API 로 오픈스택 이해하기

조성수 2017.12.21 11:54

목차

참고 도서

시스템 구축 절차 표준화 - 물리적 환경

기존 시스템을 구축할 때는 물리 장비 도입 준비부터 시작

제조사/장비 마다 다른 성능,기능 그리고 메뉴얼

신규 장비 도입시 여러가지 과정을 거치다 보면, 수 개월이 소요될 수 있음

시스템 구축 절차 표준화 - 클라우드

물리적인 리소스가 추상화된 형태로 제공

가상화 자원에 대해 표준화된 사용법을 제공

인프라 변화에도 시스템 구축 과정에 표준화를 유지할 수 있음

클라우드에서의 API

클라우드에서는 추상화된 논리적인 컴포넌트들을 조합함으로써 표준화된 절차에 따라 시스템 구축이 가능

컴포넌트를 제어하는 방법

- 웹 콘솔
- 명령어를 이용한 제어
- 직접 개발한 프로그램
- 자동화 툴

위 제어방법이 모두 API를 통해 이루어짐

오픈스택의 API 형태

https://{component}/{version}/{tenant_id}/{resource}

예) http://127.0.0.1:8774/v2/abcd-efgh/images

예) http://127.0.0.1:8774/v2/abcd-efgh/servers/dasb8z-0zx89c-zx98c7

- API를 이용하기 위해서는 반드시 HTTP 헤더에 `X-Auth-Token` 이 있어야하며 이 필드에는 **keystone**으로부터 발급받은 토큰을 넣어야합니다.

인증과 보안

테넌트(Tenant)

클라우드의 최상위 개념 `테넌트`

오픈스택의 Keystone 에서는 `프로젝트`

AWS에서는 `Account`

테넌트 간은 완전히 격리된 환경

사용자

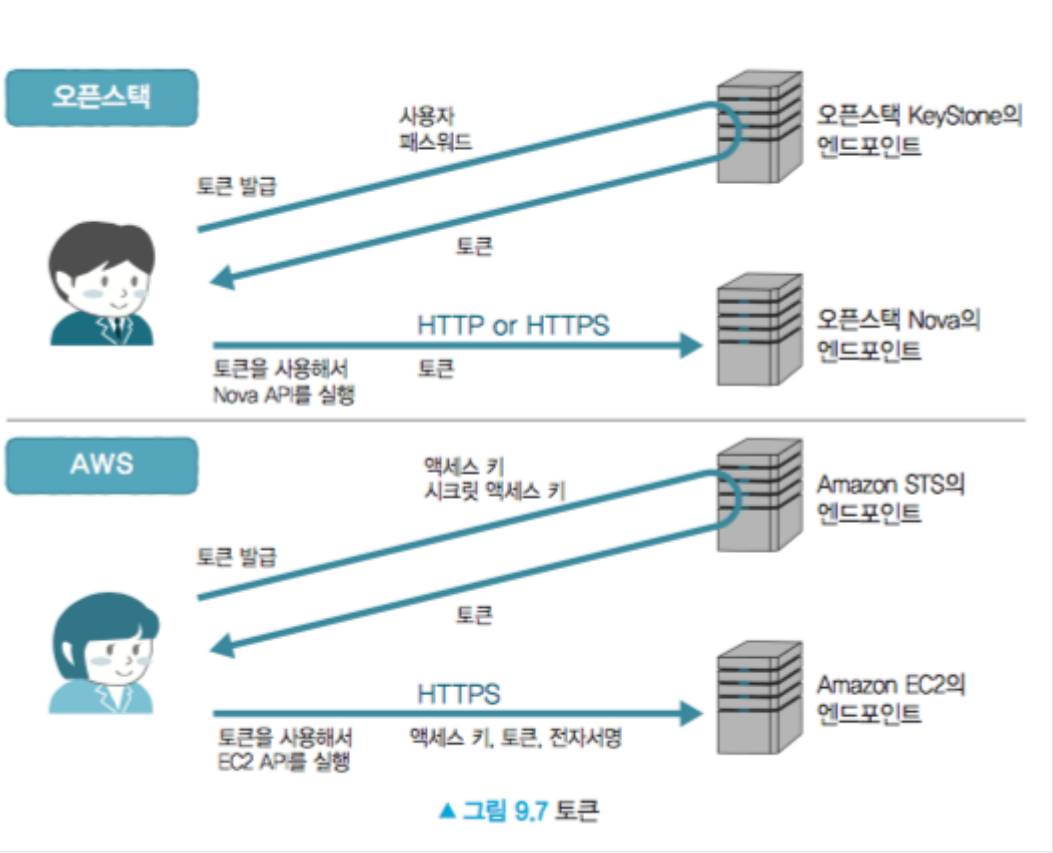
API 를 실행하는 주체

사용자도 리소스에 해당되기 때문에 API로 추가 삭제 가능

오픈스택에서는 롤이라는 개념이 존재.

- Admin 과 Member 로 구분되며, Admin 은 관리자 권한을 가짐

인증 키와 토큰



▲ 그림 9.7 토큰

- 출처: 그림으로 배우는 클라우드 인프라와 API의 구조, P327

---

keystone 인증 API 예시

```
curl -s -X POST http://127.0.0.1:5000/v2.0/tokens \
  -H "Content-Type: application/json" \
  -d '{"auth": {"tenantName": "'"admin"'", "passwordCredentials":
    {"username": "'"admin"'", "password": "'"password"'"}}}' | python -m json.tool
```

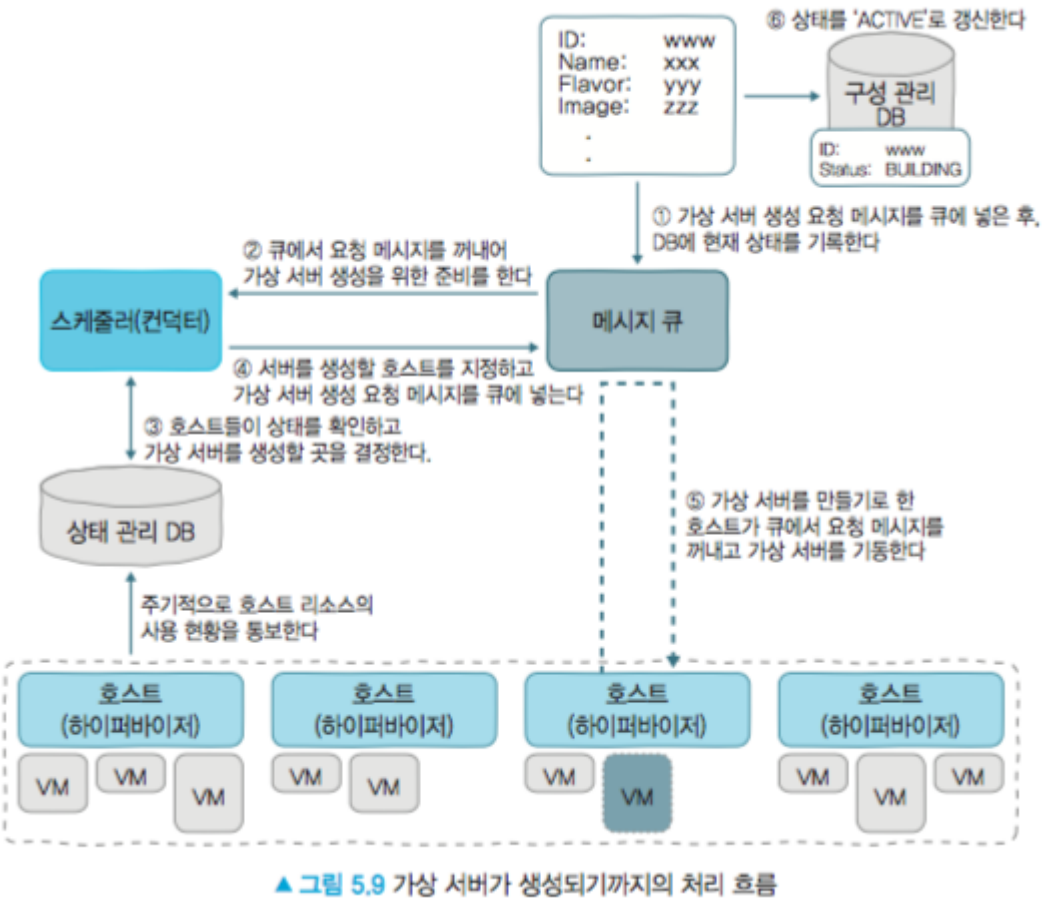
Response


```
{
    "access": {
        "metadata": {
            "is_admin": 0,
            "roles": [
                "09222e4b930c4a05bb91db56adebfedf",
                "e4e7c0eaa707473bb6d09311d74cef54",
                "9fe2ff9ee4384b1894a90878d3e92bab"
            ]
        },
        "serviceCatalog": [
            {
                "endpoints": [
                    {
                        "adminURL": "http://127.0.0.1:8774/v2/4601230eb33f45e0a1855839649f8afc",
                        "id": "0aa17302f4f44554ba8fd266773c6334",
                        "internalURL": "http://127.0.0.1:8774/v2/4601230eb33f45e0a1855839649f8afc",
                        "publicURL": "http://127.0.0.1:8774/v2/4601230eb33f45e0a1855839649f8afc",
                        "region": "RegionOne"
                    }
                ],
                "endpoints_links": [],
```

```
                "name": "nova",
                "type": "compute"
            },
            . . .
        ],
        "token": {
            "audit_ids": [
                "Z0Lb0WSETL6cAvD4KjFfPw"
            ],
            "expires": "2015-02-09T10:23:44Z",
            "id": "c5eb66493c0445e2aef0b2d65c79333f",
            "issued_at": "2015-02-09T09:23:44.394753",
            "tenant": {
                "description": null,
                "enabled": true,
                "id": "4601230eb33f45e0a1855839649f8afc",
                "name": "admin",
                "parent_id": null
            }
        },
        "user": {
            "id": "78876ccabfba4d188d6098951ef420c8",
            "name": "admin",
            "roles": [
                {
                    "name": "admin"
                },
                {
                    "name": "heat_stack_owner"
                },
                {
                    "name": "_member_"
                }
            ],
            "roles_links": [],
            "username": "admin"
        }
    }
}
```


가상 서버 생성 API

가상 서버가 생성되기까지의 처리 흐름



▲ 그림 5.9 가상 서버가 생성되기까지의 처리 흐름

- 출처: 그림으로 배우는 클라우드 인프라와 API의 구조, P187

Step1. Listing Flavors

가상 서버의 Flavor (스펙) 선택

Request

```
curl -s -H "X-Auth-Token:[YOUR_AUTH_TOKEN]" http://127.0.0.1:8774/v2/[YOUR_TENANT_ID]/flavors | python -m json.too
l
```

Response


```
{
    "flavors" : [
        {
            "id" :  "1",
            "links" : [
                {
                    "href" :  "http://127.0.0.1:8774/v2/[YOUR_TENANT_ID]/flavors/1",
                    "rel" :  "self"
                },
                {
                    "href" :  "http://127.0.0.1:8774/[YOUR_TENANT_ID]/flavors/1",
                    "rel" :  "bookmark"
                }
            ],
            "name" :  "m1.tiny"
        }
```

```
    ]
}
```


Step2. Listing Images

가상 서버의 이미지 선택 (OS 선택)

Request

```
curl -s -H "X-Auth-Token:[YOUR_AUTH_TOKEN]" http://127.0.0.1:8774/v2/[YOUR_TENANT_ID]/images | python -m json.tool
```

Response


```
{
    "images" : [
        {
            "id" :  "5e9cecc5-ae7a-4f7e-b488-ed22daa0e2ab",
            "links" : [
                {
                    "href" :
                     "http://127.0.0.1:8774/v2/[YOUR_TENANT_ID]/images/5e9cecc5-ae7a-4f7e-b488-ed22daa0e2ab",
                    "rel" :  "self"
                },
                {
                    "href" :
                     "http://127.0.0.1:8774/[YOUR_TENANT_ID]/images/5e9cecc5-ae7a-4f7e-b488-ed22daa0e2ab",
                    "rel" :  "bookmark"
                },
                {
                    "href" :
                     "http://127.0.0.1:9292/[YOUR_TENANT_ID]/images/5e9cecc5-ae7a-4f7e-b488-ed22daa0e2ab",
                    "rel" :  "alternate",
                    "type" :  "application/vnd.openstack.image"
                }
            ],
            "name" :  "cirros"
        }
    ]
}
```


Step3. SSH 키 생성

가상 서버에 접속할 때 필요한 SSH 키 생성

키 생성

```
ssh-keygen -t rsa -f cloud.key
```

SSH 키 업로드

```
curl -X POST -H "X-Auth-Token:[YOUR_AUTH_TOKEN]" -H "Content-Type: application/json" -d  '
{
 "keypair": {
   "name": "MyFirstKeyPair",
   "public_key": "[YOUR_PUBLIC_KEY]"
 }
}' http://127.0.0.1:8774/v2/[YOUR_TENANT_ID]/os-keypairs | python -m json.tool
```

Step4. 가상 서버 생성

Request

```
curl -X POST -H "X-Auth-Token:$1" -H "Content-Type: application/json" -d  '
{
 "server": {
   "name": "MyFirstApiVm",
   "imageRef": "5e9cecc5-ae7a-4f7e-b488-ed22daa0e2ab",
   "flavorRef": "1",
   "key_name" : "MyFirstKeyPair"
 }
}' http://127.0.0.1:8774/v2/[YOUR_TENANT_ID]/servers | python -m json.tool
```

블록 스토리지 제어 API

블록 스토리지

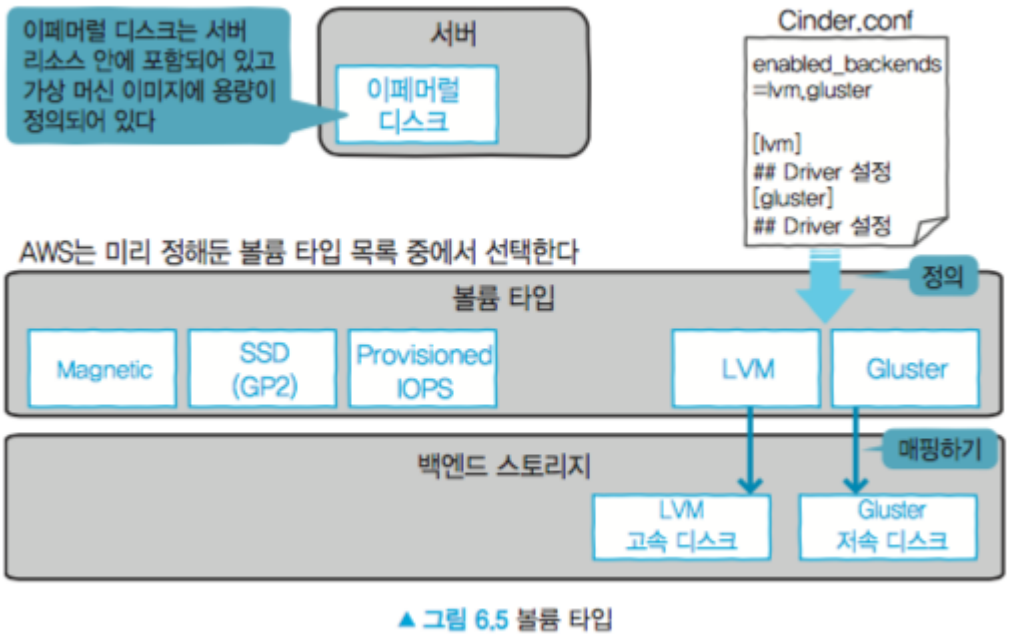
볼륨

- 서버에 연결되는 디스크. 비휘발성이고 영속적으로 보관가능
- ephemeral 디스크 는 휘발성 데이터. 이미지에 포함되어있음

스냅샷

- 볼륨을 복제한 것.
- 서버에 직접 연결할 수 없음
- 주로 백업이나 데이터를 이행할 때 사용

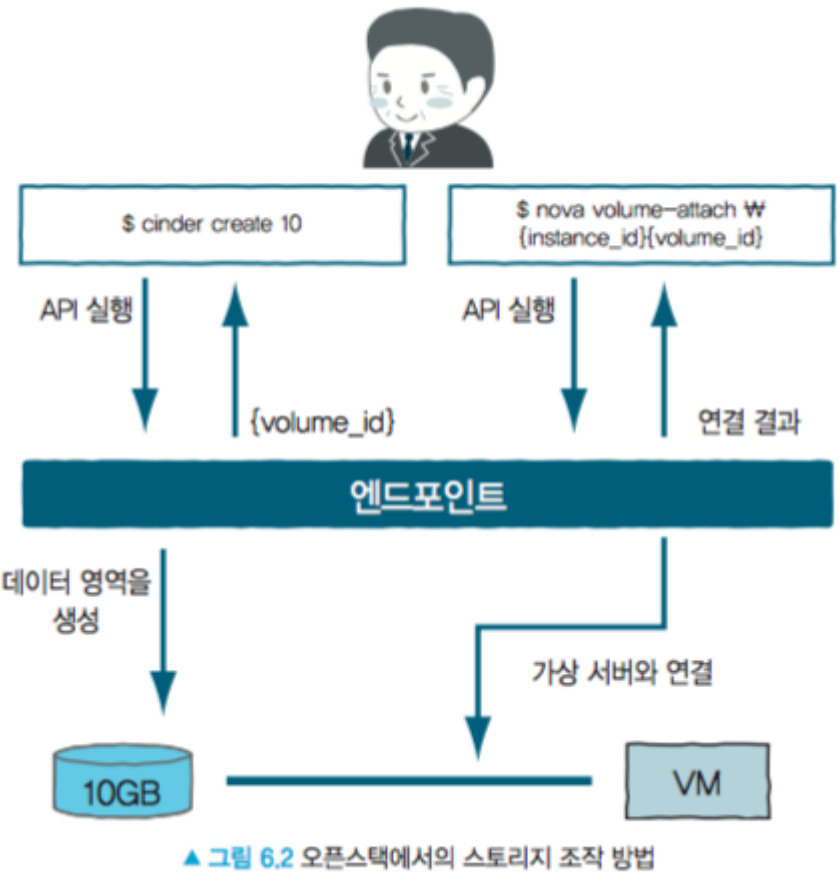
볼륨 타입



▲ 그림 6.5 볼륨 타입

- 출처: 그림으로 배우는 클라우드 인프라와 API의 구조, P202

API 처리 흐름



▲ 그림 6.2 오픈스택에서의 스토리지 조작 방법

- 출처: 그림으로 배우는 클라우드 인프라와 API의 구조, P196

Step1. 볼륨 생성

Request

POST /v3/ {project_id} /volumes

Create a volume


```
{
    "volume" : {
        "size" :  10,
        "availability_zone" :  null,
        "source_volid" :  null,
        "description" :  null,
        "multiattach " :  false,
        "snapshot_id" :  null,
        "backup_id" :  null,
        "name" :  null,
        "imageRef" :  null,
        "volume_type" :  null,
        "metadata" : {
        },
        "consistencygroup_id" :  null
    }
}
```


Response


```
{
    "volume" : {
        "status" :  "creating",
        "migration_status" :  null,
        "user_id" :  "0eea4eabcf184061a3b6db1e0daaf010",
        "attachments" : [
        ],
        "links" : [
            {
                "href" :
                 "http://23.253.248.171:8776/v3/bab7d5c60cd041a0a36f7c4b6e1dd978/volumes/6edbc2f4-1507-44f8-ac0d-e
                "rel" :  "self"
            },
            {
                "href" :
                 "http://23.253.248.171:8776/bab7d5c60cd041a0a36f7c4b6e1dd978/volumes/6edbc2f4-1507-44f8-ac0d-eed1
                "rel" :  "bookmark"
            }
        ],
        "availability_zone" :  "nova",
        "bootable" :  "false",
        "encrypted" :  false,
        "created_at" :  "2015-11-29T03:01:44.000000",
        "description" :  null,
        "updated_at" :  null,
        "volume_type" :  "lvmdriver-1",
        "name" :  "test-volume-attachments",
        "replication_status" :  "disabled",
        "consistencygroup_id" :  null,
```

```
    "source_volid" :  null,
    "snapshot_id" :  null,
    "multiattach" :  false,
    "metadata" : {
    },
    "id" :  "6edbc2f4-1507-44f8-ac0d-eed1d2608d38",
    "size" :  2
  }
}
```


Step2. 서버에 attach

Request

POST /servers/ `{server_id}` /os-volume_attachments

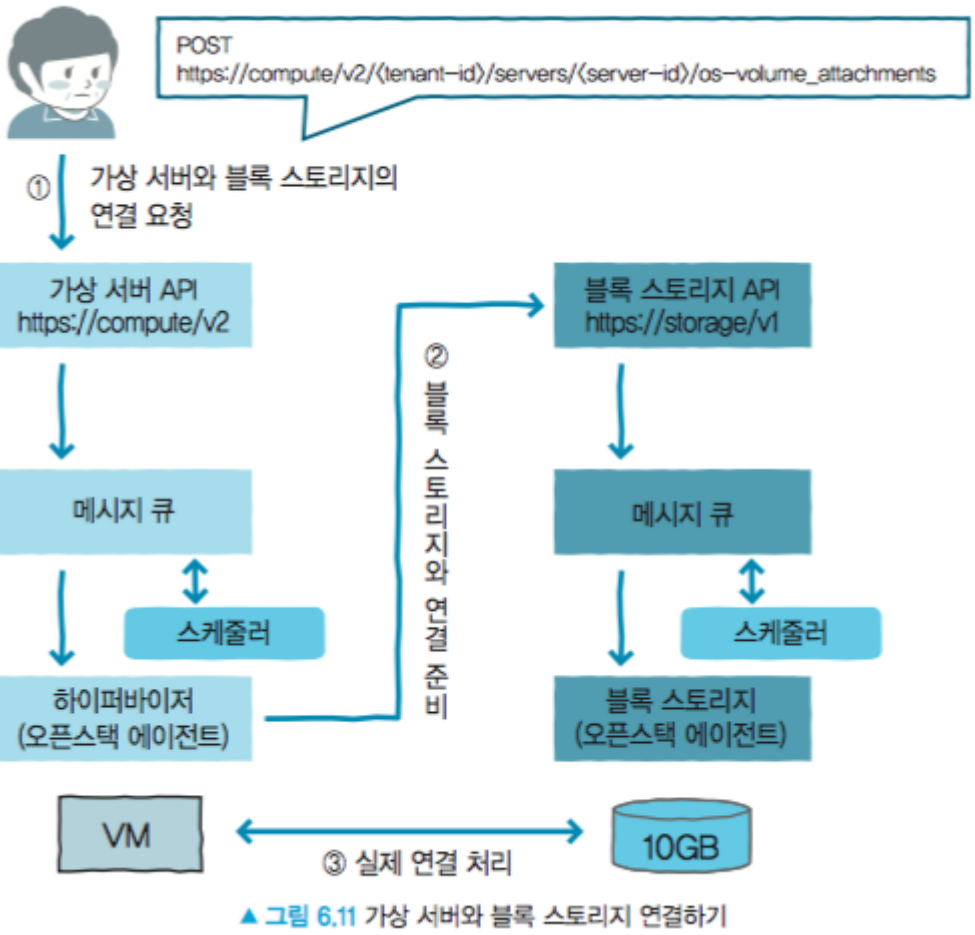
Attach a volume to an instance


```
{
  "volumeAttachment" : {
    "volumeId" :  "a26887c6-c47b-4654-abb5-dfadf7d3f803",
    "device" :  "/dev/vdd"
  }
}
```


Response


```
{
  "volumeAttachment" : {
    "device" :  "/dev/vdd",
    "id" :  "a26887c6-c47b-4654-abb5-dfadf7d3f803",
    "serverId" :  "0c92f3f6-c253-4c9b-bd43-e880a8d2eb0a",
    "volumeId" :  "a26887c6-c47b-4654-abb5-dfadf7d3f803"
  }
}
```

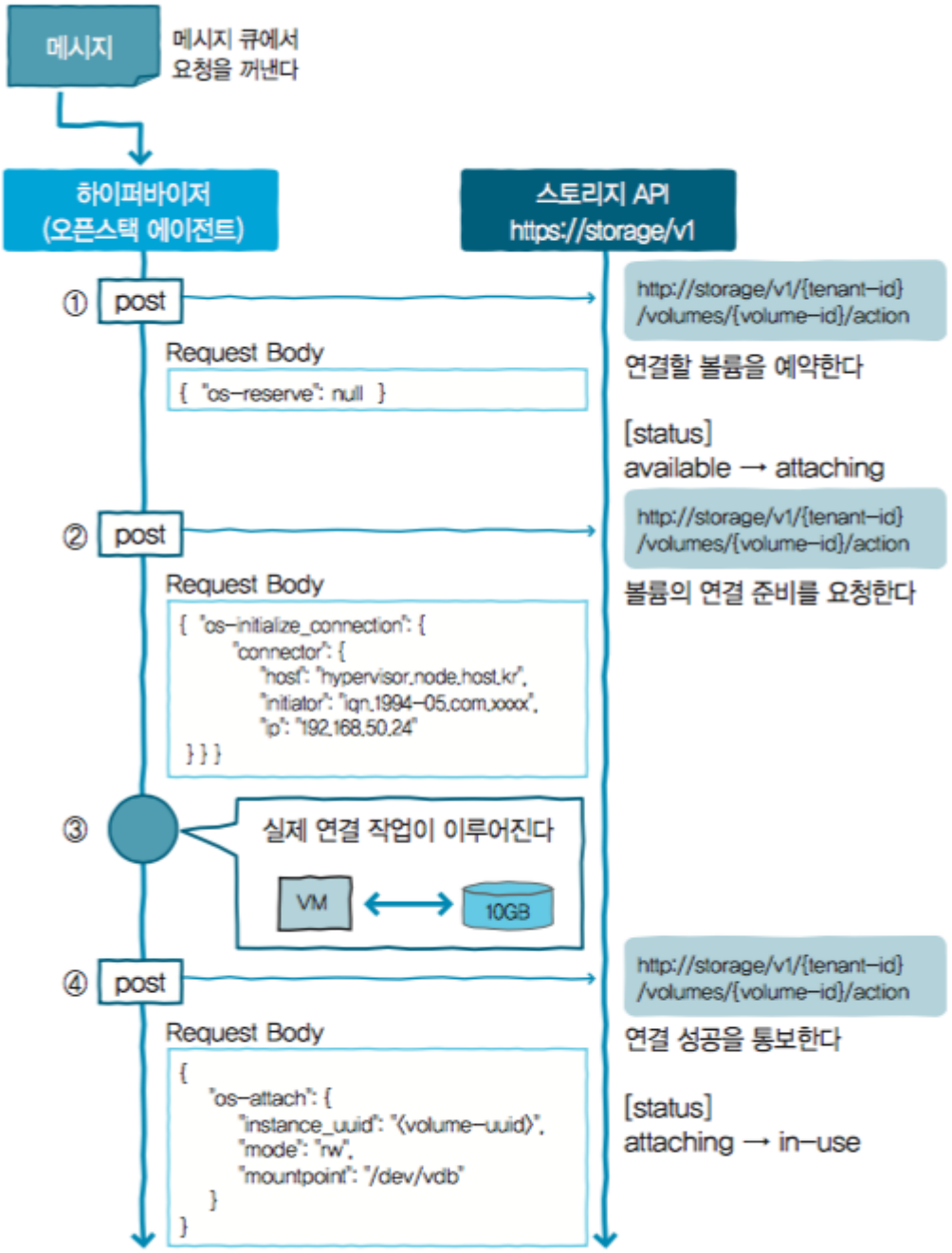

내부에서의 처리



▲ 그림 6.11 가상 서버와 블록 스토리지 연결하기

- 출처: 그림으로 배우는 클라우드 인프라와 API의 구조, P211

내부에서의 처리 - API 관점



▲ 그림 6.12 서버와 스토리지의 자동 협상

- 출처: 그림으로 배우는 클라우드 인프라와 API의 구조, P213