

# Terraform with OpenStack Provider

# Contents

## Terraform ?

### Terraform 동작 방식

Terraform 기본 명령어

### Provider

OpenStack

## Terraform Code

(1) TF 설정 파일 및 실행 관련 정보 작성

(2) Terraform 기본 문법

(3) Blocks

(4) 인스턴스 + OS 이미지

(5) Block-storage Volume

(6) Object-storage로 State 파일 관리 및  
조회

그 밖에..

# Terraform

# Terraform ?

HashiCorp가 개발한 오픈소스형 Infrastructure, **laC** 도구  
선언형 구성언어인 **HCL**로 Infrastructure 정의

# Terraform ?

HashiCorp가 개발한 오픈소스형 Infrastructure, **IaC** 도구  
선언형 구성언어인 **HCL**로 Infrastructure 정의

1

## Self-service

개발자가 원하는 시점에 배포

2

## 속도와 안정성

배포 절차의 자동화  
빠르고 안전한 리소스 배포

3

## 문서화, 버전관리

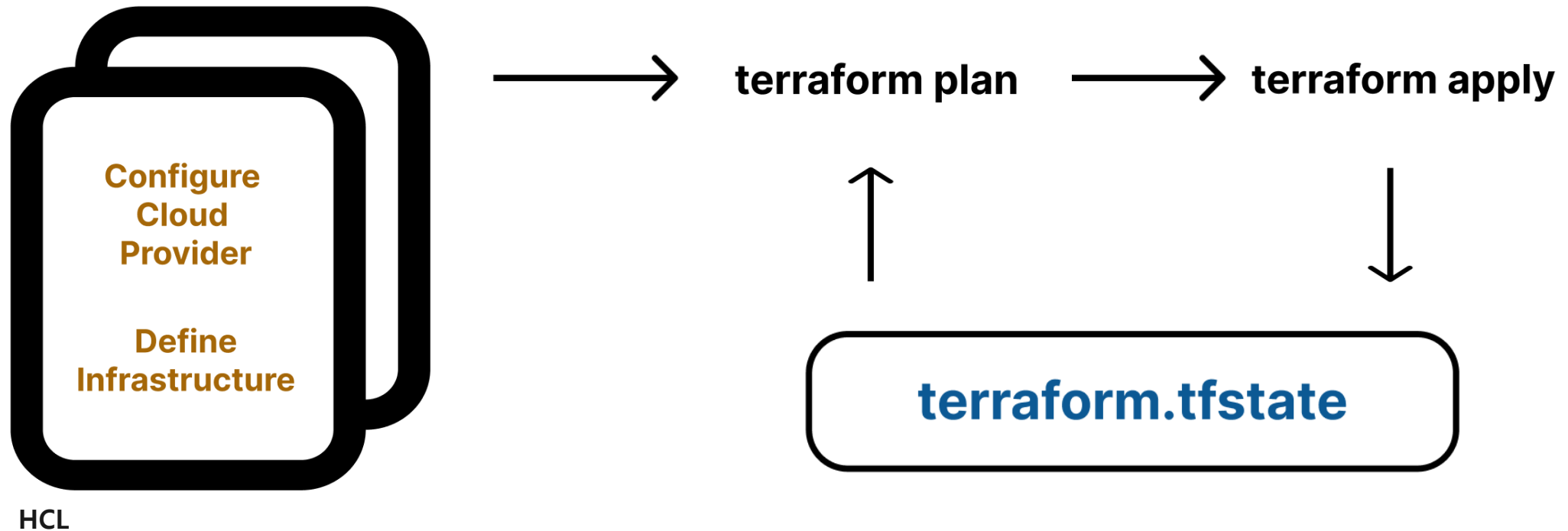
누구나 쉽게 읽기 가능  
문제 발생 시, 버전을 통해 쉽게 복구가 가능

4

## 재사용성

모듈화된 코드로 인프라를 설계하고 배포하는데 공통된 사항들을  
재사용

# Terraform 동작방식



# Terraform 기본 명령어

Terraform의 시작부터 배포 및 삭제까지

## terraform init

- 테라폼 실행에 필요한 각종 파일들을 작업 디렉토리에 생성 및 초기화

## terraform plan

- 정의한 코드가 어떤 인프라를 만들게 되는지 미리 예측 결과를 나타냄
- 어떠한 형상에도 변화를 주지 않음

## terraform apply

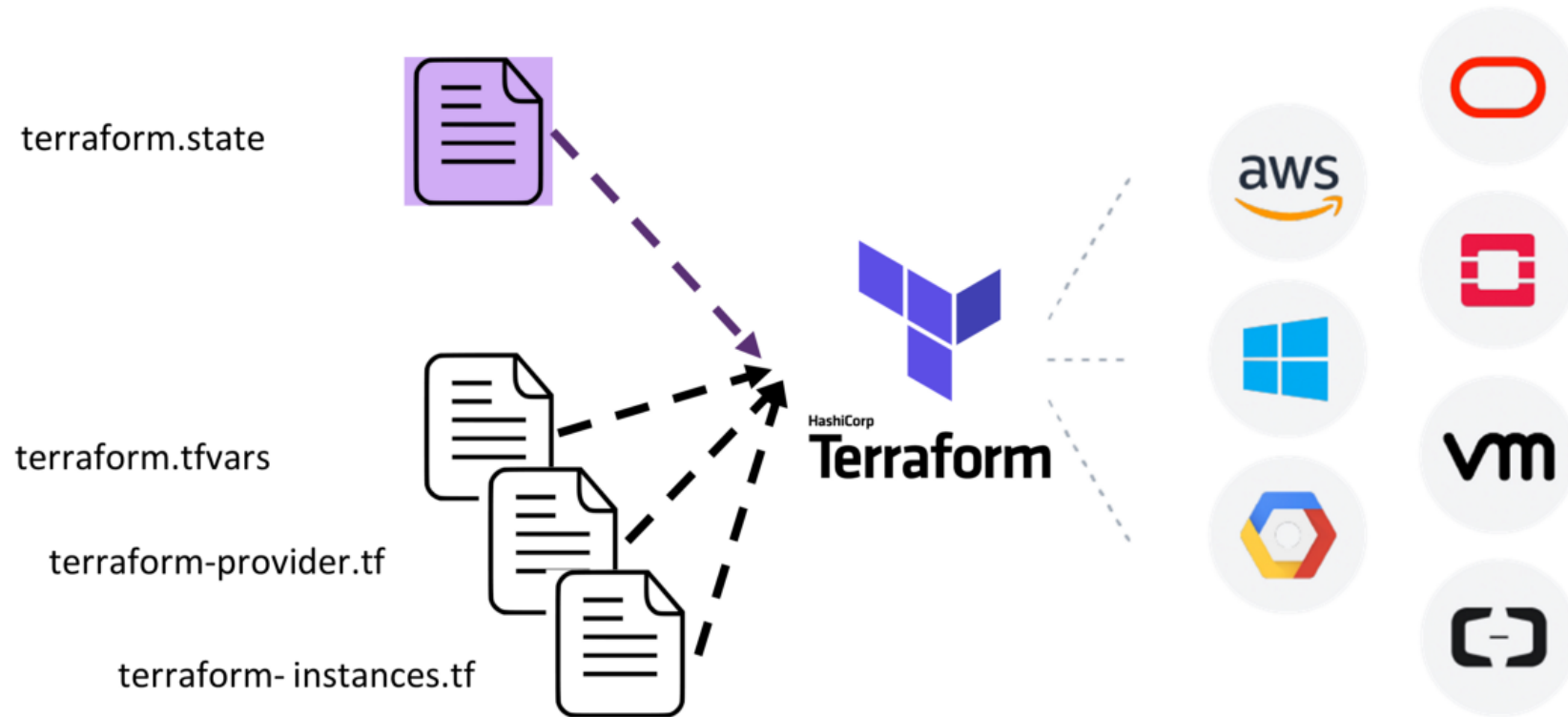
- 실제로 인프라에 배포하기 위한 명령어
- 작업결과가 .tfstate 파일에 작성됨

## terraform destroy

- 테라폼과 관련된 모든 리소스 삭제

# Provider ?

Terraform의 provider는 실제 리소스가 생성, 변경 및 삭제될 외부 서비스를 연결해주는 제공자(Plug-in).  
Terraform은 각 인프라 Provider의 API를 사용하여 리소스를 생성하고 구성하는 단계를 지원.

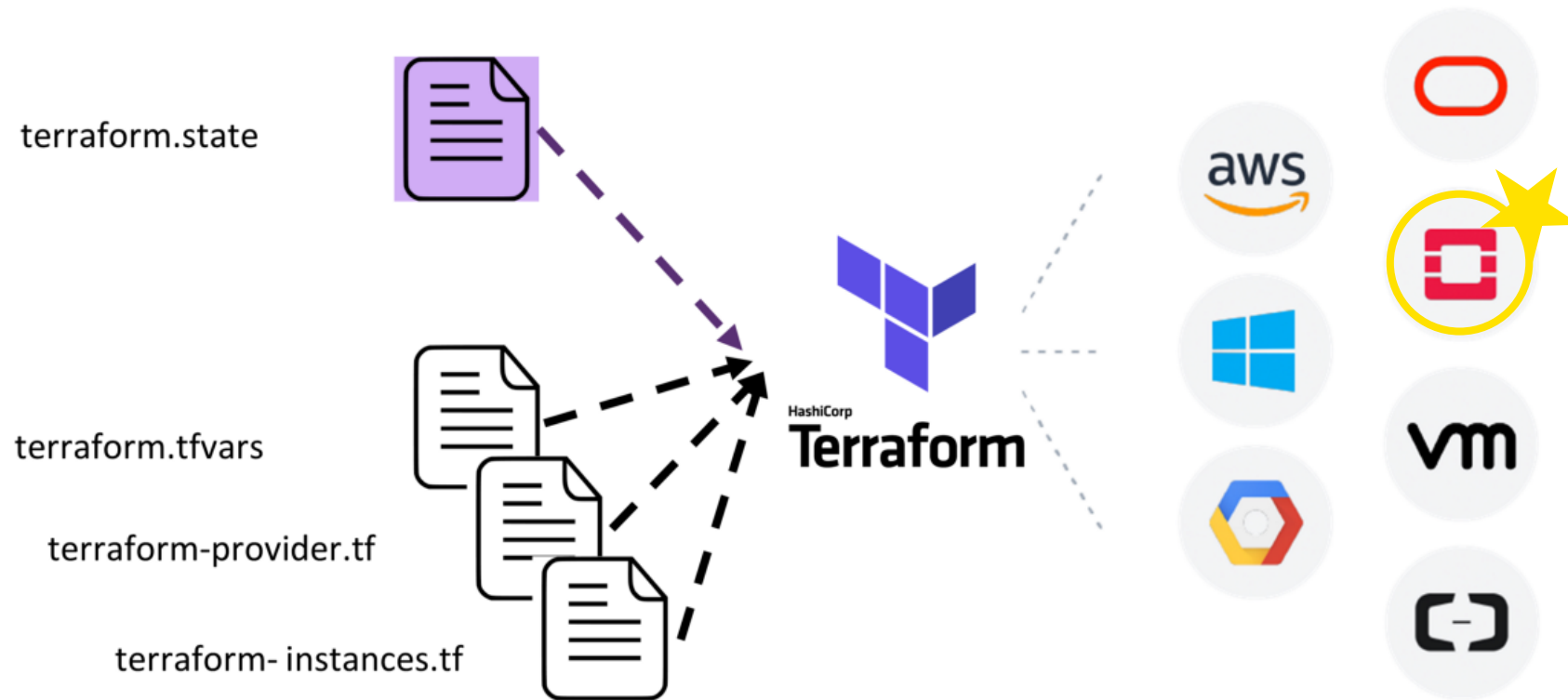




# Provider ?

Terraform의 provider는 실제 리소스가 생성, 변경 및 삭제될 외부 서비스를 연결해주는 제공자(Plug-in).







Terraform은 각 인프라 **Provider**의 API를 사용하여 리소스를 생성하고 구성하는 단계를 지원.



# OpenStack

OpenStack은 IaaS형태의 클라우드 컴퓨팅 오픈 소스 프로젝트.  
컴퓨팅, 네트워킹, 스토리지, 인증, 이미지 처리과 같은 6가지 핵심 서비스 +a



 NOVA  Nova는 OpenStack 컴퓨팅 리소스를 위한 전체 관리 및 액세스 톨로 스케줄링, 생성, 삭제를 처리합니다.	 NEUTRON  Neutron은 기타 OpenStack 서비스 전반에서 네트워크를 연결합니다.	 SWIFT  Swift는 내결함성이 뛰어난 오브젝트 스토리지 서비스로, RESTful API를 사용해 구조화되지 않은 애플리케이션을 저장 및 검색합니다.
 CINDER  Cinder는 셀프 서비스 API를 통해 액세스할 수 있는 퍼시스턴트 블록 스토리지입니다.	 KEYSTONE  Keystone은 모든 OpenStack 서비스를 인증하고 권한을 부여하며 모든 서비스를 위한 엔드포인트 카탈로그의 역할도 합니다.	 GLANCE  Glance는 다양한 위치에 있는 가상 머신 디스크의 이미지를 저장하고 검색합니다.

# Terraform Code



# TF 설정 파일 및 실행 관련 정보 작성

Terraform을 시작하기전에, **자원 생성을 요청하기 위한 Region 및 인증키와 같은 설정파일 필요**

```
# terraform 버전과 provider 등 실행에 관련된 정보 작성
terraform {
  required_version = ">= 1.0"
  required_providers {
    openstack = {
      source = "terraform-provider-openstack/openstack"
      version = ">= 1.40.0"
    }
  }
}

provider "openstack" {
  cloud = "lana-3tier"
}
```

# Terraform 기본 문법

```
resource "PROVIDER_TYPE" "NAME" {  
  [CONFIG ...]  
}
```

**PROVIDER** : OpenStack과 같은 공급자의 이름

**TYPE** : instanc와 같이 생성하고자 하는 리소스의 종류

**NAME** : 식별자

**CONFIG** : 해당 리소스에 선언할 수 있는 하나 이상의 설정 변수 값들

# Blocks

Terraform에는 infra를 구성하기 위한 Block들이 있다.  
해당 Block들이 모여 인프라를 구성하게 된다.

```
resource "PROVIDER_TYPE" "NAME" {  
  [CONFIG ...]  
}
```

## 1 resource

- 인프라에 배포하고자 하는 리소스를 정의

## 2 terraform

- 테라폼에 대한 설정 값
- 테라폼의 자체 설정

## 3 provider

- 각 provider 별 설정 값

## 4 variable

- 변수를 선언
- 입력을 받아야할 값들을 정의

## 5 local

- Variable과 마찬가지로 변수를 선언
- 동적으로 생성되는 값을 선언 -> 전(후)처리에 대한 로직

## 6 data

- 이미 terraform에서 구축된 값이나, 인프라(terraform 외부)에 생성된 값을 가져와서 변수를 저장

## 7 output

- 출력 변수 선언

# 인스턴스 + OS 이미지

기존코드

```
# 생성할 인스턴스 정보
resource "openstack_compute_instance_v2" "instance" {
  count      = var.instance_count
  name      = var.instance_name
  flavor_name = var.instance_flavor

  block_device {
    uuid          = data.openstack_images_image_v2.default_image.id
    source_type   = "image"
    volume_size   = 50
    boot_index    = 0
    destination_type = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.instance[count.index].id
  }
}
```



# 인스턴스 + OS 이미지

기존코드

```
# 생성할 인스턴스 정보
resource "openstack_compute_instance_v2" "instance" {
  count      = var.instance_count
  name      = var.instance_name
  flavor_name = var.instance_flavor

  block_device {
    uuid          = data.openstack_images_image_v2.default_image.id
    source_type   = "image"
    volume_size   = 50
    boot_index    = 0    0 -> Boot device
    destination_type = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.instance[count.index].id
  }
}
```

# 인스턴스 + OS 이미지

## 기존코드

```
# 생성할 인스턴스 정보
resource "openstack_compute_instance_v2" "instance" {
  count      = var.instance_count
  name       = var.instance_name
  flavor_name = var.instance_flavor

  block_device {
    uuid           = data.openstack_images_image_v2.default_image.id
    source_type     = "image"
    volume_size    = 50
    boot_index     = 0
    destination_type = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.instance[count.index].id
  }
}
```

tfstate 파일에서 os image가 생성되지 않은 오류 확인

```
"image_id": "Attempt to boot from volume - no image supplied",
"image_name": null,
```

## OPENSTACK DOCUMENTATION

3 matching results

## ▼ Resources

- [openstack\\_compute\\_instance\\_v2](#)

- openstack\_db\_instance\_v1

## ▼ Data Sources

- openstack\_compute\_instance\_v2

}

## Boot From Volume

```
resource "openstack_compute_instance_v2" "boot-from-volume" {  
  name          = "boot-from-volume"  
  flavor_id     = "3"  
  key_pair      = "my_key_pair_name"  
  security_groups = ["default"]  
}
```

```
  block_device {  
    uuid          = "<image-id>"  
    source_type   = "image"  
    volume_size  = 5  
    boot_index    = 0  
    destination_type = "volume"  
    delete_on_termination = true  
  }  
}
```

```
  network {  
    name = "my_network"  
  }  
}
```

## Boot From an Existing Volume

```
resource "openstack_blockstorage_volume_v1" "myvol" {  
  name = "myvol"  
}
```

## ON THIS PAGE

- [Example Usage](#)

- [Argument Reference](#)

- [Attributes Reference](#)

- [Notes](#)

- [Importing instances](#)

[Report an issue](#)

OPENSTACK DOCUMENTATION

Q instance

3 matching results

Resources

openstack\_compute\_instance\_v2

openstack\_db\_instance\_v1

Data Sources

openstack\_compute\_instance\_v2

}

Boot From Volume

```
resource "openstack_compute_instance_v2" "boot-from-volume" {
  name           = "boot-from-volume"
  flavor_id      = "3"
  key_pair       = "my_key_p
  security_groups = ["default

  block_device {
    uuid           = "
    source_type    = "
    volume_size   = 5
    boot_index     = 0
    destination_type = "
    delete_on_termination = t
  }

  network {
    name = "my_network"
  }
}
```

Boot From an Existing Volume

```
resource "openstack_blockstor
  name      = "myvol"
```

ON THIS PAGE

Example Usage

Argument Reference

Attributes Reference

Notes

SOFTWARE

USERS

COMMUNITY

MARKETPLACE

EVENTS

LEARN

DOCS

JOIN

LOG IN

Block Device Mapping in Nova

UPDATED: 2022-03-09 21:55

Nova has a concept of block devices that can be exposed to cloud instances. There are several types of block devices an instance can have (we will go into more details about this later in this document), and which ones are available depends on a particular deployment and the usage limitations set for tenants and users. Block device mapping is a way to organize and keep data about all of the block devices an instance has.

When we talk about block device mapping, we usually refer to one of two things

1. API/CLI structure and syntax for specifying block devices for an instance boot request
2. The data structure internal to Nova that is used for recording and keeping, which is ultimately persisted in the block\_device\_mapping table. However, Nova internally has several "slightly" different formats for representing the same data. All of them are documented in the code and or presented by a distinct set of classes, but not knowing that they exist might trip up people reading the code. So in addition to BlockDeviceMapping [1] objects that mirror the database schema, we have:

2.1 The API format - this is the set of raw key-value pairs received from the API client, and is almost immediately transformed into the object; however, some validations are done using this format. We will refer to this format as the 'API BDMS' from now on.

2.2 The virt driver format - this is the format defined by the classes in :mod: nova.virt.block\_device. This format is used and expected by the code in the various virt drivers. These classes, in addition to exposing a different format (mimicking the Python dict interface), also provide a place to bundle some functionality common to certain types of block devices (for example attaching volumes which has to interact with both Cinder and the virt driver code). We will refer to this format as 'Driver BDMS' from now on.

Data format and its history

Block device mapping v1 (aka legacy)

Intermezzo - problem with device names

Block device mapping v2

Valid source / destination combinations

©Kakao Enterprise Corp. All Rights Reserved.

# 인스턴스 + OS 이미지

## 해결

```
# 생성할 인스턴스 정보
resource "openstack_compute_instance_v2" "instance" {
  count      = var.instance_count
  name       = var.instance_name
  flavor_name = var.instance_flavor

  block_device {
    uuid           = data.openstack_images_image_v2.default_image.id
    source_type     = "image"
    volume_size    = 50
    boot_index     = 0
    destination_type = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.instance[count.index].id
  }
}
```

### destination\_type = "volume"

- 인스턴스를 관리하는 Nova에 BlockStorage 서비스를 제공하는 Cinder 기능을 붙임
- 그냥 BlockStorage Volume이 붙어서 인스턴스 생성

### destination\_type = "local"

- Hypervisor에서 생성된 OS 이미지를 기반으로 VM 인스턴스 생성

## 참고문서

[https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/resources/compute\\_instance\\_v2](https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/resources/compute_instance_v2)

<https://docs.openstack.org/nova/queens/user/block-device-mapping.html>

# 인스턴스 + OS 이미지

결과

Changes to Outputs:

```
+ output_image_id    = "4a658329-66fc-486e-b9b6-73b7f165d00f"
+ output_image_name = (known after apply)
```

terraform plan

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:

```
output_image_id = "4a658329-66fc-486e-b9b6-73b7f165d00f"
output_image_name = "CentOS 7.9"
```

terraform apply

<input type="checkbox"/> 인스턴스 이름	상태	타입	이미지	사설 IP	공인 IP	생성자	생성일
<input type="checkbox"/> <a href="#">lena-tf-instance</a>	● Active	A1-2-CO	CentOS 7.9	172.16.1.62	210.109.62.24	lena.o@kakaenter...	2022.12.29

KiC Console > VM > Instance

# Block-storage Volume

lena-tf-instance ▾

...

lena-tf-instance

인스턴스 상태

인스턴스 상태 변경

빠른 실행

● Active

▶ 시작

□ 정지

⏻ 종료

↺ 재시작

↺ 강제 재시작

시큐리티 그룹 설정

공인 IP 연결

시스템 로그 확인

세부 정보

네트워크

보안

볼륨

액션 로그

모니터링

인스턴스 ID

2c724f0e-4e48-4a9c-ac27-5bd743378755

인스턴스 이름

lena-tf-instance

생성자

lena.o@kakaenterprise.com

인스턴스 타입

A1-2-CO

생성일

2022.12.30 (금) 16:24:19 (0분)

vCPU

2 개

이미지 이름

CentOS 7.9

Memory

4 GB

키페어

-

볼륨

0 개

# Block-storage Volume

## (1) Block storage volume 리소스 생성

```
## Block storage ##
resource "openstack_blockstorage_volume_v3" "volume" {
  name = "test-volume"
  description = "first test volume"
  size = 3
}
```

## (2) 인스턴스에 block device로 붙이기

```
# 생성할 인스턴스 정보
resource "openstack_compute_instance_v2" "instance" {
  name = var.instance_name
  image_id = data.openstack_images_image_v2.default_image.id
  flavor_name = var.instance_flavor
  key_pair = openstack_compute_keypair_v2.test-keypair.name

  block_device {
    uuid              = data.openstack_images_image_v2.default_image.id
    source_type       = "image"
    boot_index        = 0
    destination_type  = "local"
  }

  block_device {
    source_type       = "blank"
    destination_type  = "volume"
    volume_size       = openstack_blockstorage_volume_v3.volume.size
    boot_index        = 1
    delete_on_termination = true
  }
}
```



# Block-storage Volume

## (2) 인스턴스에 block device로 붙이기

```
# 생성할 인스턴스 정보
resource "openstack_compute_instance_v2" "instance" {
  name = var.instance_name
  image_id = data.openstack_images_image_v2.default_image.id
  flavor_name = var.instance_flavor
  key_pair = openstack_compute_keypair_v2.test-keypair.name

  block_device {
    uuid              = data.openstack_images_image_v2.default_image.id
    source_type       = "image"
    boot_index        = 0
    destination_type  = "local"
  }

  block_device {
    source_type = "blank"
    destination_type = "volume"
    volume_size = openstack_blockstorage_volume_v3.volume.size
    boot_index = 1
    delete_on_termination = true
  }
}
```

### Block Device Mapping in Nova > Valid source / destination combinations

- **blank** -> **volume** : Create a blank Cinder volume and attaches it. This will also require the volume size to be set.

## 참고문서

[https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/resources/compute\\_instance\\_v2](https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/resources/compute_instance_v2)

<https://docs.openstack.org/nova/queens/user/block-device-mapping.html>

# Block-storage Volume

## (2) 인스턴스에 block device로 붙이기

```
# 생성할 인스턴스 정보
resource "openstack_compute_instance_v2" "instance" {
  name = var.instance_name
  image_id = data.openstack_images_image_v2.default_image.id
  flavor_name = var.instance_flavor
  key_pair = openstack_compute_keypair_v2.test-keypair.name

  block_device {
    uuid              = data.openstack_images_image_v2.default_image.uuid
    source_type       = "image"
    boot_index        = 0
    destination_type  = "local"
  }

  block_device {
    source_type = "blank"
    destination_type = "volume"
    volume_size = openstack_blockstorage_volume_v3.volume.size
    boot_index = 1
    delete_on_termination = true
  }
}
```

### Block Device Mapping in Nova > Valid source / destination combinations

- **blank** -> **volume** : Create a blank Cinder volume and attaches it. This will also require the volume size to be set.

### lena-tf-instance

lena-tf-instance

인스턴스 상태

인스턴스 상태 변경

빠른 실행

● Active

▶ 시작

□ 중지

⏻ 종료

↺ 재시작

↺ 강제 재시작

시큐리티 그룹 설정

공인 IP 연결

시스템 로그 확인

세부 정보

네트워크

보안

볼륨

액션 로그

모니터링

+ 볼륨 추가

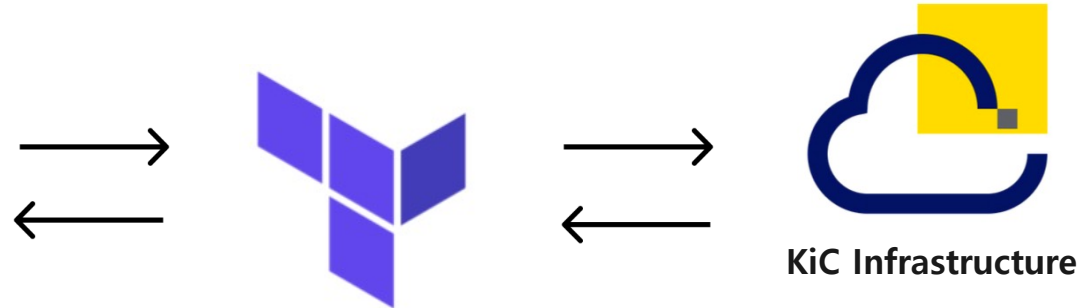
볼륨	상태	마운트포인트	타입	크기	인스턴스와 함께 삭제	생성일
 6a75b23e-144a-49dd-a621-391ba96190d2-...	● In-Use	/dev/vdb	SSD	3 GB	<input checked="" type="checkbox"/>	2023.01.05 (목) 05:54:57

# Object-storage로 State 파일 조회 및 관리

Terraform은 State 파일(tfstate)을 통해 상태를 확인하고 실제 인프라에 변경사항을 반영하고 새로 리소스를 생성.  
따라서, **실제 배포된 인프라와 State 파일(tfstate)의 상태를 일치**시키는게 테라폼의 핵심!

## terraform.tfstate

```
1  'version': 4,  
2  'terraform_version': '1.3.6',  
3  'serial': 367,  
4  'lineage': '8533b243-e46f-5ab2-0dd4-93a5d3d8985f',  
5  'outputs': {  
6    'output_image_id': {  
7      'value': '4a658329-66fc-486e-b9b6-73b7f165d00f',  
8      'type': 'string'  
9    },  
10   'output_image_name': {  
11     'value': 'CentOS 7.9',  
12     'type': 'string'  
13   },  
14   'ssh_file': {  
15     'value': './ex-vm-terraform.pem',  
16     'type': 'string'  
17   },  
18 },  
19 'resources': [  
20   {  
21     'mode': 'data',  
22     'type': 'openstack_identity_auth_scope_v3',  
23     'name': 'current',  
24     'provider': 'provider[\"registry.terraform.io/terraform-provider-openstack/openstack\"]',  
25     'instances': [  
26       {  
27         'schema_version': 0,  
28         'attributes': {  
29           'domain_id': '',  
30           'domain_name': '',  
31           'id': 'current',  
32           'name': 'current',  
33           'project_domain_id': '91c77647d41747d79de02f51dc82b576',  
34           'project_domain_name': 'kic-edu',  
35           'project_id': 'a103bfeb3e2643b6889d4d855ff63eee',  
36           'project_name': 'edu-project-27',  
37           'region': 'kr-central-1',  
38           'roles': [  
39             {  
40               'role_id': '4ce45745bbaf42d09ea3fadcd8799c0',  
41               'role_name': 'member'  
42             },  
43           ]  
44         },  
45       }  
46     ],  
47   }  
48 ],  
49 }
```

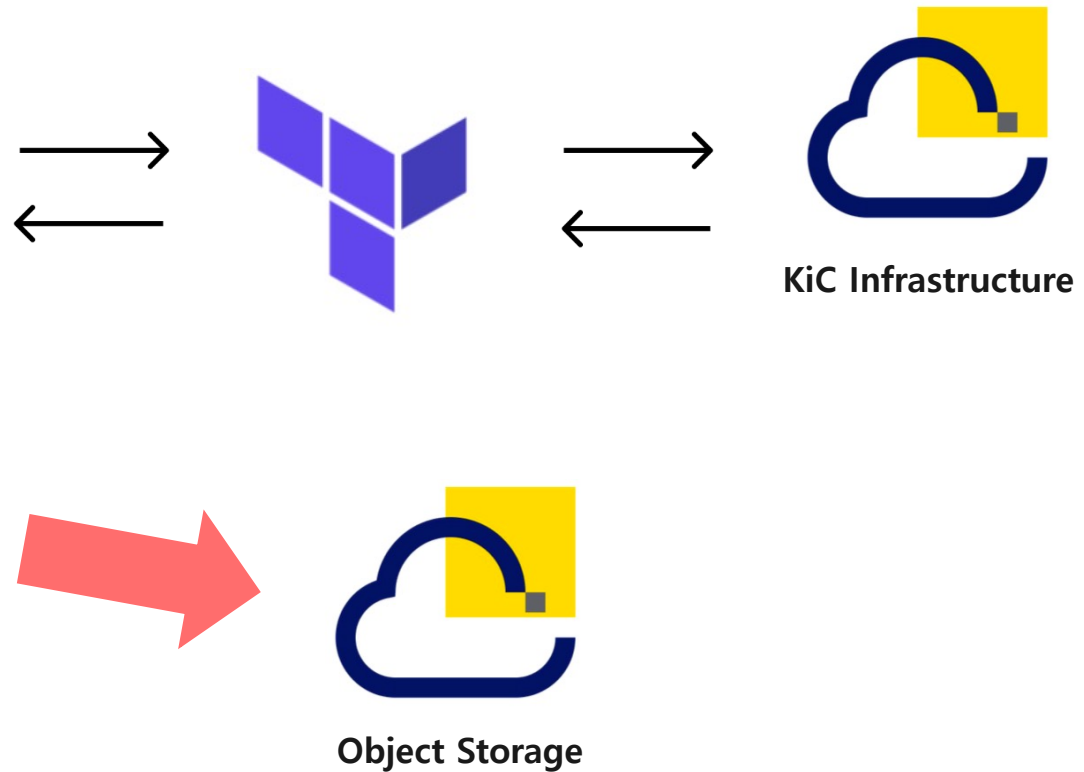


# Object-storage로 State 파일 조회 및 관리

Terraform은 State 파일(.tfstate)을 통해 상태를 확인하고 실제 인프라에 변경사항을 반영하고 새로 리소스를 생성.  
따라서, **실제 배포된 인프라와 State 파일(.tfstate)의 상태를 일치**시키는게 테라폼의 핵심!

## terraform.tfstate

```
1  'version': 4,  
2  'terraform_version': '1.3.6',  
3  'serial': 367,  
4  'lineage': '8533b243-e46f-5ab2-0dd4-93a5d3d8985f',  
5  'outputs': {  
6    'output_image_id': {  
7      'value': '4a658329-66fc-486e-b9b6-73b7f165d08f',  
8      'type': 'string'  
9    },  
10   'output_image_name': {  
11     'value': 'CentOS 7.9',  
12     'type': 'string'  
13   },  
14   'ssh_file': {  
15     'value': './ex-vm-terraform.pem',  
16     'type': 'string'  
17   },  
18 },  
19 ,  
20 'resources': [  
21   {  
22     'mode': 'data',  
23     'type': 'openstack_identity_auth_scope_v3',  
24     'name': 'current',  
25     'provider': 'provider[\"registry.terraform.io/terraform-provider-openstack/openstack\"]',  
26     'instances': [  
27       {  
28         'schema_version': 0,  
29         'attributes': {  
30           'domain_id': '',  
31           'domain_name': '',  
32           'id': 'current',  
33           'name': 'current',  
34           'project_domain_id': '91c77647d41747d79de02f51dc82b576',  
35           'project_domain_name': 'kic-edu',  
36           'project_id': 'a103bfeb3e2643b6889d4d855ff63eee',  
37           'project_name': 'edu-project-27',  
38           'region': 'kr-central-1',  
39           'roles': [  
40             {  
41               'role_id': '4ce45745bbaf42d09ea3fadcd8799c0',  
42               'role_name': 'member'  
43             },  
44           ]  
45         },  
46       }  
47     ],  
48   }  
49 ],  
50 }
```



# Object-storage로 State 파일 조회 및 관리

```
## Object storage ##
resource "openstack_objectstorage_container_v1" "container-1" {
  region = "kr-central-1"
  name = "lena-bucket"

  content_type = "application/json"
}

resource "openstack_objectstorage_object_v1" "file" {
  region          = "kr-central-1"
  container_name = openstack_objectstorage_container_v1.container-1.name
  name            = "tf-state.json"

  content_type = "application/json"
  source       = "./terraform.tfstate"
}
```

## openstack\_objectstorage\_container\_v1

- object들을 담는 container로, kic에서는 버킷과 같다.
- 리소스가 적용될 region과 content\_type 작성

## openstack\_objectstorage\_object\_v1

- 객체 형태로 저장되는 파일 생성.

## 참고문서

[https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/resources/objectstorage\\_container\\_v1](https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/resources/objectstorage_container_v1)

[https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/resources/objectstorage\\_object\\_v1](https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs/resources/objectstorage_object_v1)

# Object-storage로 State 파일 조회 및 관리

결과

Buckets Media Convert

Object Storage > Buckets > lena-bucket > Objects

+ 새 버킷 만들기

lena-bucket ▾

...

타입	최종 수정일	Life Cycle 적용 여부	빠른 실행
HOT	2023.01.05 10:47	● 미적용	Life Cycle 설정

객체	세부 정보	권한	모니터링
----	-------	----	------

lena-bucket

파일 업로드폴더 만들기

i

업로드 진행 중 또는 손상 등의 이유로 불완전한 객체의 경우 메타 데이터 수정 / 이름 바꾸기 / 복사 / 이동 기능을, 업로드를 마친 멀티 파트 업로드 객체의 경우 위의 네 기능 중 메타 데이터를 제외한 나머지를 사용할 수 없습니다.

객체 필터

객체 수 : 1

<input type="checkbox"/>	이름	최종 수정일	크기	유형	
<input type="checkbox"/>	tf-state.json	2023.01.05 10:47	27.06 KB	application/json	⋮

페이지 당 행 수 : 10 ▾

<>

# Object-storage로 State 파일 조회 및 관리

## 해결 못한 부분

- terraform.tfstate 파일에 업데이트가 생겼을 때, 웹 콘솔에서의 state 파일이 자동으로 재업로드가 되지 않는 문제점을 확인
- 암호화가 되어있지 않음

## 해결 중

- Backend 설정이 필요 (테라폼의 상태를 정의할 Remote State Storage)
- AWS의 경우, s3 bucket에 상태를 저장(및 암호화)하도록 테라폼을 구성하기 위한 백엔드 설정을 함으로써 위의 문제를 해결함.

```
terraform {  
  backend "s3" {  
    bucket = (YOUR_BUCKET_NAME)  
    key = "terraform.tfstate"  
    region = "us-east-1"  
    encrypt = true  
  }  
}
```

- Openstack의 경우 swift object storage API를 호출하여 Container(bucket)을 생성하는거 같은데, backend "swift" 라고 작성 -> 현재 지원하지 않음.
- Terraform에서 지원하는 "remote" : <https://2ham-s.tistory.com/405> 로 해결할 예정.

그 밖에도..



그 밖에..

하는 중..

Page.33

### **Object Storage**

- Backend 설정

### **3-tier Architecture**

- Nginx 설치, 프로젝트 Repository clone, build..... -> shell script

### **AWS Provider**

- 단일 웹 서버 배포
- 웹 서버 클러스터 구성 (ASG: Auto Scailing Group) 이용
- 로드밸런서 배포

**kakaoenterprise**