****

**Contents**

1. **사업 개요**
   1. **사업 명**
   2. **사업 목적**
   3. **요구 사항**
2. **네트워크 토폴로지** 
   1. **가용영역**
   2. **네트워크 설정 ( 서브넷 , 라우팅, 인터넷게이트웨이)**
   3. **그룹 설정 ( 보안, 타겟,,)**
3. **인프라 자동화 설정** 
   1. **VPC 설정**

**3.1.1 VPC 생성**

**3.1.2 서브넷 설정**

**3.1.3 라우팅 테이블 설정**

**3.1.4 인터넷 게이트웨이 설정**

**3.1.5 보안 그룹 설정**

* 1. **EC2 Make & AutoScaling**

**3.2.1 인스턴스 생성**

**3.2.2 보안 그룹 설정**

* 1. **RDS Replication**

**3.3.1 RDS 생성**

**3.3.2 RDS 서브넷 생성**

* 1. **NAT 인스턴스 생성**
  2. **Bastion 인스턴스 생성**
  3. **S3 버켓 생성**
  4. **IAM 권한**
  5. **RoadBalancing**
  6. **VPC Peering**
  7. **AutoScaling**

1. **인프라 그래픽 인터페이스 설정**
   1. **도메인 등록**
   2. **AWS Certificate Manager ( SSL /TLS )**
   3. **CloudFront ( ELB )**
   4. **Route53**
   5. **AWS WAF - ALB**
   6. **CloudFormation**
   7. **Monitoring**

**Ⅰ. PROJECT INFO**

**1. 사업 명 : KGCINEMA 의 SERVER MIGRATION**

**2. 사업 목적 :**

****

**코로나19가 극장 산업을 압박한 후 OTT 산업이 새로운 활로로 떠오르고 있다.**

**이에 KGCINEMA는 기업의 혁신을 위해서 같은 업계에 있는 글로벌 기업인 WORLDCINEMA를 인수합병하고, 기존 OTT 강자 넷플릭스에 이어 KGFLIX라는 온라인 스트리밍 플랫폼을 만들어 OTT로 사업 영역을 확장하려고 한다. 사업 영역을 확장하면서 자사의 인프라를 해외 진출과 새로운 플랫폼을 운영하기 위해 AWS로 마이그레이션 하려고 한다.**

**3. 요구 사항 :**

**1 ) 원활한 서비스 이용을 위해 ALB 로드밸런서를 통한 헤더정보를 이용한 부하분산으로 트래픽 감소와 SSL인증서 탑재로 인한 SSL 암호화/복호화로 보안 강화**

**2) 저렴한 비용으로 안정적이고 예측 가능한 성능을 유지하기위해 필요에 따라 유연하게 서버를 확장시키고 축소 가능한 Auto Scaling 설정**

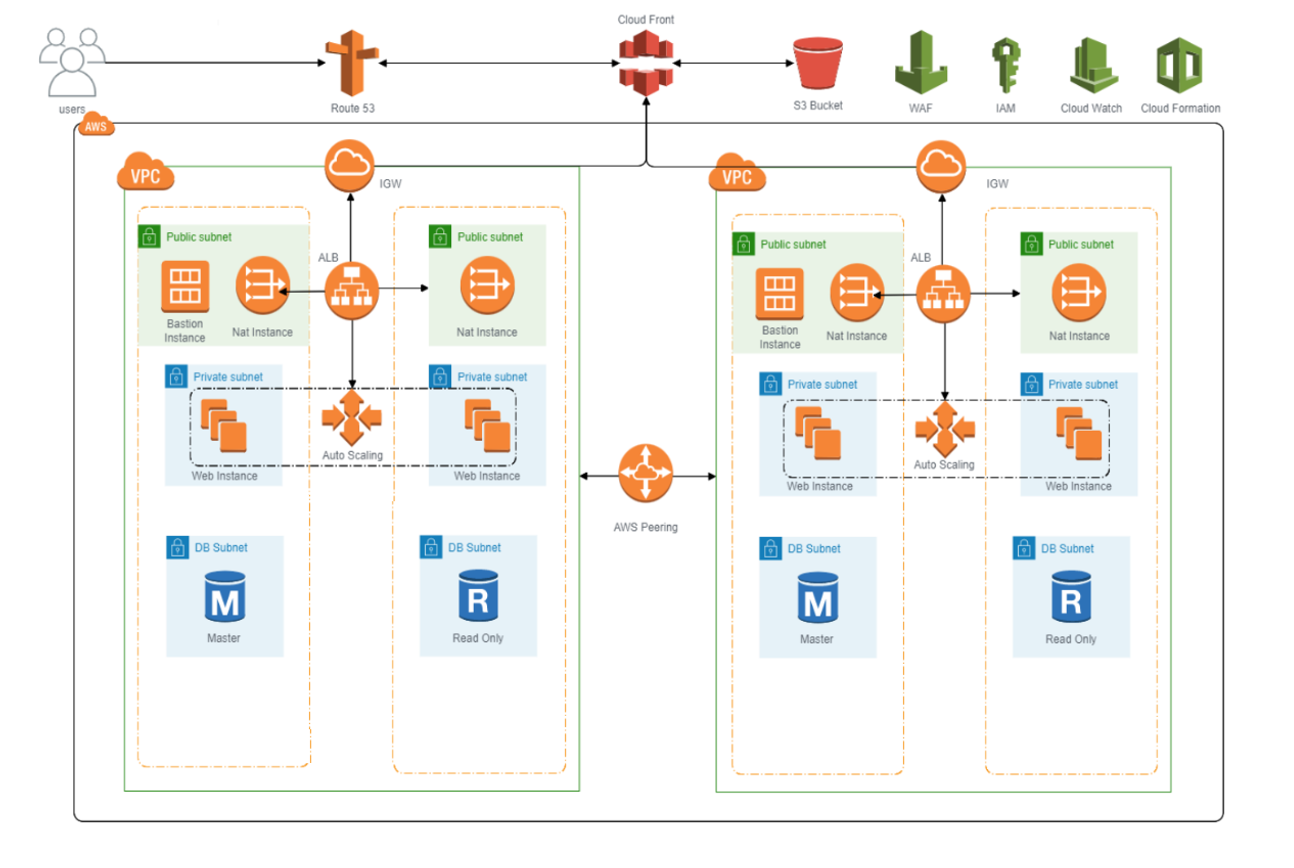
**3) AWS WAF와 CloudFront를 연동해 VPC내에 트래픽이 유입될때 CloudFront의 WAF는 적절한 임계치 룰과 L7 DOS차단 룰적용, IP Blacklist 차단 규칙으로 VPC에 부적절한 트래픽의 유입이 적어짐**

**4) AWS CloudFormation을 통해 인프라 구성을 스택단위로 구성하여 차 후 해외 진출시 배포를 한번에 구축하거나 한번에 제거 가능**

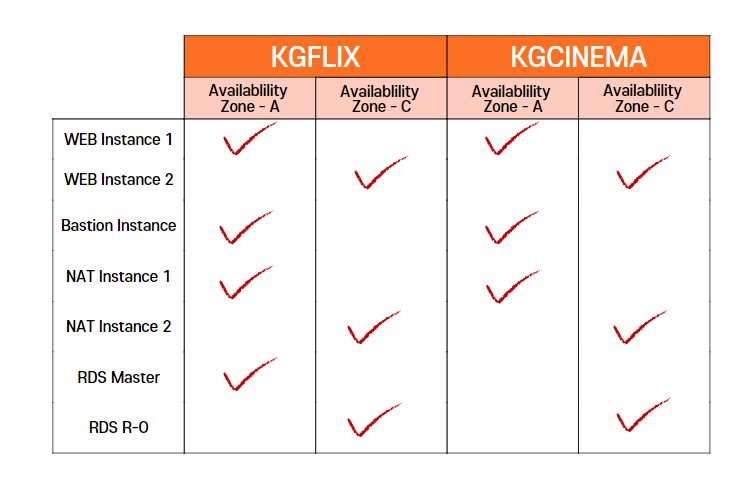
**5) 프로메테우스를 이용해 데이터를 수집하여 그라파나로 정보를 조회 가능.**

**다양한 데이터베이스를 지우너하고 다양한 플러그인을 통해 관련도구들과 연동할 수 있는 확장성을 가지고 있어 소수의 인원으로 모니터링이 가능하게 만듬**

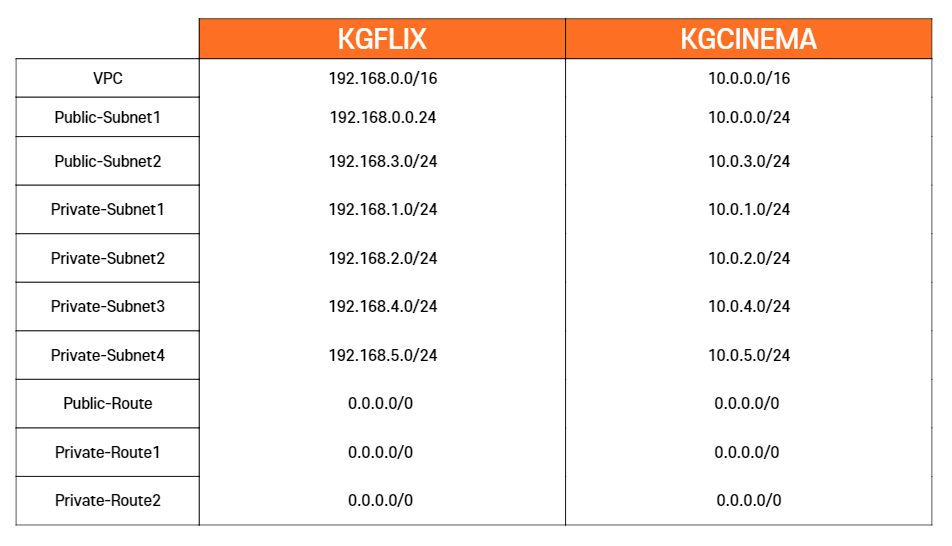
**Ⅱ. 네트워크 토폴로지**

****

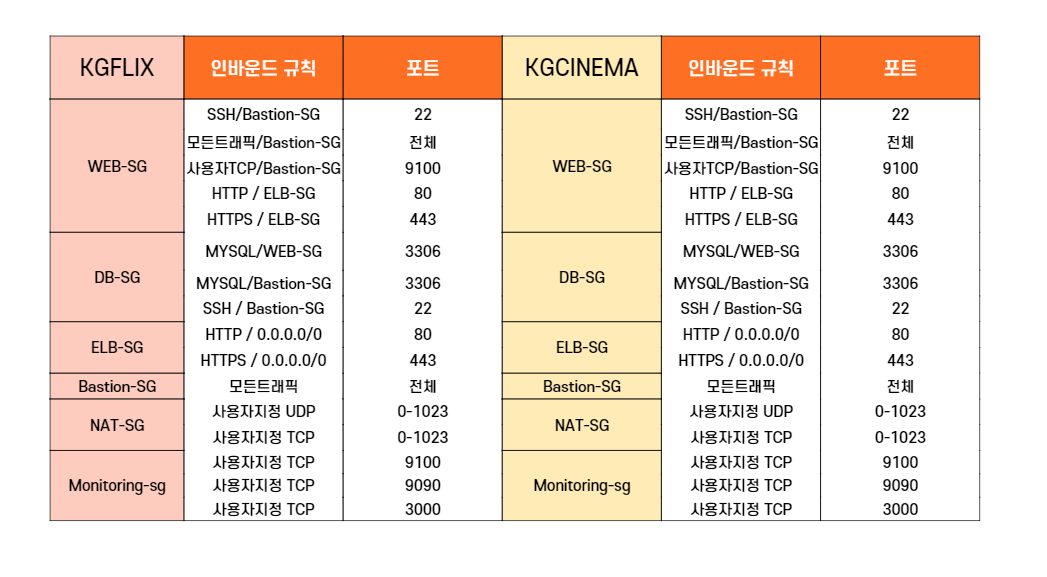
**1. 가용 영역**



**2. 네트워크 설정**

****

**3. 그룹 설정**



**Ⅲ. 인프라 구조설정**

**1. VPC 설정**

**1.1 VPC생성**

KGFLIXVPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: 192.168.0.0/16

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: KGFLIXVPC

KGCINEMAVPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: 10.0.0.0/16

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: KGCINEMAVPC

**1.2 서브넷 설정**

KGFLIXPublicSubnet1: # public subnet

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGFLIXVPC

CidrBlock: 192.168.0.0/24

AvailabilityZone: "ap-northeast-2a"

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: KGFLIX public subnet1

KGFLIXPublicSubnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGFLIXVPC

CidrBlock: 192.168.3.0/24

AvailabilityZone: "ap-northeast-2c"

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: KGFLIX public subnet2

KGCINEMAPublicSubnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGCINEMAVPC

CidrBlock: 10.0.0.0/24

AvailabilityZone: "ap-northeast-2a"

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: KGCINEMA public subnet1

KGCINEMAPublicSubnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGCINEMAVPC

CidrBlock: 10.0.3.0/24

AvailabilityZone: "ap-northeast-2c"

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: KGCINEMA public subnet2

KGFLIXPrivateSubnet1: # KGFLIX create private subnet

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGFLIXVPC

CidrBlock: 192.168.1.0/24

AvailabilityZone: "ap-northeast-2a"

Tags:

- Key: Name

Value: KGFLIX private subnet1

KGFLIXPrivateSubnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGFLIXVPC

CidrBlock: 192.168.2.0/24

AvailabilityZone: "ap-northeast-2a"

Tags:

- Key: Name

Value: KGFLIX private subnet2

KGFLIXPrivateSubnet3:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGFLIXVPC

CidrBlock: 192.168.4.0/24

AvailabilityZone: "ap-northeast-2c"

Tags:

- Key: Name

Value: KGFLIX private subnet3

KGFLIXPrivateSubnet4:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGFLIXVPC

CidrBlock: 192.168.5.0/24

AvailabilityZone: "ap-northeast-2c"

Tags:

- Key: Name

Value: KGFLIX private subnet4

KGCINEMAPrivateSubnet1: # KGCINEMA create private subnet

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGCINEMAVPC

CidrBlock: 10.0.1.0/24

AvailabilityZone: "ap-northeast-2a"

Tags:

- Key: Name

Value: KGCINEMA private subnet1

KGCINEMAPrivateSubnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGCINEMAVPC

CidrBlock: 10.0.2.0/24

AvailabilityZone: "ap-northeast-2a"

Tags:

- Key: Name

Value: KGCINEMA private subnet2

KGCINEMAPrivateSubnet3:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGCINEMAVPC

CidrBlock: 10.0.4.0/24

AvailabilityZone: "ap-northeast-2c"

Tags:

- Key: Name

Value: KGCINEMA private subnet3

KGCINEMAPrivateSubnet4:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref KGCINEMAVPC

CidrBlock: 10.0.5.0/24

AvailabilityZone: "ap-northeast-2c"

Tags:

- Key: Name

Value: KGCINEMA private subnet4

**1.3 라우팅 테이블 설정**

KGFLIXPublicRT: # KGFLIX public subnet route table

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref KGFLIXVPC

Tags:

- Key: Name

Value: KGFLIX Public RT

KGFLIXPublicRoute:

Type: AWS::EC2::Route

DependsOn: KGFLIXIGWAttachment

Properties:

RouteTableId: !Ref KGFLIXPublicRT

DestinationCidrBlock: 0.0.0.0/0

GatewayId: !Ref KGFLIXIGW

KGFLIXPublicSubnetRTAssociation1:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGFLIXPublicRT

SubnetId: !Ref KGFLIXPublicSubnet1

KGFLIXPublicSubnetRTAssociation2:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGFLIXPublicRT

SubnetId: !Ref KGFLIXPublicSubnet2

KGCINEMAPublicRT: # KGCINEMA public subnet route table

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref KGCINEMAVPC

Tags:

- Key: Name

Value: KGCINEMA Public RT

KGCINEMAPublicRoute:

Type: AWS::EC2::Route

DependsOn: KGCINEMAIGWAttachment

Properties:

RouteTableId: !Ref KGCINEMAPublicRT

DestinationCidrBlock: 0.0.0.0/0

GatewayId: !Ref KGCINEMAIGW

KGCINEMAPublicSubnetRTAssociation1:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGCINEMAPublicRT

SubnetId: !Ref KGCINEMAPublicSubnet1

KGCINEMAPublicSubnetRTAssociation2:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGCINEMAPublicRT

SubnetId: !Ref KGCINEMAPublicSubnet2

KGFLIXPrivateRT2: # KGFLIX create private route table

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref KGFLIXVPC

Tags:

- Key: Name

Value: KGFLIX Private RT2

KGFLIXPrivateRoute2:

Type: AWS::EC2::Route

DependsOn: NatInstance2

Properties:

RouteTableId: !Ref KGFLIXPrivateRT2

DestinationCidrBlock: 0.0.0.0/0

InstanceId: !Ref NatInstance2

KGFLIXPrivateSubnetRTAssociation1:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGFLIXPrivateRT1

SubnetId: !Ref KGFLIXPrivateSubnet1

KGFLIXPrivateSubnetRTAssociationa2:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGFLIXPrivateRT1

SubnetId: !Ref KGFLIXPrivateSubnet2

KGFLIXPrivateSubnetRTAssociationc3:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGFLIXPrivateRT2

SubnetId: !Ref KGFLIXPrivateSubnet3

KGFLIXPrivateSubnetRTAssociation4:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGFLIXPrivateRT2

SubnetId: !Ref KGFLIXPrivateSubnet4

KGCINEMAPrivateRT1: # KGCINEMA create private route table

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref KGCINEMAVPC

Tags:

- Key: Name

Value: KGCINEMA Private RT1

KGCINEMAPrivateRoute1:

Type: AWS::EC2::Route

DependsOn: NatInstance3

Properties:

RouteTableId: !Ref KGCINEMAPrivateRT1

DestinationCidrBlock: 0.0.0.0/0

InstanceId: !Ref NatInstance3

KGCINEMAPrivateRT2:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref KGCINEMAVPC

Tags:

- Key: Name

Value: KGCINEMA Private RT2

KGCINEMAPrivateRoute2:

Type: AWS::EC2::Route

DependsOn: NatInstance4

Properties:

RouteTableId: !Ref KGCINEMAPrivateRT2

DestinationCidrBlock: 0.0.0.0/0

InstanceId: !Ref NatInstance4

KGCINEMAPrivateSubnetRTAssociation1:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGCINEMAPrivateRT1

SubnetId: !Ref KGCINEMAPrivateSubnet1

KGCINEMAPrivateSubnetRTAssociation2:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGCINEMAPrivateRT1

SubnetId: !Ref KGCINEMAPrivateSubnet2

KGCINEMAPrivateSubnetRTAssociation3:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGCINEMAPrivateRT2

SubnetId: !Ref KGCINEMAPrivateSubnet3

KGCINEMAPrivateSubnetRTAssociation4:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref KGCINEMAPrivateRT2

SubnetId: !Ref KGCINEMAPrivateSubnet4

**1.4 인터넷 게이트웨이 설정**

KGFLIXIGW:

Type: AWS::EC2::InternetGateway

Properties:

Tags:

- Key: Name

Value: KGFLIXIGW

KGFLIXIGWAttachment:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

InternetGatewayId: !Ref KGFLIXIGW

VpcId: !Ref KGFLIXVPC

KGCINEMAIGW:

Type: AWS::EC2::InternetGateway

Properties:

Tags:

- Key: Name

Value: KGCINEMAIGW

KGCINEMAIGWAttachment:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

InternetGatewayId: !Ref KGCINEMAIGW

VpcId: !Ref KGCINEMAVPC

**2 EC2 설정**

**2.1 인스턴스 생성**

KGFLIXWEBLaunchConfiguration:

Type: AWS::AutoScaling::LaunchConfiguration

Properties:

ImageId: ami-086eee4e12dd93f7c

InstanceType: t2.micro

KeyName: !Ref KGFLIX

LaunchConfigurationName: KGFLIX\_WEB\_AS

SecurityGroups:

- !Ref KGFLIXWebSG

UserData:

Fn::Base64:

!Sub |

#!/bin/bash

sudo yum update

sudo yum -y install unzip

sudo yum install -y http\*

sudo yum install -y php

sudo wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1xgeKSbglJg-UPzngtckMnrvqN2zdyAvv' -O kgcinema1.zip

sudo unzip kgcinema1.zip -d /var/www/html/

sudo systemctl restart httpd

KGCINEMAWEBLaunchConfiguration:

Type: AWS::AutoScaling::LaunchConfiguration

Properties:

ImageId: ami-086eee4e12dd93f7c

InstanceType: t2.micro

KeyName: !Ref KGFLIX

LaunchConfigurationName: KGCINEMA\_WEB\_AS

SecurityGroups:

- !Ref KGCINEMAWebSG

UserData:

Fn::Base64:

!Sub |

#!/bin/bash

sudo yum update

sudo yum -y install unzip

sudo yum install -y http\*

sudo yum install -y php

sudo wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1xgeKSbglJg-UPzngtckMnrvqN2zdyAvv' -O kgcinema1.zip

sudo unzip kgcinema1.zip -d /var/www/html/

sudo systemctl restart httpd

**2.2 보안 그룹 설정**

KGFLIXWebSG:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: KGFLIXWebSGSG

VpcId: !Ref KGFLIXVPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: '80'

ToPort: '80'

SourceSecurityGroupId: !Ref KGFLIXELBSG

- IpProtocol: tcp

FromPort: '443'

ToPort: '443'

SourceSecurityGroupId: !Ref KGFLIXELBSG

- IpProtocol: tcp

FromPort: '22'

ToPort: '22'

SourceSecurityGroupId: !Ref KGFLIXBastionHostSG

KGCINEMAWebSG:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: KGCINEMAWebSG

VpcId: !Ref KGCINEMAVPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: '80'

ToPort: '80'

SourceSecurityGroupId: !Ref KGCINEMAELBSG

- IpProtocol: tcp

FromPort: '443'

ToPort: '443'

SourceSecurityGroupId: !Ref KGCINEMAELBSG

- IpProtocol: tcp

FromPort: '22'

ToPort: '22'

SourceSecurityGroupId: !Ref KGCINEMABastionHostSG

KGFLIXELBSG:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: KGFLIXSG

VpcId: !Ref KGFLIXVPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: '80'

ToPort: '80'

CidrIp: 0.0.0.0/0

- IpProtocol: tcp

FromPort: '443'

ToPort: '443'

CidrIp: 0.0.0.0/0

KGCINEMAELBSG:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: KGCINEMASG

VpcId: !Ref KGCINEMAVPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: '80'

ToPort: '80'

CidrIp: 0.0.0.0/0

- IpProtocol: tcp

FromPort: '443'

ToPort: '443'

CidrIp: 0.0.0.0/0

KGFLIXDBSecurityGroup: # create security group

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: KGFLIXDBSG

VpcId: !Ref KGFLIXVPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: '3306'

ToPort: '3306'

SourceSecurityGroupId: !Ref KGFLIXWebSG

- IpProtocol: tcp

FromPort: '3306'

ToPort: '3306'

SourceSecurityGroupId: !Ref KGFLIXBastionHostSG

- IpProtocol: tcp

FromPort: '22'

ToPort: '22'

SourceSecurityGroupId: !Ref KGFLIXBastionHostSG

KGCINEMADBSecurityGroup: # create security group

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: KGCINEMADBSG

VpcId: !Ref KGCINEMAVPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: '3306'

ToPort: '3306'

SourceSecurityGroupId: !Ref KGCINEMAWebSG

- IpProtocol: tcp

FromPort: '3306'

ToPort: '3306'

SourceSecurityGroupId: !Ref KGCINEMABastionHostSG

- IpProtocol: tcp

FromPort: '22'

ToPort: '22'

SourceSecurityGroupId: !Ref KGCINEMABastionHostSG

**3. RDS 설정**

**3.1 RDS 서브넷 생성**

KGFLIXDBSubnetGroup:

Type: "AWS::RDS::DBSubnetGroup"

Properties:

DBSubnetGroupDescription: KGFLIXDBSub

DBSubnetGroupName: KGFLIXDBSubnetGroup

SubnetIds:

- !Ref KGFLIXPrivateSubnet2

- !Ref KGFLIXPrivateSubnet4

KGCINEMADBSubnetGroup:

Type: "AWS::RDS::DBSubnetGroup"

Properties:

DBSubnetGroupDescription: KGCINEMADBSub

DBSubnetGroupName: KGCINEMADBSubnetGroup

SubnetIds:

- !Ref KGCINEMAPrivateSubnet2

- !Ref KGCINEMAPrivateSubnet4

**3.2 RDS 생성**

KGFLIXMasterDB:

Type: 'AWS::RDS::DBInstance'

Properties:

DBName: Webdb

MasterUsername: !Ref DBID

MasterUserPassword: !Ref DBPW

Engine: mysql

DBInstanceClass: db.t2.micro

StorageType: gp2

PubliclyAccessible: false

AllocatedStorage: "10"

DBInstanceIdentifier: WEBDB

MultiAZ: false

VPCSecurityGroups:

- !Ref KGFLIXDBSecurityGroup

DBSubnetGroupName: !Ref KGFLIXDBSubnetGroup

KGCINEMAMasterDB:

Type: 'AWS::RDS::DBInstance'

Properties:

DBName: Webdb

MasterUsername: !Ref DBID

MasterUserPassword: !Ref DBPW

Engine: mysql

DBInstanceClass: db.t2.micro

StorageType: gp2

PubliclyAccessible: false

AllocatedStorage: "10"

DBInstanceIdentifier: WEBDB

MultiAZ: false

VPCSecurityGroups:

- !Ref KGCINEMADBSecurityGroup

DBSubnetGroupName: !Ref KGCINEMADBSubnetGroup

KGFLIXRDSReplica:

Type: AWS::RDS::DBInstance

DependsOn: KGFLIXMasterDB

Properties:

SourceDBInstanceIdentifier: !Ref KGFLIXMasterDB

DBInstanceIdentifier: WEBRDSReplica

DBInstanceClass: db.t2.micro

StorageType: gp2

PubliclyAccessible: false

VPCSecurityGroups:

- !Ref KGFLIXDBSecurityGroup

DBSubnetGroupName: !Ref KGFLIXDBSubnetGroup

KGCINEMARDSReplica:

Type: AWS::RDS::DBInstance

DependsOn: KGCINEMAMasterDB

Properties:

SourceDBInstanceIdentifier: !Ref KGCINEMAMasterDB

DBInstanceIdentifier: WEBRDSReplica

DBInstanceClass: db.t2.micro

StorageType: gp2

PubliclyAccessible: false

VPCSecurityGroups:

- !Ref KGCINEMADBSecurityGroup

DBSubnetGroupName: !Ref KGCINEMADBSubnetGroup

**4. NAT 인스턴스**

**4.1 NAT 보안그룹**

NATSecurityGroup1:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Enable custom tcp custom udp

VpcId: !Ref KGFLIXVPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: '0'

ToPort: '1023'

CidrIp: 0.0.0.0/0

- IpProtocol: udp

FromPort: '0'

ToPort: '1023'

CidrIp: 0.0.0.0/0

NATSecurityGroup2:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Enable custom tcp custom udp

VpcId: !Ref KGCINEMAVPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: '0'

ToPort: '1023'

CidrIp: 0.0.0.0/0

- IpProtocol: udp

FromPort: '0'

ToPort: '1023'

CidrIp: 0.0.0.0/0

**4.2 NAT 인스턴스 생성**

NatInstance1:

Type: AWS::EC2::Instance

Properties:

InstanceType: t2.micro

ImageId: ami-00295862c013bede0

KeyName: !Ref KGFLIX

Tags:

- Key: Name

Value: NAT-Instance1

SourceDestCheck: 'false'

NetworkInterfaces:

- DeviceIndex: 0

SubnetId: !Ref KGFLIXPublicSubnet1

GroupSet:

- !Ref NATSecurityGroup1

AssociatePublicIpAddress: true

NatInstance2:

Type: AWS::EC2::Instance

Properties:

InstanceType: t2.micro

ImageId: ami-00295862c013bede0

KeyName: !Ref KGFLIX

Tags:

- Key: Name

Value: NAT-Instance2

SourceDestCheck: 'false'

NetworkInterfaces:

- DeviceIndex: 0

SubnetId: !Ref KGFLIXPublicSubnet2

GroupSet:

- !Ref NATSecurityGroup1

AssociatePublicIpAddress: true

NatInstance3:

Type: AWS::EC2::Instance

Properties:

InstanceType: t2.micro

ImageId: ami-00295862c013bede0

KeyName: !Ref KGFLIX

Tags:

- Key: Name

Value: NAT-Instance3

SourceDestCheck: 'false'

NetworkInterfaces:

- DeviceIndex: 0

SubnetId: !Ref KGCINEMAPublicSubnet1

GroupSet:

- !Ref NATSecurityGroup2

AssociatePublicIpAddress: true

NatInstance4:

Type: AWS::EC2::Instance

Properties:

InstanceType: t2.micro

ImageId: ami-00295862c013bede0

KeyName: !Ref KGFLIX

Tags:

- Key: Name

Value: NAT-Instance4

SourceDestCheck: 'false'

NetworkInterfaces:

- DeviceIndex: 0

SubnetId: !Ref KGCINEMAPublicSubnet2

GroupSet:

- !Ref NATSecurityGroup2

AssociatePublicIpAddress: true

**5. Bastion 인스턴스**

**5.1 Bastion 보안 그룹**

KGFLIXBastionHostSG:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: KGFLIX Bastion Host SG

VpcId: !Ref KGFLIXVPC

SecurityGroupIngress:

- IpProtocol: 'all'

FromPort: '0'

ToPort: '0'

CidrIp: 0.0.0.0/0

KGCINEMABastionHostSG:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: KGCINEMA Bastion Host SG

VpcId: !Ref KGCINEMAVPC

SecurityGroupIngress:

- IpProtocol: 'all'

FromPort: '0'

ToPort: '0'

CidrIp: 0.0.0.0/0

**5.2. Bastion 인스턴스 생성**

KGCINEMABastionHost:

Type: AWS::EC2::Instance

Properties:

InstanceType: t2.micro

ImageId: ami-086eee4e12dd93f7c

KeyName: !Ref KGFLIX

Tags:

- Key: Name

Value: KGCINEMABastionHost

NetworkInterfaces:

- DeviceIndex: 0

SubnetId: !Ref KGCINEMAPublicSubnet1

GroupSet:

- !Ref KGCINEMABastionHostSG

UserData:

Fn::Base64:

!Sub |

#!/bin/bash

SECURE\_ANSIBLE=$(expect -c "

set timeout 3

spawn sudo amazon-linux-extras install ansible2

expect \"Is this ok [y/d/N]:"

send \"y\r\"

expect eof

")

echo "$SECURE\_ANSIBLE"

sudo wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1dxGvrfwDAYyrxM5xsQ4iRXpOyM4nFR0v' -O KGFLIX.pem

sudo yum -y install expect

ansible-galaxy collection install ansible.posix

sed -i 's/#host\_key\_checking=False/host\_key\_checking=False/g' /etc/ansible/ansible.cfg

sudo wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1dxGvrfwDAYyrxM5xsQ4iRXpOyM4nFR0v' -O KGFLIX.pem

sudo chmod 400 KGFLIX.pem

**6. S3 Bucket 생성**

S3Bucket:

Type: 'AWS::S3::Bucket'

Properties:

BucketName: !Ref Bucketname

WebsiteConfiguration:

IndexDocument: home.html

ErrorDocument: index.html

ReadPolicy:

Type: 'AWS::S3::BucketPolicy'

Properties:

Bucket: !Ref S3Bucket

PolicyDocument:

Statement:

- Action: 's3:GetObject'

Effect: Allow

Resource: !Sub 'arn:aws:s3:::${S3Bucket}/\*'

Principal: '\*'

**7. IAM 권한**

WebAdmin:

Type: 'AWS::IAM::User'

Properties:

LoginProfile:

Password: donghan11!@

Path: "/"

Policies:

- PolicyName: EC2FullAccess

PolicyDocument: {

"Version": "2012-10-17",

"Statement": [

{

"Action": "ec2:\*",

"Effect": "Allow",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "elasticloadbalancing:\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "cloudwatch:\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "autoscaling:\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "iam:CreateServiceLinkedRole",

"Resource": "\*",

"Condition": {

"StringEquals": {

"iam:AWSServiceName": [

"autoscaling.amazonaws.com",

"ec2scheduled.amazonaws.com",

"elasticloadbalancing.amazonaws.com",

"spot.amazonaws.com",

"spotfleet.amazonaws.com",

"transitgateway.amazonaws.com"

]

}

}

}

]

}

DBAdmin:

Type: 'AWS::IAM::User'

Properties:

LoginProfile:

Password: donghan11!@

Path: "/"

Policies:

- PolicyName: RDSFullAccess

PolicyDocument: {

"Version": "2012-10-17",

"Statement": [

{

"Action": [

"rds:\*",

"application-autoscaling:DeleteScalingPolicy",

"application-autoscaling:DeregisterScalableTarget",

"application-autoscaling:DescribeScalableTargets",

"application-autoscaling:DescribeScalingActivities",

"application-autoscaling:DescribeScalingPolicies",

"application-autoscaling:PutScalingPolicy",

"application-autoscaling:RegisterScalableTarget",

"cloudwatch:DescribeAlarms",

"cloudwatch:GetMetricStatistics",

"cloudwatch:PutMetricAlarm",

"cloudwatch:DeleteAlarms",

"ec2:DescribeAccountAttributes",

"ec2:DescribeAvailabilityZones",

"ec2:DescribeCoipPools",

"ec2:DescribeInternetGateways",

"ec2:DescribeLocalGatewayRouteTablePermissions",

"ec2:DescribeLocalGatewayRouteTables",

"ec2:DescribeLocalGatewayRouteTableVpcAssociations",

"ec2:DescribeLocalGateways",

"ec2:DescribeSecurityGroups",

"ec2:DescribeSubnets",

"ec2:DescribeVpcAttribute",

"ec2:DescribeVpcs",

"ec2:GetCoipPoolUsage",

"sns:ListSubscriptions",

"sns:ListTopics",

"sns:Publish",

"logs:DescribeLogStreams",

"logs:GetLogEvents",

"outposts:GetOutpostInstanceTypes"

],

"Effect": "Allow",

"Resource": "\*"

},

{

"Action": "pi:\*",

"Effect": "Allow",

"Resource": "arn:aws:pi:\*:\*:metrics/rds/\*"

},

{

"Action": "iam:CreateServiceLinkedRole",

"Effect": "Allow",

"Resource": "\*",

"Condition": {

"StringLike": {

"iam:AWSServiceName": [

"rds.amazonaws.com",

"rds.application-autoscaling.amazonaws.com"

]

}

}

}

]

}

Ansible:

Type: 'AWS::IAM::User'

Properties:

LoginProfile:

Password: donghan11!@

Path: "/"

Policies:

- PolicyName: AdministratorAccess

PolicyDocument: {

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "\*",

"Resource": "\*"

}

]

}

**8. RoadBalancing**

KGFLIXALB:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: KGFLIXALB

Subnets:

- !Ref KGFLIXPublicSubnet1

- !Ref KGFLIXPublicSubnet2

SecurityGroups:

- !Ref KGFLIXELBSG

KGCINEMAALB:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: KGCINEMAALB

Subnets:

- !Ref KGCINEMAPublicSubnet1

- !Ref KGCINEMAPublicSubnet2

SecurityGroups:

- !Ref KGCINEMAELBSG

KGFLIXALBListener:

Type: AWS::ElasticLoadBalancingV2::Listener

Properties:

LoadBalancerArn: !Ref KGFLIXALB

Port: 80

Protocol: HTTP

DefaultActions:

- Type: forward

TargetGroupArn: !Ref KGFLIXWEBTargetGroup

KGCINEMAALBListener:

Type: AWS::ElasticLoadBalancingV2::Listener

Properties:

LoadBalancerArn: !Ref KGCINEMAALB

Port: 80

Protocol: HTTP

DefaultActions:

- Type: forward

TargetGroupArn: !Ref KGCINEMAWEBTargetGroup

**9. VPC Peering**

VPCPeering:

Type: AWS::EC2::VPCPeeringConnection

Properties:

PeerVpcId: !Ref KGFLIXVPC

VpcId: !Ref KGCINEMAVPC

**10. Auto Scaling**

KGFLIXAutoScaling:

Type: AWS::AutoScaling::AutoScalingGroup

DependsOn: KGFLIXVPC

Properties:

AutoScalingGroupName: KGFLIXAutoScaling

VPCZoneIdentifier:

- !Ref KGFLIXPrivateSubnet1

- !Ref KGFLIXPrivateSubnet3

Cooldown: 10

DesiredCapacity: 2

LaunchConfigurationName: !Ref KGFLIXWEBLaunchConfiguration

MaxSize: 2

MinSize: 2

TargetGroupARNs:

- !Ref KGFLIXWEBTargetGroup

Tags:

- Key: Name

Value: KGFLIX-WEB-AS

PropagateAtLaunch: true

KGCINEMAAutoScaling:

Type: AWS::AutoScaling::AutoScalingGroup

DependsOn: KGCINEMAVPC

Properties:

AutoScalingGroupName: KGCINEMAAutoScaling

VPCZoneIdentifier:

- !Ref KGCINEMAPrivateSubnet1

- !Ref KGCINEMAPrivateSubnet3

Cooldown: 10

DesiredCapacity: 2

LaunchConfigurationName: !Ref KGCINEMAWEBLaunchConfiguration

MaxSize: 2

MinSize: 2

TargetGroupARNs:

- !Ref KGCINEMAWEBTargetGroup

Tags:

- Key: Name

Value: KGCINEMA-WEB-AS

PropagateAtLaunch: true

KGFLIXASGPolicy:

Type: AWS::AutoScaling::ScalingPolicy

Properties:

AutoScalingGroupName: !Ref KGFLIXAutoScaling

PolicyType: TargetTrackingScaling

TargetTrackingConfiguration:

PredefinedMetricSpecification:

PredefinedMetricType: ASGAverageCPUUtilization

TargetValue: 50

KGCINEMAASGPolicy:

Type: AWS::AutoScaling::ScalingPolicy

Properties:

AutoScalingGroupName: !Ref KGCINEMAAutoScaling

PolicyType: TargetTrackingScaling

TargetTrackingConfiguration:

PredefinedMetricSpecification:

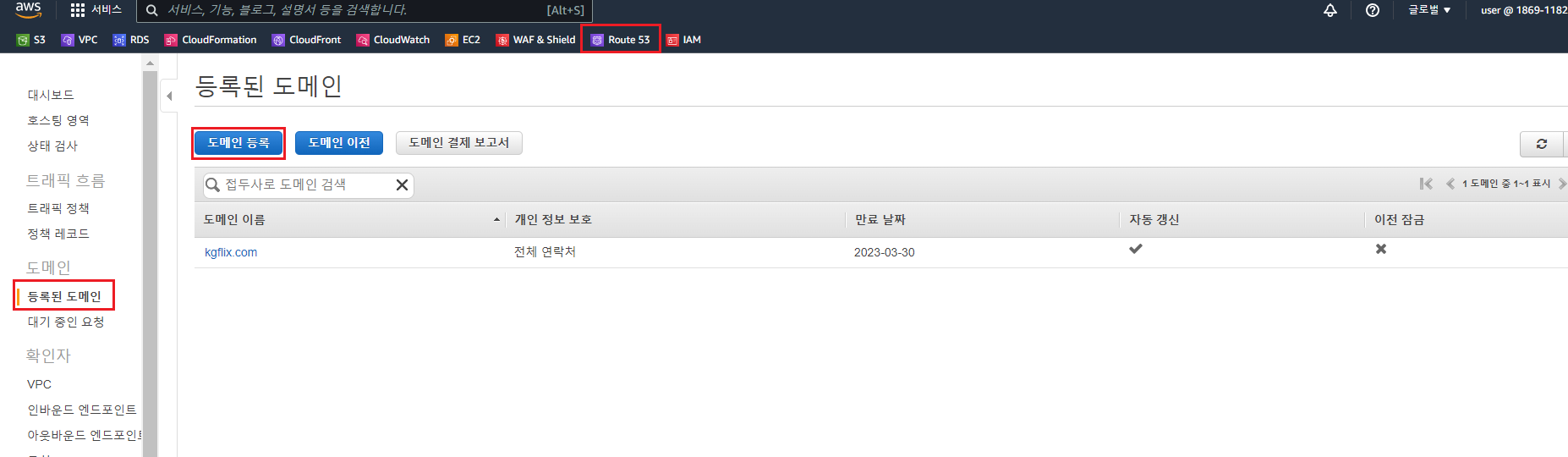
PredefinedMetricType: ASGAverageCPUUtilization

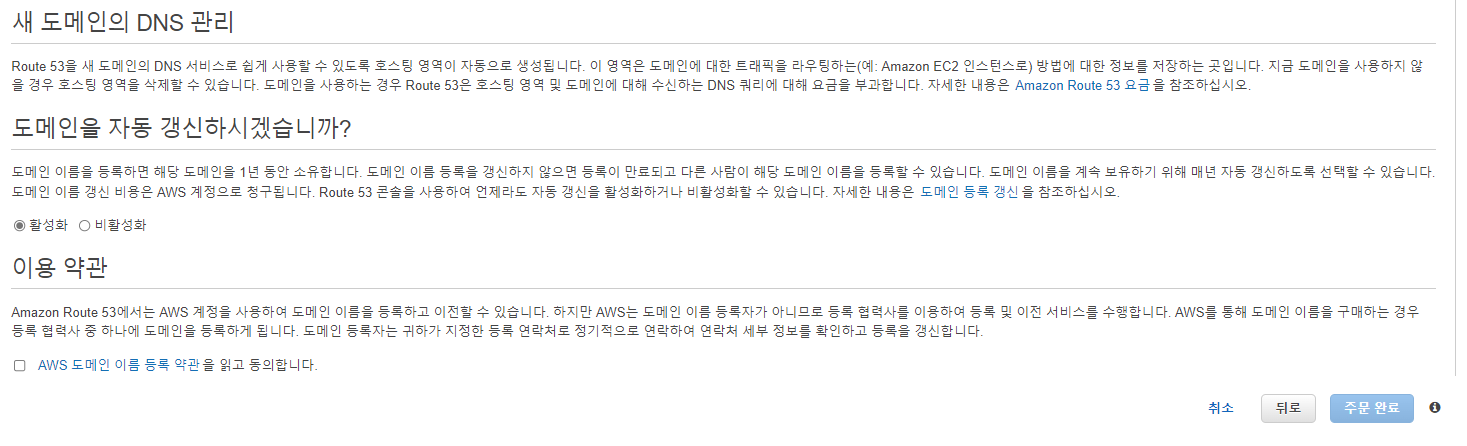
TargetValue: 50

**Ⅳ. 기술 설정**

**1. 도메인 등록**

**Route53 -> 등록된 도메인 -> 도메인 등록 -> 원하는 도메인 결정 -> 연락처 세부정보 등록 -> 도매인 구매**

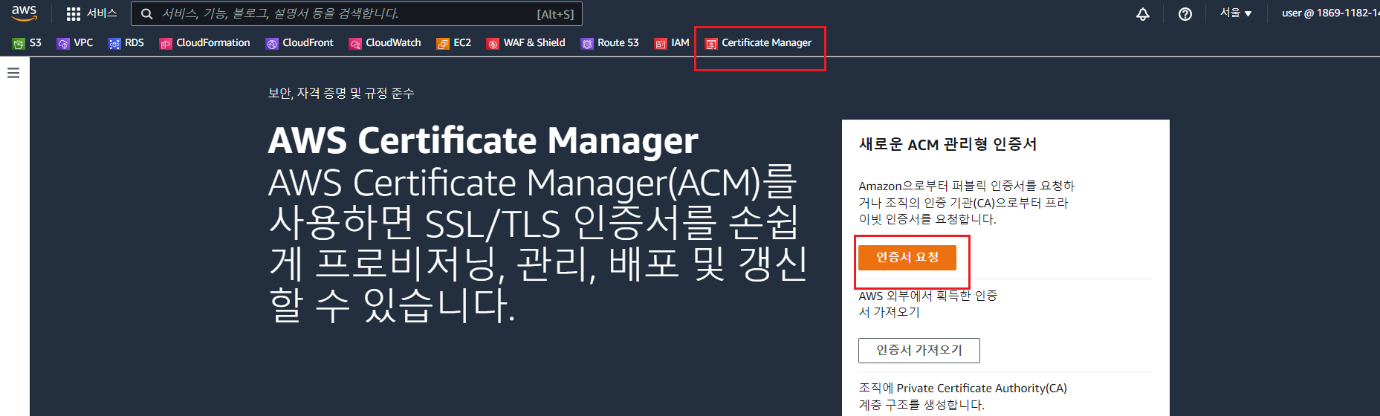
****

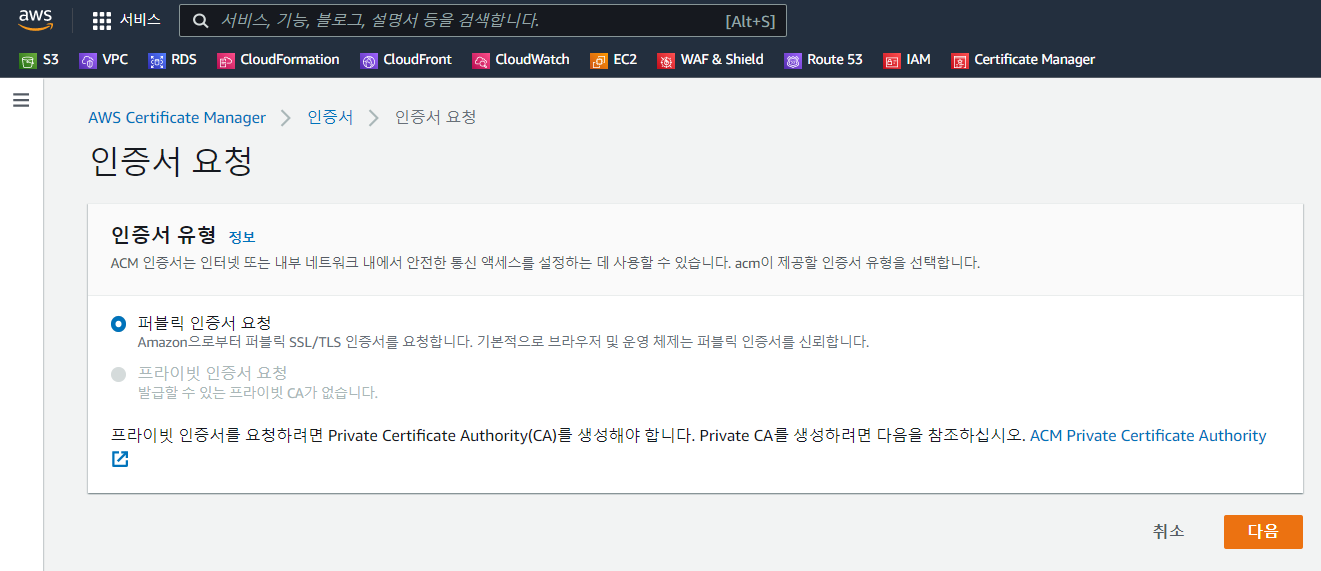
****

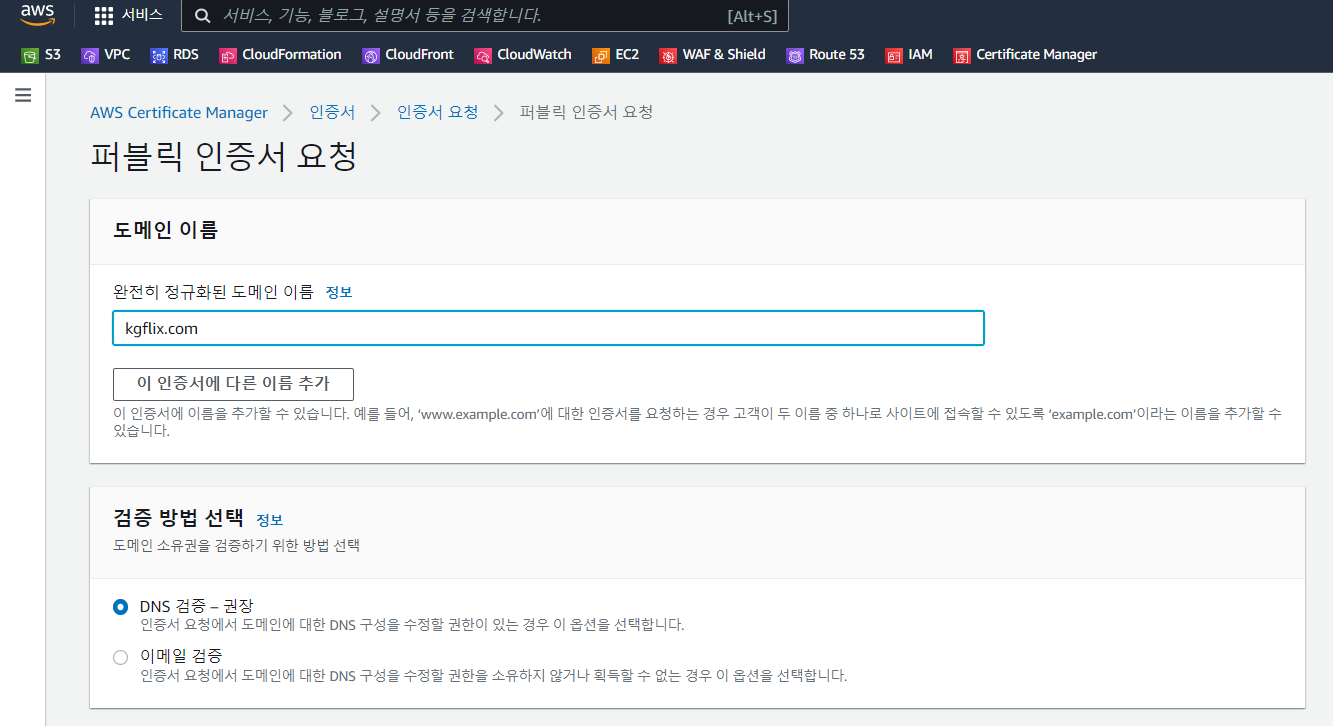
**2. AWS Certificate Manager ( SSL / TLS 인증서 발급 )**

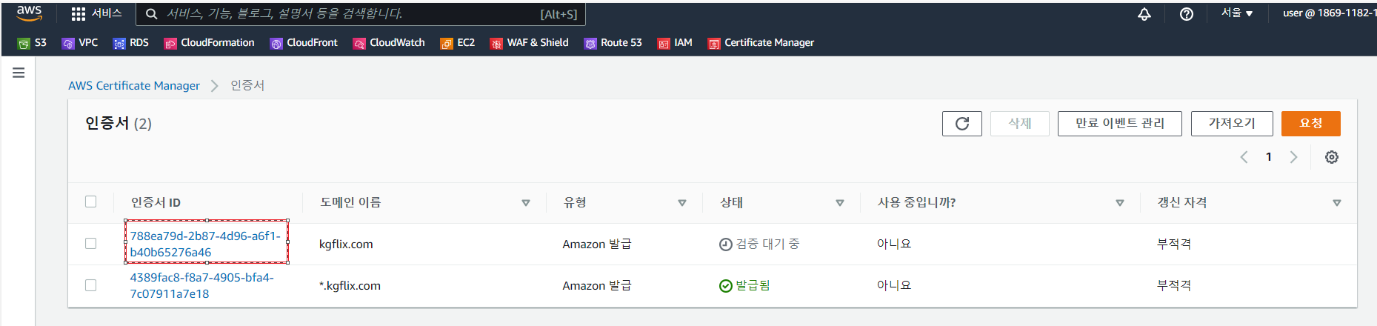
**※ 서울리전과 버지니아 북부리전 2개 인증서 발급**

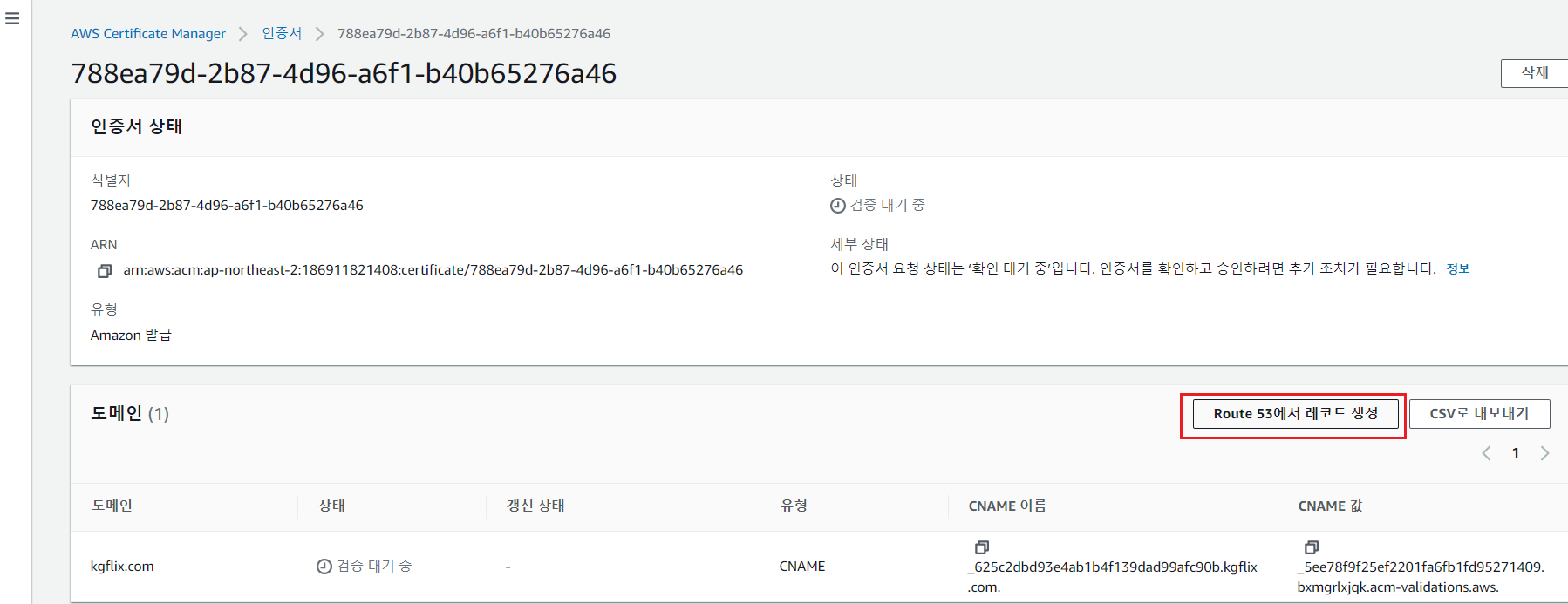
**Certificate Manager -> 인증서 요청 -> 구매한 도메인 입력 -> 인증서 ID를 클릭 -> Route53에서 레코드 생성 -> 발급 완료**

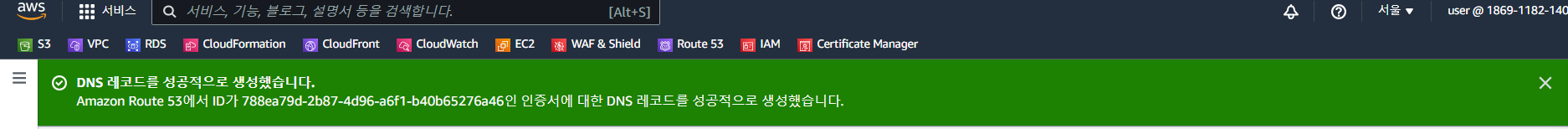
****

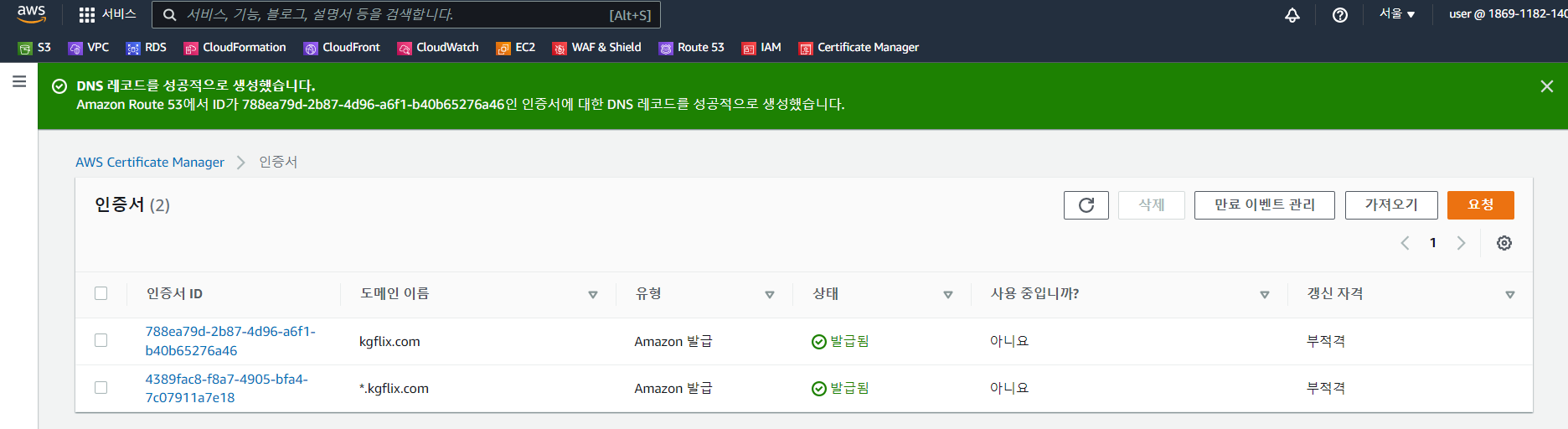
****

****

****

****

****

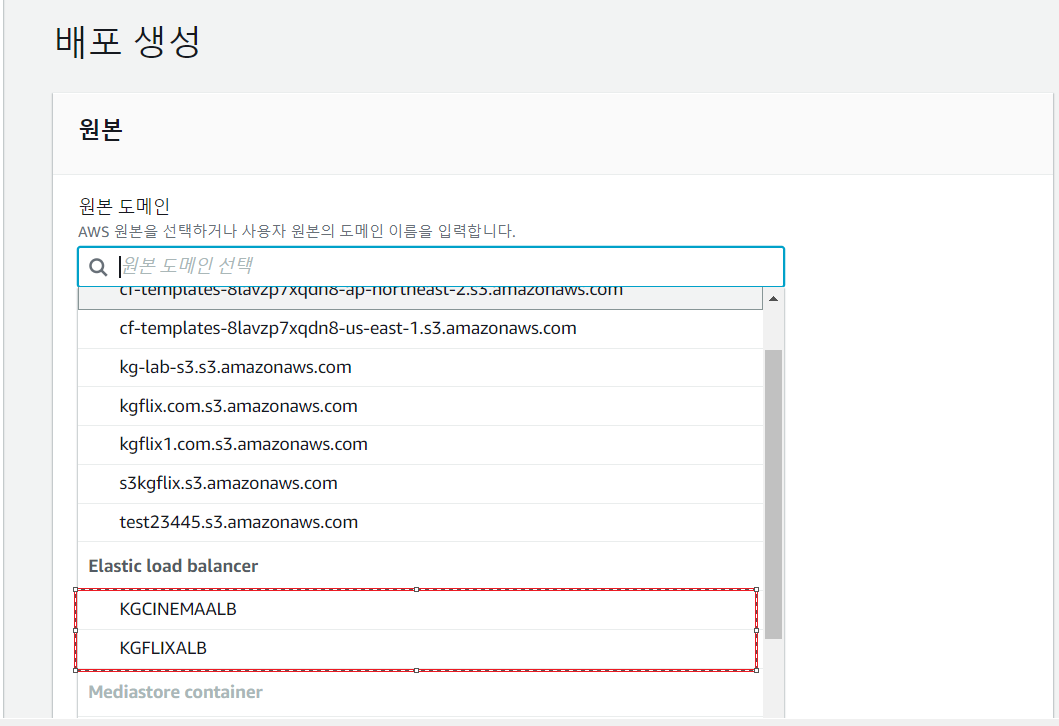
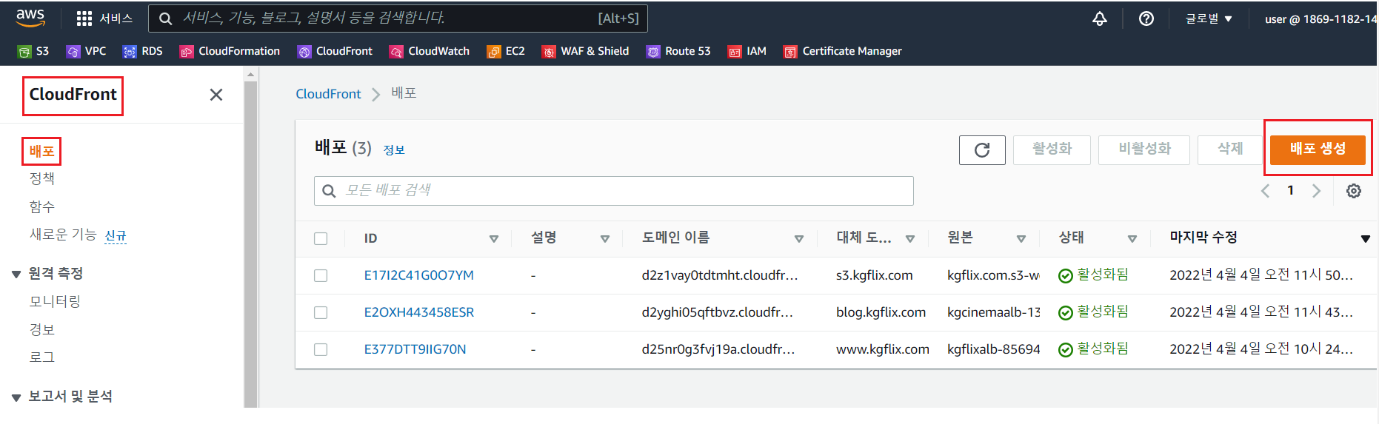
****

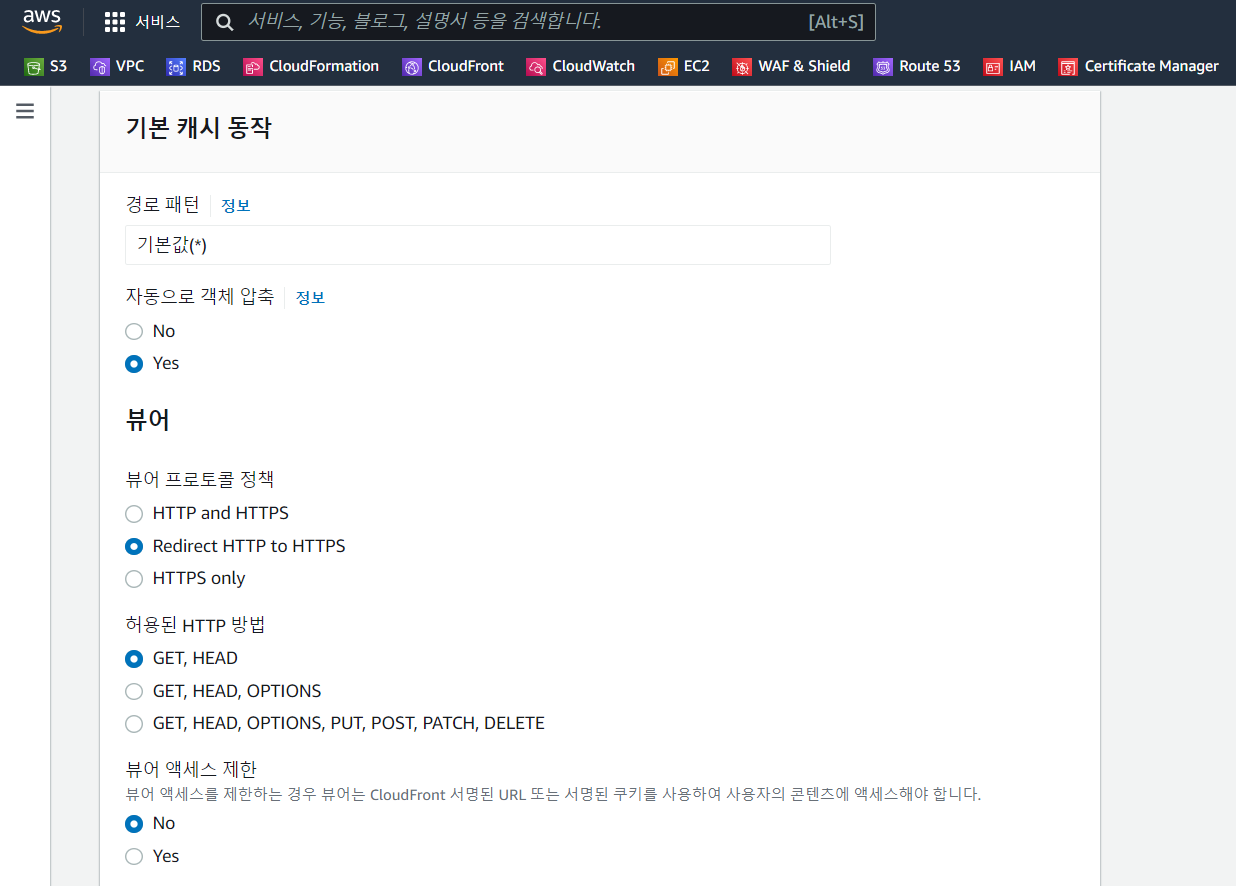
**3. CloudFront ( ELB )**

**CloudFront -> 배포 -> 배포 생성 -> ELB중 1개 설정 -> 뷰어 프로토콜 설정 ->**

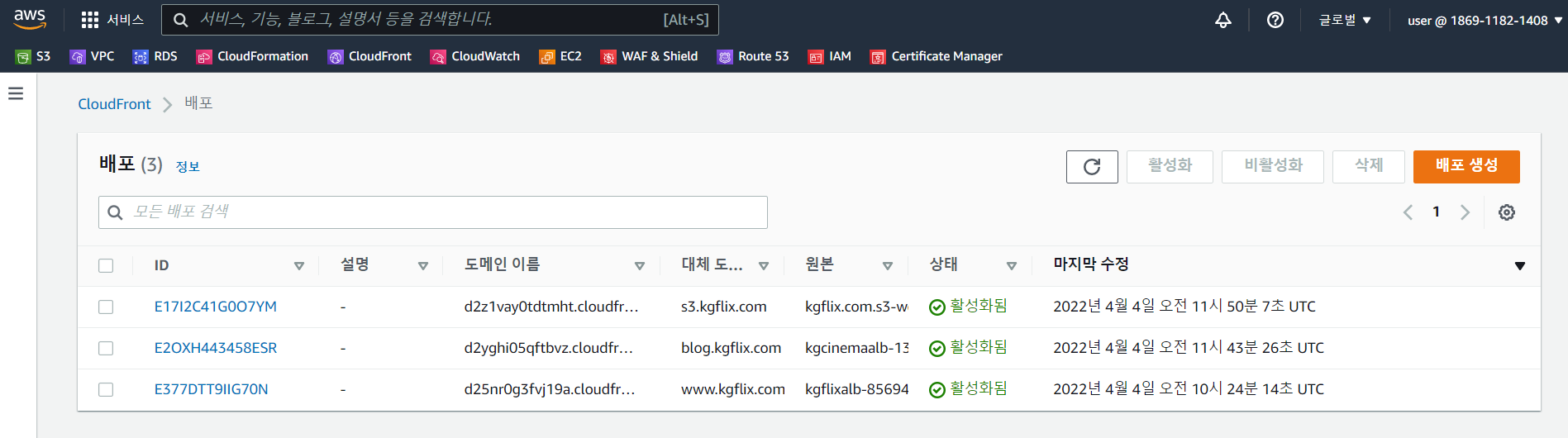
**사용자 정의 인증서 선택 -> 배포생성 -> 배포 생성완료**

**※ 같은 방법으로 나머지 ELB중 1개 설정**

****

****

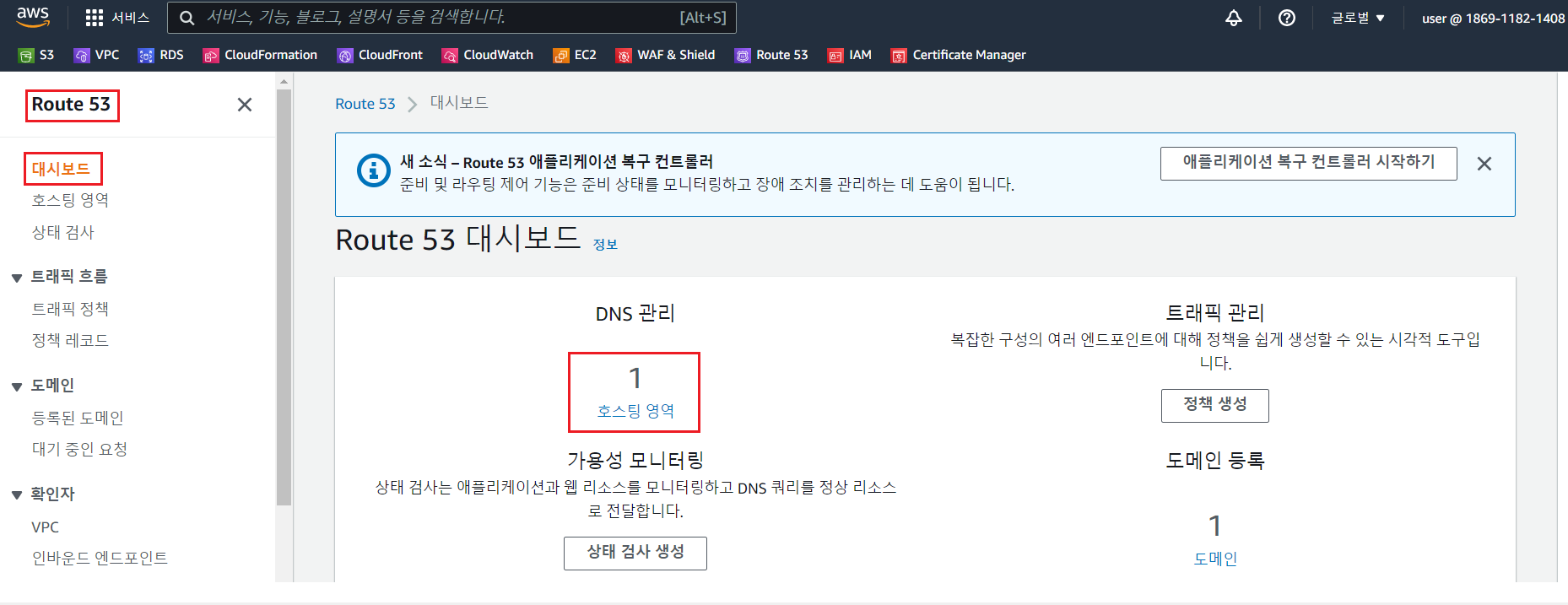
****

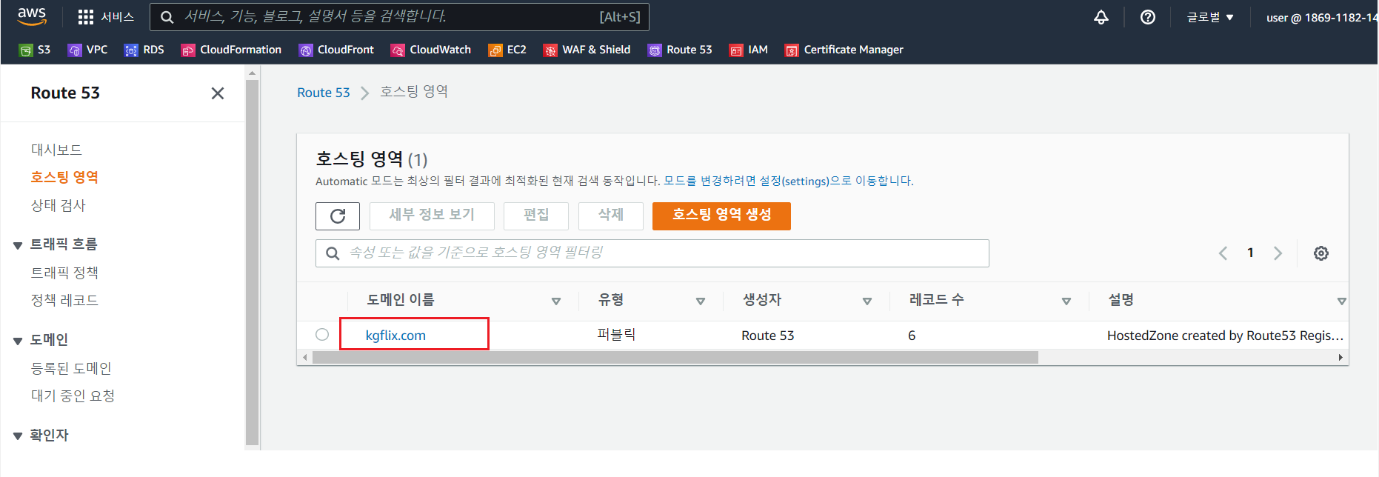
****

**4. Route53**

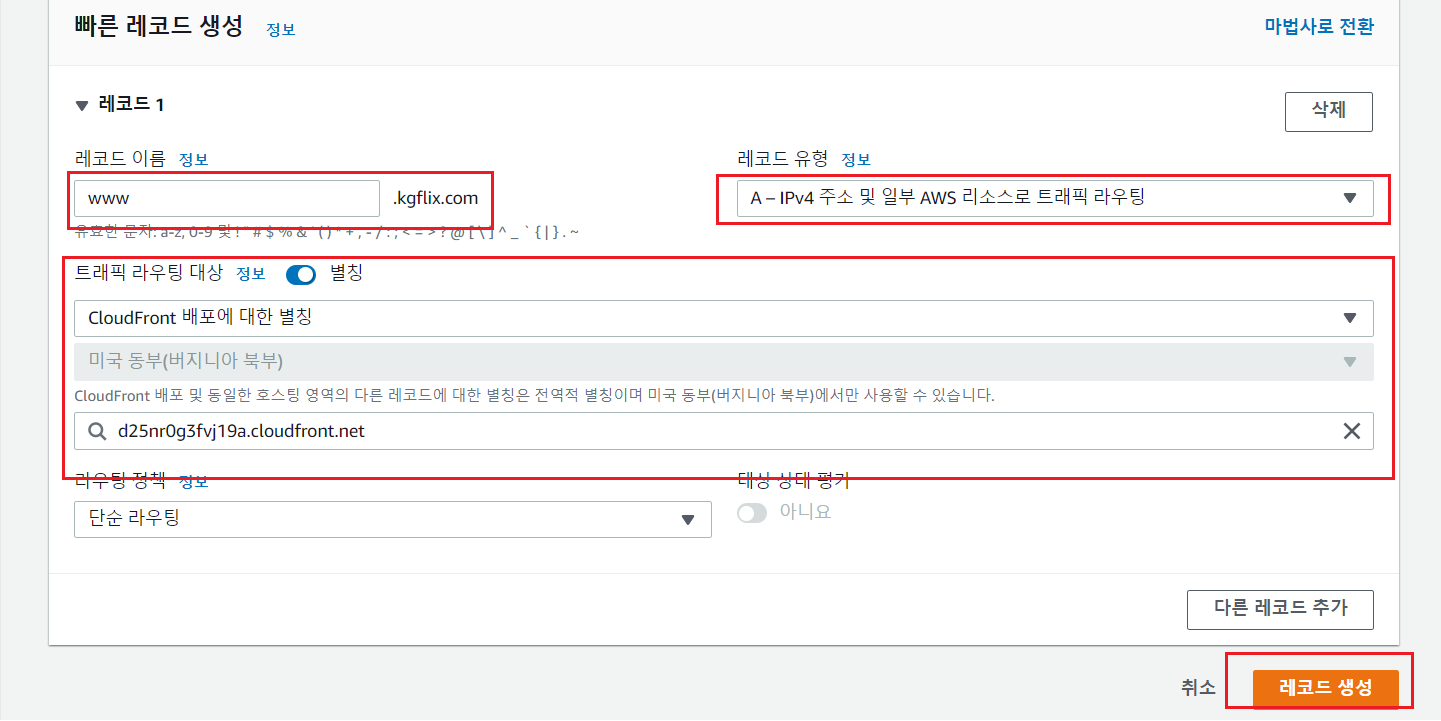
**Route53 -> 대시보드 -> 호스팅 영역 -> 도메인이름 -> 레코드 생성 -> 레코드 이름, 유형 트래픽라우터 설정 ( 별칭체크 ) -> 레코드 생성**

**※ 같은 방식으로 ALB CINEMA도 진행**

****

****

****

****

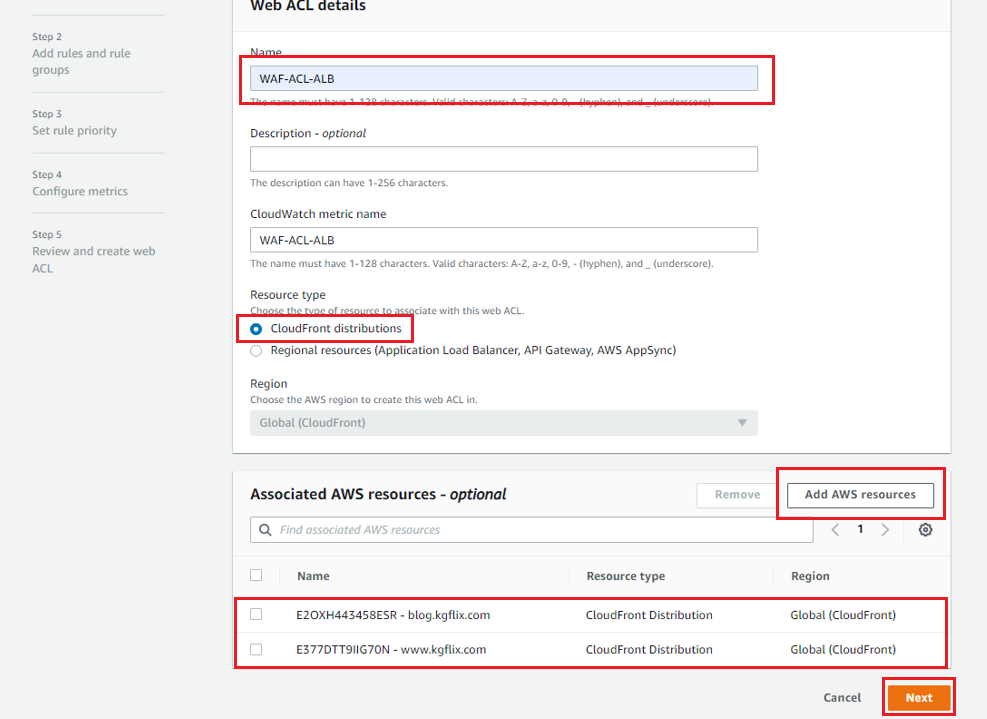
****

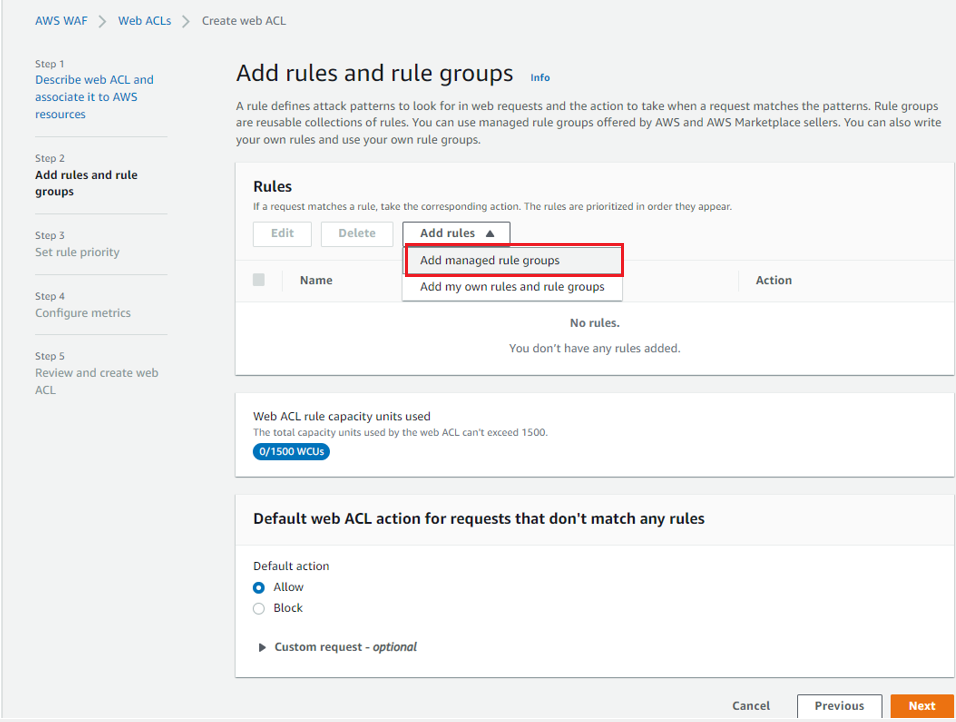
**5. AWS WAF**

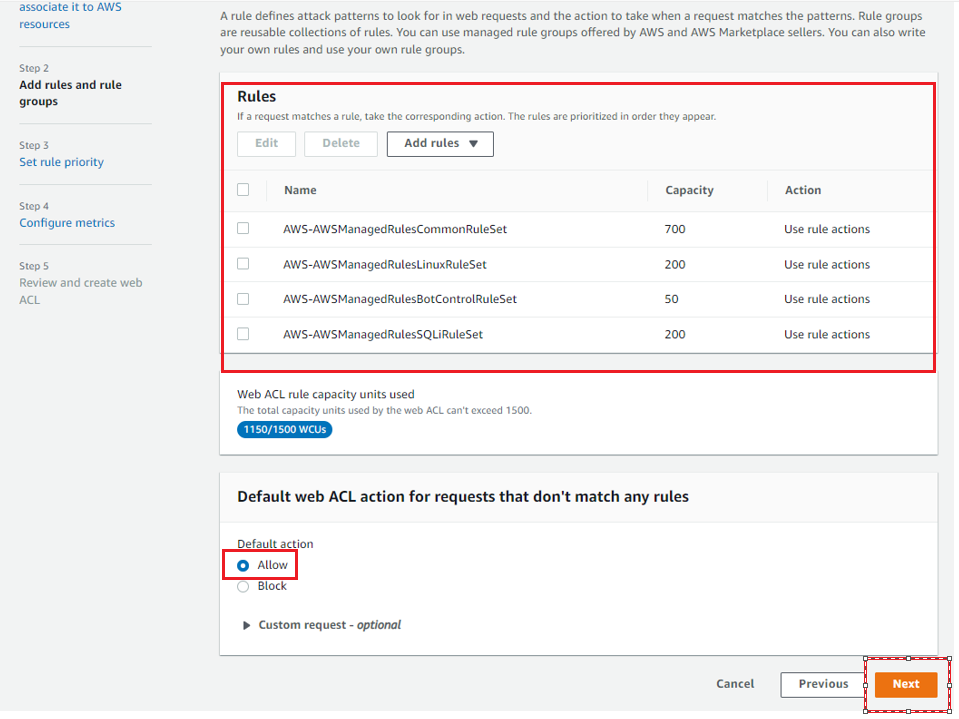
**WAF & Shield -> Create with ACL -> 이름,리소스 타입설정,관련옵션설정 ->**

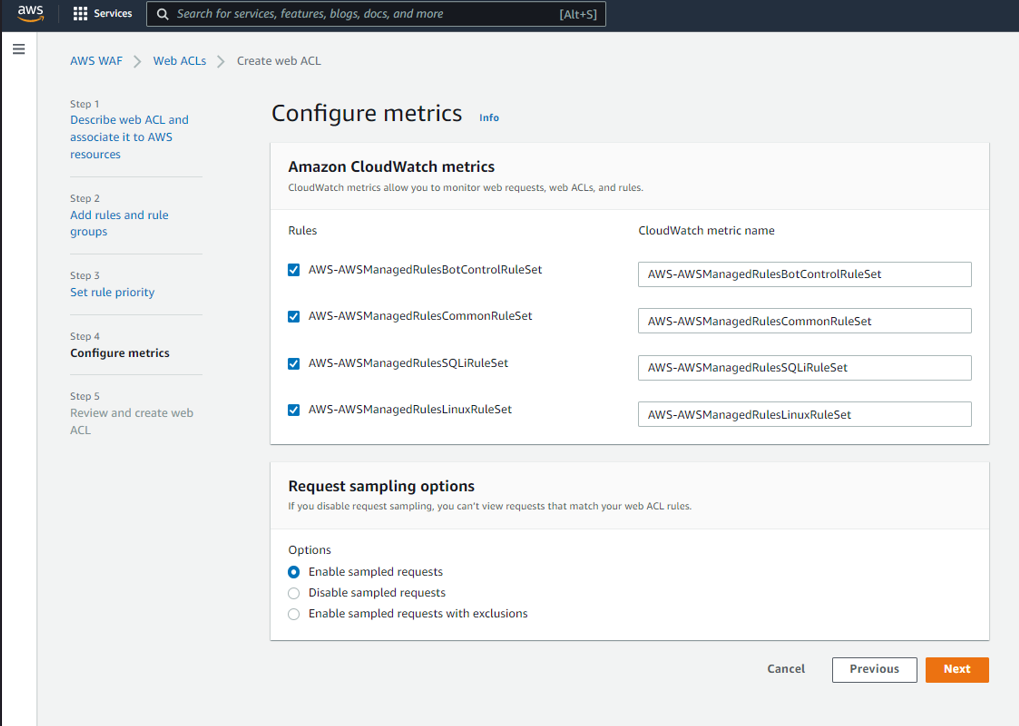
**규칙 추가 -> 액션 디폴트 -> Create with ACL -> 설정완료**

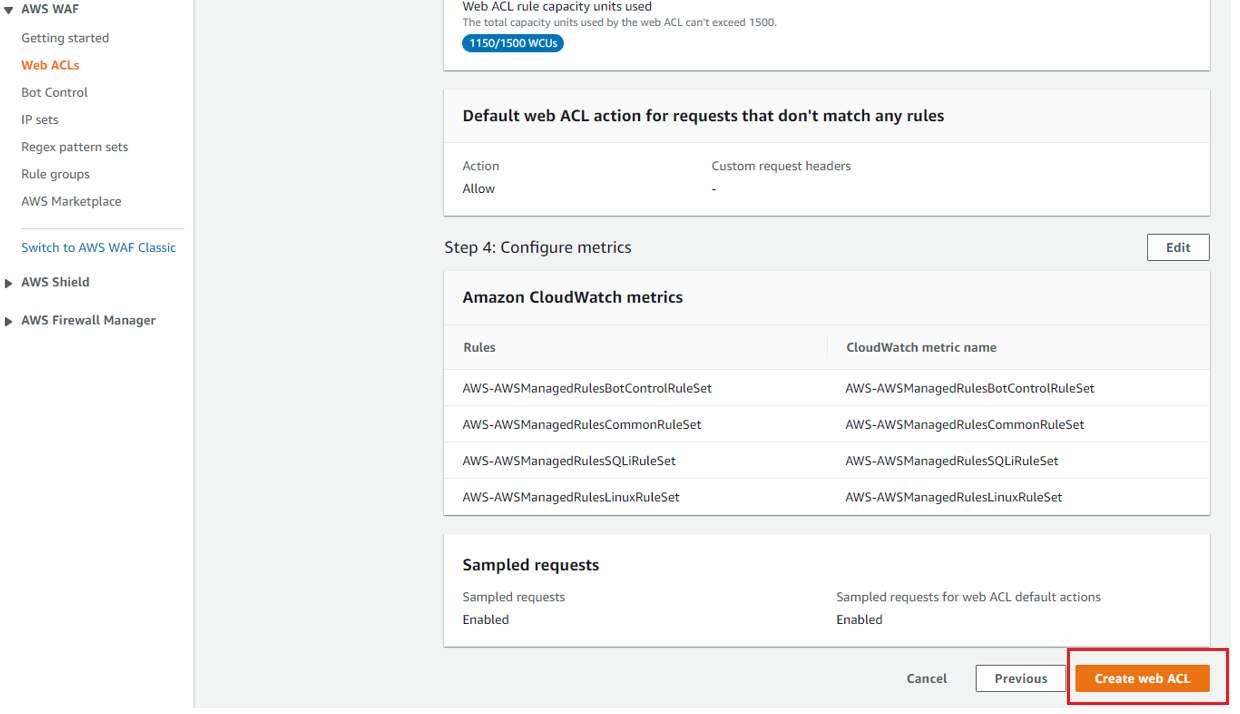
****

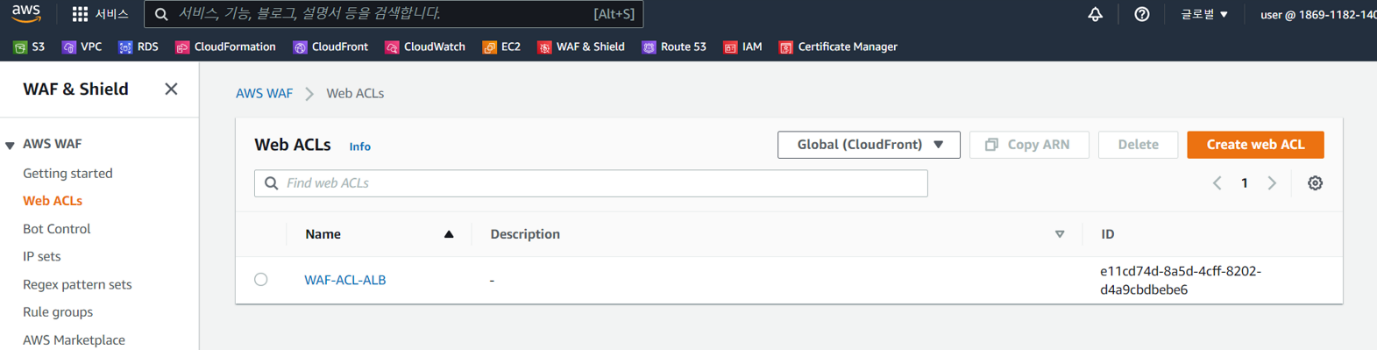
****

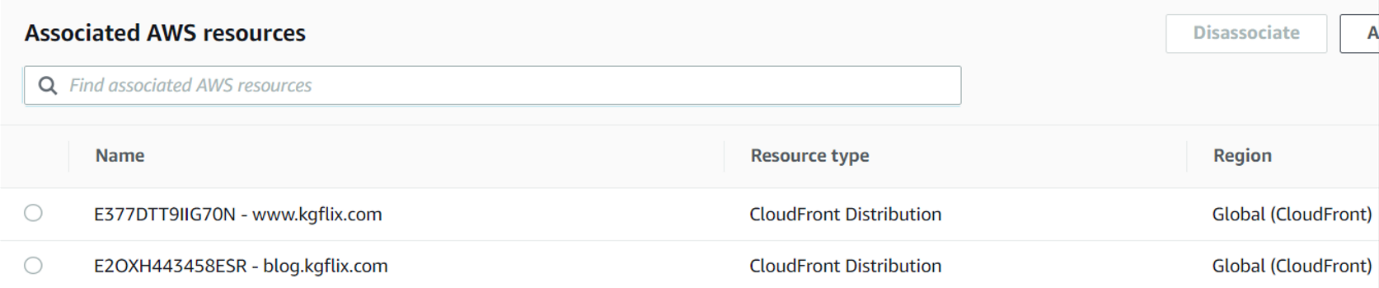
****

****

****

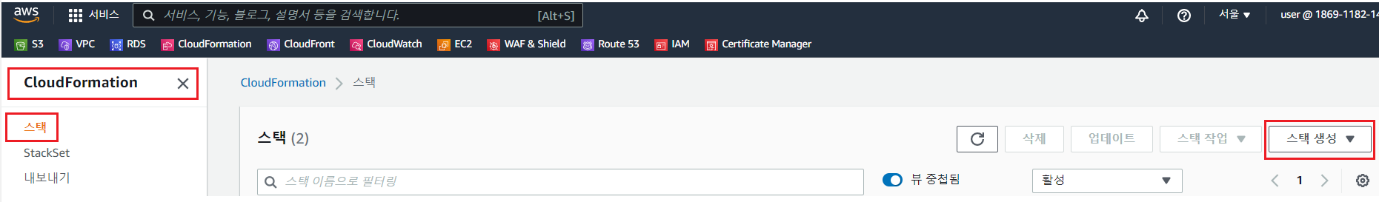
****

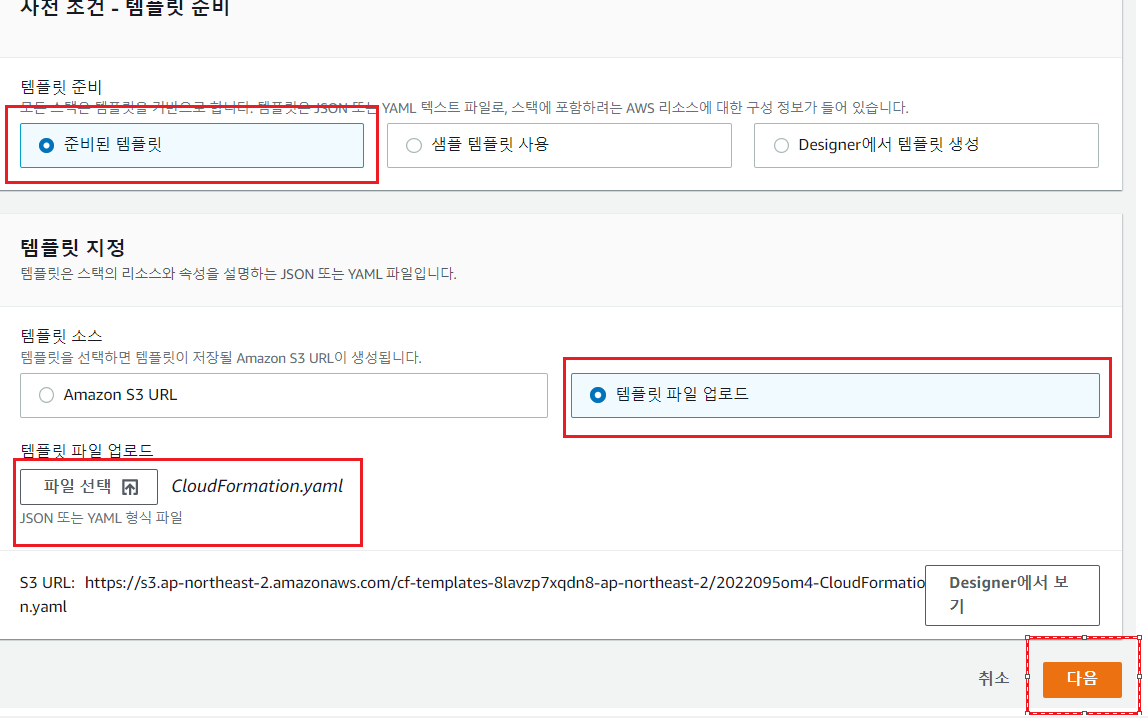
****

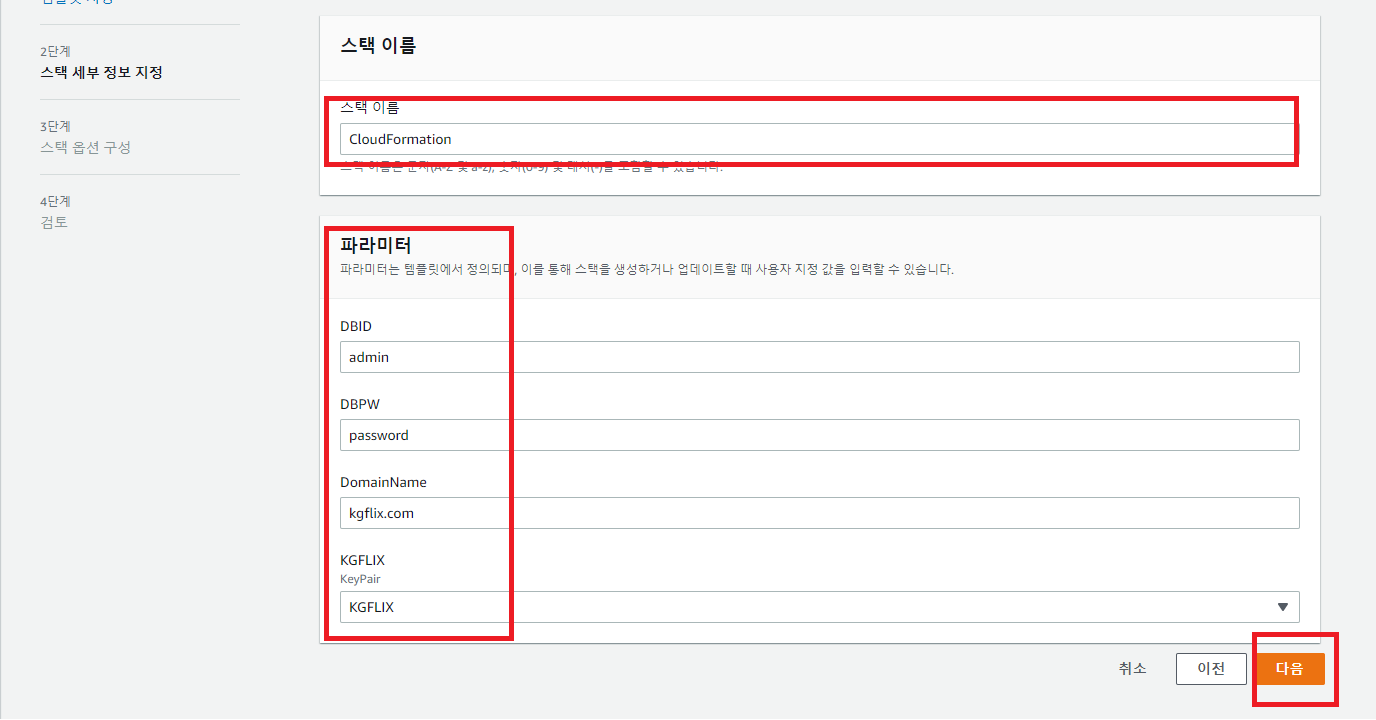
****

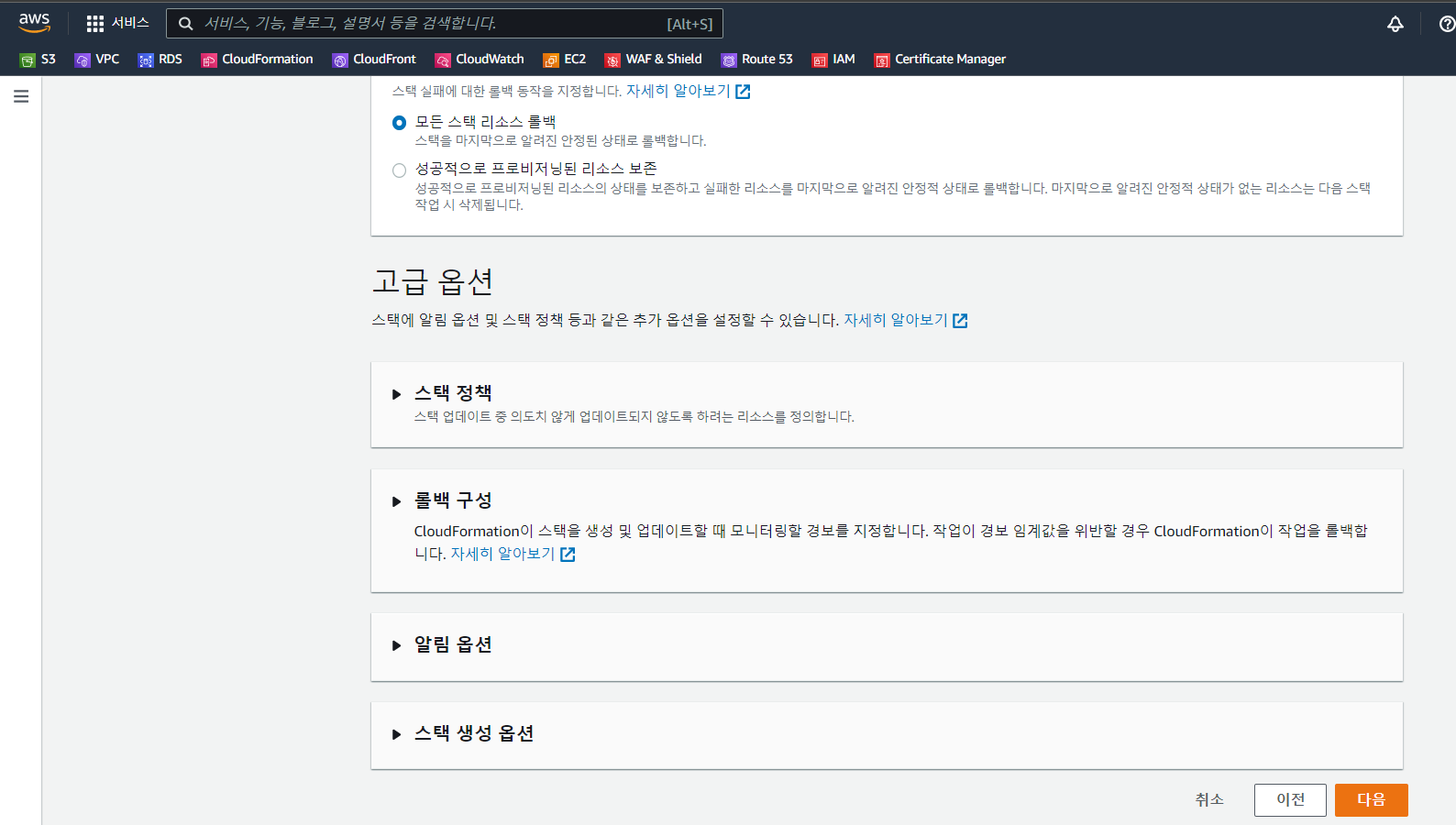
**6. CloudFormation**

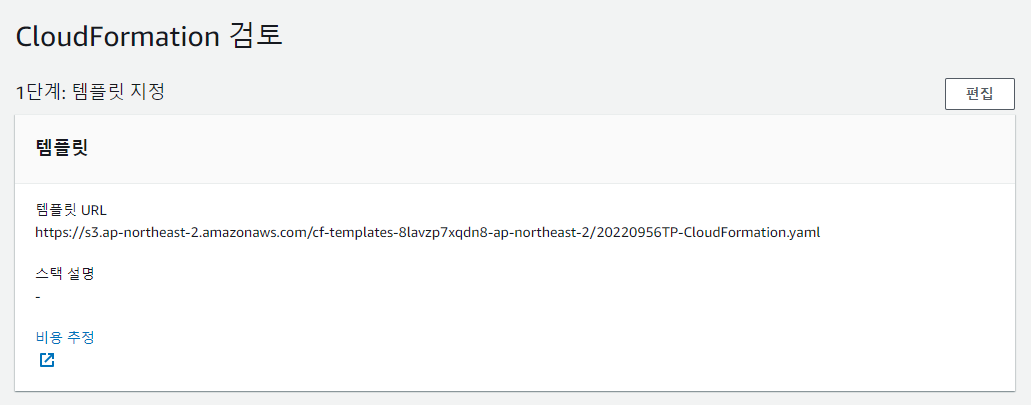
**CloudFormation -> 스택생성 -> 새 리소스 사용( 표준 ) -> 템플릿 업로드 -> 스택이름 설정 -> 필요한 파라미터 입력 -> 생성 - 스택 생성완료**

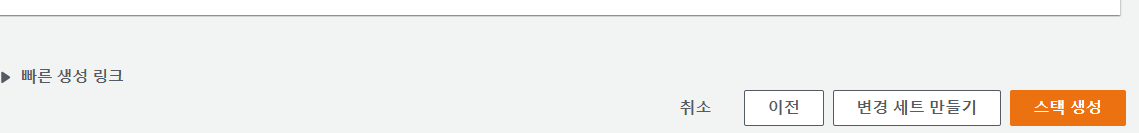
****

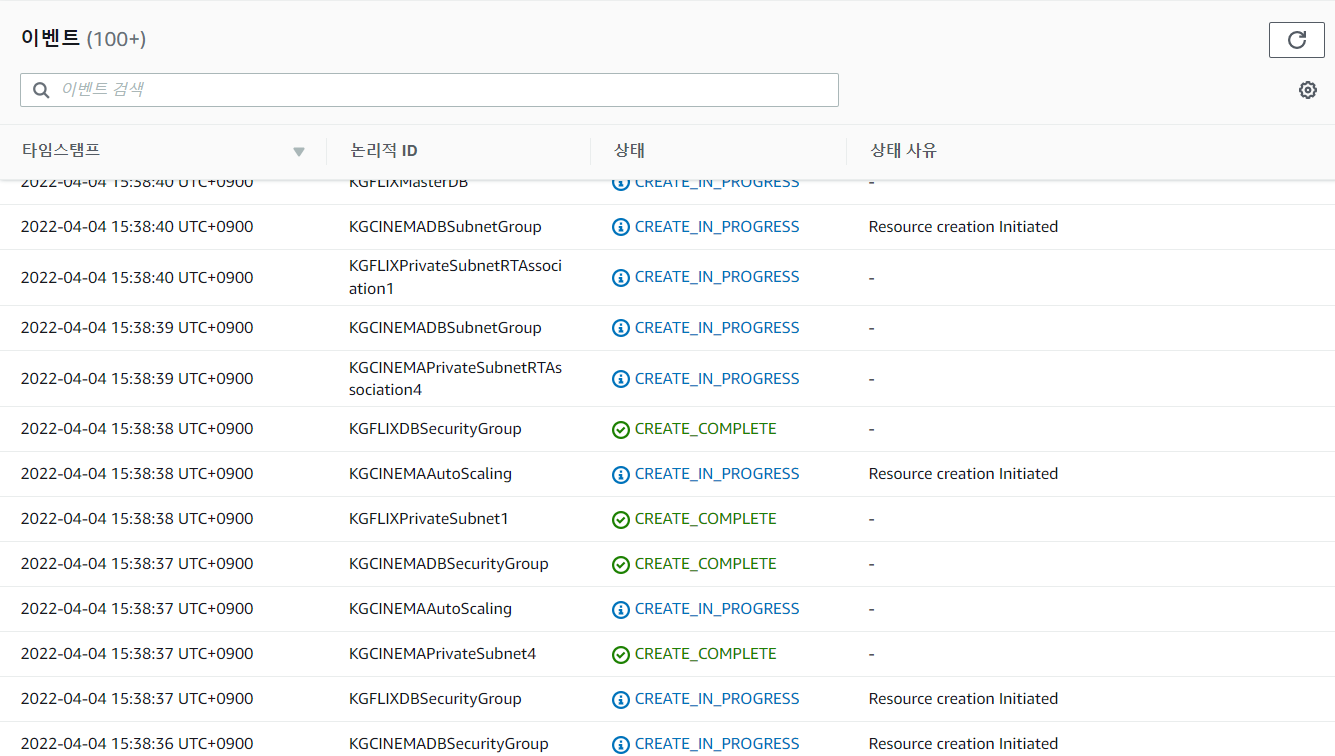
****

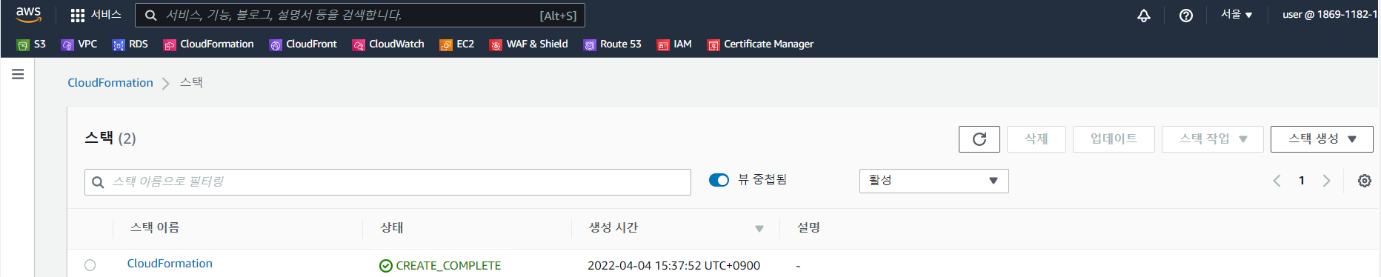
****

****

****

****

****

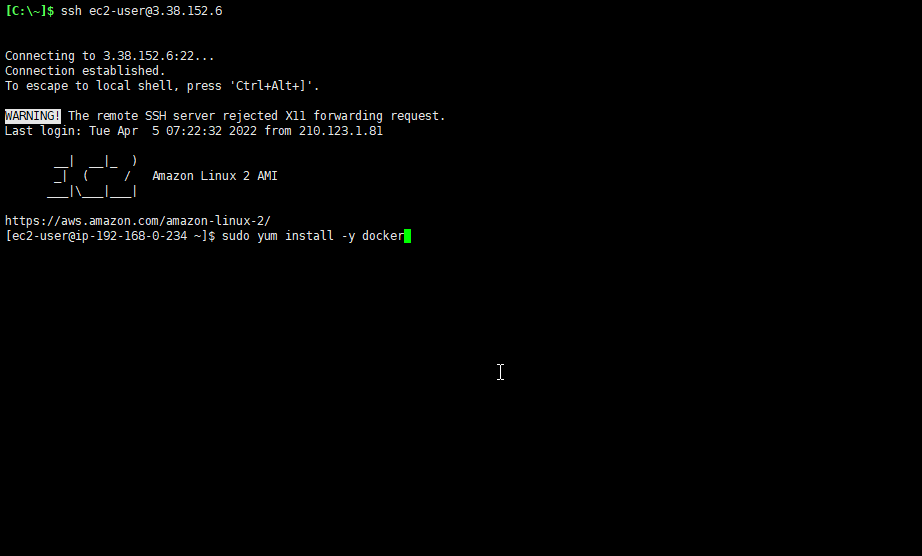
****

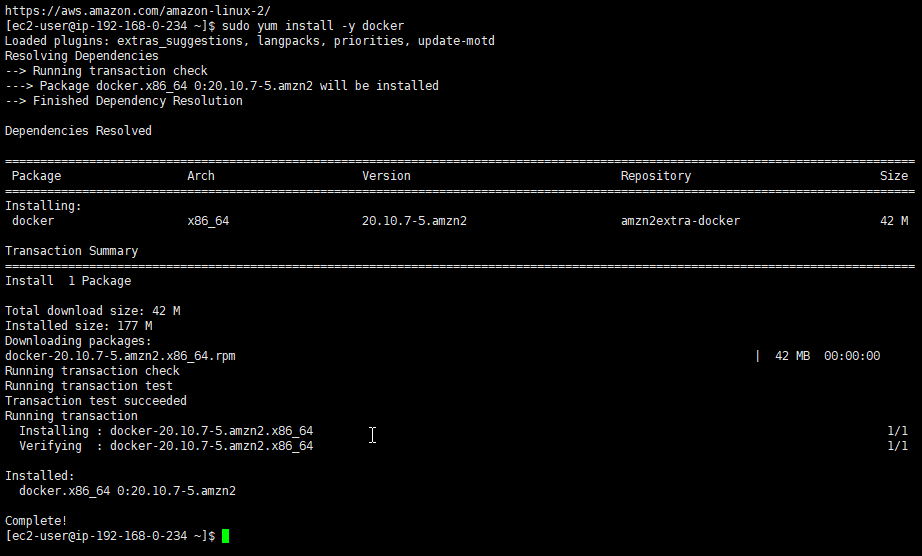
**7. Monitoring**

**# ssh [ec2-user@3.38.152.6](mailto:ec2-user@3.38.152.6)**

****

**# sudo yum install -y docker**

****

****

**# sudo mkdir /Prometheus**

**# sudo vi /Prometheus/prometheus.yml**

**global:**

**scrape\_interval: 15s**

**evaluation\_interval: 15s**

**alerting:**

**alertmanagers:**

**- static\_configs:**

**- targets:**

**# - alertmanager:9093**

**rule\_files:**

**# - "first\_rules.yml"**

**# - "second\_rules.yml"**

**scrape\_configs:**

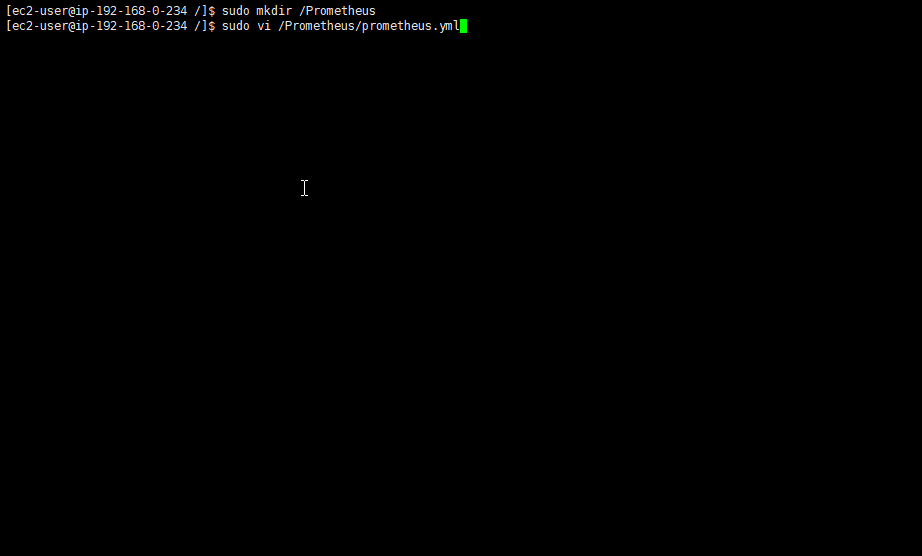
**- job\_name: 'prometheus'**

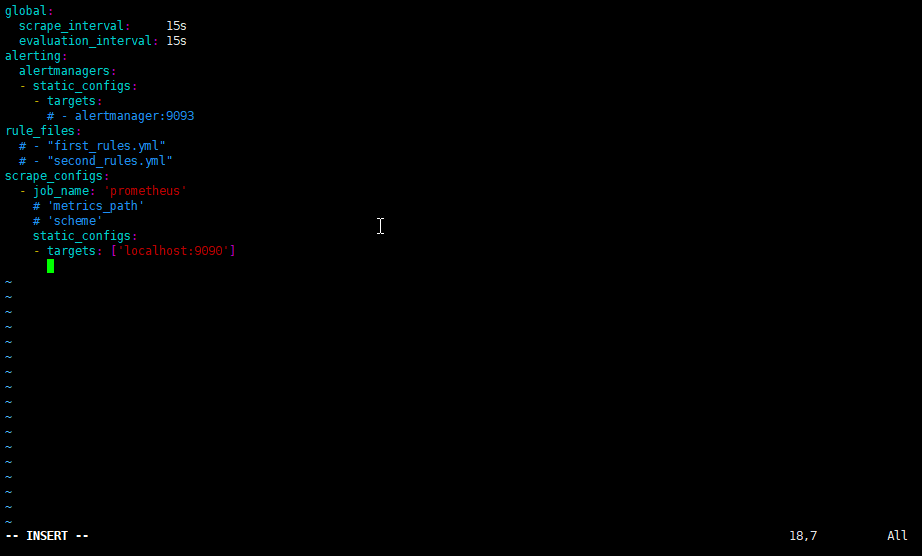
**# 'metrics\_path'**

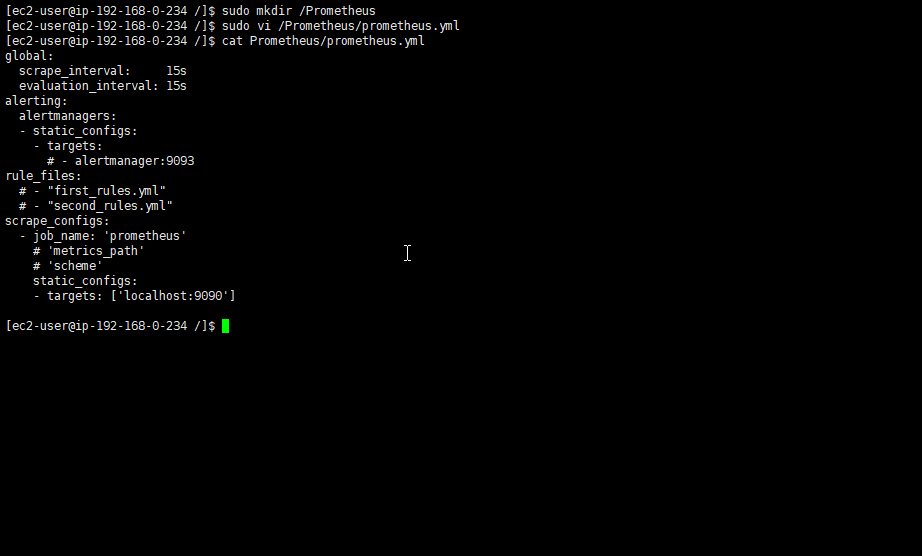
**# 'scheme'**

**static\_configs:**

**- targets: ['localhost:9090']**

****

****

****

**# sudo systemctl enable docker**

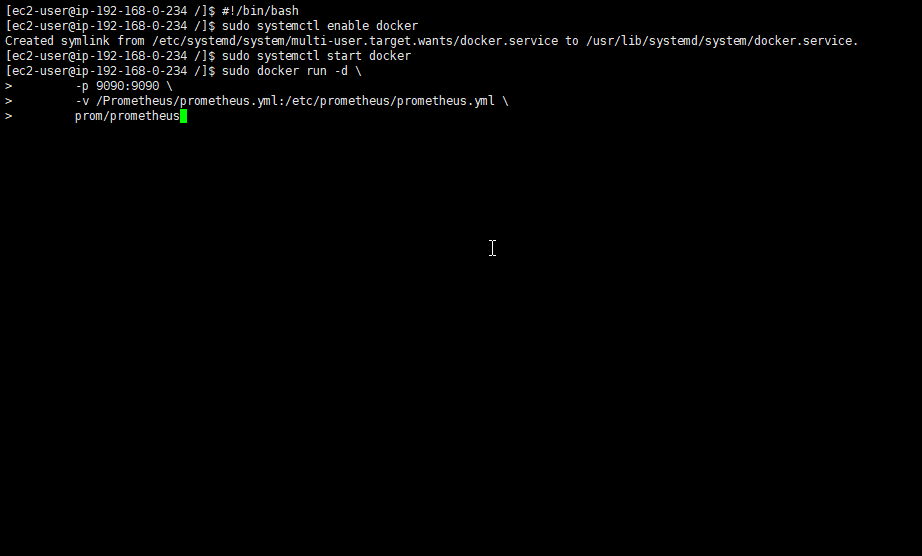
**# sudo systemctl start docker**

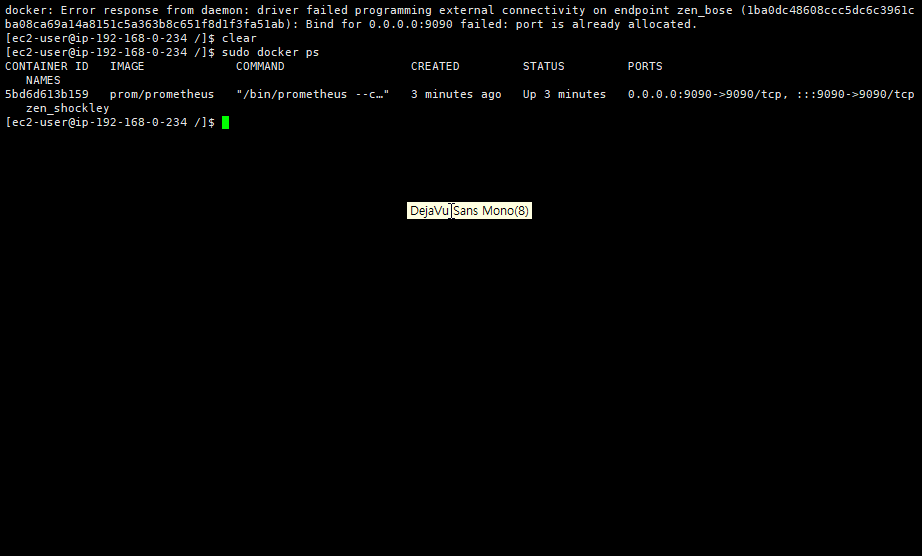
**# sudo docker run -d \**

**-p 9090:9090 \**

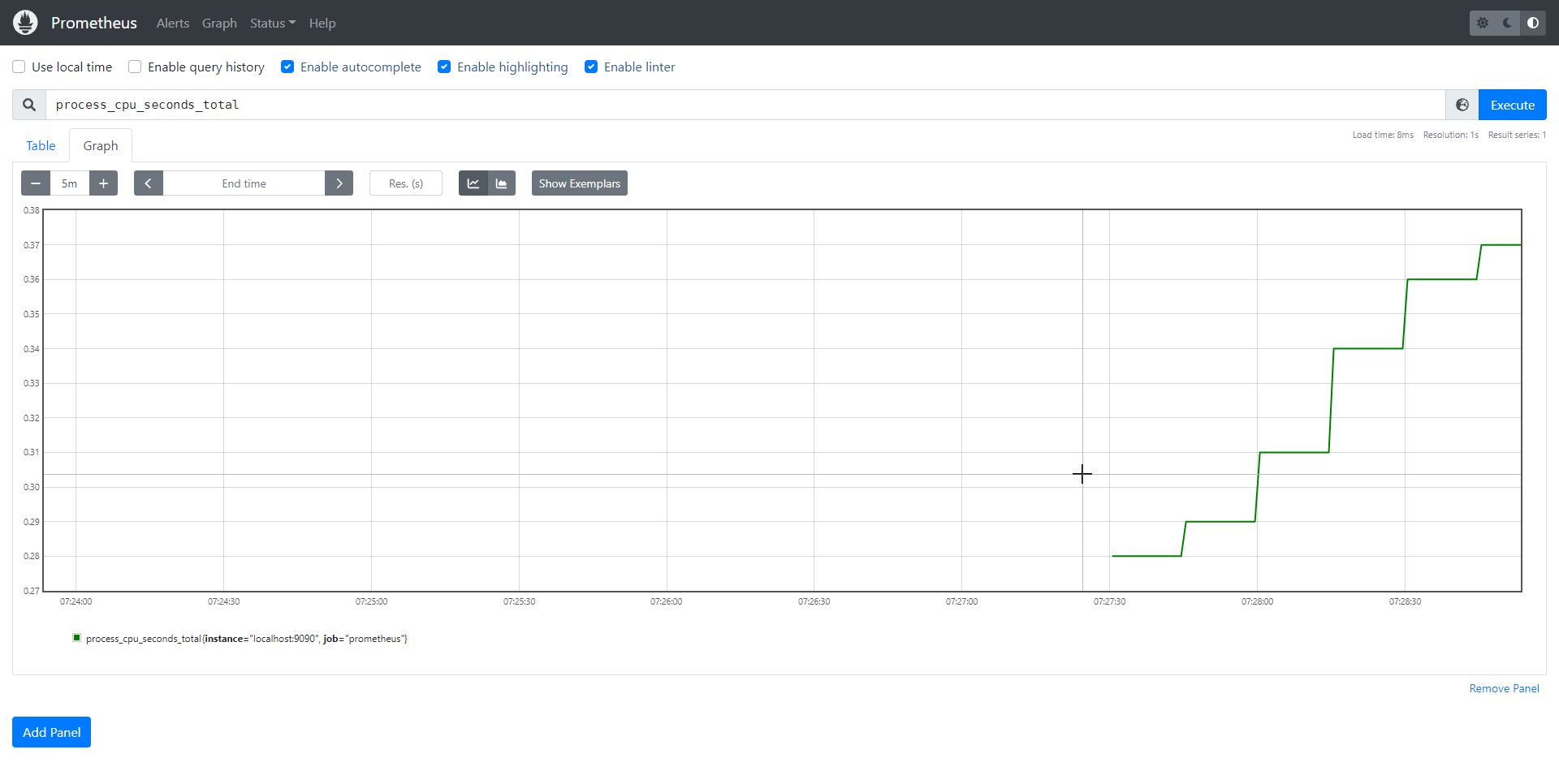
**-v /Prometheus/prometheus.yml:/etc/prometheus/prometheus.yml \**

**prom/prometheus**

****

****

**브라우저를 열어 [http://3.38.152.6:9090] 정상적으로 데이터를 가져오는지 확인**

****

**# sudo docker pull prom/node-exporter**

**# sudo docker run -d -p 9100:9100 prom/node-exporter**

**# sudo vi /Prometheus/prometheus.yml**

**global:**

**scrape\_interval: 15s**

**evaluation\_interval: 15s**

**# 'scrpae\_timeout'**

**# Alertmanager 설정**

**alerting:**

**alertmanagers:**

**- static\_configs:**

**- targets:**

**# - alertmanager:9093**

**rule\_files:**

**# - "first\_rules.yml"**

**# - "second\_rules.yml"**

**scrape\_configs:**

**- job\_name: 'prometheus'**

**# 'metrics\_path'**

**# 'scheme'**

**static\_configs:**

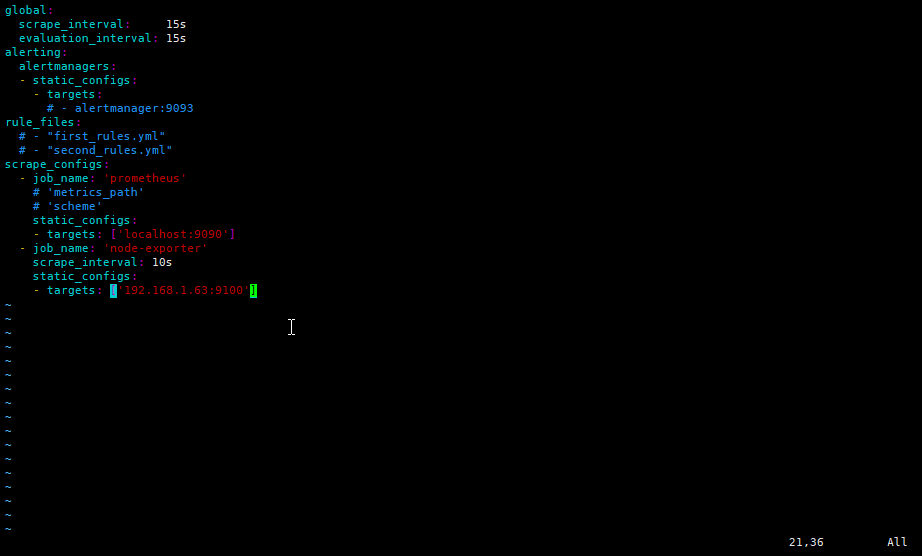
**- targets: ['localhost:9090']**

**- job\_name: 'node-exporter'**

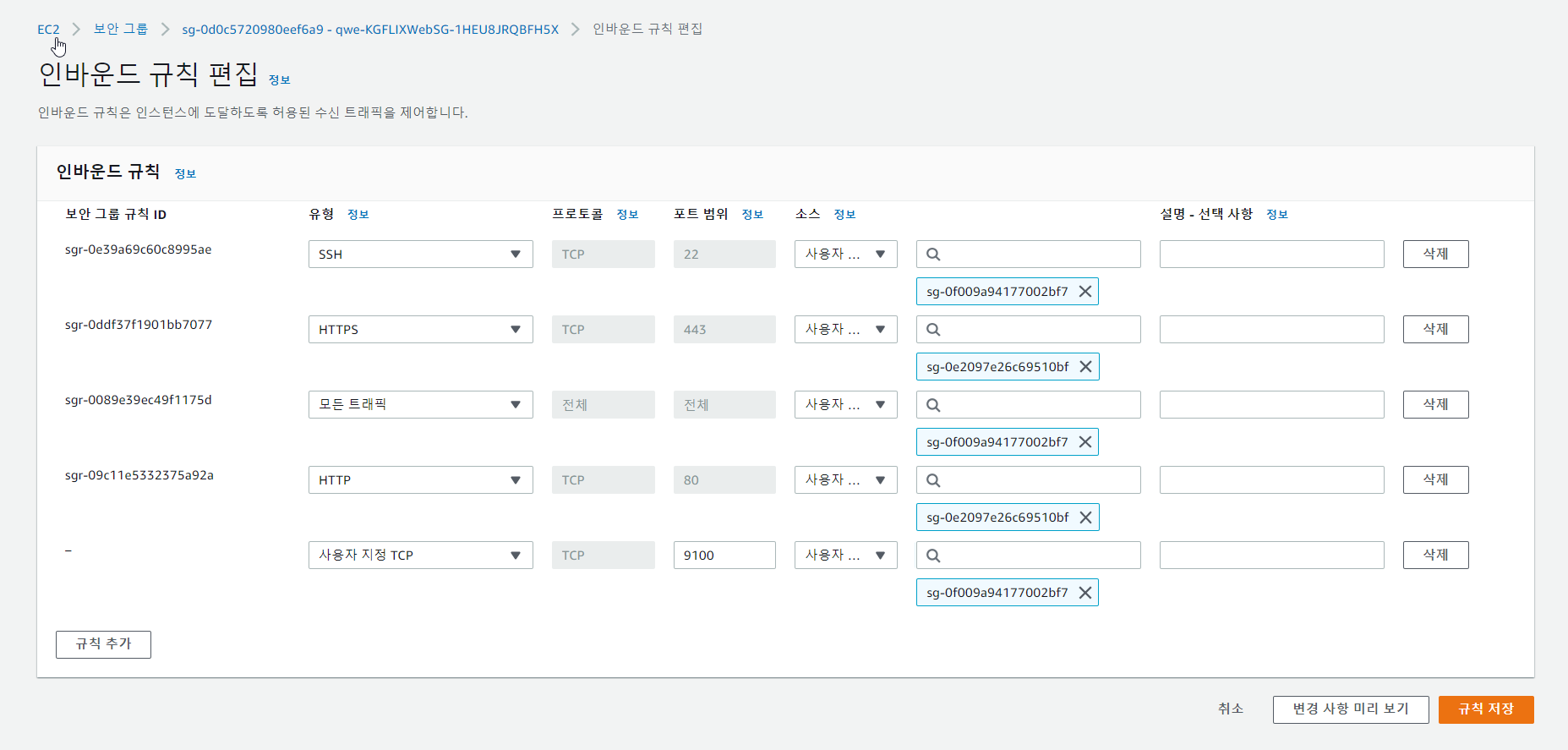
**scrape\_interval: 10s**

**static\_configs:**

**- targets: ['192.168.1.63:9100']**

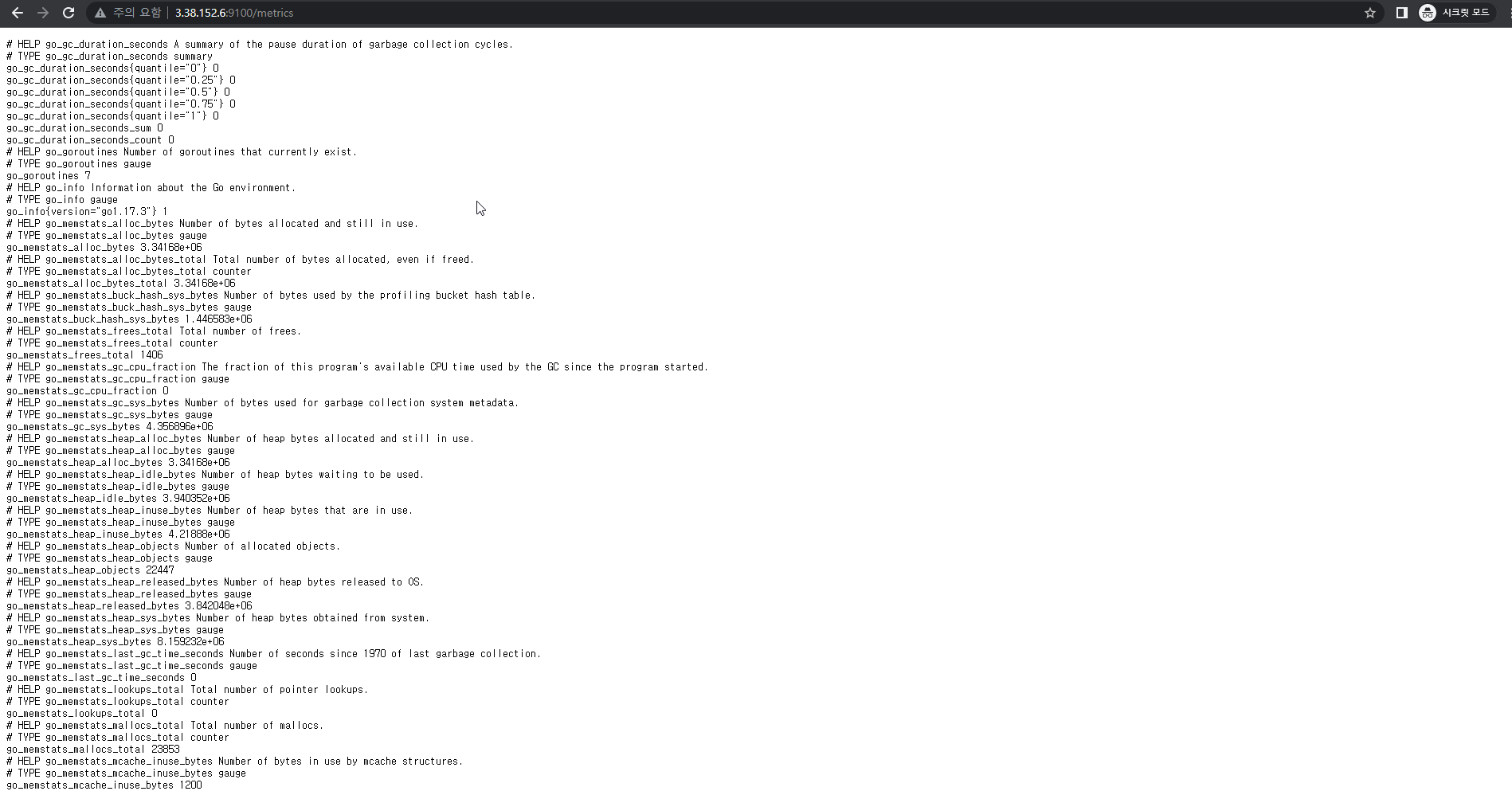
****

**모니터링할 서버의 포트를 열기 위해 인바운드 규칙 추가**

****

**브라우저를 열어 [http://3.38.152.6:9100] 정상적으로 데이터를 가져오는지 확인**

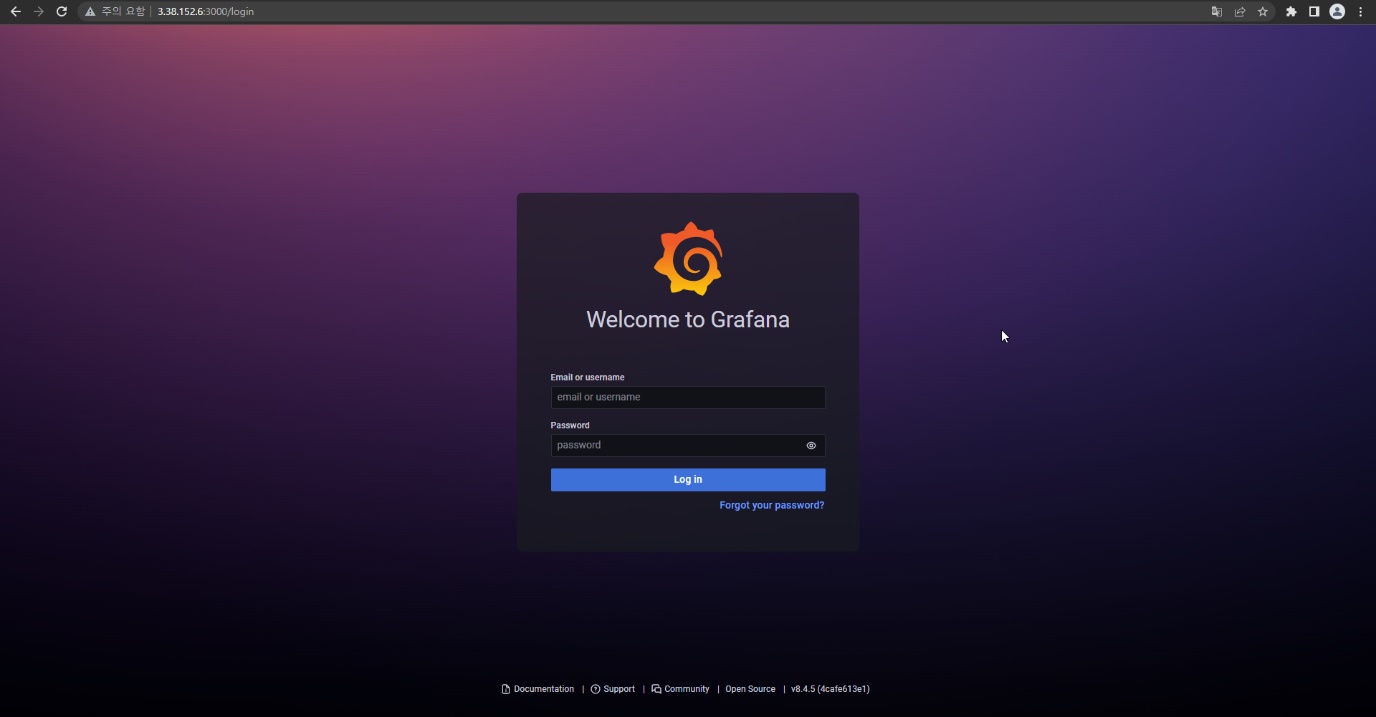
**<http://3.38.152.6:9100/metrics>**

****

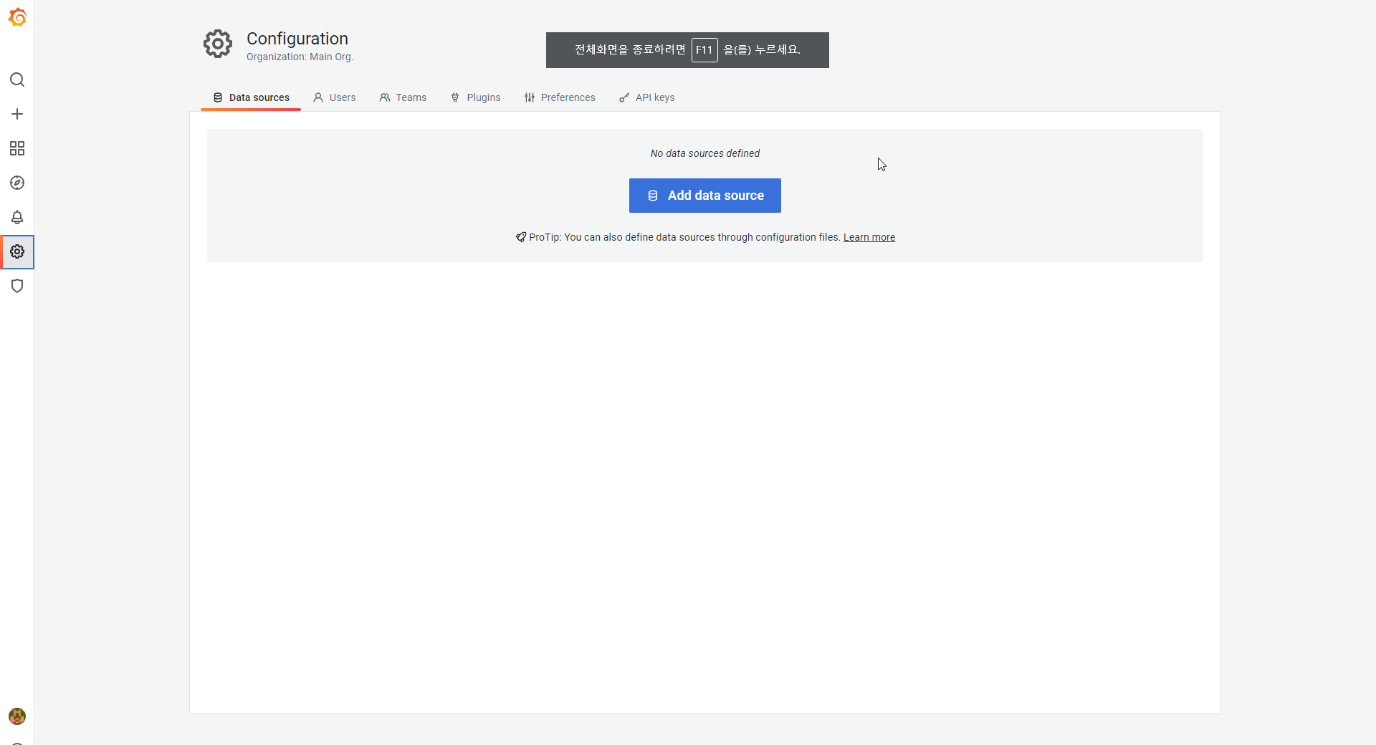
**# sudo docker run -d --name=grafana-server -p 3000:3000 grafana/grafana**

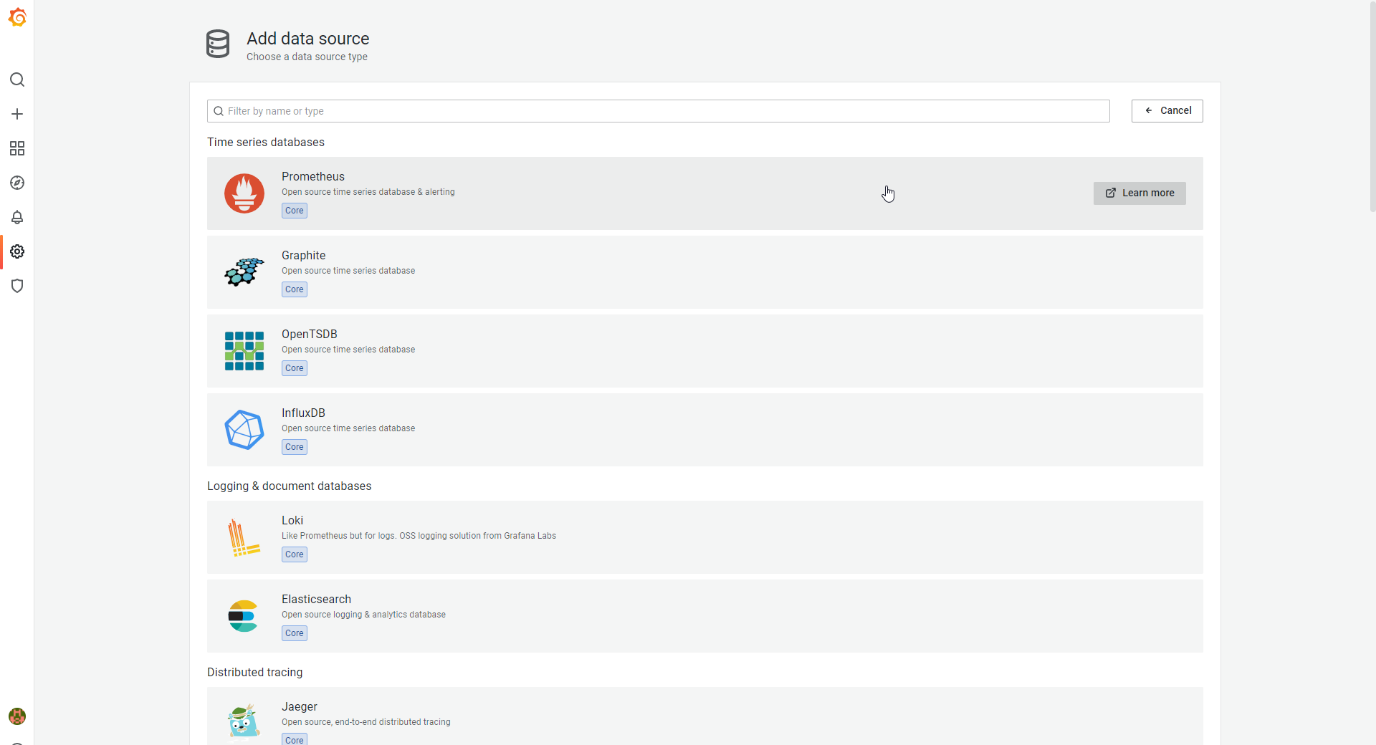
****

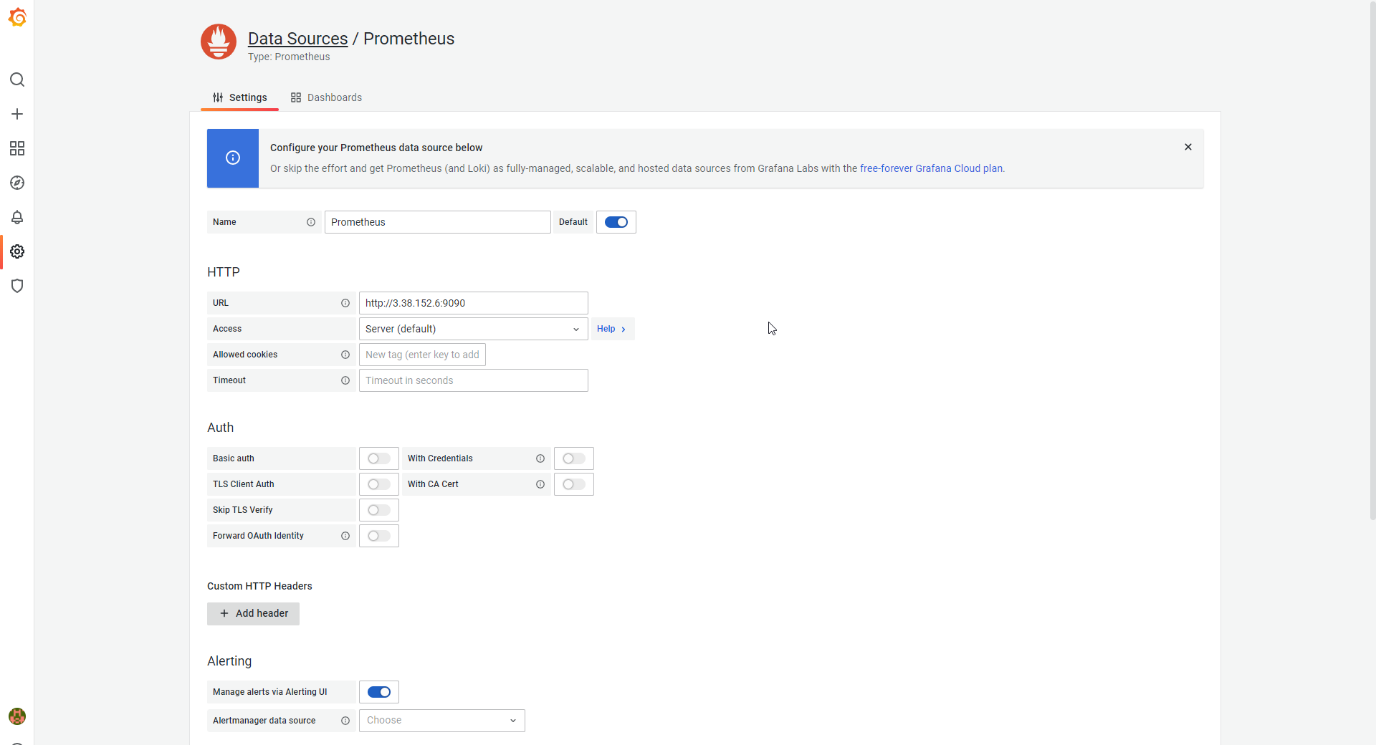
**<http://3.38.152.6:3000/> 으로 접속해 Grafana 로그인**

****

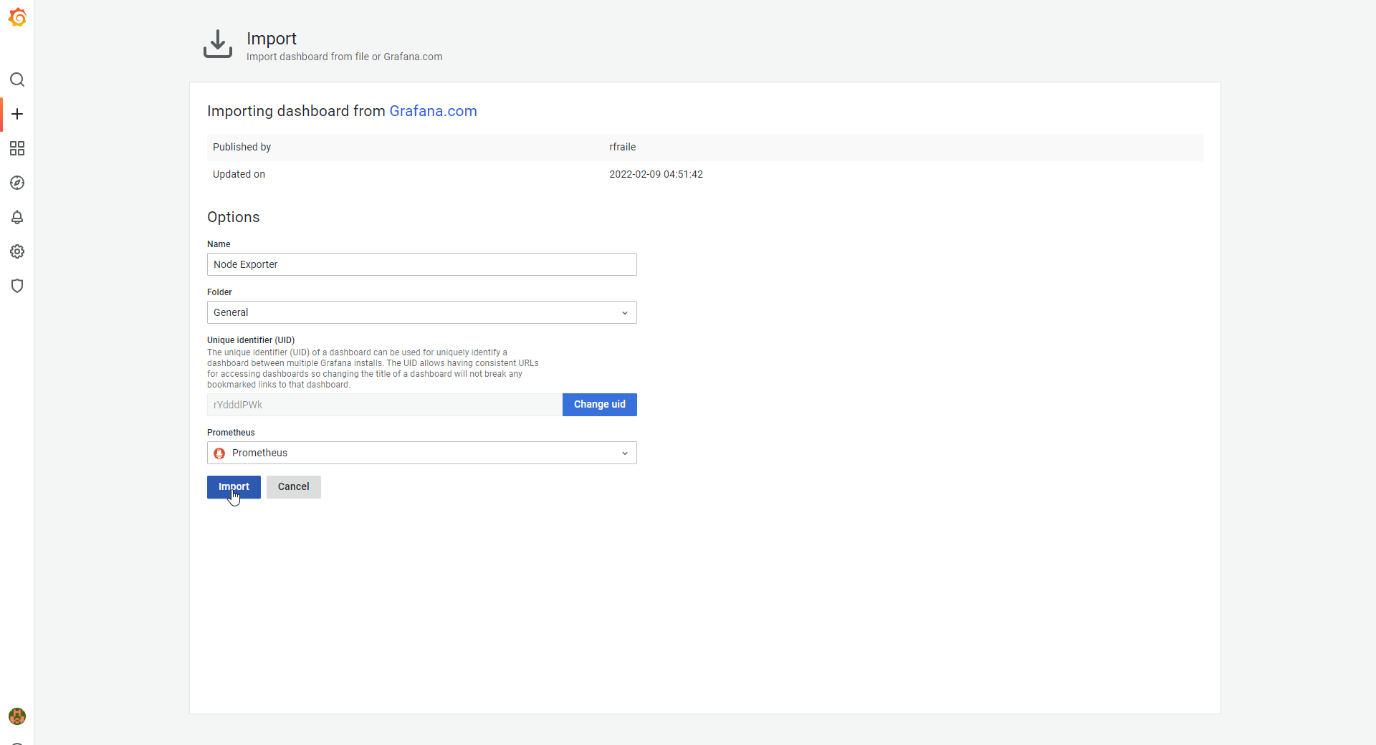
**접속 후 Prometheus data source를 추가**

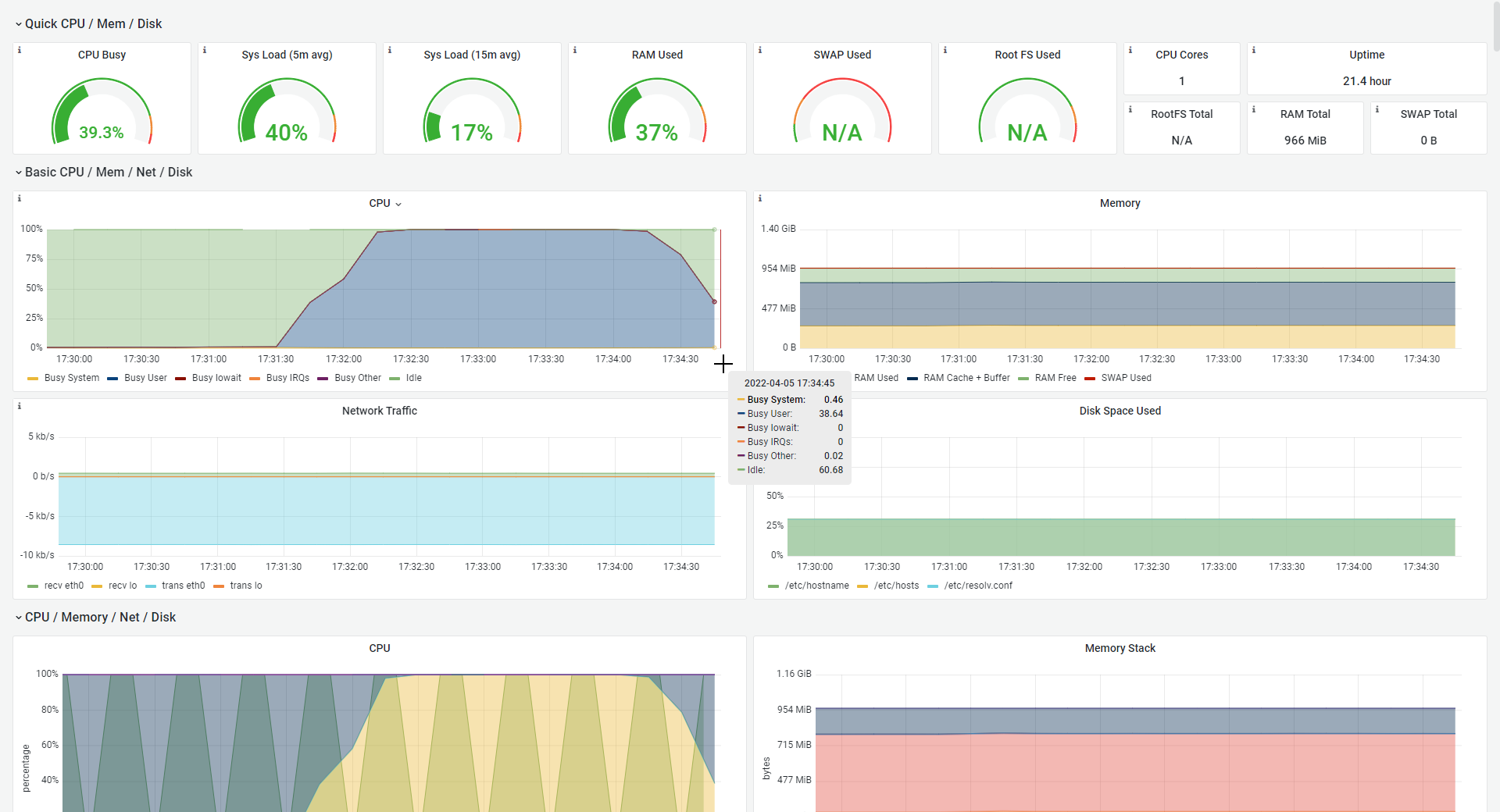
****

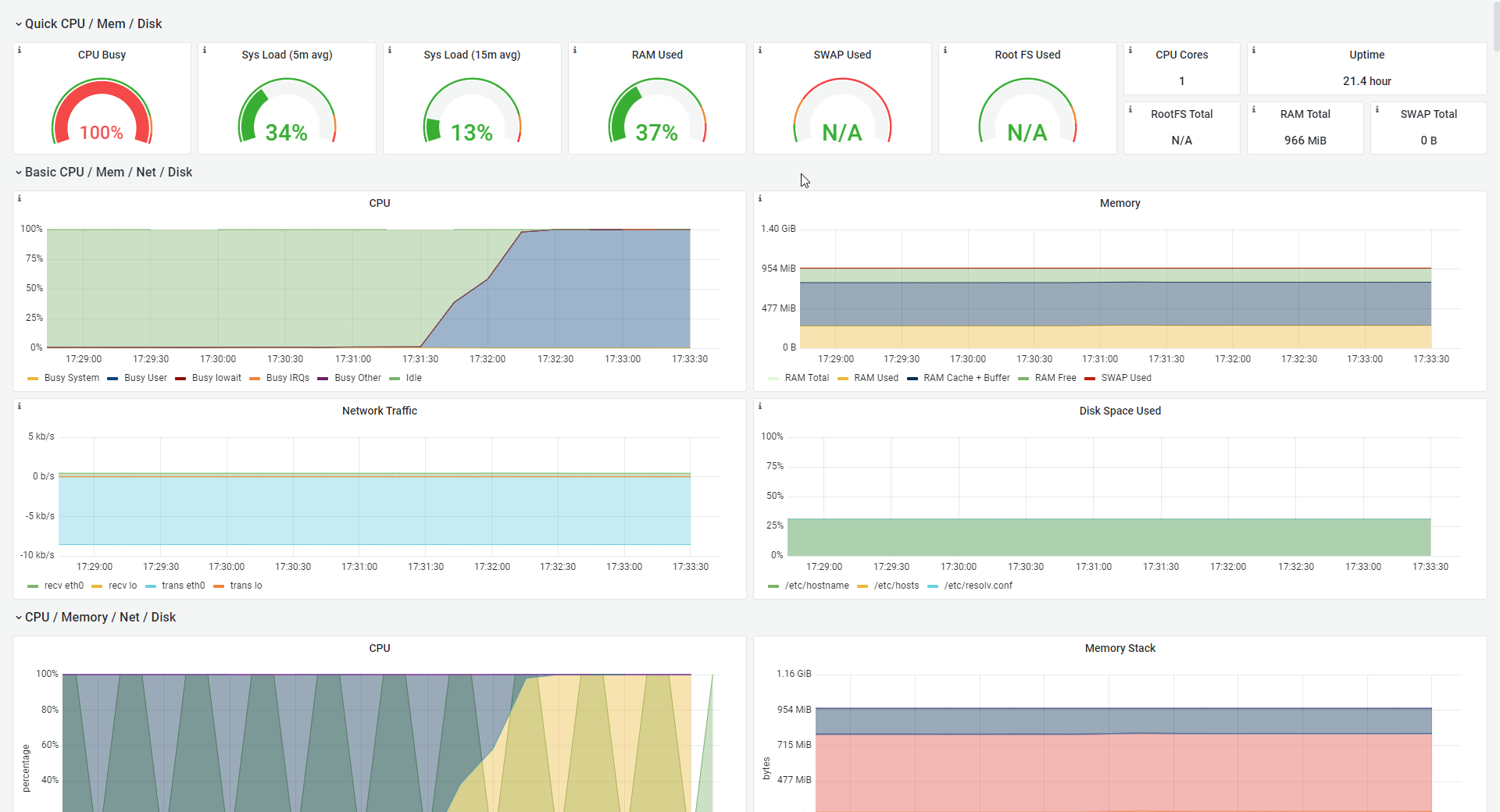
****

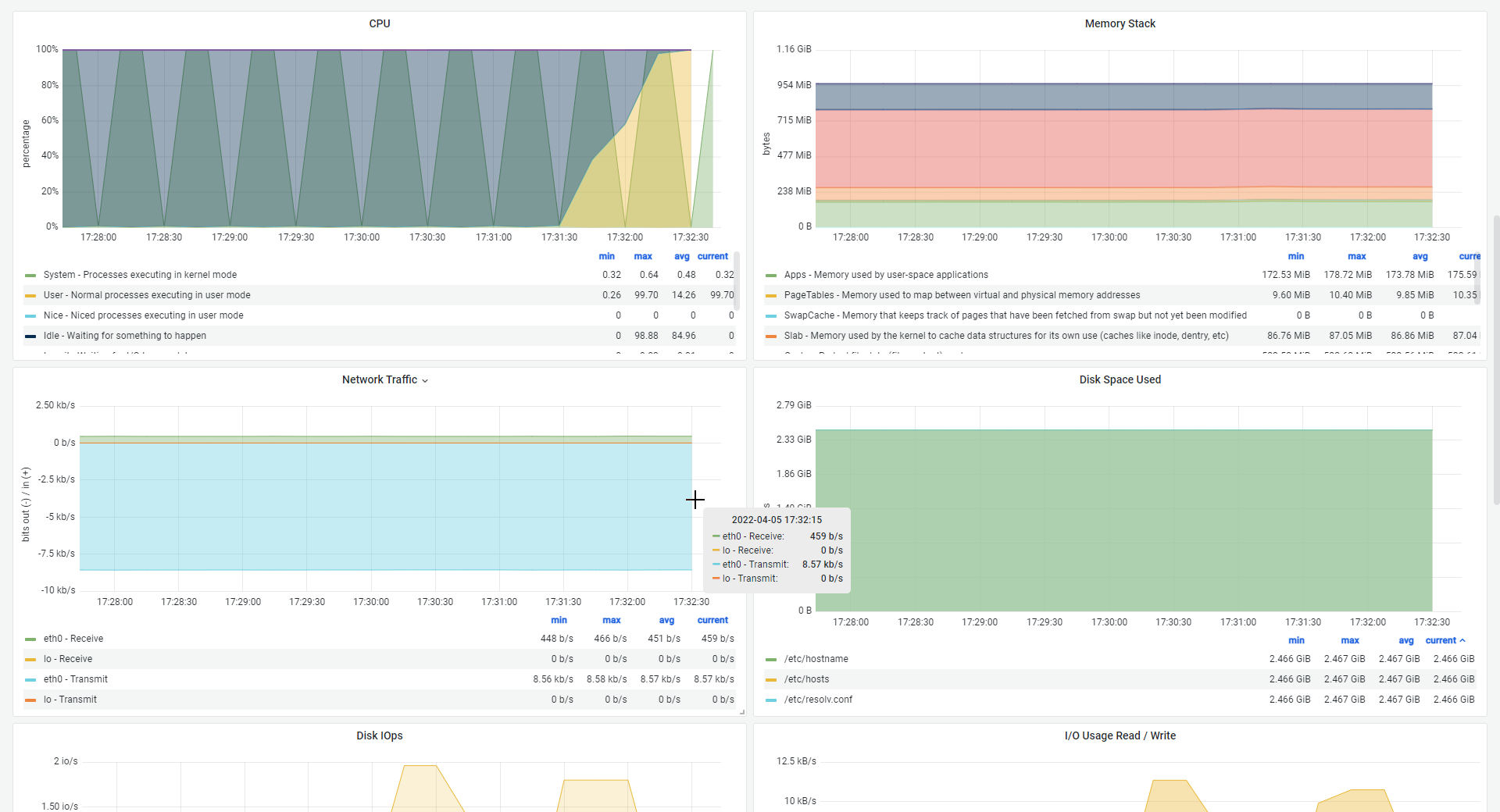
****

**원하는 스타일의 대시보드를 import하고 모니터링 시작**

****

****

****

****