목 차

- 클라우드 컴퓨팅이란 ?
- 아마존 웹 서비스(AWS)
- Service Platform Aradon

By bleujin@gmail.com

클라우드 컴퓨팅 시초

- 2007년 가을, Google과 IBM이 워싱턴 대학교 등 미국 주요 대학과 함께 교육과정 개설
- Google과 IBM 데이터센터에서 제공하는 대규모 병렬 컴퓨팅 인프라에 접속해 다양한 실험 및 서비스 개발
- IT 관련 매체들은 이 산학 협동 프로젝트를 클라우드 컴퓨팅이라고 부르기 시작

클라우드 컴퓨팅 정의

- 수많은 PC 또는 서버들이 하나의 커다란 구름모양의 집합을 이루는 것
- 인터넷기술을 활용하여 다수의 고객들에게 높은 수준의 확장성을 가진 IT 자원들을 서비스로 제공하는 컴퓨팅 환경
- 즉 클라우드 컴퓨팅 환경에서 사용자들은 인터넷이 연결된 단말을 통해 대용량의 컴퓨터 집합에 접속하여 어플리케이션, 스토리지, OS, 보안 등 필요한 IT 자원을 원하는 시점에 필요로 하는 만큼 골라서 사용하게 되며 사용량에 기반하여 대가를 지불

다양한 관점의 정의

"진정한 IT 혁명은 지금부터…
클라우드 컴퓨팅 시대 온다“
Nicholas Carr

- 가트너 : 인터넷 기술을 활용하여 다수의 고객들에게 높은 수준의 확장성을 가진 자원들을 서비스로 제공하는 컴퓨팅의 한 형태
- 포레스터리서치 : 표준화된 IT 기반 기능들이 IP를 통해 제공되며, 언제나 접근이 허용되고 수요의 변화에 따라 가변적이며, 사용량이나 광고에 기반한 과금 모형을 제공하며, 웹 혹은 프로그램적인 인터페이스를 제공하는 컴퓨팅
- 위키피디아 : 자원, 소프트웨어, 정보 등이 공유되어 OnDemand로 제공되는 인터넷 기반의 컴퓨팅으로 전력 그리드와 유사한 개념
- BM : 웹 기반 애플리케이션을 활용하여 대용량 데이터베이스를 인터넷 가상 공간에서 분산 처리하고 이 데이터를 데스크탑 PC, 휴대전화, 노트북 PC, PDA등 다양한 단말기에서 불러오거나 가공할 수 있게 하는 환경
- Cloud Computing usecase group : 클라우드 컴퓨팅은 다양한 클라이언트 디바이스에 필요한 시점에 인터넷을 이용하여 공유 풀에 있는 서버, 스토리지, 애플리케이션, 서비스 등과 같은 IT 리소스에 쉽게 접근할 수 있는 것을 가능하게 하는 모델이다. IT 자원의 배포는 쉽고 빨라야 하며 이를 위한 관리비용은 최소화

패러다임의 변화

Burden Iron Works

Edison Power Plant & Power Grid

전력 산업은 이미 중앙집중 모델로 이동

Corporate Data Center

PC

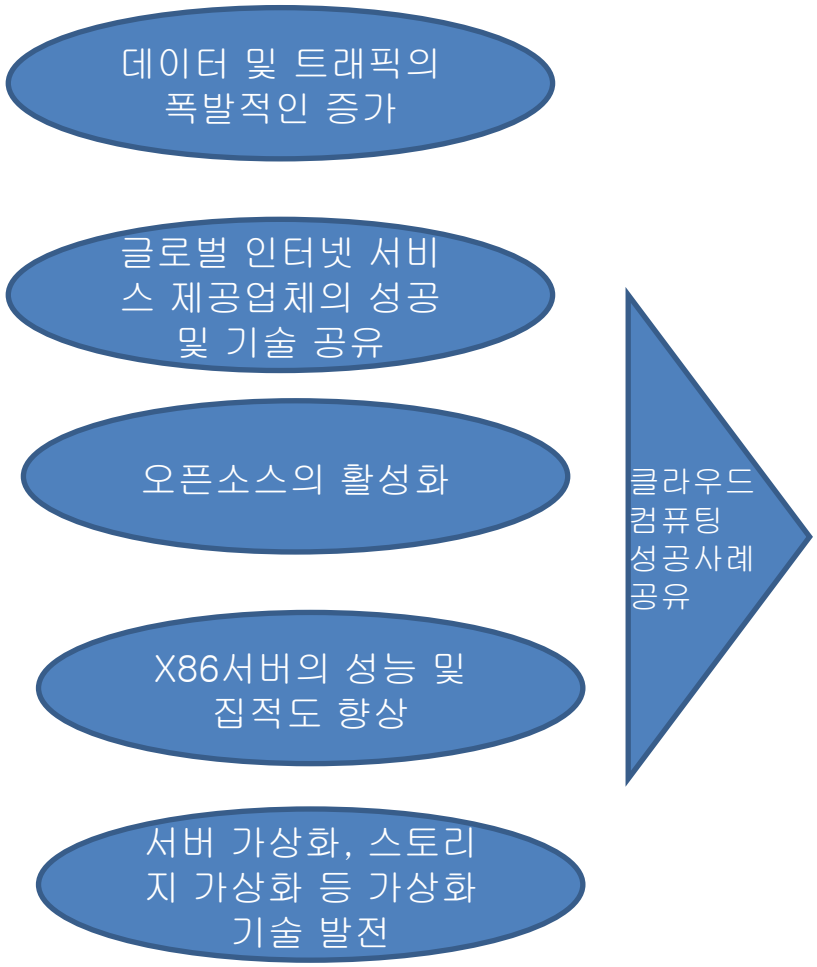
각자 데이타센터 관리

Cloud Computing Center & Internet

대규모 컴퓨팅 자원

- 컴퓨팅 자원의 소유 / 사용방식의 변화
- 데이터의 증가 및 글로벌화된 인터넷 서비스
- 시스템 복잡성 증가 및 소유 비용 증가

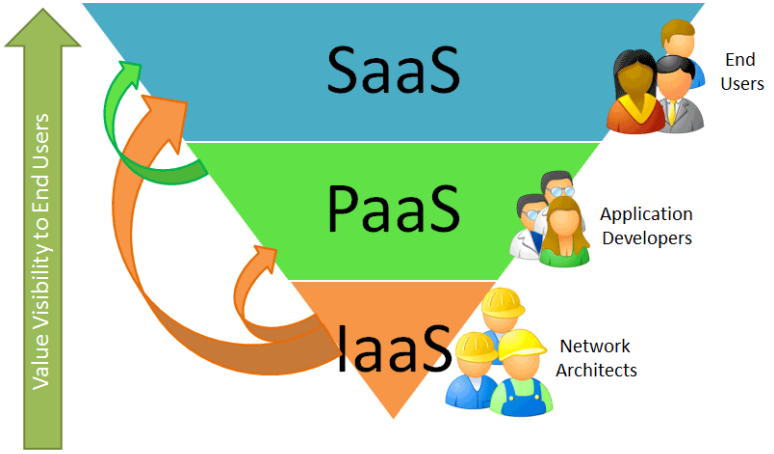
출현 및 확산 배경



- 뉴욕 타임즈의 1851-1980년 신문기사 TIFF 파일을 아마존의 웹서비스를 이용해 분산병렬처리 기술로 36시간 만에 PDF 파일로 변환 (자체 서버 이용 시 14년 소요, 별도의 컴퓨팅 자원 구매 없이 $400 미만으로 처리)
- 힐러리 미국무장관 8년 공식 일정(1만 7481 page)의 문서를 9시간(비용 $150미만) web에 공지해 독자에 제공(아마존 웹서비스 사용료 : 서버 $0.1-0.8/h, 스토리지 1Gb 0.15/1M)
- 일본 우정국은 force.com(개발 플랫폼)을 활용하여 2개월 만에 금융, 생명보험, 우편서비스 3개 시스템을 통합해 4만 고객에 제공(현재 6만 명에게 제공 중) 향후 사용자가 급격히 증가해도 확장성이 용이해 안정적인 서비스 제공
- IBM 사내 technology adoption program 운영으로 시스템 관리자가 15명에서 2명으로 감소 직원 요청에 따른 시스템을 설치 구성하는데 3일에서 1시간으로 단축

클라우드 컴퓨팅 분류

- 다양한 정의만큼 분류도 복잡하지만 일반적으로 두 가지 체계로 분류
  - Delivery Model :
    - Infrastructure as a Service
    - Platform as a Service
    - Software as a Service
  - Deployment Model :
    - Public Cloud
    - Private Cloud
    - Community Cloud
    - Hybrid Cloud



Infrastructure as a Service

- 서버, 스토리지, 방화벽 등 서비스에 필요한 모든 기반 환경을 서비스 형태로 제공하는 것을 인프라 스트럭쳐 클라우드 서비스(Infrastructure as a Service, IaaS)임
- 서비스 형태로 제공되지만 제공된 자원에 대한 제어는 서비스 사용자에게 있음

- 기존의 서버 호스팅 또는 데이터 센터 비즈니스와 유사하지만 다음과 같은 차이점
  - 무한 확장 : 자원의 용량 제한 없이 공급 가능(서버 대수, 스토리지 볼륨 등)
  - 탄력성 : 자원의 추가 및 제거를 쉽게 할 수 있다.
  - 즉시 제공 : 자원은 수~수십 분 내에 사용 가능한 형태로 제공된다.
  - 사용한 만큼만 과금 : 서버 호스팅은 월 과금 또는 특정 기간별 과금인 반면, IaaS는 시간 단위 과금 가능

- 서비스 제공자 수준에 따라 HA, 모니터링 등과 같은 고급 서비스 제공 가능
- 서버, 스토리지, DB 등과 같은 부분은 일부 PaaS 영역과 겹칠 수 있으며 엄격하게 구분해서 나누는 게 어려움

- IaaS 서비스의 사례는 아마존에서 제공하는 AWS(Amazon Web Service)가 있음

Platform as a Service

- 서비스 또는 애플리케이션이 실행되는 환경(Runtime Environment)을 서비스 형태로 제공하는 것을 플랫폼 클라우드 서비스(Platform as a Service, PaaS) 라고 한다.
- PaaS를 이용하는 사용자는 운영체제, 스토리지 등에 대한 제어권을 가질 수 없으며 PaaS 기반에 실행되는 어플리케이션에 대한 제어만을 가질 수 있다.
- 기존의 JSP 호스팅, PHP 호스팅, MySQL 호스팅과 비슷한 개념이다.
- 모든 애플리케이션이 Pass 기반에서 실행 가능하지 않으며 PaaS 서비스에서 제공하는 실행 환경에 맞게 개발된 애플리케이션만 실행 가능하다.
  J2EE + MYSQL, Python + Oracle, 자체 제공 프로그래밍 언어 + 데이터 베이스
- 단순히 개발자들에게 개발 플랫폼을 빌려주느냐, 개발한 어플리케이션을 곧바로 구동하고 서비스할 수 있는 환경까지 제공하는가에 따라 DaaS(Development as a Service)와 PaaS를 구분할 수 있다.
- PaaS(Platform as a Service)가 반드시 IaaS에서 운영될 필요는 없지만 PaaS 제공자가 인프라를 제공하거나 IaaS 서비스와의 연동을 통해 인프라로 배포되는 기능까지 같이 제공해야 한다.
- 서비스 사용자는 인프라에 대한 제어 권한이 없기 때문에 PaaS 서비스 제공업자가 로드밸런싱, 모니터링, HA 구성들을 제공해야 한다.
- PaaS 서비스 사례는 Google의 AppEngine, SalesForce.com이 있다.

Software as a Service

- 소프트웨어를 인터넷 서비스 형태로 제공하는 것을 클라우드 소프트웨어 서비스(Software as a Service, SaaS)라고 한다.

- 기존의 ASP(Application Service Provider)의 확장된 개념으로 다음과 같은 속성을 가지고 있어야 한다.
  - Multi-Tenancy : 하나의 소프트웨어 인스턴스로 여러 조직 또는 회사가 사용한다.
  - Easy-Customization : 하나의 인스턴스지만 조직 별 화면, 메뉴, 로직 등이 쉽게 적용할 수 있다.
  - Elastic Scalability : 급격한 사용자 요청에도 서비스는 안정적으로 운영된다.
  - Security : 하나의 인스턴스로 여러 조직이 사용하기 때문에 사용자가 보안 체계가 있어야 한다.

- SaaS 서비스는 반드시 IaaS, PaaS 기반에서 운영될 필요는 없다.
- 대표적인 예로는 세일즈포스(Salesforce.com)과 Google 앱스(Google Apps)가 있다.

- 국내에서는 최근 NHN의 N드라이브, KT의 uCloud 등과 같은 개인 스토리지 서비스가 있다.

Public Cloud

- 퍼블릭 클라우드는 특정 조직 내부가 아닌 외부의 서비스 제공자가 제공하는 클라우드 서비스를 사용하는 형태
- 일반적으로 클라우드 컴퓨팅이라고 하면 퍼블릭 클라우드 컴퓨팅을 의미
- 퍼블릭 클라우드라고 해서 클라우드 내에 있는 모든 정보가 공개되는 것이 아니라 서비스 제공자에게 의해 내부적으로 제한 관리 및 보안 정책이 적용

- 다음과 같은 클라우드 컴퓨팅의 속성을 충분히 보장 받을 수 있으며 이를 통해 비용 및 서비스의 안정성을 보장
  - Resource Pooling
  - Elastic Scalability

- 서비스 제공자를 신뢰할 수 있어야만 사용 가능한 단점이 있으며 서비스 제공자의 서비스 품질에 따라서 다음과 같은 항목이 고려 되어야 함
  - 서비스 제공자의 보안 서비스 제공 수준
  - SLA(Service Level Agreement)
  - 레거시 시스템과의 연계

- 주로 Google, 아마존 등과 같은 글로벌 인터넷 업체가 퍼블릭 클라우드 서비스를 제공하고 있음.

Private Cloud

- 프라이빗 클라우드는 특정 조직 내부에 클라우드 컴퓨팅의 특성을 가지고 있는 서비스를 구축하여 조직 내부 구성원만 사용 가능하도록 한 형태를 말한다.

- 조직 내부의 폐쇄된 네트워크 망에서 운영되기 때문에 다음과 같은 항목에 대해서는 퍼블릭 클라우드에 비해 장점이 있다.
  - 데이터 보안
  - 레거시 시스템과의 연계

- 조직 내부의 별도의 클라우드 서비스를 구축해야 하는 부담과 아래와 같은 단점이 있다.
  - 클라우드 서비스를 위한 시스템 구축 및 운영 노하우 필요
  - 시스템 구축 비용
  - 일정 규모가 되지 않으면 비용 절감이 아니라 비용이 더 발생할 수 있음
  - 확장성에도 한계가 있음

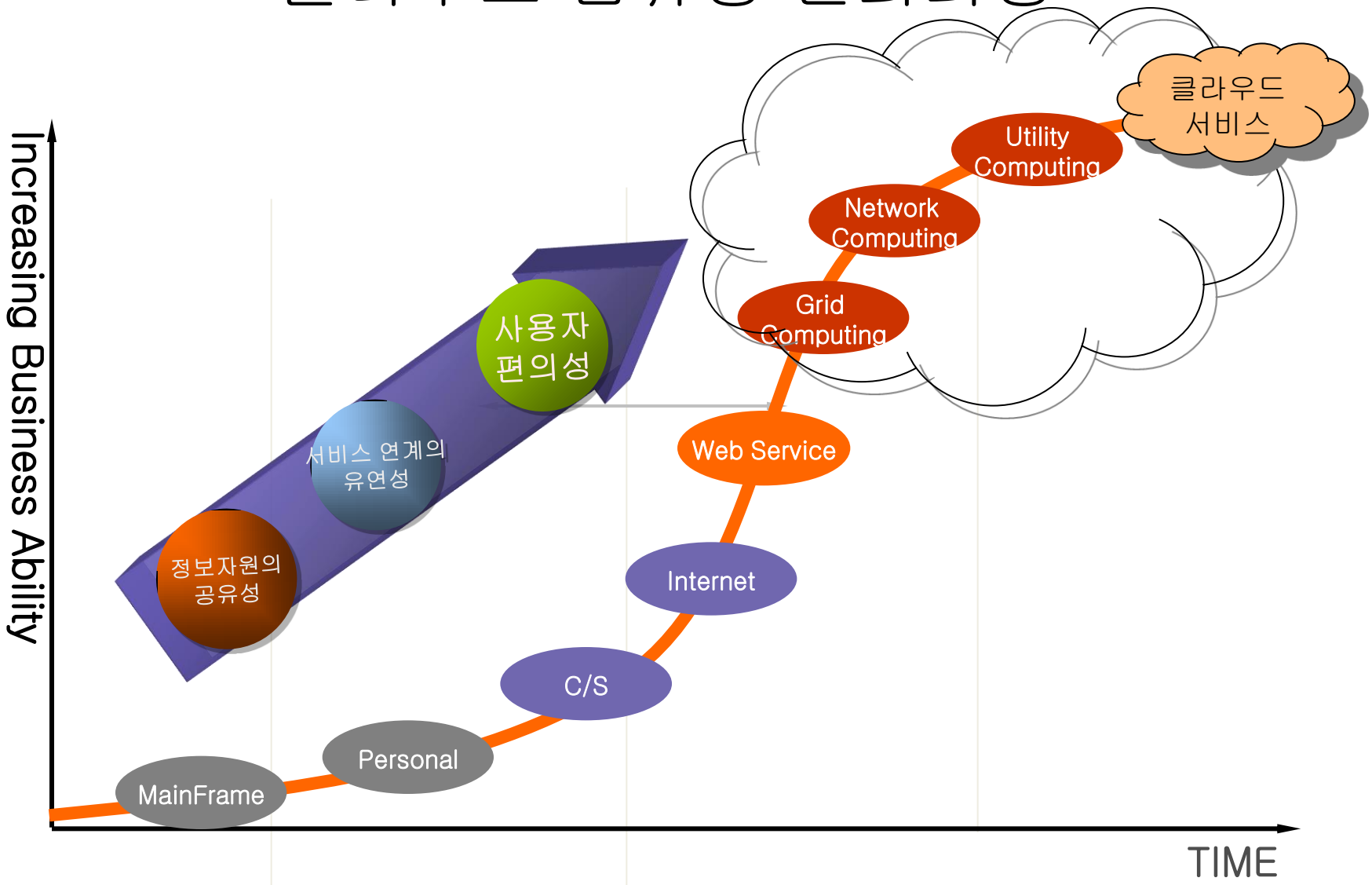
- 프라이빗 클라우드는 주로 IBM, HP, Microsoft 등과 같은 기업용 솔루션 제공자를 중심으로 발전하고 있다.

Community & Hybrid Cloud

- Community Cloud
- 커뮤니티 클라우드는 특정 목적을 가진 사용자나 조직에 의해 운영되는 형태를 의미한다.
- 주로 연구 목적이나 공익을 목적으로 운영되는 경우가 많다.
- Open Cirrus(클라우드 컴퓨팅 테스트 베드)

- Hybrid Cloud
- 하이브리드 클라우드는 퍼블릭 클라우드와 프라이빗 클라우드를 병행해서 사용하는 방식을 의미한다.
- 기업 업무 중 미션 크리티컬한 업무를 지원하는 시스템은 프라이빗 클라우드를, 일반적인 업무를 지원하는 시스템은 퍼블릭 클라우드 서비스를 이용한다.
- 이 경우 퍼블릭 클라우드와 프라이빗 클라우드 간의 상호 연동이 쉬워야 한다.

클라우드 컴퓨팅 진화과정



전통 IT VS 클라우드 컴퓨팅

- 전통적인 IT는 개개 시스템의 벽을 넘기 힘들었다. 고정된 구조를 가지고 있었으며 구조를 바꾸기는 쉽지 않았다.

- 하지만 클라우드 컴퓨팅에서는 이 구조 자체를 유연하게 가져갈 수 있으며, 자원의 공유와 자원 할당의 동적인 변화를 통해 높은 효율을 얻을 수 있다.

- 이는 가상화와 함께 현재의 그리드 컴퓨팅이나 유틸리티 컴퓨팅이라고 불리는 분산 컴퓨팅 기술을 통해 얻을 수 있는 효과이다.

- 오픈 소스는 폐쇄적 환경에서 개방적 환경으로의 개발 환경의 이동을 반영한다. 웹 2.0은 이 오픈 소스에 있어 개발자들 간의 커뮤니티 수단을 제공하게 된다. 이들도 클라우드 컴퓨팅의 발전을 가능하게 해주는 원동력 중 하나이다.

메인프레임 vs 클라우드 컴퓨팅

- 메인 프레임이 강력한 물리적인 컴퓨팅 자원을 통해 프로세스를 처리하는데 중점을 둔다면 클라우드 컴퓨팅은 할당(Allocation)과 모니터링 등으로 분산 및 이기종 환경을 지원하는 것이 핵심이다.

- 클라우드 컴퓨팅은 가상화(Virtualization)의 범주에서 이해하는 시각이 많다.

그리드 vs 클라우드 컴퓨팅

- 그리드 컴퓨팅은 병렬처리가 가능한 하나의 거대 규모 태스크를 훨씬 작은 태스크로 분할해 다수의 개별 서버를 통해 병렬 처리하는 기술이며 분산된 자원에 대한 제어가 느슨한 반면에

- 클라우드 컴퓨팅은 어플리케이션의 병렬처리 특성과 무관하게 가상화된 컴퓨팅 자원을 제공하는 기술이며 분산된 자원에 대한 제어가 중앙 집중 관리 된다.

ASP vs SaaS

- Application Service Provider와 Software as a Service는 모두 소프트웨어를 설치해서 사용하지 않고 서비스 제공자가 제공하는 서비스를 사용한다는 측면에서는 유사하다.

- SaaS는 ASP의 진화된 형태로 볼 수 있으며 멀티테넌시 개념을 이용하여 기존 ASP 서비스에 비해 다양한 커스터마이징이 가능하며 서비스 운영에 필요한 리소스를 클라우드 컴퓨팅 기술을 이용하여 최대한 효율적으로 운영함으로써 더 많은 사용자에게, 더 저렴한 비용으로 서비스 할 수 있다.

서버 호스팅 vs SaaS

- 컴퓨팅 자원을 소유하지 않고 임대 방식으로 이용하는 방식은 비슷하지만 자원을 다음과 같은 측면에서 차이점이 있다.
  - 과금 정책 : 서버 호스팅은 월 단위로 과금 되지만 클라우드는 시간단위
  - 자원 제공 시간 : 서버 호스팅은 수일 – 수주 이상 소요되지만 클라우드는 신청 즉시 사용 가능
  - 자원 한계 : 서버 호스팅은 서비스 제공 사업자가 확보한 수준에서 제공 가능하지만 클라우드는 가상화를 통해 개념적으로 무한으로 자원 서비스 가능
  - 자원의 관리 : 서버 호스팅은 자원을 사용하는 사용자가 자원을 관리해야 하지만 클라우드는 자동화된 관리와 도구를 제공

도입 시 기대 효과

- 사용도가 낮은 자원에 대한 자산 구매 불필요, 인프라를 소요할 필요가 없어 비용이 절감
- 사용자는 특정 PC, 위치 등에 상관없이 시스템 또는 데이터에 접근 가능
- 다수의 사용자가 자원을 나누어 사용하므로 인프라의 집중화로 인한 비용절감 및 최대 용량 증가, 시스템 효율성 향상
- 부족한 시스템 운영 전문 인력을 집중화 할 수 있어 안정적인 서비스 운영
- 갑작스런 사용량 증가에 저렴하고 신속하게 대응 가능
- 필요한 자원의 선택적 구매와 사용량 기반 대가 지불의 합리적인 가격 모델
- Startup 기업에 적합

해결 과제

- 서비스 안정성
  - 모든 정보가 중앙에 집중되어 있는 클라우드 컴퓨팅에서 서비스 안정성은 매우 중요한 요소이다.
  - 아마존의 S3 서비스가 2시간 동안 중단되어, 기업과 사용자가 피해를 입음

- 자료 / 정보 신뢰성
  - 기업과 개인 사용자들은 자신들의 핵심 데이터가 외부에 저장되는 것에 우려를 갖고 있다.
  - 클라우드의 특성상 사용자들은 자신의 데이터가 어디에 저장되어 있는지 알 수 없다

- 표준화
  - 대부분의 클라우드 사업자들은 자체적인 플랫폼을 이용해 서비스를 제공하고 있다.
  - 이러한 클라우드 플랫폼을 이용해서 서비스를 개발하고 제공하는 사용자들은 사업자들이 제공하는 개발환경 내에서만 어플리케이션 개발이 가능하고
  - 서로 다른 사업자들 간의 서비스 전환이 어렵기 때문에 하나의 클라우드 사업자에 종속될 수 있는 문제가 존재한다.

클라우드 컴퓨팅 서비스 사례

분류	구분	사례
SaaS	응용 소프트웨어 서비스	GoogleApps, Salesforce.com Apps, Apple MobileMe, Nokia OVI, IBM Bluehouse
	웹 기반 서비스	HP Snapfish, MS Office Live, HP Magcloud, Google Docs, Github
	응용 소프트웨어 컴포넌트 서비스	Amazon FPS API, Google MAP API, Google Calendar API, Yahoo MAP API
PaaS	엔터프라이즈 플랫폼 서비스	GigiSpace, Oracle SaaS Platform
	호스티드 플랫폼 서비스	Google AppEngine, Saflesforce Force.com MS Azure, Sun Caroline, Cloudera
IaaS	데이터베이스 클라우드 서비스	Amazon DynamoDB, Google Base
	미들웨어 클라우드 서비스	Amazon Simple Queue Service
	스토리지 클라우드 서비스	Amazon S3(Simple Storage Service), Apple, Naver ndriver
	컴퓨팅 클라우드 서비스	Amazon EC2(Elastic Compute Cloud)

Amazon Web Service

AWS EC2

- Elastic Cloud Computing 서비스는 AWS 서비스는 핵심 서비스로 가상 머신을 제공해주는 서비스이다.

- 제공 인스턴스
  Small : 1.7GB Memory, 1CPU, 160GB Storage
  Large : 7.5GB Memory, 4CPU, 850GB Storage
  Extra : 15GB Memory, 8CPU, 1690GB Storage
  Micro Instance, High-Memory Instance, High-CPU Instances, Cluster Instance, GPU Instance

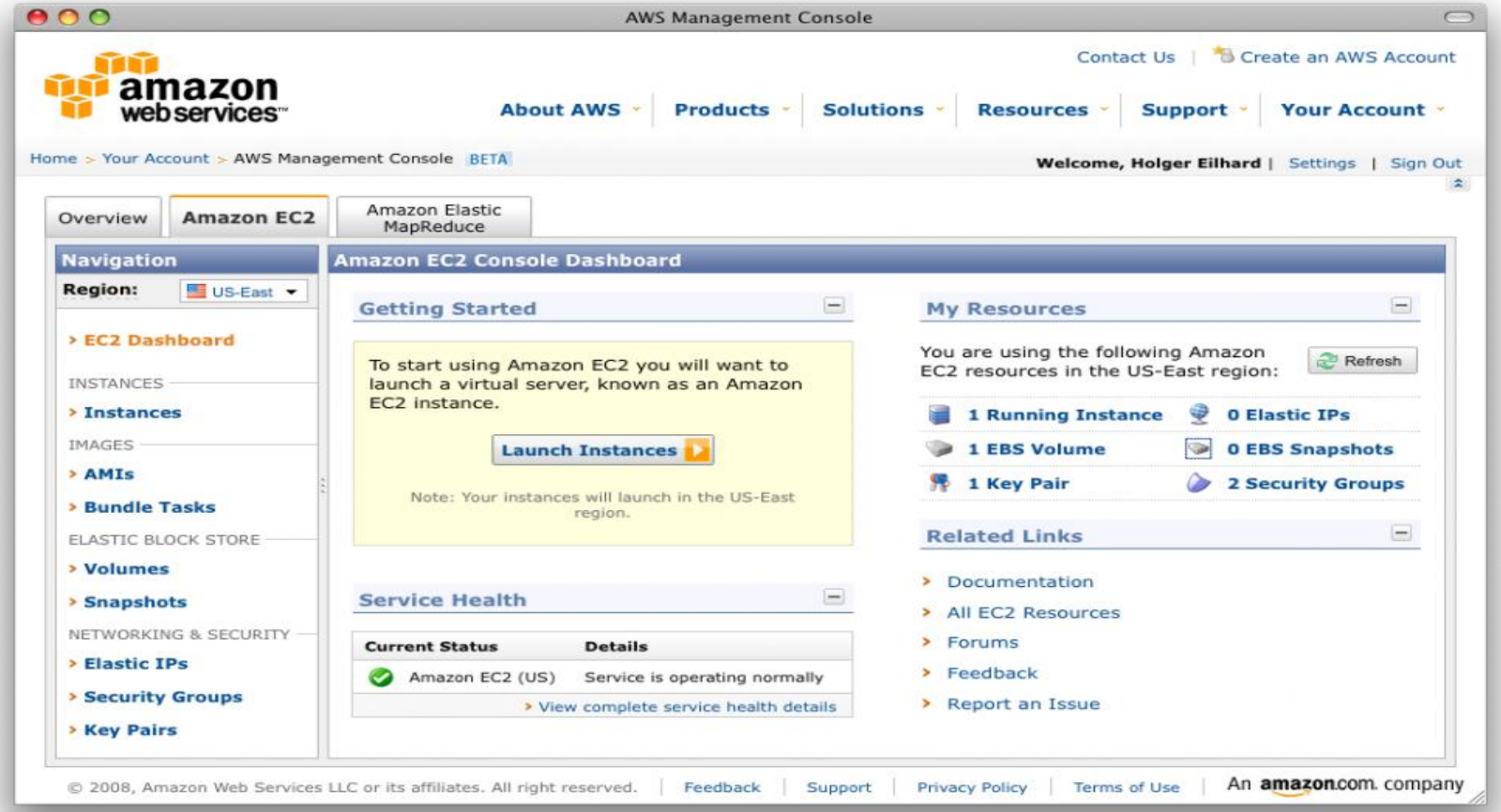
- Linux, Windows, Solaris

- AMI(Amazon Machine Image)이라고 하는 가상머신 이미지를 관리 할 수 있는 기능 제공
  - 기본 제공하는 이미지를 사용하거나 다른 사용자가 만든 공용 이미지 사용 가능
  - 이미지에 대한 저장 공간은 S3 사용

- 가격 정책
  - On-Demand -> 시간당 과금, Reserved -> 연간 단위 과금
  - Small : $61/월, Large:$245/월
  - 네트워크 비용 : In 0.1/GB, Out : 0.15/GB(월 1G까지 무료, 단위 별 과금)

AWS EC2

- https://console.aws.amazon.com/ec2/home



AWS EC2(실습)

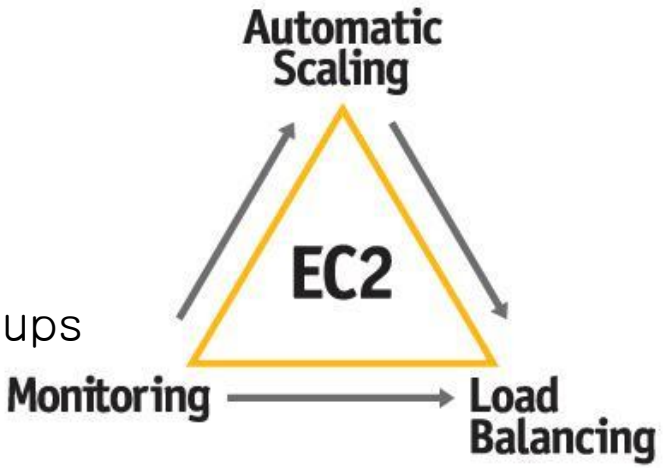
- http://www.youtube.com/watch?v=JPFoDnjR8e8
  Getting Started With Amazon EC2
  http://www.youtube.com/watch?v=tJ122hr01ZU :
  Create a Windows 2008 Server on Amazon EC2

- Register User

- Make EC2

- Make Images

- Elastic Ips, Load Balancers, Security Groups



AWS S3

- 고가용성의 대규모의 스토리지를 제공해주는 서비스로 개인용 스토리지 서비스보다는 주로 어플리케이션의 첨부파일, 웹의 정적 이미지 파일, 데이터 파일 등을 저장하는 용도로 사용한다.
- EC2 서비스에서 사용하는 이미지, 사용자 데이터 등의 백업용으로도 많이 사용된다.
- 1byte-5GB 크기의 파일을 개수의 제한 없이 저장 가능하다.
- 저장된 파일에 대한 연산은 일반적인 파일 연산을 사용하는 것이 아니라 S3에서 제공하는 API를 이용하며 사용 가능한 연산은 write, read, list 등이 있다.
- 고 가용성을 지원하기 위해 파일은 복제되어 저장된다.
- 가격 정책
  스토리지 공간, 데이터 전송량, 명령 수행 횟수가 과금 항목이다.
  스토리지 : 월 $0.14/GB, 전송량 : 0.1/GB,
  명령 수행횟수 : 0.1/10000 request
- 사용 용도
  컨텐츠 저장소, 분석용 데이터 저장소, 백업/아카이빙/장애복구 용도

AWS S3(실습)

- http://www.youtube.com/watch?v=1qrjFb0ZTm8
  Getting Started with Amazon S3
- http://www.youtube.com/watch?v=ZPBvB16MR5M
  Introduction to Amazon S3

- Create Bucket

- Upload File

- Set Property(Permissions)

- View File / Move File / Delete File
  /amazonClient
  /test/net.ion.amazon.s3.vfs.provider.s3.TestS3Provider

AWS DynamoDB

- 이전의 SimpleDB를 대체함
  http://youtu.be/oz-7wJJ9HZ0

- 대규모의 데이터를 위해 설계한 빠르고 확장성 좋은(scalable) NoSQL DBService

- 원하는 요청처리속도(reqs/sec)를 설정하고 조정해서 사용
  SimpleDB의 한계
  - Domain scaling Limitation
  - Predictability of Performance
  - Pricing complexity

- SSD를 써서 빠른 속도를 제공

AWS Eclipse PlugIn(실습)

- PlugIn : http://aws.amazon.com/eclipse/
  doc : http://aws.amazon.com/ko/eclipse/

- 플러그인 설치

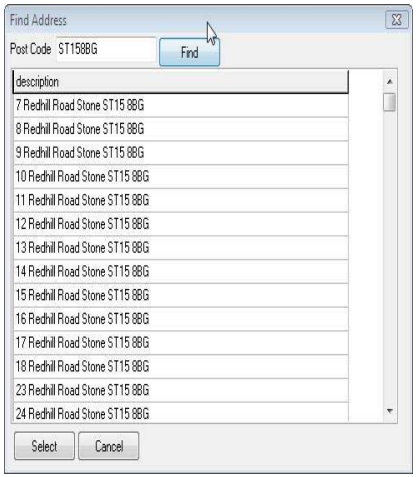
- API Sample
  https://aws.amazon.com/code/
  https://github.com/bleujin/amazonClient

- AWS Explorer
  /amazonClient
  /test/net.ion.amazon.s3.vfs.provider.s3.TestProvider

Service Platform Aradon

Aradon을 만들기 전에

- 1998년 : I모 쇼핑사이트 제작 때
  배달 주소 등록 때 우편번호 검색이 필요함
  해야 하는 작업
  - 우편번호 테이블 만듦
  - 데이터 Batch로 입력
  - SQL과 프로그램 작성
  - View 화면 제작

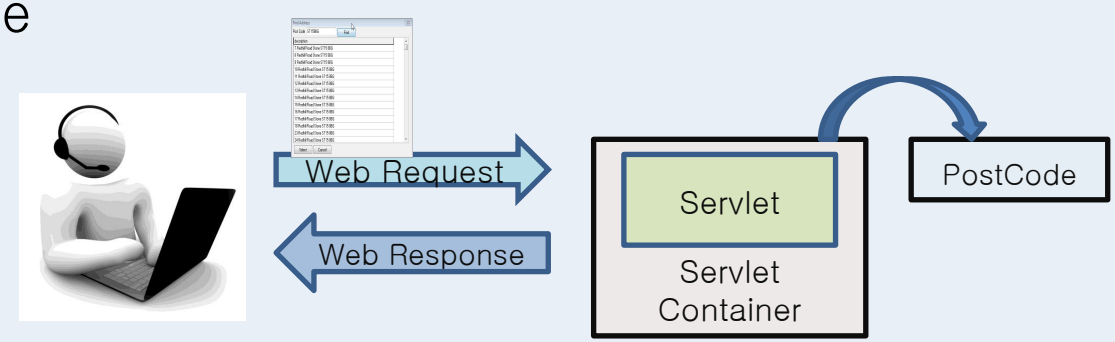


- 그런데 이게 끝이 아님…
  - 우편번호 관리(우편번호가 수정됨), 잘못 등록된 우편번호 발견
  - 새로운 우편번호가 지정됨, 과거 우편번호와 Data 충돌
  - 이걸 매번 사이트 만들 때마다 고려 ?

- 2012년 현재 : 우편번호 컴포넌트 ?
- 컴포넌트나 프레임워크가 아닌 서비스가 필요함

Aradon을 만든 후에

- Before



- After

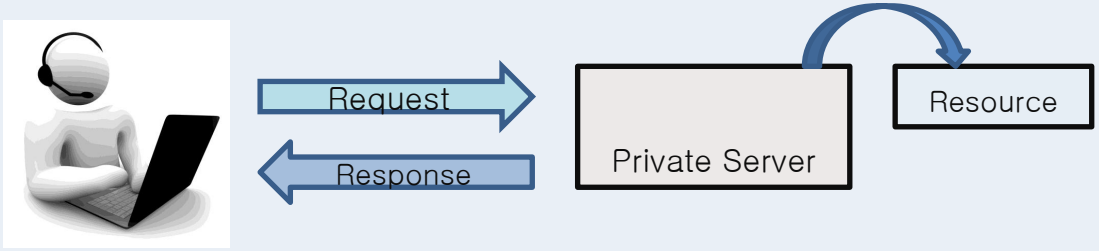
Aradon을 만들기 전에

- 2002년 : Java Applet 게임 만들 때
  - 게임마다 Hall Of Fame이나 기타 정보 등록이 필요함
  - 게임마다 서버를 구현?
  - 게임을 팔 때마다 서버를 구축?
  - 게임은 500원, 얼마나 팔릴지 모르는 상태에서 서버구입은 무리

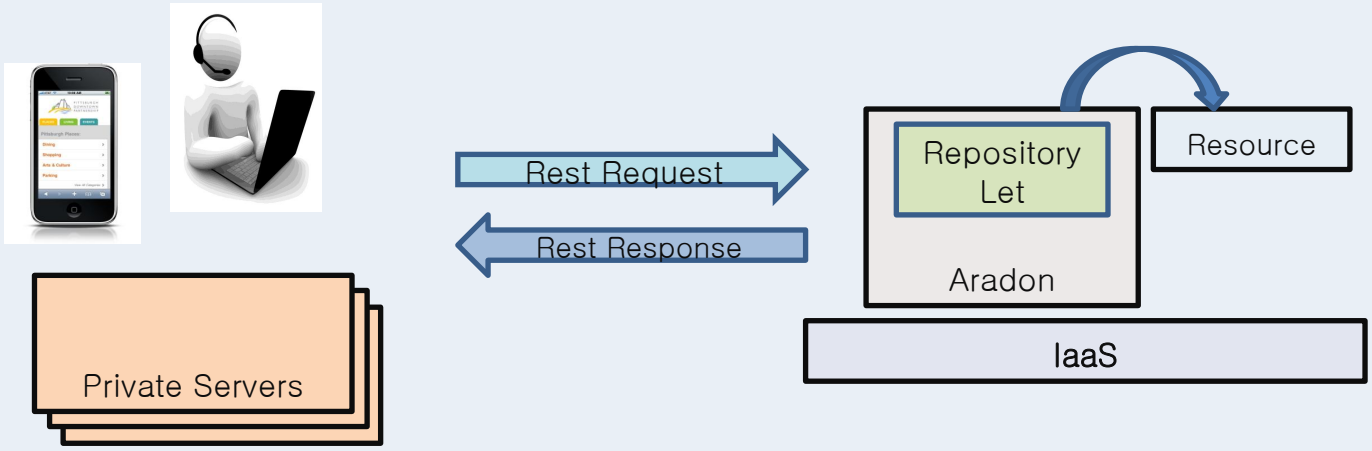
- 2012년 : Mobile App 게임 만들 때
  - 위의 문제 고스란히 상속
  - 별도의 서버 관리 비용이 부담
  - 안정적인 서비스 & 모니터링이 필요함
  - 유연한 확장이 가능해야 함.

Aradon을 만든 후에

- Before



- After



Aradon을 만들기 전에

- 2005년 : KTF WIPI 프로그램 관리툴 개발 때
  사용자 정보는 타 시스템에서 관리
  사용자 정보를 제공받을 수 있는 서비스가 없음,
  DBA는 보안상의 이유로 DB 계정 공유 불가 -> Scheduling Copy
  DW?(Data Warehousing)

- DB는 서비스가 아니라 Resource

- Resource를 다루기 위한 Service 필요
  별도의 Security 관리 정책(인증과 허가)
  서비스 제어 및 통제 필요
  그러나 DB처럼 사용할 수 있어야 한다.
  Multi-Tenancy

프로그래밍의 성배

- DRY(Don't Repeat, Yourself)

  Library(코드 중복)
  - 절차 지향 프로그램 언어(포트란 등의 고급언어)

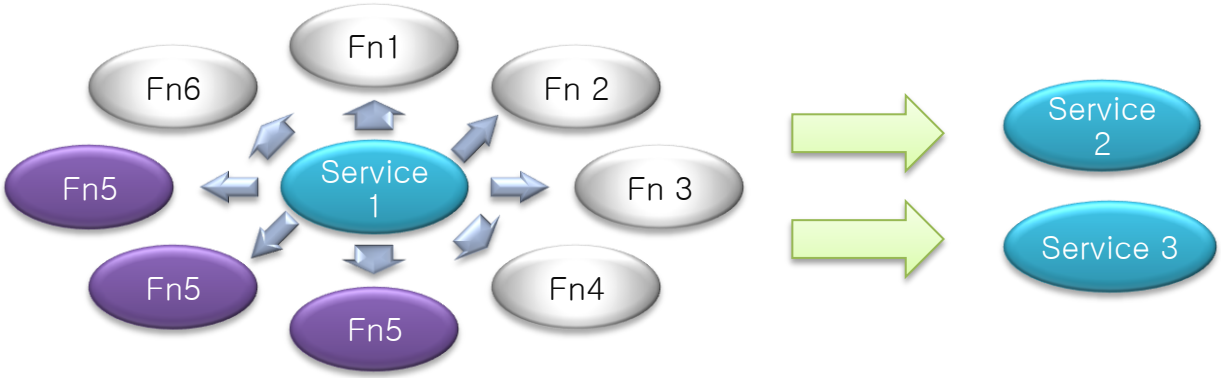
  Component
  - 객체 지향 프로그램 언어
  - High cohesion & Low coupling

  Framework(구조적 중복 해결)
  - Request와 Response 사이에 일반적인 구조를 찾아낸다

  WebService(서비스 중복 해결)

서비스를 만들자

- Multi-Tenancy가 되는 서비스
- 다양한 Device에서 접속 가능한 서비스
- 다른 서비스와 연계가 가능한 서비스(다양한 Protocol)
- 안정적인 쓰루풋과 확장이 가능한 서비스
- SNS -> Service aNd Service

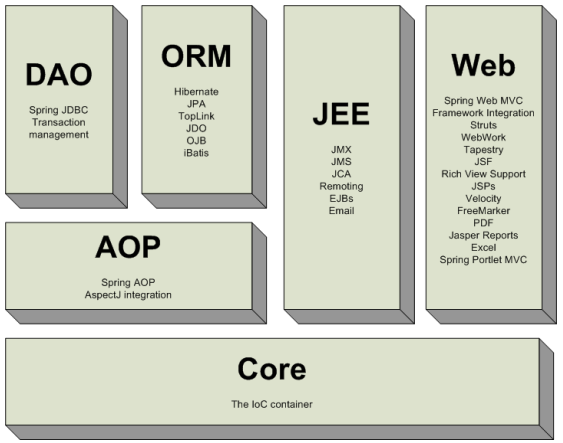


왜 Based 스프링이 아닌가?

- 불필요한 복잡함
  - 잡으려고 하는 몬스터(서비스)보다 잡을때 사용할 도구 사용이 더 복잡함
  - 자바의 복잡성은 문화적인 것이지 강제된 것은 아니다.

- 서비스의 비기능적 요구사항의 증가
  - 스프링은 비기능적 요구사항에 도움이 못됨

- 서비스를 사용하는 것은 사람이 아니라 다른 서비스임
  User View는 거의 없거나 별로 고려할 사항이 아님
  Multi-Tenancy는 중요함
  다양한 Device 고려



왜 Framework이 아닌가?

- Framework의 단점
  배우는 데 시간이 걸린다. (언제나 도움말은 충분치 못하다.)
  문제영역에 맞춘 코드가 아니라 프레임워크에 맞춰진 코드가 탄생한다.
  여러 프레임워크의 혼용 시 복잡도가 증가한다.(단일지점 제어의 혼란)

- Framework는 정글에 난 길과 같다.

- Framework와 Platform이 분리되면 Testability가 떨어지며 Deploy가 복잡해진다.
  개발, 배포, 그리고 운영을
  하나의 스펙트럼 안에서 처리하고 싶었다.
  (Aradon은 Framework인 동시에 Platform이다.)

- Framework vs Platform

왜 Based Servlet이 아닌가?

- 동적인 웹 페이지를 만들기 위해 나온 Servlet, JSP의 한계
  - Servlet Arch Model : State Model, Disconnected Web
  - 서블릿이 하기 어려운 문제들 :
    Connected Web, Streaming
    Distribute & Split Service(Not Cluster)
    데이터 직렬화(AVRO, Apache Thrift)

- Mobile Device의 확장에 따른 시장 환경
  - Required high scalability

- 배포와 테스트 곤란
  - Servlet Model은 Test 친화적이지 않다.
  - View와 너무 밀접하게 연관되어 있다.
  - Not Fun and Not Productive
  - WANT to reload service

왜 Node.js가 아닌가?

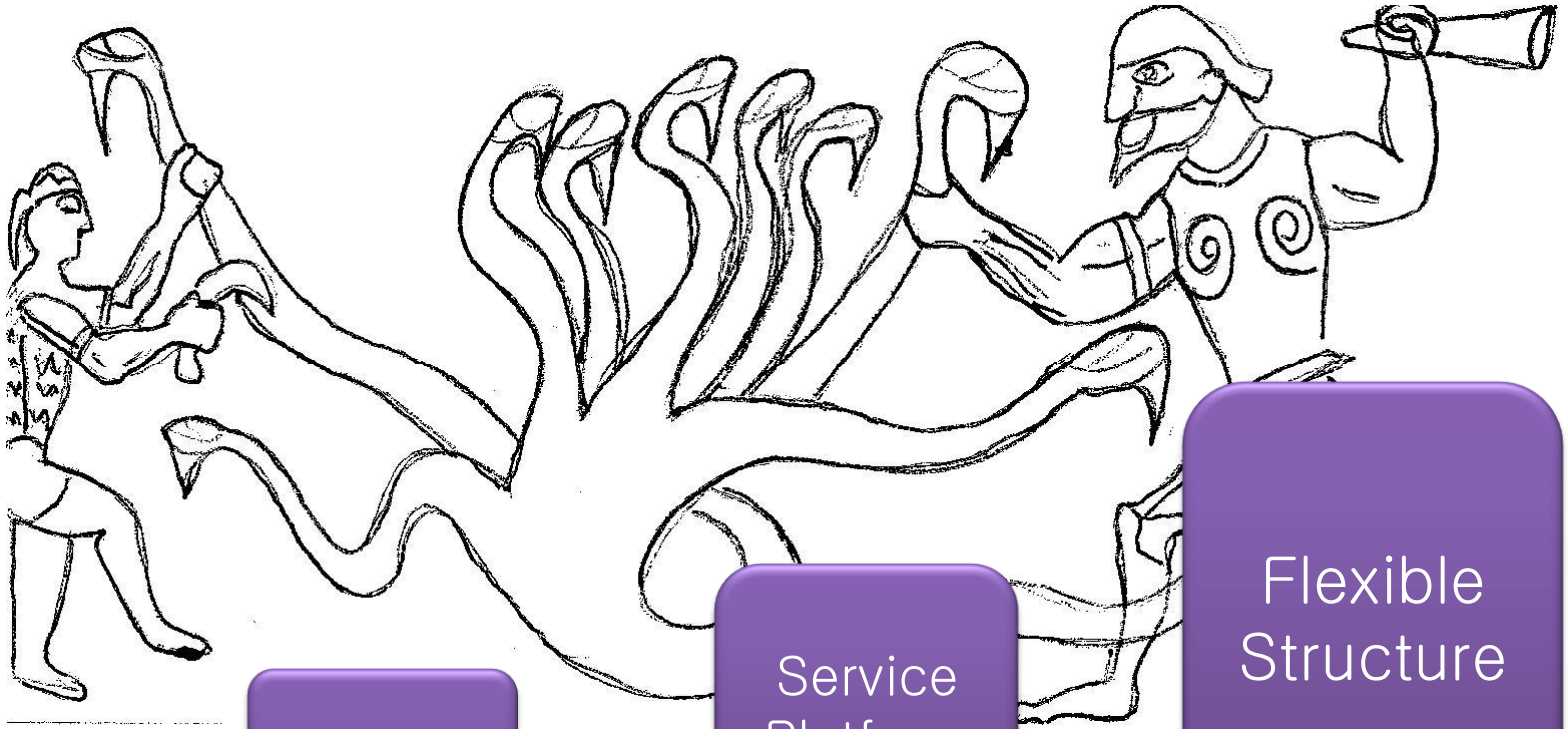
- I Need Control
  - Filter
  - Context
  - Multi Protocol on Port
  - Dynamic Handling

- Testability
  - Javascript 가독성
  - 하위 구조까지 접근할 수 없다.

- Javascript의 domain 한계
  - 멀티 쓰레딩
  - Closure 구조의 복잡성

- Aradon의 경우 EngineType에 따라 동기/비동기 선택

Aradon



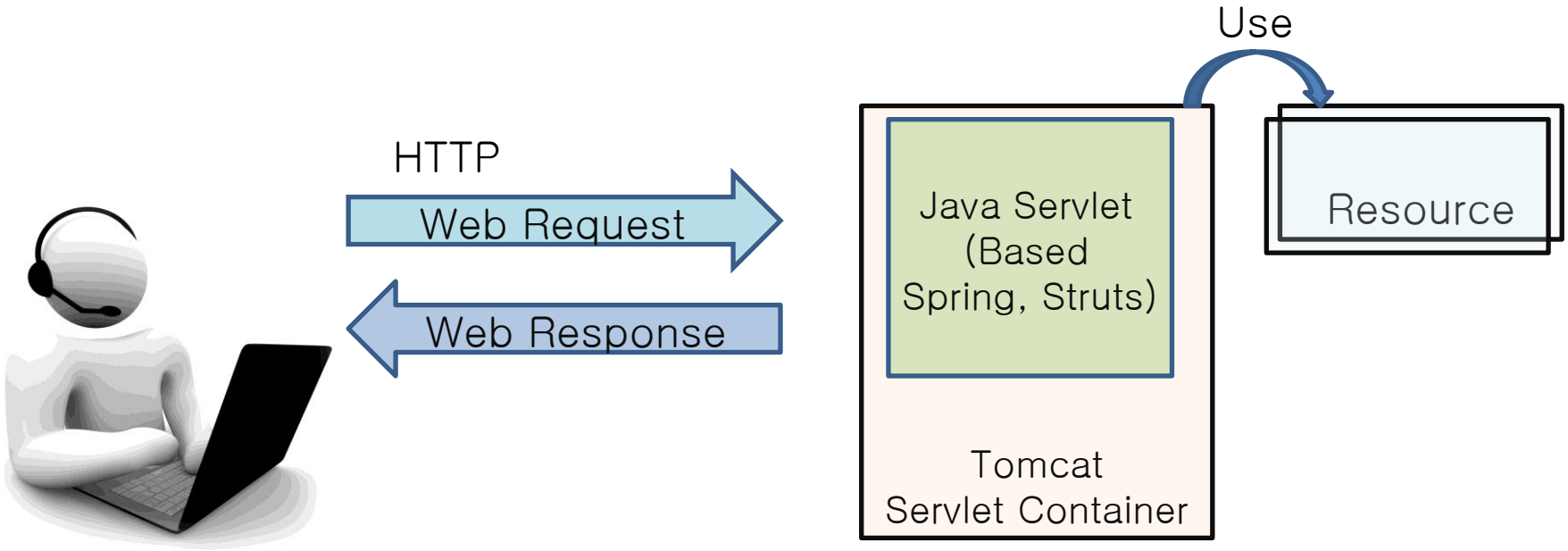
Easy Call

Service Platform

Flexible Structure

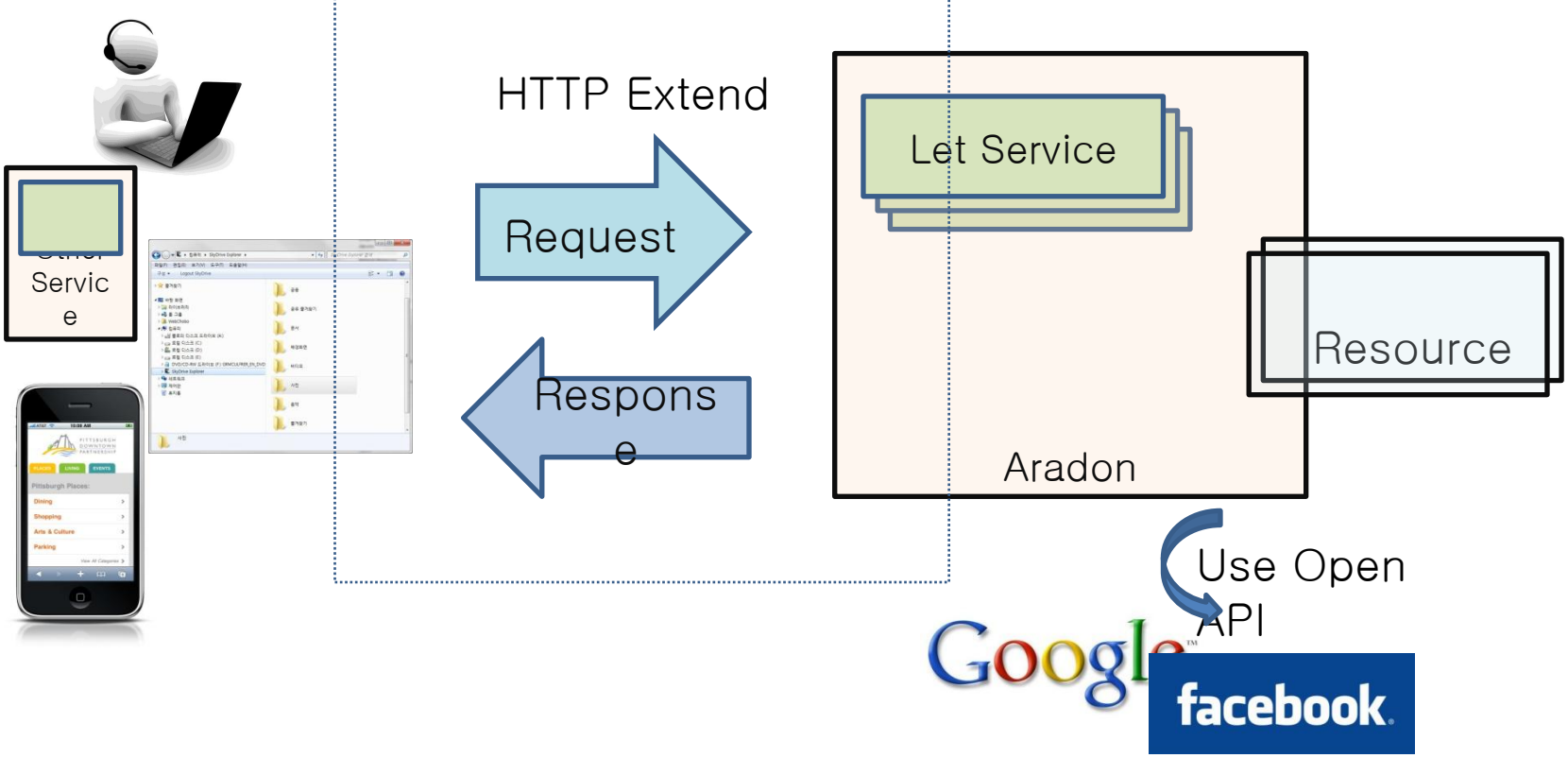
What Aradon ?

- Traditional WebService



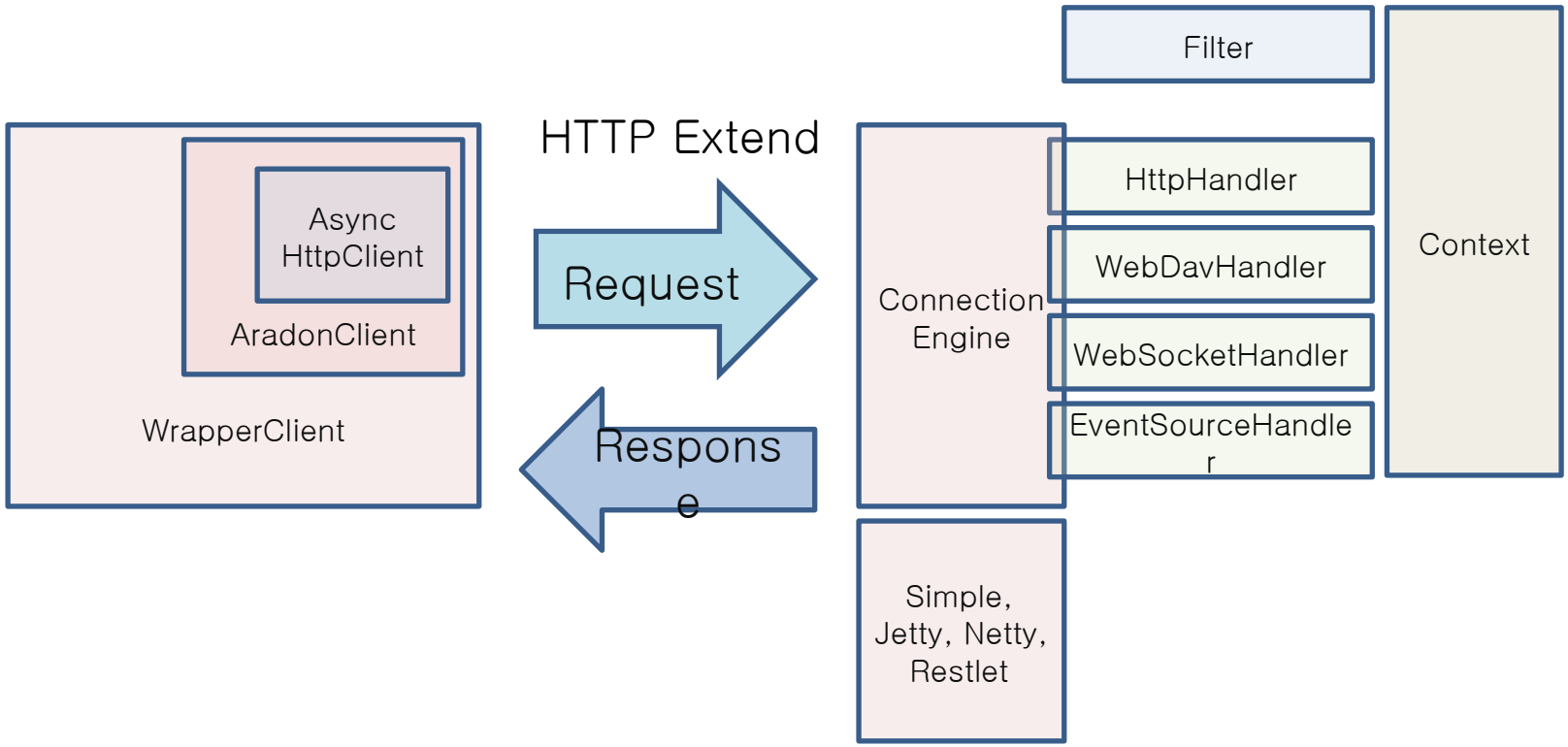
What Aradon ?

- Aradon



What Aradon ?

- Aradon



Aradon Download

- Egit 설치
  http://www.vogella.com/articles/EGit/article.html
  http://download.eclipse.org/egit/updates

- Download From gitHub
  AradonServer : https://github.com/bleujin/aradon
  AradonClient : https://github.com/bleujin/aradonClient
  AradonExtend : https://github.com/bleujin/aradonExtend

  doc : http://bleujin.springnote.com/pages/107611

Hello World 만들기(실습)

- Let 만들기
  - parameter를 받아서 Hello {name} 출력
- Let Test
- Let Service(on Browser)
- Aradon
  /test/net.ion.radon.TestFirst

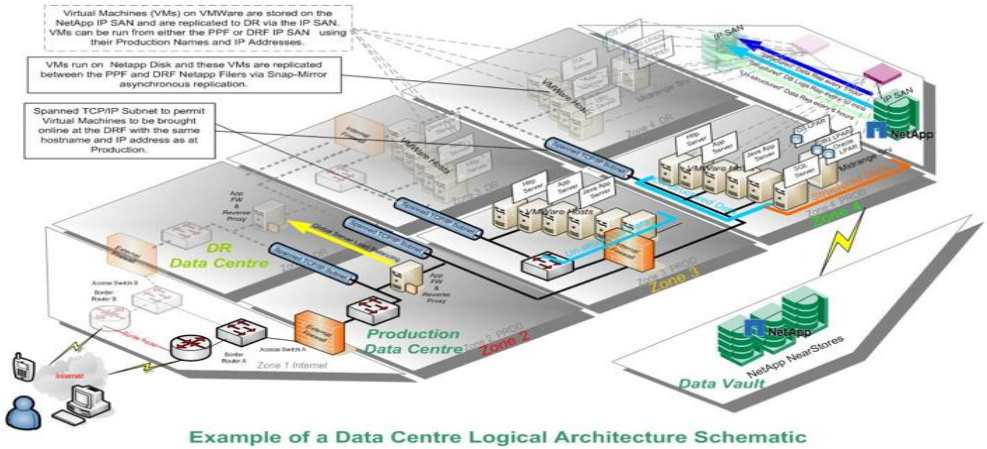
Chat 만들기(실습)

- WebSocket Handler 만들기
- WebSocket Client 로 접속하기
- StaticFile Handler 만들기
- Browser로 접속하기

  /WebSocketPlug
  /test/net.ion.chat.TestFirst

Repository 만들기(실습)

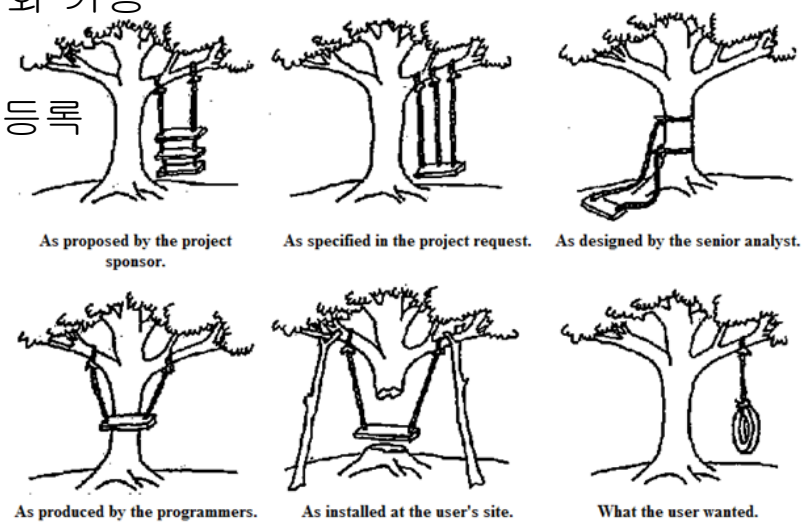
- RDB Service 등록
- Wrapper Client(MongoNode)
- /MongoSearch
  /test/net.ion.radon.repository.remote.rest.TestAmazonLet



Example of a Data Centre Logical Architecture Schematic

할 수 있는 것

- 서비스를 띄우지 않고 Test 가능
- 기존의 WAS에 Embed되어 Servlet REST Mode로 동작
- HTTP, WebDAV, WebSocket, EventSource 등의 다양한 Protocol 지원
- Streaming과 Request/Response Data 직렬화 가능
- War와 비슷하게 Zip 형태로 서비스 배포 및 등록
- 등록된 서비스의 통제 및 모니터링 등의 MetaService 가능
- 별도의 Java Developed Aradon Client



하고 싶은 것

- Aradon Agent
  Service Dynamic Deploy & ReDeploy
  Java Agent를 활용한 Debugging Mode Service 개발

- Repository Service(SaaS)
  Data, Chunk, Streaming

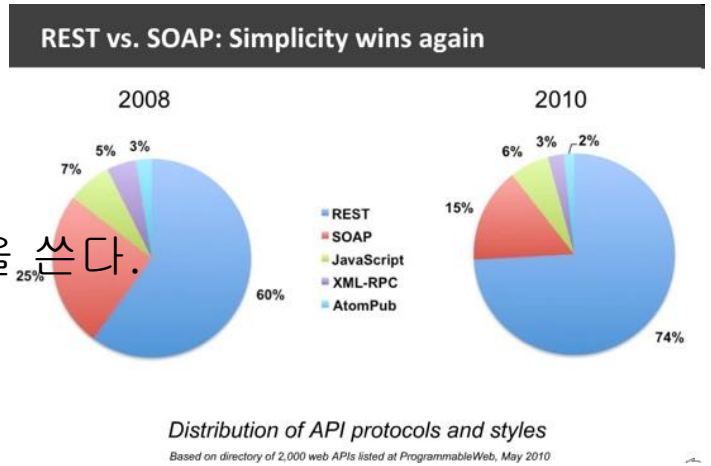
못 하는 것

- IaaS
  - AWS IaaS 사용

- 컴퓨팅 가상화 플랫폼 서비스
  - AWS나 KT의 가상화 플랫폼을 사용
  - 컴퓨팅 가상화와 관련된 OpenSource 활용

안 하는 것

- Servlet Container Service
  - 간단한 Servlet Container : http://winstone.sourceforge.net/
- View Template Service
  - 누구에게나 맞는 모자는 없다.
  - 그냥 Struts, Spring의 JSTL을 쓰거나 Velocity등 Opensource 이용
- Cluster Mode Server
- WSDL WebService
  아주 간단한걸 하기 위해 너무 복잡한 기술을 쓴다.
  Learning Curve가 높다.



왔다간거 다 알고있습니다.

댓글이라도 달아 주시지 그러세요~?

Reference

- GitHub

  ServerHome : https://github.com/bleujin/aradon

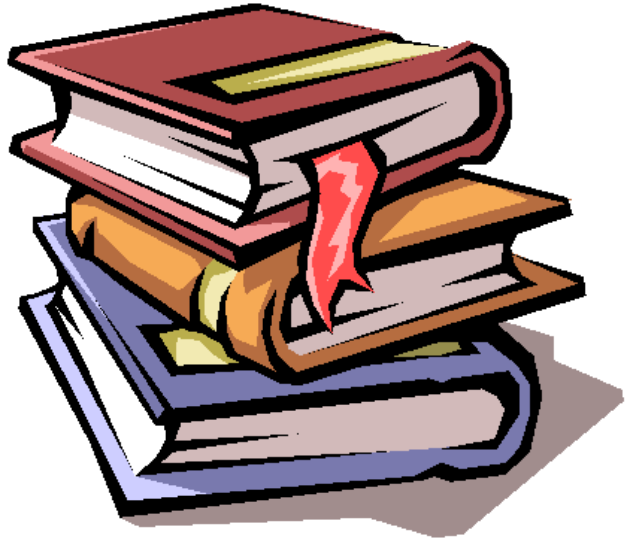
  ClientHome : https://github.com/bleujin/aradonClient

  ExtendHome : https://github.com/bleujin/aradonExtend

  doc : http://bleujin.springnote.com/pages/10761118

- AWS

  https://console.aws.amazon.com/ec2/home



Thank You