

[한 줄 요약]

~~설치는 쉬우나, 설정은 어렵다!~~

코드기반 클라우드 관리 능력이 필요하다.

## 1. Terraform

### 1-1) 기초 설명

- Terraform은 오픈소스 프로젝트이다.
- 워크 플로우를 제공한다.  
(Terraform Configuration -> Initialize -> Refresh -> Plan -> Apply -> Destroy)
- AWS, OpenStack, VM, Kubernetes, Github 등 다양한 플랫폼/서비스와 연결 가능하다.
- API를 제공하기에 IaaS - Kubernetes - DNS - CDNS 등 다양한 서비스 간 연결이 가능하다.
- 여러 명이 하나의 GitHub Repository를 사용하듯이 Terraform 워크플로우를 사용할 수 있다.
- 인프라가 휘발성 형태로 바뀌는 것이기 때문에 포렌식 하는 사람이 더 힘들어진다.
- 포렌식이 단순히 분석하는 입장이 아니라, 한 기업 인프라를 운영하기 위한 당연히 거쳐야 하는 과정이 된 것이다.

### 1-2) 키워드

- ① Provisioning(조서연)
  - Terraform 언어를 Cloud별 언어로 변경하는 단계. (Plan과 Apply 사이)
- ② Provider(최현석)
  - AWS, Adger 등 이용할 외부 서비스에 접근하기 위한 정보(configuration) 제공.
  - Provider를 이용하여 해당 외부 서비스에서 서비스를 적용시킬 수 있게 하는 것,
  - Configuration에서는 다양한 옵션을 추가할 수 있으며, 필요한 경우, 직접 개발 할 수도 있다.
- ③ Resource(김도훈)
  - Terraform을 이용하여 생성하게 되는 인프라 내부의 자원들.
  - EC2나 S3 버킷부터 시작하여 VPC(가상 네트워크)등의 객체를 말함.
- ④ HCL (Hashicorp configuration language) Similar to JSON(황희재)
  - Hashicorp에서 만든 JSON 호환 언어.
  - 추가적인 구조와 기능이 내장되어있다.
  - JSON과 완벽히 호환되는 만큼 코드가 JSON과 유사하다.

### 1-3) 동작 과정

- ① Initialize(김태룡)
  - Terraform에서 init은 Terraform 설정 파일을 통해 초기화 시키는 작업.
  - Initialize 작업은 새 설정 파일을 작성했거나, 복제본을 만들었을 때 가장 첫 번째로 실행됨.
  - terraform init [옵션] 명령어를 통해 진행 가능하며, 여러 번 실행해도 상관없음.
  - 프로바이더에 맞는 플러그인을 다운로드 받고, 기존인프라 상태를 가져 올 수 있는지 확인함.
  - Terraform 버전과 프로바이더 버전이 각각 맞지 않을 경우 Initialize를 통해 초기화 가능.
- ② Refresh(최근영)
  - 테라폼으로 만들 세상(world) 구성요소를 조정하는 단계.
  - 이를 통해 테라폼 View가 나오며 실제와 어떻게 다른지 비교 가능.

- 테라폼이 상태 파일을 통해 알고 있는 상태와 real-world 인프라를 조화시키는 단계.
  - 마지막으로 알려진 상태에서 변화를 감지하고 상태 파일을 업데이트하는 데 사용.
  - 인프라를 수정하는 것이 아니라 상태 파일을 수정하는 것.
- ③ Plan(백승훈)
- 코드에서 설계한 내용을 현실에서 구현하기 위해 필요한 사항(리소스 등)들을 준비하는 단계.
  - 코드와 현재 인프라 상태를 비교하여 생성/변경/삭제될 사항들을 보여줌.
- ④ Apply(임다연)
- 원하는 configuration 상태에 도달하는 데 필요한 변경사항 또는 terraform plan 실행 계획에  
서 생성된 사전 결정 작업 집합을 적용하는데 사용됨.
  - 현 디렉터리에서 configuration을 검색하고, 변경 사항을 적절하게 적용한다.
- ⑤ Destroy(이창엽)
- apply의 롤백 | apply에서 적용한 모든 리소스 변경 사항을 삭제함.
  - Destroy를 수행할 때 올바른 작업 순서를 결정하기 위해 종속성 그래프를 생성.  
(여러 리소스가 있는 복잡한 경우, 종속성을 존중하기 위해 적절한 순서로 폐기)
  - GCP -> 다른 리소스가 존재할 경우 VPC 네트워크 삭제를 허가하지 않음

## 2. Terraform으로 EC2 키페어 만들기 [Linux - Ubuntu]

### 2-1) Terraform 설치

- ① HashiCorp(Terraform)의 GPG키값 등록하기

```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
```

※ 다운로드 실패 시 : sudo apt install curl

- ② 공식 HashiCorp Linux repository 등록하기

```
sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com  
$(lsb_release -cs) main"
```

- ③ 업데이트 및 설치

```
sudo apt-get update && sudo apt-get install terraform
```

- ④ 정상 설치여부 확인

```
terraform -help
```

```
Usage: terraform [-version] [-help] <command> [args]  
...
```

- ⑤ 탭 완성기능 설치하기

```
terraform -install-autocomplete
```

## 2-2) AWS CLI 설치

- ① curl으로 AWS CLI 다운로드

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

- ② 다운로드 받은 AWS CLI 설치 파일 압축 해제

```
unzip awscliv2.zip
```

- ③ AWS CLI 설치

```
sudo ./aws/install
```

- ④ 정상 설치여부 확인

```
aws --version
```

```
aws-cli/2.1.1 Python/3.7.4 Linux/4.14.133-113.105.amzn2.x86_64 botocore/2.0.0
```

## 2-3) AWS 설정

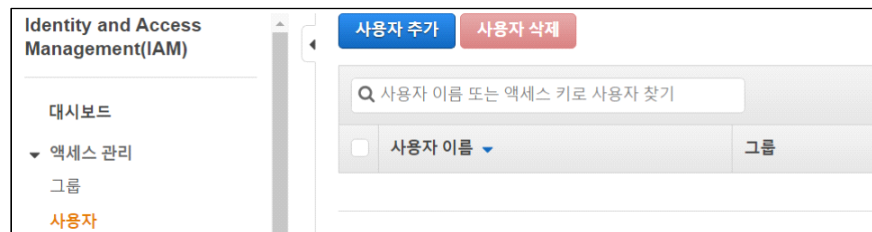
- ① 루트 사용자 계정으로 AWS 서비스 로그인

The image shows the AWS login interface. At the top is the title '로그인' (Login). Below it are two radio button options: '루트 사용자' (Root User) which is selected, and 'IAM 사용자' (IAM User). The 'Root User' option has a description: '무제한 액세스 권한이 필요한 작업을 수행하는 계정 소유자입니다. 자세히 알아보기' (Account owner who performs operations that require unlimited access permissions. Learn more). The 'IAM User' option has a description: '일일 작업을 수행하는 계정 내 사용자입니다. 자세히 알아보기' (User within the account who performs daily operations. Learn more). Below these options is a text input field labeled '루트 사용자 이메일 주소' (Root user email address) containing the text 'bob9@gmail.com'. At the bottom is a blue button labeled '다음' (Next).

[그림 2-1] 발급받은 Email로 로그인

- ② IAM 서비스 페이지 이동 (<https://console.aws.amazon.com/iam/>)

③ 좌측 메뉴에서 [사용자] 카테고리 선택 후, [사용자 추가] 선택.



④ 사용자 생성과 동시에 AccessKey 다운로드

**검토**

선택 항목을 검토합니다. 사용자를 생성한 후 자동으로 생성된 비밀번호와 액세스 키를 보고 다운로드할 수 있습니다.

**사용자 세부 정보**

사용자 이름	GoldBigDragon
AWS 액세스 유형	프로그래밍 방식 액세스 - 액세스 키 사용
권한 경계	권한 경계가 설정되지 않았습니다

**권한 요약**

다음 정책이 위에 표시된 사용자에게 연결됩니다.

유형	이름
관리형 정책	<a href="#">AdministratorAccess</a>
관리형 정책	<a href="#">AmazonEC2ContainerRegistryFullAccess</a>
관리형 정책	<a href="#">AmazonEC2ContainerServiceFullAccess</a>
관리형 정책	<a href="#">AmazonEC2FullAccess</a>

**태그**

태그가 추가되지 않았습니다.

[취소](#)
[이전](#)
[사용자 만들기](#)

**사용자 추가**

1 2 3 4 5

**성공**

아래에 표시된 사용자를 생성했습니다. 사용자 보안 자격 증명을 보고 다운로드할 수 있습니다. AWS Management Console 로그인을 위한 사용자 지침을 이메일로 보낼 수도 있습니다. 지금이 이 자격 증명을 다운로드할 수 있는 마지막 기회입니다. 하지만 언제든지 새 자격 증명을 생성할 수 있습니다.

AWS Management Console 액세스 권한이 있는 사용자가 <https://276179613427.signin.aws.amazon.com/console>에 로그인할 수 있습니다.

**.csv 다운로드**

클릭

사용자	액세스 키 ID	비밀 액세스 키
<div style="display: flex; align-items: center;"> <div style="margin-right: 5px;">▶</div> <div style="display: flex; align-items: center;"> <div style="width: 15px; height: 15px; background-color: green; border-radius: 50%; margin-right: 5px;"></div> GoldBigDragon </div> </div>	AKIAUATMWO3ZQ3PQUMHO	<div style="display: flex; align-items: center;"> <div style="width: 15px; height: 15px; background-color: #ccc; border-radius: 50%; margin-right: 5px;"></div> <div style="display: flex; flex-direction: column;"> <span>*****</span> <span>표시</span> </div> </div>

## 2-4) Terraform 설정

① 설정 파일 저장 디렉터리 생성

```
mkdir terra && cd terra
```

② Provider 설정 파일 작성 후 저장

```
vi provider.tf
```

#### provider.tf

```
provider "aws" {  
  access_key = "다운로드받은 ACCESS KEY ID"  
  secret_key = "다운로드받은 ACCESS SECRET KEY"  
  region = "us-east-2"  
}
```

※ 저장 : ESC → w! → q!

### ③ Provider 버전에 맞는 플러그인 다운로드

```
terraform init
```

```
Initializing the backend..  
..  
Terraform has been successfully initialized!  
..
```

### ④ 설치된 Provider 버전 확인

```
terraform version
```

```
Terraform v0.14.4  
+ provider registry.terraform.io/hashicorp/aws v3.23.0
```

### ⑤ 접속용 공개 키 파일 생성

```
ssh-keygen -t rsa -b 4096 -C "본인Email주소" -f "$HOME/.ssh/terraform" -N ""
```

### ⑥ Infra 설정 파일 작성 후 저장

```
vi infra.tf
```

#### infra.tf

```
resource "aws_key_pair" "web_admin" {  
  key_name = "terraform"  
  public_key = file("~/ssh/terraform.pub")  
}
```

⑦ 설정대로 생성 가능한지 미리 확인하기

### terraform plan

An execution plan has been generated and is shown below.

Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

# aws\_key\_pair.web\_admin will be created

...

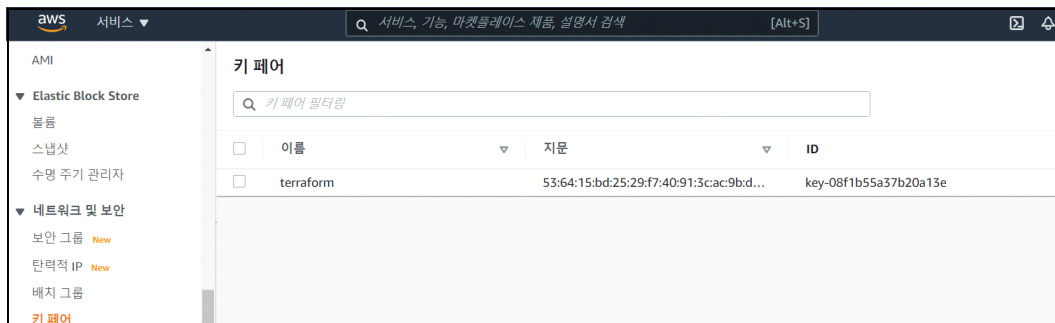
⑧ 설정을 AWS 서버에 적용

### terraform apply (중간에 yes 입력 필요)

aws\_key\_pair.web\_admin: Creating...

aws\_key\_pair.web\_admin: Creation complete after 3s [id=terraform]

⑨ AWS 홈페이지 EC2 서비스의 키 페어 카테고리에서 terraform 등록 여부 확인



⑩ EC2 인스턴스 생성 구문 작성

### infra.tf

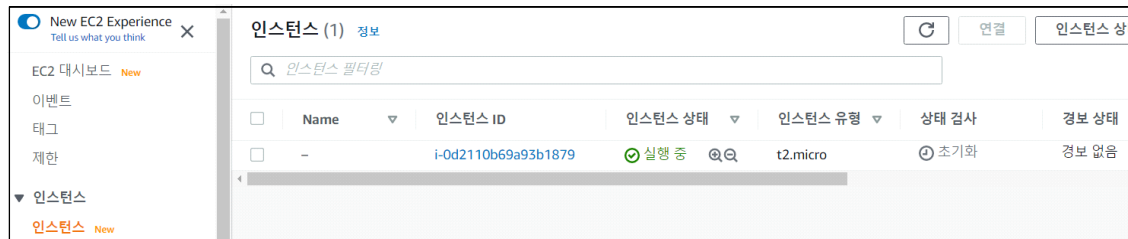
```
resource "aws_key_pair" "web_admin" {
  key_name = "terraform"
  public_key = file("~/ssh/terraform.pub")
}

resource "aws_instance" "web" {
  ami = "ami-0a0ad6b70e61be944"
  instance_type = "t2.micro"
  key_name = aws_key_pair.web_admin.key_name
}
```

⑪ EC2 인스턴스 생성

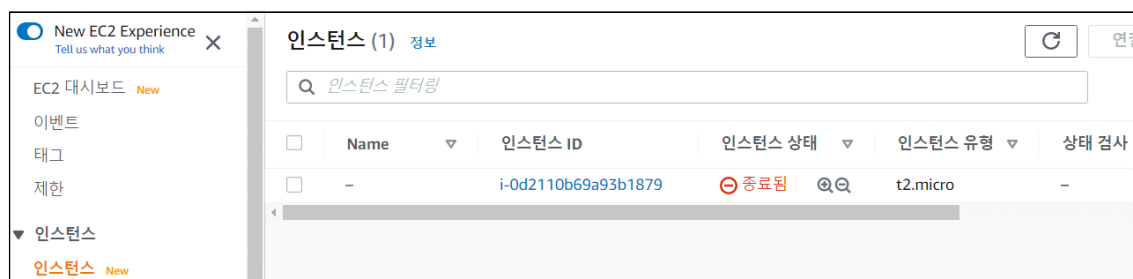
terraform plan

terraform apply  
(중간에 yes 입력 필요)



## ⑫ EC2 인스턴스 제거

terraform destroy  
(중간에 yes 입력 필요)



(종료 안하면 Niko멘토님이 때찌하실 것임)

## 3. 기타

- SSO : 개인정보를 갖지 않고 세션을 유지하면서 웹 서비스를 지속적으로 이용하는 것.
- AWS에서는 Organization으로 계정을 이용한다.
- Organization : 여러 계정을 묶어서 통합으로 관리해 주는 것. (정책도 모아서, 비용도 모아서)
- 오늘 EC2인스턴스를 만들고 Metadata를 써볼 생각이었지만, 오늘 Terraform을 해 볼 생각.
- 클라우드 3대 키워드 : Terraform / Ansible / SAM
- 얼마나 많은 자원이 들어갈지 모르는 상황에서 인프라 자원을 준비하는 상황 등 때문에 클라우드를 사용하기 시작함.
- 웹 서비스 불편할 때 : EC2 인스턴스를 서로 다른 조건으로 100개를 만들어야 할 때.
  - => 이 때문에 CLI 서비스 및 코드 작성이 나옴.
  - => 하지만 그 두 개 조치도 불편하여 Terraform이 나옴.
- IaaC : Infra as Code : 인프라를 코드로 관리하자! = Terraform, Ansible, SAM
  - => SAM은 AWS에서만 사용 가능해서 인기가 없다. 클라우드마다 제공 해 주는 서비스가 너무 다르기 때문.
- Terraform 설명에 시간이 오래 걸리므로 동영상을 끌고왔음!  
introduction to HashCorp Terraform with Amon Dadgar (<https://youtu.be/h970ZBgKINg>)
- Terraform 이해를 돕는 사이트 <https://registry.terraform.io/> <https://tinyurl.com/y4zd68h2>  
<https://registry.terraform.io/search/modules?q=aws>