.

이제 전반적인 기술 이야기는 마쳤으니, 내부적으로 어떻게 람다가 동작하는지 소개하고 더불어 레이아웃에 익숙해지기 위해 람다 콘솔을 살펴보겠다.

chapter 4

람다 배경지식

컨셉 측면에서 보면, 람다는 일종의 '실행하고 잊어버리는run and forget' 모델을 염두에 두고 설계되었다. 다시 말해 개발자가 코드를 제공할 때, 그때그때 실행할지 아니면 어떤 이벤트에 반응할지를 명시해놓기만 하면, 나머지는 AWS가 알아서 하겠다는 의미이다. 즉, AWS는 컴퓨팅 파워를 제공하고, 코드의 저장과 변경을 처리하며, 코드를 요청받은 위치에 배포하고, 적절한 요구에 맞춰 확장하고, 기본적인 자원인 디스크 공간, 메모리, CPU 요구사항을 관리하며, 네트워킹, OS 업데이트 및 그 밖에 상호 간에 필요한 나머지 모든 요구사항을 처리할 것이다. 전통적인 모델과 비교하자면, 혹은 심지어 EC2로 동작 중인 작업과 비교해봐도, 이 방대한 업무가 개발자의 역할에서 이제는 AWS의 역할로 옮겨오고 있는 중이다. 하지만 이런 방식에 단점이 없는 것은 아니다. 각자의 프로젝트를 람다로 바꾸는 것이 적절한지 확실히 확인하려면 계속 책을 읽기 바란다.

4.1 내부

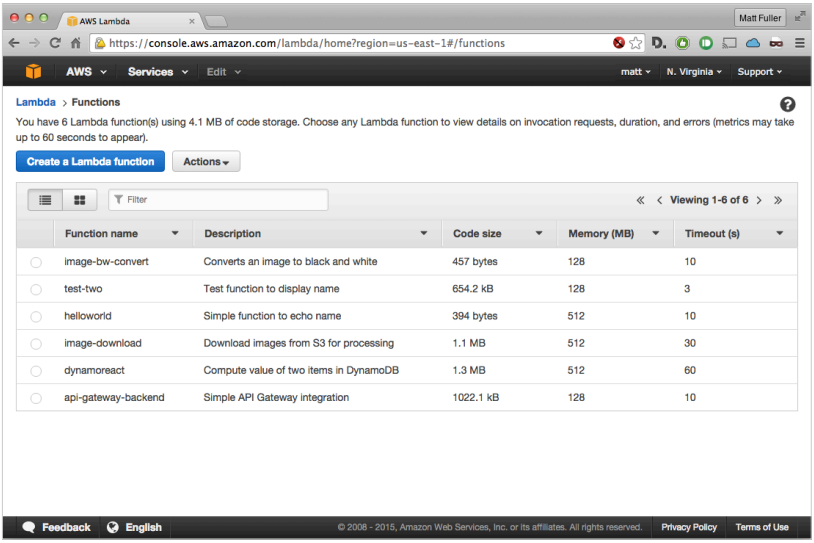
내부적으로 보면, 사실 람다는 단지 독자에게 익숙한 AWS 자원을 사용할 뿐이다. 코드는 S3에 저장되고, 함수에 대한 메타데이터는 DynamoDB에 저장되고, 실행은 Amazon Linux EC2 인스턴스에서 실행되며, 함수들은 IAM의 역할을 수행한다. AWS는 람다를 위해 기존 서비스에 새로운 기능을 많이 추가하기도 했겠지만, 기존에 있는 것들을 새롭게 만드느라 불필요한 시간을 낭비하지는 않았다.

람다는 내부적으로는 잘 알려진 서비스들을 활용하지만, 기존과는 다른 사고방식과 개발 패턴을 필요로 한다. 다음 장에서 기존 컴퓨팅에 비해 람다에서의 개발이 어떻게 다른지 살펴볼 것이다. 일단, 람다의 기본 함수부터 확인해보자.

4.2 기본 지식

4.2.1 함수

람다 프로젝트는 함수로서 각 서비스에 배치된다. 보통 앞으로는 람다 프로젝트의 전부를 람다 함수라고 말할 것이다. 각각의 함수는 한 개 또는 몇 개의 일반적인 일 처리를 수행할 수 있게 설계된다. 예를 들어 하나의 람다 함수는 업로드된 이미지를 다운로드하고 그 이미지를 흑백으로 변환할 수 있다. 전통적인 프로젝트와는 달리, 람다 함수는 보통 매우 작다. 필자는 대부분 몇백 줄 이내의 코드로 작성하곤 했다(Node.js 의존성dependency 부분 제외).



각각의 람다 함수는 분리된 프로젝트라고 볼 수 있다. 개인적으로는 함수 하나마다 깃허브 프로젝트를 생성해서 사용한다. 하지만 사용할 수 있는 저장소 개수가 정해져 있다면, 한 개의 저장소에 여러 개의 하위 폴더 방식이 더 좋을 것이다. 이런 조직적인 구조는 독자가 결정하기 나름이다. 다만 각각의 람다 함수는 독립적인 함수로 사용해야 한다. 즉, 배포 전에 다른 프로젝트에 복사하지 않는 한 다른 프로젝트에 의존할 수 없다.

람다 함수는 람다에 ZIP 파일 형태로 업로드된다. ZIP 파일은 콘솔, 명령 줄에서 업로드할 수도 있고 대용량의 파일이라면 S3에 저장할 수도 있다. AWS는 관리하는 백엔드 자원으로 다운로드한 ZIP 파일을 처리한다. 각각의 연속적인 코드 업데이트를 하기 위해서 이전과 바뀌는 부분의 ZIP 파일만 재 업로드하면 된다. 또한 AWS는 새로운 코드를 자원에 배포하는 것도 관리한다.

다음 장에서는 ZIP 파일의 내용물들에 대해 다뤄볼 것이다. 하지만 지금 당장은 그 내용물들이 완전히 코드 기반으로 이루어졌다는 것만 알면 된다. Node.js를 사용해본 적이 있다면, NPM의 의존성이 모두 포함된 완전한 프로젝트를 생각해보라.

4.2.2 언어

꽤 오랫동안 람다는 Node.js, 자바, 파이썬으로 쓴 코드를 지원해왔다. 람다 팀이 전하는 바에 따르면 추가적인 언어를 지원할 예정이라고 하지만, 구체적인 시기에 대한 계획은 밝혀지지 않았다. 이 책에서는 코드 예제의 대부분을 Node.js로 사용할 것이다.

4.2.3 리소스 할당

ZIP으로 람다 함수를 업로드하면, 이 코드가 실행되면서 할당받을 컴퓨팅 파워가 얼마나 필요할지 결정해야 한다. 물론 이 할당량은 성능과 비용에 영향을 끼칠

것이다. 자원을 적절하게 할당하기 위해서는 람다 요금 페이지[01]를 참고하길 바란다.

각자의 함수 실행에 할당받은 메모리는 함수를 위해 소비하고 싶은 만큼 허용한 메모리 크기이다. 이 의미는 모든 함수가 실행할 때 이용 가능한 만큼의 메모리를 활용한다는 의미가 아니라, 이 함수가 허용한 양까지 사용이 가능하다는 의미이다. 메모리는 128MB 단위로 할당하며 (현재는) 최대 1536MB까지 할당할 수 있다.

예를 들어 누군가 `imageProcessor`라는 함수를 개발했다고 해보자. 이 함수는 S3로부터 이미지를 다운받고, 이 이미지의 메타데이터에 대한 분석을 수행한다. 이미지 사이즈가 매우 다양할 수 있기 때문에, 이 함수의 성능에 대한 다양한 메모리 할당 테스트가 필요할 것이다. 일단 메모리 할당량을 256MB로 설정했다고 하자. 함수를 5번 실행했을 때, 이 함수가 30MB, 100MB, 70MB, 200MB, 40MB만큼 메모리를 소비했다고 해보자. 각 실행마다 256MB에 해당하는 요금이 부과될 것이다(소비된 양이 아니다. 따라서 리소스를 과도하게 할당하지 않는 것이 중요하다). 만약 함수를 6번째 실행했을 때, 280MB만큼의 메모리가 필요했다면 6번째는 실패할 것이고, 메모리가 부족하다는 에러 메시지가 나올 것이다. 이때는 256MB 실행에 대한 요금이 부과될 것이다.

메모리 할당이 각 함수가 사용할 수 있는 구체적인 메모리 크기를 한정하는 것처럼, 타임아웃 값을 정해 각 함수에 대한 실행 시간을 한정할 수 있다. 타임아웃 값은 1초부터 300초까지(집필 시점 기준) 다양하다. 높은 타임아웃 값을 설정하면 모든 함수의 실행이 최대 속도로 실행됨을 의미하는 것이 아니라, 함수들이 그 특정 시간 안에까지는 실행될 수 있다는 것을 의미한다. 위의 예에서, 첫 번째 실행에 타임아웃을 10초로 설정했는데 실제 실행은 2초가 걸렸다면, 10초에 대한 것

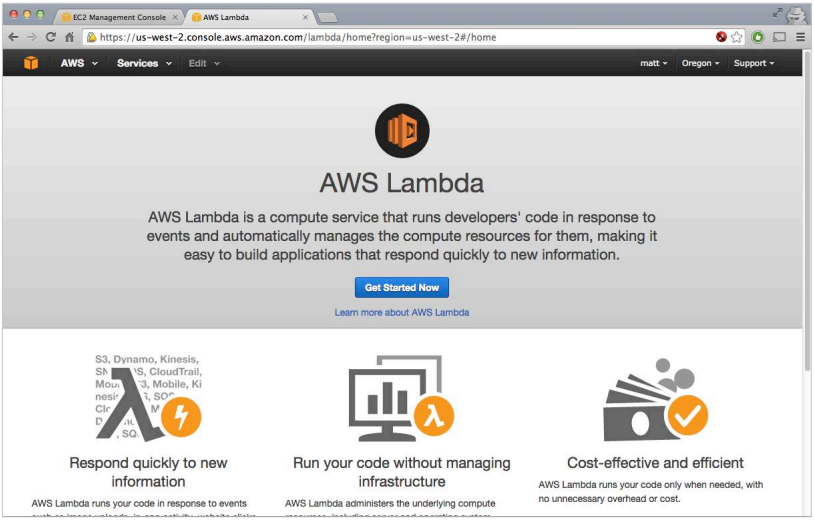
01 https://aws.amazon.com/lambda/pricing

이 아니라 2초에 대한 요금만 부과된다. 이 값은 AWS에 의해 100 밀리세컨드 단위에서 반올림된다.

4.3 설정하기

다음 장에서는 단순히 'Hello World'를 출력하는 가장 간단한 람다 함수를 만들어볼 것이다. 계속 진행하기 전에, 람다 콘솔에 대해 이해하는 것이 도움이 될 것이다.

콘솔을 시작하려면, AWS 콘솔 페이지에서 'Lambda'를 클릭한다. 만약 처음 서비스를 사용하는 경우라면, 새 함수를 만들라는 시작 페이지가 표시될 것이다.[02]



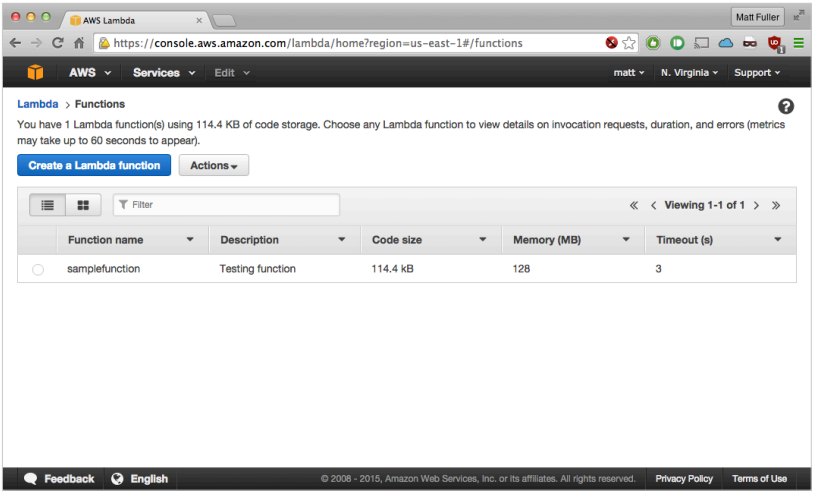
여기서 [Get Started Now] 버튼을 선택하면 몇몇 블루프린트blueprint[03]를 보여줄 것이다. AWS는 시작하는 데 도움이 될 샘플 람다 함수들을 제공한다. 다음 장에서 우리는 'Hello World' 함수를 만들어볼 텐데, 이때 블루프린트를 이용할 수도

02 역자주: 람다는 2016년 8월 기준으로 아직 서울 리전을 지원하지는 않는다.

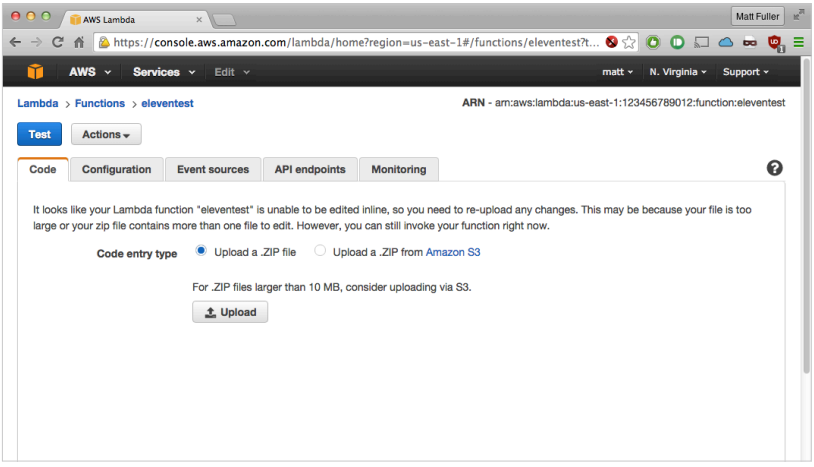
03 역자주: 람다 함수와 이벤트에 대한 샘플 설정을 의미한다.

있지만 이용하지 않고 함수를 만들어볼 것이다.

우선 여기서는 함수를 이미 만들었다고 가정하고 함수(Functions) 메인 페이지 화면이 어떤 식으로 되어 있는지 살펴보자. 람다 콘솔은 함수 이름, 설명, 기본 코드 크기, 메모리 및 타임아웃을 보여준다.



추가 정보를 보기 위해서는 함수명을 클릭한다.



세부 함수 정보 페이지에서 새로운 코드를 업로드할 수 있고, 함수의 엔트리포인트와 이벤트 소스 등에 대해 설정할 수 있다. 과거 며칠간의 성능 그래프도 볼 수 있다.

이제 'Hello World' 람다 함수를 만들어보자.

chapter 5

Hello World

대부분의 다른 프로그래밍 교재와 마찬가지로, 간단한 람다 함수부터 작성해보겠다. "Hello World"를 출력하는 함수다. 람다 콘솔에서 직접 코드를 편집할 수도 있지만, 그렇게는 아주 간단한 함수밖에 작성할 수 없다. 두 개 이상의 파일로 이루어졌거나, 의존성이 있다면 파일 압축 및 업로드를 할 필요가 있다. 사실 대부분의 함수가 이에 해당하므로 이번에 만들 'Hello World' 함수도 이 방식을 사용할 것이다.

각자 사용하는 편집기로 index.js라는 이름으로 새 파일을 만든다. 이 파일에 아래 내용을 추가한다.

```
exports.handler = function(event, context) {
  context.succeed("Hello World");
};
```

끝이다! 가장 간단한 람다 함수를 만들어보았다. 이 코드에서 각각의 부분에 대해 이야기해보자.

| exports.handler |

함수가 실행될 때 람다에 의해 호출되어 내보내는[export] 메서드이다. 여러 개 메서드를 내보낼 수 있지만, 콘솔에서 람다 함수를 만들 때에는 한 개만 선택해야 한다. 내보내는 메서드 핸들러[handler]를 반드시 사용해야 하는 것은 아니지만, 기본적으로 람다에는 핸들러가 사용되므로 사용하는 것을 추천한다.

| (event, context) |

내보내는 메서드는 두 가지 인수argument를 받는다. 이벤트(event)는 실행하는 이벤트에 대한 자세한 정보를 담는 객체이다. 예를 들어, S3가 람다 함수를 트리거할 때, 이벤트는 버킷bucket과 키key에 대한 정보를 포함한다. API 게이트웨이가 람다 함수를 트리거한다면, 이벤트는 HTTP 메서드, 사용자 입력, 기타 웹 세션 정보에 대한 자세한 정보를 포함한다. 콘텍스트(context) 객체는 함수 그 자체의 내부 정보를 포함할 뿐만 아니라, 함수의 성공/실패 종료 메서드도 포함한다. 앞으로 이런 객체들에 대해 자세히 살펴볼 것이다.

| context.succeed |

콘텍스트 객체의 성공(succeed) 메서드를 통해 개발자는 람다에게 함수가 성공적으로 실행되었음을 알려준다. succeed의 인수(이 경우 "Hello World")는 호출자에게 다시 반환된다.

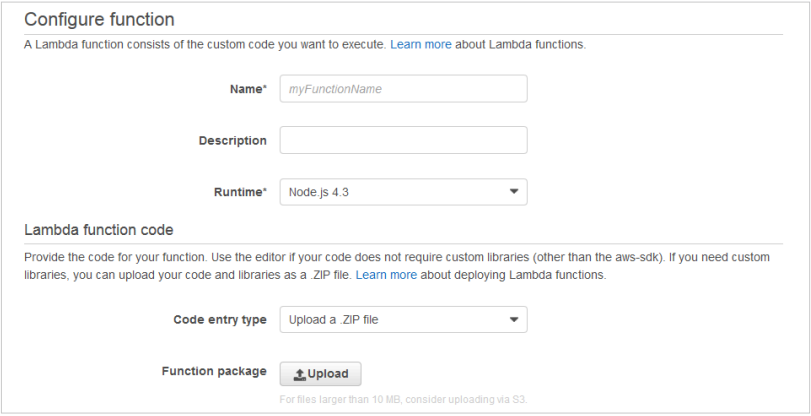
5.1 함수 업로드

함수가 한 개의 파일로 구성되더라도, 묶어서 ZIP 파일로 생성해야 한다. ZIP 파일의 이름은 중요하지 않다. 앞에서 만든 index.js 파일을 ZIP 파일로 압축하자.

이제 람다 콘솔에 로그인하고 [Create a Lambda function] 버튼을 클릭한다. AWS는 여러 람다 블루프린트를 제공하지만, 우리는 이미 ZIP을 갖고 있기 때문에, 블루프린트를 고르지 않고 바로 [Next] 버튼을 클릭한다. 트리거도 고르지 않고, 다음으로 넘어간다.

그다음 페이지에서는 만든 함수에 대한 적당한 이름과 설명을 넣은 후, 런타임runtime은 'Node.js'를 선택한다. 코드를 직접 입력하거나, ZIP으로 업로드하거나, S3에 호스트한 ZIP의 S3 링크 URL 경로를 입력할 수도 있다. ZIP 업로드 옵션(Upload a .ZIP file)을 선택하고 [Upload] 버튼을 눌러 앞에서 만든 ZIP 파일

을 찾아서 업로드한다.

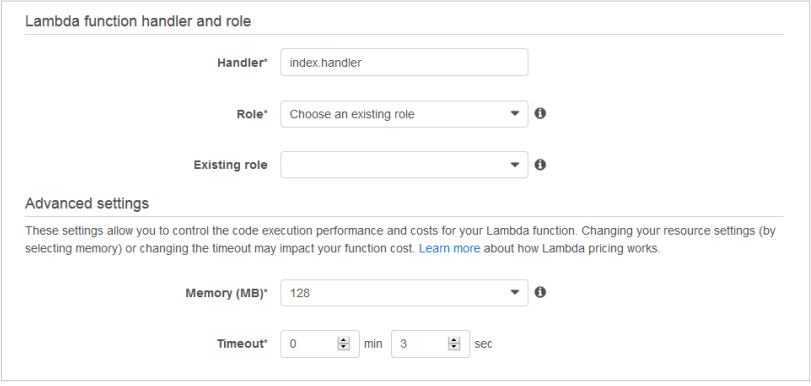


그 아래 'Lambda function handler and role' 항목에서는 이벤트를 처리하는 데 사용할 파일과 내보낼 메서드를 정해야 한다. 지금까지 잘 따라 했다면 기본값인 `index.handler` 그대로 둬도 상관없다. 만약 핸들러를 다르게 작성했다면, 앞 절에서 설명한 가이드라인을 준수했는지 확인해야 한다. 즉 `event`와 `context`를 인수로 받으며, `context.succeed`로 끝나야 한다.

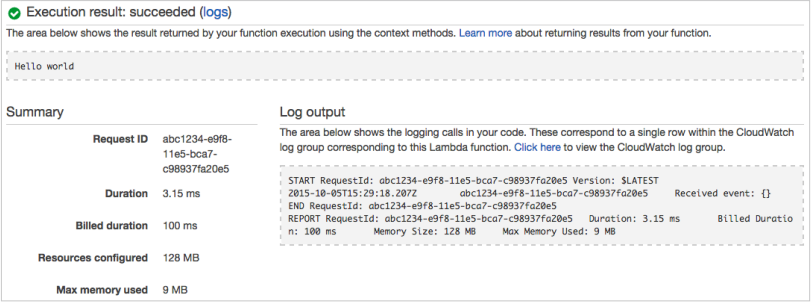
다음으로, 함수를 실행하기 위한 역할을 선택해야 한다. 곧 역할에 대한 설명을 하겠지만, 일단은 EC2 IAM의 역할처럼 행동한다고 생각하면 된다. 즉 계정 내에서 접근 가능한 자원에 람다 함수를 허용해주는 것이라 이해하면 되겠다. 생성한 함수는 최소한 클라우드워치CloudWatch에 이벤트를 기록할 수 있는 권한이 필요하다. AWS는 드롭다운 리스트에서 미리 정의된 역할을 선택할 수 있게 역할의 생성을 단순화했다. 'Hello World' 함수의 경우, 기존에 있는 역할(Existing role)에서 첫 번째 기본값을 선택하면 된다. 새로운 역할을 생성하려고 하면 IAM 내에서 역할을 생성하라는 창이 뜰 것이다.

'Advanced settings' 항목에서는 만든 함수에 할당될 메모리와 타임아웃을 정할 수 있다. `"Hello World"`를 출력하는 것은 아주 작은 메모리와 시간으로도 가

능하므로 기본값으로 둬도 된다.



다음 페이지에서는 생성한 함수를 검토할 수 있다. [Create function] 버튼을 눌러 생성을 완료하면, 테스트할 수 있는 화면으로 넘어간다. 여기서 처음 [Test] 버튼을 누르면 테스트용 이벤트를 설정하라는 화면이 뜰 것이다. [Actions] 버튼을 누르고 'Configure test event'를 선택해도 설정할 수 있다. 'Hello World' 함수는 어떤 이벤트 속성도 필요하지 않기 때문에, 어떤 이벤트 템플릿을 선택해도 상관없고, {}를 입력해 빈 이벤트 객체를 전달해도 된다. [Save and test] 버튼을 클릭하면, 함수의 실행 결과인 `"Hello World"`를 볼 수 있을 것이다. 그리고 로그, 실행 시간, 메모리 사용량 등에 대한 다양한 추가 정보도 볼 수 있을 것이다.



이 결과를 보면 클라우드워치 내에 로그 그룹log group과 스트림stream이 생성되었다는 사실도 알 수 있다. 람다를 실행할 때마다 각 이벤트 및 애플리케이션 내의 모든 `console.log`의 결과가 클라우드워치 로그 그룹에 기록된다.

한빛 리얼타임

한빛 리얼타임은 IT 개발자를 위한 전자책입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾기도 쉽지 않습니다. 또한, 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1 eBook First – 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다

500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 좀 더 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한, 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2 무료로 업데이트되는 전자책 전용 서비스입니다

종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3 독자의 편의를 위해 DRM-Free로 제공합니다

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 어려운 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가겠습니다.