**IaC**

목차

[**0. IaC** 4](#_Toc111130641)

[**1. Vagrant** 5](#_Toc111130642)

[**1.1. 설치 및 초기 작업** 5](#_Toc111130643)

[**1.2. 가상머신 생성/접속/제거** 7](#_Toc111130644)

[**1.3. 포트 포워딩, 쉘** 9](#_Toc111130645)

[**1.4. VM 세부 설정, 폴더 동기화** 10](#_Toc111130646)

[**1.5. VM 동시에 2개 생성** 11](#_Toc111130647)

[**1.5.1. WEB VM 2개 생성** 11](#_Toc111130648)

[**1.5.2 WEB, FTP VM 생성** 13](#_Toc111130649)

[**2. Terraform** 14](#_Toc111130650)

[**2.1. Terraform 기본** 14](#_Toc111130651)

[**2.1.1. Terraform 사용** 14](#_Toc111130652)

[**2.1.2. Terraform 초기화 및 Provider** 16](#_Toc111130653)

[**2.1.3. 키 페어 생성 및 업로드** 17](#_Toc111130654)

[**2.2. VPC** 19](#_Toc111130655)

[**2.2.1. VPC 생성** 19](#_Toc111130656)

[**2.2.2. 서브넷 생성** 20](#_Toc111130657)

[**2.2.3. 인터넷 게이트웨이 생성** 21](#_Toc111130658)

[**2.2.4. 라우팅 테이블 생성** 22](#_Toc111130659)

[**2.2.5. IGW Routing Table Association** 23](#_Toc111130660)

[**2.2.6. NAT 게이트웨이 생성** 24](#_Toc111130661)

[**2.2.7. NGW Routing Table Association** 25](#_Toc111130662)

[**2.2.8. 보안 그룹 생성** 27](#_Toc111130663)

[**2.3. EC2** 29](#_Toc111130664)

[**2.3.1. EC2 인스턴스 생성** 29](#_Toc111130665)

[**2.3.2. AMI 생성** 31](#_Toc111130666)

[**2.4. Load Balancer** 32](#_Toc111130667)

[**2.4.1. Load Balancer 생성** 32](#_Toc111130668)

[**2.4.2. Target Group (Backend)** 33](#_Toc111130669)

[**2.4.3. Listener (Frontend)** 34](#_Toc111130670)

[**2.5. AutoScale** 35](#_Toc111130671)

[**2.5.1. 시작 구성 생성** 35](#_Toc111130672)

[**2.5.2. AutoScaling Group 생성** 36](#_Toc111130673)

[**2.5.3. AutoScaling Attachment** 37](#_Toc111130674)

[**2.6. RDS 생성 및 연동** 38](#_Toc111130675)

# **0. IaC**

**IaC : Infrastructure As Code**

1. Infra를 Code로 관리하는 기술

2. 1000대의 서버 세팅 - 완벽하게 똑같은 설정임을 보장할 방법이 없다.

3. Code로 관리한다면 - 멱등성을 보장할 수 있다.

**배포 관리 - 설치**

1. Terrafrom : Platform에 종속되지 않음

2. Vagrant : Private Cloud에서

**구성 관리 - 설치 후 설정 작업**

1. puppet : redhat에서

2. chef : 설정이 힘들다.

3. Ansible : 배우기 쉽다.

**Cloud Formation**

1. AWS에서만 사용하는 IaC

# **1. Vagrant**

## **1.1. 설치 및 초기 작업**

www.vagrantup.com/downloads 에서 Vagrant를 다운로드 받아 설치할 수 있다.

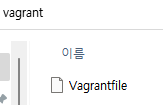
텍스트이(가) 표시된 사진

자동 생성된 설명

Vagrantfile을 생성할 폴더 -> Shift + 마우스 우클릭 -> 여기에 PowerShell 창 열기

텍스트이(가) 표시된 사진

자동 생성된 설명



vagrant init -> Vagrantfile이 생성된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Vargrantfile은 VS Code로 열어 수정해준다.

텍스트, 스크린샷, 점수판이(가) 표시된 사진

자동 생성된 설명 텍스트, 실외, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Vagrant 확장팩과 한글 언어팩이 있으면 유용하니 설치해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

터미널을 vagrant 폴더로 이동하고 vagrant-vbguest 플러그인 설치 (미설치시 GuestAddition 문제 발생)

## **1.2. 가상머신 생성/접속/제거**

**1.2.1. 가상머신 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명

Vagrantfile 수정

vm.box -> Base Image, Vagrant Cloud(app.vagrantup.com)에서 다운로드

vbguest.installer\_options = { allow\_kernel\_upgrade: true } -> vbguest 커널 업데이트 혹은 최신 버전 설치



vagrant up -> Vagrant 실행

텍스트이(가) 표시된 사진

자동 생성된 설명

VirtualBox에 가상머신이 생성되었다.

**1.2.2. SSH 접속**



vagrant ssh 명령어로 가상머신에 접속

텍스트이(가) 표시된 사진

자동 생성된 설명

sudo su - 명령어로 관리자 권한을 얻어 net-tools를 설치

텍스트이(가) 표시된 사진

자동 생성된 설명

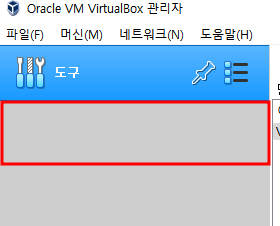
아무런 네트워크 설정을 하지 않은 가상머신은 다음과 같은 네트워크 정보를 가지게 된다.

**1.2.3. 가상머신 제거**

텍스트이(가) 표시된 사진

자동 생성된 설명

vagrant destroy 명령어로 가상머신 제거



VirtualBox에 가상머신이 제거되었다.

## **1.3. 포트 포워딩, 쉘**

텍스트, 스크린샷, 화면, 검은색이(가) 표시된 사진

자동 생성된 설명

다음 명령을 추가하여 포트 포워딩과 쉘을 통한 명령어를 사용

auto\_correct: true -> 포트 충돌 오류를 막기 위해 입력해준다.

inline: “yum install -y httpd” 와 같은 방식으로 사용 가능 / inline 대신 path로 경로를 지정할 수 있다.



Vagrant 실행

텍스트이(가) 표시된 사진

자동 생성된 설명

WEB 서비스 실행 확인

## **1.4. VM 세부 설정, 폴더 동기화**

텍스트이(가) 표시된 사진

자동 생성된 설명

다음 명령을 추가하여 VM 세부 설정과 폴더 동기화

텍스트이(가) 표시된 사진

자동 생성된 설명 

동기화 폴더에 출력 페이지 HTML 파일(index.html) 준비 -> Vagrant 실행

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

VM 세부 정보와 웹사이트 확인

## **1.5. VM 동시에 2개 생성**

### **1.5.1. WEB VM 2개 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명

2개의 VM을 생성하는 코드 -> Host 이름, VM 이름, 포트가 겹치면 안 된다.

end를 입력할 위치를 잘 정해줘야 VM이 순서와 설정에 맞게 잘 생성된다.

provision “shell”에 path를 사용하여 스크립트 파일을 통해서 Web 설정을 하도록 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

스크립트 파일



Vagrant 실행



텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

VM 생성, 웹사이트 확인

### **1.5.2 WEB, FTP VM 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명

Vagrantfile

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

스크립트 파일



Vagrant 실행

텍스트이(가) 표시된 사진

자동 생성된 설명

FTP 연결 확인

# **2. Terraform**

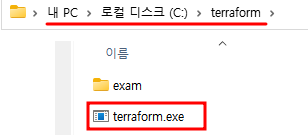
## **2.1. Terraform 기본**

Flatform에 종속되지 않음, 선언적인 코딩 툴

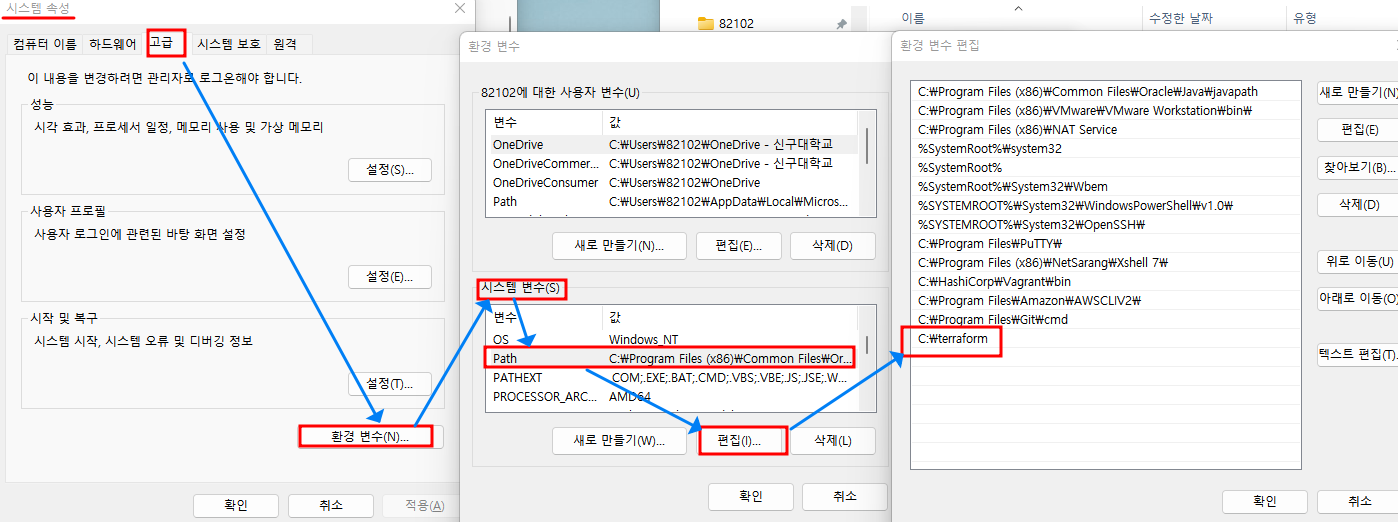
resource -> 새로운 자원 생성 / data source -> 기존 자원 가져오기

registry.terraform.io에서 Terraform 사용 방법, 코드 등의 정보를 얻을 수 있다.

### **2.1.1. Terraform 사용**



terraform.io에서 terraform.exe를 다운로드 받고 특정 디렉터리에 저장한다.



terraform.exe가 저장된 디렉터리를 시스템 속성의 환경변수 시스템 변수 Path에 넣어준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

어느 위치에서나 terraform을 사용할 수 있다.

(Terraform을 사용하기 앞서 AWS에서 사용할 것이기 때문에 AWS CLI에서 사용자 접속을 해준다.)

### **2.1.2. Terraform 초기화 및 Provider**

텍스트, 스크린샷, 모니터, 화면이(가) 표시된 사진

자동 생성된 설명

VS Code에서 HashiCorp Terraform 설치하고 Terraform을 사용할 폴더를 열어 준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

터미널에서 terraform init 명령어로 초기화 -> 디렉터리에 Terraform configuration file이 없어서 실패

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

main.tf 파일을 만들어 다음 명령어를 입력하고 다시 terraform init 명령어를 사용 -> 초기화 성공

### **2.1.3. 키 페어 생성 및 업로드**

텍스트이(가) 표시된 사진

자동 생성된 설명

ssh-keygen으로 SSH 키 페어 생성

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

jhkim-key 이름으로 .ssh 폴더에 있는 jhkim\_key.pub 키 업로드

텍스트이(가) 표시된 사진

자동 생성된 설명

터미널에서 terraform plan 명령어로 테스트



텍스트이(가) 표시된 사진

자동 생성된 설명

터미널에서 terraform apply 명령어로 적용 -> 1 added (키 페어 1)



terraform apply 명령어에 --auto-approve 옵션을 사용하여 자동 승인 가능 (아무것도 묻지 않는다.)

테이블이(가) 표시된 사진

자동 생성된 설명

키가 업로드 되었다.



terraform apply 명령어에 -destroy 옵션을 사용하여 제거 가능

텍스트이(가) 표시된 사진

자동 생성된 설명

키가 제거되었다.

## **2.2. VPC**

### **2.2.1. VPC 생성**

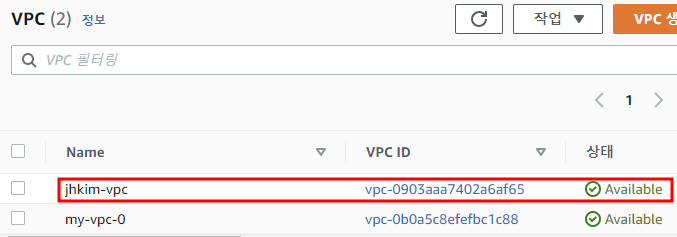
텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

CIDR Block 10.0.0.0/16에 jhkim-vpc라는 이름의 VPC 생성



terraform apply



VPC가 생성되었다.



terraform apply -destroy

텍스트이(가) 표시된 사진

자동 생성된 설명

VPC가 제거되었다.

### **2.2.2. 서브넷 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

jhkim\_vpc에 CIDR Block 10.0.0.0/24, 가용 영역 ap-northeast-2a, 이름 jhkim-weba인 서브넷 생성

jhkim\_vpc에 CIDR Block 10.0.1.0/24, 가용 영역 ap-northeast-2c, 이름 jhkim-webc인 서브넷 생성

jhkim\_vpc에 CIDR Block 10.0.2.0/24, 가용 영역 ap-northeast-2a, 이름 jhkim-wasa인 서브넷 생성

jhkim\_vpc에 CIDR Block 10.0.3.0/24, 가용 영역 ap-northeast-2c, 이름 jhkim-wasc인 서브넷 생성

jhkim\_vpc에 CIDR Block 10.0.4.0/24, 가용 영역 ap-northeast-2a, 이름 jhkim-dba인 서브넷 생성

jhkim\_vpc에 CIDR Block 10.0.5.0/24, 가용 영역 ap-northeast-2c, 이름 jhkim-dbc인 서브넷 생성





terraform apply -> 6 added (이미 생성된 키 페어, VPC는 다시 생성되지 않고 서브넷 6개만 생성된다.)

테이블이(가) 표시된 사진

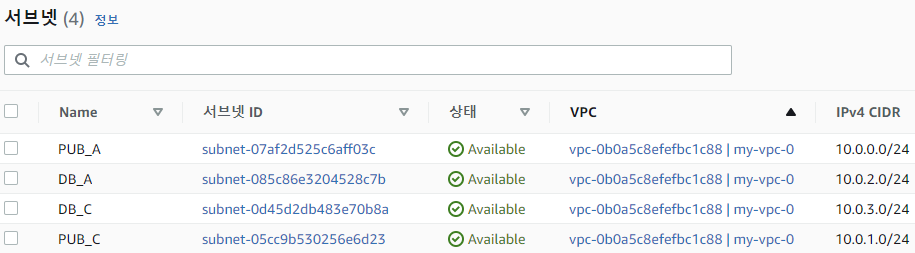
자동 생성된 설명

서브넷이 생성되었다.





terraform apply -destroy -> 8 destroyed (키 페어 1, VPC 1, 서브넷 6)



서브넷이 제거되었다.

### **2.2.3. 인터넷 게이트웨이 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명

jhkim-vpc에 jhkim-igw라는 이름의 인터넷 게이트웨이 생성





terraform apply -> 9 added (키 페어 1, VPC 1, 서브넷 6, 인터넷 게이트웨이 1)

테이블이(가) 표시된 사진

자동 생성된 설명

인터넷 게이트웨이가 생성되었다.

### **2.2.4. 라우팅 테이블 생성**

텍스트이(가) 표시된 사진

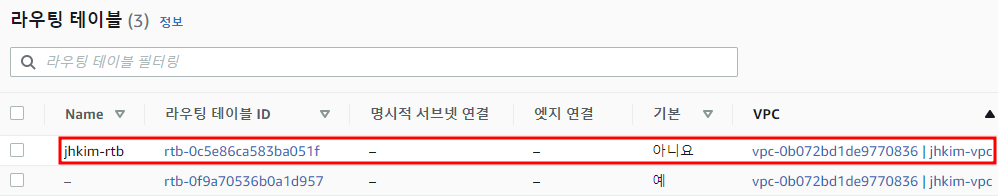
자동 생성된 설명

jhkim-vpc에 ‘0.0.0.0/0 -> jhkim-igw’ 라우팅 규칙이 등록된 라우팅 테이블 jhkim-rtb 생성





terraform apply -> 1 added (라우팅 테이블 1)



텍스트이(가) 표시된 사진

자동 생성된 설명

‘0.0.0.0/0 -> jhkim-igw’ 라우팅 규칙이 등록된 라우팅 테이블이 생성되었다.

### **2.2.5. IGW Routing Table Association**

텍스트이(가) 표시된 사진

자동 생성된 설명

jhkim\_weba, jhkim\_webc 서브넷을 라우팅 테이블에 Association





terraform apply -> 2 added (jhkim\_weba, jhkim\_webc Subnet Association)

텍스트이(가) 표시된 사진

자동 생성된 설명

라우팅 테이블에 jhkim-weba, jhkim-webc가 명시적 서브넷 연결되었다.

### **2.2.6. NAT 게이트웨이 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명

탄력적 IP 생성, 탄력적 IP를 연결한 jhkim-ngw라는 이름의 NAT 게이트웨이를 jhkim-weba 서브넷에 생성





terraform apply -> 2 added (탄력적 IP 1, NAT 게이트웨이 1)

테이블이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

탄력적 IP와 탄력적 IP가 연결된 NAT 게이트웨이가 생성되었다.

### **2.2.7. NGW Routing Table Association**

텍스트, 모니터, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

jhkim-vpc에 ‘0.0.0.0/0 -> jhkim-ngw’ 라우팅 규칙이 등록된 라우팅 테이블 jhkim-nrtb 생성





terraform apply -> 1 added (라우팅 테이블 1)

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

‘0.0.0.0/0 -> jhkim-ngw’ 라우팅 규칙이 등록된 라우팅 테이블 jhkim-nrtb가 생성되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

jhkim\_wasa, jhkim\_wasc, jhkim-dba, jhkim-dbc 서브넷을 라우팅 테이블 jhkim-nrtb에 Association





terraform apply -> 4 added (Subnet Association 4), 1 chacnged (nrtb 1)

텍스트이(가) 표시된 사진

자동 생성된 설명

라우팅 테이블에 jhkim-wasa, jhkim-wasc, jhkim-dba, jhkim-dbc가 명시적 서브넷 연결되었다.

### **2.2.8. 보안 그룹 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 화면이(가) 표시된 사진

자동 생성된 설명

jhkim-vpc에 jhkim-sg 보안 그룹 생성 -> 22 TCP Port 모든 IP 개방 규칙 추가 (SSH)

80 TCP Port 모든 IP 개방 인바운드 규칙 추가 (HTTP)

3306 TCP Port 모든 IP 개방 인바운드 규칙 추가 (MySQL)

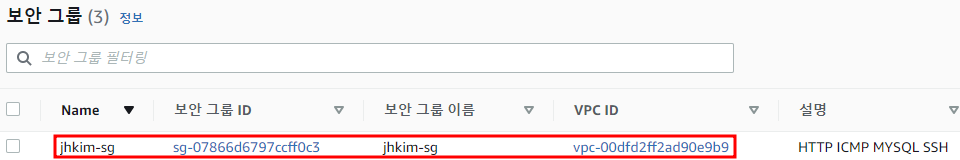
ICMP Protocal 모든 IP 개방 인바운드 규칙 추가 (ICMP)

모든 IP 개방 아웃바운드 규칙 추가





terraform apply -> 20 added



테이블이(가) 표시된 사진

자동 생성된 설명

보안 그룹이 생성되었다.

## **2.3. EC2**

### **2.3.1. EC2 인스턴스 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

AWS에서 amzn2 AMI 가져오기, AMI 기반으로 인스턴스 유형 t2.micro, 보안 그룹 jhkim-sg, 가용 영역 ap-northeast-2a, 사설 IP 10.0.0.10, 서브넷 jhkim-weba, 공인 IP 할당 활성화, userdata install.sh 파일, 이름 jhkim-web인 EC2 인스턴스 생성, 인스턴스의 고정 IP 주소 출력

텍스트이(가) 표시된 사진

자동 생성된 설명

userdata 스크립트 파일 -> 웹 서버 생성



텍스트이(가) 표시된 사진

자동 생성된 설명

terraform apply -> 1 added (인스턴스 1), 1 changed (nrtb 1), output (EC2 인스턴스 Public IP 주소)

테이블이(가) 표시된 사진

자동 생성된 설명

인스턴스가 생성되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

출력된 IP 주소로 웹 접속

### **2.3.2. AMI 생성**

텍스트, 화면, 검은색, 스크린샷이(가) 표시된 사진

자동 생성된 설명

AMI 생성 -> jhkim-ami 이름, jhkim-web 소스 인스턴스

depends\_on: 리소스와 모듈간 종속성 지정





terraform apply -> 1 added (AMI 1), 1 changed (nrtb 1)

텍스트이(가) 표시된 사진

자동 생성된 설명

AMI가 생성되었다.

## **2.4. Load Balancer**

### **2.4.1. Load Balancer 생성**

텍스트, 모니터, 스크린샷, 검은색이(가) 표시된 사진

자동 생성된 설명

Application Load Balancer 생성 -> jhkim-alb 이름, jhkim-weba, jhkim-webc 서브넷, jhkim-sg 보안 그룹

Load Balancer의 DNS 이름 출력



텍스트이(가) 표시된 사진

자동 생성된 설명

terraform apply -> 1 added (로드 밸런서 1), 1 changed (nrtb 1), output (EC2 인스턴스 Public IP 주소, Load Balancer DNS 이름)

텍스트이(가) 표시된 사진

자동 생성된 설명

Application Load Balancer가 생성되었다.

### **2.4.2. Target Group (Backend)**

텍스트이(가) 표시된 사진

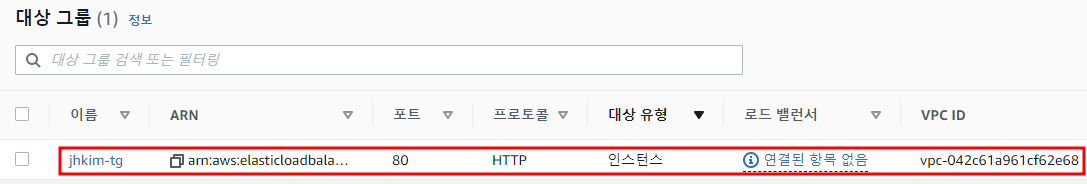
자동 생성된 설명

Target Group 생성 -> jhkim-tg 이름, 80 Port, HTTP Protocol, jhkim-vpc VPC, Health Check





terraform apply -> 1 added (대상 그룹 1), 1 changed (nrtb 1)



대상 그룹이 생성되었다.

### **2.4.3. Listener (Frontend)**

텍스트, 모니터, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

80 Port, HTTP Protocol, jhkim-tg으로 forward 작업을 수행하는 Listener 생성





1 added (Listener 1), 1 changed (nrtb 1)

텍스트이(가) 표시된 사진

자동 생성된 설명

대상 그룹에 로드 밸런서가 연결되었다.

## **2.5. AutoScale**

### **2.5.1. 시작 구성 생성**

텍스트이(가) 표시된 사진

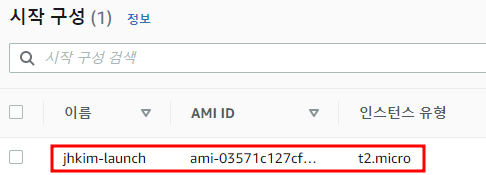
자동 생성된 설명

시작 구성 생성 -> jhkim-launch 이름, jhkim-ami 소스 AMI, t2.micro 인스턴스 타입, jhkim-sg 보안 그룹, admin\_role IAM 인스턴스 프로파일, jhkim-key 키 페어, user\_data에 혹시나 httpd 서비스가 enable이 아닐 때를 대비한 스크립트를 작성하였다.





terraform apply -> 1 added (시작 구성 1), 1 changed (nrtb 1)



시작 구성이 생성되었다.

### **2.5.2. AutoScaling Group 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명

Placement Group 생성 -> jhkim-pg 이름, Cluster 전략

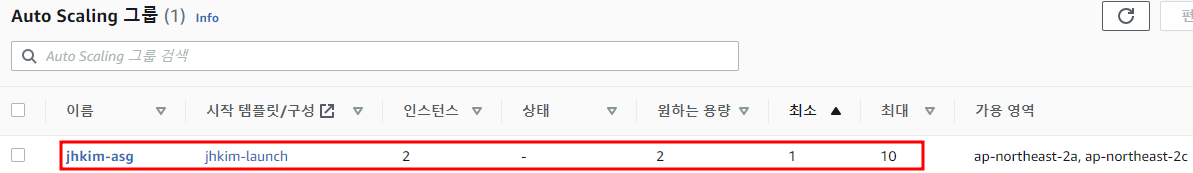
AutoScaling Group 생성 -> jhkim-asg 이름, 최소 1 최대 2 기본 2 사이즈, 상태 유예 기간 60초,

jhkim-launch 시작 구성, jhkim-weba/jhkim-webc VPC 영역

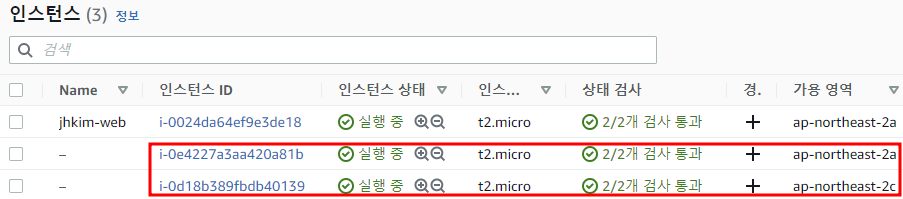




terraform apply -> 2 added (Placement Group 1, AutoScaling Group 1), 1 changed (nrtb 1)



Auto Scaling 그룹이 생성되었다.



Auto Scaling을 통해서 2개의 인스턴스 생성되었다.

### **2.5.3. AutoScaling Attachment**

텍스트이(가) 표시된 사진

자동 생성된 설명

Auto Scaling 대상 그룹에 Attachment



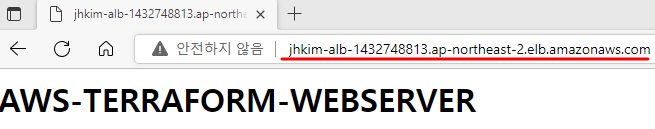


terraform apply -> 1 added (attachment 1), 1 changed (nrtb 1)

텍스트이(가) 표시된 사진

자동 생성된 설명

대상 그룹에 Auto Scaling 그룹이 연결되어 대상 인스턴스를 확인할 수 있다.



로드 밸런서 DNS 이름으로 접근

## **2.6. RDS 생성 및 연동**

인스턴스에 RDS를 연동하기 위해 모두 지우고 다시 생성하도록 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

서브넷 그룹 생성 -> jhkim-dba, jhkim\_dbc 서브넷

RDS 생성 -> mydb 식별자, root 관리자 이름, It12345! 패스워드

텍스트이(가) 표시된 사진

자동 생성된 설명

인스턴스 userdata로 사용한 인스턴스 생성 스크립트(install.sh) 수정

wordpress와 php를 설치하여 RDS와 연동 -> localhost는 RDS를 미리 생성하여 엔드포인트를 복사하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

albtg.tf 수정 -> 로드 밸런스 대상 그룹 생성 시 health check 경로 파일 수정



terraform apply

텍스트이(가) 표시된 사진

자동 생성된 설명

RDS가 생성되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

인스턴스에 RDS가 연동되어 WordPress 접속 성공