클라우드기초

4주차 최종석(jschoi@ssu.ac.kr)

01 IaaS의 가상화

» 스토리지 가상화

클라우드 이전에도 존재했던 기술

→ 물리적인 저장장치를 논리적으로 나누거나 결합시켜 사용률을 극대화시키는 추상화 기술

→ RAID된 장치를 가상화함

스토리지 가상화 특징

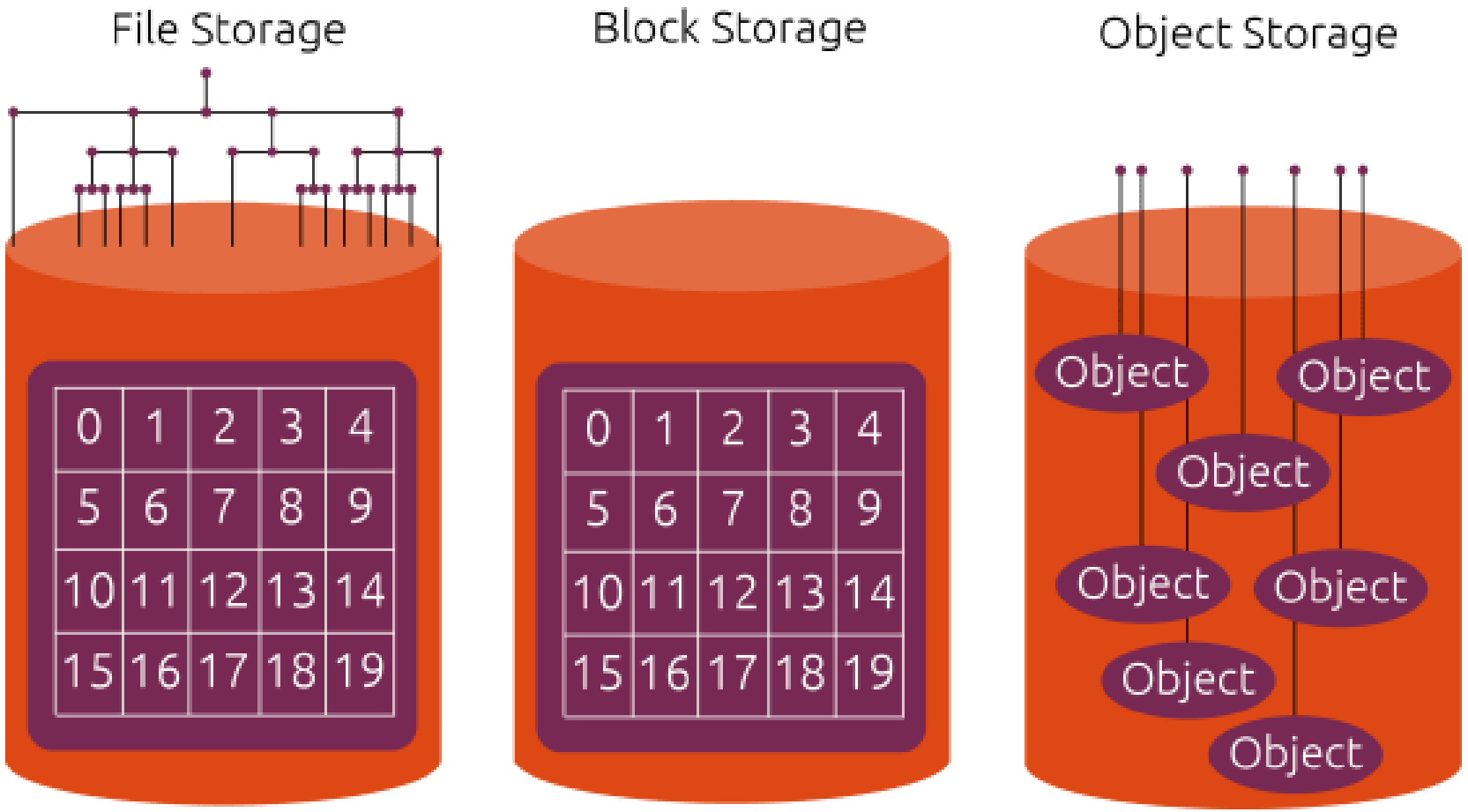
→ 1. 저장 디바이스의 복잡한 원리 등 실제 구조를 숨김

2. 여러 개의 디스크 드라이브를 그룹화하여 하나의 가상 드라이브로 통합 운용이 가능

3. 일부 디스크에 장애가 발생해도 서비스 운용을 계속함

4. 사용 중에도 디스크 용량의 재할당(추가 확장, 축소 등) 가능

» 스토리지 종류


File Storage
Block Storage
Object Storage
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
Object
Object
Object
Object
Object
Object
Object

» 스토리지 종류

파일 스토리지

기본적으로 사용하는 파일 스토리지

→ 폴더/디렉토리 구조를 통해 계층 구조에 데이터를 파일 형태로 저장하는 방식

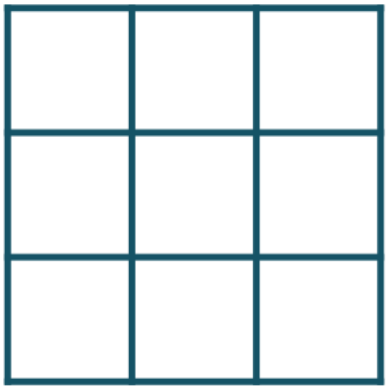
ex) NAS(Network Attached Storage)

NAS server, 외부 저장소

파일들은 이름, 위치, 생성일, 수정일, 크기 등의 제한적인 메타데이터를 가지고 있어 파일이 늘어나면 데이터가 늘어나고, 파일을 찾는 데에도 문제가 발생할 수 있음

» 스토리지 종류

파일 스토리지 : 폴더 단위로 저장되는 계층적인 구조. 검색이 느림.
블록 스토리지 : 매핑하는 구조이므로 검색이 빠름.



블록 스토리지

→ 데이터를 고정된 크기의 블록으로 처리하여 저장하는 방식

ex) SAN(Storage Area Network)

각각의 블록은 고유한 주소를 가지고, 이 주소를 통해 블록을 재구성하여 데이터를 불러옴

운영체제마다 파일 시스템과 종속되어있음.

계층구조가 없기 때문에, 데이터를 신속하게 검색할 수 있으며, 파티션 분할 등이 가능하여 서로 다른 운영체제에서 액세스할 수 있음. 대규모 DB운영에 적합

하지만 블록 스토리지는 운영체제와 독립적임.

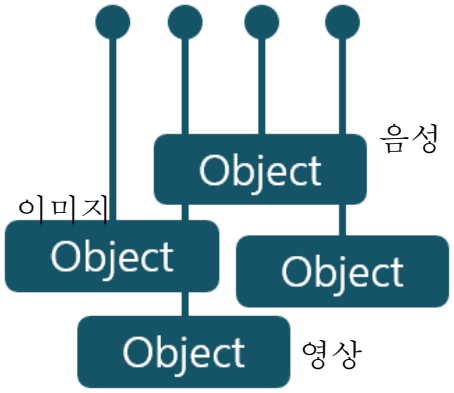
구축에 비용이 많이 들고, 메타데이터가 제한적이기 때문에 데이터 단위가 아닌 어플리케이션, DB수준에서 작업을 진행하여 관리자의 부담이 있음

돈많이 듬 왜? 큰파일단위는 저장하기 좋음. 작은 여러가지의 세세한 파일들은 메타데이터가 제한되어 블록이 많이 필요함.

01 IaaS의 가상화

» 스토리지 종류

파일 스토리지, 블록 스토리지의 단점을 극복하기 위해 나온 스토리지



객체(오브젝트) 스토리지

→ 각각의 데이터 단위로 저장되는 데이터 저장소 유형

PDF, 비디오, 오디오, 텍스트, 웹사이트 데이터 등 파일 유형별로 객체화 하고 객체의 이름을 색인하는 테이블이 존재하며, 객체의 이름이 키가 됨

계층구조 없이 평면 구조로 데이터를 저장하고, 키를 통해 색인테이블을 참조하여 해당 데이터를 편리하게 접근할 수 있음

빠른 접근 가능, 하지만 수정힘듬

객체를 수정하기 힘들기 때문에, 덮어쓰는 방식을 사용
자주 변경되는 데이터에는 맞지 않고, 수정이 잘 일어나지 않는 이미지 영상에 적합

관리자 부담때문에 객체 스토리지나 파일 스토리지를 많이 사용한다.

여기서 4,5주차 강의자료로

00 용어의 설명과 정리

» 알아 두어야 할 용어

AngularJS

React

Bootstrap

Javascript

Node.js

Spring, Spring boot

전자정부프레임워크

Native App, Hybrid App, Progressive Web App

Object-C

Swift

Kotlin

Flutter, Dart

Java

Python(Django, Pytorch, Tensorflow, Pandas, Numpy)

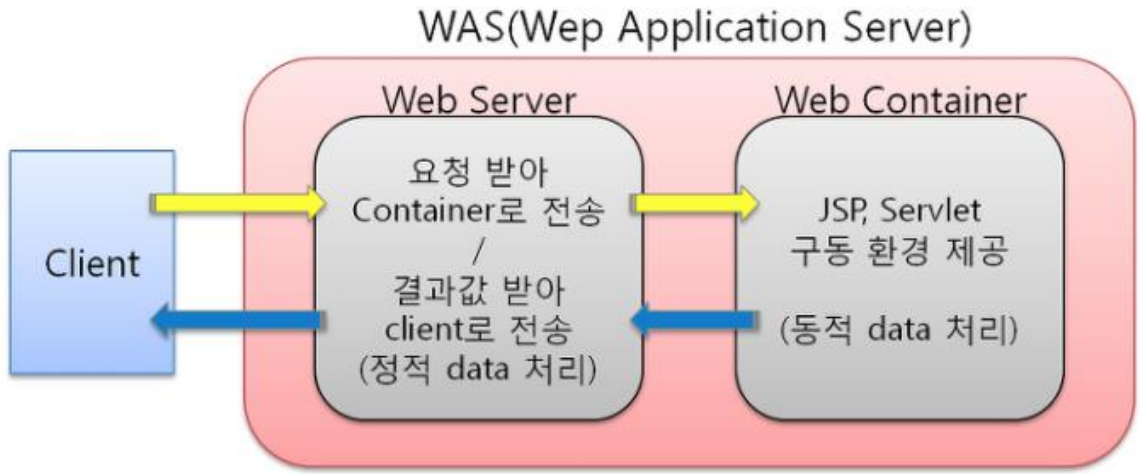
C#, R, PHP, RUBY, GO, MATLAB, SCALA

Hadoop eco System, SAP-ABAP

» 알아 두어야 할 용어

API

WAS(Web Application Server)



WAR, JAR(Web Application ARchive, Java ARchive)

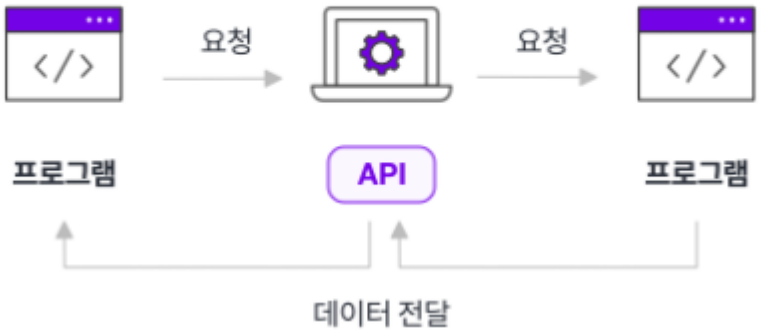
API

Application Programming Interface

어떤 서버의 특정한 부분에 접속해서 그 안에 있는 데이터와 서비스를 이용할 수 있게 해주는 소프트웨어 도구

- 점원의 역할

- API의 역할



API

» Application Programming Interface

서버와 어플리케이션간 출입구 역할은 한다.
단 허용된 사람들에게만 접근성을 부여한다.(Key, 토큰 사용)

기본적으로는 jhttps처럼 프로토콜
자체에서 허용을 받고 접근하지만
더 보안성을 높이려면 key나 토큰 사용.

기계/운영체제 등과 상관없이 누구나 동일한 액세스를 얻을 수 있게 해 준다.

응용프로그램과 운영체제간, 응용프로그램과 프레임워크(플랫폼)간
기능을 제어할 수 있게 만든 인터페이스이다.

SOAP vs REST

» Simple Object Access Protocol, Representational State Transfer

차이점	SOAP 표준이 되지 못함. MS 발.	REST 오픈소스 발.
유형	프로토콜	아키텍처 스타일
기능	기능 위주: 구조화된 정보 전송	데이터 위주: 데이터를 위해서 리소스에 접근
데이터 포맷	XML만 사용	일반 텍스트, HTML, XML, JSON 등 다양한 포맷을 허용
보안	WS-Security와 SSL을 지원	SSL과 HTTPS를 지원
대역폭	상대적으로 더 많은 리소스와 대역폭이 필요	상대적으로 리소스가 적게 필요하고, 무게가 가벼움
데이터 캐시	캐시를 사용할 수 없음	캐시를 사용할 수 있음
페이로드 처리	엄격한 통신 규약을 갖고 있으며, 모든 메시지는 보내기 전에 알려져야 함	미리 알릴 필요 없음

REST
웹
어플
관련된 데에서는
'승'

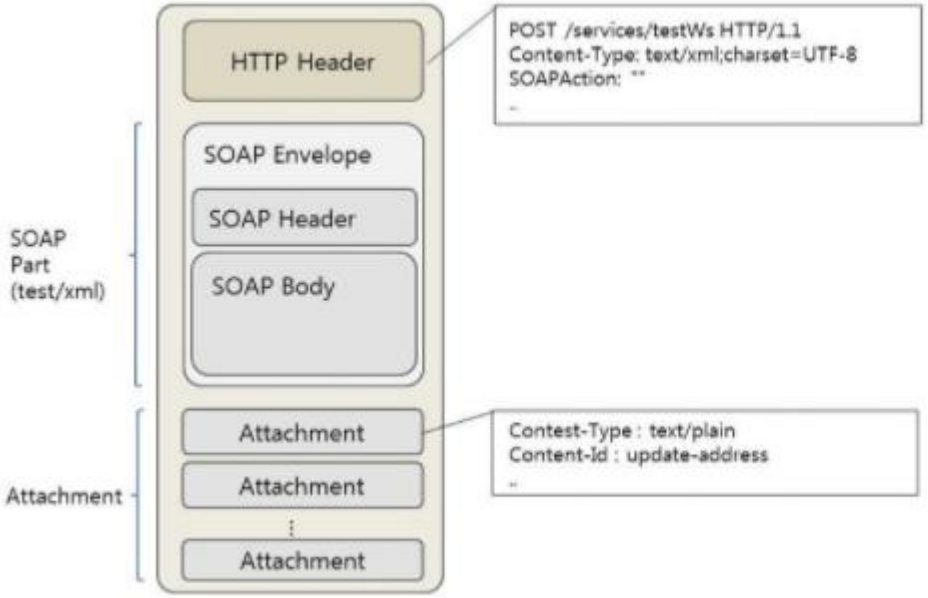
SOAP vs REST

» Simple Object Access Protocol

완벽한 규격화를 지향했다

네트워크를 통해서 컴퓨팅 자원 간 통신을 가능하게 해주는 프로토콜

보안 수준이 엄격하며, 일관적인 데이터 전공 및 성공/반복 실행 로직이 규정되어 있기 때문에 통신의 신뢰성을 보장



SOAP vs REST

» Simple Object Access Protocol

```
<?xml version="1.0"?>

<soap:Envelope
        xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
        soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

  <soap:Header>
  ...
  </soap:Header>

  <soap:Body>
      ...
      <soap:Fault>
      ...
      </soap:Fault>
  </soap:Body>

</soap:Envelope>
```

SOAP vs REST

» Simple Object Access Protocol

Result Item

```
<resultElements xmlns:ns3="http://schemas.xmlsoap.org/soap/encoding/"
   xsi:type="ns3:Array" ns3:arrayType="ns1:ResultElement[10]">
<item xsi:type="ns1:ResultElement">
  <URL xsi:type="xsd:string">http://www.yahoo.com/</URL>
  <cachedSize xsi:type="xsd:string">32k</cachedSize>
  <directoryCategory xsi:type="ns1:DirectoryCategory">
    <fullViewableName xsi:type="xsd:string">
    Top/Computers/Internet/Searching/Directories/Yahoo</fullViewableName>
    <specialEncoding xsi:type="xsd:string"></specialEncoding>
  </directoryCategory>
  <directoryTitle xsi:type="xsd:string">Yahoo!</directoryTitle>
  <hostName xsi:type="xsd:string"></hostName>
  <relatedInformationPresent xsi:type="xsd:boolean">true</relatedInformationPresent>
  <snippet xsi:type="xsd:string">... Business··· Media Newspapers, TV... </snippet>
  <summary xsi:type="xsd:string">The first large scale directory of the Internet...</summary>
  <title xsi:type="xsd:string">Yahoo!</title>
</item> ···
```

SOAP vs REST

» Representational State Transfer

웹 서비스와 모바일 어플리케이션 경량화의 필요에 맞춘 아키텍처 원칙 세트이며 설계적 지침

아주 간단한 정도의 설계적인 아키텍쳐적인 지침만 지키면 됨.

HTTP 프로토콜을 기반으로 웹에 최적화되어 있고 인간과 기계가 모두 읽을 수 있으며, 구성요소 간 통합된 인터페이스가 필요

중간 매체없이 고유 URL을 통해 직접 바로 전송하기 때문에 빠르며 특정 개발환경을 셋팅 할 필요가 없음

HTTP URI를 통해 자원을 명시하고 HTTP Method를 통해 해당 자원에 대한 CRUD 오퍼레이션을 적용하는 것

SOAP vs REST

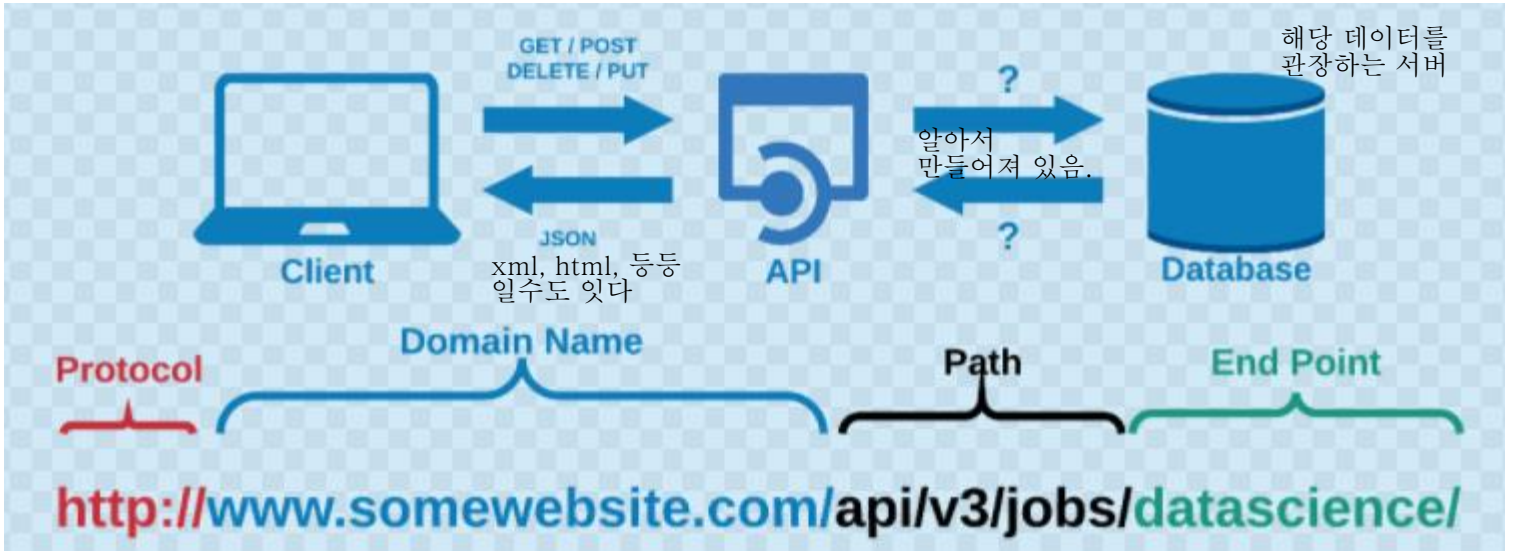
» Representational State Transfer

url로 리퀘스트하고

Json xml 등등으로 파일로 응답 받음.

REST API 설계 예시

CRUD	HTTP verbs	Route
resource들의 목록을 표시	GET	/resource
resource 하나의 내용을 표시	GET	/resource/:id
resource를 생성	POST	/resource
resource를 수정	PUT	/resource/:id
resource를 삭제	DELETE	/resource/:id



SOAP vs REST

받는 데이터 형식 중에서
제일 많이 사용되는 확장자는 JSON

» Representational State Transfer – JSON

JavaScript Object Notation
키–값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 사용

- `id`
- `first_name`
- `last_name`
- `dob` (생년월일)
- `addresses` (중첩 및 반복 필드)
  - `addresses.status` (현재 또는 이전)
  - `addresses.address`
  - `addresses.city`
  - `addresses.state`
  - `addresses.zip`
  - `addresses.numberOfYears` (주소 연도)

```
[
    {
        "name": "id",
        "type": "STRING",
        "mode": "NULLABLE"
    },
    {
        "name": "first_name",
        "type": "STRING",
        "mode": "NULLABLE"
    },
    {
        "name": "last_name",
        "type": "STRING",
        "mode": "NULLABLE"
    },
    {
        "name": "dob",
        "type": "DATE",
        "mode": "NULLABLE"
    },
    {
        "name": "addresses",
        "type": "RECORD",
        "mode": "REPEATED",
        "fields": [
            {
                "name": "status",
                "type": "STRING",
                "mode": "NULLABLE"
```

SOAP vs REST

» Representational State Transfer – BSON

Binary JavaScript Object Notation
키-값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 사용

문자열이 아닌 바이너리로

```
{
  "BSON": [
   "awesome", 5.05, 1986
  ]
}
```

파싱 시스템을 사용지 못하는
아주 로우레벨 게계나 마이크로 기계들한테
BSON이 유용하다

```
₩x31₩x00₩x00₩x00₩x04BSON₩x00
₩x26₩x00₩x00₩x00₩x020₩x00₩x08₩x00₩x00₩x00awesome₩x00₩x01
1₩x00₩x33₩x33₩x33₩x33₩x33₩x33₩x14₩x40₩x102₩x00₩xc2₩x07₩x00₩x00₩x00₩x00
```

YAML

요즘 또 많이 사용되는 형식

YAM

» YAML

E-mail 양식에서 개념을 얻어 만들어진 데이터 직렬화 양식

문법이 이해하기 쉽고, 가독성이 좋도록 디자인되었으며, 고급 컴퓨터 언어에 적합

Xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <apiVersion>v1</apiVersion>
  <kind>Pod</kind>
  <metadata>
    <name>hello-pod</name>
    <labels>
      <app>hello</app>
    </labels>
  </metadata>
  <spec>
    <containers>
      <name>hello-container</name>
      <image>tmkube/hello</image>
      <ports>
        <containerPort>8000</containerPort>
      </p
    </co
  </spec>
</root>
```

Object

Array

Key

Value

Json

```
{
  "apiVersion": "v1",
  "kind": "Pod",
  "metadata": {
    "name": "hello-pod",
    "labels": {
      "app": "hello"
    }
  },
  "spec": {
    "containers": [
      {
        "name": "hello-container",
        "image": "tmkube/hello",
        "ports": [
          {
            "containerPort": 8000
          }
        ]
      }
    ]
  }
}
```

Object

Array

Key

Value

요즘 클라우드에서 만히 쓰인다. 쿠버네티스에서

Yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-pod
  labels:
    app: hello
spec:
  containers:
  - name: hello-container
    image: tmkube/hello
    ports:
    - containerPort: 8000
```

Object

Array

Key

Value

쿠버네티스에서 배포할때 YAML을 이요하여 작성한다.

다시 4,5주차 교육자료로