

1. 테라폼 환경 설정

- 강사님 제공 파일로 vagrant 하기

```
O:WTestWT1>vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
=> default: Box 'ubuntu/jammy64' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
=> default: Loading metadata for box 'ubuntu/jammy64'
    default: URL: https://vagrantcloud.com/api/v2/vagrant/ubuntu/jammy64
=> default: Adding box 'ubuntu/jammy64' (v20240403.0.0) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/ubuntu/boxes/jammy64/versions/20240403.0.0/providers/virtualbox/unknown/vagrant.box
Download redirected to host: cloud-images.ubuntu.com
    default:
=> default: Successfully added box 'ubuntu/jammy64' (v20240403.0.0) for 'virtualbox'!
=> default: Preparing master VM for linked clones...
    default: This is a one time operation. Once the master VM is prepared,
    default: it will be used as a base for linked clones, making the creation
    default: of new VMs take milliseconds on a modern system.
=> default: Importing base box 'ubuntu/jammy64'...
=> default: Cloning VM...
=> default: Matching MAC address for NAT networking...
=> default: Checking if box 'ubuntu/jammy64' version '20240403.0.0' is up to date...
=> default: Setting the name of the VM: T1_default_1713250301891_20566
=> default: Clearing any previously set network interfaces...
```

- 테라폼 다운로드

```
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/hashicorp-archive-keyring.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/hashicorp.list
```

```
sudo apt update && sudo apt install terraform
```

```
vagrant@ubuntu-jammy:~$ terraform -v
Terraform v1.8.0
on linux_amd64
vagrant@ubuntu-jammy:~$
```

- 현재 셸에 대한 Terraform 자동 완성을 활성화하기

```
terraform -install-autocomplete
```

- 다음 명령어 입력

```
sudo -i
```

```
cat ~/.bashrc # .bashrc 파일 : Bash 셸에서 사용자 정의 환경 설정을 포함하는 파일
```

IAM > 사용자 > access > 액세스 키 만들기

1단계
액세스 키 보편 사례 대안

보편 개성별 액세스 키와 같은 일반적인 자격 증명들을 사용하지 마세요. 다음과 같은 사용 사례와 대안을 고려하십시오.

- Common Command Line Interface (CLI)**
AWS CLI를 사용하여 AWS 명령어 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☐ **호출 코드**
호출 가능 환경의 애플리케이션 코드로 사용하여 AWS 명령어 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☐ **AWS 통합된 서비스에서 실행하는 애플리케이션**
Amazon EC2, Amazon ECS 또는 AWS Lambda의 일부 AWS Lambda로 실행되는 서비스에서 실행되는 애플리케이션 코드로 사용하여 AWS 명령어 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☐ **서드 파티 서비스**
AWS 리소스를 모니터링 또는 관리하는 서드 파티 애플리케이션으로 서비스의 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☐ **AWS 외부에서 실행되는 애플리케이션**
이 액세스 키를 사용하여 AWS 리소스에 액세스하여 하는 AWS 로보틱스 엔지니어링 팀은 기타 애플리케이션에서 실행 중인 리소스를 인증할 것입니다.
- ☐ **기타**
귀하를 사용 사례와 여기에 나열된 일치 없습니다.

생성 결과

ⓘ 액세스 키 생성됨
지금에 아니면 비밀 액세스 키를 보거나 다운로드할 수 없습니다. 나중에 복구할 수 없습니다. 하지만 언제든지 새 액세스 키를 생성할 수 있습니다.

IAM > 사용자 > mzcroot > 액세스 키 만들기


1단계
액세스 키 모범 사례 및 태연

2단계 - 선택 사항
설명 태그 설정

3단계
액세스 키 검색

액세스 키 검색 정보

액세스 키
본실하거나 잃어버린 비밀 액세스 키는 검색할 수 없습니다. 대신 새 액세스 키를 생성하고 이전 키를 비활성화합니다.

액세스 키	비밀 액세스 키
	***** 표시

액세스 키 (1) 액세스 키 만들기

액세스 키를 사용하여 AWS CLI, AWS Tools for PowerShell, AWS SDK 또는 직접 AWS API 호출을 통해 AWS에 프로그래밍 방식 호출을 전송합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 가질 수 있습니다. 자세히 알아보기

설명	상태
laC	Active
마지막 사용	생성됨
없음	1분 전
마지막으로 사용한 리전	마지막으로 사용한 서비스
N/A	N/A

작업 ▼

- 액세스 키는 알려지면 X
- 이를 깃헙 등에 올리면 나중에 다른 사용자가 접근할 수 있음

3. AWS 프로비저닝하기 위한 최소 권한 설정하기

- cli에서 다음 설정하기

```
apt update
```

```
apt install unzip -y
```

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

```
unzip awscliv2.zip
```

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

```
aws --version
```

```
/usr/local/bin/aws --version
```

```
root@ubuntu-jammy:~# /usr/local/bin/aws --version  
aws-cli/2.15.38 Python/3.11.8 Linux/5.15.0-102-generic exe/x86_64.ubuntu.22 prompt/off  
root@ubuntu-jammy:~#
```

```
aws configure list
```

- 아무것도 뜨지 않음

```

root@ubuntu-jammy:~# aws configure list
      Name                               Value                               Type    Location
      ----                               -
      profile                            <not set>                          None     None
      access_key                         <not set>                          None     None
      secret_key                         <not set>                          None     None
      region                             <not set>                          None     None

```

aws configure

- 아까 액세스키 만들때 csv 파일 다운 받은게 있는데 그 key id랑 access key 복붙하기

```

root@ubuntu-jammy:~# aws configure
AWS Access Key ID [None]: 
AWS Secret Access Key [None]: 
Default region name [None]: ap-northeast-2
Default output format [None]: 
root@ubuntu-jammy:~# 
root@ubuntu-jammy:~# aws configure list
      Name                               Value                               Type    Location
      ----                               -
      profile                            <not set>                          None     None
      access_key                        ***** shared-credentials-file
      secret_key                        ***** shared-credentials-file
      region                             ap-northeast-2                    config-file  ~/.aws/config
root@ubuntu-jammy:~# 

```

- 명령 완성 활성화하기

```
complete -C '/usr/local/bin/aws_completer' aws
```

- 프로비저닝을 모니터링하기

```
apt-get install tree jq watch
```

- default에 대한 정보 확인하기

```
aws ec2 describe-vpcs --filter 'Name=isDefault,Values=true' | jq
```

```

root@ubuntu-jammy:~# aws ec2 describe-vpcs --filter 'Name=isDefault,Values=true' | jq
{
  "Vpcs": [
    {
      "CidrBlock": "172.31.0.0/16",
      "DhcpOptionsId": "dopt-09410619a83ee66c5",
      "State": "available",
      "VpcId": "vpc-038e825f65ac34b04",
      "OwnerId": "710665100022",
      "InstanceTenancy": "default",
      "CidrBlockAssociationSet": [
        {
          "AssociationId": "vpc-cidr-assoc-0aee472e2633c86f2",
          "CidrBlock": "172.31.0.0/16",
          "CidrBlockState": {
            "State": "associated"
          }
        }
      ]
    }
  ],
  "IsDefault": true
}
root@ubuntu-jammy:~# 

```

- 이는 access 권한이 있기 때문에 접근가능

```
aws ec2 describe-vpcs --filter 'Name=isDefault,Values=true' | jq '.Vpcs[0].VpcId'
```

```
root@ubuntu-jammy:~# aws ec2 describe-vpcs --filter 'Name=isDefault,Values=true' | jq '.Vpcs[0].VpcId'
"vpc-038e825f65ac34b04"
```

```
aws ec2 describe-vpcs --filter 'Name=vpc-id,Values=vpc-038e825f65ac34b04' --
output table
```

```

root@ubuntu-jammy:~# aws ec2 describe-vpcs --filter 'Name=vpc-id,Values=vpc-038e825f65ac34b04' --output table
+-----+-----+-----+-----+-----+-----+-----+
|                                     DescribeVpcs                                     |
+-----+-----+-----+-----+-----+-----+-----+
|                                     Vpcs                                     |
+-----+-----+-----+-----+-----+-----+-----+
| CidrBlock | DhcpOptionsId | InstanceTenancy | IsDefault | OwnerId | State | VpcId |
+-----+-----+-----+-----+-----+-----+-----+
| 172.31.0.0/16 | dopt-09410619a83ee66c5 | default | True | 710665100022 | available | vpc-038e825f65ac34b04 |
+-----+-----+-----+-----+-----+-----+-----+
|                                     CidrBlockAssociationSet                                     |
+-----+-----+-----+-----+-----+-----+-----+
|                                     AssociationId                                     | CidrBlock |
+-----+-----+-----+-----+-----+-----+-----+
| vpc-cidr-assoc-0aee472e2633c86f2 | 172.31.0.0/16 |
+-----+-----+-----+-----+-----+-----+-----+
|                                     CidrBlockState                                     |
+-----+-----+-----+-----+-----+-----+-----+
| State | associated |
+-----+-----+-----+-----+-----+-----+-----+
root@ubuntu-jammy:~#

```

⇒ 여기까지 사전 준비 과정임

4. 테라폼 통해 프로비저닝되는지 확인하기

실습 2

- AWS의 ubuntu AMI 사용할 것임
- 코드 리뷰) user data 사용함
- 외부로 접근했을 때 8080로 포트포워딩

```
# Terraform 코드로 Webserver 구성
cat <<EOT > main.tf
provider "aws" {
  region = "ap-northeast-2"
}

resource "aws_instance" "example" {
  ami           = "ami-09a7535106fbd42d5"
  instance_type = "t2.micro"

  user_data = <<-EOF
    #!/bin/bash
    echo "Hello, MZC-CLOUD" > index.html
    nohup busybox httpd -f -p 8080 &
  EOF

  tags = {
    Name = "MZC-Cloud-101"
  }
}
EOT
```

terraform init

```
vagrant@ubuntu-jammy:~/Terraform$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.45.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

terraform plan

terraform apply

```

aws_instance.example: Still creating... [4m33s elapsed]
aws_instance.example: Still creating... [4m43s elapsed]
aws_instance.example: Creation complete after 4m47s [id=i-06c2c5dd76f19634b]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
vagrant@ubuntu-jammy:~/Terraform$

```

인스턴스 (1) 정보

인스턴스를 속성 또는 (case-sensitive) 태그로 찾기

모든 상태

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가용 영역	퍼블릭 IPv4 DNS	퍼블릭 IPv4 ...
MZC-Cloud-101	i-06c2c5dd76f19634b	실행 중	t2.micro	초기화	경보 보기	ap-northeast-2a	ec2-3-38-150-143.ap-n...	3.38.150.143

```

MZC-Cloud-101 3.38.150.143 running
MZC-Cloud-101 3.38.150.143 running
MZC-Cloud-101 3.38.150.143 running
MZC-Cloud-101 3.38.150.143 running

```

EC2 퍼블릭 IP로 PIP 설정하기

PIP={EC2 퍼블릭 IP}

```

root@ubuntu-jammy:~# echo $PIP
3.38.150.143

```

#웹접근

```

while true; do curl --connect-timeout 1 http://$PIP:8080/ ; echo "-----
-----"; date; sleep 1; done

```

=>하지만 안됨. 외부에 접근할 수 있는 sg 설정 안돼있어서 그럼

인스턴스 (1/2) 정보

인스턴스를 속성 또는 (case-sensitive) 태그로 찾기

모든 상태

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가용 영역	퍼블릭 IPv4 DNS	퍼블릭 IPv4 ...	탄력적 ...
Single-WebSrv	i-06c2c5dd76f19634b	종료됨	t2.micro	-	경보 보기	ap-northeast-2a	-	-	-
MZC-Cloud-101	i-0ac696e6d24009977	실행 중	t2.micro	2/2개 검사 통과...	경보 보기	ap-northeast-2a	ec2-43-201-54-73.ap-n...	43.201.54.73	-

인스턴스: i-0ac696e6d24009977(MZC-Cloud-101)

sg-087245873ca288f04 (default)

인바운드 규칙

이름	보안 그룹 규칙 ID	포트 범위	프로토콜	원본	보안 그룹	설명
-	sgr-0c4c0555e41b60230	전체	전체	sg-087245873ca288f04	default	-

```
cmd (3).txt  cmd2.txt  cmd2 (1).txt  cmd2
파일  편집  보기
ami = "ami-09a7535106fbd42d5"
instance_type = "t2.micro"
vpc_security_group_ids = [aws_security_group.instance.id]

user_data = <<-EOF
#!/bin/bash
echo "Hello, MZC-CLOUD" > index.html
nohup busybox httpd -f -p 8080 &
EOF

tags = {
  Name = "Single-WebSrv"
}

resource "aws_security_group" "instance" { #중속성 가져옴
  name = var.security_group_name

  ingress {
    from_port = 8080
    to_port = 8080
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

variable "security_group_name" {
  description = "The name of the security group"
  type = string
  default = "terraform-example-instance"
}

output "public_ip" {
  value = aws_instance.example.public_ip
  description = "The public IP of the Instance"
}

EOT
```



```

# 웹 서버 접근( 보안그룹정책 적용)
cat <<EOT > main.tf
provider "aws" {
  region = "ap-northeast-2"
}

resource "aws_instance" "example" {
  ami           = "ami-09a7535106fbd42d5"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.instance.id]

  user_data = <<-EOF
    #!/bin/bash
    echo "Hello, MZC-CLOUD" > index.html
    nohup busybox httpd -f -p 8080 &
  EOF

  tags = {
    Name = "Single-WebSrv"
  }
}

resource "aws_security_group" "instance" {
  name = var.security_group_name

  ingress {
    from_port = 8080
    to_port   = 8080
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

variable "security_group_name" {
  description = "The name of the security group"
  type        = string
  default     = "terraform-example-instance"
}

output "public_ip" {
  value     = aws_instance.example.public_ip
  description = "The public IP of the Instance"
}
EOT

```

terraform apply

```

Plan: 1 to add, 1 to change, 0 to destroy.

Changes to Outputs:
  + public_ip = "3.38.150.143"

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_security_group.instance: Creating...
aws_security_group.instance: Creation complete after 2s [id=sg-01352da47b69bf93b]
aws_instance.example: Modifying... [id=i-06c2c5dd76f19634b]
aws_instance.example: Modifications complete after 2s [id=i-06c2c5dd76f19634b]

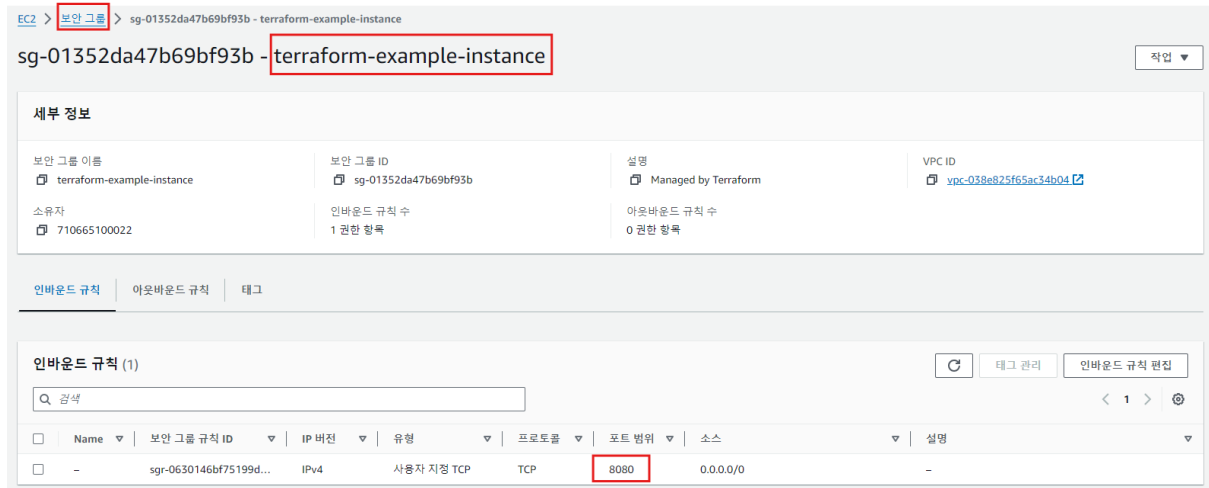
Apply complete! Resources: 1 added, 1 changed, 0 destroyed.

Outputs:

public_ip = "3.38.150.143"
vagrant@ubuntu-jammy:~/Terraform$

```

- 새로운 보안그룹이 생성됨도 확인 가능



웹 접속하면 모니터링하면 Hello, MZC-CLOUD 나와야함 => 에러 뜸

```
root@ubuntu-jammy:~# while true; do curl --connect-timeout 1 http://$PIP:8080/ ; echo
"-----"; date; sleep 1; done
curl: (28) Connection timeout after 1006 ms
-----
Wed Apr 17 02:11:48 UTC 2024
curl: (28) Connection timeout after 1013 ms
-----
Wed Apr 17 02:11:50 UTC 2024
curl: (28) Connection timeout after 1001 ms
-----
Wed Apr 17 02:11:53 UTC 2024
curl: (28) Connection timeout after 1009 ms
-----
```

포트번호 바꿈(또 다른 것도 바꿨음)

웹 접근 포트 번호 변경

-userdata는 서버가 시작되었을 때 필요한 정보임. 서버 재시작할 때만 사용되기 때문에 ~must be replaced라고 나옴

terraform apply

=> 정상적으로 활성화 된 것을 확인할 수 있음

모니터링으로 새롭게 생성된 ip를 PIP 변수로 다시 생성

PIP={새로운퍼블릭 IP}

=>하지만 TIMEOUT 뜸. 이는 아직 서버가 생성 중이라 뜨는 것이기도하고, 모니터링 포드를 바꿔줘야 함

웹 접근

```
while true; do curl --connect-timeout 1 http://$PIP:9090/ ; echo "-----  
-----"; date; sleep 1; done
```

ts

- rm main.tf로 코드 삭제함



=>하지만 아직 인스턴스는 실행 중. 코드는 삭제해도 기존 인스턴스는 계속 실행중임을 확인할 수 있음

Single-WebSrv	3.38.150.143	running

Single-WebSrv	3.38.150.143	running

Single-WebSrv	3.38.150.143	running

Single-WebSrv	3.38.150.143	running

인스턴스 (1) 정보				
<input type="text" value="인스턴스를 속성 또는 (case-sensitive) 태그로 찾기"/> <input type="button" value="모든 상태"/>				
<input type="checkbox"/>	Name 	인스턴스 ID	인스턴스 상태	인스턴스 유형
<input type="checkbox"/>	Single-WebSrv	i-06c2c5dd76f19634b	 실행 중	t2.micro

- 테라폼을 삭제하기

```
terraform destroy -auto-approve
```

=> 아래처럼 인스턴스가 삭제됨을 확인할 수 있음

```
Single-WebSrv  3.38.150.143  running
Single-WebSrv  3.38.150.143  running
Single-WebSrv  3.38.150.143  running
-----
-----
-----
-----
-----
```

인스턴스 (1) 정보

Q 인스턴스를 속성 또는(case-sensitive) 태그로 찾기					모든 상태
<input type="checkbox"/>	Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	
<input type="checkbox"/>	Single-WebSrv	i-06c2c5dd76f19634b	종료 중	t2.micro	

다시 ec2 생성하기

terraform init //프로젝트 초기화, 현재 디렉토리에 있는 Terraform 구성 파일 (일반적으로 "main.tf")을 읽고 필요한 백엔드 구성 및 제공자 플러그인을 다운로드하고 설치

Terraform 코드로 Webserver 구성

terraform plan

```
vagrant@ubuntu-jammy:~/Terraform$ terraform plan

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.example will be created
+ resource "aws_instance" "example" {
  + ami                      = "ami-09a7535106fbd42d5"
  + arn                     = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone        = (known after apply)
  + cpu_core_count           = (known after apply)
  + cpu_threads_per_core     = (known after apply)
  + disable_api_stop         = (known after apply)
  + disable_api_termination  = (known after apply)
  + ebs_optimized            = (known after apply)
  + get_password_data        = false
  + host_id                  = (known after apply)
  + host_resource_group_arn  = (known after apply)
  + iam_instance_profile     = (known after apply)
  + id                       = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle       = (known after apply)
  + instance_state           = (known after apply)
  + instance_type            = "t2.micro"
  + ipv6_address_count       = (known after apply)
  + ipv6_addresses           = (known after apply)
  + key_name                 = (known after apply)
  + monitoring               = (known after apply)
  + outpost_arn              = (known after apply)
  + password_data            = (known after apply)
  + placement_group          = (known after apply)
  + placement_partition_number = (known after apply)
  + primary_network_interface_id = (known after apply)
  + private_dns              = (known after apply)
  + private_ip               = (known after apply)
  + public_dns               = (known after apply)
  + public_ip                = (known after apply)
  + secondary_private_ips    = (known after apply)
  + security_groups          = (known after apply)
  + source_dest_check        = true
  + spot_instance_request_id = (known after apply)
```

terraform apply

```
aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Still creating... [20s elapsed]
aws_instance.example: Still creating... [30s elapsed]
aws_instance.example: Creation complete after 33s [id=i-0ac696e6d24009977]
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
 vagrant@ubuntu-jammy:~/Terraform\$

```
-----
-----
-----
-----
-----
MZC-Cloud-101  43.201.54.73  running
-----
MZC-Cloud-101  43.201.54.73  running
```

인스턴스 (2) 정보

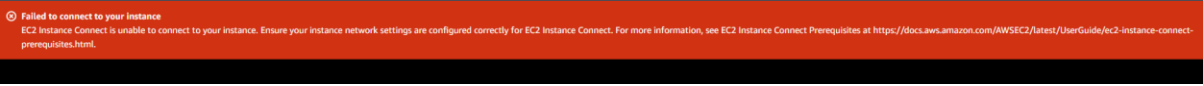
<input type="text"/> 인스턴스를 속성 또는 (case-sensitive) 태그로 찾기					<input type="button" value="모든 상태"/>
<input type="checkbox"/>	Name ↗	인스턴스 ID	인스턴스 상태		인스턴스 유형
<input type="checkbox"/>	Single-WebSrv	i-06c2c5dd76f19634b	⊖ 종료됨	🔍 🔍	t2.micro
<input type="checkbox"/>	MZC-Cloud-101	i-0ac696e6d24009977	🟢 실행 중	🔍 🔍	t2.micro

PIP 변수 변경

```
vagrant@ubuntu-jammy:~$ PIP=43.201.54.73
vagrant@ubuntu-jammy:~$ echo $PIP
43.201.54.73
vagrant@ubuntu-jammy:~$
```

웹 서버 접근(보안그룹정책 적용)

```
-----
MZC-Cloud-101  43.201.54.73  running
-----
MZC-Cloud-101  43.201.54.73  running
-----
MZC-Cloud-101  43.201.54.73  running
-----
Single-WebSrv  43.201.54.73  running
-----
Single-WebSrv  43.201.54.73  running
-----
Single-WebSrv  43.201.54.73  running
-----
Single-WebSrv  43.201.54.73  running
-----
```



접근이 안됨

=> 인바운드 22포트 설정, IGW 생성 후 라우팅테이블에 IGW 연결

VPC > 라우팅 테이블 > rtb-0f7bdd828d59e6a57

rtb-0f7bdd828d59e6a57

세부 정보

라우팅 테이블 ID rtb-0f7bdd828d59e6a57	기본 예	명시적 서브넷 연결 -	엣지 연결 -
VPC vpc-038e825f565ac34b04	소유자 ID 710665100022		

라우팅 | 서브넷 연결 | 엣지 연결 | 라우팅 전파 | 태그

라우팅 (2)

Q 라우팅 필터링

대상	대상	상태	전파됨
0.0.0.0/0	igw-0f528ddd30b1e9a96	활성	아니요
172.31.0.0/16	local	활성	아니요

EC2 > 보안 그룹 > sg-0eb1a7628a607b494 - terraform-example-instance

sg-0eb1a7628a607b494 - terraform-example-instance

세부 정보

보안 그룹 이름 terraform-example-instance	보안 그룹 ID sg-0eb1a7628a607b494	설명 Managed by Terraform
소유자 710665100022	인바운드 규칙 수 3 권한 항목	아웃바운드 규칙 수 0 권한 항목

인바운드 규칙 | 아웃바운드 규칙 | 태그

인바운드 규칙 (3)

Q 검색

	Name	보안 그룹 규칙 ID	IP 버전	유형	프로토콜	포트 범위	소스
<input type="checkbox"/>	-	sgr-055f8cd7fd7761114	IPv4	사용자 지정 TCP	TCP	8080	0.0.0.0/0
<input type="checkbox"/>	-	sgr-03b09d14c5084c6...	IPv4	사용자 지정 TCP	TCP	8000	0.0.0.0/0
<input type="checkbox"/>	-	sgr-035b2b45e37eea7...	IPv4	SSH	TCP	22	0.0.0.0/0

=> iam user에 정책 연결함

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "ec2-instance-connect:SendSSHPublicKey",
7       "Resource": [
8         "arn:aws:ec2:region:account-id:instance/i-0eb40401a5660bac6"
9       ],
10      "Condition": {
11        "StringEquals": {
12          "ec2:osuser": "ubuntu"
13        }
14      }
15    },
16    {
17      "Effect": "Allow",
18      "Action": "ec2:DescribeInstances",
19      "Resource": "*"
20    }
21  ]
22 }

```

IAM > 사용자 > mzcroot

mzcroot 정보

요약

ARN arn:aws:iam::710665100022:user/mzcroot	콘솔 액세스 MFA 없이 활성화됨	엑세스 키 1 AKIA2K5XQ3L3GLJJRHUQ - Active 오늘 사용됨. 21시간 기권.
생성됨 April 16, 2024, 16:45 (UTC+09:00)	마지막 콘솔 로그인 안 함	엑세스 키 2 엑세스 키 만들기

권한 | 그룹 | 태그 (1) | 보안 자격 증명 | 액세스 관리자

권한 정책 (2)

사용자에게 직접 연결된 정책을 통해 또는 그룹을 통해 권한을 정의합니다.

검색 필터링 기준 유형

<input type="checkbox"/> 정책 이름	유형	연결 방식
<input type="checkbox"/> AdministratorAccess	AWS 관리형 - 직무	직접
<input type="checkbox"/> EC2_Instance_Connect_1	고객 관리형	직접

그러면 접근 됨

ap-northeast-2.console.aws.amazon.com/ec2-instance-connect/ssh?region=ap-northeast-2&connType=standard&instanceId=i-0eb40401a5660bac6&osUser=ubuntu&sshPort=22#

NAVER | Keep | YouTube | fliation | MGZ 클래스 | 챗gpt | 도지 | 수업자료(인프런) | notion | notion(mzc) | Qwiklabs | AWS Skill Builder | aws training | AWS | mgz-myt

AWS 서비스 검색 [알트+S]

EC2 VPC RDS CloudFront AWS Glue Athena MSK QuickSight S3

```

Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/pro

System information as of Wed Apr 17 05:05:19 UTC 2024

System load:  0.0          Processes:    99
Usage of /:   20.4% of 7.5GB Users logged in: 0
Memory usage: 20%         IPv4 address for eth0: 172.31.3.76
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

Updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

buntu@ip-172-31-3-76:~$

```

근데 이때 웹접근하니까 됨 => default vpc에 igw 연결이 안되어 있어서 생긴 문제

원랜 기본 igw가 있나?

```
-----  
Wed Apr 17 05:21:33 UTC 2024  
Hello, MZC-CLOUD  
-----  
Wed Apr 17 05:21:34 UTC 2024  
Hello, MZC-CLOUD  
-----  
Wed Apr 17 05:21:35 UTC 2024  
Hello, MZC-CLOUD  
-----  
Wed Apr 17 05:21:36 UTC 2024
```

포트 9090으로 변경하는거 다시하게

실습 3

우분트 서버에 APACHE 서버 설치, 사진&글 출력하게 만들기

#다운이가 카톡으로보내준 코드를 main에 얹어씌우기

=> 이미지를 url로 해서 띄우게, 한문장 띄우게

terraform plan

terraform apply


```

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.example: Destroying... [id=i-0eb40401a5660bac6]
^[A^[[Baws_instance.example: Still destroying... [id=i-0eb40401a5660bac6, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0eb40401a5660bac6, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0eb40401a5660bac6, 30s elapsed]
aws_instance.example: Still destroying... [id=i-0eb40401a5660bac6, 40s elapsed]
aws_instance.example: Still destroying... [id=i-0eb40401a5660bac6, 50s elapsed]
aws_instance.example: Still destroying... [id=i-0eb40401a5660bac6, 1m0s elapsed]
aws_instance.example: Destruction complete after 1m2s
aws_security_group.instance: Modifying... [id=sg-0eb1a7628a607b494]
aws_security_group.instance: Modifications complete after 1s [id=sg-0eb1a7628a607b494]
aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Still creating... [20s elapsed]
aws_instance.example: Still creating... [30s elapsed]
aws_instance.example: Creation complete after 33s [id=i-056de26088f65d2c2]

Apply complete! Resources: 1 added, 1 changed, 1 destroyed.

Outputs:

public_ip = "43.203.210.152"
vaarant@ubuntu-iammv:~/Terraform$

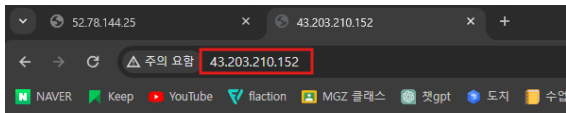
```

인스턴스 (2) 정보

모든 상태 ▼

<input type="checkbox"/>	Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사
<input type="checkbox"/>	Single-WebSrv	i-0eb40401a5660bac6	<div> <div>종료됨</div> <div>🔍 🔍</div> </div>	t2.micro	-
<input type="checkbox"/>	Single-WebSrv	i-056de26088f65d2c2	<div> <div>실행 중</div> <div>🔍 🔍</div> </div>	t2.micro	<div> <div>2/2개 검사 통과.</div> </div>

결과물



MIN KYEONG, HELLO



실습 - s3에 이미지 올려서 띄우기

s3 객체 연동 띄우기 (바탕화면)

위의 코드를 main에 덮어씌우기

terraform plan

terraform apply

```
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

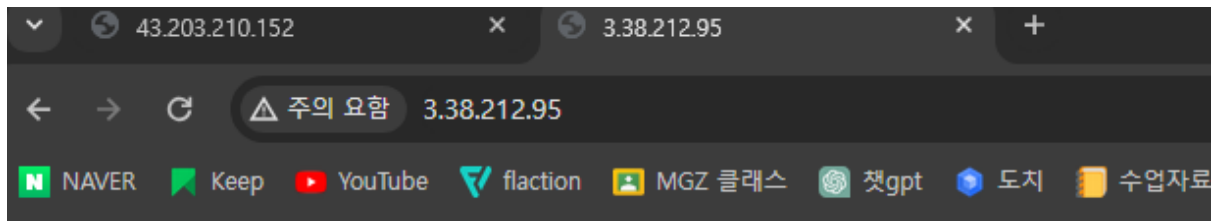
aws_instance.example: Destroying... [id=i-056de26088f65d2c2]
aws_instance.example: Still destroying... [id=i-056de26088f65d2c2, 10s elapsed]
aws_instance.example: Still destroying... [id=i-056de26088f65d2c2, 20s elapsed]
aws_instance.example: Still destroying... [id=i-056de26088f65d2c2, 30s elapsed]
aws_instance.example: Still destroying... [id=i-056de26088f65d2c2, 40s elapsed]
aws_instance.example: Destruction complete after 42s
aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Still creating... [20s elapsed]
aws_instance.example: Creation complete after 23s [id=i-0a032cc6b030a305e]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

Outputs:

public_ip = "3.38.212.95"
vaarantaubuntu-iammv:~/Terraform$
```

위의 퍼블릭 ip로 접속하기 => 이미지를 불러올 수 없음



Pochacoo, HELLO

iam user가 s3객체를 읽어올 수 있도록 권한(AmazonS3ReadOnlyAccess) 추가하기

요약

ARN
arn:aws:iam::710665100022:user/mzcroot

콘솔 액세스
MFA 없이 활성화됨

액세스 키 1
AKIA2K5XQ3L3GLJJRHUQ - Active
오늘 사용됨, 22시간 기준.

생성됨
April 16, 2024, 16:45 (UTC+09:00)

마지막 콘솔 로그인
안 함

액세스 키 2
액세스 키 만들기

권한 | 그룹 | 태그 (1) | 보안 자격 증명 | 액세스 관리자

권한 정책 (3)
사용자에게 직접 연결된 정책을 통해 또는 그룹을 통해 권한을 정의합니다.

필터링 기준 유형: 모든 유형

<input type="checkbox"/>	정책 이름	유형	연결 방식
<input type="checkbox"/>	AdministratorAccess	AWS 관리형 - 직무	직접
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	AWS 관리형	직접

⇒ 하지만 안됨

정책 생성

정책 (1196) 정보

정책은 권한을 정의하는 AWS의 객체입니다.

필터링 기준 유형

Q s3acc X 모든 유형

정책 이름 ▲ 유형 ▼ 다음

[s3access](#) 고객 관리형

s3access

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "s3:GetObject",
8         "s3:ListBucket"
9       ],
10      "Resource": [
11        "arn:aws:s3::mzc-s3-bucket",
12        "arn:aws:s3::mzc-s3-bucket/*"
13      ]
14    }
15  ]
16 }

```

iam user에 정책 연결

실습 - 자주 바뀌는 변수를 매일 바꿔주는게 아니라 자동으로 가져오게 >?

입력 변수

cat <<EOT > variables.tf

variable "server_port" {

description = "The port the server will use for HTTP requests"

type = number

}

EOT

=>이걸로 MAIN 바꾸기

Enter and value: 8080 입력

terafform init

terraform plan

대화형태 싫을 때 옵션 줄 수 있음

```
terraform plan -var "server_port=8080"
```

```
export TF_VAR_server_port=8080
```

```
terraform plan
```

```
export | grep TF_VAR_
```

=> 환경변수로 지정해줌을 확인 가능

```
unset TF_VAR_server_port //풀기
```

#디폴드값 지정

```
cat <<EOT > variables.tf
```

```
variable "server_port" {
```

```
    description = "The port the server will use for HTTP requests"
```

```
    type        = number
```

```
    default     = 8080    // default = 8080을 추가해준 것
```

```
}
```

```
EOT
```

실습 - 웹 서버 포트 지정 배포

새로운 디렉토리 만들기

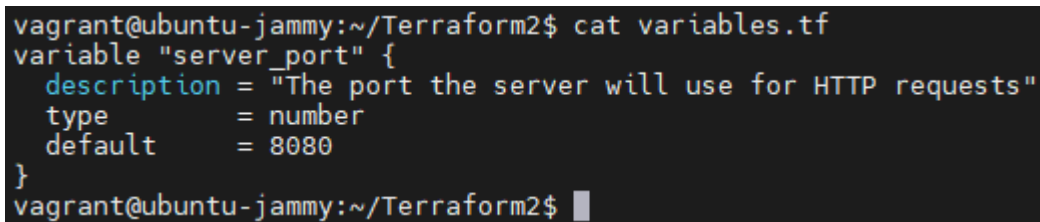
mkdir Terraform2

웹 서버 포트 지정 배포

cat <<EOT > variables.tf

```
variable "server_port" {  
    description = "The port the server will use for HTTP requests"  
    type        = number  
    default     = 8080  
}
```

EOT

A terminal window screenshot showing the command 'cat variables.tf' and its output. The output displays the Terraform variable definition for 'server_port' with its description, type, and default value.

```
vagrant@ubuntu-jammy:~/Terraform2$ cat variables.tf  
variable "server_port" {  
    description = "The port the server will use for HTTP requests"  
    type        = number  
    default     = 8080  
}  
vagrant@ubuntu-jammy:~/Terraform2$
```

cat <<EOT > main.tf

```
provider "aws" {  
    region = "ap-northeast-2"  
}  
  
resource "aws_instance" "example" {  
    ami                = "ami-09a7535106fbd42d5"  
    instance_type      = "t2.micro"  
    vpc_security_group_ids = [aws_security_group.instance.id]
```

```
user_data = <<-EOF
```

```
#!/bin/bash
```

```
echo "My Web Server - var test" > index.html
```

```
nohup busybox httpd -f -p ${var.server_port} &
```

```
EOF
```

```
user_data_replace_on_change = true
```

```
tags = {
```

```
  Name = "Single-MyWebSrv"
```

```
}
```

```
}
```

```
resource "aws_security_group" "instance" {
```

```
  name = var.security_group_name
```

```
  ingress {
```

```
    from_port = var.server_port
```

```
    to_port   = var.server_port
```

```
    protocol = "tcp"
```

```
    cidr_blocks = ["0.0.0.0/0"]
```

```
  }
```

```
}
```

```
variable "security_group_name" {  
    description = "The name of the security group"  
    type        = string  
    default     = "terraform-my-instance"  
}
```

```
output "public_ip" {  
    value      = aws_instance.example.public_ip  
    description = "The public IP of the Instance"  
}
```

EOT

terraform plan

terraform apply

terraform apply -auto-approve

모니터링 탭 하나 만들기

```
-----  
Single-WebSrv   3.38.212.95   running  
Single-MyWebSrv 43.203.141.249   running  
-----  
Single-WebSrv   3.38.212.95   running  
Single-MyWebSrv 43.203.141.249   running
```

terraform state show aws_instance.example


```
vagrant@ubuntu-jammy:~/Terraform2$ terraform state show aws_instance.example
# aws_instance.example:
resource "aws_instance" "example" {
  ami              = "ami-09a7535106fbd42d5"
  arn              = "arn:aws:ec2:ap-northeast-2:710665100022:instance/i-0d2c3d931c1b896c1"
  associate_public_ip_address = true
  availability_zone = "ap-northeast-2a"
  cpu_core_count   = 1
  cpu_threads_per_core = 1
  disable_api_stop  = false
  disable_api_termination = false
  ebs_optimized     = false
  get_password_data = false
  hibernation       = false
  host_id           = null
  iam_instance_profile = null
  id               = "i-0d2c3d931c1b896c1"
  instance_initiated_shutdown_behavior = "stop"
  instance_lifecycle = null
  instance_state     = "running"
  instance_type      = "t2.micro"
  ipv6_address_count = 0
  ipv6_addresses     = []
  key_name           = null
  monitoring         = false
  outpost_arn        = null
  password_data      = null
  placement_group     = null
  placement_partition_number = 0
  primary_network_interface_id = "eni-0fe5bfdd1eb288a05"
  private_dns         = "ip-172-31-9-33.ap-northeast-2.compute.internal"
  private_ip          = "172.31.9.33"
  public_dns          = "ec2-43-203-141-249.ap-northeast-2.compute.amazonaws.com"
  public_ip           = "43.203.141.249"
  secondary_private_ips = []
}
```

아까 8080으로 넣었으니까 실제 8080으로 접속되는지 확인하기

탭하나 더 만들기

PIP=43.203.141.249

PPT=8080

#모니터링

```
while true; do curl --connect-timeout 1 http://$PIP:$PPT/ ; echo "-----"
-----"; date; sleep 1; done
```

```
Wed Apr 17 06:58:33 UTC 2024
My Web Server - var test
-----
Wed Apr 17 06:58:34 UTC 2024
My Web Server - var test
-----
Wed Apr 17 06:58:35 UTC 2024
My Web Server - var test
-----
Wed Apr 17 06:58:36 UTC 2024
My Web Server - var test
-----
Wed Apr 17 06:58:37 UTC 2024
My Web Server - var test
```