



클라우드 컴퓨팅 유즈케이스 백서

제 4 판 (한국어판)

*Cloud Computing
Use Cases Discussion Group*

클라우드 컴퓨팅 유즈케이스

A white paper produced by the
Cloud Computing Use Cases Discussion Group

제 4.0 판

2010 년 7 월 2 일

기여자: Miha Ahronovitz, Dustin Amrhein, Patrick Anderson, Andrew de Andrade, Joe Armstrong, Ezhil Arasan B, James Bartlett, Richard Bruklis, Ken Cameron, Mark Carlson, Reuven Cohen, Tim M. Crawford, Vikas Deolaliker, Pete Downing, Andrew Easton, Rodrigo Flores, Gaston Fourcade, Thomas Freund, Tom Hanan, Valery Herrington, Babak Hosseinzadeh, Steve Hughes, William Jay Huie, Nguyen Quang Hung, Pam Isom, Shobha Rani J, Sam Johnston, Ravi Kulkarni, Anil Kunjunny, Edmond Lau, Thomas Lukasik, Bob Marcus, Gary Mazzaferro, Craig McClanahan, Meredith Medley, Walt Melo, Andres Monroy-Hernandez, Ayman Nassar, Dirk Nicol, Lisa Noon, Santosh Padhy, Gilad Parann-Nissany, Greg Pfister, Thomas Plunkett, Ling Qian, Balu Ramachandran, Jason Reed, German Retana, Bhaskar Prasad Rimal, Dave Russell, Matt F. Rutkowski, Clark Sanford, Krishna Sankar, Alfonso Olias Sanz, Mark B. Sigler, Wil Sinclair, Erik Sliman, Patrick Stingley, Phillip Straton, Robert Syputa, Robert J. Taylor, Doug Tidwell, Kris Walker, Kurt Williams, John M Willis, Yutaka Sasaki, Michael Vesace, Eric Windisch, Pavan Yara and Fred Zappert.

이 문서에 대한 의견은 <http://cloudusecase.org>에 있는 토론 게시판을 이용하여 주시기 바랍니다.

본 문서는 동국대학교 (서울 캠퍼스) 정보통신공학과 김양우 교수(네트워크 컴퓨팅 시스템 구조 연구실)에 의해 번역된 것으로 번역에 관한 문의와 의견은 ywkim@dongguk.edu로 해주시기 바랍니다.



이 문서에는 [Creative Commons Attribution Share Alike 3.0 Unported License](https://creativecommons.org/licenses/by-nc-sa/3.0/) 저작권이 적용됩니다.

목 차

1 서론	4
2 정의와 분류	6
2.1 클라우드 컴퓨팅 개념의 정의	6
2.2 분류	10
2.3 표준과 분류 간의 관계	13
2.4 어플리케이션 프로그래밍 인터페이스(APIs)	15
3 유즈케이스 시나리오	18
3.1 최종 사용자와 클라우드	19
3.2 기업과 클라우드와 최종 사용자	20
3.3 기업과 클라우드	23
3.4 기업과 클라우드와 기업	24
3.5 프라이빗 클라우드	25
3.6 클라우드 벤더 교체	26
3.7 하이브리드 클라우드	28
3.8 상호참조: 요구사항들과 유즈케이스	31
4 고객 시나리오	32
4.1 고객 시나리오: 클라우드에서의 급여처리	32
4.2 고객 시나리오: 클라우드에서의 프로젝트 관리	33
4.3 고객 시나리오: 클라우드에서의 중앙 정부 서비스	35
4.4 고객 시나리오: 하이브리드 클라우드에서의 지방 정부 서비스	36
4.5 고객 시나리오: 천문 데이터 처리	37
5 개발자 요구사항	39
6 보안 시나리오	41
6.1 규정	41
6.2 보안 제어	42
6.3 보안 연합 패턴	44
7 보안 유즈케이스 시나리오	46
7.1 클라우드에서의 컴퓨팅 파워	46
7.2 클라우드 기반 개발과 테스트	47
7.3 클라우드에서의 저장공간	48
7.4 상호참조: 보안 제어와 고객 시나리오	50
7.5 상호참조: 보안 연합 패턴과 고객 시나리오	50
8 서비스 수준 협약(SLAs)	51
8.1 서비스 수준 협약이란 무엇인가?	51
8.2 서비스 수준 목표	52
8.3 서비스 수준 관리	53
8.4 SLA에 대한 고려사항들	53
8.5 SLA 요구사항	56
8.6 신뢰성에 대한 언급	59

8.7 상호 참조: SLA 요구사항들과 클라우드 서비스 모델 60
8.8 상호 참조: SLA 요구사항들과 유즈케이스 시나리오 61
9 결론과 추천 62
변경 내역 요약 64

버전 4에 대해: 버전 4에 신규로 추가된 부분은 섹션 8 임, 서비스 수준 협약(SLAs).
상세한 사항은 페이지 64의 변경 내용 요약을 보기 바람.

1 서론

클라우드 컴퓨팅 유즈케이스 그룹은 클라우드 컴퓨팅을 위한 유즈케이스 시나리오를 정의하기 위해 클라우드 소비자와 클라우드 벤더들을 모두 모았다. 유즈케이스 시나리오들은 클라우드 컴퓨팅의 성능과 경제적 이익을 입증하고 최대한 광범위한 소비자들의 요구에 근거하여 작성되었다.

본 백서의 목표는 클라우드 환경에서 상호운영성, 통합의 편의성, 그리고 이식성을 보장하기 위해 표준화를 되어야 하는 요구사항들과 기능을 강조하기 위한 것이다. 본 문서에서 기술된 모든 유즈케이스들은 비공개된 사유 기술 없이도 구현이 가능하여야 한다. 클라우드 컴퓨팅은 반드시 벤더의 종속을 최소화 하고 사용자 선택을 증가시키는 개방형 환경으로 발전해야 한다.

이 유즈케이스들은:

- 상호운영성과 표준에 대한 논의를 위해 현실적이고 사용자 경험에 기반을 둔 내용을 제공하고,
- 기존의 표준들이 어디에 사용되어야 하는지를 명확히 하며,
- 개방형 클라우드 컴퓨팅의 중요성에 대해 산업계의 관심을 집중시키며,
- 어디에 표준화 작업이 필요한지를 명확히 한다. 만약 특정 유즈케이스가 현재 구현될 수 없거나, 독점적인 API와 제품들로만 만들어질 수 있다면 산업계는 그 유즈케이스를 가능하게 할 표준을 정의할 필요가 있다.

어떤 공통의 작업을 명확히 묘사하고 그것을 이루기 위한 어려움을 기술하는 유즈케이스는 표준을 정립하기 위한 모든 노력들 중에서 가장 명분 있는 작업이다.

오픈 클라우드 선언(Open Cloud Manifesto, opencloudmanifesto.org)은 클라우드 컴퓨팅의 개방성 유지를 위한 원칙들에 대한 성명서이다. 발표한지 2달 내에 250개 기관들이 지지자로 가입했다. 이 그룹의 활동은 오픈 클라우드 선언의 6가지 원칙에 따라 진행되었다.

- 클라우드 제공자들은 개방된 협조와 적절한 표준들의 활용을 통해 클라우드 도입에 따른 난제들을 도출하기 위해 함께 일해야 한다.
- 클라우드 제공자들은 해당되는 모든 경우에 현재 존재하는 표준을 채택하고 사용해야 한다. IT 산업계는 현존하는 표준과 표준기관에 이미 많은 투자를 해왔고, 이에 중복된 투자나 재개발을 할 필요는 없다.
- 새로운 표준이 필요할 때(또는 기존의 표준을 교정할 때), 우리는 너무 많은 표준을

만드는 것을 피하기 위해 분별력 있고 실용적이어야 한다. 우리는 표준이 기술혁신을 장려해야지 방해하지 않도록 보장해야 한다.

- 오픈 클라우드 주위의 어떠한 공동체 노력도 클라우드 제공자의 기술적인 필요가 아닌 고객의 필요에 의해 추진되어야 하며, 실제 고객의 요구사항을 상대로 시험되고 검증되어야 한다.
- 클라우드 컴퓨팅 표준화 기관과 활동 그룹 그리고 공동체들은 이들의 노력이 반감되거나 중복을 피하기 위해 함께 일하고 협조를 유지해야 한다.
- 클라우드 제공자들은 고객들을 그들의 특정 플랫폼으로 종속시키거나 고객들의 제공자 선택권 제한을 위해 그들의 시장적 지위를 사용해서는 안 된다.

이 문서는 이런 원칙들을 현실로 만들기 위한 지속적인 노력의 일부분이다.

2 정의와 분류

이번 장에서 설명하는 정의와 분류는 클라우드 컴퓨팅 개념의 개요를 제공하기 위하여 포함되었다. 그러나 이 백서의 초점은 클라우드 컴퓨팅 자체를 정의하는 것이 아니라 실생활의 실제 어플리케이션들과 요구사항들을 기반으로 한 클라우드 시나리오들과 유즈케이스들을 정의하는 것이다. 우리의 목표는 어떻게 이러한 시나리오들이 정의되거나 분류될 것 인가에 관계없이 명확하고 흥미롭고 유용한 유즈케이스 시나리오들을 제공하는 것이다.

2.1 클라우드 컴퓨팅 개념의 정의

클라우드 컴퓨팅: 클라우드 컴퓨팅은 언제 어디서나, 편하게, 구성이 가능한 컴퓨팅 자원 (e.g. networks, servers, storage, application, service)들의 공유된 풀에 온디맨드 네트워크 접근이 가능한 모델이다. 이 컴퓨팅 자원들은 최소한의 관리 노력 혹은 최소한의 서비스 제공자와의 상호 작용을 통해 원할 때 신속히 제공되고 회수되어야 한다. (이 정의는 U.S. Government's National Institute of Standards and Technology¹ 가 발행한 NIST Working Definition of Cloud Computing의 최종 버전에서 따옴.)

2.1.1 서비스 전달 모델 (Delivery Models)

클라우드 컴퓨팅에 대한 NIST 정의는 3가지의 서비스 전달(Delivery) 모델들을 정의한다:

- 소프트웨어형 서비스 **Software as a Service(SaaS):** 소비자는 단지 어플리케이션만을 사용하고 그 것이 실행되는 운영체제, 하드웨어, 또는 네트워크 인프라는 제어하지 않는다.
- 플랫폼형 서비스 **Platform as a Service(PaaS):** 소비자는 그들의 어플리케이션들을 위해 제공되는 호스팅 환경을 사용한다. 이 소비자는 이 환경에서 실행되는 어플리케이션들(그리고 그 호스팅 환경의 일부 기능)은 제어하지만, 어플리케이션들을 실행시키고 있는 운영체제, 하드웨어, 또는 네트워크 인프라는 제어하지 않는다. 이 플랫폼은 전형적인 어플리케이션 프레임워크이다.
- 인프라형 서비스 **Infrastructure as a Service(IaaS):** 여기의 소비자는 프로세싱 파워, 스토리지, 네트워크 또는 미들웨어와 같은 기본적인 컴퓨팅 자원을 사용한다. 이 소비자는 운영체제, 스토리지, 배치된 어플리케이션들, 그리고 방화벽과 부하분산기와 같은 네트워크 컴포넌트들을 제어할 수 있다.

¹ 본 NIST 클라우드 컴퓨팅 페이지는 <http://csrc.nist.gov/groups/SNS/cloud-computing/>에서 찾을 수 있을 수 있으며, 본 백서에서 논의된 기본적 특성, 딜리버리 모델, 그리고 운용 모델들은 2009년 8월 19일 날짜의 버전 15에 기반을 두고 있음.

2.1.2 운용 모델 (Deployment Model)

NIST 정의는 4가지의 운용모델을 정의 한다:

- **퍼블릭 (Public) 클라우드:** 퍼블릭 클라우드 서비스는 인터넷을 통해 제 3의 서비스 제공자로부터 클라이언트들에게 제공된다는 특징을 갖는다. ‘공공(public)’이라는 용어는 그것이 무료일 수도 있고 사용료가 저렴할 수도 있지만 항상 무료를 의미하는 것은 아니다. 퍼블릭 클라우드라고 해서 사용자의 데이터가 일반에 공개된다는 것을 의미하는 것은 아니다; 퍼블릭 클라우드 벤더는 일반적으로 그들의 사용자들을 위해 접근 제어 메커니즘을 제공한다. 퍼블릭 클라우드는 여러 솔루션들을 서비스하기 위한 유연하고 경제적인 수단을 제공한다.
- **프라이빗 (Private) 클라우드:** 프라이빗 클라우드는 탄성적이고 서비스 기반이라는 등의 퍼블릭 클라우드 환경의 장점들을 많이 제공한다. 퍼블릭 클라우드와 프라이빗 클라우드의 차이점은, 프라이빗 클라우드 서비스에서는 퍼블릭 클라우드 서비스들을 사용하는데 따를 수 있는 법적 요구사항, 보안 노출, 그리고 네트워크 용량과 같은 제약들이 없이 한 기관 내에서 데이터와 프로세스를 관리한다는 점이다. 또한 프라이빗 클라우드 서비스는 제공자와 사용자에게 클라우드 인프라에 대한 확장된 제어권을 제공하여 보안을 향상시킨다. 이는 사용자 접근과 이용된 네트워크를 지정하고 제한할 수 있기 때문이다.²
- **커뮤니티 (Community) 클라우드:** 커뮤니티 클라우드는 특정한 보안적 요구사항이나 공통의 임무와 같이 동일한 관심을 가진 기관들로 구성된 그룹에 의해 운영되고 사용된다. 그룹의 구성원들은 클라우드의 데이터와 어플리케이션을 접근하고 공유할 수 있다.
- **하이브리드 (Hybrid) 클라우드:** 하이브리드 클라우드는 상호 운영되는 퍼블릭과 프라이빗 클라우드의 조합이다. 이 모델에서 사용자들은 비즈니스에 중요하지 않은 정보와 처리는 퍼블릭 클라우드로 외부위탁을 하고, 중요한 서비스와 데이터는 그들이 직접 운영한다.³

2.1.3 클라우드 컴퓨팅 본질적 특성 (Essential Characteristics)

NIST 정의에서는 클라우드 컴퓨팅의 5가지의 본질적 특성을 기술한다.

- **빠른 탄성(Rapid Elasticity):** 여기서 탄성이란 필요에 따라 자원의 양을 증가 또는 감소시킬 수 있는 능력을 의미한다. 사용자에게 클라우드는 무한하게 보이고 사용자는

² 프라이빗 클라우드는 제3자에 의해 관리되고 외부에 존재할 수 있다. 굳이 이 클라우드를 사용하는 기관에 의해서 관리되고 호스팅 되어야 할 필요는 없다.

³ 하이브리드 클라우드는 커뮤니티 클라우드에서 사용된 기술의 슈퍼셋이다. 이러한 이유로 이 둘 개의 운용모델들의 요구사항은 섹션3의 “하이브리드 클라우드”에서 같이 논의된다.

그들의 필요에 따라 적거나 많은 컴퓨팅 파워를 살 수 있다. 이것은 NIST가 정의한 클라우드 컴퓨팅의 필수적인 특성 중의 하나이다.

- **측정된 서비스(Measured Service):** 측정된 서비스에서는 클라우드 서비스의 여러 요소들이 클라우드 제공자에 의해 모니터 되고 관리된다. 이것은 과금 정책, 접근제어, 자원 최적화, 사용량 예측 등을 위해 필수적이다.
- **요청기반 셀프서비스(On-Demand Self-Service):** 클라우드 컴퓨팅의 요청(On-Demand)과 셀프서비스(Self-Service)는 소비자가 클라우드 제공자와의 어떤 인간적인 상호작용(human interaction)없이 클라우드 서비스를 필요한 만큼 이용할 수 있다는 것을 의미한다.
- **유비쿼터스 네트워크 접속(Ubiquitous Network Access):** 유비쿼터스 네트워크 접속은 클라우드 제공자의 서비스가 클라이언트의 성능에 상관없이(고성능, 저성능 클라이언트 모두) 표준 절차에 의해 접근 가능하며 네트워크상에서 이용할 수 있다는 것을 의미한다.⁴
- **자원 풀(Resource Pooling):** 자원 풀은 클라우드 제공자가 그들의 소비자들을 다중 소유 모델(multi-tenant model)로 서비스할 수 있게 한다. 물리적인 그리고 가상 자원들은 소비자의 요청에 따라 할당되고 재활당된다. 자원의 위치는 독립적이고 고객은 제공되는 자원의 정확한 위치에 대한 인지도나 제어가 불가능하지만 높은 레벨의 추상적인 위치는 지정할 수도 있다. (나라, 주, 데이터센터 등)⁵

2.1.4 기타 용어

상호운영성(Interoperability): 상호운영성은 시스템들의 통신을 하기 위한 능력과 연관된다. 전달된 정보는 정보를 받은 시스템에 의해 이해되어야 함을 요구한다. 클라우드 컴퓨팅에서, 이것은 제공자들 간의 차이에 상관없이 한 개 이상의 클라우드 제공자에서 동시에 작동하는 코드를 작성할 수 있는 능력을 의미한다.⁶

이식성(Portability): 이식성은 한 특정 환경을 위해 작성된 컴포넌트나 시스템을 다른 환경에서 실행할 수 있는 능력이다. 클라우드 컴퓨팅에서, 이것은 소프트웨어와 하드웨어 환경을 포함한다. (물리적인 그리고 가상 환경 모두)

⁴ 이는 꼭 인터넷을 통한 접근을 의미하는 것은 아니다. 정의에 따르면, 프라이빗 클라우드는 방화벽 내부에서만 접근할 수 있다. 네트워크 유형에 관계없이 클라우드에 대한 접근은 특정 유형의 클라이언트로 한정되지는 않는다.

⁵ 많은 경우에 정보보호 법안들과 규정들은 클라우드 제공자의 자원들이 특정 지역에 있을 것을 요구한다. 클라우드 제공자와 소비자는 이 규정들에 준수하도록 협력하여야 한다.

⁶ 상호운영성과 이식성, 그리고 통합의 정의는 아래의 작업 내용을 기준으로 했다.
http://www.testingstandards.co.uk/interop_et_al.htm.

통합(Integration): 통합은 컴포넌트나 시스템을 전체 시스템으로 결합하는 과정이다. 클라우드 기반의 컴포넌트나 시스템 사이의 통합은 다중소유, 연계(federation), 정부규약과 같은 이슈에 의해 복잡해 질 수 있다.

서비스 수준 협약(Service Level Agreement, SLA): 서비스 수준 협약(SLA)은 소비자의 요구와 제공자의 의무를 기술한 제공자와 소비자 사이의 규약이다. 일반적인 SLA는 가동시간, 사적 기밀성, 보안, 백업 절차 등을 포함한다.

연합(Federation): 연합은 여러 시스템간의 데이터 혹은 식별자(identity)를 결합하는 것이다. 연합은 클라우드 브로커나 클라우드 제공자에 의해 제공될 수 있다.

브로커(Broker): 브로커는 자신의 클라우드 자원을 직접 가지지는 않지만 소비자의 요구에 따른 SLA를 기반으로 소비자와 제공자를 연결한다. 소비자는 브로커가 자원을 제어하지 않는다는 것을 모른다.

다중 소유(Multi-Tenancy): 다중 소유는 동일한 물리적 하드웨어에 여러 기업의 다양한 데이터, 시스템, 어플리케이션들을 수용할 때의 특징(property)이다. 다중소유는 대부분의 클라우드 시스템에 흔히 적용된다.

클라우드 버스팅(Cloud bursting): 클라우드 버스팅은 프라이빗 클라우드의 필요 요청 시 추가적인 자원을 할당해주는 하이브리드 클라우드에 의해 사용되는 기술이다. 만약 프라이빗 클라우드가 그들의 작업을 처리하기 위한 충분한 처리 용량을 갖고 있다면, 하이브리드 클라우드는 사용되지 않는다. 그러나 작업량이 프라이빗 클라우드의 처리용량을 초과할 때는 하이브리드 클라우드가 자동적으로 프라이빗 클라우드에 추가적인 자원을 할당한다.

정책(Policy): 정책은 운영절차를 위한 일반적인 용어이다. 예를 들어, 어떤 한 보안 정책은 특정 클라우드 서비스에 대한 모든 요청을 암호화하라고 명시할 수 있다.

통제(Governance): 통제는 정책이 적용되는 것을 보장하기 위한 절차와 제어를 의미한다.

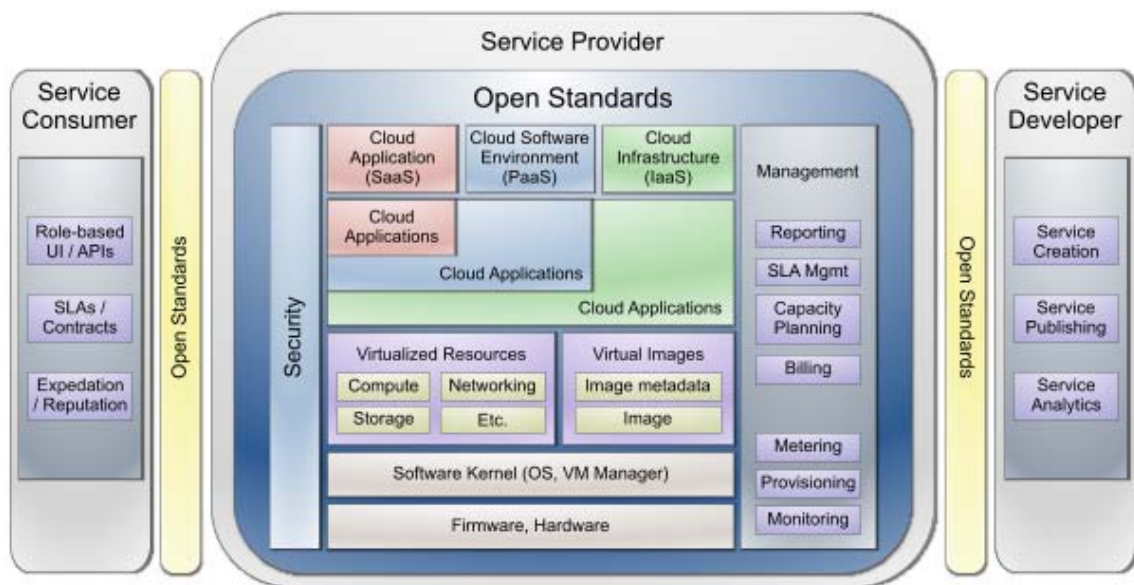
가상 머신(Virtual Machine, VM): 실행될 때 사용자에게 실제 머신처럼 보이는 (일반적으로 이미지라 불리는) 파일이다. 인프라형 서비스(IaaS)는 필요에 따라 실행되거나 멈출 수 있는 VM 이미지 형태로 종종 제공된다. 실행 중에 가상 머신에게 적용된 변경사항은 보관을 위하여 디스크에 저장될 수 있다.

애플리케이션 프로그래밍 인터페이스(Application Programming Interface, API): API는 특정 유형의 시스템과 동작하는 코드를 어떻게 작성할 것인지 개발자에게 알려주는 규약이다. API는 시스템에 의해 지원되는 기능의 문법을 기술한다. 각각의 기능에 대해, API는 시스템에 보내지는 정보, 시스템에서 돌아오는 정보, 그리고 발생할 수 있는 각종 에러 조건들에 대한 정보를 기술한다.

- API는 특정 프로그래밍 언어나 WSDL(Web Services Description Language), 또는 IDL(Interface Definition Language)과 같은 보다 일반적인 형식으로 정의된다. REST 명세서는 컴퓨터가 해석할 수 있는 언어를 갖지 않음에도 불구하고 API를 정의한다.
- API는 (HTTP 같은) 프로토콜의 세부사항들과 데이터 형식 항목을 포함할 수도 있다 (JSON 또는 XML 스키마와 같은).
- API는 데이터와 기능들의 의미를 이해하기 위한 인적 지능을 요구한다. 머신은 두 개의 정수를 변수로 갖는 함수 x 를 발견할 수 있지만, 인간인 개발자가 무수히 많은 두 개의 정수 중 어떤 것인지를 알아내야 한다.

2.2 분류 (taxonomy)

이 그림은 클라우드 컴퓨팅의 분류를 정의한 것이다:



이 다이어그램에서, 서비스 소비자(Service Consumer)는 클라우드를 통해 제공되는 서비스를 사용하고, 서비스 제공자(Service Provider)는 클라우드 인프라를 관리하고, 서비스 개발자(Service Developer)는 클라우드 서비스 자체를 만든다. (그들의 역할 사이의 상호작용을 위한 개방형 표준들이 필요하다.) 각각의 역할은 계속되는 섹션에서 자세히 기술된다.

2.2.1 서비스 소비자(Service Consumer)

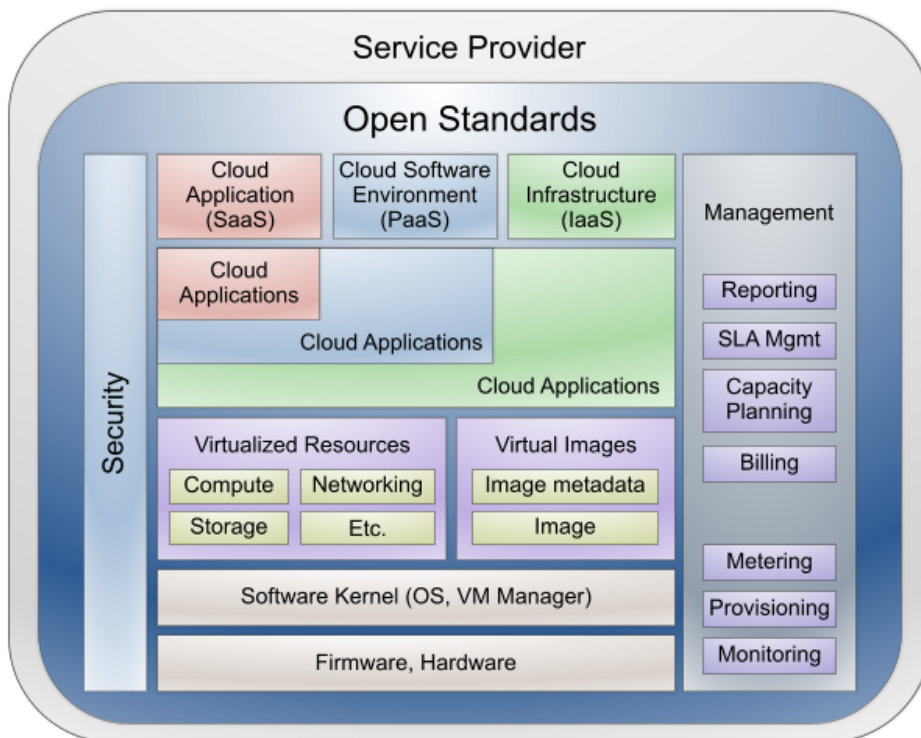
서비스 소비자는 SaaS, PaaS, 또는 IaaS 서비스를 사용하는 실질적인 최종 사용자나 기업이다.

서비스의 유형과 그들의 역할에 따라 그 소비자는 서로 다른 사용자 인터페이스와 프로그래밍 인터페이스를 가지고 작업을 한다. 어떤 사용자 인터페이스는 또 다른 어플리케이션처럼 보이기에 사용자는 그 어플리케이션을 사용하는 동안에 클라우드 컴퓨팅에 대해 알 필요가 없다. 다른 사용자 인터페이스는 가상 머신을 시작시키고 멈추거나 스토리지를 관리하는 것과 같은 관리자 기능들을 제공한다. 어플리케이션 코드를 작성하는 소비자는 그들이 작성하는 어플리케이션에 따라 서로 다른 프로그래밍 인터페이스들을 사용한다.

소비자는 또한 SLA와 규약에 따라 일한다. 일반적으로 SLA와 규약들은 소비자와 제공자 사이에서 사람의 조정에 의해 결정된다. 소비자의 기대치와 제공자의 평판은 이런 협상의 중요한 부분이다.



2.2.2 서비스 제공자(Service Provider)



서비스 제공자는 소비자에게 서비스를 제공한다. 제공자의 실제적인 일은 서비스 종류에 따라 다르다.

- SaaS 서비스에서는 제공자가 소프트웨어를 설치하고 관리하고 유지한다. 제공자는 이런 소프트웨어가 실행되는 물리적 인프라를 소유 할 필요는 없다. 소비자는 인프라에 접속할 권한이 없고, 단지 그들은 오직 어플리케이션에만 접근할 수 있다.
- PaaS 서비스에서 제공자는 특정 유형의 어플리케이션용 프레임워크인 플랫폼을 위한 클라우드 인프라를 관리한다. 소비자의 어플리케이션은 플랫폼 하단의 인프라에 접근할 수 없다.
- IaaS 서비스에서 제공자는 스토리지, 데이터베이스, 메시지 큐 혹은 미들웨어, 또는 가상 머신을 위한 호스팅 환경을 유지한다. 소비자는 이러한 서비스들이 각각 디스크 드라이브, 데이터베이스, 메시지 큐, 또는 머신들인 것처럼 사용하지만 이들을 호스팅하고 있는 인프라에는 접근할 수 없다.

서비스 제공자 그림에서, 가장 하위 레벨 단위는 모든 것에 기초가 되는 펌웨어와 하드웨어이다. 그 위에는 운영체제 또는 가상 머신 관리자인 소프트웨어 커널이 존재하고 이들은 클라우드 하부단의 인프라를 관리한다. 가상화된 자원과 이미지는 프로세싱 파워, 스토리지, 미들웨어와 같은 기본적인 클라우드 컴퓨팅 서비스들을 포함한다. 가상 머신 관리자에 의해 관리되는 가상 이미지들은 이미지 자체와 그들을 관리하기 위한 메타데이터를 포함한다.

서비스 제공자의 동작 중 가장 중요한 부분은 관리 레이어이다. 하위레벨에서는 누가 서비스를 사용하는지와 얼마만큼의 규모로 사용하는지를 측정하는 메터링, 어떤 방법으로 자원을 소비자에게 할당할 것인지를 결정하는 프로비저닝, 시스템과 자원의 상태를 계속 모니터링 하는 등의 관리 기능이 요구된다.

상위 레벨에서의 관리 기능에는 경비를 회수하기 위한 요금청구, 소비자의 요청을 만족시키기 위한 용량계획, 제공자와 소비자가 동의한 서비스 항목들을 충족시키기 위한 SLA 관리, 그리고 관리자에게 보고하는 기능들이 있다.

보안은 서비스 제공자 제공기능의 모든 측면에 적용된다. (다양한 레벨의 보안 요구사항은 본 문서의 범위에서 제외된다.) 오픈 표준은 서비스 제공자의 제공기능에도 역시 적용된다. 잘 작성된 표준들은 제공자 내에서의 기능들과 제공자들 사이의 상호운영성을 간소화시킨다.

2.2.3 서비스 개발자(Service Developer)

서비스 개발자는 클라우드 서비스를 만들고 배치하고 모니터링 한다. 이러한 클라우드 서비스는 일반적으로 SaaS 모델을 통해 최종 사용자에게 직접 전달되는 비즈니스 관련

(line-of-business) 어플리케이션들이다. IaaS와 PaaS 레벨에서 작성된 어플리케이션들도 추후 SaaS 개발자나 클라우드 제공자들에 의해 사용된다.

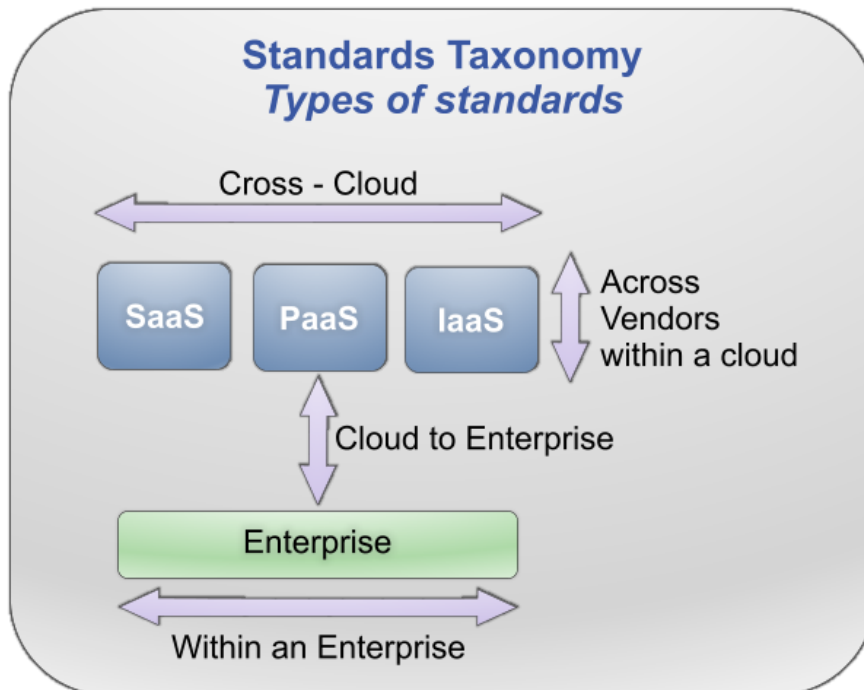
서비스 개발 환경은 다양하다. 만약 개발자들이 SaaS 어플리케이션을 개발한다면 그들은 클라우드 제공자에 의해 호스팅되는 환경을 위한 어플리케이션을 개발하고 있을 가능성이 높다. 이런 경우에 서비스를 발행한다는 것은 그 서비스를 클라우드 제공자의 인프라에 배치하는 것이다.

서비스를 개발하는 과정에는 소비자에게 서비스가 제공되기 전에 서비스를 테스트하기 위한 원격 디버깅이 포함된다. 일단 서비스가 오픈된 후 서비스 분석(Analytics)은 개발자들이 그들의 서비스의 성능을 감시하고 필요에 따라 변경을 할 수 있게 한다.



2.3 표준과 분류 간의 관계

표준들이 클라우드 유즈케이스들에게 영향을 끼치는 경우는 네 가지가 있다. 표준은, 각 유형의 한 클라우드 서비스 내에서, 서로 다른 유형의 클라우드 서비스들 간에, 클라우드와 엔터프라이즈 간에, 그리고 엔터프라이즈 내의 프라이빗 클라우드 사이 등의 4분야에 영향을 미친다.



2.3.1 클라우드 서비스 유형들 간의 표준(Standards Across Cloud Service Type)

클라우드 컴퓨팅이 더 보편화되면, 어플리케이션은 여러 유형의 클라우드 서비스들을 사용할 것이다. 어떤 어플리케이션은 클라우드 스토리지 서비스, 클라우드 메시지 큐, 그리고 클라우드에서 실행되고 있는 가상 머신 관리(실행, 정지, 모니터링)를 사용할 수도 있다. 이러한 여러 다른 서비스들이 어떻게 함께 동작할 것인지를 정의하는 표준은 가치를 가진다.

2.3.2 한 클라우드 서비스 유형에서의 표준(Standards Within Cloud Service Type)

각각의 클라우드 서비스 타입(IaaS, PaaS, SaaS) 내에서 오픈 표준은 특정 제조업체(vender)에 종속되는 것을 막을 수 있다.

인프라형 서비스(IaaS)에서, 클라우드 데이터베이스와 동작하는 표준화된 API 셋은 여러 제조업체로 부터의 데이터들과 함께 동작하는 어플리케이션들을 가능하게 한다. 이러한 공통된 API는 사용자들이 많은 변경 작업 없이 클라우드 데이터베이스 제공자를 교체할 수 있는 자유를 제공하는 것과 더불어 기존의 어플리케이션에 새로운 데이터 자원 통합을 용이하게 한다. 스토리지, 메시지 큐, 혹은 맵리듀스(MapReduce)와 같은 다른 클라우드 인프라스트럭처 서비스들을 위한 공통된 API도 데이터와 데이터 교환을 위한 공통된 포맷을 지원하는 경우처럼 유사한 장점을 제공할 것이다. 가상 머신의 경우, 공통된 가상 머신 포맷은 매우 중요하다. 사용자는 한 클라우드 제공자가 생성하고 배치한 가상 머신을 또 다른 클라우드 제공자에게 변경 없이 배치할 수 있어야 한다.

플랫폼형 서비스(PaaS)의 경우, 클라우드에서 제공되는 많은 플랫폼은 어플리케이션 프레임워크이다. 이런 프레임워크는 일반적으로 소비자 인터페이스, 스토리지, 그리고 데이터베이스와 같은 공통적인 서비스를 제공하지만, 그들은 이런 프레임워크의 API를 통해서만 접근할 수 있다

소프트웨어형 서비스(SaaS)에서, 오픈 표준은 어플리케이션 레벨에서 적용된다. 이 경우, 클라우드와 관련된 표준은 매우 적기 때문에 이런 표준은 이 문서의 범위를 벗어난다. 예를 들어 클라우드 기반의 워드 프로세싱 어플리케이션은 문서의 이식성을 위한 표준을 지원해야 하는데, 워드 프로세싱 어플리케이션을 지원하는 표준에 대한 요구사항은 그 어플리케이션이 클라우드 내에서의 실행 여부와는 관련이 없다.

2.3.3. 클라우드와 엔터프라이즈 간의 표준(Standards Between the Cloud and the Enterprise)

클라우드 컴퓨팅이 계속 발전을 하더라도 Java EE와 같은 엔터프라이즈 아키텍처는 사라지지 않는다. 엔터프라이즈 어플리케이션이 어떻게 클라우드 데이터베이스 혹은 클라우드 메시지 큐와 같은 자원들과 통신을 하는가에 대한 표준은 이런 어플리케이션들이 수정 없이 또는 약간의 수정만으로도 클라우드 서비스들을 사용할 수 있도록 한다. 클라우드 컴퓨팅을

현존하는 아키텍처들 그리고 개발 패러다임들과 어떻게 통합할 것인가는 본 그룹의 큰 과제가 될 것이다.

2.3.4 엔터프라이즈 내부의 표준(Standards Within an Enterprise)

기업(enterprise) 내부의 표준은 상호운영성(Interoperability), 감사(auditability), 보안, 그리고 관리(management) 등의 요구사항에 의해 결정될 것이고, 추후 엔터프라이즈와 클라우드 간의 표준으로 발전할 것이다. 기업은 프라이빗, 퍼블릭, 그리고 하이브리드 클라우드의 일부 조합들과 상호작용할 것이다.

2.4 어플리케이션 프로그래밍 인터페이스 (APIs)

클라우드 컴퓨팅 솔루션을 만들기 위한 가장 기본적인 방법은 클라우드 제공자에 의해 제공되는 API를 통하는 것이다. 클라우드 API는 4개의 레벨에서 동작하고, 5개의 기본 유형으로 분류된다.

2.4.1 API들의 레벨(Levels of APIs)

4개 레벨의 API들이 존재하는데, 각 레벨에서 개발자는 각기 다른 작업들과 데이터 구조에 집중해야 한다.

Level 1 - The Wire: 이 레벨에서, 개발자는 특정 요청의 와이어 포맷으로 직접 코드를 작성한다. 만약 REST기반의 서비스라면 개발자는 적절한 HTTP 헤더를 개발하며, 요청에 따라 페이로드(payload)를 만들고, 서비스에게 HTTP 연결을 열어준다. 그러면 그 REST 서비스는 HTTP 응답코드를 갖는 데이터를 회신한다. 많은 REST서비스의 단순 특성 때문에, 이 레벨에서 코드 작성 시 상대적으로 효율적일 수 있다.

만약 어떤 서비스가 SOAP 기반이라면, 그 개발자는 SOAP 엔벨롭(envelope)을 만들고 적절한 SOAP 헤더를 추가한 후, 데이터 페이로드로 SOAP의 본문(body)을 채운다. 그러면 이 SOAP서비스는 요청의 결과를 담은 SOAP 엔벨롭으로 응답한다. SOAP 서비스로 작업한다는 것은 엔벨롭의 XML 내용을 파싱해야 함을 의미한다. 그런 이유로 대부분의 SOAP 서비스들은 상위레벨 API에서 호출된다.

Level 2 - 언어 종속적 툴킷(Language-Specific Toolkits): 이 레벨의 개발자는 SOAP와 REST요청에 대한 작업을 위해 언어 종속적인 툴킷을 사용한다. 비록 개발자가 와이어를 통해 전달되는 데이터들의 포맷이나 구조에 아직 치중을 해야 하지만 많은 세부사항(예를 들어 응답 코드의 처리와 시그니취 계산)들은 그 툴킷에 의해 처리된다.

Level 3 - 서비스 종속적 툴킷(Service-Specific Toolkits): 이 레벨의 개발자는 어떤 한 특정 서비스와 작업하기 위해 상위 레벨의 툴킷을 사용한다. 이 레벨에서 작업하는 개발자는 비즈니스 목적과 절차에 집중할 수 있다. 개발자는 와이어 프로토콜에 집중할 때 보다 기관

에 관련된 데이터와 프로세스 문제에 집중할 때 훨씬 더 생산적이 될 수 있다.

Level 4 - 서비스 독립적 툴킷(Service-Neutral Toolkits): 이 레벨은 API의 최상위 레벨이다. 이 레벨에서의 개발자는 여러 클라우드 컴퓨팅 제공자들과의 연결을 위해 하나의 공통된 인터페이스를 사용한다. 레벨 3과 마찬가지로 이 레벨의 개발자는 비즈니스 목적과 비즈니스 절차에 집중한다. 그러나 레벨 3과는 달리 레벨 4에서는 개발자는 어떤 클라우드 서비스와 그들이 작업하고 있는지 걱정할 필요가 없다. 서비스 독립적 툴킷으로 작성된 한 어플리케이션은 코드상의 작은 수정만으로도 다른 클라우드 벤더를 사용할 수 있어야 한다. (26 페이지의 클라우드 벤더 변경을 참조)

2.4.2 API들의 유형(Categories of APIs)

프로그래밍 인터페이스는 다섯 가지의 유형으로 나누어 질 수 있다.

유형 1 - 일반적인 프로그래밍(Ordinary Programming): C#, PHP, Java 등과 같은 일반적인 어플리케이션 프로그래밍이다. 이 카테고리에서는 클라우드와 연관되는 부분이 없다.

유형 2 - 배치(Deployment): 클라우드에 어플리케이션을 배치하기 위한 프로그래밍 인터페이스이다. 클라우드 종속적인 모든 패키징 기술과 더불어 .Net 어셈블리와 EAR/WAR 파일 등과 같은 전통적인 기존의 패키징 방법들이 포함된다.

유형 3 - 클라우드 서비스(Cloud Service): 클라우드 서비스와 함께 동작하는 프로그래밍 인터페이스이다. 전 섹션에서 논의된 것과 같이 클라우드 서비스 API들은 서비스 종속적이거나 서비스 독립적이다. 이러한 API들은 클라우드 스토리지 서비스, 클라우드 데이터베이스, 클라우드 메시지 큐, 그리고 그 이외의 클라우드 서비스 등 여러 세부 유형으로 나뉘어진다. 클라우드 서비스 API를 사용하여 코드를 작성하는 개발자들은 그들이 클라우드를 사용하고 있다는 것을 알고 있다.

유형 4 - 이미지와 인프라 관리(Image and Infrastructure Management): 가상 머신 이미지와 인프라의 세부 사항들을 관리하기 위한 프로그래밍 인터페이스이다. 이런 API들은 이미지의 업로드, 배치 시작, 중지, 재시작, 그리고 삭제를 지원한다. 인프라 관리 API들은 방화벽, 노드 관리, 네트워크 관리, 그리고 부하분산 등의 세부적인 사항들을 제어한다.

유형 5 - 내부 인터페이스(Internal Interface): 클라우드 인프라의 서로 다른 여러 부분들 사이의 내부 인터페이스를 위한 프로그래밍 인터페이스이다. 이런 API는 클라우드 아키텍처 내의 스토리지 레이어 벤더 변경 시 사용되는 API이다.

2.4.3 개발자 역할(Developer Roles)

개발자들을 위한 요구사항들을 논의하기 위해서는 개발자들이 하는 다양한 역할들을 명시하는 것이 도움이 된다. API들과 클라우드 서비스들에 대한 요구사항은 각각의 역할에 따라

다르다. 아래는 본 문서에서 논의할 역할들과 그들이 사용하는 API 유형들이다.

고객 응용 개발자(Client Application developer): 최종 사용자를 위한 클라우드 기반의 클라이언트 어플리케이션을 개발한다. 이 개발자들은 클라우드 서비스를 위한 API를 사용한다. (유형 3)

응용 개발자(Application developer): 클라우드를 사용하는 전통적인 어플리케이션을 개발한다. 이런 개발자들은 유형 3의 클라우드 서비스 API 뿐만 아니라 유형 1의 일반적인 API들을 사용한다.

배치자(Deployers): 클라우드를 사용하는 어플리케이션들을 포장(package), 배치, 그리고 유지한다. 생명주기(lifecycle) 관리 또한 여기에 속한다. 이러한 개발자들은 배치, 클라우드 서비스, 그리고 이미지 관리를 위한 API들을 사용한다. (유형 2, 3, 4)

관리자(Administrator): 배치와 인프라 관리를 포함하는 여러 레벨에서 어플리케이션들과 작업한다. 이런 개발자들은 유형 2, 3, 4의 API들을 사용한다.

클라우드 제공자(Cloud Provider): 그들이 제공하는 클라우드의 하부에 있는 인프라와 작업을 한다. 이러한 개발자들은 유형 5의 API들을 사용한다.

3 유즈케이스 시나리오

엔터프라이즈 클라우드 사용 시나리오는 가장 전형적인 클라우드 컴퓨팅 유즈케이스들을 설명하고 있을 뿐이지, 클라우드 환경에서 나타날 수 있는 모든 케이스들을 나열한 것은 아니다.

이번 섹션에 있는 그래픽은 공통의 요소들을 갖고 있다. 만약 주어진 요소가 특정 유즈케이스에 적용되지 않는다면, 회색으로 표시되거나 점선으로 그려졌다. 예를 들어, 프라이빗 클라우드 유즈케이스는 최종 사용자 또는 퍼블릭 클라우드를 포함하지 않으므로, 엔터프라이즈만 색상이 표현됐다.

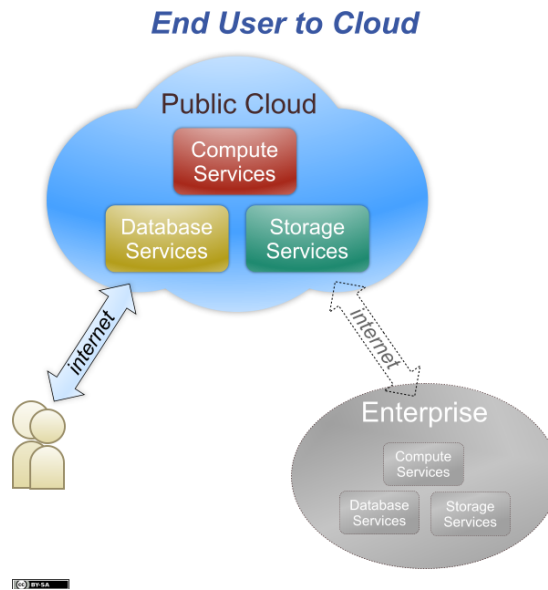
<p>최종 사용자와 클라우드</p>	<p>클라우드 안에서 실행 중이고 최종사용자가 접근하는 어플리케이션</p>	
<p>기업과 클라우드와 최종 사용자</p>	<p>퍼블릭 클라우드 안에서 실행 중이고 직원들과 고객이 접근하는 어플리케이션</p>	
<p>기업과 클라우드</p>	<p>클라우드 어플리케이션은 기업 내부의 IT 자원 및 기능들과 통합</p>	
<p>기업과 클라우드와 기업</p>	<p>퍼블릭 클라우드 안에서 실행 중인 클라우드 어플리케이션은 파트너 사의 어플리케이션과 상호운용 (공급망, supply chain)</p>	
<p>프라이빗 클라우드</p>	<p>기관의 방화벽 안에 호스팅된 클라우드</p>	

<p>클라우드 벤더 교체</p>	<p>클라우드 서비스를 이용하는 기관은 해당 클라우드 제공자를 바꾸거나 추가적인 제공자와 작업하는 경우</p>	
<p>하이브리드 클라우드</p>	<p>여러 클라우드가 함께 작업하고, 데이터, 어플리케이션들, 사용자 식별, 보안 등의 다른 세부사항들은 클라우드 브로커에 의해 조정</p>	

3.1 최종 사용자와 클라우드

이 시나리오에서, 최종 사용자는 클라우드의 데이터와 어플리케이션에 접근한다. 이런 타입의 흔한 어플리케이션에는 이메일 호스팅과 소셜 네트워킹 등이 있다. Gmail 이나 Facebook, LinkedIn의 사용자는 어떤 단말의 어떤 브라우저를 통해서도 어플리케이션과 데이터에 접근할 수 있다. 그리고 그 사용자는 그들의 데이터가 클라우드에서 저장되고 관리 되기 때문에 오직 비밀번호만 알면 된다.

가장 중요한 것은, 사용자는 구조적으로 어떻게 동작하는지 모른다는 점이다. 단지 사용자는 인터넷만 접근 가능하면, 자신들의 데이터에 접근할 수 있다.

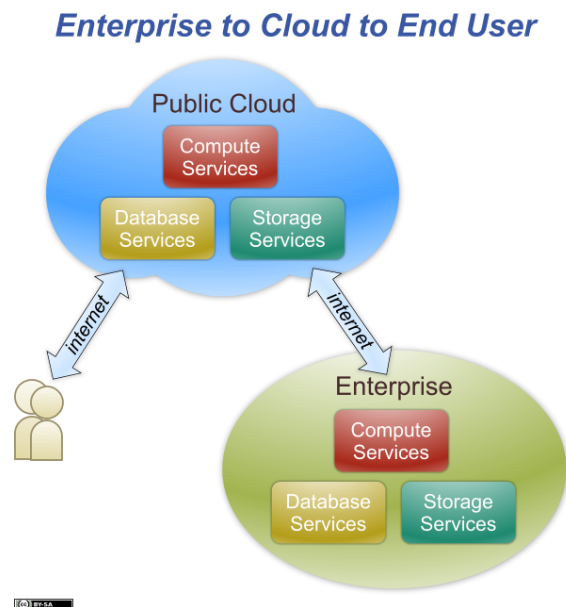


3.1.1 요구사항

- **식별자(Identity):** 클라우드 서비스는 특정 최종 사용자를 인증해야 한다.
- **오픈 클라이언트(An open client):** 클라우드 서비스에 대한 접근은 특정 플랫폼이나 기술 없이도 가능하여야 한다.
- **보안(Security):** 보안의 세부 사항들은 각각의 유즈케이스마다 조금씩 다르고 다양하겠지만, 보안은 모든 유즈케이스에서 공통적인 요구사항이다. 클라우드 컴퓨팅에서의 보안에 대한 총체적인 논의는 이 문서의 범위를 벗어난다.
- **서비스 수준 협약(SLAs):** 비록 최종 사용자에 대한 SLA가 기업에 대한 SLA보다 매우 단순하겠지만, 클라우드 벤더들은 자신들이 제공하는 서비스가 어디까지 보장될 수 있는지를 명확히 하여야 한다.

3.2 기업과 클라우드와 최종 사용자

이 시나리오에서, 기업은 최종 사용자에게 데이터와 서비스를 전달하기 위해 클라우드를 사용한다. 최종 사용자가 기업과 접촉하면, 기업은 데이터를 가져오고 가공하기 위해서 클라우드에 접근하고 그 결과를 사용자에게 보내준다. 최종 사용자는 기업 내의 누군가가 될 수도 있고 외부의 고객이 될 수도 있다.



3.2.1 요구사항

- **식별자(Identity):** 클라우드 서비스는 최종 사용자를 인증해야 한다.
- **오픈 클라이언트(An open client):** 클라우드 서비스에 대한 접근은 특정 플랫폼이나 기술 없이도 가능하여야 한다.
- **연합된 식별자(Federated identity):** 최종 사용자에게 필요한 기본적인 인증과 더불어, 기업 사용자도 기업과의 인증이 추가로 필요할 수 있다. 그러나 클라우드 서비스에서 요구되는 식별자들을 모두 연합하는 인프라가 있어서 기업 사용자는 단 하나의 식별자만 관리하는 것이 이상적이다.
- **위치인지(Location awareness):** 기업이 사용자를 대신하여 관리하는 데이터의 종류에 따라, 데이터가 저장되는 물리적 서버의 위치에 대한 법적 제한이 있을 수 있다. 비록 이것이 사용자가 물리적 인프라의 세부사항을 몰라도 된다는 클라우드 컴퓨팅의 이상에는 위반되는 것이지만, 꼭 필요한 요소이다. 많은 어플리케이션들은 클라우드 서비스를 전달하는 물리적 하드웨어의 위치를 알려주는 API를 클라우드 벤더가 제공하기 전까지는 클라우드로 이동 될 수 없다.
- **계량과 모니터링(Metering and monitoring):** 모든 클라우드 서비스들은 비용관리, 환불처리, 프로비저닝을 위해서 계량되고, 모니터링 되어야 한다.
- **관리와 통제(Management and Governance):** 퍼블릭 클라우드 제공자는 쉽게 거래를 시작하고, 쉽게 클라우드 서비스를 사용할 수 있도록 해준다. 하지만 이것으로 한 기업에 속한 개인이 자신의 개인적인 이익을 위해 클라우드 서비스를 사용할 위험이 생긴다. 가상 머신의 관리와 스토리지, 데이터베이스, 그리고 메시지 큐 같은 클라우드 서비스의 관리에는 어떤 서비스들이 사용되었는지 추적하는 것이 필요하다.

통제는 클라우드 컴퓨팅이 어디에서 사용이 되든지, 정책과 정부 규율을 따르는 것을 보장하는 하기 위해 중요하다. 그 외 통제적 요구 사항은 산업과 지역에 관련된 것들이다.

- **보안(Security):** 기업이 연관된 모든 유즈케이스들은 단일 사용자와 연관된 케이스보다 더 정교한 보안 요구사항들을 가질 것이다. 유사하게, 앞으로 나올 보다 더 진보한 기업 유즈케이스들은 그에 따른 보안 요구사항도 더 진보할 것이다.
- **가상 머신용 공통된 파일 형식(A Common File Format for VMs):** 어느 한 클라우드 벤더의 플랫폼을 위해 만들어진 가상 머신은 다른 벤더의 플랫폼에도 이식 가능해야 한다. 이 요구사항에 대한 모든 솔루션들은 클라우드 벤더가 가상 머신에 스토리지를 붙이는 방법을 설명해야 한다.

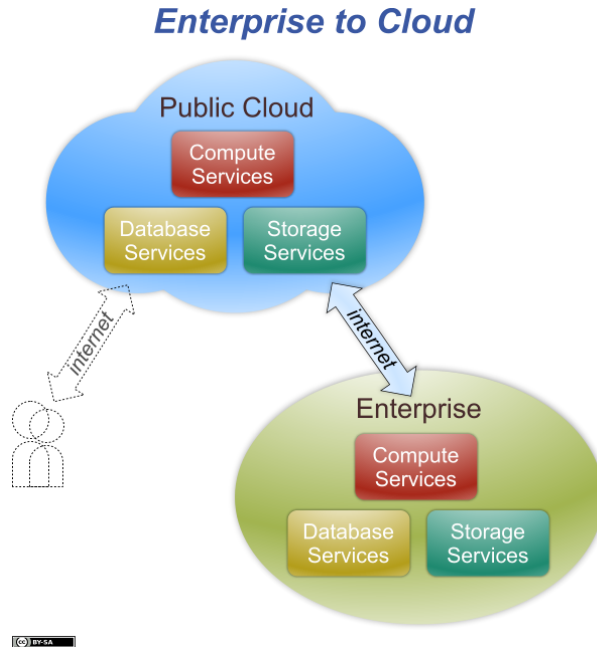
- **클라우드 스토리지와 미들웨어를 위한 공통 APIs(Common APIs for Cloud Storage and Middleware):** 기업의 유즈케이스는 클라우드 스토리지 서비스, 클라우드 데이터 베이스, 그리고 메시지 큐 같은 클라우드 미들웨어 서비스에 접근하기 위한 공통된 API 를 요구한다.

특정 벤더의 클라우드 서비스에서만 동작하는 맞춤 전용 코드를 사용하는 것은 그 해당 벤더의 시스템에 기업이 종속되고, 클라우드 컴퓨팅이 제공하는 경제적인 이득과 유연성이 반감된다.

- **데이터와 어플리케이션의 결합(Data and Application Federation):** 기업용 어플리케이션은 여러 클라우드 기반의 소스로부터 데이터를 결합하고 서로 다른 클라우드에서 동작하는 어플리케이션의 활동을 조정할 수 있어야 한다.
- **SLA와 벤치마크(SLAs and Benchmarks):** 최종 사용자에게 필요한 기본적 SLA에 추가적으로, SLA를 기반으로 계약을 체결한 기업은 성능 벤치마킹을 위한 표준적인 방법이 필요하다. 클라우드 제공자가 전달해야 하는 것에 대해 명확히 정의할 수 있는 방법이 있어야 하고, 실제로 전달된 것을 명확히 측정할 수 있는 방법이 있어야 한다. (SLA에는 추가적으로 클라우드 보안에 대한 내용이 포함되어야 한다; 추가적인 정보를 위해서는 섹션 6의 SLA, 감사와 모니터링에 대한 논의를 보기 바람)
- **라이프사이클 관리(Life cycle Management):** 기업들은 어플리케이션과 문서들의 라이프사이클을 관리할 수 있어야 한다. 여기에는 어플리케이션의 버전, 데이터의 유지 및 삭제 관리 등이 포함된다. 디스커버리(Discovery)는 많은 기관에서 중요 이슈이다. 특정 데이터가 더 이상 사용할 수 없다면 실질적인 법적 책임이 따른다. 데이터의 유지뿐만 아니라 어떤 경우에 기업들은 어떤 특정 시점에 데이터가 확실히 삭제되었는지를 명확하게 하고 싶어 한다.

3.3 기업과 클라우드

이번 유즈케이스는 기업이 내부의 프로세스를 위해 클라우드 서비스를 사용하는 경우이다. 이 경우는 기업에게 대부분의 제어권을 주기 때문에 클라우드 컴퓨팅의 초기 단계에서 가장 흔한 케이스다.



이번 시나리오에서, 기업은 필요한 리소스를 보충하기 위해 클라우드 서비스를 사용한다.

- 백업 혹은 드물게 사용되는 데이터를 저장하기 위해 클라우드 스토리지를 사용한다.
- 최대 부하를 처리하기 위해 추가 프로세서를 온라인으로 가져오기 위해 클라우드의 가상 머신을 사용한다. (물론, 더 이상 필요하지 않다면 가상 머신을 중지할 수 있다.)
- 기업의 특정 기능(email, calendaring, CRM 등)을 위해 클라우드의 어플리케이션을 사용한다.
- 어플리케이션 처리의 부분으로서 클라우드 데이터베이스를 사용한다. 이것은 파트너사, 정부, 에이전시 등과 데이터베이스를 공유할 때 매우 유용하다.

3.3.1 요구사항

기업과 클라우드 유즈케이스의 기본 요구사항은 기업과 클라우드 그리고 최종 사용자간 (Enterprise to Cloud to End User) 유즈케이스와 많은 부분 동일하다. **오픈 클라이언트, 연합된 식별자, 위치인지, 계량과 모니터링, 관리와 통제, 보안, 가상머신용 공통 파일 형식,**

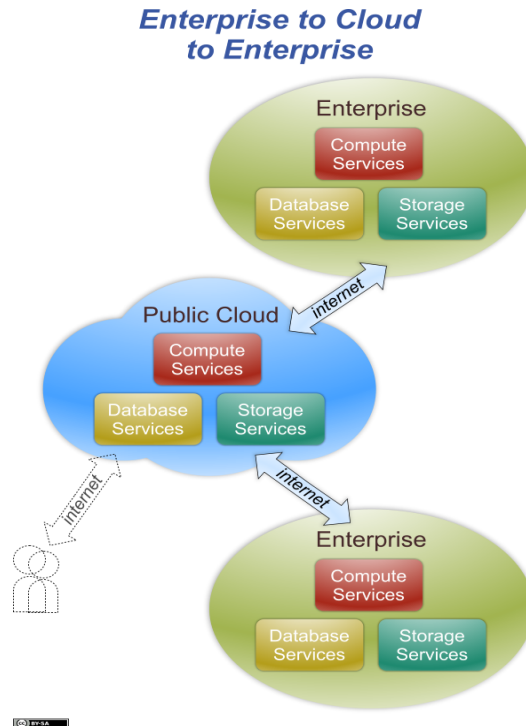
클라우드 스토리지와 미들웨어를 위한 공동된 API, 데이터와 어플리케이션 연동, SLA와 라이프사이클 관리 등이 모두 적용된다.

그 밖의 요구사항은 다음과 같다.

- **배치:** 필요에 따라 가상 머신 이미지를 구축하고 배포하는 작업이 단순해야 한다. 가상 머신 이미지가 구축될 때, 한 클라우드 제공자에서 다른 제공자로의 이미지 이동이 가능해야 하고, 벤더들 마다 다를 수 있는 가상 머신에 스토리지를 장착하는 방법을 알려주어야 한다. 클라우드로의 어플리케이션 배포 역시 간단해야 한다.
- **기업관련 표준과 프로토콜:** 기업들 간의 많은 클라우드 컴퓨팅 솔루션들은 RosettaNet이나 OAGIS 같은 기존의 표준을 사용할 것이다. 이 적용 가능한 표준들은 어플리케이션에 따라, 또 산업에 따라 달라질 것이다.

3.4 기업과 클라우드와 기업

이번 유즈케이스에서는 두 기업이 같은 클라우드를 사용한다. 여기서 중요한 것은 기업들의 어플리케이션들이 서로 상호작용할 수 있도록 클라우드 내에서 자원을 호스팅하는 것이다. 공급 사슬(supply chain)이 이번 유즈케이스의 가장 확실한 예이다.



3.4.1 요구사항

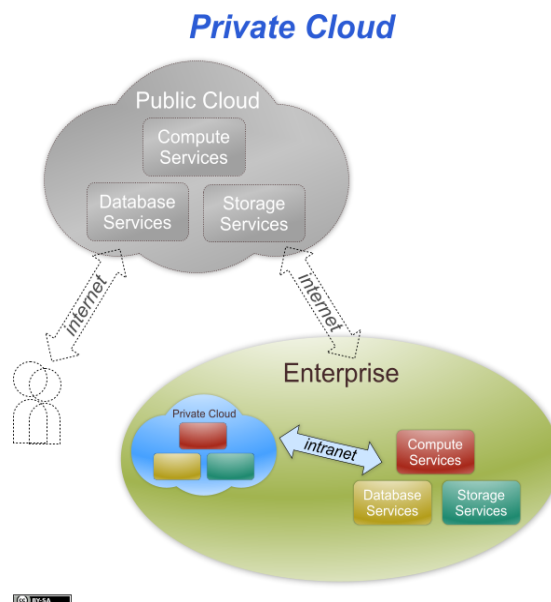
기업과 클라우드 그리고 기업 간(Enterprise to Cloud to Enterprise)의 기본 요구사항은 기업과 클라우드 간의 것들과 거의 유사하다. 식별자, 오픈 클라이언트, 연합된 식별자, 위치 인지, 계량과 모니터링, 관리와 통제, 보안, 산업관련 표준, 스토리지와 미들웨어를 위한 공통된 API, 데이터와 어플리케이션 연동, 그리고 SLA와 라이프사이클 관리 등이 모두 적용된다.

그 밖의 요구사항은 다음과 같다.

- **트랜잭션과 동시성:** 서로 다른 기업들 간에 어플리케이션과 데이터를 서로 공유하기 위해서는 트랜잭션과 동시성이 필수적이다. 만약 두 기업이 같은 클라우드에 호스트된 어플리케이션과 가상 머신, 미들웨어, 그리고 스토리지를 공유한다면, 어느 한쪽 기업에 의한 모든 변경들이 신뢰 있게 적용된다는 것은 중요하다.
- **상호운영성:** 하나 이상의 기업들이 연관되기 때문에, 기업들 간의 상호운영성은 필수다.

3.5 프라이빗 클라우드

프라이빗 클라우드 유즈케이스는 클라우드가 기업 내에 포함되어 있다는 점에서 다른 케이스들과 다르다. 이런 경우는 대규모의 기업들에게 유용하다. 예를 들어, 각 달의 15일과 30일에 어느 기업의 급여부서에 작업량이 몰린다면, 비록 나머지 날에는 작업량이 조금 일지라도, 최대치의 작업량을 다루기 위한 컴퓨팅 파워가 충분히 필요하다. 프라이빗 클라우드를 이용하면 컴퓨팅 파워가 기업 전반에 분포되어 있다. 위의 급여부서는 컴퓨팅 파워가 필요할 때 여분의 파워를 이용할 수 있고, 다른 부서들도 마찬가지다. 이렇게 기업은 기업 전반에 걸쳐 컴퓨팅 파워를 상당부분 절약 할 수 있다.



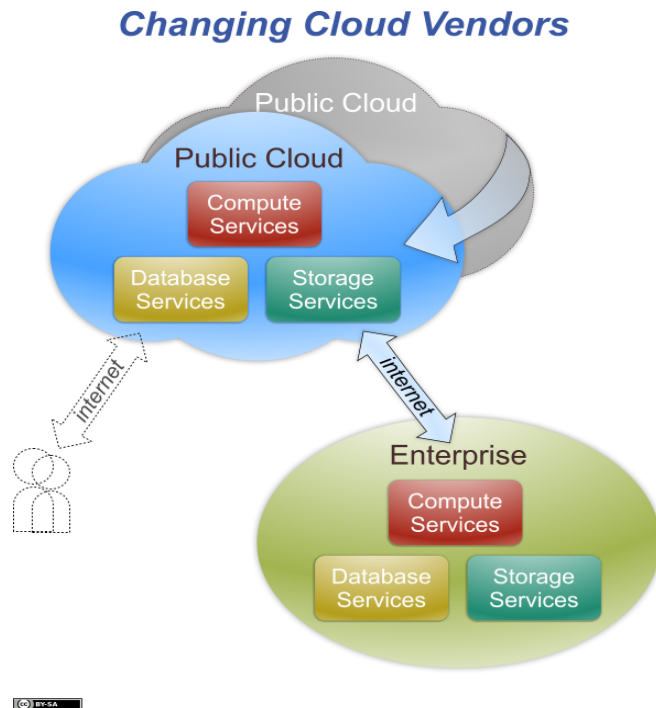
3.5.1 요구사항

프라이빗 클라우드의 기본 요구사항들은 **오픈 클라이언트, 계량과 모니터링, 관리와 통제, 보안, 배치, 상호운영성, 가상머신용 공통 파일 형식, 그리고 SLAs**이다.

식별자, 연합된 식별자, 위치인지, 트랜잭션, 산업관련 표준, 미들웨어를 위한 공동된 API, 그리고 라이프사이클 관리 등은 프라이빗 클라우드에서 요구되지 않다는 점에 주목해야 한다. 많은 경우에, 소비자들은 위치인지가 더 이상 문제가 되지 않기 때문에 프라이빗 클라우드를 사용해야 할 것이다. 기업 내에서 클라우드를 유지하게 되면, 식별자 관리, 표준과 공통 APIs에 대한 요구사항이 필요 없어진다.

3.6 클라우드 벤더 교체

이번 유즈케이스는 다른 벤더를 추가하거나 기존의 벤더를 교체함으로써, 다른 벤더와 작업하는 경우이고, 이 문서에 언급되었던 모든 유즈케이스에 적용된다. 큰 변경 없이 다른 벤더와 작업할 수 있다는 것은 개방과 표준의 중요 이점이다.



여기에는 요구사항들이 약간씩 다른 4개의 시나리오가 있다. 일반적으로, 클라우드 벤더 교체에는 **오픈 클라이언트, 위치인지, 보안, SLAs, 가상 머신용 공통 파일 포맷, 그리고 클라우드 스토리지와 미들웨어를 위한 공통 APIs**가 필요하다. 자세한 내용은 다음 세부항목에서

확인 할 수 있다.

3.6.1 시나리오 1: SaaS 벤더의 교체

이번 시나리오에서 클라우드 고객은 SaaS 벤더를 교체한다. 두 SaaS 벤더는 같은 어플리케이션(CRM, 회계, 문서 작성기 등)을 제공한다. 기존 벤더의 소프트웨어에서 만들어진 문서와 데이터는 두 번째 벤더의 소프트웨어에서 인식/호환될 수 있어야 한다. 어떤 경우에는 고객이 두 벤더를 번갈아 사용할지도 모른다.

3.6.1.1 요구사항

- **산업관련 표준:** 한 벤더의 어플리케이션에서 다른 벤더로 문서와 데이터를 이동시키기 위해서는 두 어플리케이션이 같은 포맷을 지원해야 한다. 그 포맷은 어플리케이션의 타입에 따라 다를 것이다.

어떤 경우에는, 다른 어플리케이션 유형들을 위한 표준 API들이 필요할 것이다.

이 요구사항들은 클라우드와 관계가 없다는 것을 알 필요가 있다. Zoho에서 Google Docs로 문서 이동 할 때나 Microsoft Office에서 OpenOffice로 문서 이동할 때나 사용되는 표준은 같다.

3.6.2 시나리오 2: 미들웨어 벤더의 교체

이번 시나리오에서 클라우드 고객은 클라우드 미들웨어 벤더를 교체한다. 기존 벤더의 데이터, 쿼리, 메시지 큐, 어플리케이션들은 내보내질 수 있어야 하고, 다른 벤더의 미들웨어는 읽어 올 수 있어야 한다.

3.6.2.1 요구사항

- **산업관련 표준:** 한 벤더의 어플리케이션에서 다른 벤더로 문서와 데이터를 이동시키기 위해서는 두 어플리케이션이 같은 포맷을 지원해야 한다. 그 포맷은 어플리케이션의 타입에 따라 다를 것이다.
- **클라우드 미들웨어를 위한 공통 API:** 이것은 클라우드 데이터베이스, 클라우드 메시지 큐, 미들웨어를 포함한 오늘날의 클라우드 서비스를 위해 지원되는 모든 동작이 포함된다. 데이터베이스와 테이블과의 연결, 생성, 삭제 등을 위한 APIs.

클라우드 데이터베이스 벤더들은 그들의 제품을 더 탄력적으로 만들고, 처리에 많은 자원을 요구하는 큰 데이터 셋에 대한 질의처리를 제한하기 위해 특정한 제약사항들을 적용했다. 예를 들어, 어떤 클라우드 데이터베이스는 테이블들을 대상으로 하는 조인

을 불허하고, 또 어떤 데이터베이스는 트루(true) 데이터베이스 스키마를 지원하지 않는다. 이와 같은 제한은 클라우드 데이터베이스 벤더들 사이에서의 이동, 특히 트루 관계형 모델을 기반으로 구축된 어플리케이션에 대해서는 해결해야 할 큰 과제이다.

메시지 큐 같은 그 밖의 미들웨어 서비스는 서로 유사하기 때문에, 공통 API를 만들기가 더 단순할 것이다.

3.6.3 시나리오 3: 클라우드 스토리지 벤더의 교체

이번 시나리오에서 클라우드 고객은 클라우드 스토리지 벤더를 교체한다.

3.6.3.1 요구사항

- **클라우드 스토리지를 위한 공통 API:** 한 클라우드 스토리지 시스템에서 데이터를 읽고 쓰는 코드는 가능하면 최소한의 변경사항만으로 다른 시스템과 연동할 수 있어야 한다. 그리고 그 변경 사항은 구성/설정 코드에 국한시켜야 한다. 예를 들어, JDBC 어플리케이션에서 URL과 드라이버 이름의 포맷은 다른 데이터베이스 벤더들마다 다르지만 그 데이터베이스와 상호작용하는 코드는 동일하다.

3.6.4 시나리오 4 : 가상머신 호스트의 변경

이번 시나리오에서는 클라우드 고객은 한 클라우드 벤더의 시스템에 구축된 가상머신을 다른 벤더의 시스템에서 실행 시키길 원한다.

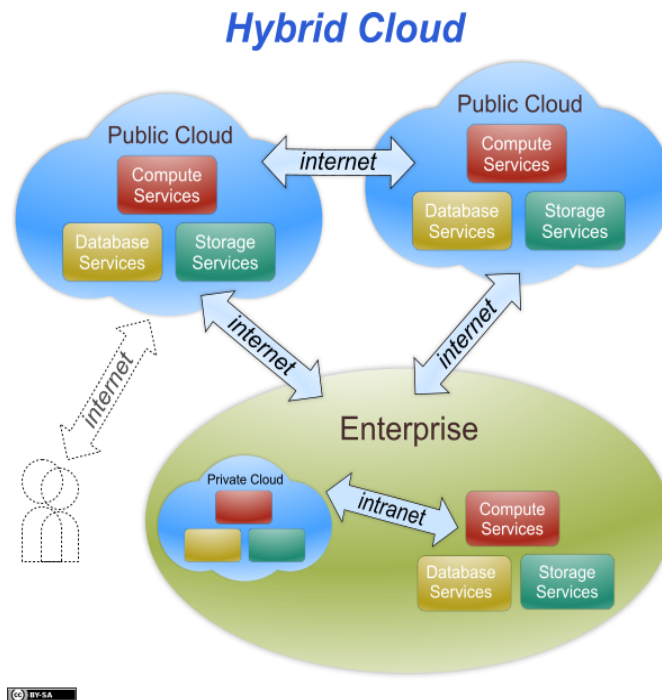
3.6.4.1 요구사항

- **가상머신을 위한 공통 포맷:** 가상머신 포맷은 어떤 운영체제와도 작동해야 한다.

여기서의 가정은 가상머신 자체가 윈도우나 리눅스 같은 운영체제를 실행하고 있다는 점이다. 이것은 가상머신의 사용자가 클라우드에 가상머신을 구축하기 전에 플랫폼을 선택한 것이고, 가상머신 안에서 동작하는 소프트웨어를 위한 요구사항들은 클라우드와 연관되는 것이 없다.

3.7 하이브리드 클라우드

이번 유즈케이스는 퍼블릭 클라우드와 프라이빗클라우드를 포함해서 여러 클라우드가 함께 동작하는 경우이다. 하이브리드 클라우드는 자기의 리소스와 다른 제공자의 리소스를 결합하는 연합 클라우드 제공자에 의해 전달될 수 있다. 브로커도 또한 하이브리드 클라우드를 제공할 수 있다. 차이점은 브로커는 자신의 클라우드 리소스를 갖고 있지 않다는 점이다. 하이브리드 클라우드 제공자는 소비자 요구사항에 따라 클라우드 리소스를 관리해야 한다.

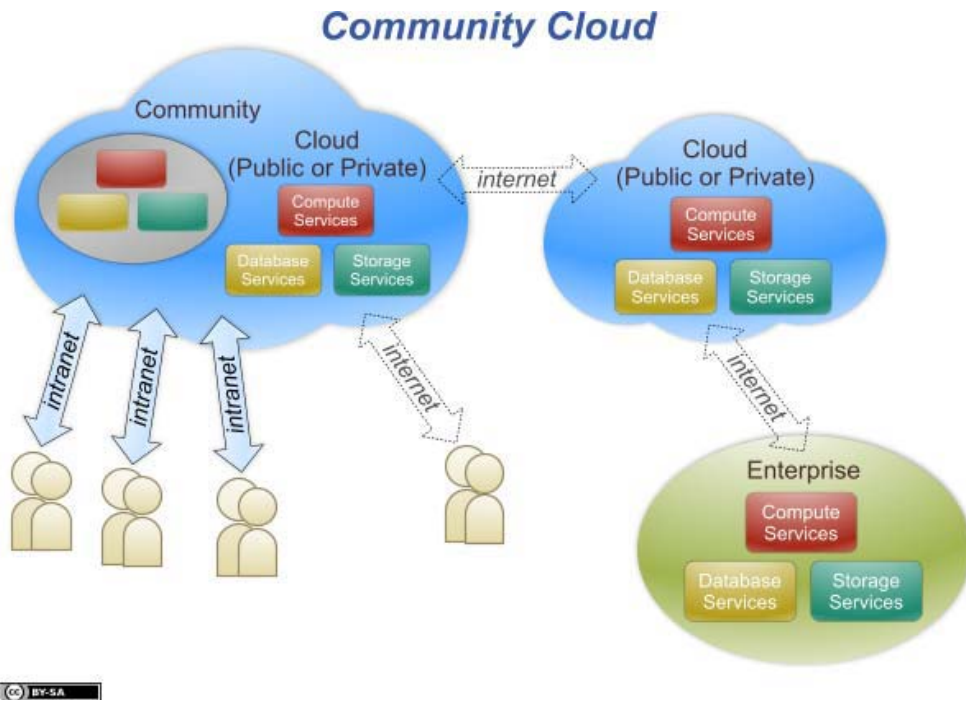


하이브리드 클라우드의 소비자에게는 본 유즈케이스는 앞에서 언급했던 최종 사용자와 클라우드 간(End User to Cloud)의 유즈케이스와 거의 차이가 없다는 점이 중요하다. 사용자는 하이브리드 클라우드 제공자가 실제로 무엇을 하는지 알지 못한다.

3.7.1 요구사항

- **트랜잭션과 동시성(concurrency)**을 제외하고 앞에서 언급했던 모든 요구사항들이 적용된다. 특히, **보안, 데이터와 어플리케이션의 연함, 그리고 상호운영성**이 적용된다.
- **SLAs:** SLA를 표현하는 표준 포맷은 기계가 읽을 수 있어야 한다. 이는 하이브리드 클라우드 제공자로 하여금 사람의 개입 없이 소비자의 조건에 따라 리소스를 선택할 수 있게 한다.

섹션 2에서 언급했듯이, 커뮤니티 클라우드의 요구사항은 하이브리드 클라우드의 요구사항에 포함된다. 커뮤니티 클라우드는 공통의 목적을 갖는 기업들 간에 공유되는 인프라를 갖고 있다. 아래는 커뮤니티 클라우드의 구성도이다.



커뮤니티와 커뮤니티 클라우드 사이의 통신은 인트라넷(Intranet)을 통해 이루어진다. VPN이 될 수도 있지만, 접근은 퍼블릭(공공) 인터넷을 사용하지는 않는다.

3.8 상호참조: 요구사항들과 유즈케이스

다음 표는 요구사항과 유즈케이스 사이의 관계를 요약하고 있다.

요구사항	최종 사용자와 클라우드	기업과 클라우드와 최종 사용자	기업과 클라우드	기업과 클라우드와 기업	프라이빗 클라우드	클라우드 벤더 교체	하이브리드 클라우드
식별자	✓	✓		✓			✓
오픈 클라이언트	✓	✓	✓	✓	✓	✓	✓
연합된 식별자		✓	✓	✓			✓
위치인지		✓	✓	✓		✓	✓
계량과 모니터링		✓	✓	✓	✓		✓
관리와 통제		✓	✓	✓	✓		✓
보안	✓	✓	✓	✓	✓	✓	✓
배치			✓		✓		✓
트랜잭션과 동시성				✓			
상호운용성				✓			✓
산업관련 표준			✓	✓			✓
가상머신 이미지 포맷		✓	✓	✓	✓	✓	✓
클라우드 스토리지 API		✓	✓	✓		✓	✓
클라우드 데이터베이스 API		✓	✓	✓		✓	✓
클라우드 미들웨어 API		✓	✓	✓		✓	✓
데이터와 어플리케이션의 결합		✓	✓	✓			✓
SLAs	✓	✓	✓	✓	✓	✓	✓
라이프사이클 관리		✓	✓	✓			✓

4 고객 시나리오

이번 섹션은 본 유즈케이스의 고객 경험을 설명한다. 여기에 그러한 고객들의 시나리오가 요약되어 있다.

고객 시나리오	고객의 문제 해결	요구사항 및 기능	적용 가능한 유즈케이스
급여 처리	<ul style="list-style-type: none"> 처리 시간 감소 하드웨어 요구사항 감소 미래 확장 가능한 신축성 	IaaS(VMs), 클라우드 스토리지	기업과 클라우드
물류 및 프로젝트 관리	<ul style="list-style-type: none"> 처리 시간 감소 수작업 제거 개발 환경의 개선과 간소화 	PaaS (응용 프레임워크), 클라우드 스토리지	기업과 클라우드와 최종 사용자
중앙 정부	<ul style="list-style-type: none"> IT 전문 기술 통합 하드웨어 요구사항 감소 	IaaS, PaaS	프라이빗 클라우드
지방 정부	<ul style="list-style-type: none"> IT 전문 기술 통합 하드웨어 요구사항 감소 	IaaS, PaaS	하이브리드 클라우드
천문학적 데이터 처리	<ul style="list-style-type: none"> 하드웨어 비용을 크게 감소(프로세싱 파워와 저장장치) 에너지 비용을 크게 축소 관리의 간소화 	IaaS(VMs), 클라우드 스토리지	기업과 클라우드와 최종 사용자

4.1 고객 시나리오: 클라우드에서의 급여처리

4.1.1 섹션 3에서 적용 가능한 유즈케이스:

기업과 클라우드

4.1.2 고객 시나리오:

이 시나리오에서는 두 대의 서버가 복잡하고 시간이 소요되는 급여처리를 위해 할당되어 있다. 이 기관에서는 클라우드 환경에서 급여처리를 하는 것이 얼마나 실용적인지를 보기로 결정했다. 기존의 급여 시스템은 분산 어플리케이션으로 설계되어 있어서 클라우드 환경으로 옮기기가 상대적으로 수월했다.

급여 어플리케이션은 직원들의 데이터 처리를 위해 SQL 데이터베이스를 사용했다. 클라우

드 데이터베이스 서비스를 사용하기 위해 어플리케이션을 재 작성하기 보다는, 데이터베이스 서버가 있는 가상머신을 배치했다. 데이터베이스 서버는 클라우드 스토리지 시스템으로부터 데이터를 읽어 그것을 가지고 관계형 테이블을 만들었다. 원본 데이터베이스의 크기 때문에, 오직 급여 처리를 위해 필수적인 정보들만을 추출하도록 추출도구들을 사용하였다. 그 추출된 정보는 클라우드 스토리지 서비스로 전송되었고, 데이터베이스 서버에 의해 사용되었다.

급여 어플리케이션은 동시에 실행되는 가상머신 4곳에 배치되었다; 그러한 4개의 가상머신은 데이터베이스 서버를 호스팅하는 가상머신과 함께 작동한다. 급여 어플리케이션의 구성은 데이터베이스 서버를 호스팅하는 가상머신을 사용하도록 변경되었으며, 그 이외의 어플리케이션 변경은 없었다.

4.1.3 고객의 문제 해결

클라우드 기반 어플리케이션 버전에서는 급여 작업의 처리 시간이 80% 감소되었다. 추가적인 혜택으로, 이전에 급여처리 전용으로 사용되던 두 대의 서버가 다른 작업을 위해 사용될 수 있었다. 마지막으로, 클라우드 기반 버전은 훨씬 더 신축성이 있고 이는 그 기관이 확장됨에 따라 상당한 이점이 될 것이다.

4.1.4 요구사항들과 기능:

사용된 클라우드 서비스들은 가상 머신들과 클라우드 스토리지(IaaS)였다. 급여 어플리케이션은 수정될 필요가 없었고, 단순히 가상 머신에 배치되었다. 원래의 어플리케이션은 관계형 데이터베이스를 사용하였다. 클라우드 데이터베이스를 사용하도록 데이터 구조와 어플리케이션을 수정하는 대신에 관계형 데이터베이스 서버가 클라우드에 배치되었다.

유일하게 사용된 API는 S3 클라우드 스토리지 API이다.

4.1.5 이식성에 대한 우려:

급여 어플리케이션은 Fedora와 Java 1.5에서 실행되기에 Fedora를 지원하는 모든 클라우드 제공 업체의 플랫폼에서 아무런 변경 없이 실행될 것이다. 만약 다른 업체가 급여 처리 과정에서 사용되었던 특정 S3 API를 지원하지 않는다면, 그 클라우드 스토리지 제공자를 사용하기 위해 어플리케이션을 수정하는 것은 문제가 될 수도 있다. 마지막으로 클라우드 데이터베이스를 사용하도록 어플리케이션을 변경하는 것은 무척 어려울 수도 있다. 특히 혹시라도 관계형 모델을 지원하지 않는 클라우드 데이터베이스로 이동하는 것이라면 더욱 어려울 수 있다.

4.2 고객 시나리오: 클라우드에서의 프로젝트 관리

4.2.1 섹션 3에서 적용 가능한 유즈케이스

기업과 클라우드와 최종 사용자

4.2.2 고객 시나리오:

약 20명의 행정 직원을 둔 중소 건설 회사는 그들의 자원을 관리하고, 프로젝트 스케줄링을 최적화하며 작업 비용을 계산 할 방법이 필요했다. 이 회사는 일반적인 시스템과는 달리 매우 구체적인 요구사항들을 가지고 있었기에 회계 소프트웨어인 Quickbooks와 스프레드시트의 조합을 사용하였다. 이 시스템은 신축적이지 않았으며 큰 인적 자원의 낭비였다.

이 문제에 대한 해결책은 고유의 클라이언트용 어플리케이션을 구축하는 것이었다. 모든 비즈니스 논리는 클라이언트에 있다. 그 어플리케이션을 위한 데이터는 Google App Engine (GAE) 데이터스토어에서 제공된다. 이 데이터스토어는 비록 RDF-OWL 온톨로지를 호스트 하지만, RDF 그래프 이외에는 어떤 종류의 스키마도 지원하지 않는다. 클라이언트는 이 온톨로지를 사용하여 데이터의 유효성을 검사한 후에, 사용자에게 데이터를 보여주거나 또는 데이터를 GAE로 되돌려 보낸다.

데이터 작업들은, HTTP를 통한 어플리케이션에 종속적인 RESTful 프로토콜을 사용하여, 데이터스토어와 통신한다. 데이터스토어는 서버에서 관리되는 silos 내에서 제공되고 있는 어플리케이션에 관련된 RDF 그래프를 유지한다. 보안은 데이터의 특정 사일로(silo)를 사용하는 어플리케이션의 요구사항에 따라 별도로 각각의 사일로를 위해 구현된다. 이러한 시스템을 사용하면 여러 개의 어플리케이션들도 각각의 새로운 코드 기반을 구축하지 않고 데이터스토어를 사용할 수 있다.

로컬에 호스트 된 Quickbooks SQL 서버와 사용자 맞춤형 스프레드시트에 있는 데이터는 일회성 데이터 마이그레이션 스크립트(script)를 사용하여 데이터를 일치시킨 후에 GAE 데이터스토어로 옮겨졌다. 데이터 세트는 작고 로컬 자원을 이용하여 쉽게 처리되었다.

클라이언트 어플리케이션은 데이터의 가장 최근 변경 사항의 일부를 포함하는 로컬 데이터스토어를 유지한다. 어플리케이션의 REST 아키텍처는 HTTP에 내장된 캐싱 기능을 이용하여 자동으로 마스터 데이터 스토어의 변경 사항이 클라이언트에게 전달되도록 한다. 데이터의 일부 서브 셋만을 사용함에 따른 성능상의 이점과 더불어 이러한 설계는 보안을 단순화한다. 만약 클라이언트 어플리케이션이 특정 필드 또는 레코드에 접근할 필요가 없다면, 데이터스토어의 그러한 부분은 절대 서버를 벗어나지 않는다.

4.2.3 고객의 문제 해결:

회계 소프트웨어와 스프레드시트 프로그램이 설치된 비효율적인 시스템에 있던 데이터를 클라우드 기반 시스템으로 옮겼다. 그 결과 데이터스토어가 다양한 종류의 어플리케이션에서 사용될 수 있고, 앞으로의 개발 및 유지 보수를 훨씬 간단하게 만들 수 있다.

비록 원래 어플리케이션 인프라는 여전히 사용되고 있지만, 그 인프라에서 작성된 어플리케이션들은 데이터를 분석하고 조작하기 위해서 더 이상 스프레드시트에 의지할 필요가 없다. 스프레드시트를 더 이상 유지할 필요가 없다는 점은 큰 비용 절감을 가져온다. 게다가 손으로 데이터를 자르고 붙이는 일은 더 이상 작업의 일부가 아니며, 따라서 지루한 작업과 오류의 원인을 제거 할 수 있다.

4.2.4 요구사항들과 기능:

여기서 사용된 클라우드 서비스는 데이터베이스를 지원하는 PaaS 구현물인 Google App Engine을 사용하였다. RESTful API와 클라우드 데이터스토어의 조합은 전통적인 관계형 데이터베이스를 사용하여 구현한 경우보다 더욱 신속성 있는 어플리케이션을 만드는 것이 가능하다.

4.2.5 이식성에 대한 우려:

이 어플리케이션은 Google App Engine과 그것의 데이터베이스인 BigTable에서 실행된다. BigTable은 신속성을 갖는 분산된 다차원의 정렬된 맵이다. 이것은 다른 대부분의 데이터스토어들과는 큰 차이를 갖는 것으로 어플리케이션 개발에 대해 근본적으로 다시 생각할 것을 요구한다. 보다 전통적인 데이터스토어에서 돌아가도록 그 어플리케이션을 이식한다면 그 어플리케이션의 구조를 크게 바꾸어야 했을 것이다.

4.3 고객 시나리오: 클라우드에서의 중앙 정부 서비스

4.3.1 섹션 3에서 적용 가능한 유즈케이스

프라이빗 클라우드

4.3.2 고객 시나리오:

일본 정부의 부처들은 그들의 인프라에 수 천대의 서버들을 가지고 있다. 중앙 정부는 정부 어플리케이션들을 호스팅하기 위해서 보안과 중앙집중식 인프라를 제공하는 "카수미가세키 (Kasumigaseki)"라 불리는 프라이빗 클라우드 환경을 발표했다.

급여, 회계와 인사 관리 같은 현존하는 백 오피스 시스템들은 프라이빗 클라우드 환경에서 가상화되고 호스팅될 것이다. 전자 조달과 같은 일부 프론트 오피스 시스템들은 퍼블릭 클라우드에 가상화 될 것이지만 그것은 이 프로젝트의 범위 밖이다. 이 프로젝트의 궁극적인 목표는 중복되는 시스템과 각 부처에 있는 관리자의 필요성을 제거하여 총 소유 비용을 줄이는 것이다.

4.3.3 고객의 문제 해결

카수미가세키 클라우드에 의해 해결된 3가지 문제는 비용 절감, 에너지 소비 절감, IT 인력의 감축이다.

4.3.4 요구사항들과 기능

클라우드 인프라는 일본 통신 회사가 구축한 사설 네트워크 상에 만들어지게 된다. 개인정보 및 보안이 주요 우려 사항이기 때문에 프라이빗 클라우드가 요구된다. 많은 유형의 개인정보들이 일본 국외의 서버에 저장되는 것은 불법이다.

4.3.5 이식성에 대한 우려

정부가 정부의 자체 어플리케이션들을 호스트 하기 위해서 자체 프라이빗 클라우드를 만들고 있기 때문에 이식성은 문제가 되지 않는다. 정부는 중앙 집중된 어플리케이션과 데이터들을 프라이빗 클라우드에서 퍼블릭 클라우드로 이동할 의도가 없다.

4.4 고객 시나리오: 하이브리드 클라우드에서의 지방 정부 서비스

4.4.1 섹션 3에서 적용 가능한 사용 케이스 시나리오

하이브리드 클라우드

4.4.2 고객 시나리오:

일본 전역에는 자체 서버와 IT인력을 가지고 있는 지방 자치 정부가 1800개 이상 있다. 카수미가세키 클라우드의 2차 목표는 하이브리드 클라우드 환경을 제공하는 것이다. 카수미가세키 클라우드에 추가적으로 일본의 중앙 정부는 일본의 행정 구역인 현 수준에서 지방 정부를 그룹화하기로 결정했다. 각 현은 프라이빗 클라우드를 가질 것이고 이것을 카수미가세키 하이브리드 클라우드와 연결할 것이다. 내부 업무들과 일부 데이터는 현의 프라이빗 클라우드에 호스트 될 것이며, 그 외 다른 데이터는 로컬에 저장된다. 가능한 최대한으로 기존 시스템들은 가상화되고 카수미가세키 클라우드에 호스트 될 것이다.

4.4.3 고객의 문제 해결:

이 하이브리드 클라우드에서 해결된 세 가지 문제들이 비용 절감, 에너지 소비 절감 그리고 IT 인력의 감소이다.

4.4.4 요구 사항들과 기능:

개인 정보 보호와 보안은 이 시나리오에서 아주 중요하다. 일본 법률은 일부 유형의 데이터가 지방 자치 단체 서버 이외에 외부에 저장되는 것을 금지하고 있다. 그래서 어플리케이션 및 데이터가 카수미가세키 클라우드로 이동하는 것은 고려할 옵션이 아니다. 일부 프로세스

은 지방 자치 단체의 인프라에 있기 때문에, 하이브리드 클라우드 내의 어플리케이션들과 데이터들을 연합(federation)하는 것은 매우 중요하다.

4.4.5 이식성에 대한 우려:

이전 고객 시나리오와 마찬가지로 정부가 자체 프라이빗 클라우드(private cloud)를 구축하기 때문에 이식성에 대한 우려는 없다. 정부는 중앙 집중된 어플리케이션과 데이터들을 프라이빗 클라우드에서 퍼블릭 클라우드로 이동할 의도가 없다.

4.5 고객 시나리오: 천문 데이터 처리

4.5.1 섹션 3에서 적용 가능한 유즈케이스

최종 사용자와 클라우드

4.5.2 고객 시나리오:

가이아(Gaia)는 우리의 은하계에 있는 10 억 개의 별들을 조사하는 유럽 우주국(European Space Agency, ESA)의 임무이다. 그것은 5 년간의 기간 동안 약 70번씩 각각의 해당 별들을 관찰하고, 정확한 별들의 위치, 거리, 움직임과 밝기의 변화를 차트화 시킨다. 여분의 태양 항성들 또는 갈색 난쟁이라 불리는 떨어진 항성들과 같은 새로운 수십만 개의 천상의 개체들을 발견할 것으로 기대되고 있다.

이 임무는 분석해야 할 대량의 데이터를 수집할 것이다. ESA는 그 데이터를 분석하기 위해 클라우드 기반의 프로토타입 시스템을 만들기로 결정했다. 목표는 대규모 데이터 세트를 처리하기 위해 클라우드 컴퓨팅을 사용할 경우 기술적인 그리고 비용적인 면들을 결정하기 위한 것이다.

프로토타입 시스템에서는 작업 실행 및 데이터 처리를 위해 분산 컴퓨팅 (자체 개발) 프레임워크가 사용된다. 프레임워크는 AGIS(Astrometric Global Iterative Solution)를 실행하도록 구성되어 있고, 이 프로세스는 그 결과가 수렴할 때까지 이 데이터들을 대상으로 반복적으로 실행된다.

처리를 위해서, 각각의 작업 노드들은 데이터베이스에서 작업 설명을 수신하고, 데이터를 가져온 후, 처리하여 중간 단계의 서버로 결과를 보낸다. 이 중간 단계의 서버는 그 다음 반복을 위하여 데이터를 갱신한다.

이 프로토타입은 2 백만 개 행성들의 5년치 데이터를 평가했으며, 이는 실제 프로젝트에서 처리해야 할 전체 데이터의 일부분이다. 프로토타입은 각각 100분씩 소요되는 작업을 24번 반복했는데, 이는 20개의 가상머신들로 이루어진 그리드 컴퓨터를 40시간 동안 실행시키는 것과 동등한 것이다. 전체 프로젝트에서는 1억 개의 주 행성들의 6년치 데이터가 분석될 것

이고, 이는 20개의 가상머신 클러스터를 16,200시간 동안 실행시켜야 한다.

클라우드 기반 솔루션의 신축성(elasticity)을 평가하기 위해, 프로토타입은 120개의 고성능 CPU를 가진 초대형 가상머신들로 두 번째 테스트를 시행하였다. 각 가상머신에는 12개의 쓰레드들이 실행되고 1,440개의 프로세스들이 병렬로 동작하고 있었다.

4.5.3 고객의 문제 해결:

클라우드 기반 솔루션에 대한 예상 비용은 자체 솔루션 비용의 절반 이하이다. 그 비용 견적은 추가적인 전기 또는 시스템 자체 솔루션의 관리 비용을 포함하지 않았기에 실제 비용 절감은 더 클 것이다. 데이터세트의 스토리지 또한 클라우드 기반이 될 것이다.

두 번째 테스트에서는 데이터베이스에서 SQL 쿼리와 락(lock) 경합에 관련된 성능 문제가 몇 개 감지되었으나 해결되었다. 이러한 문제는 현재의 자체 시스템에서는 감지될 수 없었을 수도 있다. 이 프로토타입은 이 기관이 실제 구축 전에 성능과 신축성관련 문제를 찾아내고 해결할 수 있도록 했다.

4.5.4 요구사항들과 기능:

이 프로토타입은 AGIS 소프트웨어와 데이터베이스를 위해 가상머신을 사용했다. 데이터베이스는 가상머신 내에서 실행되는 전통적인 데이터베이스로서 클라우드 데이터베이스가 아닌 완전한 관계형 데이터베이스이다. 스토리지는 각 100GB짜리 클라우드 스토리지 5개가 데이터베이스 서버에 연결되었다.

4.5.5 이식성에 대한 우려:

모든 가상머신들은 표준 운영 체제를 사용하고 있고 프로젝트에서 사용된 어떤 소프트웨어도 클라우드와 관련되지 않았다. 이러한 응용에서 이식성에 대한 고려사항은 가상머신 이미지들을 다시 만들거나 재설정하지 않고도 다른 제공자의 시스템으로 옮기는 것이다.

5 개발자 요구사항

개발자들을 위한 요구 사항들은 모든 유즈케이스 및 고객 시나리오들을 통해 도출된 것이다. 논의를 단순화하기 위해, 개발자들의 모든 요구 사항들을 여기에 나열하였다.

- **캐싱(Caching):** 클라우드 리소스에 대한 캐싱을 지원하는 프로그래밍 툴킷은 어플리케이션의 성능을 향상시킨다. 개발자는 캐시를 비우기 위한 API는 물론, 특정 개체 또는 리소스를 캐시에 넣을 때도 API가 필요할 수 있다.
- **중앙집중 로깅(Centralized Logging):** 개발자 자신의 역할 또는 그들이 작성하고 있는 어플리케이션들의 유형에 관계없이 로깅은 모든 개발자들의 공통된 요구사항이다. API는 로그의 쓰기, 로그 기록 검토, 로그 생성, 그리고 로그 파일을 열고 닫는 것 등을 지원해야 한다.
- **데이터베이스(Database):** 개발자는 클라우드 데이터베이스에 접근할 수 있는 수단이 필요하다. 클라우드 데이터베이스들은 설계가 매우 다양(일부는 스키마 기반이면서 관계형이고, 또 많은 경우에는 둘 다 아닌)하기 때문에 클라우드 어플리케이션의 개발자들은 각각의 어플리케이션의 요구에 따라 클라우드 데이터베이스 제공자를 선택한다. 데이터베이스 API는 기본적인 CRUD(Create, Read, Update, Delete) 기능들을 지원해야 한다.
- **식별자 관리(Identity Management):** 개발자는 ID를 관리하는 방법이 필요하다. 아주 간단한 경우라도 해당 사용자의 자격을 인증하는 수단이 있어야 한다. 하나의 어플리케이션이 여러 데이터 소스 및 응용프로그램들에 걸쳐 실행되는 경우에는 개발자가 각각의 사용자 ID를 연합 관리할 수 있는 수단이 필요하다. 연합 식별자 관리 시스템은 서비스와 사용자가 상대의 정보를 가지고 있지 않는 경우에도 특정 서비스에 액세스하려는 해당 사용자를 허용하도록 자격 증명을 생성할 수 있다. 식별자 관리를 위한 API는 자격 증명을 적절히 캐시하거나 삭제하여야 한다.
- **점대점(P2P) 메시징 :** 개발자들은 메시지를 큐에 추가하고 메시지들을 사용하는 API가 필요하다. 개발자가 메시지를 엿보는 것(메시지의 내용을 사용하지 않고 단지 검사만 함)을 허용하는 API도 필요하다.
- **Pub-Sub 메시징 :** 개발자는 메시지 큐잉 시스템의 주제(topic)들로 작업하는 API가 필요하다. 이 API는 개발자들이 주제에 메시지를 게시하거나 주제에서 메시지를 가져오는 것을 허용해야 한다.
- **원시 컴퓨팅 / 작업 처리 :** 개발자들은 하둡(Hadoop)식의 데이터마이닝과 같은 대용량 프로세싱 작업을 하기 위한 API가 필요하다. API는 개발자들로 하여금 처리 작업을 시작, 정지, 모니터 및 일시 중지하도록 허용해야 한다.
- **세션관리 :** 클라우드 환경에서는 사용자 세션을 관리하는 능력이 아주 중요하다. 클라우

드의 인프라는 노드 장애에 대비한 리던던시와 탄력성이 있다. 따라서 특정 클라우드 노드가 다운되더라도 세션은 유지되어야 한다. 세션 API는 사용자 세션의 현재 상태에 쉽게 접근하거나 조작할 수 있어야 한다.

- **서비스 발견(Service Discovery)** : 개발자는 어떤 클라우드 서비스가 사용 가능한 지 찾아내는 수단이 필요하다. 클라우드 서비스는 서비스의 유형에 따라 검색되고, API는 각 서비스 유형에 따라 적절한 추가적인 기능들을 제공해야 한다.
- **서비스수준협약(SLAs)** : 서비스 검색을 이용하는 개발자들은 검색된 코드의 서비스의 정책을 결정하는데 자동화된 방법이 필요하다. 이러한 API를 통해 개발자는 클라우드 서비스와 연동되며, 어플리케이션의 서비스수준 조건에 가장 적합한 서비스를 선택하고, 어플리케이션을 작성할 수 있다.
- **스토리지** : 개발자들은 클라우드 스토리지 서비스에 접근할 수 있는 방법이 필요하다. API는 데이터와 메타 데이터를 저장하고 꺼내오는 능력을 제공해야 한다.

다음의 표는 2.4.2장에서 논의된 5가지 API 유형들을 개발자의 요구사항들과 함께 비교하고 있다.

개발자 요구사항	일반 프로그래밍	배치 및 운용	클라우드 서비스	이미지와 인프라 관리	내부 인터페이스
캐싱				✓	✓
중앙집중 로깅	✓	✓	✓	✓	✓
데이터베이스			✓		✓
식별자 관리		✓	✓	✓	✓
점대점(P2P) 메시징			✓		
Pub-Sub 메시징			✓		
원시 컴퓨팅/작업 처리			✓	✓	✓
서비스 발견		✓	✓	✓	✓
세션 관리	✓		✓		
사용자수준협약	✓	✓	✓	✓	✓
스토리지		✓	✓	✓	✓

6 보안 시나리오

다른 분야에서도 마찬가지지만, 클라우드 컴퓨팅에서 보안은 아주 중요한 문제이다. 여기서는 설계자나 개발자들이 클라우드 컴퓨팅으로 옮겨 갈 때 고려해야 할 보안 이슈들을 강조하기 위한 것이다.

명심할 것은 클라우드 컴퓨팅을 적용한다고 해서 보안에 대한 어떤 새로운 위협이나 문제가 생기지는 않는다는 점이다. 보안의 관점에서, 클라우드 컴퓨팅은 클라우드 배포 모델에 관계없이 일관성, 투명성, 표준 기반 보안 프레임워크에 대한 필요성을 강조하는 이상적인 유즈케이스로 간주될 수 있다. 기업들이 클라우드 컴퓨팅으로 솔루션을 옮기거나 구축할 때, 이 일관된 보안 모델을 갖추는 것은 개발을 단순화하고 벤더 종속(lock-in)을 피하며 IT 투자를 유지하는 데에 필수적이다.

어떤 특정 기술적인 어려움과는 달리, 클라우드 컴퓨팅의 관점에서 보안을 고려할 때 가장 큰 차이점은 기업의 제어권 손실이다. 기업 내부의 자체 어플리케이션에서 민감한 데이터와 어플리케이션에 대한 접근 제어는 매우 중요하다. 클라우드 기반의 어플리케이션에서도 접근 제어는 동등하게 중요하지만, 인프라, 보안 플랫폼과 어플리케이션 등은 클라우드 제공자에게 직접적인 제어권이 있다.

이번 장에서는 아래의 순서에 맞추어 보안 주제들을 설명한다.

- **규정** : 규정은 기술적인 문제는 아니지만, 꼭 다뤄져야 한다. 법과 규정은 기능적 요구사항 보다 상위의 보안 요구사항을 정의한다.
- **보안 제어** : 비록 소비자가 모든 보안 제어를 필요로 할런지는 모르겠지만, 소비자들은 모든 보안제어가 가능한 인프라도 없이 보안과 관련해서 안심시키는 주장을 하는 클라우드 제공자들을 경계해야 한다.
- **보안 연합 패턴** : 이러한 보안 제어를 구현하기 위해서는 몇 가지 연합된 패턴들이 필요하다. 클라우드 제공자들은 기존의 보안 표준들을 통해서 이러한 패턴들을 구현해야 한다.

이러한 관점에서 사용자 유즈케이스 시나리오에 대한 논의를 다음 장에서 하도록 한다.

6.1 규정 (Regulations)

클라우드 컴퓨팅을 사용하는 데에 따른 모든 기술적인 이슈들을 제외하면 규제라는 엄격한 현실이 존재한다. 여러 가지 이유로, 세계의 정부들은 클라우드 컴퓨팅의 사용에 관해 우려를 하고 있다. 많은 나라들은 그 나라 밖에 위치한 물리적인 머신에 특정 데이터를 저장하는 것을 금지하는 엄격한 개인정보 법률을 가지고 있다. 이러한 법률을 위반 할 시에는 기업을 (또는, 경영진들을) 처벌하는 무거운 처벌 규정들이 종종 존재한다. 클라우드에 민감한

데이터를 저장하는 모든 기업과 기관은 그들의 클라우드 제공자가 특정 지역 밖에 위치한 물리적 서버에 데이터를 저장하지 않는다는 것을 증명할 수 있어야 한다.

정부기관을 포함하여 많은 무역 및 산업 단체들도 역시 규제들을 만든다. 비록 이런 규제들은 법에 의해 요구되지 않을 수는 있지만, 잘 알려진 관행들이다.

비슷한 우려가 클라우드에서 동작하는 어플리케이션들에게도 적용된다. 한 가상 머신이 클라우드에서 실행되고 있다면, 그 가상 머신에서 작동하는 어플리케이션은 민감한 데이터에 접근할 수 있는가? 비록 새로운 법률과 규정들이 지속적으로 만들어지고 있다 하더라도, 이것은 많은 나라들이 해결하지 못한 부분이다.

이러한 법률과 규정들을 따르는 것은 모든 다른 요구사항들에 비해 높은 우선순위를 가질 것이다. 한 신규 법률이 기관에게 그들의 자원을 이용하여 기능을 추가하는 정도가 아니라 어플리케이션의 인프라스트럭처를 변경하도록 요구할지도 모른다. 최고 정보 책임자(CIO) 사무실은 이런 변화를 관리해야 하고 새로운 법률과 규정들이 만들어지는 것에 대해 민감히 대처하여야 한다.

6.2 보안 제어

몇 개의 제어들은 모든 시스템들의 적절한 보안을 위해서 필요하다. 다음 장은 여러 유즈케이스들과 그들 각각에 따른 보안 요구사항들을 기술한다. 유즈케이스들에 대한 논의는 여기 정의된 제어권에 대한 유즈케이스들의 요구사항들과 관련이 있다.

보안 제어	설명
자산관리 (Asset Management)	클라우드 인프라를 구성하는 모든 물리적 혹은 가상의 하드웨어, 네트워크, 소프트웨어 자산을 관리할 수 있어야 한다. 여기에는 감사 및 규정 준수의 목적을 위해서는 자산에 대한 어떠한 물리적 혹은 네트워크 기반의 접근도 설명될 수 있어야 한다.
암호화: 키와 인증서 관리 (Cryptography : Key and Certificate Management)	모든 안전한 시스템은 암호화 키와 인증서를 적용하고 관리하는 인프라가 필요하다. 여기에는 작동할 때나 안 할 때나 정보 보안을 지원하는 표준 기반의 암호화 기능과 서비스를 적용하는 것이 포함된다.
데이터/스토리지 보안 (Data / Storage Security)	암호화된 형식으로 데이터를 저장하는 것이 가능해야 한다. 또한, 일부 소비자들은 그들의 데이터가 다른 소비자들의 데이터와 분리되어 저장되는 것을 요구할 것이다.
종점 보안 (Endpoint Security)	소비자들은 그들의 클라우드 자원에 대한 종점을 보호할 수 있어야 한다. 여기에는 네트워크 프로토콜과 디바이스 타입에 따라 종점을 제한할 수 있는 기능을 포함한다.
이벤트 감사와 보고	소비자들은 클라우드에서 발생하는 이벤트, 특히 시스템 오류와 보안 침해에 관한 데이터에 접근할 수 있어야 한다. 이벤트에

(Event Auditing and Reporting)	대한 접근은 과거의 이벤트들에 대한 인지능력과 신규 이벤트들의 발생에 대한 보고가 포함된다. 클라우드 제공자들은 적시에 이벤트들을 보고하지 못 할 경우, 자신들의 평판에 상당한 피해를 입게 될 것이다.
식별자, 역할, 접근제어와 속성 (Identity, Roles, Access Control and Attributes)	클라우드 기반 자원들에 대해 효과적으로 접근 제어를 구현하고 보안 정책을 시행하기 위해서는 일관성 있고 기계가 판독할 수 있는 방식으로 개인과 서비스들의 식별자, 역할, 권리와 그 밖의 다른 속성들을 정의할 수 있어야 한다.
네트워크 보안 (Network Security)	스위치, 라우터, 패킷 레벨에서 네트워크 트래픽의 안전을 확보할 수 있어야 한다. IP 스택 자체도 역시 안전을 확보할 수 있어야 한다.
보안 정책 (Security Policies)	일관성 있고 기계 판독 가능한 방법으로 정책을 정의하고, 접근 제어와 자원 할당, 그 밖의 다른 결정들을 지원하는 보안정책을 결정하고 적용할 수 있어야 한다. 정책을 정의하는 방법은 SLA와 라이선스들이 자동으로 적용될 수 있을 정도로 충분히 견고해야 한다.
서비스 자동화 (Service Automation)	보안 준수 감사를 지원하기 위해 보안 제어 흐름과 과정을 자동화된 방법으로 관리하고 분석할 수 있어야 한다. 여기에는 보안 정책이나 고객 라이선스 계약을 위반하는 모든 이벤트들을 보고해야 한다는 점도 포함된다.
작업량과 서비스 관리 (Workload and Service Management)	정의된 보안 정책과 고객 라이선스 계약에 따라서 서비스들을 구성하고, 배포하고, 모니터 할 수 있어야 한다.

아래는 이러한 제어권들에 적용될 수 있는 일부 표준들이다.

Security Control	Relevant Standard
암호화: 키와 인증서 관리 (Cryptography : Key and Certificate Management)	KMIP , the OASIS Key Management Interoperability Protocol (http://www.oasis-open.org/committees/kmip/)
데이터/스토리지 보안 (Data / Storage Security)	IEEE P1619 , developed by the IEEE Security in Storage Working Group (http://siswg.net)
식별자, 역할, 접근제어와 속성 (Identity, Roles, Access Control and Attributes)	SAML , the OASIS Security Assertion Markup Language (http://saml.xml.org/)
식별자, 역할, 접근제어와 속성 (Identity, Roles, Access Control and Attributes)	X.509 Certificates , part of the ITU Public Key and Attribute Certificate Frameworks Recommendations (http://www.itu.int/rec/T-REC-X.509)

보안 정책 (Security Policies)	XACML , the OASIS eXtensible Access Control Markup Language (http://www.oasis-open.org/committees/xacml/)
작업량과 서비스 관리 (Workload and Service Management)	SPML , the OASIS Service Provisioning Markup Language (http://www.oasis-open.org/committees/provision/)

6.3 보안 연합(federation) 패턴

연합은 많은 수의 독립된 자원들이 마치 하나의 자원처럼 동작하도록 하는 능력이다. 클라우드 컴퓨팅 자체가 자원들의 연합이고, 그래서 많은 자산들, 식별자들, 설정들, 그리고 클라우드 컴퓨팅 솔루션의 그 밖의 다른 세부 사항들은 클라우드 컴퓨팅을 실질적으로 만들기 위해서는 연합되어야 한다.

앞 절에서 언급했던 요구사항들은 다음의 연합 패턴들을 통해 구현된다.

- **신뢰** : 인증기관을 통해 두 단체가 신뢰 관계를 정의하는 능력을 말한다. 그 인증기관은 신임장(일반적으로 X.509 인증서)을 교환 할 수 있고, 그러면 이러한 인증서를 이용하여 메시지들의 안전을 확보하고 서명된 보안 토큰(일반적으로 SAML)을 생성할 수 있다. 연합된 신뢰는 다른 모든 보안 연합 패턴의 기초가 된다.
- **식별자 관리** : 사용자의 자격증(사용자 ID와 비밀번호, 인증서 등)을 받고, 그 사용자를 인증하는 서명된 보안 토큰을 반환하는 식별자 제공자를 정의하는 능력. 그 식별자 제공자를 신뢰하는 서비스 제공자는 비록 그 서비스 제공자가 사용자에게 대한 정보가 하나도 없더라도, 그 사용자에게 적절한 접근을 허용하는 토큰을 사용할 수 있다.
- **접근 관리** : 클라우드 자원들에 대한 접근을 관리하기 위한 보안 토큰을 검사하는 정책(일반적으로 XACML)을 만드는 능력. 자원들에 대한 접근은 한 가지 이상의 요인들에 의해 제어될 수 있다. 예를 들어, 어느 한 자원에 대한 접근은 어느 특정 역할을 담당하는 사용자들에게는 특정 프로토콜을 통해서만 그리고 하루 중 어떤 특정 시간만으로 제한될 수 있다.
- **단일 사인-온/사인-오프** : 신뢰기관으로부터의 자격증을 기반으로 로그인을 연합하는 능력. 특정 역할의 인증된 사용자가 있다면, 연합 단일 사인-온은 그 사용자가 한 어플리케이션에 로그인 한 후, 같은 신뢰기관을 신뢰하는 다른 어플리케이션에 대해서도 접근할 수 있도록 한다. 연합 단일 사인-오프도 이 같은 패턴을 따른다; 많은 경우에 사용자가 하나의 어플리케이션에 대해서 로그아웃 하면, 필히 다른 어플리케이션에 대해서도 로그아웃 되어야 할 것이다. 단일 사인-온 패턴은 식별자 관리 패턴에 의해 가능하다.
- **감사 및 규정 준수** : 하이브리드 클라우드를 포함하여, 여러 도메인들에 퍼져 있는 감사 및 규정 준수 데이터를 모으는 능력. 연합 감사는 SLAs와 규제 요구 사항들에 대한 준

수 보장과 문서화를 위해 필수이다.

- **설정 관리** : 서비스들, 어플리케이션들, 그리고 가상 머신들을 위한 설정 데이터를 연합하는 능력. 이런 데이터는 여러 도메인들 간에 걸친 접근 정책과 라이선스 정보를 포함할 수 있다.

기존의 보안 모범 사례가 클라우드에 적용되기 때문에, 제공자들은 위와 같은 연합 패턴들을 제공하기 위해 기존의 표준들을 사용해야 한다.

7 보안 유즈케이스 시나리오

이번 장에서는 보안 요구 사항을 만족하면서 클라우드 컴퓨팅을 사용하는 고객의 경험을 설명한다. 정의된 요구 사항들과 패턴들을 고려하면서 실제로 적용되는 클라우드 보안을 묘사하기 위한 실제 시나리오를 보도록 한다. 이 시나리오는 다양한 어플리케이션 유형, 운용(배포) 모델, 그리고 패턴과 역할을 다룬다.

7.1 클라우드에서의 컴퓨팅 파워

7.1.1 고객 시나리오:

어느 한 보험 회사는 그들의 정책 소유자(고객)와 그들이 받은 모든 재산 피해에 관한 데이터를 추출하는데 사용되는 보험청구 어플리케이션을 가지고 있다. 한 허리케인이 미국의 걸프 해안 지역을 공격하여, 막대한 재산 피해를 입힐 것으로 예상된다. 이것은 수많은 보험청구를 만들어낼 것이고, 따라서 기업의 IT 인프라에 엄청난 부하를 만들 것이다.

이 회사는 예상되는 수요를 처리하기 위한 가상 머신들을 제공하는 퍼블릭 클라우드 제공자를 이용하기로 결정했다. 이 회사는 회사의 인증된 대리인들만 접근이 허용되도록 하면서, 기업 자체의 시스템과 클라우드 제공자로부터 호스팅 받은 가상 머신들 사이의 접근을 제어해야만 한다. 또한, 회사는 클라우드 기반의 어플리케이션에 의해 생성된 모든 데이터도 회사 방화벽 안으로 안전하게 전송해야 한다. 마지막으로, 클라우드 제공자는 가상 머신이 종료할 때는 언제나 어플리케이션이나 관련 데이터의 흔적이 남아있지 않도록 보장해야 한다.

7.1.2 고객의 문제 해결:

퍼블릭 클라우드를 사용하면, 회사는 평소보다 몇 십 배 더 큰 규모의 작업량을 처리 할 수 있다. 상상할 수 있는 가장 높은 작업량을 처리하기 위해 충분한 여분의 시스템을 구매, 유지, 전원공급, 그리고 냉각하는 것은 경제적으로 비효율적이다. 회사는 그날 그날의 일상적인 작업을 처리 할 수 있는 자체 능력을 가지고 있지만, 필요할 때마다 자동으로 컴퓨팅 파워를 추가 할당 할 수 있다.

7.1.3 요구사항들과 제어:

다음은 이번 유즈케이스에 필요한 요구사항들과 관계되는 제어권이다.

요구사항	제어권(Controls)
특정 역할들로 제한하기 위한 비디오 접근	<ul style="list-style-type: none"> • 식별자, 역할, 접근 제어와 속성들 • 자산 관리 • 네트워크 보안
어플리케이션이나 데이터의 모든 흔적들은	

가상 머신이 종료되면 모두 제거되어야 한다.	• 워크로드와 서비스 관리
--------------------------	----------------

7.1.4 연합 패턴:

신뢰, 자산 관리, 설정 관리

7.1.5 역할

보험청구 조정자, 보험 에이전트, 감사자

7.2 클라우드 기반 개발과 테스트

7.2.1 고객 시나리오:

한 온라인 소매업자는 새로운 웹 2.0 프론트(상품진열) 어플리케이션의 개발이 필요하지만, 자사의 IT 직원과 기존의 자원들에게 부담을 주지 않으려 한다. 이 회사는 개발 도구와 소스 코드 저장소가 호스팅된 클라우드 기반의 개발 환경을 제공해주는 클라우드 제공자를 선택한다. 또, 새로운 어플리케이션이 많은 다른 타입의 머신들 및 대규모 워크로드와 상호작용 할 수 있도록 테스트 환경을 제공해주는 다른 클라우드 제공자도 선택한다.

클라우드 기반의 개발과 테스트를 다루는 두 제공자를 선택한 것은 연합이 중요하다는 것을 의미한다.

7.2.2 고객의 문제 해결:

개발자 관점에서, 클라우드로 호스트된 개발 툴들을 사용하면 각각의 개발자의 머신마다 툴들을 설치하고 설정하고 관리할 필요가 없어진다. 클라우드 제공자의 사이트에서 툴들이 일단 한번 설치되고 업데이트되면, 모든 개발자들은 자동으로 같은 버전의 툴들을 사용할 수 있다.

큰 규모의 빌드는 클라우드 제공자 인프라의 확장성을 이용할 수 있기 때문에 제품 빌드를 훨씬 더 빠르게 할 수 있다. 또한, 클라우드 기반의 빌드는 클라우드 기반 저장소에서 최신 소스 코드를 검색, 추출할 수 있다.

테스팅 관점에서, 회사가 전통적인 인터페이스로부터 웹 2.0 인터페이스로 바뀌어 감에 따른 서버의 추가적인 부담은 알 수 없다. 웹 2.0 인터페이스는 정적 웹 페이지보다 서버와 훨씬 더 많은 상호작용을 한다. 그래서 클라우드 환경에서 새로운 어플리케이션을 테스트 하면 테스트 팀은 새로운 인터페이스의 영향을 측정 할 수 있다.

클라우드 환경에서 새로운 어플리케이션에 대한 스트레스 테스트는 더 쉽다. 만약 테스트 팀이 1초마다 500개의 다른 머신으로 작업 요청이 들어왔을 때, 그 어플리케이션이 어떻게 수행하는지를 알고 싶다면, 클라우드는 500개의 가상 머신을 구동시키고 테스트할 수 있도록 한다. 서로 다른 VM 이미지를 사용하면, 많은 서로 다른 타입의 머신들(운영체제, 버전, 프로토콜 등)에서 어플리케이션을 테스트하기 편하다. 그 테스트 팀이 1,000개 머신이나 10,000개 머신으로 테스트하고자 할 때, 클라우드에서 그렇게 하는 것은 경제적이지만, 자체 인프라를 구축한다는 것은 매우 비경제적이다.

7.2.3 요구사항들과 제어:

다음은 이번 유즈케이스에 필요한 요구사항들과 이에 관계되는 제어권이다.

요구사항	제어권(Controls)
개발자 톨들은 하나의 중심지에서 설치되고 유지된다.	• 자산 관리
어플리케이션이나 데이터의 모든 흔적들은 가상 머신이 종료되면 모두 제거되어야 한다.	• 워크로드와 서비스 관리
단일 사인-온은 개발과 클라우드 테스트 모두에 적용된다.	• 암호화 • 종점 보안 • 식별자, 역할, 접근 제어와 속성들 • 네트워크 보안
소스 코드와 테스트 계획에 대한 접근 제어	• 자산 관리 • 식별자, 역할, 접근 제어와 속성들
빌딩과 테스트는 가상 머신을 자동으로 시작하고 종료해야 한다.	• 서비스 자동화
빌딩과 테스트는 가상 머신 사용과 성능에 대한 통계를 보고해야 한다.	• 이벤트 감사와 보고

7.2.4 연합 패턴:

신뢰, 식별자 관리, 접근 관리, 단일 사인-온, 감사 및 규정 준수, 설정 관리

7.2.5 역할

개발자, 테스터, 관리자, 감사자

7.3 클라우드에서의 저장공간

7.3.1 고객 시나리오

한 금융 투자 회사는 회사의 에이전트와 계열사에 대해 새로운 투자 상품을 출시한다. 여러 개의 비디오가 새로운 상품에 대한 이익과 특징들을 회사의 에이전트와 계열사에게 설명하

기 위해 만들어졌다. 이 비디오들은 용량이 매우 크고 수요에 따라 즉시 공급되어야 하기 때문에, 클라우드 환경에 저장하게 되면 기업 인프라의 사용 요구를 줄일 수 있다. 그러나 이 비디오들에 대한 접근은 면밀히 제어되어야 한다. 경쟁상의 이유로, 오로지 인증을 걸친 회사의 에이전트만이 비디오를 볼 수 있어야 한다. 이와 더불어 더 큰 제약 조건으로, 회사는 제품 출시 전까지 상품과 비디오에 대한 기밀을 유지해야 한다는 것이다.

이 회사의 결정은 안전하게 비디오를 호스팅하고 스트리밍 할 수 있도록 퍼블릭 클라우드 스토리지 제공자를 사용하기로 하는 것이다. 그 클라우드 솔루션은 그 회사의 보안 정책을 적용한 접근 제어 메커니즘으로 비디오들을 컨트롤해야 한다.

7.3.2 고객의 문제 해결

퍼블릭 클라우드 스토리지 서비스를 사용하면 고객들은 그들 데이터 센터의 물리적 자원들의 증가 없이 원하는 만큼의 대규모의 데이터 파일들과 대역폭 요청을 관리할 수 있다. 그러나 정부 규제와 기관(기업)의 요구사항은 보안이 필수적으로 중요하다는 것을 의미한다. 규정 준수를 보장 할 수 없는 퍼블릭 클라우드 스토리지 제공자는 그들이 제공하는 퍼포먼스와 가격, 확장성에 관계없이 고려 대상이 될 수 없다.

7.3.3 요구 사항과 제어권

다음은 이번 유즈케이스에 필요한 요구사항들과 이에 관계되는 제어권이다.

요구사항	제어권(Controls)
특정 역할에게만 제한적으로 제공되는 어플리케이션에 대한 접근	<ul style="list-style-type: none"> • 식별자, 역할, 접근 제어 및 속성들 • 자산 관리 • 네트워크 보안 • 정책들
클라우드에 저장된 데이터는 안전해야 한다.	<ul style="list-style-type: none"> • 암호화 • 데이터 / 스토리지 보안
클라우드에 저장된 데이터는 그 회사의 방화벽 안으로 반납되어야 한다.	<ul style="list-style-type: none"> • 암호화 • 데이터 / 스토리지 보안 • 종점에 대한 보안 • 네트워크 보안

7.3.4 연합 패턴

신뢰, 식별자 관리, 접근 관리, 감사와 규정 준수

7.3.5 역할

비디오 프로듀서, 회사 대리인(에이전트), 회사 계열사, 감사자, 규제 기관

7.4 상호참조: 보안 제어와 고객 시나리오

다음 테이블은 보안 제어와 고객 시나리오 사이의 관계를 요약한 것이다.

보안 제어	클라우드 컴퓨팅 파워	클라우드 기반 개발과 테스트	클라우드 스토리지
자산관리	✓	✓	✓
암호화: 키와 인증서 관리		✓	✓
데이터/스토리지 보안			✓
종점 보안		✓	✓
이벤트 감사와 보고		✓	
식별자, 역할, 접근제어와 속성	✓	✓	✓
네트워크 보안	✓		✓
보안 정책			✓
서비스 자동화		✓	
작업량과 서비스 관리	✓	✓	

7.5 상호참조: 보안 연합 패턴과 고객 시나리오

다음 테이블은 보안 연합 패턴과 고객 시나리오 사이의 관계를 요약한 것이다.

보안 연합 패턴	클라우드 컴퓨팅 파워	클라우드 기반 개발과 테스트	클라우드 스토리지
신뢰	✓	✓	✓
식별자 관리		✓	✓
접근 관리	✓	✓	✓
단일 사인-온		✓	
감사와 규정 준수		✓	✓
설정 관리	✓	✓	

8 서비스 수준 협약(SLAs)

클라우드 제공자와 소비자 사이의 관계는 반드시 서비스 수준 협약(SLA)을 가지고 설명되어야 한다. 클라우드 소비자들은 클라우드 제공자들이 그들의 인프라 서비스의 일부를 제공할 것이라 기대하기 때문에 이러한 서비스들을 정의하는 것과 어떻게 제공되고 또 어떻게 사용되어지는 것인가는 아주 중요하다.

SLA는 제공자에 대한 소비자 신뢰의 기본이다. 하나의 잘 작성된 SLA는 제공자의 평판을 결정짓는다.

소비자와 제공자간에 관계를 정의하는 문장과 더불어 SLA는 클라우드 서비스에 대한 객관적이고 측정 가능한 조건들을 정의하는 서비스 수준 목표(SLOs)를 포함한다. 클라우드 소비자들은 그들의 사업 목표에 비추어 클라우드 서비스 제공자를 선택하기 위한 SLA와 SLO 항목들을 반드시 평가해야 한다.

클라우드 서비스 소비자가 제공자 SLA의 모든 항목들을 완전한 이해해야 한다는 것과, 소비자들이 어떤 협약에 서명하기 전에 그들의 기관에서 필요한 요구들을 고려하는 것은 매우 중요하다.

8.1 서비스 수준 협약(SLA)이란 무엇인가?

SLA는 클라우드 서비스 제공자와 클라우드 서비스 소비자 간의 계약관계를 정의한다. SLA는 아래와 같은 여러 가지를 내포하고 있다.

- 제공자가 전달하고자 하는 여러 서비스들의 집합
- 각 서비스의 완전하고 구체적인 정의
- 제공자와 소비자의 책임들
- 제공자가 약속된 대로 서비스를 제공하고 있는지 여부를 결정하기 위한 측정기준들의 집합
- 서비스를 모니터링 하기 위한 감사 방법
- SLA 항목이 충족 되지 않았을 경우 제공자와 소비자에게 가능한 대안들
- SLA가 시간이 지남에 따라 어떻게 변화하는지(SLA 변경조건)

시장에는 두 가지 타입의 SLA가 있다: 기존에 개발된 일반적인 협약서와 소비자의 구체적인 요구를 만족시키기 위해 제공자와 소비자 간에 상호 협의를 거친 협약서가 있다. 중요한

데이터와 어플리케이션을 가진 어떤 소비자가 첫 번째 타입을 사용할 수 있을 것 같진 않다. 그러므로 SLA(그리고 일반적으로 클라우드)에 접근하기 위한 소비자의 첫 번째 단계는 얼마나 그들의 데이터와 어플리케이션들이 중요한지 결정하는 것이다.

대부분의 퍼블릭 클라우드 서비스들은 협상이 불가능한 SLA를 제공한다. 이러한 제공자들로부터 요구사항이 만족되지 못한 소비자는 두 가지 대안을 가진다.

1. 다음 달은 서비스를 무료로 사용할 수 있는 권리를 받는다. (*이번 달* 요금은 완전히 지불한 후) 또는
2. 서비스 사용을 중단한다.

매우 중요한 애플리케이션이나 데이터의 경우에는 이러한 조항들이 있는 SLA는 결코 수용될 수 없다. 반면에, 이러한 조항들을 가진 SLA는 상호 협약된 SLA 하에서 제공되는 클라우드 서비스 보다 매우 저렴할 것이다.

8.2 서비스 수준 목표

SLO는 정확하고 측정 가능한 서비스 성격을 정의한다. 여기에 몇 가지 SLO에 대한 예시가 있다.

- 시스템은 10개 이상의 보류(pending) 요청을 가져서는 절대 안 된다.
- 요청에 대한 처리는 3초 이내이어야 한다.
- 읽기 요청에 대한 데이터 스트리밍은 2초 이내 시작되어야 한다.
- 최소한 5개의 VM 인스턴스들이 99.99999%의 가용성을 가져야 하고, 제공자의 3개의 데이터 센터에서 각각 최소 1개 이상의 인스턴스들이 이용 가능해야 한다.

당연히 서로 다른 서비스 수준 목표들이 다양한 사용 케이스들, 어플리케이션들 그리고 데이터 타입에 적용될 것이다. SLO는 다양한 SLO의 상대적인 중요성을 나타내기 위한 긴급성 등급을 또한 포함 할 수 있다. 만약 클라우드 제공자가 응답시간과 가용성 SLO들을 둘 다 같이 만족시키지 못 한다면 소비자는 응답시간보다 가용성이 더 중요하다는 지표간의 우위를 매길 수 있다.

다양한 역할들은 또한 적용되는 SLO에 영향을 끼친다. 예를 들어, 클라우드 제공자에 의해 호스트 되고 최종 소비자가 접근하며 클라우드 소비자에 의해 작성된 한 어플리케이션을 고려 해보자. 만약 그 어플리케이션과 그 어플리케이션의 데이터가 같은 클라우드 제공자에 의해 호스트 되면, 그 어플리케이션은 제공자의 데이터 센터를 떠나지 않아도 그 데이터에 접근할 수 있을 것이다. 클라우드 소비자는 어플리케이션이 그 어플리케이션의 데이터에 접

속할 때마다 매우 빠른 응답 시간을 기대할 것이다. 반면에 이 소비자는 최종 소비자가 웹을 통하여 이 어플리케이션에 접속 할 때에는 상대적으로 낮은 응답시간 기대치를 가질 것이다.

8.3 서비스 수준 관리

서비스의 성능을 측정하고 모니터링하지 않는다면 SLA 항목들이 충족되고 있는지 또는 아닌지를 파악하는 것은 불가능하다. 서비스 수준 관리(Service Level Management)는 어떻게 이런 성능 정보가 취합되고 다루어지는지에 관한 것이다. 서비스의 측정은 SLA의 서비스 수준 목표들에 근거한다.

클라우드 제공자는 그들의 인프라에 대한 결정을 하기 위해 서비스 수준 관리를 사용한다. 예를 들어, 제공자는 특정 서비스에 대한 성능이 고객의 요구사항들을 간신히 만족시킨다는 것을 알 수도 있다. 이 제공자는 물리적인 하드웨어를 더 추가하거나 대역폭을 재할당 함으로써 그러한 상황에 대응할 수도 있다. 그러나 만약 한 소비자에게 더 많은 자원들을 추가함으로써 또 다른 소비자의 SLA 항목을 만족시키는 것이 불가능해진다면, 제공자는 다른 고객의 희생을 통해 한 고객을 만족시키기로 결정할 수도 있다. 서비스 수준 관리의 목표는 제공자들로 하여금 사업 목적들과 기술적 현실에 근거하여 결정을 내리도록 도와주는 것이다.

클라우드 소비자는 클라우드 서비스들을 어떻게 사용할지에 대한 결정을 내리기 위해서 서비스 수준 관리를 사용한다. 예를 들어서, 한 소비자가 어느 특정 VM의 CPU 이용률이 90%이상인 것을 알았다. 이에 따라 이 소비자는 또 다른 VM을 시작시킬 수도 있다. 그러나 만약 소비자의 SLA에 처음 100개까지의 VM 이용 가격 보다 추가로 이용하는 VM에 대한 가격이 더 높게 책정되어 있다면 이 소비자는 그 추가 가격을 지불하면서 새로운 VM을 생성하는 결정을 하지 않을 수도 있다. 제공자와 마찬가지로 서비스 수준 관리의 소비자가 어떻게 클라우드 서비스를 사용할 것인지에 대한 결정을 내리는데 도움을 준다. (가능하면 자동적으로)

8.4 SLA에 대한 고려사항들

소비자들이 SLA에 필요한 항목들이 무엇인지를 결정하는 데는 그들이 고려해야 할 몇 개의 요인들이 있다.

8.4.1 사업 수준 목표

만약 기관이 사업 수준 목표(Business Level Objectives)를 아직 정의하지 않았다면 SLA 항목들에 대한 검토는 무의미 하다. 소비자는 그 기관의 목적에 따라 제공자들과 서비스들을 선택해야 한다. 만약 기관이 처음부터 왜 이 서비스를 사용할 것인지에 대해 정의를 하지 않았다면 정확하게 무슨 서비스를 사용할 것이라고 정의하는 것은 가치가 없다.

클라우드 컴퓨팅 관점에서 볼 때 가장 어려운 문제는 기술적인 이슈라기보다는 기관 자체의 정치적인 이해 문제이다. 한 기관에서 모든 부서들이 목표들에 동의하게 만들다 보면 어떤 그룹은 예산 삭감을 받아들이고, 다른 어떤 그룹은 그들의 인프라에 대한 통제권을 잃어버릴 수도 있고, 또 그 이외 어려운 결정들이 따를 수도 있다. 이러한 어려움들에도 불구하고, 기관은 사업 수준 목표가 정의되기 이전에는 클라우드 컴퓨팅(또는 다른 어떤 기술)을 최대한 활용할 수 없다. 소비자들은 그들이 *어떻게* 클라우드 컴퓨팅을 사용할 것인지 결정하기 전에 그들이 *왜* 클라우드 컴퓨팅을 사용하는지를 반드시 알아야 한다.

8.4.2 제공자와 소비자의 책임들

비록 일반적으로 SLA는 제공자의 책임을 정의하는 것이라고 생각하지만, 소비자의 책임들에 대한 부분 또한 명확해야 한다. 소비자의 책임들에는, 시스템 사용량 제한, 저장할 수 있는 데이터 타입에 대한 제한, 또는 제공자 시스템에 사용된 모든 소프트웨어의 유효한 라이선스에 대한 책임 등이 포함 될 수 있다. 제공자와 소비자 간의 책임 균형은 서비스 형태에 따라 다양하다. 예를 들면, 클라우드 제공자는 SaaS(Software as a Service)에 대한 대부분의 책임들을 가진다. 반면에 라이선스 있는 소프트웨어를 포함하고 민감한 데이터를 가진 VM의 경우에는 더 많은 책임들이 그것을 사용하고 관리하는 소비자에게 있다.

8.4.3 비즈니스 연속성과 재해 복구

많은 소비자들은 비즈니스 연속성 때문에 클라우드 컴퓨팅을 사용한다. 몇몇의 소비자들은 백업을 위해서 여러 클라우드 시스템에 소중한 데이터의 사본을 저장한다. 다른 소비자들은 회사 자체 내부 데이터 센터들이 부하들을 처리할 수 없을 때 클라우드버스팅(Cloud bursting)을 이용한다. 클라우드 컴퓨팅은 자체 시스템이 다운되었을 때 기관 운영을 유지시키기 위한 큰 가치를 가지고 있다. 그러나 만약 클라우드 제공자 자신이 적절한 연속성과 재해 복구 절차들을 가지고 있지 않다면 그러한 문제들은 아무 소용이 없다. 소비자들은 그들의 클라우드 제공자들이 재난 발생 시 적절한 보호책을 가졌다는 것을 확인해야 한다.

8.4.4 시스템 중복

많은 클라우드 제공자들은 거대한 여분(redundant)의 시스템들을 통해서 그들의 서비스를 제공하고 있다. 그러한 시스템들은 하드 드라이브들 또는 네트워크 연결들 또는 서버들의 오류가 일어날 경우에 대비해 설계되어서 소비자들이 어떠한 사용중단도 경험하지 않도록 하고 있다. 연속적으로 사용 가능해야 하는 어플리케이션과 데이터를 옮기는 소비자들은 그들의 제공자의 시스템 중복과 여분을 고려해야 한다.

8.4.5 유지관리

제공자들은 그들의 인프라를 그들 자신이 유지(관리)함으로써 소비자들을 시스템 유지관리에 대한 문제로부터 해방시켰다. 그러나 소비자들은 그들의 제공자들이 어떻게 또는 언제 유지관리 업무들을 할 것인지에 대한 이해를 하고 있어야 한다. 소비자의 서비스들이 얼마

동안 사용할 수 없는 것인지? 그들의 서비스들이 이용 가능한 하겠지만 훨씬 더 낮은 성능을 가질 것인지? 이 유지관리가 소비자의 어플리케이션에 영향을 미칠 가능성이 있다면 업데이트된 서비스를 대상으로 소비자들은 그들의 어플리케이션들을 테스트할 기회를 가질 수 있는지? 유지관리는 소프트웨어만 아니라 하드웨어 등 모든 유형의 클라우드 서비스에 영향을 미칠 수 있다는 점을 유념해야 한다.

8.4.6 데이터의 위치

많은 유형의 데이터에 대한 물리적인 위치는 제한되어 있다. 예를 들어서, 많은 나라들에서는 그들 국가의 국민들에 대한 개인 정보를 국경 밖의 어떤 시스템에도 저장하는 것을 금지하고 있다. 만약 클라우드 제공자가 소비자의 데이터를 특정 위치에만 저장되도록 보장할 수 없다면 소비자는 제공자의 서비스들을 사용할 수 없다. 만약 클라우드 서비스 제공자가 데이터 위치 규정을 적용할 것을 약속한다면, 소비자는 반드시 제공자가 그러한 규정을 잘 따르고 있는지 제공자에 대한 감사를 시행할 수 있어야 한다.

8.4.7 데이터 압수

사법 당국이 호스팅 업체의 자산을 압류한 몇 가지 잘 알려진 예가 있어왔다. 비록 법 집행 기관은 특정 소비자와 연관된 데이터와 어플리케이션들을 표적으로 하겠지만 다중소유(multi-tenant)라는 클라우드 컴퓨팅의 속성상 다른 소비자에게도 영향을 미칠 수 있다. 비록 SLA가 커버 할 수 있는 것이 한계가 있을 지라도, 소비자들은 제공자들에게 적용된 법 조항들을 고려해야 한다. 소비자들은 그들의 데이터와 어플리케이션들을 백업하기 위해서 서드파티(제3자)를 이용하는 것을 또한 고려해야 한다.

8.4.8 제공자의 불이행

어떤 클라우드 제공자라도 사업을 접거나 또는 다른 사업자에게 인수될 수 있는 잠재적 가능성을 가진다. 소비자들은 제공자들의 재정적인 건전성을 고려해야 하고 만약 제공자가 문을 닫을 것을 대비해서 연속성을 가지게 하는 계획들을 만들어야 한다. 게다가 만약 소비자의 사용료 계정이 연체되거나 분쟁의 소지를 가지고 있다면 소비자의 데이터와 어플리케이션의 접근에 대한 제공자들의 정책들을 명확하게 알고 있어야 한다.

8.4.9 법적 관할권

소비자들은 그들이 고려한 모든 클라우드 제공자들에게 적용될 법 조항들을 이해해야 한다. 예를 들어서, 클라우드 제공자는 클라우드 제공자의 서비스들을 사용하는 모든 데이터 또는 어플리케이션들을 모니터할 수 있는 권리를 보유한 국가에 근거지를 둘 수도 있다. 소비자의 데이터와 어플리케이션들의 특성을 감안할 때, 이러한 점은 받아들일 수 없을지도 모른다.

8.4.10 클라우드 중개인과 재판매업자

만약 클라우드 제공자가 실제로 또 다른 클라우드 제공자의 재판매업자 또는 중개인이라면, SLA 조항들은 중개인 또는 재판매업자 또는 제공자의 시설들에 문제가 발생할 경우를 고려하여 책임과 의무에 대한 모든 사항들을 명확히 하여야 한다.

8.5 SLA 요구사항

8.5.1 보안

일반적인 요구사항으로서의 보안은 본 문서의 섹션 6과 7에 자세하게 논의 되었다. SLA의 보안관련 사항들은 섹션 6의 보안 제어와 연합 패턴을 고려하여 작성되어야 한다. 클라우드 소비자는 그들의 보안 요구사항과 어떤 제어와 연합 패턴이 그러한 요구조건을 만족시키기 위해 필수인지 반드시 이해해야 한다. 또한 클라우드 제공자들은 적절한 제어와 연합 패턴들을 가능하게 하기 위해서 소비자들에게 어떤 것을 제공해야 하는지 파악하여야 한다.

8.5.2 데이터 암호화

만약 소비자가 클라우드 안에 대단히 중요한 데이터를 저장하고 있다면, 데이터가 이동하거나 저장되어 있는 동안 암호화시키는 것은 중요하다. 암호화 알고리즘과 접근 제어 정책에 대한 세부사항들은 SLA에서 구체화 되어야 한다.

8.5.3 개인정보보호

기본적인 개인정보보호 문제는 데이터 암호화, 유지, 삭제와 같은 요구사항들로 언급되었다. 추가로, SLA는 다중 소유자 환경에서 클라우드 제공자가 어떻게 데이터들과 어플리케이션들을 분리시킬 수 있는지 명확히 해야 한다.

8.5.4 데이터 유지와 삭제

많은 기관들은 데이터를 일정 기간 동안 보관해야 한다는 법적인 요구사항들을 가진다. 어떤 기관들은 또한 일정 기간 후에는 데이터의 삭제를 필요로 한다. 클라우드 제공자들은 그들이 이러한 정책들에 부합하는지를 증명할 수 있어야 한다.

8.5.5 하드웨어 삭제 및 파괴

데이터 유출의 공통적인 원인은 하드웨어의 부적절한 폐기이다. 만약 클라우드 제공자의 하드 드라이브가 오류가 났다면, 그 디스크의 플래터들은 드라이브가 폐기되거나 재활용되기 전에 반드시 제로 상태가 되어야 한다. 이와 유사하게, 많은 클라우드 제공자들은 소비자가 VM의 전원을 끄면 메모리 공간을 제로의 상태로 만들어 주는 추가 보호 기능을 제공한다.

8.5.6 규정 준수

많은 유형의 데이터와 어플리케이션들은 규정들의 영향을 받는다. 그러한 규정들 중 몇몇은 법 조항들(미국의 의료 기록에 대한 HIPAA)이고, 반면에 다른 것들은 산업 관련(신용카드들을 받는 소매업자에 대한 PCI DSS)된 것이 있다. 만약 규정들이 반드시 시행되어야 한다면, 클라우드 제공자들은 그들이 규정을 준수하고 있는지 증명할 수 있어야 한다.

8.5.7 투명성

몇몇 클라우드 제공자들의 SLA에는, 제공자들이 SLA의 항목들에 맞게 시행하지 못한 경우 이를 증명해야 할 부담을 소비자가 갖는다는 조항이 있다. 한 제공자의 서비스가 몇 시간 동안 다운될 수도 있지만, 이를 증명하지 못하는 소비자는 어떤 종류의 보상도 받을 자격이 없다.

중요한 데이터와 어플리케이션에 대해서 제공자들은 SLA 항목에 위배 되었을 경우 소비자들에게 능동적으로 통지해야 한다. 이것은 보안사고 뿐만이 아니라 서비스 중단과 성능문제와 같은 인프라적인 문제도 포함한다.

8.5.8 인증

특정 유형의 데이터와 어플리케이션들에 적용되는 많이 다양한 인증들이 있다. 예를 들면, 소비자는 그들의 클라우드 제공자가 ISO 27001 인증을 받아야 한다는 요구사항을 가질 수도 있다. 제공자는 그들의 인증을 증명하고 그것을 최신 상태로 유지 하려는 책임을 가져야 한다.

8.5.9 핵심 성과 지표에 대한 전문 용어

가동시간(uptime)이란 용어는 다양한 방법으로 정의 될 수 있다. 종종 이 정의는 제공자의 구조와 관련이 있다. 만약 제공자가 여섯 대륙에 데이터 센터를 가지고 있다면 가동시간이란 어느 특정 한 데이터 센터에 적용되나? 아니면 모든 데이터 센터에 적용되는 것인가? 만약 오직 이용 가능한 데이터 센터가 다른 대륙에 있다면 가동시간은 별 의미가 없고 수용될 수 없을 것이다. 설상가상으로 다른 클라우드 제공자들은 그들의 구조에 특화된 정의를 사용할 것이다. 이러한 점은 클라우드 서비스들을 비교하는 것을 어렵게 한다.

다양한 핵심 성능 지표들에 대한 일련의 산업적 용어 정의는 특히 SLA들을 비교하기 쉽게 한다. (그리고 일반적으로는 클라우드 서비스들을)

8.5.10 모니터링

만약 SLA 항목들을 만족시키는데 실패한 경우 재정적이거나 법률적인 문제들이 발생한다면, 누가 제공자(또는 소비자가 책임을 같이 져야 하는지)의 성능을 감시해야 하는가라는

질문이 중요해진다. 제공자들은 가능한 한 폭넓은 용어와 조항으로 가동시간을 정의하려 할 것이고, 반면에 소비자들은 발생한 모든 시스템 문제를 가지고 제공자를 비난하려고 할 것이다. 이 문제에 대한 가장 최선의 해결책은 제공자의 성능을 모니터 하는 중립적인 서드-파티(third-party) 기관이다. 이렇게 하면 제공자가 단독 재량으로 장애를 보고 하거나 또는 소비자가 발생한 장애를 증명하는 책임을 가질 때 발생할 수 있는 서로 간의 이해관계 문제를 해결할 수 있다.

8.5.11 감사 능력

많은 소비자 요구 사항들은 법적 규제 또는 업계 표준들의 준수를 포함한다. 소비자는 발생하는 모든 위반들에 대해서 책임져야 하기 때문에 소비자에 의한 제공자의 시스템들과 절차들에 대한 감사는 필수적으로 가능해야 한다. SLA는 어떻게 그리고 언제 그러한 감사를 실시할지를 명확하게 해주어야 한다. 감사는 불편하고 비싸기 때문에 제공자들은 대부분 제한을 두거나 비용을 청구하려고 할 것이다.

8.5.12 측정항목

사실에 대한 감사이고 발생한 것에 대한 모니터링이기 때문에 모니터링과 감사는 실재하는 어떤 것을 요구한다. SLA의 측정항목은 객관적으로 모호하지 않고 분명하게 정의 되어야 한다. 클라우드 소비자들은 그들의 어플리케이션과 데이터의 특성에 따라 끝없이 다양한 측정(항목)을 가질 것이다. 비록 모든 측정항목의 나열은 불가능할 지라도 가장 공통적인 것들은:

- **처리량(Throughput)** - 얼마나 빠르게 서비스가 반응하는지
- **신뢰성(Reliability)** - 얼마나 자주 서비스가 사용가능한지
- **부하 조절(Load balancing)** - 언제 탄력성이 시작할지 (예를 들어, 새로운 VM의 부팅 또는 소멸)
- **내구성(Durability)** - 데이터가 손실될 수 있는 가능성
- **탄력성(Elasticity)** - 한계가 명확한 내에서 주어진 자원의 확장성(예를 들어, 최대 저장 공간 또는 대역폭)
- **선형성(Linearity)** - 부하의 증가에 따른 시스템의 성능
- **민첩성(Agility)** - 소비자 자원 부하의 증감에 따라 얼마나 빠르게 제공자가 대응하는지
- **자동화(Automation)** - 사람의 개입 없이 처리되는 제공자에 대한 요청의 비율

- **고객 서비스 응답 시간(Customer service response times)** - 얼마나 빠르게 제공자가 서비스 요청에 반응하는지. 이것은 클라우드의 셀프서비스와 온디맨드 측면에서 원가가 잘못되었을 때 요구되는 사람의 개입을 의미함.

8.5.13 컴퓨터가 인식할 수 있는 SLAs

컴퓨터로 인식할 수 있는 SLA는 동적으로 클라우드 제공자를 선택 할 수 있는 자동화된 클라우드 브로커를 가능하게 한다. 클라우드 컴퓨팅의 기본 특징 중에 하나는 온디맨드 셀프 서비스이다; 자동화된 클라우드 브로커는 사용자 요청에 따라 클라우드 제공자를 선택함으로써 이러한 특징을 확장 할 수도 있다. 이 브로커는 소비자에 의해 정의된 비즈니스 기준에 근거하여 클라우드 제공자를 선택 할 수도 있다. 예를 들어, 소비자의 정책은 중개인이 일부 업무에 대해서 가장 저렴한 제공자를 사용해야 한다고 하는 반면 다른 업무에 대해서는 가장 보안이 강한 제공자를 사용하도록 할 수 있다. 비록 이러한 요구사항에 대한 시장의 수요가 전개되기까지는 일정 시간이 필요할 것이나 SLA 표준화를 위한 작업은 이점을 염두에 두어야 한다.

8.5.14 인간의 상호작용

비록 사용자 주문형 셀프서비스는 클라우드 컴퓨팅의 가장 기본적인 특징 중에 하나이나 오직 인간의 상호 작용으로만 해결될 수 있는 문제들이 항상 있을 것이라는 사실은 계속 존재한다. 이러한 상황들은 거의 발생하지 말아야 하지만 많은 SLA들은 제공자의 지원 요청들에 대한 대응 보장을 포함 할 것이다. 일반적인 보증들은 소비자들이 몇 번의 요청을 할 수 있으며, 그들의 비용, 그리고 얼마나 빨리 제공자가 응답할 것이란 부분을 포함한다.

8.6 신뢰성에 대한 언급

신뢰성에 관해 논의에서 일반적인 측정은 제공자가 제공하는 숫자 “9”들의 개수만큼 이루어지고 있다. 예를 들어, 5개 9의 신뢰성은 서비스가 이용 가능한 시간이 99.99999% 하다는 것을 의미하고, 이것은 모든 시스템에서 매 12달 중 5분 동안 장애가 있다는 것으로 해석이 된다. 이러한 측정의 한 가지 문제는 무엇이 장애인지 명확한 정의가 없다면 이러한 측정은 빠르게 의미가 없어진다는 것이다. (특히 클라우드 제공자가 사고에 동작정지를 포함시키면 더 의미가 없어진다.)

9들의 모호한 특징을 뛰어넘어서, 많은 클라우드 서비스 제공들이 다른 클라우드 서비스 제공들의 위에 구축된다는 점은 중요하게 고려되어야 한다. 여러 인프라들을 조합하는 능력은 매우 큰 유연성과 파워를 제공하지만, 각각의 추가적인 제공자들은 시스템의 신뢰성을 다소 떨어지게 만든다. 만약 한 클라우드 제공자가 두 번째 클라우드 제공자의 스토리지 서비스와 세 번째 클라우드 제공자의 데이터베이스 서비스를 기반으로 서비스를 제공하고, 그리고 이러한 제공자들이 모두 숫자 9가 5개 되는 신뢰성을 제공한다면, 전체 시스템의 신뢰성은 숫자 9가 5개인 경우보다 떨어진다. 이 서비스는 첫 번째 클라우드 제공자의 시스템이 다운

이 될 때 사용이 불가능하고 역시 두 번째나 세 번째 제공자의 시스템에 문제가 생겼을 때 마찬가지로 사용이 불가능하다. 더 많은 클라우드 제공자가 관련될수록 시스템 장애시간은 더욱 늘어날 것이다.

마지막으로, 클라우드 제공자의 숫자가 늘어날수록 외부요인의 수는 또한 늘어난다. 만약 VM과 클라우드 데이터베이스가 같은 데이터 센터에 있다면 VM과 데이터베이스 사이에 통신은 네트워크 접속을 필요로 하지 않는다. 반면에, 클라우드 데이터베이스가 또 다른 제공자에 의해 공급된다면, VM 사이에 이용 가능한 대역폭과 데이터베이스는 전반적인 시스템의 신뢰성과 성능에 영향을 미친다. 두 클라우드 제공자들의 클라우드가 모두 잘 작동할 수도 있지만, 만약 그들 사이의 네트워크 연결이 실패한다면 모든 시스템이 다운 될 것이다.

요약하면, 클라우드 서비스의 신뢰성을 평가할 필요가 있는 모든 소비자는 직접적이든 간접적이든, 서비스를 제공하는 클라우드 제공자에 대해서 되도록 많이 알아야만 한다.

8.7 상호 참조: SLA 요구사항들과 클라우드 서비스 모델

다음 표는 여기에 나열된 SLA 요구사항과 함께 섹션 2.1.1에서 나온 세 가지의 NIST 서비스 모델들의 상호 참조 모델의 상호 참조이다.

요구사항	플랫폼형 서비스(PaaS)	인프라형 서비스(IaaS)	소프트웨어형 서비스(SaaS)
데이터 암호화	✓	✓	
개인 정보보호	✓	✓	✓
데이터 유지와 삭제		✓	✓
하드웨어 삭제 및 파괴		✓	✓
규정 준수	✓	✓	✓
투명성	✓	✓	✓
인증	✓	✓	✓
핵심 성과지표에 대한 전문용어		✓	✓
측정항목	✓	✓	✓
감사능력	✓	✓	✓
모니터링	✓	✓	✓
컴퓨터가 인식할 수 있는 SLA		✓	

8.8 상호참조: SLA 요구사항들과 유즈케이스 시나리오

최선으로, SLA는 클라우드 소비자들과 제공자 양쪽 모두의 이해관계들을 보호한다. 주어진 SLA가 모든 소비자들의 요구들을 충족시키지 못하는 것과 같이 모든 요구사항은 모든 고객 시나리오들에 적용되지는 않는다. 다음 표는 여기서 열거된 SLA 요구사항들을 섹션 3의 7가지 유즈케이스들과 상호 참조 한 것이다.

요구사항	최종 사용자와 클라우드	기업과 클라우드 와 최종 사용자	기업과 클라우드	기업과 클라우드 와 기업	프라이빗 클라우드	클라우드 벤더 교체	하이브리 드 클라우드
데이터 암호화			✓				
개인 정보보호	✓	✓	✓	✓	✓	✓	✓
데이터 유지와 삭제			✓	✓			✓
하드웨어 삭제 및 파괴			✓	✓			✓
규정 준수	✓	✓	✓	✓	✓	✓	✓
투명성	✓	✓	✓	✓	✓	✓	✓
인증	✓	✓	✓	✓	✓		✓
핵심 성과지표에 대한 전문용어			✓	✓	✓	✓	✓
측정항목	✓	✓	✓	✓	✓		✓
감사능력	✓						
모니터링	✓	✓	✓	✓	✓		✓
컴퓨터가 인식할 수 있는 SLA				✓			

9 결론과 추천

클라우드 컴퓨팅은 가상화, SOA, 그리고 Web 2.0을 포함하는 산업계의 많은 트렌드를 구축하고 보완한다. 따라서 본 문서에서 언급된 많은 요구사항들에 대한 표준은 이미 존재한다. 계속 진행되어 가면서, 고객의 요구를 만족시키는 현존하는 표준을 명시하고, 진행 중인 표준화 작업을 통합하며, 현존하는 표준으로 만족이 안 되는 갭을 찾아내기 위하여 우리는 커뮤니티 형태로 함께 일할 것이다.

본 문서는 1400명이 넘는 참가자가 있는 개방 웹 커뮤니티에 의해 작성되었다. 초기 그룹은 오픈 클라우드 매니페스토의 지원자들로 구성되었지만 조기에 전 세계의 다른 참가자들을 포함할 수 있도록 성장하였다. 본 커뮤니티는 크고 작은 기업, 정부기관, 자문기관, 그리고 벤더들의 대표자들을 포함한다.

본 문서를 작성하면서 3개의 중요한 원칙이 있었는데: 1) 사용자가 같이 작업을 해야 한다, 2) 클라우드의 개방화 노력은 고객이 주도하여야 한다, 그리고 3) 가능한 경우에는 언제나 현존하는 표준이 이용되어야 한다. 본 문서에는 이 3가지 원칙이 적용되었고 추후에 나올 수 있는 작업들에도 적용될 것이다.

여기서 설명된 유즈케이스들은 소비자를 위해서 아래의 일반적인 요구사항들을 도출하였다.

- **공통적인 VM 포맷, 데이터 포맷 그리고 APIs:** 어느 한 클라우드 제공자를 위해 만들어진 가상머신, 데이터, 그리고 어플리케이션은 다른 클라우드 제공자에서도 변경 없이 작동되어야 한다.
- **클라우드 관리:** 클라우드 컴퓨팅은 서비스 관리, 통제, 미터링, 모니터링, 연합된 식별자, SLAs, 벤치마크, 데이터와 어플리케이션 연합, 배치(deployment), 그리고 생명주기 관리와 같은 기능들이 없이는 실현이 불가능하다. 이러한 요구사항들은 섹션 3에 정의되어 있다.
- **보안:** 비록 보안에 대한 요구사항은 어플리케이션과 데이터 유형에 따라 다르지만 보안은 클라우드 컴퓨팅에서 필수적이다.
- **위치인지:** 클라우드 인프라스트럭처를 구성하는 물리적인 머신의 위치를 파악하는 것은 여러 정부 규정을 위해 필수적인 요구사항이다.

소비자들은 여기에 나열된 모든 유즈케이스들을 벤더에 종속 될 수 있는 폐쇄적이고 사적인 솔루션에 의존하지 않고도 구현 가능해야 한다.

여기서 논의된 개발자 요구사항들은 크게 두 유형으로 분류할 수 있다.

- **서비스들을 위한 APIs:** 데이터베이스, 메시징 (P2P와 발표-구독 모두), 컴퓨팅 파워

와 저장기기 등의 모든 API 요구사항들은 모두 직접적으로 클라우드 서비스와 연관된다.

- **APIs 지원:** 캐싱, 로깅, 식별자 관리, 서비스 발견, 세션 관리, 그리고 SLA 등의 API 요구사항들은 클라우드 서비스를 효율적으로 사용하기 위해서는 필수적인 것들이다.

개발자들은 여기에 나열된 모든 유즈케이스들을 벤더에 종속 될 수 있는 폐쇄적이고 사적인 API에 의존하지 않고도 구현 가능해야 한다.

보안은 그들의 데이터와 어플리케이션을 클라우드로 이동하려는 소비자들에게 끊임없이 가장 많은 우려를 발생시킨다. 클라우드 컴퓨팅은 일반적인 IT 보안에서 이미 발생하지 않은 어떤 새로운 보안 이슈를 추가로 만들지는 않는다. 클라우드로 이동할 때의 우려는 이제 보안 정책을 구현하고 적용하는데 제 3자가 개입된다는 것이다. 이런 통제의 상실은 클라우드 제공자의 투명성에 대한 필요성을 강조한다. 여기서의 보안에 대한 논의는 보안 제어, 연합 패턴, 그리고 표준과 같이 클라우드 제공자가 제공해야 하는 것들을 포함한다.

기관들이 클라우드 서비스를 사용함에 따라 소비자와 제공자 모두의 책임은 서비스 수준 협약(SLA)에 명확하게 명시되어야 한다. SLA는 소비자가 어떻게 서비스들을 사용할 것이고, 제공자는 그 서비스들을 어떻게 제공할 것에 대한 정의이다. 어떠한 계약에 서명을 하기 전에, 클라우드 서비스의 소비자는 제공자의 SLA 조항들을 충분히 이해하고 또 그들 소비자 기관의 필요성들을 고려한다는 것은 매우 중대하다.

클라우드 컴퓨팅의 모든 측면에서, 현존하는 기존의 표준이 요구사항들을 만족시킨다면 우리는 그 기존의 표준들이 일관성 있게 개발되고 적용되도록 노력하여야 한다. 기존의 표준들이 요구사항들을 만족시키지 못하는 경우에는 우리는 만족되는 표준들을 정의하고 개발하여야 한다. 커뮤니티에서 작성한 본 문서는 진실 된 개방형 클라우드 컴퓨팅 환경을 구축하기 위한 참조로서의 의미를 갖는다.

변경 내역 요약

버전 2, 10월 30일 2009년:

- 섹션 5 추가, 개발자 요구사항. 다양한 레벨의 APIs, APIs의 유형과 개발자의 역할에 대한 논의가 섹션 2.4에 추가됨. 버전 2에서는 개발자와 그들의 요구사항들이 주된 내용이었음.
- “scalable”이란 용어를 “elastic”이란 용어로 해당되는 곳에서 대체함.
- 클라우드 컴퓨팅에 대한 NIST의 정의 변경을 반영하기 위하여 섹션 2.1에서 분류에 배한 내용을 업데이트함.
- 섹션 4.4에 고객 시나리오를 업데이트함.
- 클라우드 벤더들이 가상 머신에 스토리지를 붙이는 다양한 방법들을 추가하기 위해 일관된 배치와 공통된 가상머신 형식의 요구사항에 대한 논의를 확장함.

버전 3, 2월 2일 2010년:

- 보안을 고려하여 섹션 3.2.1에서 서비스 수준 협약에 대한 논의를 약간 수정함.
- 섹션 6, 보안 시나리오와 섹션 7, 보안 유즈케이스 시나리오를 추가함.
- 결론과 추천 섹션에 보안에 대한 간단한 요약을 추가함.

버전 4, 6월 30일 2010년:

- 섹션 8, 서비스 수준 협약(SLAs)을 추가함.
- 결론과 추천 섹션에 서비스 수준 협약에 대한 간단한 요약을 추가함.
- 섹션 3.2.1의 에러를 수정함 (“basic the identity”를 “the basic identity”로).