



서비스 브로커 & 빌드팩

교재

NIA 한국정보화진흥원



A top-down view of a wooden desk with a laptop, glasses, a coffee cup, and a small plant. The text "02. Service Package 배포 및 관리" is overlaid on the desk.

02. Service Package 배포 및 관리

M1. Cloud Basic

01. Cloud 이해

02. Cloud Model 및 특징 이해

M2. PaaS-TA 개발 실무

01. PaaS-TA 이해

02. PaaS-TA 개발 환경의 이해

03. PaaS-TA 개발도구 이해 및 실습

04. PaaS-TA 개발 검증 및 문제해결

M3. PaaS-TA 배포 및 운영

01. PaaS-TA 배포 및 관리

02. Service Package 배포 및 관리

PaaS-TA Service Package ✓
concept

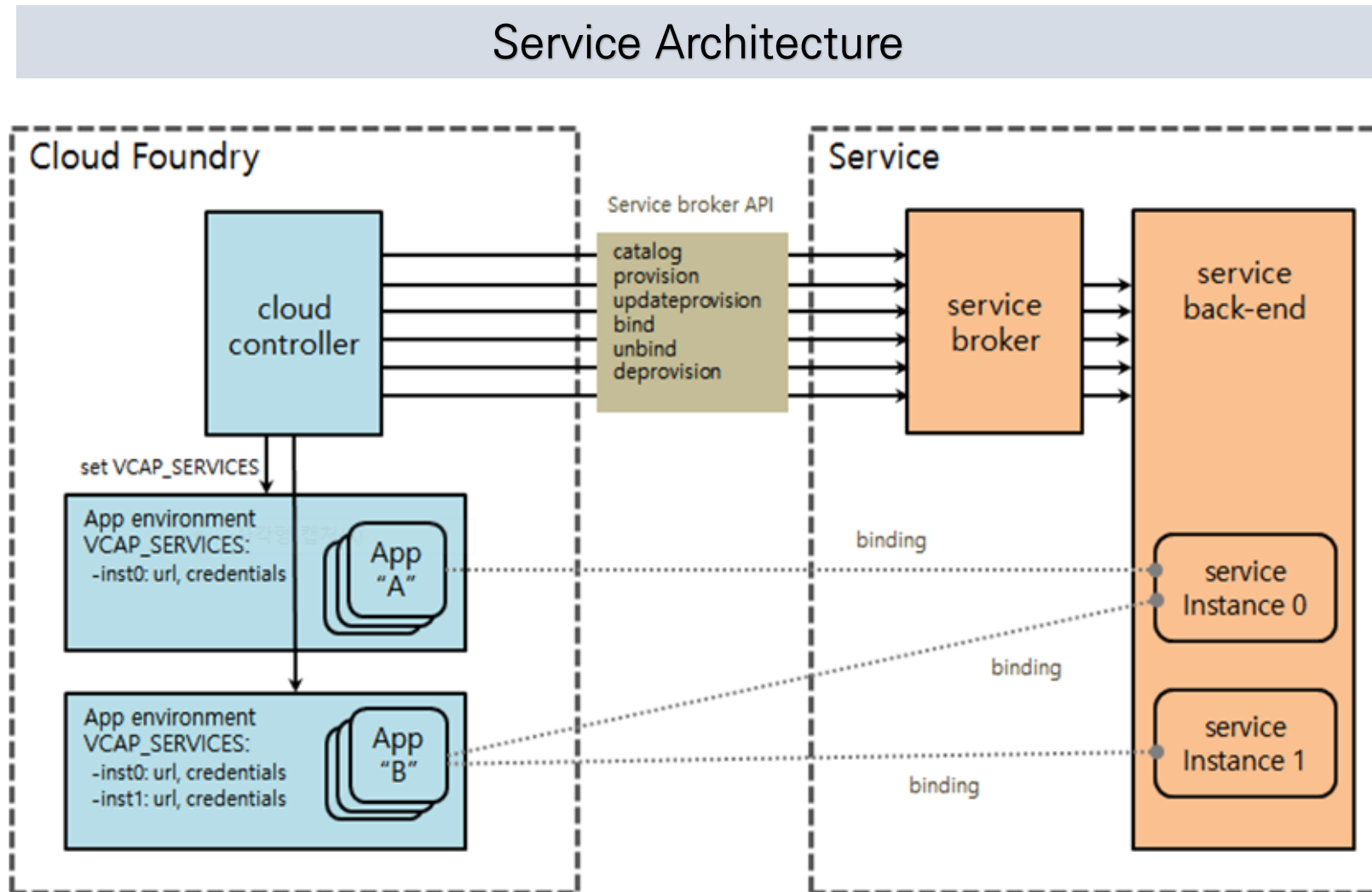
Service Broker 예제 분석

Service Broker 등록 및 활성화

03. Custom Buildpack 개발

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명



PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Service API

개방형 클라우드 플랫폼 **Service API**는 **Cloud Controller**와 **Service Broker** 사이의 규약을 정의

 Note : catalog, provision, deprovision, update provision plan, bind, unbind

Broker는 HTTP (or HTTPS) endpoints URI 형식으로 구현됨

하나 이상의 Service가 하나의 Broker 에 의해 제공 될 수 있고, 로드 밸런싱이 가능하게 수평 확장성 있게 제공 될 수 있음

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Service API

서비스는 Service Broker API 라고 불리우는 cloud controller 클라이언트 API를 구현하여 개방형 클라우드 플랫폼에서 사용됨

Services API는 독립적인 cloud controller API의 버전

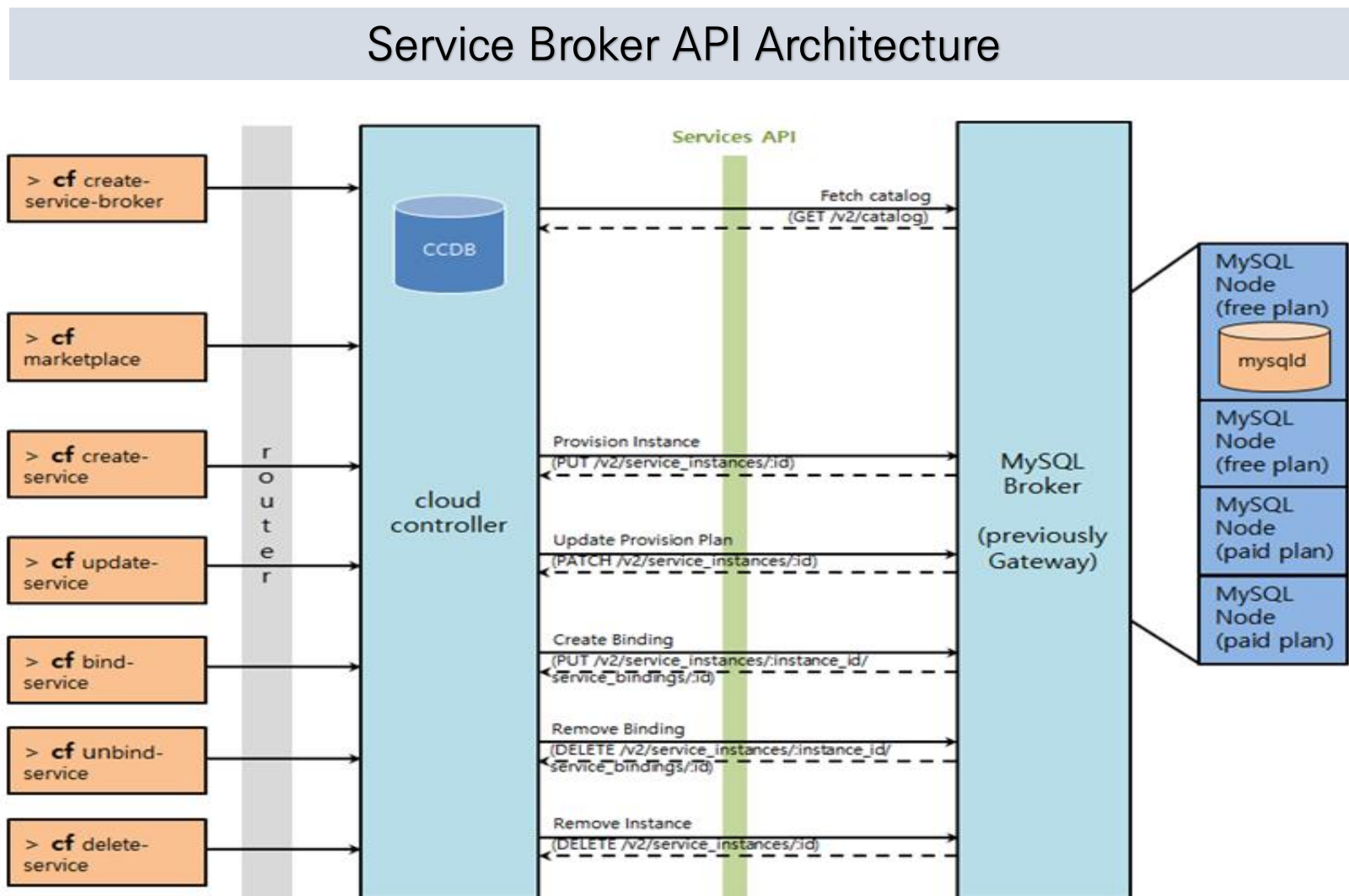
이는 플랫폼에서 외부 application을 이용 가능하게 함

 Note : database, message queue, rest endpoint , etc

Service Broker는 RESTful API로 구현하고 Cloud Controller에 등록함

PaaS-TA Service Package concept

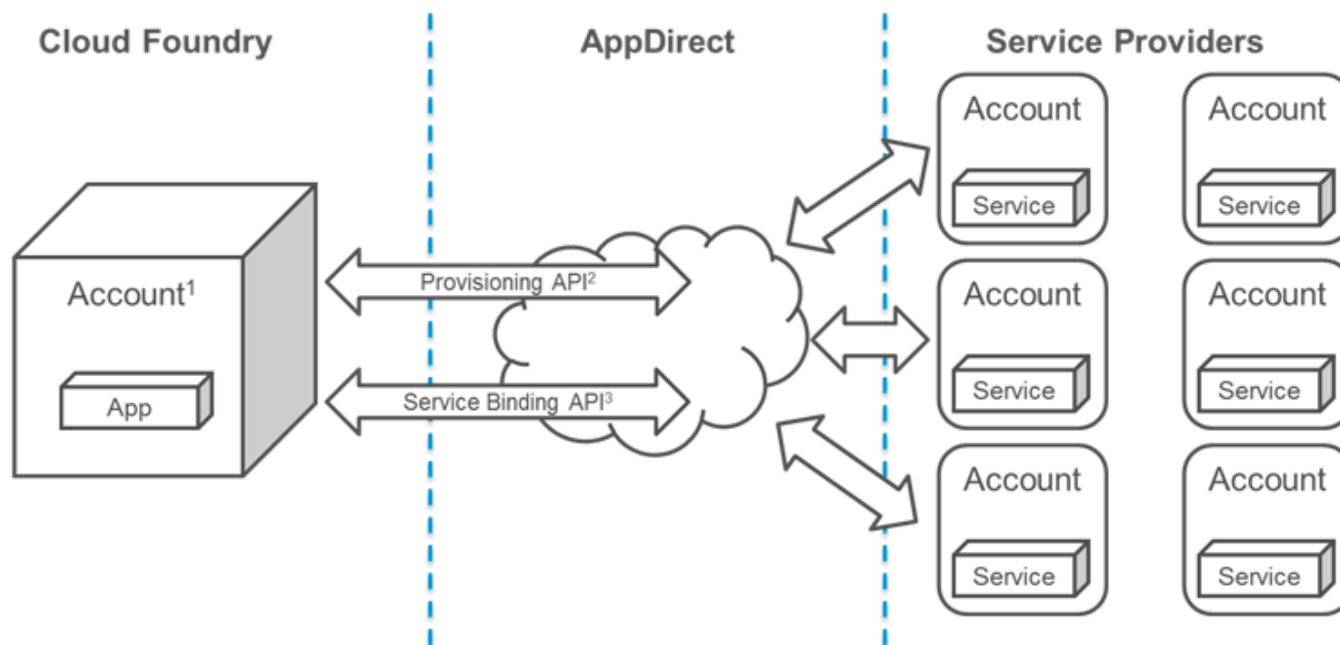
» 서비스 예제를 통한 컨셉 설명



PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Pivotal(Cloud Foundry) Marketplace Model



AppDirect

클라우드 서비스 marketplace 및 관리 솔루션의 선두 업체이고 많은 글로벌 회사의 marketplace를 구축 (삼성, Cloud Foundry, ETC)

Cloud Foundry 서비스 중개(brokerage) 기능과 부가 서비스를 제공

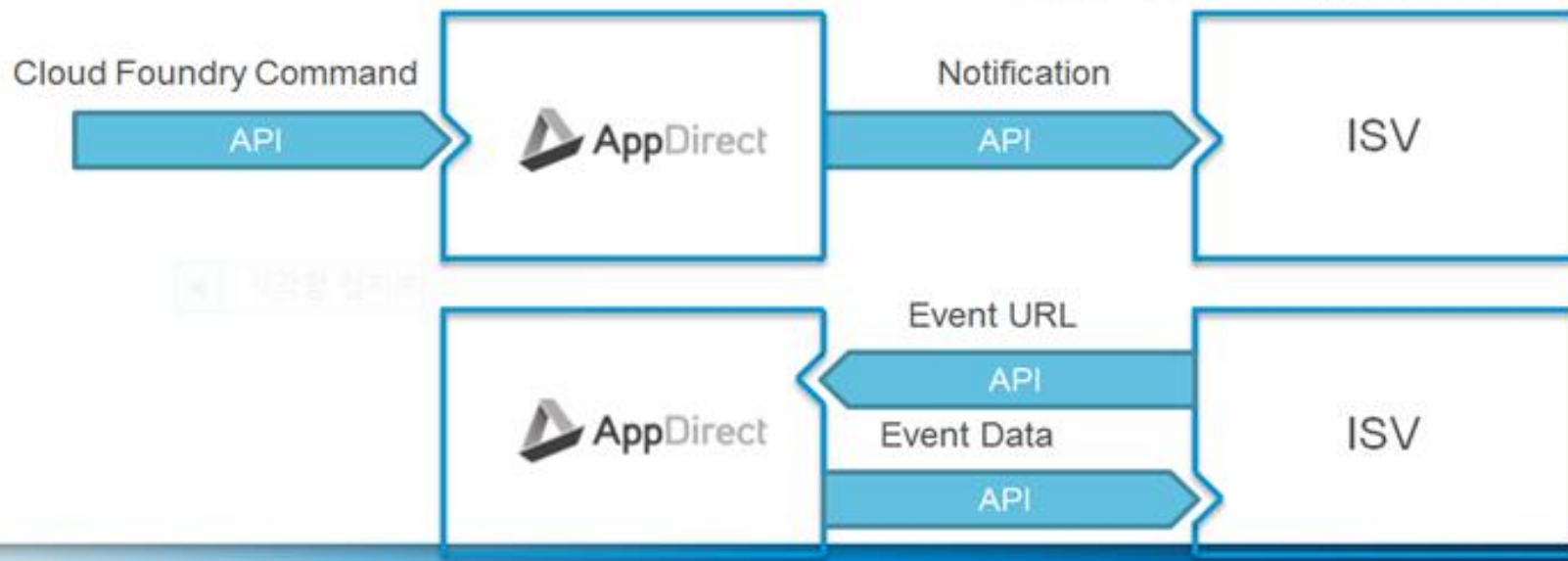
PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Service Provider Integration

- cf create-service
- cf bind-service
- cf unbind-service
- cf delete-service

- SUBSCRIPTION_ORDER
- ADDON_ORDER
- ADDON_BIND
- ADDON_UNBIND
- ADDON_CANCEL
- SUBSCRIPTION_CANCEL



PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

개발 가이드

개방형 클라우드 플랫폼은 서비스 제공자가 **6가지의 Service Broker API를 구현**

이때 2.4 Pivotal Marketplace Model를 이용해서 AppDirect에서 제공중인 서비스 제공자와 협의 하여 AppDirect의 중개 기능을 이용해서 제공할 수도 있음

또한 Broker는 별도의 애플리케이션으로 구현하든지 기존 서비스에 필요한 HTTP endpoint를 추가함으로써 구현 될 수 있음

개발 가이드는 **Service Broker에서 service back-end를 제어하는 방식**을 가이드 함



Note : AppDirect를 사용하는 경우 다음을 참고하여 개발
<http://go.appdirect.com/request-more-information>

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

개발 가이드

Service Broker는 6개의 기본 API 기능이 필요

| Service Broker API | Routes(API) | Method | Description |
|--------------------|--|--------|-----------------------------------|
| catalog | /v2/catalog | GET | 서비스 및 서비스 plans 정보 조회 |
| provision | /v2/service_instances/:id | PUT | 서비스를 위한 인스턴스 생성 |
| deprovision | /v2/service_instances/:id | DELETE | 이전에 생성된 서비스 인스턴스 삭제 |
| updateprovision | /v2/service_instances/:id | PATCH | 이전에 생성된 서비스 인스턴스 Plan 수정 |
| bind | /v2/service_instances/:id /service_bindings/:id | PUT | 서비스 사용에 관련 사용자 생성 및 권한 등 설정 정보 생성 |
| unbind | /v2/service_instances/:id /service_bindings/:id | DELETE | 서비스 사용 설정 정보 삭제 |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

개발 가이드

버전 정보

본 가이드는 2.5 버전을 기준으로 작성함

| CURRENT VERSION | PAST VERSIONS |
|--|------------------|
| 2.5 (cf-release 209 버전 이상, CLI version 6.12.1) | 2.4 |
| | 2.3 |
| | 2.2 |
| | 2.1 |
| | 2.0 |
| | v1 (unversioned) |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

개발 가이드

인증

Cloud Controller는 모든 요청에 HTTP 기본 인증(인증 헤더)을 사용하여 Broker와 인증하여 사용자 이름과 암호를 포함하지 않는 모든 Broker 등록을 거부

Broker는 사용자 이름과 암호를 확인하고 자격 증명이 유효하지 않은 경우 401 Unauthorized 메시지를 반환

Cloud Controller에서 추가 보안이 요구되는 경우 SSL을 사용하여 브로커에 접속 지원

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

서비스 Catalog는 서비스 및 서비스 Plan의 정보를 조회

Cloud Controller는 처음에 모든 Broker에서 endpoint를 취득해서 Cloud Controller 데이터베이스에 저장되어 있는 user-facing service catalog를 조회

또한 Cloud Controller는 Broker가 업데이트 될 때 마다 catalog를 업데이트함

Catalog API를 구현하면 CF CLI를 통해서 Service Broker를 등록할 수 있음

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Request

Route

```
GET /v2/catalog
```

cURL

```
curl -H "X-Broker-API-Version: 2.4"  
http://username:password@broker-url/v2/catalog  
예) curl -H "X-Broker-API-Version: 2.4"  
http://admin:eea139af583c@10.30.40.61/v2/catalog
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Response

Status Code

| SATUS CODE | DESCRIPTION |
|------------|--------------------------|
| 200 OK | Response body 정보는 아래에 제공 |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Response

Body (* 필드는 필수)

| RESPONSE FIELD | TYPE | DESCRIPTION |
|----------------|------------------|---|
| services* | array-of-objects | 서비스 객체들의 스키마를 정의 |
| id* | string | 카탈로그에 요청시 서비스의 상관 관계를 구별하는 식별자. 개방형 클라우드 플랫폼에서 unique해야 하기 때문에 GUID를 사용하기를 권장 |
| name* | string | 카탈로그에 표시되는 service 이름 (소문자, 공백없음) |
| description* | string | 카탈로그에 표시되는 서비스의 설명 |
| tags | array-of-strings | Tags는 빌드팩 또는 다른 서비스에서 로직을 변경하지 않고 교체할 수 있게 서비스를 가능하게하며 서비스의 분류, 속성, 또는 기반 기술을 노출할 수 있는 유연한 메커니즘을 제공(MySQL, relation, redis, key-value, 캐싱, 메시징, AMQP) |
| metadata | object | 서비스 제공을 위한 메타 데이터의 목록 |
| requires | array-of-strings | 사용자가 서비스를 제공하는 권한 목록. 현재는 syslog_drain 권한만 지원 |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Response

Body (* 필드는 필수)

| RESPONSE FIELD | TYPE | DESCRIPTION |
|------------------|------------------|---|
| plan_updateable | boolean | 서비스 plans의 다운그레이드/업그레이드 지원 여부 |
| plans* | array-pf-objects | 서비스에 대한 plans의 스키마를 정의 |
| description* | string | 카탈로그에 표시되는 plan의 설명 |
| metadata | object | 서비스 plan을 위한 메타 데이터의 목록 |
| free | boolean | 유료 서비스를 제공할지 설정. Default는 true |
| Dashboard_client | object | 서비스에 대한 대시보드 SSO 기능을 활성화하는데 필요한 데이터를 포함 |
| id | string | 서비스를 이용하고자 하는 Oauth2 클라이언트의 ID |
| secret | string | 대시보드 토큰 서버 인증에 사용하는 공유 secret 정보 |
| redirect_uri | string | UAA SSO 인증을 위한 서비스 도메인 URL 정보 |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Response

Service Metadata

| BROKER API FIELD | TYPE | DESCRIPTION | Cloud Controller API FIELD |
|------------------------------|--------|------------------------|----------------------------|
| name | string | 카탈로그에 표시되는 서비스의 이름 | label |
| description | string | 서비스에 대한 설명 | description |
| metadata.displayName | string | 그래픽 클라이언트에 표시되는 서비스 이름 | extra.displayName |
| metadata.imageUrl | string | 이미지 URL | extra.imageUrl |
| metadata.long.Description | string | 서비스 상세 설명 | extra.longDescription |
| metadata.providerDisplayName | string | 실제 서비스를 제공하는 기업 이름 | extra.providerDisplayName |
| metadata.documentationURL | string | 서비스를 위한 문서 링크 URL | extra.documentationURL |
| metadata.supportUrl | string | 서비스 지원 URL | extra.supportUrl |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Response

Plan Metadata

| BROKER API FIELD | TYPE | DESCRIPTION | Cloud Controller API FIELD |
|------------------------|------------------|------------------------|----------------------------|
| name | string | 카탈로그에 표시되는 서비스의 이름 | name |
| description | string | 서비스에 대한 설명 | description |
| metadata.bullets | array-of-strings | 그래픽 클라이언트에 표시되는 서비스 이름 | extra.bullets |
| metadata.costs | cost object | 이미지 URL | metadata.costs |
| metadata.displayatName | string | 서비스 상세 설명 | extra.displayName |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Response

Quota Plan

| NAME | DESCRIPTION | VALID VALUES | EXAMPLE VALUE |
|----------------------------|--|---|---------------|
| name | Plan을 식별하는데 사용 | '숫자', '_' 및 '문자' 형식. Quota plan 이름은 계정 내에서 고유해야 함 | silver_quota |
| memory_limit | 허용되는 최대 메모리 사용량(MB) | integer | 2048 |
| non_basic_services_allowed | 유료 서비스를 제공할지 설정. false로 설정하면 marketplace에는 표시되지만 인스턴스는 제공되지 않음 | true/false | true |
| total_routes | 허용되는 최대 라우터 개수 | integer | 500 |
| total_services | 허용되는 서비스 개수 | integer | 25 |
| trial_db_allowed | 기존 필드. 값은 무시 | true/false | true |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

sample body response message

```
{
  "services": [
    {
      "id": "44b26033-1f54-4087-b7bc-da9652c2a539",
      "name": "p-mysql",
      "description": "A MySQL service for application development and testing",
      "tags": [
        "mysql"
      ],
      "metadata": {
        "displayName": "MySQL for Pivotal CF",
        "imageUrl": "data:image/png;base64,iVBORw0KGgoAAAA...AAAAAEIFTkSuQmCC",
        "longDescription": "Provisioning a service instance creates a MySQL database. Binding applications to the instance creates unique credentials for each application to access the database.",
        "providerDisplayName": "Pivotal Software",
        "documentationUrl": "http://docs.gopivotal.com/",
        "supportUrl": "http://gopivotal.com/support/"
      }
    }
  ]
}
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

sample body response message

```
"plans": [  
  {  
    "id": "ab08f1bc-e6fc-4b56-a767-ee0fea6e3f20",  
    "name": "100mb-dev",  
    "description": "Shared MySQL Server",  
    "metadata": {  
      "costs": [  
        {  
          "amount": {  
            "usd": 0  
          },  
          "unit": "MONTH"  
        }  
      ],  
    }  
  ],  
]
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

sample body response message

```
"bullets": [
  "Not for production use – server is not replicated",
  "Shared MySQL server",
  "100 MB storage",
  "40 concurrent connections"
],
"displayName": "(( merge || \"100 MB Dev\" ))"
},
],
"bindable": true,
"dashboard_client": {
  "id": "p-mysql",
  "secret": "eaa139af583c",
  "redirect_uri": "http://10.30.40.61/"
}
}
]
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Catalog Rest API 구현 - JAVA 방식

```
-- CatalogRestController.java (Spring 프레임워크 사용)

@Controller
@RequestMapping("/v2/catalog") // Spring 어노테이션을 사용
class CatalogRestController {
    def settings;

    @RequestMapping(method=RequestMethod.GET)
    @ResponseBody
    synchronized Map getCatalog() {
        if (!settings) {
            Yaml yaml = new Yaml();
            // settings.yml 파일 안에 서비스 정보와 plan 정보가 들어 있음
            settings = yaml.load(this.getClassLoader().getResourceAsStream("settings.yml"));
        }

        return settings;
    }
}
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Catalog Rest API 구현 - JAVA 방식

```
-- settings.yml 파일

services:
- name: p-mysql
  id: 3101b971-1044-4816-a7ac-9ded2e028079
  description: MySQL service for application development and testing
  tags:
    - mysql
    - relational
  max_db_per_node: 250
  metadata:
    provider:
      name:
    listing:
      imageUrl: ~
      blurb: MySQL service for application development and testing
```


PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

Catalog Rest API 구현 – JAVA 방식

plans:

- name: 5mb

id: 2451fa22-df16-4c10-ba6e-1f682d3dc9

description: Shared MySQL Server, 5mb persistent disk, 40 max concurrent connections

max_storage_mb: 5 # in MB

metadata:

- cost: 0.0

bullets:

- content: Shared MySQL server

- content: 5 MB storage

- content: 40 concurrent connections

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Catalog API 가이드

서비스 별 Catalog API 개발 명세 Catalog API 경우에는 서비스의 종류와 관계없이 Service 및 Plan 정보를 저장되어 있는 settings.yml 파일이나 기타 메타 파일 또는 소스 안에 정보를 저장한 후 제공

AppDirect 를 이용하는 경우에는 Catalog 정보를 조회해오는 AppDirect API를 호출하여 그 결과를 제공

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

Broker가 Cloud Controller로 부터 provision 요구를 수신하면 개발자를 위한 새로운 서비스 인스턴스를 생성

provision 시 서비스들의 종류에 따라 provision 결과는 다름

Mysql DataBase 인 경우에는 새로운 DATABASE 스키마를 생성

또한 non-data 서비스 인 경우의 provision은 기존 시스템에 계정을 얻는 의미 일 수도 있음

PaaS-TA Service Package concept


» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

Request

Route

PUT /v2/service_instances/:instance_id

 Note : 서비스 인스턴스의 instance_id는 Cloud Controller에 의해 제공되며 이 ID는 인스턴스 삭제, 바인드 및 바인드 해지에 사용됨

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id  
-d '{ "service_id": "service-guid-here", "plan_id": "plan-guid-here",  
"organization_guid": "org-guid-here", "space_guid": "space-guid-here" }' -X PUT  
-H "X-Broker-API-Version: 2.4" -H "Content-Type: application/json"
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

Request

Body

| RESPONSE FIELD | TYPE | DESCRIPTION |
|--------------------|-------------|--|
| service_id* | string | 카탈로그 내의 서비스의 ID는 카탈로그 endpoint에서 사용자가 provision 할 때 필요한 고유 식별자 |
| plan_id* | string | 서비스 내의 plan ID는 카탈로그 endpoint에서 사용자가 provision 할 때 필요한 고유 식별자 |
| organization_guid* | string | Cloud Controller에서 사용자가 사용하는 ORG GUID 값 |
| space_guid* | string | organization_huid 필드와 같이 SPACE GUID 값 |
| parameters | JSON object | JSON 형태의 피라미터 값을 제공 |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

Response

Status Code

| STATUS CODE | DESCRIPTION |
|--------------|--|
| 201 Created | 서비스 인스턴스 생성 |
| 200 OK | 서비스 인스턴스가 이미 존재하고 요청 된 매개 변수가 기존의 서비스 인스턴스와 동일한 경우 |
| 409 Conflict | 요청 된 서비스의 인스턴스가 이미 존재하는 경우 반환. 에러 메시지 형태는 {} 안에 description 필드를 사용 예) { “description”: “something went wrong. Please contact support at http://support.example.com .” } |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

Response

Body

모든 응답 bodies 는 JSON Object ({}) 형식으로 함

| RESPONSE FIELD | TYPE | DESCRIPTION |
|----------------|--------|---|
| dashboard_url | string | 서비스 인스턴스에 대한 웹 기반 대시보드 유저 인터페이스의 URL 예) { "dashboard_url": "http://mongomgmthosst/databases/9189kdfsk0vfnku" } |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

Dashboard Single Sign-On. Single Sign-On (SSO)는 개방형 클라우드 플랫폼 사용자들이 개방형 클라우드 플랫폼 자격 증명을 사용하여 third-party 서비스의 대시 보드에 접근

서비스 대시 보드는 서비스가 제공하는 기능의 일부 또는 전부를 사용할 수 있는 웹 인터페이스

SSO는 반복되는 로그인과 여러 서비스의 계정을 통합 관리한다. OAuth2 프로토콜 인증을 처리하기 때문에 사용자의 자격 증명은 직접 서비스로 전송하지 않음

SSO 기능을 사용하려면 Cloud Controller UAA client에 서비스 브로커의 생성 및 삭제 할 수 있는 권한이 있어야 함

이 클라이언트는 개방형 클라우드 플랫폼 설치시 구성

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

CF 설치시 Dashboard SSO 설정 예)

```
properties:  
  uaa:  
    clients:  
      cc-service-dashboards:  
        secret: cc-broker-secret  
        scope: openid,cloud_controller_service_permissions.read  
        authorities: clients.read,clients.write,clients.admin  
        authorized-grant-types: client_credentials
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

Provision Rest API 구현 - JAVA 방식

```
-- ServiceInstanceRestController.java (Spring 프레임워크 사용)

@Controller @RequestMapping("/v2/service_instances/{id}") class
ServiceInstanceRestController { @Autowired private ServiceInstanceService service;

@RequestMapping(method = RequestMethod.PUT) @ResponseBody Map
update(@PathVariable String id) { ServiceInstance instance = service.findById(id); //
Spring 프레임워크 사용으로 서비스 구현 if (!service.isExists(instance))
{ service.create(instance); // 서비스 인스턴스를 생성하는 부분 (개발 명세 내용 구현) }
return [:];

}}
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

서비스 별 Provision API 개발 명세

인스턴스 생성시 unique한 이름으로 생성

생성 요청한 인스턴스 ID가 이미 존재 하는지 체크

선택 한 plan 정보로 인스턴스가 생성 가능한지 체크 하고 가능할 경우 해당 인스턴스를 생성

인스턴스 생성이 완료 되면 위에서 기술된 JSON Object 형식으로 Cloud Controller에 전송

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

서비스 별 Provision API 개발 명세

RDBMS

Mysql 경우

- ✓ 생성할 데이터 베이스가 존재 하는지 체크
SHOW DATABASES LIKE '\${instance.database}'
- ✓ 새로운 데이터 베이스 생성
CREATE DATABASE IF NOT EXISTS \${instance.database}
- ✓ 생성 후 Dashboard 정보를 JSON Object 형식으로 Cloud Controller에 전송

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

서비스 별 Provision API 개발 명세

RDBMS

Cubrid DB 경우

- ✓ 데이터 베이스 생성할 디렉토리 생성 및 이동
`$ mkdir $ cd`
- ✓ 데이터 베이스 생성
`$ cubrid created--db-volume-size=100M --log-volume-size=100M en_US`
- ✓ 데이터 베이스 실행
`$ cubrid service start $ cubrid server start`

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

서비스 별 Provision API 개발 명세

대용량 저장소

GlusterFS 경우

- ✓ GlusterFS로 파일을 업로드 하기 위해서는 먼저 GlusterFS 와 OpenStack swift 로 service back-end로 구성하여 Object Storage 방식으로 파일을 업로드 다운로드를 할 수 있게 제공 (아마존 S3 방식과 유사)

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

서비스 별 Provision API 개발 명세

대용량 저장소

새로운 Swift Account를 생성

Method : PUT

Req URL : http(s)://[IP Address OR HostName]/auth/v2/[AccountID]

Header : X-Auth-Admin-User: .super_admin X-Auth-Admin-Key:
swauthkey X-Account-Meta-Quota-Bytes: [Size(Byte)]

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Provision API 가이드

서비스 별 Provision API 개발 명세

NoSQL DB

mongoDB 경우 새로운 데이터 베이스를 생성

use switched to db

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Update Instance API 가이드

Update Instance API는 기존의 서비스 인스턴스의 plan를 수정, 즉 서비스 인스턴스의 plan을 업그레이드나 다운그레이드 함

이 기능을 사용하려면 브로커는 카탈로그 endpoint 에서 “plan_updateable: true” 설정해야 함

이 옵션 필드가 포함되어 있지 않은 경우에는 service plan 변경 요청에 대해 의미 있는 오류를 반환하고 브로커는 API 호출을 하지 않음

이 필드가 포함된 경우 개방형 클라우드 플랫폼의 모든 plan 변경 요청에 브로커로 API 호출을 수행하며 브로커에서는 plan 지원 여부를 확인

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Update Instance API 가이드

Request

Route

```
PATCH /v2/service_instances/:instance_id
```

 Note : instance_id는 이전에 provision 서비스 인스턴스의 GUID

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id  
-d '{ "plan_id": "plan-guid-here" }' -X PATCH -H "X-Broker-API-Version: 2.4" -H  
"Content-Type: application/json"
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Update Instance API 가이드

Request

Body

| RESPONSE FIELD | TYPE | DESCRIPTION |
|---------------------------------|-------------|---|
| plan_id | string | 변경할 새로운 plan ID |
| service_id* | string | 카탈로그 내의 서비스의 ID는 카탈로그 endpoint 에서 사용자가 provision 할 때 필요한 고유 식별자 |
| parameters | JSON object | JSON 형태의 파라미터 값을 제공 |
| previous_values | object | 업데이트하기 전 인스턴스에 대한 정보 |
| previous_values.plan_id | string | 업데이트하기 전 plan ID |
| previous_values.service_id | string | 업데이트하기 전 서비스 ID |
| previous_values.organization_id | string | 업데이트하기 전 조직 ID |
| previous_values.space_id | string | 업데이트하기 전 스페이스 ID |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Update Instance API 가이드

Response

Status Code

| STATUS CODE | DESCRIPTION |
|--------------------------|---|
| 200 OK | 요청한 plan으로 변경하고 응답 본문은 “{}”으로 전송 |
| 422 Unprocessable entity | <p>요구 된 특정 plan 변경이 지원되지 않는 경우 예) 인스턴스 사용률이 요청한 계획의 할당량을 초과</p> <p>에러 메시지 형태는 {} 안에 description 필드를 사용 예) { “description”: “Something went wrong. Please contact support at http://support.example.com.” }</p> |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Update Instance API 가이드

Response

Body

모든 응답 bodies는 JSON Object ({}) 형식으로 함

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Update Instance API 가이드

Update Service instance Rest API 구현 - JAVA 방식

```
-- ServiceInstanceRestController.java (Spring 프레임워크 사용)

@Controller @RequestMapping("/v2/service_instances/{id}") class
ServiceInstanceRestController { @Autowired private ServiceInstanceService service;

@RequestMapping(method = RequestMethod.PATCH) @ResponseBody Map
updateInstance(@PathVariable String id) { ServiceInstance instance =
service.findById(id); // Spring 프레임워크 사용으로 서비스 구현 if
(!service.exists(instance)) { service.update(instance); // 서비스 인스턴스를 정보
수정하는 부분 (개발 명세 내용 구현) } return [:];

}}
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Update Instance API 가이드

서비스 별 Update Service instance API 개발 명세

현재 제공 중인 plan 정보와 변경 요청 받은 plan 정보가 다른지 체크

다운 그레이드 할 경우 이미 사용하는 용량이 다운 그레이드 할 용량보다 클 경우
에러를 발생시킴

업 그레이드 할 경우 plan 정보를 업데이트 함

예를들어 DBMS 서비스 경우 connection 수, storage 용량

변경된 내용을 Cloud Controller에 전달

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Deprovision API 가이드

브로커가 개방형 클라우드 플랫폼으로부터 deprovision 요청을 수신 할 때 provision 생성시 제공했던 모든 리소스를 삭제

Request

Route

```
DELETE /v2/service_instances/:instance_id
```


PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Deprovision API 가이드

Request

Parameters

| QUERY-STRING FIELD | TYPE | DESCRIPTION |
|--------------------|--------|------------------|
| service_id* | string | 카탈로그에 있는 서비스 ID |
| plan_id* | string | 카탈로그에 있는 Plan ID |

cURL

```
$ curl 'http://username:password@broker-  
url/v2/service_instances/:instance_id?service_id=  
service-id-here&plan_id=plan-id-here' -X DELETE -H "X-Broker-API-Version:  
2.4"
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Deprovision API 가이드

Response

Status Code

| Status Code | DESCRIPTION |
|-------------|------------------------------------|
| 200 OK | 서비스 인스턴스를 삭제. “{}” 메시지를 응답 |
| 410 Gone | 서비스 인스턴스가 존재하지 않을 경우. “{}” 메시지를 응답 |

Body

모든 응답 bodies는 JSON Object ({}) 형식으로 함

성공시 “{}” 값을 전송받음

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Deprovision API 가이드

Deprovision Rest API 구현 - JAVA 방식

-- ServiceInstanceRestController.java (Spring 프레임워크 사용)

```
@Controller @RequestMapping("/v2/service_instances/{id}") class
ServiceInstanceRestController { @Autowired private ServiceInstanceService
service;

@RequestMapping(method = RequestMethod.DELETE) @ResponseBody
Map destroy(@PathVariable String id) { ServiceInstance instance =
service.findById(id); if (service.isExists(instance)) { service.delete(instance);
// 서비스 instance 삭제 기능 구현 (개발 명세 내용 구현) } return [:] }
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Deprovision API 가이드

서비스 별 Deprovision API 개발 명세

삭제 요청하는 서비스 인스턴스가 존재 하는지 체크

인스턴스가 존재 하면 삭제할 인스턴스에 Application이 bind 되어 있는지 체크

만일 bind 되어 있는 Application이 존재 할 경우 Error를 Cloud Controller에 전송

bind 되어 있는 Application이 존재 하지 않을 경우 해당 서비스 인스턴스를 삭제

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Deprovision API 가이드

서비스 별 Deprovision API 개발 명세

RDBMS

Mysql 경우

- ✓ 데이터 베이스 삭제 DROP DATABASE IF EXISTS
`#{connection.quote_table_name(database_name)}`

Cubrid DB 경우

- ✓ 서비스 종료 후 데이터베이스 제거
`$ cubrid service stop $ cubrid deletedb`
- ✓ 제거한 데이터베이스 디렉터리 제거
`$ rm -rf <database 설치 path>/`

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Deprovision API 가이드

서비스 별 Deprovision API 개발 명세

대용량 저장소

GlusterFS 경우 Swift Account를 삭제

Method : DELETE

Req URL : http(s)://[IP Address OR HostName]/auth/v2/[AccountID]

Header : X-Auth-Admin-User: .super_admin X-Auth-Admin-Key:
swauthkey

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Deprovision API 가이드

서비스 별 Deprovision API 개발 명세

NoSQL DB

mongoDB 경우 데이터 베이스 삭제

```
use switched to db db.dropDatabase() { "dropped" : "", "ok" : 1 }
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Provision만으로 서비스를 사용할 수 있을 경우에는 bind 기능 구현은 필요 없고 결과 성공 메시지만 개방형 클라우드 플랫폼에 전송하면 됨

브로커가 개방형 클라우드 플랫폼으로부터 바인딩 요청을 수신 할 때 프로비저닝 된 자원을 활용하는데 필요한 정보를 반환

해당 정보는 credentials(자격증명)안에 제공

Applicatoin에 고유한 credentials(자격증명)을 발급하여 다른 Application에는 영향을 주어서는 안됨

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Request

Route

```
PUT /v2/service_instances/:instance_id/service_bindings/:binding_id
```

 Note : binding_id는 서비스 바인딩을 하기 위해 Cloud Controller에 의해 제공됨
binding_id는 향후 바인딩 해제 요청에 사용됨

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/  
:instance_id/service_bindings/:binding_id -d '{ "plan_id": "plan-guid-here",  
"service_id": "service-guid-here", "app_guid": "app-guid-here" }' -X PUT -H "X-  
Broker-API-Version: 2.4" -H "Content-Type: application/json"
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Request

Body

| RESPONSE FIELD | TYPE | DESCRIPTION |
|----------------|-------------|------------------------------|
| service_id* | string | 카탈로그 내의 서비스의 ID |
| plan_id* | string | 서비스 내의 plan ID |
| app_guid* | string | 서비스 바인드를 할 Application의 GUID |
| parameters | JSON object | JSON 형태의 피라미터 값을 제공 |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Response

Status Code

| STATUS CODE | DESCRIPTION |
|--------------------------|---|
| 201 Created | 바인딩 생성 |
| 200 OK | 서비스 바인딩이 이미 존재하고 요청 된 매개 변수가 기존의 바인딩과 동일한 경우 |
| 409 Conflict | 요청 된 바인딩이 이미 존재하는 경우 반환 에러 메시지 형태는 {}안에 description 필드를 사용 예) { "description": "Something went wrong. Please contact support at http://support.example.com." } |
| 422 Unprocessable Entity | 브로커가 요청 본문에 app_guid를 포함 할 것을 요구하는 경우 예) { "error": "RequiresApp", "descriptoin": "This service supports generation of credentials through binding an application only." } |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Response

Body

| RESPONSE FIELD | TYPE | DESCRIPTION |
|------------------|--------|--|
| credentials | object | Application이 서비스에 접근할 수 있는 credentials 정보. 해시 형태로 제공 |
| syslog_drain_url | string | 개방형 클라우드 플랫폼에 bound 된 Application에 대한 로그 URL |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Response

Binding Credentials 서비스 바인딩 경우 바인드 API 호출에 응답하여 사용자가 Application에서 사용 할 수 있는 인증 정보를 반환

개방형 클라우드 플랫폼 환경 변수 VCAP_SERVICES에 이러한 자격 증명을 제공

가능하면 credentials(자격증명) 필드 목록에서 사용하기를 권장

필요에 따라 추가 필드를 제공 할 수 있지만 제공되는 필드로 사용자의 요구 사항을 충족하는 경우 해당 필드를 사용

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Response

연결 문자열(connection string)을 지원하는 서비스를 제공하는 경우 적어도 uri 키를 제공해야 함

 Note : 별도의 자격 증명 필드를 제공 할 수 있음

Buildpacks 및 Application 라이브러리는 uri 키를 사용함

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Response

| STATUS CODE | DESCRIPTION |
|-------------|--|
| uri | dbtype 같은 연결 문자열(Connection string)의 형태는 아래와 같음 dbtype://username:password@hostname:port/name dbtype: mysql, postgres, mongodb, amqp, etc. |
| hostname | 서버 호스트의 FQDN(Full Qualified Domain Name) |
| port | 서버 호스트의 포트 번호 |
| name | 서비스 인스턴스 이름 (예: database name) |
| vhost | 메시징 서버의 가상 호스트의 이름 (AMQP 공급자에서 특정 이름 교체) |
| username | 서버 사용자 |
| password | 서버 사용자 비밀번호 |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Example VCAP_SERVICES 결과

```
VCAP_SERVICES=  
{  
  cleardb: [  
    {  
      name: "cleardb-1",  
      label: "cleardb",  
      plan: "spark",  
      credentials: {  
        name: "ad_c6f4446532610ab",  
        hostname: "us-cdbr-east-03.cleardb.com",  
        port: "3306",  
        username: "b5d435f40dd2b2",  
        password: "ebfc00ac",  
        uri: "mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab",  
        jdbcUrl: "jdbc:mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab"  
      }  
    }  
  ],  
}
```


PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Example VCAP_SERVICES 결과

```
cloudamqp: [  
  {  
    name: "cloudamqp-6",  
    label: "cloudamqp",  
    plan: "lemur",  
    credentials: {  
      uri: "amqp://ksvyjmiv:lwN6dCdZmeQD4O0ZPKpu1YOaLx1he8wo@lemur.cloudamqp.com/ksvyjmiv"  
    }  
  },  
  {  
    name: "cloudamqp-9dbc6",  
    label: "cloudamqp",  
    plan: "lemur",  
    credentials: {  
      uri: "amqp://vhuklnxa:9INFxpTuJsAdTts98vQldKHW3MojoyMyV@lemur.cloudamqp.com/vhuklnxa"  
    }  
  }  
],
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Example VCAP_SERVICES 결과

```
rediscloud: [  
  {  
    name: "rediscloud-1",  
    label: "rediscloud",  
    plan: "20mb",  
    credentials: {  
      port: "6379",  
      host: "pub-redis-6379.us-east-1-2.3.ec2.redislabs.com",  
      password: "1M5zd3QfWi9nUyya"  
    }  
  },  
],  
}
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Response

Application Log Streaming 개방형 클라우드 플랫폼은 서비스 인스턴스에 바인딩된 Application에 대한 로그를 스트리밍 함

서비스 인스턴스에 바인딩 된 모든 Application에 대한 로그는 해당 인스턴스로 스트리밍됨

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Response

동작하는 방법은 다음과 같음

- ✓ 브로커는 바인드에 대한 응답으로 syslog_drain_url에 대한 값을 반환
- ✓ Application이 재구동 할 때 VCAP_SERVICES 안의 syslog_drain_url의 key와 value를 갱신
- ✓ DEAs는 지속적으로 Loggregator에서 Application 로그를 스트리밍
- ✓ VCAP_SERVICES 안에 syslog_drain_url이 존재하면 DEA는 그 로그에 그 필드를 태그
- ✓ Loggregator는 값으로 지정된 위치에 이 key를 태그하여 로그 스트리밍 함

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

Bind Rest API 구현 - JAVA 방식

-- ServiceBindingRestController.java (Spring 프레임워크 사용)

```
@Controller
```

```
@RequestMapping("/v2/service_instances/{instanceId}/service_bindings/{bindingId}") class ServiceBindingRestController { @Autowired ServiceBindingService bindingService;
```

```
@RequestMapping(method = RequestMethod.PUT) @ResponseBody ServiceBinding update(@PathVariable String instanceId, @PathVariable String bindingId) { ServiceBinding binding = bindingService.findById(bindingId, instanceId); bindingService.save(binding); // 서비스 바인드 기능 구현 (개발 명세 내용 구현) return binding; }
```

```
}
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

서비스 별 Bind API 개발 명세

Bind 할 서비스 인스턴스가 존재 하는지 체크

Application 에 bind 할 정보를 생성하여 Application에 전달

이때 bind 정보는 랜덤하게 생성하고 base64 인코딩해서 보냄

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

서비스 별 Bind API 개발 명세

RDBMS

Mysql 경우

- ✓ 데이터 베이스에 접속할 사용자를 생성
CREATE USER #{username} IDENTIFIED BY #{password}
- ✓ provision 시 생성한 데이터 베이스에 생성한 사용자가 사용 가능하게 권한을 주고 plan에 해당하는 connection 수를 제공
GRANT ALL PRIVILEGES ON #{databasename}.* TO
#{username}@'%' WITH MAX_USER_CONNECTIONS
#{max_user_connections}
- ✓ 서버에 권한 테이블을 재배치
FLUSH PRIVILEGES

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

서비스 별 Bind API 개발 명세

RDBMS

Cubrid DB 경우

- ✓ 데이터 베이스에 접속할 사용자를 생성
CREATE USER #{username};



Note : Cubrid DB에서 권한 부여의 최소 단위는 테이블임
자신이 만든 테이블은 모든 접근을 허용

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

서비스 별 Bind API 개발 명세

대용량 저장소

GlusterFS 경우 새로운 Swift User를 생성

Method : PUT

Req URL : http(s)://[IP Address OR
HostName]:[PORT]/auth/v2/[AccountID]/[UserId]

Header : X-Auth-Admin-User: .super_admin X-Auth-Admin-Key:
swauthkey X-Auth-User-Key: [Password] X-Auth-User-Admin: [true
OR false]

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Bind API 가이드

서비스 별 Bind API 개발 명세

NoSQL DB

mongoDB 경우 데이터 베이스에 접속할 사용자를 생성하고
접근 role(read, Write)을 부여

```
use switched to db db.getSiblingDB("").runCommand( { createUser: "",  
pwd: "", roles: [ "readWrite" ] } )
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Unbind API 가이드

바인딩 서비스를 제공하지 않는 브로커는 Unbind API를 구현할 필요가 없음

브로커가 개방형 클라우드 플랫폼으로부터 unbind 요청을 받으면
바인드(bind)에서 만든 모든 자원(resource)을 삭제

삭제 되면 service에 접근 할 수 없음

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Unbind API 가이드

Request

Route

```
DELETE /v2/service_instances/:instance_id/service_bindings/:binding_id
```

Parameters

| QUERY-STRING FIELD | TYPE | DESCRIPTION |
|--------------------|--------|------------------|
| service_id* | string | 카탈로그에 있는 서비스 ID |
| plan_id* | string | 카탈로그에 있는 Plan ID |

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Unbind API 가이드

Request

cURL

```
$ curl 'http://username:password@broker-url/v2/service_instances/:instance_id/service_bindings/:binding_id?service_id=service-id-here&plan_id=plan-id-here' -X DELETE -H "X-Broker-API-Version: 2.4"
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Unbind API 가이드

Response

Status Code

| STATUS CODE | DESCRIPTION |
|-------------|-----------------------------------|
| 200 OK | 서비스 바인딩이 삭제. “{}” 메시지를 응답 |
| 410 Gone | 서비스 바인딩이 존재하지 않을 경우. “{}” 메시지를 응답 |

Body

모든 응답 bodies는 JSON Object ({}) 형식으로 함

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Unbind API 가이드

Unbind Rest API – JAVA 방식

-- ServiceBindingRestController.java (Spring 프레임워크 사용)

```
@Controller
```

```
@RequestMapping("/v2/service_instances/{instanceId}/service_bindings/{bindingId}") class ServiceBindingRestController { @Autowired ServiceBindingService bindingService;
```

```
@RequestMapping(method = RequestMethod.DELETE) @ResponseBody Map destroy(@PathVariable String instanceId, @PathVariable String bindingId) { ServiceBinding binding = bindingService.findById(bindingId, instanceId); bindingService.destroy(binding); // 서비스 unbind 기능 구현 (개발 명세 내용 구현) return [:]; } }
```

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Unbind API 가이드

서비스 별 Unbind API 개발 명세

Unbind 할 bind 인스턴스가 존재 하는지 체크

Application에 bind 된 정보를 삭제하고 결과를 Application에 전달

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Unbind API 가이드

서비스 별 Unbind API 개발 명세

RDBMS

Mysql 경우

- ✓ unbind 할 사용자가 존재 하는지 체크
SHOW GRANTS FOR #{username}}
- ✓ 생성된 사용자를 삭제 DROP USER #{username}
- ✓ 서버에 권한 테이블을 재배치 FLUSH PRIVILEGES

Cubrid DB 경우

- ✓ bind 시 생성한 사용자를 삭제 DROP USER #{username};

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Unbind API 가이드

서비스 별 Unbind API 개발 명세

대용량 저장소

GlusterFS 경우 Swift User를 삭제

Method : DELETE

Req URL : http(s)://[IP Address OR
HostName]/auth/v2/[AccountID]/[UserId]

Header : X-Auth-Admin-User: .super_admin X-Auth-Admin-Key:
swauthkey X-Auth-User-Key: [Password] X-Auth-User-Admin: [true
OR false]

PaaS-TA Service Package concept

» 서비스 예제를 통한 컨셉 설명

Unbind API 가이드

서비스 별 Unbind API 개발 명세

NoSQL DB

mongoDB 경우 bind 시 생성한 사용자를 삭제

```
use switched to db db.runCommand( { dropUser: "" } )
```



MEMO



M1. Cloud Basic

01. Cloud 이해

02. Cloud Model 및 특징 이해

M2. PaaS-TA 개발 실무

01. PaaS-TA 이해

02. PaaS-TA 개발 환경의 이해

03. PaaS-TA 개발도구 이해 및 실습

04. PaaS-TA 개발 검증 및 문제해결

M3. PaaS-TA 배포 및 운영

01. PaaS-TA 배포 및 관리

02. Service Package 배포 및 관리

PaaS-TA Service Package concept

Service Broker 예제 분석 ✓

Service Broker 등록 및 활성화

03. Custom Buildpack 개발

02 Service Broker 예제 분석

» mysql service broker를 통한 예제 분석

<https://github.com/PaaS-TA/OPENPAAS-SERVICE-JAVA-BROKER-MYSQL>

PaaS-TA / OPENPAAS-SERVICE-JAVA-BROKER-MYSQL

 Watch 7 Star 16 Fork 0

Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights

OPENPAAS-SERVICE-JAVA-BROKER-MYSQL

1 commit
1 branch
0 releases
1 contributor

Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download

| seongwookmoon init | Latest commit 93f23c9 on 5 Feb | |
|--------------------------|--------------------------------|---------------|
| gradle/wrapper | init | 11 months ago |
| libs | init | 11 months ago |
| src | init | 11 months ago |
| README.md | init | 11 months ago |
| build.gradle | init | 11 months ago |
| build.gradle_emma_backup | init | 11 months ago |
| gradlew | init | 11 months ago |
| gradlew.bat | init | 11 months ago |
| manifest.yml | init | 11 months ago |



MEMO



M1. Cloud Basic

01. Cloud 이해

02. Cloud Model 및 특징 이해

M2. PaaS-TA 개발 실무

01. PaaS-TA 이해

02. PaaS-TA 개발 환경의 이해

03. PaaS-TA 개발도구 이해 및 실습

04. PaaS-TA 개발 검증 및 문제해결

M3. PaaS-TA 배포 및 운영

01. PaaS-TA 배포 및 관리

02. Service Package 배포 및 관리

PaaS-TA Service Package concept

Service Broker 예제 분석

Service Broker 등록 및 활성화 ✓

03. Custom Buildpack 개발

03 Service Broker 등록 및 활성화

» 서비스 배포 및 등록

MySQL 서비스 브로커 등록

Mysql 서비스팩 배포가 완료 되었으면 Application에서 서비스 팩을 사용하기 위해
먼저 **MySQL 서비스 브로커를 등록**해 주어야 함

서비스 브로커 등록 시 **PaaS-TA에서 서비스브로커를 등록할 수 있는 사용자로 로그인**이 되어 있어야 함

03 Service Broker 등록 및 활성화

» 서비스 배포 및 등록

MySQL 서비스 브로커 등록

STEP. 1 서비스 브로커 목록을 확인한다.

```
$ cf service-brokers
```

```
inception@inception:~$ cf service-brokers
Getting service brokers as admin...

name                url
cubrid-service-broker http://10.30.60.22:8088
inception@inception:~$
```

03 Service Broker 등록 및 활성화

» 서비스 배포 및 등록

MySQL 서비스 브로커 등록

STEP. 2 MySQL 서비스 브로커를 등록한다.

```
$ cf create-service-broker {서비스팩 이름} {서비스팩 사용자ID} {서비스팩 사용자비밀번호}
http://{서비스팩 URL(IP)}
```

- **서비스팩 이름** : 서비스 팩 관리를 위해 PaaS-TA에서 보여지는 명칭이다. 서비스 Marketplace에서는 각각의 API 서비스 명이 보여지니 여기서 명칭은 서비스팩 리스트의 명칭이다.
- **서비스팩 사용자ID / 비밀번호** : 서비스팩에 접근할 수 있는 사용자 ID입니다. 서비스팩도 하나의 API 서버이기 때문에 아무나 접근을 허용할 수 없어 접근이 가능한 ID/비밀번호를 입력한다.
- **서비스팩 URL** : 서비스팩이 제공하는 API를 사용할 수 있는 URL을 입력한다.

03 Service Broker 등록 및 활성화

» 서비스 배포 및 등록

MySQL 서비스 브로커 등록

STEP. 2 MySQL 서비스 브로커를 등록한다.

```
$ cf create-service-broker mysql-service-broker admin cloudfoundry  
http://10.0.0.95:8080
```

```
inception@inception:~$ cf create-service-broker mysql-service-broker admin cloudfoundry http://10.30.40.195:8080  
Creating service broker mysql-service-broker as admin...  
OK  
inception@inception:~$ █
```

03 Service Broker 등록 및 활성화

» 서비스 배포 및 등록

MySQL 서비스 브로커 등록

STEP. 3 등록된 MySQL 서비스 브로커를 확인한다.

```
$ cf service-brokers
```

```
inception@inception:~$ cf service-brokers
Getting service brokers as admin...

name                url
cubrid-service-broker http://10.30.60.22:8088
mysql-service-broker http://10.30.40.195:8080
inception@inception:~$
```

03 Service Broker 등록 및 활성화

» 서비스 배포 및 등록

MySQL 서비스 브로커 등록

STEP. 4 접근 가능한 서비스 목록을 확인한다.

```
$ cf service-access
```

```
inception@inception:~$ cf service-access
Getting service access as admin...
broker: cubrid-service-broker
  service  plan  access  orgs
  CubridDB utf8  all
  CubridDB euckr all

broker: mysql-service-broker
  service  plan  access  orgs
  Mysql-DB Mysql-Plan1-10con none
  Mysql-DB Mysql-Plan2-100con none

inception@inception:~$
```

서비스 브로커 생성시 디폴트로 접근을 허용하지 않는다.

03 Service Broker 등록 및 활성화

» 서비스 배포 및 등록

MySQL 서비스 브로커 등록

STEP. 5

특정 조직에 해당 서비스 접근 허용을 할당하고 접근 서비스 목록을 다시 확인한다.
(전체 조직)

```
$ cf enable-service-access Mysql-DB
```

```
$ cf service-access
```

```
inception@inception:~$ cf enable-service-access Mysql-DB
Enabling access to all plans of service Mysql-DB for all orgs as admin...
OK
inception@inception:~$ cf service-access
Getting service access as admin...
broker: cubrid-service-broker
  service  plan  access  orgs
  CubridDB utf8  all
  CubridDB euckr all

broker: mysql-service-broker
  service  plan  access  orgs
  Mysql-DB Mysql-Plan1-10con all
  Mysql-DB Mysql-Plan2-100con all

inception@inception:~$
```

실습

Service Broker 등록 및 활성화: 서비스 배포 및 등록

실습 사이트

배포한 샘플 서비스

실습 소요시간

약 15분

진행 방법

배포된 서비스의 api를 확인하여 ① 서비스로 등록
② 활성화 ③ marketplace에서 확인한다.

필요 도구

노트북, cf CLI



MEMO





MEMO



A top-down view of a wooden desk with a laptop, glasses, a coffee cup, and a small plant. The text "03. Custom Buildpack 개발" is overlaid on the left side of the image.

03. Custom Buildpack 개발

M1. Cloud Basic

01. Cloud 이해

02. Cloud Model 및 특징 이해

M2. PaaS-TA 개발 실무

01. PaaS-TA 이해

02. PaaS-TA 개발 환경의 이해

03. PaaS-TA 개발도구 이해 및 실습

04. PaaS-TA 개발 검증 및 문제해결

M3. PaaS-TA 배포 및 운영

01. PaaS-TA 배포 및 관리

02. Service Package 배포 및 관리

03. Custom Buildpack 개발

Buildpack 개발 가이드 ✓

Creating Custom Buildpacks

» Buildpack

빌드팩은 애플리케이션 구동에 필요한 환경(런타임, 컨테이너, 프레임워크 등)을 조립하고 드롭릿을 구성하는 스크립트의 모음

빌드팩은 개방형 클라우드 플랫폼의 요청에 의해 동작

빌드팩 개발자는 개방형 클라우드 플랫폼과의 연동을 위해 3개의 스크립트(검출, 컴파일, 릴리즈)를 필수로 구현

API는 쉘스크립트로 작성하며 실 동작부분의 개발언어에 대한 제약사항은 없음

검출
(Detect)

컴파일
(Compile)

릴리즈
(Release)

» 필수 기능

검출 (Detect)

배포되는 애플리케이션의 런타임 환경 구성 방법을 빌드팩이
아는지 여부를 확인하는 기능

검출 스크립트에는 빌드팩의 적용 가능성(스테이지 할 수 있는지)을 검사하는 내용을
작성

빌드팩의 적용 가능성은 일반적으로 특정 파일의 존재 여부로 판단

Buildpack 개발 가이드

» 필수 기능

검출 (Detect)

동작 설명

```
$ bin/detect <BUILD_DIR>
```

검출 스크립트를 호출할 때 전달되는 인자 값은 **빌드 디렉터리(BUILD_DIR)**

빌드 디렉터리는 애플리케이션 파일들이 위치한 디렉터리로써 **스크립트**는 **파일들을 검사하여 빌드팩 적용 여부를 결정**

배포하는 애플리케이션이 빌드팩에서 지원하는 유형이라면 '0' 종료 값을 리턴

스크립트가 '0'을 리턴하면 검출된 환경의 이름을 사용자에게 출력

» 필수 기능

검출 (Detect)

작성 예

검출 스크립트에는 적용 여부를 판단할 수 있는 기준이 제시되어야 함

루비 애플리케이션을 검출하는 예

Gemfile 존재여부를 기준으로 하여 Ruby 환경을 검출하고 사용자에게 이를 출력

```
\#!/usr/bin/env ruby
gemfile\_path **=** File.**join** [ARGV\[0\], "**Gemfile**"
if File.exist?(gemfile\_path)
  puts "Ruby"
  exit 0
else
  puts "no"
  exit 1
end
```


» 필수 기능

컴파일 (Compile)

컴파일은 실질적으로 드롭릿을 빌드하는 빌드팩의 핵심기능

컴파일 스크립트에는 애플리케이션 구동 시 필요한 바이너리들을 다운로드 및 설치하고, 이를 드롭릿 파일시스템에 배치시키는 내용을 작성

애플리케이션 구동에 필요한 바이너리들의 예로는 런타임(JRE, Ruby, PHP, Node 등), 웹컨테이너(Tomcat, JBoss, Webrick 등)가 있음

Buildpack 개발 가이드

» 필수 기능

컴파일 (Compile)

동작 설명

```
$ bin/compile <BUILD_DIR> <CACHE_DIR>
```

컴파일 스크립트를 호출할 때 전달되는 인자 값 2개는 **빌드 디렉터리(BUILD_DIR)**와 **캐시 디렉터리(CACHE_DIR)**

캐시 디렉터리는 빌드팩 컴파일 프로세스 동안 다운로드 하는 종속성들을 임시로 저장하는데 사용할 수 있음

스크립트 실행 중 사용자에게 컴파일 과정을 출력

Buildpack 개발 가이드

» 필수 기능

컴파일 (Compile)

동작 설명

컴파일 스크립트는 빌드팩과 애플리케이션이 구동 시 필요로 하는 환경에 따라 다양하게 작성됨

컴파일 스크립트의 간단한 작성 예

해당 스크립트는 Ruby 애플리케이션을 구동시키기 위한 환경을 구성

BUILD_PATH에 Ruby 인터프리터를 우선 설치하고, 환경설정이나 필요한 바이너리들을 설치하는 등의 내용을 작성

» 필수 기능

컴파일 (Compile)

작성 예

컴파일 스크립트의 간단한 작성 예

```

#!/usr/bin/env ruby
#sync output
\$.stdout.**sync** **=* **true**
build\_path **=* ARGV\[0\]
cache\_path **=* ARGV\[1\]
def compile
  instrument 'ruby.compile' do
    Dir.chdir(build\_path)
    install\_ruby
    \#...중략
    setup\_language\_pack\_environment
  end
end

```

```

install\_binaries
\#...중략
end
end
def install\_ruby
  puts "Installing Ruby"
  \# !!! build tasks go here !!!
  \# download ruby to cache\_path
  \# install ruby
end

```

» 필수 기능

릴리즈 (Release)

릴리즈는 애플리케이션 실행방법에 대한 정보를 플랫폼에
응답해주는 기능을 함

릴리즈 스크립트에는 플랫폼에 응답할 정보를 생성하는 내용을 작성

Buildpack 개발 가이드

» 필수 기능

릴리즈 (Release)

동작 설명

```
$ bin/release <BUILD_DIR>
```

릴리즈 스크립트를 호출할 때 전달되는 인자 값은 **빌드 디렉터리(BUILD_DIR)**

실행방법 정보는 반드시 아래와 같은 포맷의 YAML로 제공해야 함

```
config\_vars:
```

```
  name: value
```

옵션으로 제공 할 환경변수들을 작성하며, 해당 환경변수들은 애플리케이션이 실행되는 환경에서 정의될 변수들을 의미

```
default\_process\_types:
```

```
  web: commandLine
```

실행될 애플리케이션의 유형과 실행시킬 때 사용할 명령행을 작성. 현재는 웹 애플리케이션 유형만을 지원

Buildpack 개발 가이드

» 필수 기능

릴리즈 (Release)

작성 예

릴리즈 스크립트에 작성된 응답 정보의 예

해당 응답 정보는 `Rack[^6]` 애플리케이션에서 필요한 환경 변수와 실행하는 방법을 포함하고 있으며 개방형 클라우드 플랫폼에 전달됨

```
config\_vars:  
  RACK\_ENV: production  
default\_process\_types:  
  web: bundle exec rackup config.re -p $PORT
```

Buildpack 개발 가이드

» 부가기능

빌드팩은 컴파일 시 애플리케이션 구동을 위해 필요한 종속성들(바이너리, 라이브러리 등)을 설치

기본적으로 빌드팩은 이러한 종속성을 네트워크를 통해 다운로드하기 때문에 네트워크가 단절된 환경에서의 사용에 제약이 있음

따라서 빌드팩 개발 시 개방형 클라우드 플랫폼이 설치되는 환경에 대해서도 고려할 필요가 있음

» 부가기능

패키지 (Package)

패키지는 빌드팩을 하나의 압축파일로 만드는 기능 혹은 압축파일 자체를 의미

빌드팩 패키지의 목적

- ✓ 시스템 빌드팩으로 등록 가능한 형태로 만드는 것
- ✓ 네트워크가 단절된 환경에서 빌드팩 컴파일을 지원하는 것

개발자는 이를 위해 빌드팩에 **패키징(Packaging)** 및 **버저닝(Versioning)** 패키지 기능들을 추가로 구현 할 수 있음

Buildpack 개발 가이드

» 부가기능

패키지 (Package)

패키징(Packaging) 기능

- ✓ 온라인 패키징 컴파일 시 네트워크에 접속하여 종속성들을 다운로드
- ✓ 오프라인 패키징 패키지 시 지원하는 종속성들을 모두 패키지 안에 포함

버저닝(Versioning) 기능

- ✓ 빌드팩 패키징 시 생성 할 패키지 이름에 버전 정보를 추가하는 기능

패키지를 지원하기 위해서는 **패키징과 버저닝 기능을 직접 구현**하거나 PaaS-TA에서 제공하는 **Buildpack packager** 애플리케이션을 사용하는 방법이 있음

Buildpack 개발 가이드

» 부가기능

패키지 (Package)

1. 직접 구현하는 예

PaaS-TA의 JAVA-BUILDPACK은 소스에 포함된 Package모듈을 통해 패키지 기능을 지원

패키지는 `bundle exec rake[^7]` 명령어를 통해 수행됨

인자 값으로 `OFFLINE=true`를 사용하면 **오프라인 패키지를 생성**하고
`VERSION=<VERSION>` 인자 값을 사용하면 **패키지 이름에 버전 정보를 추가**

```
\$ bundle install
```

```
\$ bundle exec rake package OFFLINE=true VERSION=2.1
```

```
\$ Creating build/java-buildpack-offline-2.1.zip
```

» 부가기능

패키지 (Package)

1. 직접 구현하는 예

`bundle exec rake` 명령어는 `Rakefile[^8]`에 정의된 Package모듈을 순서대로 실행하며 실행 순서는 다음과 같음

- 1 오프라인 패키지를 만드는 경우 우선 설정파일들에 정의된 의존성들을 모두 다운로드
- 2 이후 `cache.yml`에 있는 `remote_download값`을 `disables`로 만듦
- 3 패키지.zip 파일을 생성

Buildpack 개발 가이드

» 부가기능

패키지 (Package)

1. 직접 구현하는 예

아래는 Rakefile과 Package 모듈의 일부를 보여줌

```

\#Rakefile
require 'raketlib/dependency\_cache\_task'
require 'raketlib/stage\_buildpack\_task'
require 'raketlib/package\_task'
Package::**DependencyCacheTask**.new
Package::**StageBuildpackTask**.new(Dir\['bin/\*\*/\*', 'config/\*\*/\*', 'lib/\*\*/\*', 'resources/\*\*/\*\*\'].reject { |f|
File.directory? f })
Package::**PackageTask**.new
----
\#Package Module
require 'java\_buildpack/buildpack\_version'
module Package
  def self.offline
    '-offline' if BUILDPACK\_VERSION.offline
  end
  def self.version
    BUILDPACK\_VERSION.version || 'unknown'
  end
end

```

Buildpack 개발 가이드

» 부가기능

패키지 (Package)

1. 직접 구현하는 예

```

BUILD\_DIR = 'build'.freeze
STAGING\_DIR = "#{BUILD\_DIR}/staging".freeze
BUILDPACK\_VERSION = JavaBuildpack::BuildpackVersion.new(false).freeze
PACKAGE\_NAME = "#{BUILD\_DIR}/java-buildpack\#{offline}-\#{version}.zip".freeze
class DependencyCacheTask** < Rake::TaskLib
  def initialize
    return unless BUILDPACK\_VERSION.offline
    @cache = cache
    uris(configurations).each { .. \[cache\_task(uri)\] }
  end
  def cache
    JavaBuildpack::Util::Cache::DownloadCache.new(..).freeze
  end
  def cache\_task(uri)
    task uri do |t|
      rake\_output\_message "Caching \#{t.name}"
      cache.get(t.name) {}
    end
    uri
  end
end
end

```

» 부가기능

패키지 (Package)

1. 직접 구현하는 예

```
class StageBuildpackTask** < Rake::TaskLib
  def initialize(source\_files)
    disable\_remote\_downloads\_task if BUILDPACK\_VERSION.offline
  end
  def disable\_remote\_downloads\_task
    file "\#{STAGING\_DIR}/config/cache.yml" do |t|
      content = File.open(t.source, 'r') { |f| f.read.gsub(/enabled/, 'disabled') }
      File.open(t.name, 'w') { |f| f.write content }
    end
  end
end
```

» 부가기능

패키지 (Package)

1. 직접 구현하는 예

```

class PackageTask** < Rake::TaskLib
  def initialize
    desc 'Create packaged buildpack'
    task :package => [:PACKAGE_NAME]
    multitask PACKAGE_NAME => [:BUILD_DIR, :STAGING_DIR] do |t|
      rake_output_message "Creating #{t.name}"
      Zip::File.open(t.name, Zip::File::CREATE) do |zipfile|
        Dir[File.join(STAGING_DIR, '**', '**')].each do |file|
          zipfile.add(file.sub("#{STAGING_DIR}/", ""), file)
        end
      end
    end
  end
end
end
end
end
end

```


» 부가기능

패키지 (Package)

2. Buildpack packager 사용 예

Buildpack packager는 PaaS-TA에서 제공하는 빌드팩 패키지 도구

buildpack-packager는 애플리케이션의 종속성이 아니라 **빌드팩의 종속성을 캐시하는 것**

Buildpack 개발 가이드

» 부가기능

패키지 (Package)

2. Buildpack packager 사용 예

아래는 buildpack-packager를 사용하여 RUBY 빌드팩을 패키지 하는 방법

ruby-buildpack git에서 submodules(compile-extensions)를 fetch

```
\$ git submodule update -init
```

최종 빌드팩 종속성들을 다운로드

```
\$ BUNDLE\_GEMFILE=cf.Gemfile bundle
```

buildpack packager를 실행

```
\$ BUNDLE\_GEMFILE=cf.Gemfile bundle exec buildpack-packager \[uncached | cached \]
```

Buildpack 개발 가이드

» 부가기능

패키지 (Package)

2. Buildpack packager 사용 예

Buildpack packager는 실행 시 **manifest.yml** 파일을 확인하므로 해당 파일을 **우선 작성**해야 함

packager는 manifest.yml의 `exclude_files`에 명시된 파일들을 제외한 빌드팩 디렉터리의 모든 것을 패키지 파일에 추가

cached 옵션으로 실행한 경우 manifest.yml에 명시된 종속성들을 다운로드하고 패키지 파일에 포함

즉 cached 옵션은 오프라인 패키지와 동일한 역할을 함

» 부가기능

패키지 (Package)

2. Buildpack packager 사용 예

manifest.yml 파일의 작성 예

```

\#manifest.yml
---
language: ruby
url\_to\_dependency\_map:
- match: bundler-(\d+\.\d+\.\d+)
  name: bundler
  version: \d1
- match: ruby-(\d+\.\d+\.\d+)
  name: ruby
  version: \d1

```

language

- ✓ 패키지 파일에 사용될 이름을 작성

url_to_dependency_map

- ✓ 해당 종속성의 이름과 버전을 추출하고 맵핑 시키기 위한 정규 표현식의 목록을 작성

» 부가기능

패키지 (Package)

2. Buildpack packager 사용 예

manifest.yml 파일의 작성 예

```
dependencies:
- name: bundler
  version: 1.7.12
  uri: https://pivotal-
buildpacks.s3.amazonaws.com/ruby/binaries/lucid64/bundler-
1.7.12.tgz
  md5: ab7ebd9c9e945e3ea91c8dd0c6aa3562
  cf\_stacks:
  - lucid64
- cflinuxfs2
- name: ruby
  version: 2.1.4
  uri: https://pivotal-
buildpacks.s3.amazonaws.com/ruby/binaries/lucid64/ruby-
2.1.4.tgz
  md5: 72b4d193a11766e2a4c45c1fed65754c
  cf\_stacks:
  - lucid64
exclude\_files:
- .gitignore
- private.key
```

dependencies

- ✓ 컴파일 시 다운로드할 리소스들에 대해 name, version, uri, md5, cf_stacks[^9] 정보를 명시
- ✓ packager는 명시된 리소스들을 dependencies 폴더에 다운로드하고 설치
- ✓ dependencies 폴더는 패키지 파일에 포함됨

» 부가기능

저장소 (Repository)

저장소는 빌드팩 컴파일 시 다운로드 하는 다양한 종속성들이 존재하는 공간

저장소는 개방형 클라우드 플랫폼이 설치된 환경에 따라 외부 또는 내부 네트워크 위치에 구성할 수 있음

빌드팩은 저장소의 위치를 설정 및 변경할 수 있는 방법을 제공하며 이러한 저장소 설정관련 부분은 빌드팩 소스마다 다름

단, Build-packager를 사용하는 빌드팩(e.g. ruby)의 경우 앞서 패키지에서 설명한 manifest.yml 파일의 dependencies:uri 항목에 다운로드 할 저장소 위치 및 라이브러리 정보를 작성함



MEMO



M1. Cloud Basic

01. Cloud 이해

02. Cloud Model 및 특징 이해

M2. PaaS-TA 개발 실무

01. PaaS-TA 이해

02. PaaS-TA 개발 환경의 이해

03. PaaS-TA 개발도구 이해 및 실습

04. PaaS-TA 개발 검증 및 문제해결

M3. PaaS-TA 배포 및 운영

01. PaaS-TA 배포 및 관리

02. Service Package 배포 및 관리

03. Custom Buildpack 개발

Buildpack 개발 가이드

Creating Custom Buildpacks ✓

02 Creating Custom Buildpacks

» Package Custom Buildpacks

Use the Buildpack Packager

- 1 buildpack-packager를 설치했는지 확인
- 2 빌드팩에 manifest.yml을 생성
- 3 cached 모드에서 packager 실행
`$ buildpack-packager build -cached -any-stack`

packager는 빌드팩 디렉토리의 모든 것을 zip 파일로 추가

cached 모드에서 packager는 manifest에 설명된 대로 **종속성을 다운로드하고 추가**


02 Creating Custom Buildpacks

» Package Custom Buildpacks

Use and Share the Packaged Buildpack

buildpack-packager를 사용하여 빌드팩을 패키징한 후에는 결과 .zip 파일을 로컬로 사용하거나 CLI에서 액세스할 수 있는 모든 네트워크 위치에 업로드하여 다른 사람과 공유할 수 있음

그런 다음 사용자는 앱을 푸시할 때 **-b 옵션**을 사용하여 빌드팩을 지정할 수 있음

 Note : 오프라인 빌드 팩 패키지는 배포 라이선스 또는 수출 제어 수단을 필요로 하는 독점적 의존성을 포함할 수 있음

02 Creating Custom Buildpacks

» Package Custom Buildpacks

Use and Share the Packaged Buildpack

`cf create-buildpack` 명령을 사용하여 buildpack을 PaaS-TA 구현으로 업로드하여 `-b` 플래그 없이 액세스할 수 있도록 할 수도 있음

example

```
$ cf create-buildpack BUILDPACK PATH POSITION [--enable|--disable]
```

02 Creating Custom Buildpacks

» Package Custom Buildpacks

Specify a Default Version

buildpack-packager 버전 2.3.0을 기준으로 **default_versions** 개체를 **manifest.yml** 파일에 추가하여 **종속성의 기본 버전을 지정**할 수 있음

default_versions 개체에는 **이름**과 **버전**이라는 두 가지 속성이 있음

```
default_versions:  
- name: go  
  version: 1.6.3  
- name: other-dependency  
  version: 1.1.1
```

02 Creating Custom Buildpacks

» Package Custom Buildpacks

Specify a Default Version

기본 버전을 지정하는 방법

1 default_version 개체를 매니페스트에 추가

2 `default_version_for` 스크립트를 컴파일 확장 저장소에서 실행하여 manifest.yml 및 종속성 이름을 인수로 전달

example

```
$ ./compile-extensions/bin/default_version_for manifest.yml go 1.6.3
```

02 Creating Custom Buildpacks

» Deploy Apps with a Custom Buildpack

Deploy Apps with a Custom Buildpack

custom 빌드 팩을 만들어 공용 Git 저장소에 푸시하면 앱을 푸시할 때 cf CLI를 통해 git URL을 전달할 수 있음

GitHub에 푸시된 빌드 팩의 경우

```
$ cf push my-new-app -b git://github.com/johndoe/my-buildpack.git
```

https 및 사용자 이름/암호 인증과 함께 개인 Git 저장소를 사용할 수 있음

```
$ cf push my-new-app -b git://github.com/johndoe/my-buildpack.git
```

02 Creating Custom Buildpacks

» Deploy Apps with a Custom Buildpack

Deploy Apps with a Custom Buildpack

기본적으로 PaaS-TA 애플리케이션 런타임은 **빌드 팩의 git 저장소의 기본 분기를 사용함**

git url을 사용하여 다른 분기를 지정할 수 있음

```
$ cf push my-new-app -b https://github.com/johndoe/my-buildpack.git#my-branch-name
```

git 저장소에서 태그를 사용할 수 있음

```
$ cf push my-new-app -b https://github.com/johndoe/my-buildpack#v1.4.2
```

02 Creating Custom Buildpacks

» Deploy Apps with a Custom Buildpack

Deploy Apps with a Custom Buildpack

그런 다음 이 앱을 PaaS-TA 애플리케이션 런타임에 배포하고, 빌드 팩을 저장소에서 복제하여 앱에 적용함

`cf push -b`를 사용하여 buildpack을 지정하면 탐지 단계를 건너뛰고 결과적으로 buildpack 탐지 스크립트는 실행되지 않음

／ 핵심정리 ／

- ✓ Service는 PaaS-TA 주요 구성요소 이외에 Application을 운용하는데 도움을 주는 요소이다.
- ✓ Service를 등록하여 PaaS-TA service로 활용 할 수 있으며 이를 위해 서비스 브로커를 개발해야한다.
- ✓ Application만의 특수한 설정이 필요할 경우 Application에 맞는 custom buildpack을 개발하여 상용할 수 있다.

M1. Cloud Basic

01. Cloud 이해

기본속성
Cloud 발전과정
Cloud 기반 사업 추진 기업
트렌드

02. Cloud Model 및 특징 이해

Cloud Model
IaaS
PaaS
SaaS

M2. PaaS-TA 개발 실무

01. PaaS-TA 이해

PaaS-TA 소개
PaaS-TA 아키텍처의 이해
PaaS-TA 주요 구성요소
PaaS-TA 운영 및 편의 도구
PaaS-TA 상황별 구성요소 기능

02. PaaS-TA 개발 환경의 이해

애플리케이션 환경 이해
서비스 이해
개발 환경 이해

03. PaaS-TA 개발도구 이해 및 실습

Portal
CLI
IDE
SCM
배포 Pipeline

04. PaaS-TA 개발 검증 및 문제해결

Cloud Native Application 개념
12 Factors
Micro Service Architecture

M3. PaaS-TA 배포 및 운영

01. PaaS-TA 배포 및 관리

PaaS-TA 배포를 위한 기본 구성
BOSH 개념
Director 설치
BOSH를 통한 PaaS-TA 구축

02. Service Package 배포 및 관리

PaaS-TA Service Package concept
Service Broker 예제 분석
Service Broker 등록 및 활성화

03. Custom Buildpack 개발

Buildpack 개발 가이드
Creating Custom Buildpacks



MEMO





MEMO

