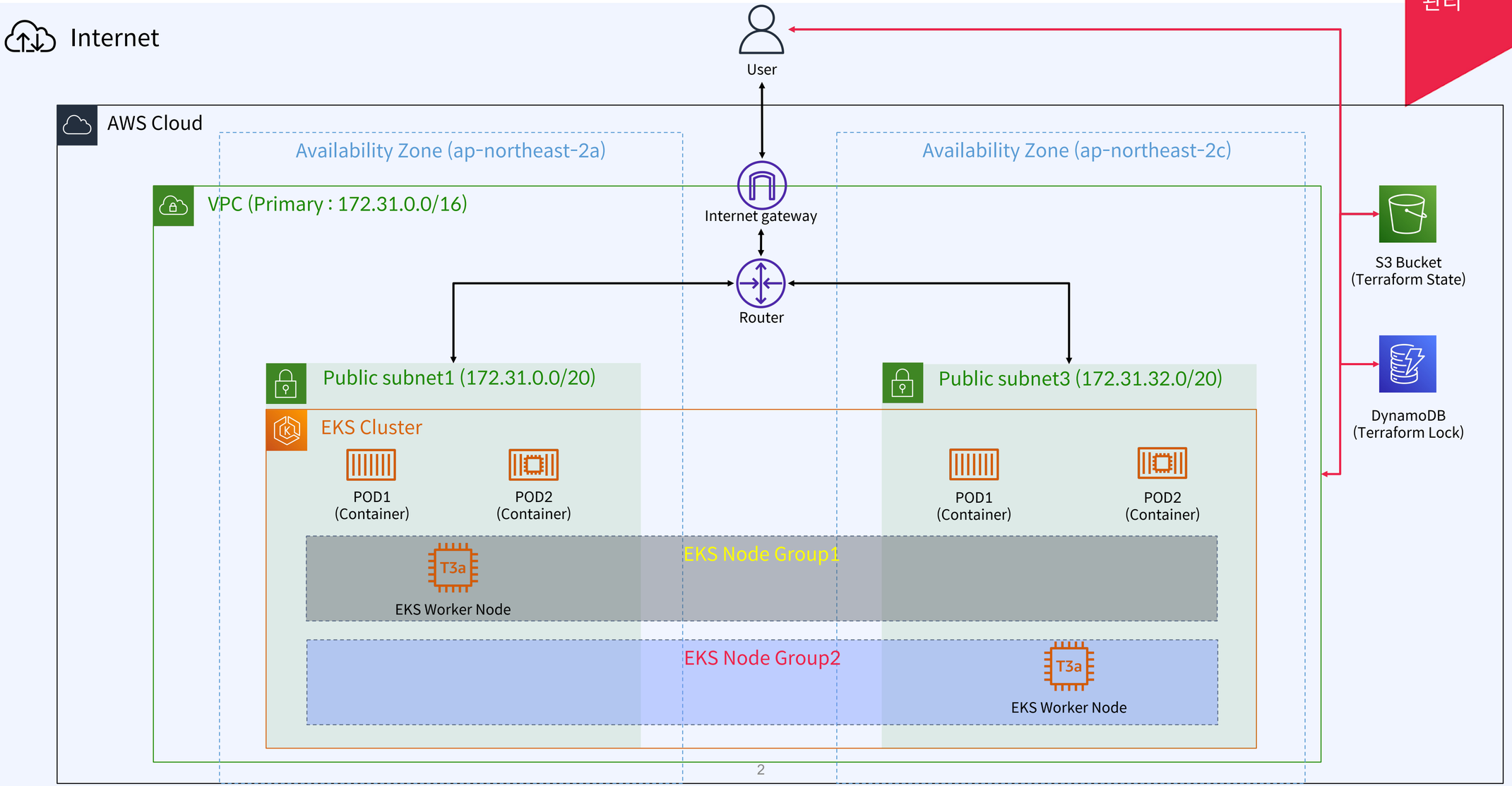
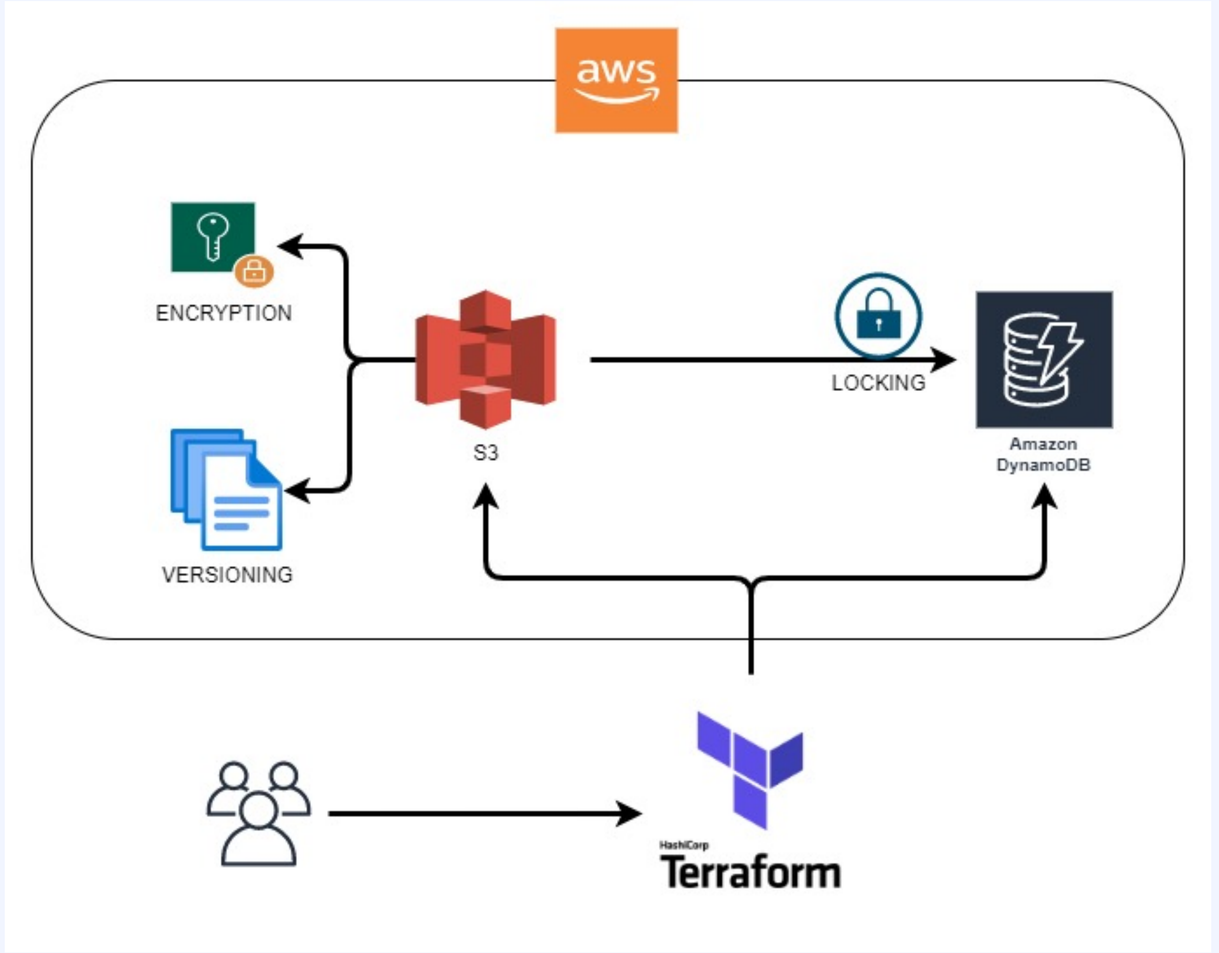
2 Terraform을 활용한 AWS EKS 생성

06 Terraformer를 활용한 AWS EKS 관리

AWS 네트워크 및 EKS 구성도 (실습 환경)

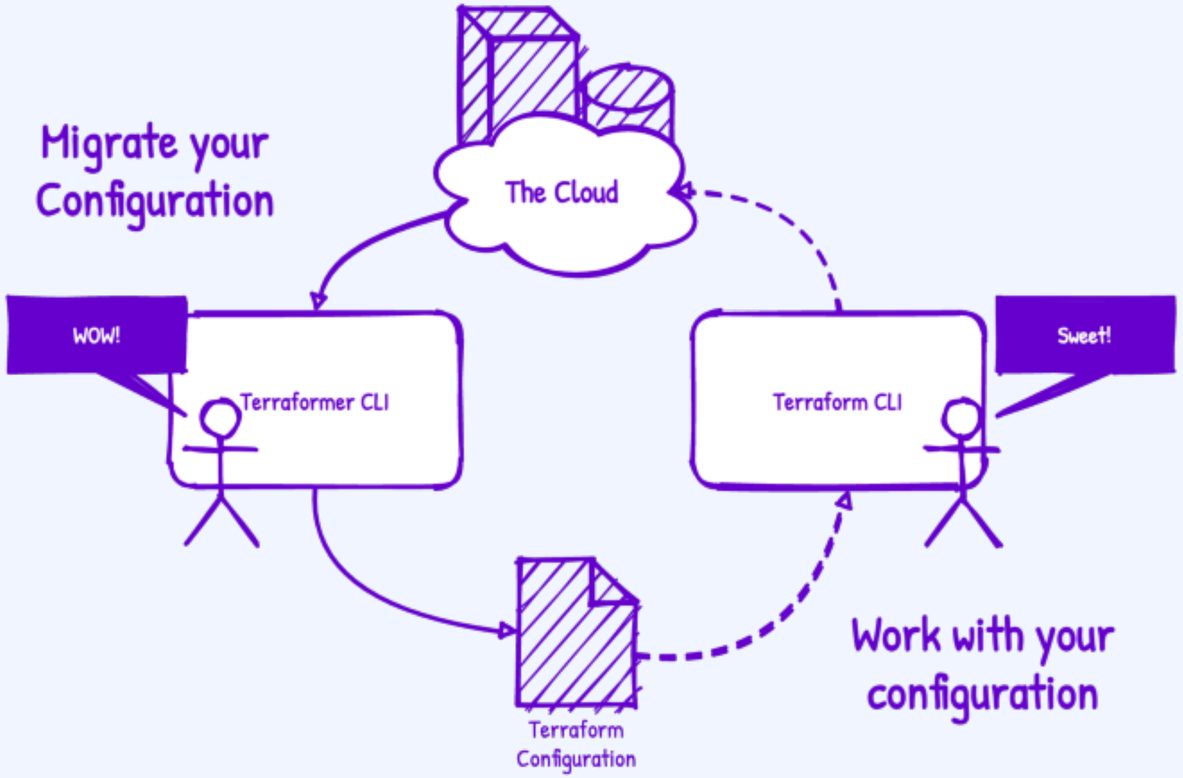


Terraform Backend 소개



출처 : https://medium.com/clarusway/how-to-use-s3-backend-with-a-locking-feature-in-terraform-to-collaborate-more-efficiently-fa0ea70cf359

Terraformer 소개



출처 : https://arivictor.medium.com/turn-your-gcp-project-into-terraform-with-terraformer-cli-eeec36cbe0d8

사전 준비사항

1. Terraform으로 EKS까지 프로비저닝 완료
2. 로컬PC에서 Kubectl로 EKS 접속 가능한 수준까지 준비
3. 로컬PC에 Terraformer 설치

실습 내용

순서

1. AWS내 Terraform Backend 저장소 생성
2. Terraformer를 활용한 기존 EKS Autoscaling 자원 대상 IaC 코드 추출
3. 추출 IaC 코드를 활용한 2번째 EKS NodeGroup 생성 및 확인
4. Terraformer를 활용한 기존 AWS 네트워크 자원 대상 IaC 코드 추출
5. 추출 IaC 코드를 활용한 Terraform 상태 및 형상 파일에 저장, 관리

실습 예제코드 경로

Chapter02 > Ch02_06-terraformer-eks

사전 준비 - Terraformer 설치 방법 (Mac OS 및 Windows)

Mac OS 설치 명령어

$ brew install terraformer

Windows 설치 명령어

1. 다음 URL에서 exe 파일 다운로드

https://github.com/GoogleCloudPlatform/terraformer/releases

2. 다운로드한 exe 파일을 실행가능 위치에 두고 실행

Terraformer 설치 방법 – Linux

설치 명령어

```
$ export PROVIDER={all,google,aws,kubernetes}
$ curl -LO
https://github.com/GoogleCloudPlatform/terraformer/releases/download/$(curl -s
https://api.github.com/repos/GoogleCloudPlatform/terraformer/releases/latest |
grep tag_name | cut -d '"' -f 4)/terraformer-${PROVIDER}-linux-amd64
$ chmod +x terraformer-${PROVIDER}-linux-amd64
$ sudo mv terraformer-${PROVIDER}-linux-amd64 /usr/local/bin/terraformer
```

1. AWS내 Terraform Backend 저장소 생성

1. 로컬에 있는 Terraform 상태파일을 S3 Backend Bucket으로 전송

$ aws s3 cp terraform.state s3://<S3 Backend Bucket명>/<저장할 파일명>

2. Terraform 상태파일 내 오브젝트 현황 확인

$ terraform state list

3. Terraformer로 AWS 추출 가능한 대상 확인

- 참고 링크 :

https://github.com/GoogleCloudPlatform/terraformer/blob/master/docs/aws.md

2. Terraformer를 활용한 기존 EKS Autoscaling 자원 대상 IaC 코드 추출

1. Terraformer를 활용한 기존 EKS Autoscaling 자원 대상 IaC 코드 추출

$ terraformer import aws --regions=<리전명> --resources=<자원명> --path-pattern=<추출한 파일 저장 디렉토리명>

2. 추출된 Terraform 상태파일 내 오브젝트 현황 확인

$ terraform state list

3. 추출 Terraform 상태파일을 기존 Terraform Backend 상태파일에 Import 방법

$ terraform state mv -state-out=<기존 Terraform Backend 상태파일 저장 경로> <추출 Terraform Object명> <Import되서 저장될 Terraform Object명>

3. 추출 IaC 코드를 활용한 2번째 EKS NodeGroup 생성 및 확인

1. 로컬에 있는 Terraform 상태파일을 S3 Backend Bucket으로 업로드

$ aws s3 cp terraform.state s3://<S3 Backend Bucket명>/<업로드할 파일명>

2. 초기화 전 AWS Provider로 전환

$ terraform state replace-provider -auto-approve registry.terraform.io/-/aws hashicorp/aws

3. 프로비저닝을 통한 생성 및 확인

$ terraform init

$ terraform plan

$ terraform apply

4. Terraformer를 활용한 기존 AWS 네트워크 자원 대상 IaC 코드 추출 #1

1. S3 Backend Bucket에 있는 Terraform 상태파일을 로컬로 다운로드

$ aws s3 cp s3://<S3 Backend Bucket명>/<저장된 파일명> <로컬의 다운로드 위치>

2. Terraformer를 활용한 기존 EKS Autoscaling 자원 대상 IaC 코드 추출

$ terraformer import aws --regions=<리전명> --resources=<자원명> --path-pattern=<추출한 파일 저장 디렉토리명>

4. Terraformer를 활용한 기존 AWS 네트워크 자원 대상 IaC 코드 추출 #2

3. 추출된 Terraform 상태파일 내 오브젝트 현황 확인

$ terraform state list

4. 추출 Terraform 상태파일을 기존 Terraform Backend 상태파일에 Import 방법

$ terraform state mv -state-out=<기존 Terraform Backend 상태파일 저장 경로> <추출 Terraform Object명> <Import되서 저장될 Terraform Object명>

5. 추출 IaC 코드를 활용한 Terraform 상태 및 형상 파일에 저장, 관리

1. 로컬에 있는 Terraform 상태파일을 S3 Backend Bucket으로 업로드

$ aws s3 cp terraform.state s3://<S3 Backend Bucket명>/<업로드할 파일명>

2. 초기화 전 AWS Provider로 전환

$ terraform state replace-provider -auto-approve registry.terraform.io/-/aws hashicorp/aws

3. 프로비저닝을 통한 생성 및 확인

$ terraform init

$ terraform plan

$ terraform apply