그림으로 배우는
클라우드 인프라와 API의 구조

히라야마 쯔요시 외 지음
신상재 옮김

그림으로 배우는 클라우드 인프라와 API의 구조

지은이 히라야마 쯔요시 외 **옮긴이** 신상재 **1판 1쇄 발행일** 2017년 5월 22일 **펴낸이** 임성춘 **펴낸곳** 로드북
편집 조서희 **디자인** 이호용(표지) 심용희(본문) **주소** 서울시 동작구 동작대로 11길 96-5 401호
출판 등록 제 25100-2017-000015호(2011년 3월 22일) **전화** 02)874-7883 **팩스** 02)6280-6901
정가 27,000원 **ISBN** 978-89-97924-31-8 93000

책 내용에 대한 의견이나 문의는 출판사 이메일이나 블로그로 연락해 주십시오.
잘못 만들어진 책은 서점에서 교환해 드립니다.
이메일 chief@roadbook.co.kr **블로그** www.roadbook.co.kr

클라우드를 바라보는 패러다임의 변화

수년 전까지만 하더라도 '클라우드'를 이야기할 때는 사용한 만큼 돈을 내고, 필요할 때 즉시 이용할 수 있으며, 다 쓴 후에는 바로 버릴 수 있다는 장점을 부각시켜 스타트업과 게임 회사를 중심으로 '합리적인 가격의 인프라', '즉시 쓰고 버릴 수 있는 인프라'의 관점에서 언급되었습니다. 이후, 대형 IT기업에서는 글로벌 고객에게 좀 더 빠른 서비스를 제공하기 위해 엣지 로케이션을 활용하거나 재난 대응을 위한 다중 가용 영역의 활용, 그리고 오브젝트 스토리지 적용 측면에서의 전략이 전개되었습니다. 최근에는 이러한 클라우드의 도입보다는 실제 활용 측면에서의 다양한 접근 방법이 주목을 받고 있는데 특히, DevOps를 고려한 효율적인 인프라 운영이나 신속한 서비스의 배포를 위한 클라우드의 활용 방법 등이 관심을 끌고 있습니다. 이와 같이 클라우드에 대한 관심은 이전보다 더 구체적이고 더 세분화된 목적을 만족시키기 위한 방향으로 전개되기 시작했고 이러한 동향을 반영하듯 이전에는 매니지드 서비스의 기능을 수동적으로 따라하던 단계를 넘어, 이제는 원하는 기능을 직접 골라 호출해서 쓰려는 적극적인 사용 패턴이 늘기 시작했습니다. 즉, 클라우드의 API를 직접 다룰 수 있다는 사실에 눈을 뜨게 되면서 클라우드 서비스가 기존에 제공하지 않았던 기능들을 기존의 API들을 조합하여 스스로의 목적에 맞게 사용할 수 있게 된 것입니다.

이 책은 어떤 책인가

이 책은 클라우드 도입을 고민하고 있을 때, 왜 클라우드를 도입해야 하는지 당위성을 설명하고 있는 책이 아닙니다. 또한 언제 어떤 방법으로 마이그레이션을 해야 하는지 안내하는 책도 아닙니다. 심지어 클라우드의 기능에는 어떤 것이 있고, 어떨 때 사용해야 한다고 설명하는 책도 아닙니다. 이 책은 이제 클라우드를 그럭저럭 사용하게 된 상태에서 우리는 과연 얼마나 그 내부를 잘 이해하고 있는지, 보다 세부적인 단위 기능들을 얼마나 잘 활용할 수 있는지 확인하기 위한 책이며, 수많은 클라우드 서비스들 간의 공통된 맥락이 어떤 것이 있는지 상상하게 하고 이들을 어떻게 조합하고 응용할지 고민하게 만들어주는 책입니다.

클라우드 관련 서적들을 건강 관련 서적으로 비유하자면, 이제까지의 책들은 좋은 식재료를 저렴하게 사서 요리를 해먹는 방법(적은 비용으로 시스템을 활용하는 방법)과 건강한 식습관을 통한 다이어트 방법(각 기능의 올바른 사용법과 비용 절감 방법)을 소개하여 궁극적으로는 건강하게 잘 사는 방법(서비스를 성공적으로 운영하는 방법)을 설명하였습니다. 그 결과 나는 암을 극복하고 잘 살고 있다는 성공 사례(클라우드를 도입하여 비즈니스 이슈를 해결한 사례)가 여러 경험자들 사이에서 회자되기도 합니다.

반면, 이 책은 사람의 대표적인 체질에 대해 이야기하고 성별과 인종이 다르더라도 공통된 신체 장기들의 움직임에 관한 보편적인 기능(클라우드 서비스의 공통점과 차이점)을 설명합니다. 그리고 그러한 신체 기능을 활성화하려면 어떤 혈(穴)들이 있는지(어떤 API가 있는지), 언제 어떤 혈을 어떤 방식으로 건드려주면 되는지(언제 어떤 API를 활용하면 되는지)를 설명하는 책입니다. 실제로 이 책은 클라우드의 보편적인 기능을 설명할 때, 밖으로는 가장 많이 활용되는 AWS를 예로 들고 안으로는 소스가 공개된 오픈스택을 예로 들어 설명하고 있습니다.

이런 독자에게 권합니다

만약 한 장, 한 장 넘겨보면서 따라하는 실습을 기대했던 클라우드 초심자나 기업에서 클라우드를 도입할 대의명분을 찾으려는 경영전략 부서의 독자라면 다른 책을 보기를 권합니다. 대신 이미 클라우드를 조금은 써 봤고 서로 다른 여러 클라우드 서비스들도 함께 사용해보면서 클라우드의 공통된 기능과 유사성이 보이기 시작한 클라우드 사용자라면 이 책이 각 기능 요소들에 대한 생각을 정리하는데 도움이 될 것입니다. 또한 기존의 웹 콘솔 방식의 제어 방법보다 더 세부적인 기능을 직접 다루고 싶은 클라우드 개발자라면 서비스별 API의 내부 동작 방식에 대한 설명이 사고를 넓히는데 도움이 될 것이며, 서로 다른 여러 클라우드 기능들을 통합하여 관리를 효율적으로 하고 싶은 클라우드 운영자라면 멀티 클라우드에 대한 설명에 여러 아이디어가 샘솟을 것입니다.

그 밖에도 인프라 경험은 적지만 REST API로 웹 애플리케이션을 만들어 본 서비스 개발자라면 막연했던 클라우드의 구조가 머릿속에 그려지기 시작할 것입니다. 특히 개발에서 운영까지 확장하여 DevOps에 대해 고민하는 분이라면 오케스트레이션, 이뮤터블 인프라스트럭처를 보면서 앞으로의 개발과 운영 방식이 어떠해야 하는지 인사이트를 얻을 수 있습니다.

이 책의 원서는 일본 책으로 현지에서는 '인프라 외적인 웹 기술 등에 너무 많은 것을 다루고 있다', '챕터별로 다루는 깊이가 일관되지 않다', '너무 AWS 위주로만 설명한 것 아니냐' 등의 비교적 혹독한 평가가 있는 것이 사실입니다. 다만 이런 약점이 오히려 국내 상황으로 비춰보면 인프라는 잘 모르지만 웹 개발에 경험이 있는 웹 개발자가 읽기 편하고, 국내의 AWS 사용자가 오픈스택의 내부 구조를 함께 살펴 봄으로써 특정 벤더에 종속되지 않는 클라우드 전반적인 개념을 보다 더 이해할 수 있는 효과가 있다고 생각합니다. 실제로 이 책을 번역한 역자 스스로도 웹 개발자 출신이며 주로 AWS를 사용했었는데, 번역 작업을 하면서 오픈스택에도 입문하고 막연했던 DevOps도 클라우드 환경에서 어떤 방식으로 구현할 수 있는지 이해하게 되었습니다.

이 책은 철저하게 웹 API의 관점에서 클라우드를 설명합니다. API를 통해 어떤 기능이 있는지 살펴 보고, API를 통해 내부 동작을 이해하고, API를 통해 더 많은 기능 등을 통합하고 관리하는 방법에 대해 알려줍니다. 그래서 이 책은 기존의 클라우드 책과는 다른 독특한 매력이 있습니다.

여러분이 인프라 운영자라면 웹 API를 통해 웹 개발을 간접적으로 알게 될 것이고, 거꾸로 웹 개발자라면 웹 API를 통해 인프라를 간접적으로 이해하게 되리라 생각합니다. 여러분의 역할이 어떤 것이든 이 책을 보면서 웹 API 요청에서 인프라 하부 구조의 밑바닥까지 전체를 관통하는 여정을 할 것입니다. 저는 그 과정을 도와드릴 가이드입니다. 부디 제가 번역한 이 책이 클라우드를 여행할 히치하이커들에게 좋은 안내서가 되길 바랍니다.

하늘과 바람과 구름과 API
신상재

지은이의 말

이 책은 클라우드, 그 중에서도 IaaSInfrastructure as a Service와 같은 인프라 부문에 종사하는 엔지니어를 위한 책입니다. 클라우드를 설명함에 있어서 특정 클라우드 서비스에 종속된 기능보다는 좀 더 거시적인 관점에서 클라우드 인프라를 이해할 수 있도록 기획했습니다. 공통적인 인프라 구성 요소들과 이를 제어하기 위한 API의 개념, 그리고 동작 원리를 개념 중심으로 풀어냈습니다. 또한 이 책을 읽는 독자가 클라우드 시대에서 요구하는 풀 스택 클라우드 아키텍트가 되기 위해 반드시 알아두면 좋을 지식들을 제공하려고 했습니다.

최근, 스타트업 기업을 중심으로 확산되던 클라우드 환경이 중대형 기업의 전산 시스템에도 도입되기 시작했고 이에 따라 전문 ICT 업체의 정보화 전략도 클라우드 쪽으로 무게중심이 옮아가고 있습니다. 소수의 일부 신생 기업이나 최신 기술에 발빠른 엔지니어들이 사용했던 '클라우드'가 이제는 보다 대중화된 기술로 재조명되고 있습니다. 이러한 현상은 ICT 기술이 진정한 의미에서의 사회적 인프라로 자리매김을 하고 있다는 것을 보여주기에 충분합니다.

이 책은 기획 과정에서 상당한 난항을 겪었습니다. 가장 어려운 점은 클라우드라는 것이 애당초 하부 구조를 모르더라도 비교적 쉽게 시스템을 만들 수 있도록 설계되어 있다는 점 때문이었습니다. 즉, 내부 구조는 기본적으로 블랙박스처럼 구성되어 있어서, 사용자는 내부를 의식하지 않고도 클라우드를 이용할 수 있어서 책으로 다룰 만한 마땅한 내용이 없었습니다. 그렇게 곤란해 하던 중에 문득, 클라우드 환경이 기존의 인프라와 다른 점은 API를 사용해서 제어할 수 있다는 점을 깨달았고 이런 특징에 착안하여 책의 전개 방향을 API를 사용한 인프라의 관리로 잡게 되었습니다. 그래서 이 책을 다 읽고 나면 이전의 인프라 환경에서는 감히 해볼 엄두가 나지 않았던 유연한 시스템 구축과 운영, 그리고 API를 통한 제어를 할 수 있게 될 것입니다.

이 책은 이러한 API야말로 클라우드의 본질이라고 전제하고 있습니다. 하지만 아쉽게도 이러한 API를 깊이 이해하고 있는 사람은 웹 엔지니어나 애플리케이션 엔지니어들이고 그 중에서도 극히 일부에만 한정되어 있는 것 같습니다. 그나마 다행스러운 것은 이러한 API들이 GUIGraphical User Interface 기반의 관리 콘솔이나 CLICommand Line Interface와 같은 커맨드라인 인터페이스 뒤에 숨겨져, 내부적으로 동작하도록 만들어져 있기 때문에 클라우드 사용자가 API에 대해 자세히 모르더라도 클라우드 환경을 활용할 수 있는 다른 방법이 제공된다는 점입니다.

이 책에서는 API를 잘 모르는 인프라 엔지니어와 인프라를 잘 모르는 애플리케이션 엔지니어를 대상으로 클라우드 인프라의 구성 과정을 마치 애플리케이션을 설계하는 것과 같은 관점으로 설명하려고 노력했습니다. 이 책을 다 읽고 나면 클라우드 인프라와 API가 그 간 인프라 엔지니어와 애플리케이션 개발자 사이에 있던 벽을 허물어줄 수 있는 기술이라는 것을 깨닫게 될 것입니다.

전반부에서는 클라우드 컴퓨팅의 개념과 공통적인 컴포넌트들에 대해 알아 보고 API의 구조에 대해 자세히 다룹니다. 일반적으로 클라우드를 가상화 기술의 연장선에 있다고 생각하는 경향이 있는데 다른 시각에서는 근본적으로 인터넷 기술의 연장선에 있다고도 볼 수 있습니다. 왜냐하면 인터넷 기술은 클라우드 환경을 만드는 데 매우 중요한 기초 지식이며 인터넷 기술이야말로 널리 대중화되었음에도 불구하고 내부 구조를 잘 몰라도 활용할 수 있는 대표적인 기술이 되었기 때문입니다. 클라우드도 곧 이와 같은 행보를 따라갈 것이라고 믿어 의심치 않습니다.

중반부에서는 클라우드에서 서버와 스토리지, 네트워크와 같은 인프라 컴포넌트를 어떻게 하면 API로 제어할 수 있는지에 대해 살펴 봅니다. 후반부에는 그 동안 배운 지식들을 바탕으로 클라우드에 대규모 글로벌 시스템을 구성하기 위해 이뮤터블Immutable[1]한 하이브리드Hybrid 시스템의 구축 방법에 대해 설명합니다. 덧붙여 저자의 경험을 바탕으로 현장에서 사용 가능한 최신의 구성 관리 기법에 대해서도 소개합니다. 이 책의 집필진은 모두 서로 다른 기업과 조직에 몸을 담고 있는 클라우드 전문가들입니다. 이들의 경험과 다양한 노하우들이 이 책 구석구석에 녹아 들 수 있도록 최대한 노력했습니다.

클라우드는 구축은 비교적 간단한 반면에 논리적으로 추상화되어 있기 때문에 눈에 보이지 않는 가상의 시스템 구성을 머릿속에 그려야만 하고 이 과정에서 어려움을 느낄 수 있습니다. 이 책에서는 가능한 한 많은 그림을 실어 클라우드 인프라의 구성 형태를 머릿속에 쉽게 이미지화할 수 있도록 했습니다. 또한 책에서 익힌 내용을 범용적으로 응용할 수 있도록 예로 든 서비스들을 모두 실전에서 많이 활용되는, 사실상 표준 기술들을 사용했습니다. 클라우드의 실제 내부 동작을 설명할 때는 오픈소스라서 내부를 들여다 볼 수 있고 API가 비교적 간

1 역자 주 : Immutable은 사전적인 의미로 '변경할 수 없는, 불변의'와 같은 뜻으로 이전의 인프라 환경이 한번 구축한 후, 부분적으로 설정을 변경하면서 사용하던 것과 달리 이뮤터블한 시스템은 사용 중인 시스템을 변경하지 않고 변경 전의 시스템을 파기하고 바뀐 시스템으로 새로 구축하는 방식으로 변경을 적용합니다. 자세한 내용은 12장을 참고 바랍니다.

결한 오픈스택을 기준으로 설명 배운 내용을 응용할 수 있도록 상용 서비스로 많이 활용되는 AWSAmazon Web Services와 비교 설명하는 방식으로 내용을 구성했습니다.

시중에는 다양한 클라우드 서비스들이 있고 이러한 서비스들에는 API 레퍼런스가 준비되어 있습니다. 이 책을 다 읽은 후, 각 서비스의 API 레퍼런스를 살펴 보면 그 클라우드에서는 어떤 일을 할 수 있는지, 써보지 않고서도 짐작할 수 있게 될 것입니다. 앞으로 이 책을 읽은 독자들이 API 중심으로 클라우드를 파악해 나가면서 특정 클라우드 서비스에 종속되지 않는 클라우드의 본질을 이해할 수 있게 된다면 필자의 한 사람으로써 더한 보람은 없을 것 같습니다.

필자를 대표하여
히라야마 쯔요시(平山 毅)

지은이 소개

히라야마 쯔요시(平山 毅) _3, 5, 6, 7, 8, 9, 10, 11장 집필 및 책 전체 감수

도쿄 이과대학 공학부를 졸업했다. 재학 시절부터 도쿄 이과대학에서 운영한 Sun Site 호스트의 사용자로 컴퓨터 과학과 통계학을 전공했고 전자상거래를 연구했다. Amazon Web Services에서 아키텍트와 컨설턴트 역할을 각각 1년 9개월 정도 수행했는데 2015년 말 시점까지 이 두 역할을 모두 경험한 유일한 일본인이다. AWS Certified Solutions Architect(Professional), AWS Certified DevOps Engineer(Professional)를 비롯한 여러 기술 자격을 보유하고 있다. 난이도가 높은 첨단 기업의 시스템을 글로벌화하기 위해 클라우드 네이티브한 시스템으로 커스터마이징하는 프로젝트를 다수 맡고 있으며 이러한 활동을 할 수 있는 클라우드 아키텍트도 육성하고 있다. 그 전에는 인터넷 관련 ISP와 광고 회사에서 인터넷의 기본 기술을 익혔고 도쿄 증권거래소와 노무라 종합연구소에서 기존의 미션 크리티컬한 증권 시스템을 개방형 시스템으로 마이그레이션하는 일을 진행했다. 이 과정에서 개방형 시스템 기술을 적극적으로 도입하게 되었다. 2016년 2월부터는 주로 아키텍트 기술 고문(顧問) 역할을 하고 있으며 글로벌 서비스, 코그니티브(cognitive) 컴퓨팅, API 경제(economy), 핀테크(fintech) 등 다양한 분야로 활동 폭을 넓히고 있다. 존경하는 엔지니어로는 썬 마이크로 시스템즈(Sun Microsystems)의 빌 조이(Bill Joy)를 꼽는다.

〈그림으로 배우는 시스템 퍼포먼스의 구조〉(쇼에이샤), 〈RDB 기술자를 위한 NoSQL 가이드〉(슈와시스템), 〈서버/인프라 철저공략〉(기술평론사) 등을 공동 집필했다.

나카지마 토모아키(中島 倫明) _4, 5, 6, 7장 집필

일본 오픈스택 사용자 그룹 회장(2012~), 일반 사단법인 클라우드 이용촉진기구 기술 고문(2012~), 국립정보학연구소/TOPSE 강사 (2014~), 도쿄대학 비상근 강사(2015~)를 하고 있으며 일본에서 오픈스택과 클라우드 기술의 보급을 위한 인재 육성에 매진하고 있다. 이토츄 테크노 솔루션스에 근무하며 오픈소스 소프트웨어(OSS)를 사용한 클라우드 기술 개발과 기획을 맡고 있다.

〈오픈소스 클라우드 기반 오픈스택 입문〉(KADOKAWA/아스키 미디어웍스), 〈오픈스택 클라우드 인테그레이션, 오픈소스 클라우드를 사용한 서비스 구축 입문〉(쇼에이샤) 등을 공동 집필했다.

나가이 에쯔지(中井 悦司) _1, 2장 집필

전직 입시학원 강사로 외국계 기업에서 리눅스와 오픈소스를 사용한 프로젝트를 리딩했고 많은 기술 관련 문서를 집필하며 기술 잡지에도 기고했다. 이후 레드햇(Red Hat)으로 옮겨 에반젤리스트 역할을 했는데 특히 기업 시스템에 리눅스와 오픈소스 기술을 보급하는데 주력했다. 저서로는 〈오픈소스 클라우드 기반 오픈스택 입문〉(KADOKAWA/아스키 미디어웍스), 〈오픈스택 클라우드 인테그레이션, 오픈소스 클라우드를 사용한 서비스 구축 입문〉(쇼에이샤)를 공동 집필했다. 최근에 출간한 〈도커 실천 입문-리눅스 컨테이너 기술의 기초부터 응용까지〉(기술평론사)에서는 컨테이너 기술을 활용한 새로운 시스템 운용 방법과 같은 클라우드 환경을 고려한 인프라 기술을 다루고 있다.

야구치 사토시(矢口 悟志) _12장 집필

공학 박사이자 경영학 석사(MBA)이다. 2007년 노무라 종합연구소에 입사했다. 고급 테크니컬 엔지니어이자 공인 IT 아키텍트 자격 보유자로 IT 기반 기술 관련 R&D 부문에서 클라우드 기술에 대한 연구 개발과 Amazon Web Services를 도입하려는 기업을 위해 비즈니스를 개발하고 있다.

모리야마 쿄우헤이(森山 京平) _8장 집필 Twitter : @kyo_flogofrein

나라첨단과학기술대학원 공학 석사로 일본 휴렛팩커드에서 근무 중이다. HP Helion OpenStack Professional Service의 오픈스택 인테그레이터로서 아시아 여러 나라의 인터넷 및 클라우드 서비스 제공 업체를 위해 오픈스택의 통합을 지원하고 있다. 오픈스택을 활용한 IaaS, PaaS 서비스 구축을 지원하고 있으며 그동안 쌓아온 지식과 경험을 살려 오픈스택 기반의 NFV(Network Function Virtualization)를 설계, 구축하고 있다. 처음 접한 OS는 MS-DOS이고 처음 사용한 PC는 NEC PC-9801이다. 고등학생 때 Fedora Core 5, FreeBSD와 같은 유닉스나 리눅스에서 사용되는 오픈소스에 감명 받아 이후 여러 오픈소스 소프트웨어를 사용해 왔는데 특히 최근에는 리눅스 커널 튜닝이나 네트워크 스택 등을 관심있게 들여다 보고 있다.

인터넷 기술에도 매력을 느껴 L1(전원, 설비)부터 L7(애플리케이션)까지 폭넓은 지식을 쌓기도 했다. 클라우드는 누구를 위한 것인가, 클라우드는 어떤 것이어야 하는가에 대해 밤낮으로 고민하며 연구를 계속하고 있다.

모토키 아키히로(元木 顕弘) _7장 집필

NEC 오픈소스 소프트웨어 추진 센터에서 근무 중이다. 오픈스택의 Neutron과 Horizon의 핵심 개발자로 오픈스택 개발에 참여하고 있으며 오픈스택을 활용한 사설 클라우드를 운영 및 지원하고 있다. 라우터, 광역 이더넷 장치부터 스팸 메일 어플라이언스에 이르기까지 연구 개발 경험이 다양하며 FPGA에서 클라우드까지 섭렵한 엔지니어이다.

개인 시간에는 자전거를 타는 것을 좋아하며 오픈스택과 리눅스에 관련된 글을 번역하기도 한다. 맛있는 맥주와 코딩은 둘도 없는 친구라고 믿고 있다. 〈OpenStack 클라우드 인테그레이션, 오픈소스 클라우드를 사용한 서비스 구축 입문〉(쇼에이샤)을 공통 집필했다.

베타 리더의 말

이 책은 대표적인 클라우드 인프라 기술인 아마존 웹 서비스와 오픈스택에 대해 API 레벨부터 인프라 설계, 구축 및 응용까지 상세하게 다루고 있습니다. 또한, 서버 리소스, 네트워크, 스토리지, 인증/보안 등을 구성하고 제어하는 방법을 그림과 함께 다양한 예를 통해 보여주고 있습니다. 한 권으로 마스터하기에는 다소 방대한 주제일 수 있으나 초급자에게는 클라우드 인프라 기술 입문서로, 중급자에게는 전반적인 구동 원리와 대비되는 기술을 습득하여 고급 엔지니어로 도약할 수 있는 실무서가 될 수 있을 것 같습니다.

정유선 님_오픈스택 한국 커뮤니티(OpenStack Korea User Group)

클라우드 쪽에도 관심이 많아 여러 권의 클라우드 관련 서적들을 찾아서 보았지만 이해하는데 있어서는 어느 정도 어려움이 있었습니다. 이 책은 제목 그대로 그림으로 설명을 하려는 노력이 상당히 엿보이는 책이어서 이해하기도 쉽고 입문하는 사람들도 선뜻 볼 수 있게끔 쓰여 좋았습니다. 클라우드 인프라와 이에 쓰이는 API를 알고 싶고, 공부하고 싶다면 이 책을 주저 없이 추천합니다.

이동혁 님_DrawingTeam

클라우드 컴퓨팅의 동작 방법의 거의 모든 것이 이 책 한 권에 있다고 해도 과언이 아닐 정도입니다. 책에서도 말하듯이 클라우드 컴퓨팅의 본질은 인터넷을 통해 필요한 자원을 제어하는 API입니다. 이 책은 AWS와 오픈스택이 API를 통해 어떻게 개발자에게 자원을 할당하고 제공하는지 이해하기 쉽게 설명하고 있습니다. AWS와 오픈스택을 조금이라도 접하고 이 책을 본다면 클라우드 컴퓨팅의 원리를 깊게 이해할 수 있는 매우 좋은 교과서가 될 것이라 생각합니다.

조성수 님_오픈스택 한국 커뮤니티(OpenStack Korea User Group)

오픈스택과 AWS(아마존 웹 서비스)를 어떻게 사용하는지 그 방법과 개념을 간략하게 알 수 있어서 좋았습니다. 특히 그 둘의 차이점을 쉽게 설명하고 있어서 처음 클라우드를 접하는 사람에게 적합하리라고 봅니다. 저는 클라우드를 사용하기가 어려워 망설인 적이 있는데 실습으로 클라우드를 이해하기보다 개념을 먼저 쉽게 파악하여 접근하고 싶은 분에게 이 책을 추천합니다.

이태헌 님_STA 테스팅 컨설팅

오픈스택과 AWS는 각각 오픈소스 및 상용 클라우드에서 자리매김을 하고 있음에도 불구하고, 때로는 같은 내용인데 서로 다른 용어를 사용한다거나, 클라우드를 활용하였을 때 얻을 수 있는 이점을 서로 다른 관점에서 해석하기도 합니다. 이 책은 각 클라우드 분야의 전문가들이 두 클라우드를 그림과 함께 상세히 설명하여 API와 함께 서로를 재미있게 비교해볼 수 있는 점이 가장 매력적입니다. 다소 어려울 수 있는 개념들을 그림과 함께 설명하고, API를 통해 어떻게 데이터가 오가는지를 살피다보면 어느새 책 전체를 두루 읽게 됩니다.

한때 번역팀 활동을 같이 했던 친구인 모토키 아키히로가 집필한 부분인 오픈스택 네트워크 파트를 살펴 보고 싶은 마음에 베타 리딩에 참여했습니다. 베타 리딩 기간 동안 역자를 비롯한 베타 리더들과 이메일로도 직접 소통했고 뿐만 아니라 AWS 사용자 그룹 및 오픈스택 사용자 그룹에서의 논의, 슬랙, 깃(git) 등 다양한 채널을 활용해 의견 수렴을 하였기에, 번역의 질 또한 좋다고 생각합니다. 개인적으로 좋은 많은 분들과 베타 리딩을 함께할 수 있어 영광이었으며, 차후 다른 클라우드와 같이 살펴볼 수 있는 좋은 책들이 많이 나왔으면 좋겠습니다.

최영락 님_오픈스택 한국 커뮤니티(OpenStack Korea User Group)

번역에 도움 주신 분들

보통은 저자가 감사의 글을 쓰는데 역자가 감사 인사를 드리려 합니다. 이 책에서 다루는 내용이 상당히 폭이 넓고 클라우드 서비스별로 단어나 표현에 차이가 있어 번역 과정에서 관련 커뮤니티 분들의 도움을 받았습니다. 도움을 주신 모든 분들께 감사 드립니다.

■ 아마존 웹 서비스(Amazon Web Service) 한국 사용자 모임

김현민 님(evilskel) 박상욱 님(sanguk.park.37) 정도현 님(dohyjung)

■ 오픈스택 한국 커뮤니티(OpenStack Korea User Group)

강대명 님(daemyung.kang) 강성진 님(ujuckr) 김태욱 님(chatmat)

박광희 님(bebule) 송인섭 님(csssis35) 이어형 님(eohyung.lee)

이보성 님(gotocloud) 이현석 님(hyunsuk80) 조성수 님(seongsoo.cho)

황후순 님(whosoon.hwang)

■ 일본에서 일하는 한국인 개발자 모임(Kodeveloper)

김세화 님(negabaro) 김태욱 님(chatmat) 심재민 님(shim0413)

이상준 님(richellin7) 전민수 님(minsoo.jun)

이 책에서 사용하는 표현법

이 책에서는 독자의 이해를 돕기 위해 다음과 같은 표기 방법을 사용합니다. 오픈스택과 AWS와 관련된 내용들 중, 문자열을 치환하는 표현법이 다양한 형태로 혼용되어 있습니다. 이는 오픈스택과 AWS의 공식 문서를 참고할 때 이질감이 없도록 각 공식 문서의 관례를 최대한 반영한 내용이니 문맥에 따라 해당 의미를 잘 파악할 수 있도록 다음 표기 방법을 미리 확인해두기 바랍니다.

외래어 표기

■ 영문을 그대로 쓰는 경우

이 책이 주로 다루고 있는 AWS와 오픈스택 관련 서비스나 컴포넌트명은 모두 영문 표기를 그대로 썼습니다. 이는 독자가 이 책을 본 후, 검색 엔진에서 관련 자료를 찾기 쉽게 하기 위한 것으로 무리한 번역으로 연계 학습이 방해되지 않도록 했습니다.

■ 영문 발음을 한글로 음차(音借) 표기한 경우

서비스명이나 제품명이 아닌 용어 중, 통상적으로 관련 업계에서 영어 단어로 널리 사용되는 것으로 무리하게 번역할 경우, 의미 전달이 어렵다고 판단되는 단어는 표기는 한글로 하되, 발음은 영문 발음이 나도록 외래어 표기법에 맞춰 작성했습니다.

■ 한글로 표기하고 영문이나 한자로 보조한 경우

문맥상 한글 표현으로 해도 무방한 것은 가능한 한 한글로 표기했습니다. 단, 보다 정확한 의미 전달이 필요하다고 판단되는 단어는 한글 단어와 함께 영문 혹은 한자를 병행하여 표기했습니다. 병행 표기는 처음 해당 단어가 사용될 때에만 사용하고 이후부터는 생략합니다. 단, 해당 단어의 사용 위치가 최초로 언급된 지면에서 멀리 떨어져 다시 상기시킬 필요가 있다고 판단될 경우는 병행 표기를 다시 했습니다.

CLI에서 사용하는 표현법[1]

■ '[ ]'를 사용하는 경우

'[ ]'를 포함하여 생략이 가능하다는 의미입니다.

예시	의미	활용 예
[container]	컨테이너명	swift post [container] [object]
[object]	오브젝트명	

■ '〈 〉'를 사용하는 경우

'〈 〉'를 포함하여 내용이 치환된다는 의미입니다.

예시	의미	활용 예
〈container〉	컨테이너명	swift download 〈container〉 〈object〉
〈object〉	오브젝트명	

■ '[〈 〉]'를 사용하는 경우

'〈 〉'를 포함하여 내용이 치환되는데 생략도 가능하다는 의미입니다.

예시	의미	활용 예
[〈container〉]	컨테이너명	swift delete [〈container〉] [〈object〉]
[〈object〉]	오브젝트명	

1 역자 주 : 표기법은 다음 문서를 참고했습니다.
http://docs.openstack.org/cli-reference/swift.html

■ '_'를 사용하는 경우

위의 '[ ]', '〈 〉' 사이에 들어가는 단어가 두 개 이상의 단어로 구분되면 단어 사이의 구분자로 사용합니다.

예시	의미	활용 예
〈container〉	컨테이너명	swift upload 〈container〉 〈file_or_directory〉
〈file_or_directory 〉	파일명 혹은 디렉터리명	

■ '{ }', '_'를 사용하는 경우

'{ }'를 포함하여 내용이 치환된다는 의미입니다. '{ }' 사이에 들어가는 단어가 두 개 이상의 단어로 구분될 경우 단어 사이의 구분자로 '_'를 사용합니다.

예시	의미	활용 예
{name}	이름	X-Remove-Container-Meta-{name}: {value}
{value}	값	

■ '$'를 사용하는 경우

'$'를 포함하여 인접한 단어가 치환된다는 의미입니다

예시	의미	활용 예
$publicURL	URL	curl -I $publicURL -H "X-Auth-Token: $token" "X-Container-Write: account1"
$token	토큰 값	

URI 경로에서 사용하는 표현법[2]

■ '{ }', '_'를 사용하는 경우

'{ }'를 포함하여 내용이 치환된다는 의미입니다. '{ }' 사이에 들어가는 단어가 두 개 이상의 단어로 구분되면 단어 간의 구분자로 '_'를 사용합니다.

예시	의미	활용 예
{server_id}	서버 ID	/servers/{server_id}/ips/{network_label}
{network_label}	네트워크 레이블	

ARN[3]에서 사용하는 표현법[4]

■ 이탤릭체, '–'를 사용하는 경우

이탤릭체로 표현된 부분이 치환된다는 의미입니다. 이탤릭체로 표현된 단어가 두 개 이상의 단어로 구분되면 단어 간의 구분자로 '–'를 사용합니다.

예시	의미	활용 예
region	리전	arn:aws:apigateway:*region*::*resource–path*
resource–path	리소스 경로	

2 역자 주 : 다음 문서를 참고했습니다. http://developer.openstack.org/api–ref/compute/

3 역자 주 : Amazon Resource Names의 두 문자어입니다.
http://docs.aws.amazon.com/ko_kr/general/latest/gr/aws–arns–and–namespaces.html

4 역자 주 : 다음 문서를 참고했습니다.
http://docs.aws.amazon.com/ko_kr/general/latest/gr/aws–arns–and–namespaces.html

리소스 프로퍼티 타입에서 사용하는 표현법[5]

■ 이탤릭체를 사용하는 경우

이탤릭체로 표현된 부분이 치환된다는 의미입니다. 이탤릭체로 표현된 단어가 두 개 이상의 단어로 구분되면각 단어의 첫 글자를 대문자로 표기하거나 '-'과 '_'를 쓰기도 합니다.

예시	의미	활용 예
ComponentName	컴포넌트명	"Type": "AWS::*ComponentName::ResourceName::PropertyName*"
ResourceName	리소스명	
PropertyName	프로퍼티명	

그 외 기타 표현법

■ OOO를 사용하는 경우

상황에 따라 다른 문자로 치환된다는 의미입니다.

■ …를 사용하는 경우

해당 부분을 생략했다는 의미입니다.

5 역자 주 : 다음 문서를 참고했습니다.
http://docs.aws.amazon.com/ko_kr/AWSCloudFormation/latest/UserGuide/aws-product-property-reference.html

리소스 용어집

이 책에서는 클라우드를 설명할 때 오픈스택OpenStack과 AWSAmazon Web Services를 예로 들고 있습니다. 각 서비스의 컴포넌트나 리소스의 역할은 비슷하나 똑같은 이름으로 사용되는 것은 아니기 때문에 필요할 경우 다음 비교표[1]를 참고하기 바랍니다.

컴포넌트명	리소스명	오픈스택[2]	AWS (Amazon Web Services)[3]
테넌트	–	세입자/고객	테넌트
리전	–	리전	리전
가용 영역	–	가용 구역	가용 영역
관리 콘솔	–	OpenStack Dashboard 호라이즌(Horizon)	AWS Management Console
서버	–	Nova	EC2
	서버	인스턴스	인스턴스
	타입	플레이버(flavor)	인스턴스 유형
	이미지	가상 머신 이미지	Amazon 머신 이미지
블록 스토리지	–	Cinder	EBS
	디스크	볼륨	볼륨
	스냅샷	스냅샷	스냅샷
네트워크	–	Neutron	VPC
	스위치	네트워크	–
	서브넷	서브넷	서브넷

1 역자 주 : AWS의 용어는 가능한 AWS 공식 페이지의 한글 번역 내용과 차이가 없도록 했습니다.

2 역자 주 : http://docs.openstack.org/mitaka/ko_KR/install-guide-debian/common/glossary.html

3 역자 주 : http://docs.aws.amazon.com/general/latest/gr/glos-chap.html

컴포넌트명	리소스명	오픈스택[2]	AWS (Amazon Web Services)[3]
네트워크	라우터	라우터	인터넷 게이트웨이, 라우팅 테이블
	포트	포트	엘라스틱 네트워크 인터페이스
	시큐리티 그룹	시큐리티 그룹	보안 그룹
	네트워크 접근 제어	방화벽	네트워크 ACL
	공인(public) IP	플로팅 IP(Floating IP)	엘라스틱 IP(Elastic IP)
오케스트레이션	–	Heat	AWS CloudFormation
	단위	스택	스택
	템플릿	템플릿	템플릿
인증	–	Keystone	IAM
	사용자	사용자	사용자
	그룹	그룹	그룹
	역할	역할	–
	IAM 역할	–	IAM 역할
오브젝트 스토리지	–	Swift	Amazon S3
	저장소	컨테이너	버킷
	파일	오브젝트	객체

목차

4장 IT 인프라의 진화와 API의 기본 철학

5장 서버 리소스를 제어하는 방법

6장 블록 스토리지 리소스를 제어하는 방법

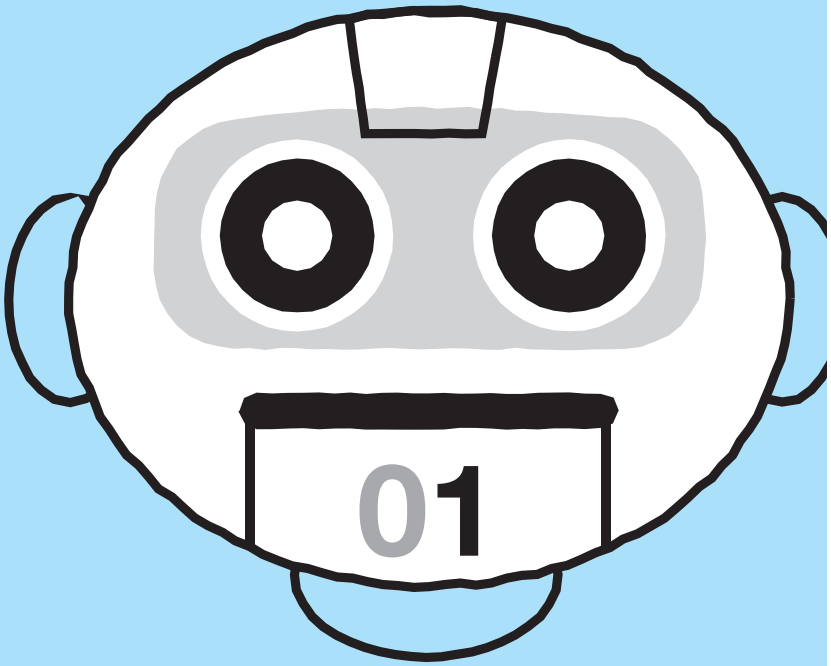
7장 네트워크 리소스를 제어하는 방법

8장 오케스트레이션

10장 오브젝트 스토리지 리소스를 제어하는 방법

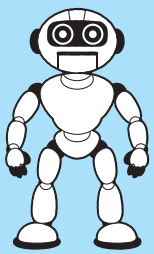
11장 멀티 클라우드

12장 이뮤터블 인프라스트럭처



클라우드 컴퓨팅과 API의 역할

이 책은 '클라우드 컴퓨팅 환경을 API를 통해 관리한다'라는 주제로 내용을 전개하고 있습니다. 클라우드 컴퓨팅이 탄생한 배경을 이 장에서 살펴본 후, 클라우드 컴퓨팅에서 API가 어떤 역할을 하는지 알아 보겠습니다.

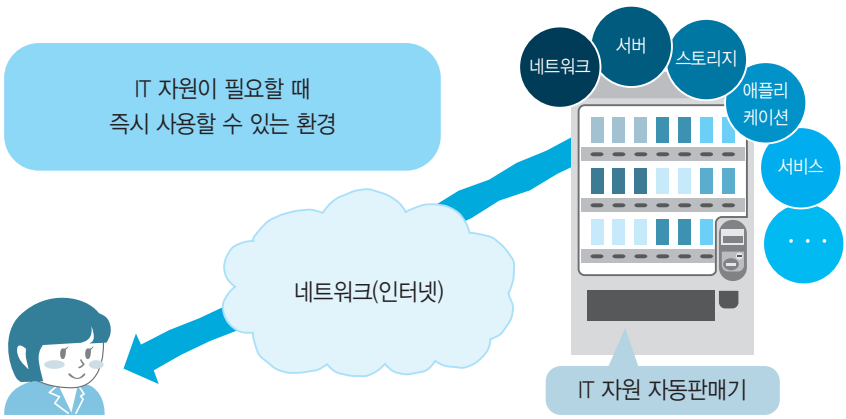


1.1 클라우드 컴퓨팅

흔히 한 마디로 '클라우드 컴퓨팅'이라고 부르지만 사실은 클라우드 컴퓨팅에도 여러 가지 종류가 있습니다. 이 책에서는 그 중에서도 IaaSInfrastructure as a Service형태의 클라우드를 다룹니다. 우선 클라우드 컴퓨팅이 태어난 배경에 대해서 살펴본 후, 전체 클라우드 환경에서 IaaS형 클라우드가 어떤 위치를 차지하고 있는지 알아봅니다.

1.1.1 클라우드 컴퓨팅의 탄생

'클라우드 컴퓨팅'이라는 단어는 2006년에 구글Google의 전 CEO, 에릭 슈미트Eric Schmidt가 자신의 프레젠테이션에서 이 단어를 처음 사용했다고 알려지고 있습니다. 이후 인터넷을 기반으로 한 각종 서비스를 '클라우드 서비스'라고 부르게 되었는데 당시에는 일종의 마케팅 용어로 사용할 뿐 실제로는 클라우드 컴퓨팅이 무엇인지, 그 누구도 명확하게 제시하지 못했습니다. 그래서 당시 ICT 관련 기업과 엔지니어 사이에서는 클라우드 컴퓨팅에 대해 나름대로의 정의를 세워 갑론을박하는 일도 많았습니다.



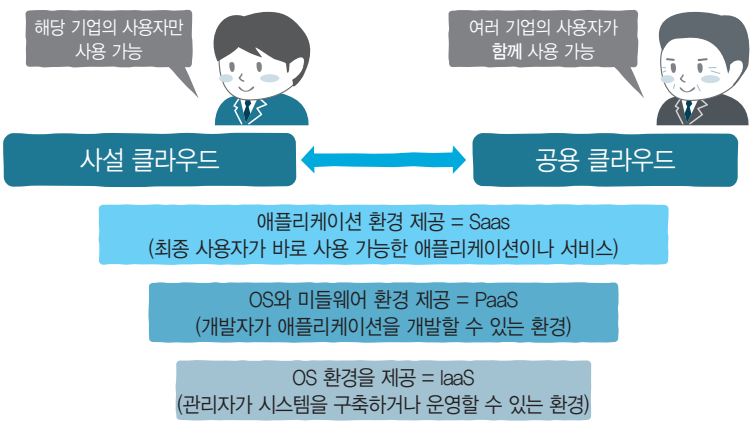
▲ 그림 1.1 클라우드 컴퓨팅의 실현

과거의 클라우드 서비스에 대한 역사를 되짚어 보면 실제로 클라우드 컴퓨팅이 제대로 실현되었다고 볼 수 있는 시기는 '필요한 IT 자원을 즉시 사용할 수 있는 환경'이 만들어진 시점입니다. 이른바 'IT 자원의 자동판매기'와 같은 역할이 가능하게 된 시기라고 볼 수 있습니다([그림 1.1] 참고).

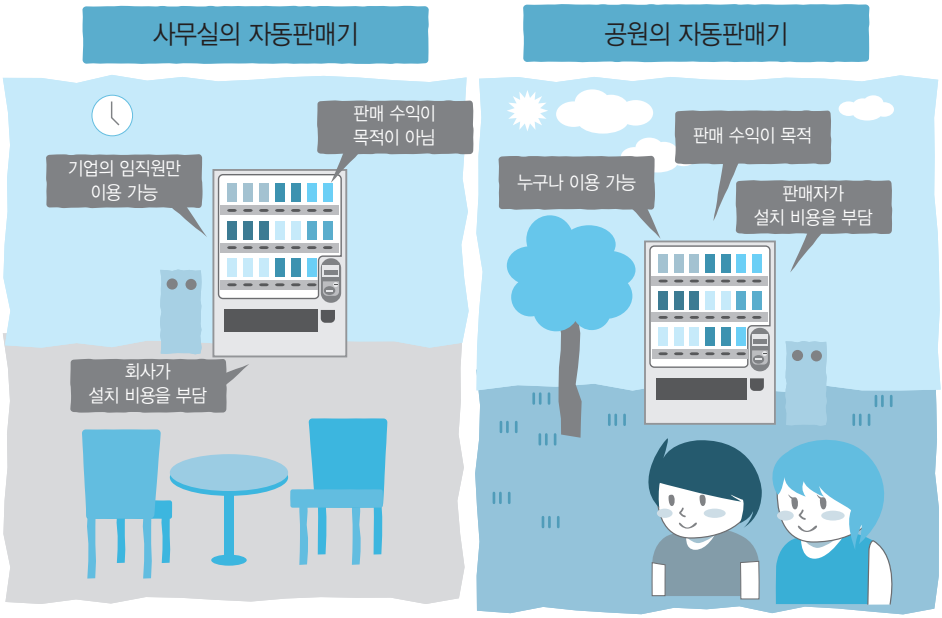
이렇게 만들어진 클라우드 컴퓨팅 환경은 그 기능을 구현하기 위한 다양한 기술 요소를 필요로 합니다. 그리고 클라우드 컴퓨팅의 진정한 본질은 물리적이고 기술적인 시스템 구조나 전산 장비들로만 만들어지는 것이 아닙니다. 이 책에서는 클라우드 컴퓨팅의 본질이 API에 있다고 보고 책 전반에 걸쳐 이에 대해 설명을 풀어내려고 합니다.

1.1.2 공용 클라우드와 사설 클라우드의 차이

오늘날 수많은 클라우드 서비스가 나오고 있지만 결국은 '누가 사용하는가?'와 '무엇을 제공하는가?'의 두 가지 관점으로 분류할 수 있습니다([그림 1.2] 참고). 이러한 관점을 앞서 예로 든 자동판매기에 빗대어 보면 '누가 사용하는가?'는 '자동판매기 사용자'에, '무엇을 제공하는가?'는 '판매 상품'에 해당합니다.



▲ 그림 1.2 사용자와 제공하는 서비스에 따른 클라우드의 분류



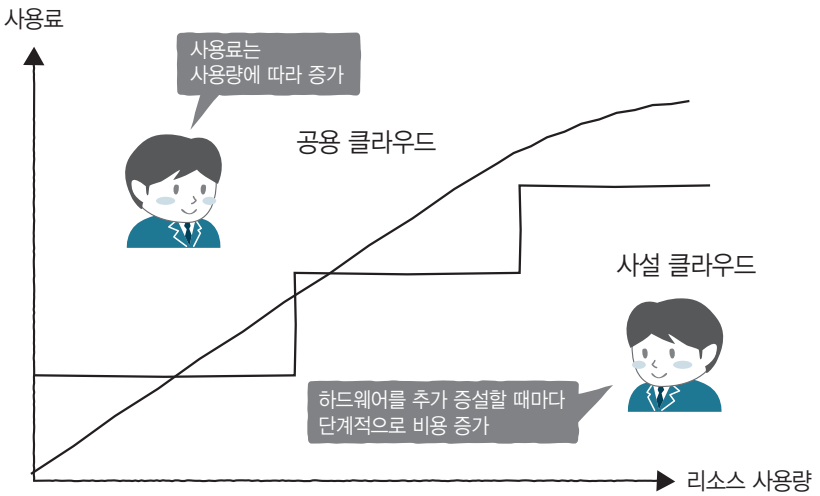
▲ **그림 1.3** 사무실의 자동판매기와 공원의 자동판매기

우선 '사용자' 관점에서의 클라우드 서비스에 대해 살펴 보겠습니다([그림 1.3] 참고). 이른바 '사설private 클라우드'는 특정 기업 내부에서 사용하는 기업 자체의 전용 클라우드 환경을 말하며, 해당 기업의 사용자만 접근할 수 있도록 통제됩니다. 회사 휴게실에 설치된 자동판매기와 비슷한데 그 회사의 임직원은 자유롭게 자동판매기를 이용할 수 있지만 출입이 제한된 외부인은 사용할 수 없습니다. 이런 자동판매기는 회사에서 설치할 장소를 마련해주기 때문에 공간 임대료는 발생하지 않으며 전기 사용료도 회사가 부담합니다. 그리고 상품을 판매하여 이익을 남길 필요가 없기 때문에 상품 가격이 시중에 비해 상대적으로 싸다는 장점이 있습니다.

한편 '공용public 클라우드'는 여러 기업의 사용자가 함께 사용할 수 있도록 만들어져 있습니다. 공용 클라우드의 대부분은 '멀티 테넌트multi-tenant'라는 기능을 갖추고 있어서 겉으로는 자신만 사용하는 클라우드로 보이지만 실제로는 여러 다른 사용자도 자신만의 격리된 공간에서 같은 클라우드 환경을 사용할 수 있습니다. 실제로 클라우드를 구성하는 물리적인 장비나 소프트웨어가 여러 사용자를 수용할 수 있도록 만들

어져 상호 간의 간섭 없이 독립적인 형태로 사용할 수 있습니다. 공원의 자동판매기와 같이 누구나 이용할 수 있다는 장점이 있는 반면 자동판매기를 설치한 사업자는 상품을 판매하여 수익을 내야 하기 때문에 사내의 자동판매기보다 상품 자체로만 보자면 가격이 상대적으로 비싸다고 느껴지는 단점도 있습니다.

이러한 현상을 실제 클라우드 환경에 빗대어 본다면 비용 구조의 차이로 설명할 수 있습니다. 기업에 전용 클라우드 환경을 구축하려면 데이터 센터와 같은 공간을 시작으로 각종 하드웨어 자산들을 확보할 필요가 있는데, 이것은 곧 초기 투자 비용이 발생한다는 것을 의미합니다. 한편 공용 클라우드 서비스를 이용하면 이런 비용 부담은 발생하지 않는데 그 이유는 리소스가 필요할 때마다 필요한 만큼만 확보해서 이용할 수 있기 때문입니다. 그래서 필요로 하는 리소스의 총량에 큰 변화가 예상되거나 수요 추이를 예측하기 어려운 상황이라면 공용 클라우드를 이용하는 것이 유리할 수 있습니다.



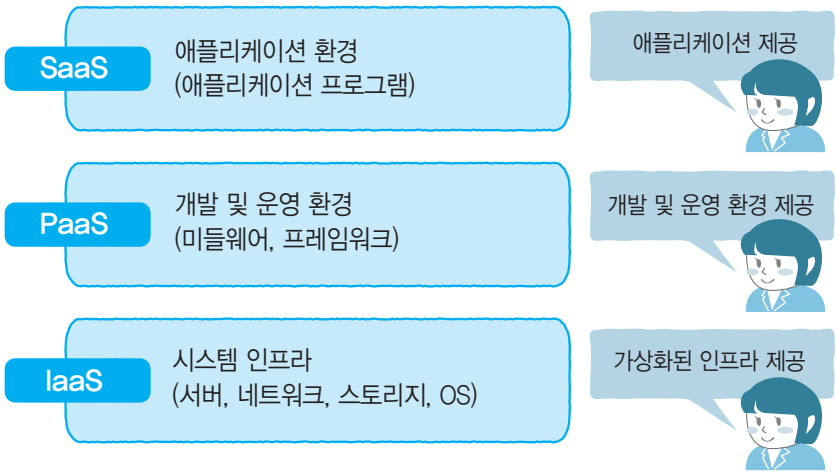
▲ **그림 1.4** 사설 클라우드와 공용 클라우드의 비용 구조

반대로, 일정 규모로 리소스가 활용될 것이라고 예측할 수 있는 경우, 그 규모에 맞는 사설 클라우드를 도입하고 운영하는 것이 비용 측면에서 유리할 수 있습니다. 공용 클라우드는 리소스의 사용량에 따라 과금되기 때문에 보통, 비용이 선형적(線形的)으로

증가합니다(실제로는 사용량이 늘어나면 할인되기도 합니다). 한편 사설 클라우드는 처음 준비한 리소스가 고갈되기 전까지는 추가 투자가 발생하지 않습니다. 다만 리소스가 부족할 때마다 일정량의 하드웨어를 추가로 투자하게 되므로 비용은 단계적(段階的)으로 상승하게 됩니다. 이런 비용의 변화를 나타낸 것이 [그림 1.4]입니다.

1.1.3 IaaS, PaaS, SaaS의 차이점

다음은 '판매 상품' 관점에서 살펴 보겠습니다. 클라우드 환경에서의 판매 상품은 IaaSInfrastructure as a Service, PaaSPlatform as a Service, SaaSSoftware as a Service에 해당합니다([그림 1.5] 참고). 역사적으로 클라우드 서비스는 SaaS 형태의 서비스에서 시작하여 점차 다양한 형태로 분화되었습니다. SaaS는 기업의 CRMCustomer Relationship Management 애플리케이션과 개인용 이메일 서비스 등과 같이 최종 사용자가 직접 사용하는 애플리케이션 환경을 클라우드의 형태로 서비스합니다. 사실 이런 형태의 서비스는 클라우드 컴퓨팅이라는 용어가 등장하기 이전부터 ASPApplication Service Provider라는 이름으로 제공되었습니다. 이후 클라우드가 관심을 받게 되자 일종의 마케팅 전략으로 '클라우드 서비스'라고 부르게 되었습니다.

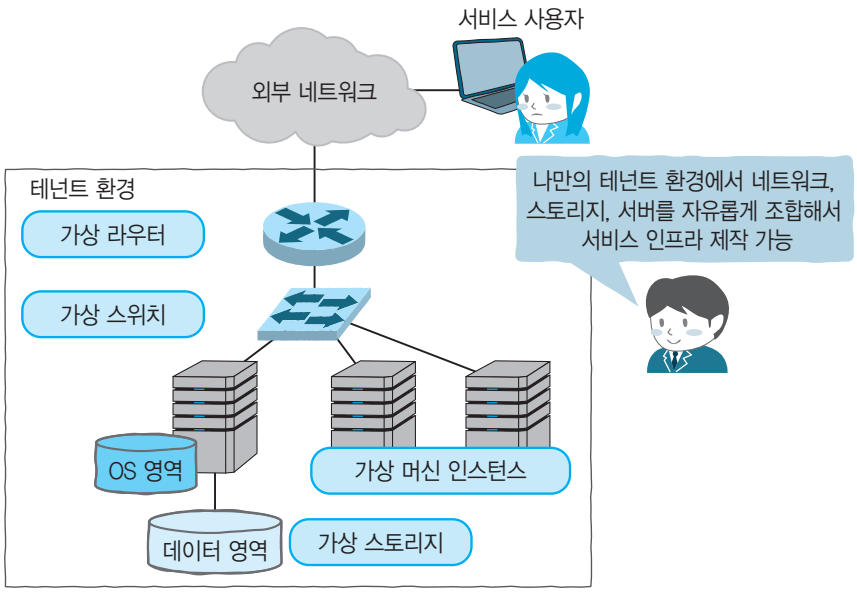


▲ 그림 1.5 IaaS, PaaS, SaaS가 제공하는 리소스

PaaS 형태의 클라우드 서비스에서는 애플리케이션 개발 환경이나 운영 환경 등을 제공합니다. 애플리케이션을 개발하려면 애플리케이션 서버와 백엔드 데이터베이스, 개발 프레임워크와 컴파일러 등이 필요합니다.

PaaS를 활용하면 이러한 개발 환경이 기본으로 제공되기 때문에 애플리케이션 개발자는 보다 신속하게 개발에 착수할 수 있습니다. 그리고 기존에 사용하던 프레임워크나 데이터베이스를 클라우드 환경에서도 똑같이 사용할 수도 있고[1] 특정 클라우드 서비스에서만 제공하는 고유한 프레임워크나 데이터 저장소를 사용할 수도 있습니다.[2]

다음은 이 책의 주제인 IaaS 형태의 클라우드 서비스에 대해 알아 보겠습니다. IaaS에서는 서버, 네트워크, 스토리지 등의 IT 인프라에 필요한 다양한 구성 요소들을 서비스 형태로 제공합니다([그림 1.6] 참고). 그리고 IaaS 서비스 사용자에게는 자신만의 전용



▲ 그림 1.6 IaaS 형태의 클라우드 서비스

1 역자 주 : AWS에서는 Oracle, Microsoft SQL Server, PostgreSQL, MySQL, MariaDB를 사용할 수 있습니다.
• https://aws.amazon.com/ko/rds/

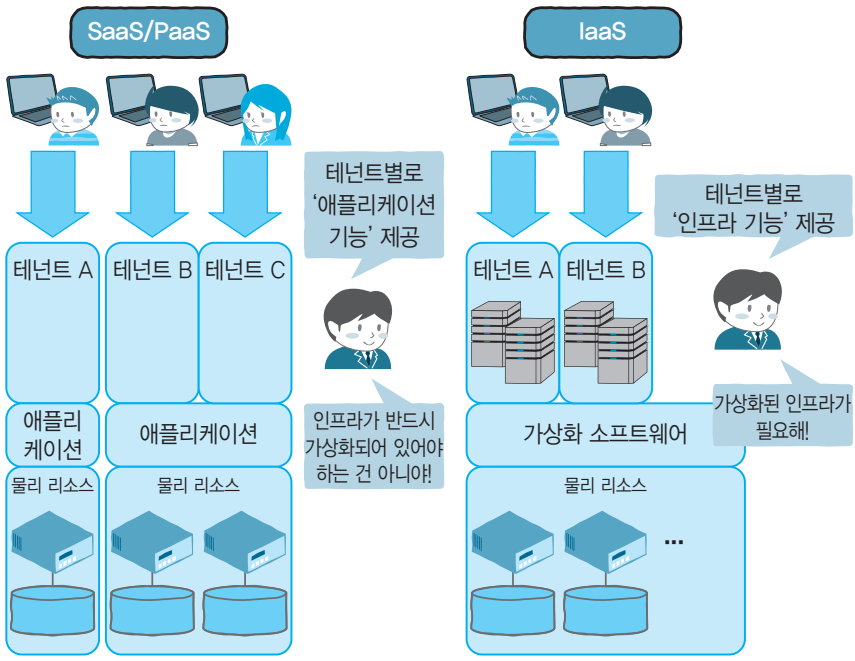
2 역자 주 : AWS의 경우, MySQL과 PostgreSQL을 확장한 Amazon Aurora가 있습니다.
• https://aws.amazon.com/ko/rds/aurora/

테넌트 환경이 제공됩니다. 사용자는 이 환경에서 가상 라우터나 가상 스위치와 같은 가상 네트워크 장비, 그리고 가상 머신 인스턴스라고 하는 가상 서버, 데이터 저장을 위한 블록 스토리지 등을 자유롭게 추가하여 시스템을 구성할 수 있습니다. 즉 IaaS 형태의 클라우드에서는 가상화된 환경에서 서버, 네트워크, 스토리지와 같은 IT 인프라의 3대 요소를 자유롭게 조합하여 자신만의 서비스 인프라를 만들 수 있습니다.

[그림 1.5]와 같이 SaaS, PaaS, IaaS의 차이는 사용자에게 제공하는 IT 자원의 차이라고 볼 수 있습니다. 이때 IaaS 위에 PaaS가, PaaS 위에 SaaS가 만들어지는 것은 아니라는 점에 주의해야 합니다. 예를 들어 SaaS에서는 서비스 사용자 관점에서 볼 수 있는 영역이 애플리케이션 사용자 인터페이스UI: User Interface로 제한됩니다. 만약 애플리케이션에서 멀티 테넌트 기능을 구현하고 있다면 같은 서버, 같은 애플리케이션을 사용하더라도 여러 사용자를 수용하는 것이 가능합니다. 즉, SaaS 입장에서는 인프라 자체를 반드시 가상화할 필요는 없습니다. 한편, IaaS는 서버나 네트워크와 같은 인프라 구성 컴포넌트들을 사용자별로 독립된 형태로 제공할 수 있어야 합니다. 그래서 IaaS 입장에서는 대부분의 인프라를 구성하는 컴포넌트들을 가상화해서 제공해줄 필요가 있습니다[3]([그림 1.7] 참고).

이와 같이 IaaS형 클라우드에서는 제공되는 리소스가 물리적인 환경과는 독립된 형태로 가상화되어 있다는 것이 가장 두드러진 특징 중의 하나입니다. 뒤에 자세히 다루겠지만 이렇게 가상화된 자원을 효과적으로 제어하는데 API가 큰 역할을 하게 됩니다.

3 저자 주 : IaaS형 클라우드 서비스 중에도 가상 서버가 아닌 물리 서버를 이용할 수 있는 것이 있습니다. 다만 이러한 서비스에서도 실제 서버를 의식할 필요 없이 가상 머신을 다루는 것처럼 API로 제어할 수 있어, 클라우드 서비스의 'API로 제어 한다'는 본질 자체는 변하지 않습니다.



▲ 그림 1.7 SaaS, PaaS, IaaS의 차이점

1.2 클라우드가 실현하는 인프라의 표준화

앞서 IaaS형 클라우드에서는 각종 인프라 리소스가 가상화를 통해 물리적 환경과는 독립되어 있다고 설명했습니다. 이제 이 내용에 대해 좀 더 자세히 살펴 보겠습니다. 이 절을 보고 나면 가상화로 비롯되는 클라우드의 장점이 무엇인지 이해할 수 있을 것입니다.

1.2.1 클라우드에 의한 시스템 구축 절차의 표준화

과거에는 새로운 시스템을 구축할 때 서버나 스토리지와 같은 물리적 장비를 도입하기 위한 준비부터 하는 것이 순서였습니다. 그리고 이러한 과정은 쉽지 않고 복잡한 절차를 거쳐야 하는 고된 작업이었습니다.

그 밖에도 서버를 설치하고 배선하는 작업 역시 상당히 번거로울 뿐만 아니라 작업 시간도 오래 걸린다는 것을 쉽게 짐작할 수 있습니다. 하지만 실제로는 이 정도 작업은 쉬운 편에 속하고 그 밖에도 귀찮은 작업들이 많이 남아 있습니다.

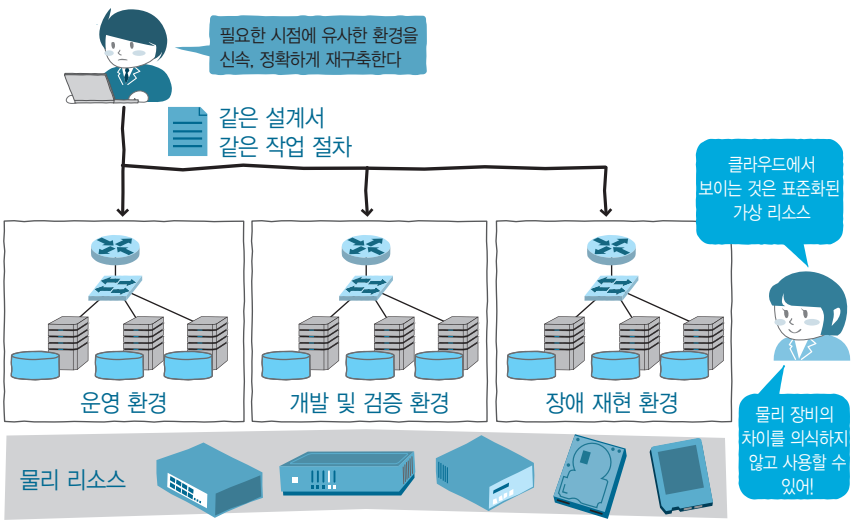
장비를 도입할 때는 구축하려는 시스템의 규모에 맞춰 적절한 기능과 성능을 충족하도록 장비의 사양을 선택하는 과정이 선행되어야 합니다. 하드웨어의 경우 각 제조업체들이 치열하게 경쟁하면서 수시로 새로운 제품을 쏟아내고 있어 제품을 비교, 분석하는 것 자체가 쉽지 않습니다. 각 기기들의 기능에도 변화가 있기 마련이고 오래된 기종은 언젠가 기술 지원이 중단되거나 판매 중지가 되기도 합니다. 그래서 각 하드웨어 벤더의 제품 판매 상황이나 최신 기종의 기능 및 성능을 면밀히 검토하여 도입 시점에서 가용한 제품들 중, 가장 적절한 것을 선택할 수 있어야 합니다([그림 1.8] 참고).

▲ **그림 1.8** 하드웨어를 선정하는 절차

지금까지 없었던 새로운 기능을 사용해야 하는 경우, 시스템 구축 작업은 더 복잡해지는데 이전 장비와 비슷한 기능이더라도 제품이 다르면 설정 방법도 달라질 수 있습니다. 그래서 미리 제품 설명서를 보면서 설정 방법에 어떤 차이가 있는지 확인해둘 필요가 있습니다. 경우에 따라서는 해당 제품의 전문가를 불러 사전 기술 검토를 하기도 합니다. 이와 같은 복잡한 과정들을 겪어 보면 시스템 구축 준비 과정만으로도 몇 달이 걸린다는 말이 결코 거짓이 아님을 체감하게 됩니다.

이에 반해, IaaS형 클라우드에서는 이와 같은 물리적 장비의 설정 방법이나 기능의 차이를 의식하지 않아도 됩니다. 앞서 살펴본 [그림 1.6]의 가상 라우터나 가상 스위치와 같은 가상 네트워크 장비의 경우, 실제로는 물리적인 네트워크 장비가 있고 그 위에 네트워크를 가상화하기 위한 소프트웨어가 동작하는 방식으로 만들어져 있습니다.

클라우드 사용자가 볼 수 있는 부분은 서비스 형태로 표준화되고 추상화된 라우터나 스위치의 기능입니다. 일단 클라우드상에서 가상 네트워크 장비의 사용법을 익히고 나면 이후 물리적인 장비의 차이에 상관없이 몇 번이든 같은 방법으로 시스템을 구축할 수 있습니다([그림 1.9] 참고).

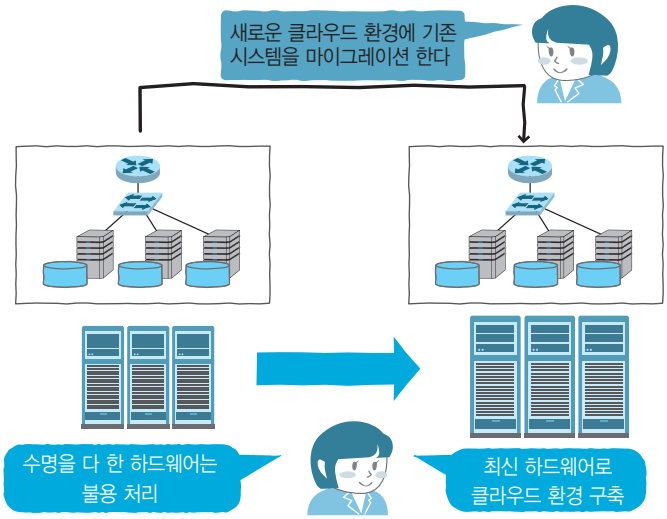


▲ 그림 1.9 물리 장비의 차이를 의식하지 않고 시스템을 구축

이와 같이 클라우드 환경을 이용하면 시스템을 구축해야 할 때마다 그 시점에 가용한 장비를 수배하거나 그 장비에 대한 설정 방법을 다시 익힐 필요가 없습니다. 이전에 한 번 구축해 본 시스템이 있다면 같은 방법으로 여러 번 반복해서 재구축할 수 있으므로 새 시스템을 구축해야 하는 상황에도 시행착오 없이 이전의 경험을 살려 효율적으로 작업할 수 있습니다.

클라우드 인프라를 구성하는 물리적 장비는 기술이 발전함에 따라 점차 고도화되고 진화해 나갑니다. 사내에서 사설 클라우드를 사용하고 있다면 정기적으로 최신 장비로 교체하면서 항상 새로운 클라우드 인프라로 유지할 수 있을 것입니다. 하드웨어는 신제품일수록 성능이 더 나아지고 전력 소비 또한 줄어듭니다. 그래서 같은 수의 서버여도 이전보다 더 많은 가상 머신을 제공할 수도 있습니다.

분명한 것은 이런 경우에도 클라우드의 사용 방법은 바뀌지 않는다는 것입니다. 그러다 보니 기존의 클라우드 인프라에서 동작하는 시스템을 새로운 클라우드 인프라로 옮기는 작업이 이전보다 수월해졌고 그러다보니 성능이 좋은 클라우드 시스템으로 옮겨 넣는 시스템 마이그레이션migration도 과거보다는 쉽게 할 수 있게 되었습니다. 그래서 몇몇 기업에서는 기존의 장비에서 운영되는 시스템을 클라우드 환경으로 단계적으로 옮기면서 노후 장비를 줄이는 목적으로 클라우드를 활용하기도 합니다([그림 1.10] 참고).



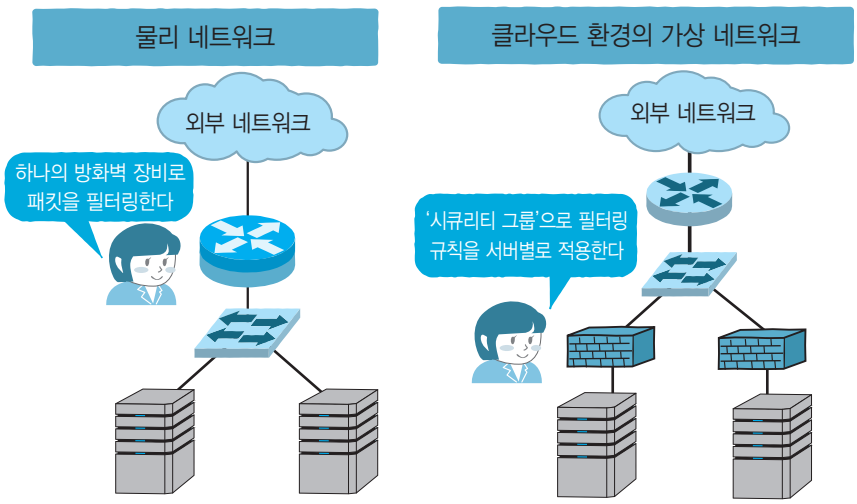
▲ **그림 1.10** 클라우드를 활용한 하드웨어의 생명 주기 관리

1.2.2 인프라의 표준화를 위한 컴포넌트의 추상화

클라우드 환경에서 시스템의 구축 과정을 표준화할 수 있는 것은 시스템을 구성하는 컴포넌트들이 추상화되어 있기 때문입니다. 물리적인 장비가 제공하는 고유 기능은 그 장비의 물리적인 구성 방식에 의존하지만 클라우드가 제공하는 컴포넌트에는 그러한 제약이 없습니다. 그 이유는 클라우드의 컴포넌트들이 철저하게 사용자가 필요로 하는 기능에 충실하도록 만들어졌기 때문입니다.

가상 머신에서 오가는 네트워크 패킷을 필터링해야 하는 상황을 가정해 보겠습니다. 물리적인 환경이라면 방화벽 장치의 설정에 '특정 서버에 오고 가는 특정 패킷을 통제한다'와 같은 규칙을 정의해야 합니다. 하지만 서버 관리자의 관점에서 보면 필터링을 하고 싶은 대상이 자신이 관리하는 서버에 오가는 패킷이지 특정 방화벽 장치에 오고 가는 패킷은 아닙니다.

클라우드 환경에서 제공되는 방화벽 기능은 물리적인 방화벽 장치를 의식하지 않고도 사용할 수 있도록 만들어져 있습니다. 필터링 규칙을 미리 정의해서 '시큐리티 그룹'을 만든 다음 이 규칙이 필요한 서버에 적용시키기만 하면 됩니다([그림 1.11] 참고). 이렇게 하면 마치 가상 머신 앞에 새로운 방화벽 장치가 설치된 것처럼 패킷을 필터링할 수 있습니다.



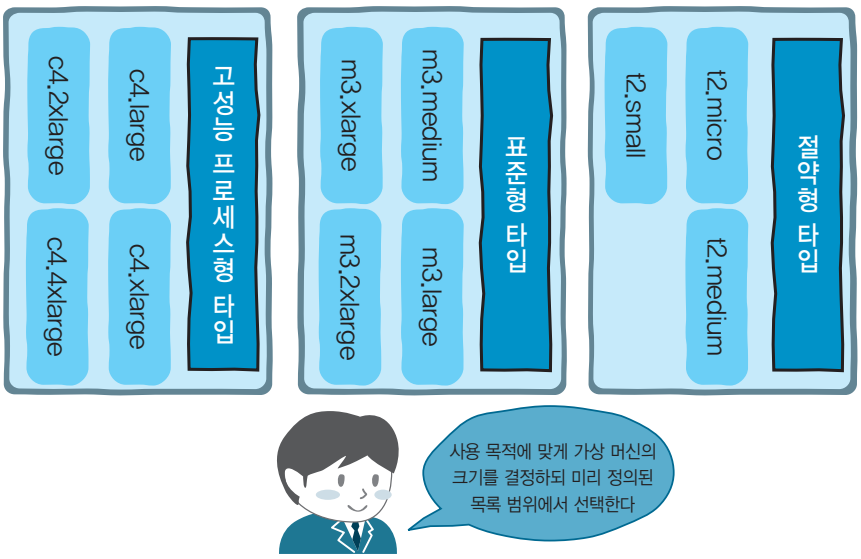
▲ 그림 1.11 클라우드 환경의 방화벽 기능

실제로 물리적인 장비를 사용해서 시스템을 설계해 왔던 인프라 엔지니어에게는 이러한 컴포넌트의 추상화 개념이 처음에는 익숙하지 않을 수 있지만 곧 그 편리함을 실감하게 될 것입니다. 즉 물리적인 환경에 의존적인 내용에 대해서는 신경을 덜 쓰면서도 시스템을 구성하기 위해 근본적으로 필요한 것이 무엇인지, 좀 더 본질적인 고민과 설계에 집중할 수 있게 됩니다.

이러한 장점은 비단 네트워크 기능뿐만이 아니라 다른 기능들에서도 비슷하게 적용됩니다. 가상 머신을 준비해야 할 경우라면 가상 머신에 할당할 가상 CPU의 개수나 메모리 용량 등을 결정해야 합니다. 이때 물리적인 서버를 설계하는 것이 익숙한 엔지니어라면 실제 장비의 용량을 산정할 때와 비슷한 감각으로 CPU의 코어 개수나 메모리의 용량을 정하려고 합니다. 왜냐하면 실제 물리적인 장비를 발주(發注)할 때는 제품의 수량을 구체적이고 정확하게 명시해야 하기 때문입니다.

하지만 클라우드 환경에서 가상 머신을 준비할 때는 이렇게 세부적인 부분까지 지정하지 않아도 됩니다. 't2.small', 'm3.large'[4]와 같이 미리 정의된 인스턴스 유형instance type을 지정하면 그에 맞는 구성으로 가상 머신이 만들어지기 때문입니다. 이러한 인스턴스 유형들은 시스템 관리자가 미리 정의해서 준비를 해두는 것으로 'CPU 성능이 중요하다', '메모리 용량이 많아야 한다'와 같이 가상 머신의 사용 목적에 따라 유형이 분류되어 있습니다([그림 1.12] 참고). 그래서 클라우드 사용자는 서버 사양에 대한 세부적인 수치에 너무 집착하지 말고 시스템의 사용 목적이 무엇인지에 대해 좀더 무게를 두고 생각하는 요령이 필요합니다.

4 역자 주 : AWS의 가상 머신인 EC2 인스턴스 유형을 예로 들었습니다.



▲ 그림 1.12 가상 머신 생성 시 인스턴스 유형에서 선택

1.2.3 API에 의한 제어 방법의 표준화

이와 같이 IaaS형 클라우드에서는 추상화된 논리적인 컴포넌트들을 조합함으로써 표준화된 절차에 따라 시스템을 구축할 수 있습니다. 즉, 한번 정의한 작업 절차에 따라 같은 사양의 시스템을 몇 번이든 반복해서 구축할 수 있게 됩니다. 물론 이때는 물리적인 인프라 작업이 필요 없습니다. 이렇게 네트워크, 서버, 스토리지 등과 같은 시스템 인프라 환경을 자신의 PC에서 혼자서 조립하는 것이 가능한데 기존의 물리적 장치를 사용한 구축 방법과 비교하면 놀랍도록 빠른 속도로 시스템을 완성할 수 있습니다.

클라우드의 장점은 여기에 그치지 않습니다. 같은 시스템을 구축하기 위해 같은 작업을 여러 번 반복하다 보면 그 과정도 자동화하고 싶어지게 마련입니다. 이전에는 OS를 설치한 후 서버의 환경 설정은 쉘 스크립트 등으로 자동화했는데 최근에는 쉐프Chef[5]나 퍼펫Puppet[6]과 같은 구성 관리 툴을 활용하여 구성이 더 복잡한 경우라도 자동

5 역자 주 : https://www.chef.io/

6 역자 주 : https://puppet.com/

화하는 것이 가능해졌습니다. 다만 서버의 설치나 네트워크 연결과 같은 작업은 자동화하는 것이 어려웠는데 이제 클라우드 환경으로 이마저도 자동화할 수 있게 되었습니다.

이러한 자동화를 가능하게 해주는 것이 바로 API를 사용한 제어 방식입니다. 우선 클라우드 환경에서 시스템을 구축할 때 사용 가능한 다양한 제어 방법들을 살펴 보겠습니다. 크게 [표 1.1]과 같이 정리할 수 있습니다.

가상 머신을 기동하는 것을 예로 들어보겠습니다. 웹 콘솔을 사용하는 경우에는 GUI 메뉴에서 가상 머신을 시작(launch)하는 버튼이나 링크를 선택합니다([그림 1.13] 참고). 가상 머신의 구성 정보를 입력하는 화면이 나오는데 필요한 정보를 입력한 후, 완료를 선택합니다. 내부적으로는 웹 콘솔 프로그램에서 클라우드 관리 서버로 가상 머신을 기동하라는 명령이 전달되는데 이때 사용되는 것이 바로 클라우드 API입니다.

▼ 표 1.1 클라우드 환경에서의 제어 방법

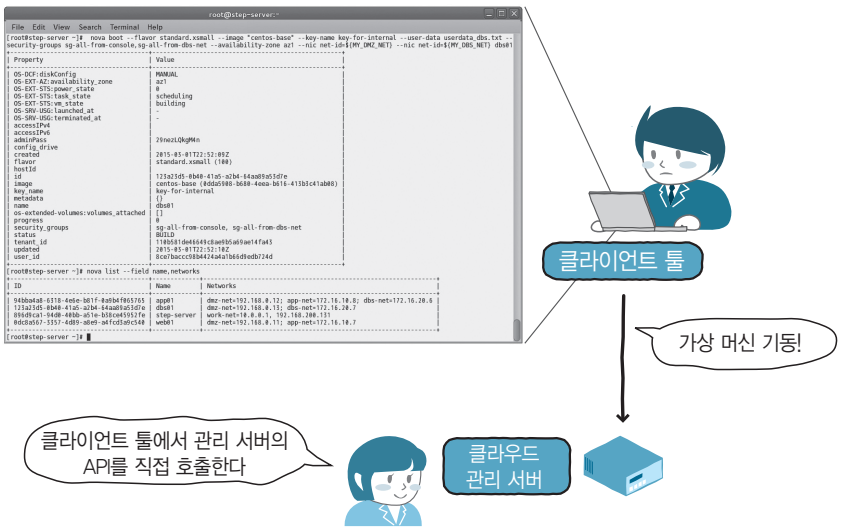
제어 방법	설명
웹 콘솔(GUI)을 사용한 제어	웹 브라우저를 사용해서 GUI 방식으로 제어함
명령어를 사용한 제어	클라이언트 툴이 제공하는 명령을 실행하여 제어함
직접 개발한 프로그램을 사용한 제어	직접 개발한 프로그램이 클라우드 API 라이브러리를 활용하여 제어함
자동화 툴을 사용한 제어	클라우드 환경의 자동화 툴을 사용해서 제어함

▲ 그림 1.13 웹 콘솔을 사용한 제어

API Application Program Interface는 프로그램과 프로그램이 서로 명령을 주고 받기 위해 미리 약속해둔 일종의 규칙입니다. [그림 1.14]는 클라우드 관리 서버를 표현한 것인데 웹 콘솔에서 명령을 내리면 클라우드 관리 서버의 관리 프로그램이 미리 정의된 규칙에 따라 웹 콘솔의 지시 사항을 수행하게 됩니다. 이와 같이 클라우드 관리 서버는 클라우드에서 할 수 있는 다양한 작업들을 API 형태로 제공하고 있습니다.

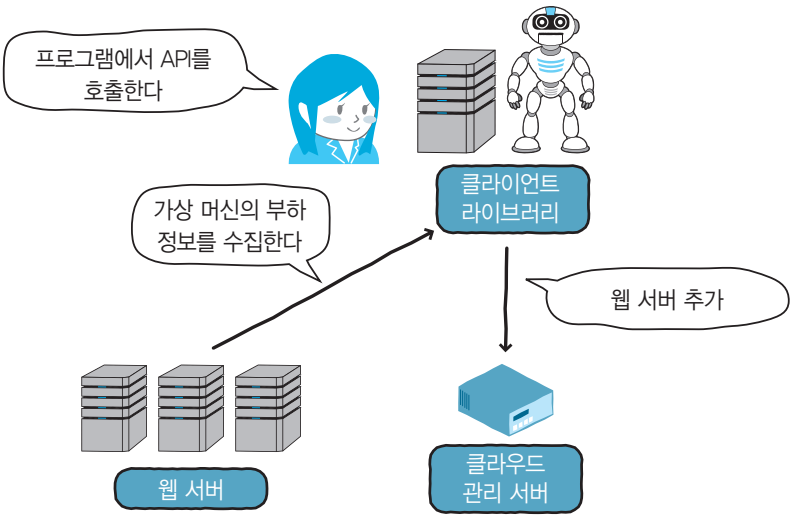
▲ **그림 1.14** 웹 콘솔에서 API를 통해 관리 서버 제어하기

또 다른 방법으로는 PC에 클라우드 관리용 클라이언트 툴을 설치하여 클라우드를 제어할 수도 있습니다. 앞서 살펴본 것과 같이 가상 머신을 기동하는 경우를 예로 들면, 웹 콘솔에서는 위저드wizard 방식의 화면으로 가상 머신을 구성하는데 필요한 정보를 하나씩 입력받아 가상 머신을 기동합니다. 반면 클라이언트 툴에서는 구성에 필요한 세부 설정 정보를 명령어의 옵션으로 지정한 다음, 한 줄의 명령을 실행하는 방법으로 가상 머신을 기동합니다. 내부적으로는 클라이언트 툴 프로그램에서 실행한 명령이 클라우드 관리 서버의 API를 호출하고 그 결과 가상 머신이 제어되는 것입니다 ([그림 1.15] 참고).



▲ **그림 1.15** 클라이언트 툴에서 API를 통해 관리 서버 제어하기

클라이언트 툴과 마찬가지로 클라우드 관리 서버로 API를 요청하는 프로그램을 개발할 수 있다면 자신이 직접 만든 프로그램에서도 클라우드를 제어할 수 있습니다. 각 클라우드 서비스들은 이를 위해 각종 개발 언어를 위한 API 라이브러리를 제공하고 있는데 이러한 라이브러리를 활용하면 고도의 프로그래밍 기술이 없더라도 클라우드를 제어하는 프로그램을 만들 수 있습니다. 예를 들어 루비Ruby와 파이썬Python과 같은 프로그램 언어를 사용하면 웹 서버의 부하에 따라 가상 머신 개수를 조절하여 부하를 분산하는 오토스케일링Auto Scaling과 같은 처리도 프로그램적으로 어렵지 않게 할 수 있습니다([그림 1.16] 참고).



▲ 그림 1.16 프로그램에서 API를 통해 클라우드 관리 서버 제어하기

하지만 최근에는 API를 제어하는 클라이언트 툴의 기능이 좋아져서 굳이 API를 호출하는 프로그램을 직접 개발할 필요가 없어졌습니다. 특히 오토스케일링처럼 자주 사용하는 자동화 패턴은 이미 클라우드 측에서 기능을 만들어 서비스하기 때문입니다. 이와 같이 어떤 다양한 방법을 사용하더라도 그 이면에는 API가 동일하게 사용되고 있습니다. 클라우드의 API를 이해하는 것은 클라우드를 제어하기 위한 '최소 단위', 혹은 클라우드의 '기본 단위 기능'을 이해하는 것과 같습니다. 이러한 기본 단위 기능을 이해하면 향후 각종 자동화 툴들을 능수능란하게 다루는 데 꼭 필요한 배경 지식이 되므로 반드시 잘 익혀두기 바랍니다.

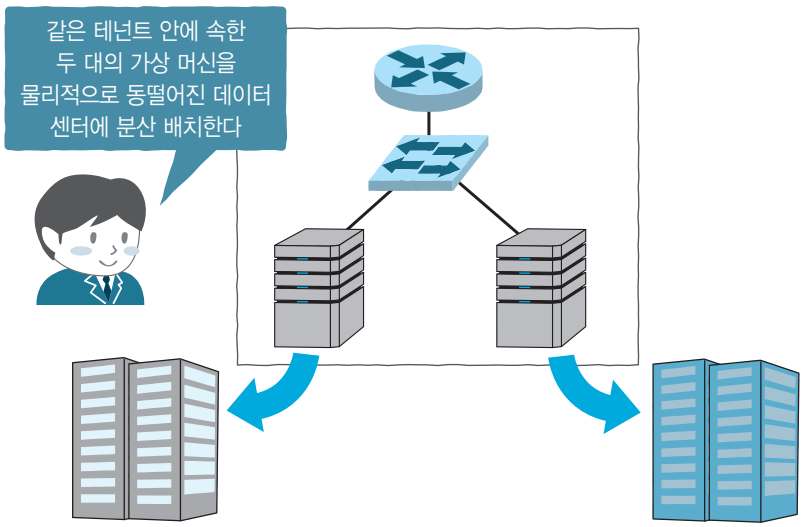
1.3 클라우드 컴퓨팅의 활용

클라우드 컴퓨팅, 특히 IaaS형 클라우드 서비스에 대해 간단히 살펴 보았습니다. 클라우드 환경에서는 시스템을 구성하는 컴포넌트들이 추상화되어 있어서 사용자가 원하는 기능을 원하는 시기에 바로 사용할 수 있습니다. 또한 시스템을 구축하는 과정도 표준화되어 있어서 API를 사용한 자동화도 가능하다는 것을 알게 되었습니다. 이러한 장점들은 시스템을 빠르게 구축하는 데 큰 도움이 됩니다. 클라우드 서비스의 본질을 제대로 이해하고 자유자재로 사용할 수 있게 된다면 시스템 구축의 신세계를 경험하게 될 것입니다.

클라우드 API 하나하나는 기본적으로 클라우드의 단위 기능들과 연결됩니다. 클라우드 API의 목록만 훑어 보더라도 그 클라우드 서비스가 제공하는 기능들을 파악할 수 있어 API 목록[7]이 곧 클라우드의 기능 목록이라고 말할 수 있습니다([그림 1.17] 참고). 그리고 웹 콘솔에서는 모든 API를 사용하는 것이 아닙니다. 그래서 지금까지 주로 웹 콘솔로만 클라우드를 제어한 사용자라면 꼭 한번 API를 살펴 보길 권합니다. 시간을 가지고 API 문서를 보다 보면 이전에는 미처 몰랐던 새로운 기능을 발견할지도 모릅니다.

▲ 그림 1.17 API 목록은 곧 클라우드의 기능 목록

7 역자 주 : AWS는 각 서비스별 문서에 API 레퍼런스가 포함되어 있습니다.
• https://aws.amazon.com/ko/documentation/
오픈스택(OpenStack)은 API 레퍼런스 안에 각 서비스별 문서가 포함되어 있습니다.
• http://developer.openstack.org/ko_KR/api-guide/quick-start/index.html)



▲ 그림 1.18 물리적인 배치를 의식한 시스템 구축 방법

앞서 살펴본 것처럼 클라우드의 장점은 시스템의 물리적인 환경을 의식하지 않고도 인프라 자원을 사용할 수 있다는 점입니다. 여기에 실제 물리적 환경에 대한 이해까지 갖추고 있다면 클라우드 서비스를 더욱 효과적으로 이용할 수 있습니다. 예를 들어 데이터 센터가 재난이나 화재로 피해를 입어 제 기능을 하지 못하는 경우에도 지속적인 서비스를 제공하려면 DRDisaster Recovery을 구축해야 합니다. 즉, 지리적으로 일정 거리 떨어진 곳의 서로 다른 데이터 센터에 가상 머신을 분산 배치하여 위험에 대비할 수 있어야 합니다. 이 말은 클라우드 환경이라고 하더라도 물리적인 환경에 대해 전혀 몰라도 되는 것은 아니라는 말입니다([그림 1.18] 참고).

오픈스택OpenStack은 오픈소스이기 때문에 내부 구조가 모두 공개되어 있습니다. 그래서 분석해보겠다는 의지만 있다면 각종 컴포넌트들이 어떻게 동작하는지, 내부 구조가 어떻게 생겼는지 확인하는 것은 결코 불가능한 일이 아닙니다. '웹 콘솔이나 명령어로 가상 머신을 기동하면 클라우드 내부에서는 무슨 일이 벌어질까?'와 같이 내부 동작을 추측하고 분석하면 클라우드를 보다 더 잘 활용할 수 있게 될 것입니다.

이어지는 2장부터는 대표적인 클라우드 환경으로 AWSAmazon Web Services와 오픈스택을 예로 들어 클라우드의 주요 구성 요소들을 소개합니다. 3장에서는 각 컴포넌트들의 기능과 그것들을 제어하는 API에 대해 살펴 보는데 API의 내부 구조에 대해서는 오픈소스인 오픈스택으로 설명합니다. 그리고 그 이후부터는 '추상화'와 '표준화'라는 클라우드 서비스 본연의 특징을 이해한 후, 클라우드 컴퓨팅의 장점을 극대화하는 활용 방법에 대해 배우겠습니다.