“deploy-a-high-availability-web-app-using-cloudformation”septamber 2022

Advanced Cloud DevOps

Nanodegree Program

Prepared by

Aya RABIH Mostafa

INFORMATION FOR USER

- NAME: Aya Rabih Mostafa

- Project : deploy-a-high-availability-web-app-using-cloud formation

- LINK PROJECT: https://github.com/ayarabih/-project-2---Deploy-a-high-availability-web-app-using-CloudFormation.git

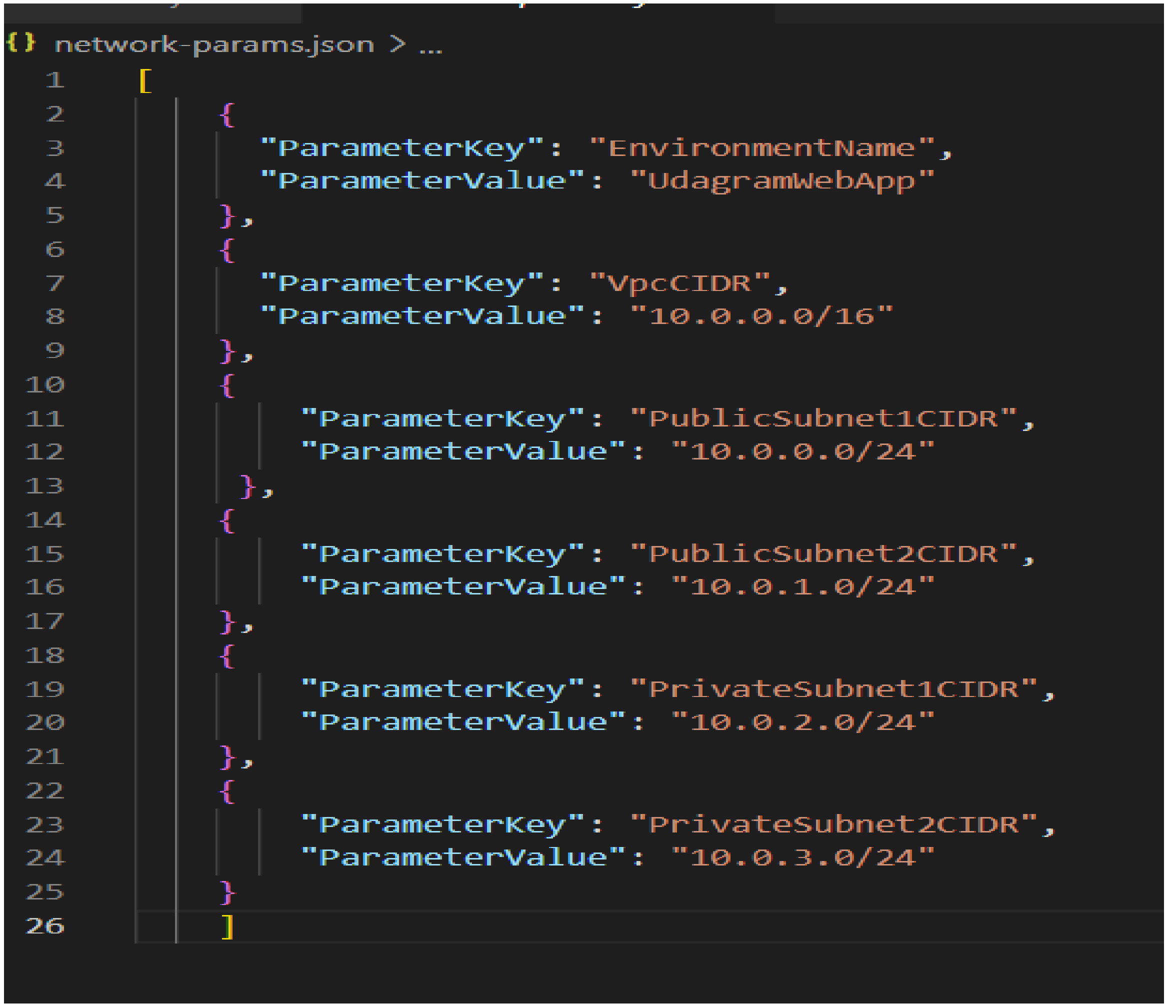
PROJECT OVERVIEW

In this project, you’ll deploy web servers for a highly available web app using Cloud Formation. You will write the code that creates and deploys the infrastructure and application for an Instagram-like app from the ground up. You will begin with deploying the networking components, followed by servers, security roles and software. The procedure you follow here will become part of your portfolio of cloud projects. You’ll do it exactly as it’s done on the job - following best practices and scripting as much as possible. There will be two parts to this project:

- Diagram: You'll first develop a diagram that you can present as part of your portfolio and as a visual aid to understand the Cloud Formation script.

- Script (Template and Parameters): The second part is to interpret the instructions and create a matching Cloud Formation script.

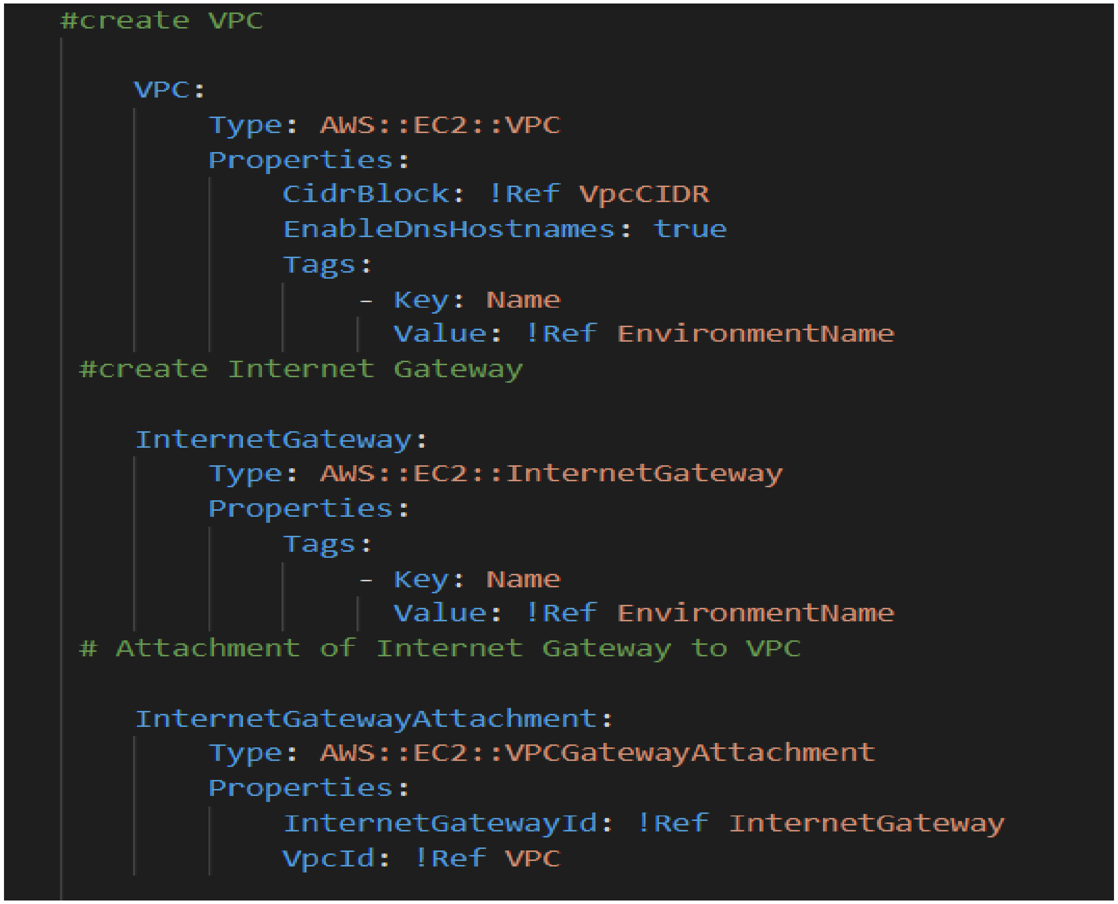
- Recommended to start your project with describe your project and more details about it like that
- Will create parameters with json.file :-
- It is parameters key we will used it in yaml file second steps



{} network-params.json > ...

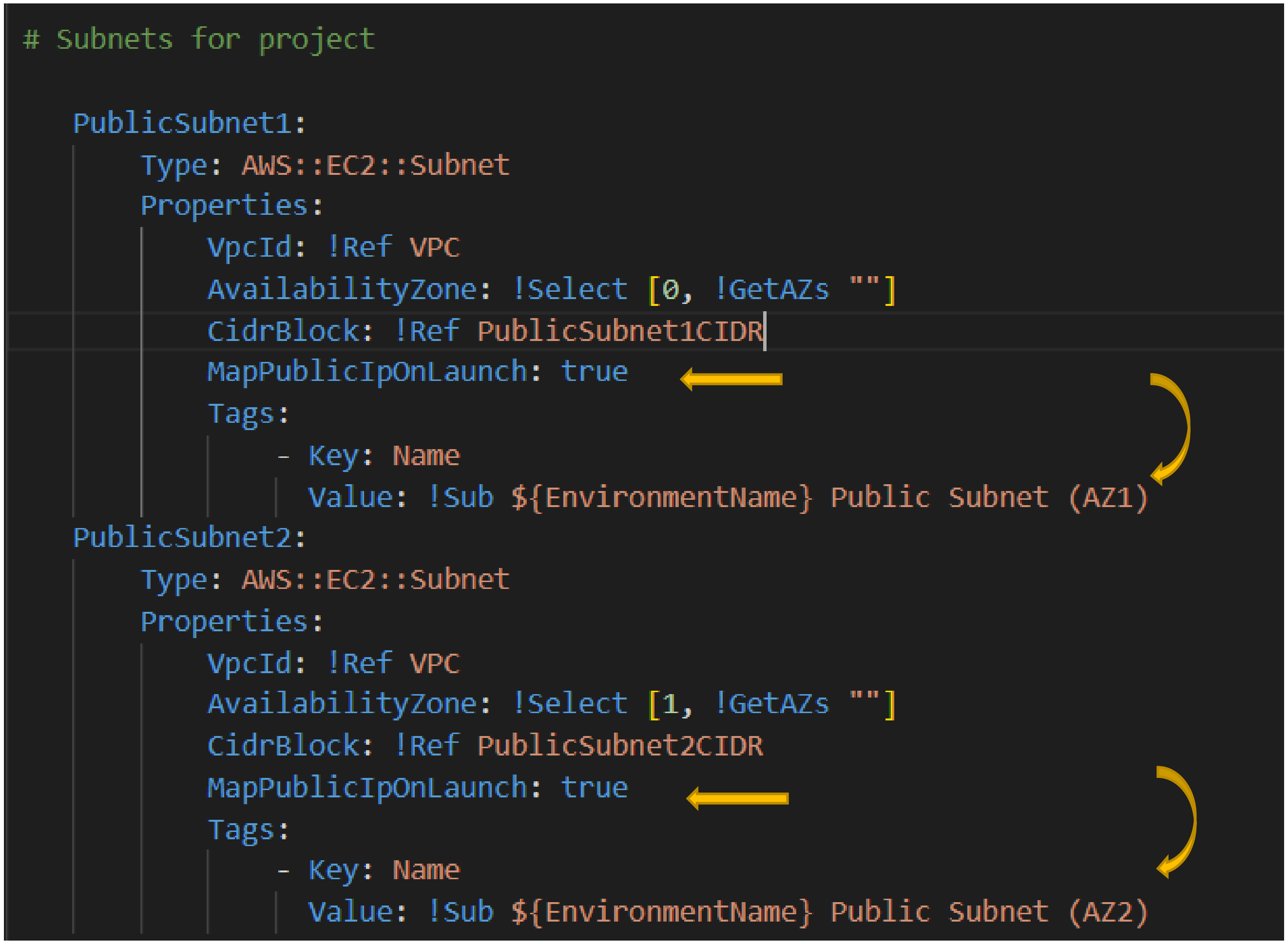
```
[
    {
      "ParameterKey": "EnvironmentName",
      "ParameterValue": "UdagramWebApp"
    },
    {
      "ParameterKey": "VpcCIDR",
      "ParameterValue": "10.0.0.0/16"
    },
    {
        "ParameterKey": "PublicSubnet1CIDR",
        "ParameterValue": "10.0.0.0/24"
     },
    {
        "ParameterKey": "PublicSubnet2CIDR",
        "ParameterValue": "10.0.1.0/24"
    },
    {
        "ParameterKey": "PrivateSubnet1CIDR",
        "ParameterValue": "10.0.2.0/24"
    },
    {
        "ParameterKey": "PrivateSubnet2CIDR",
        "ParameterValue": "10.0.3.0/24"
    }
    ]
```

- Now we will create yaml file and we will create in it our data:-
- create vpc by default
- create internet gateway and it will send traffics from public subnets to private subnets
- we create attachment because we have to internet getaway



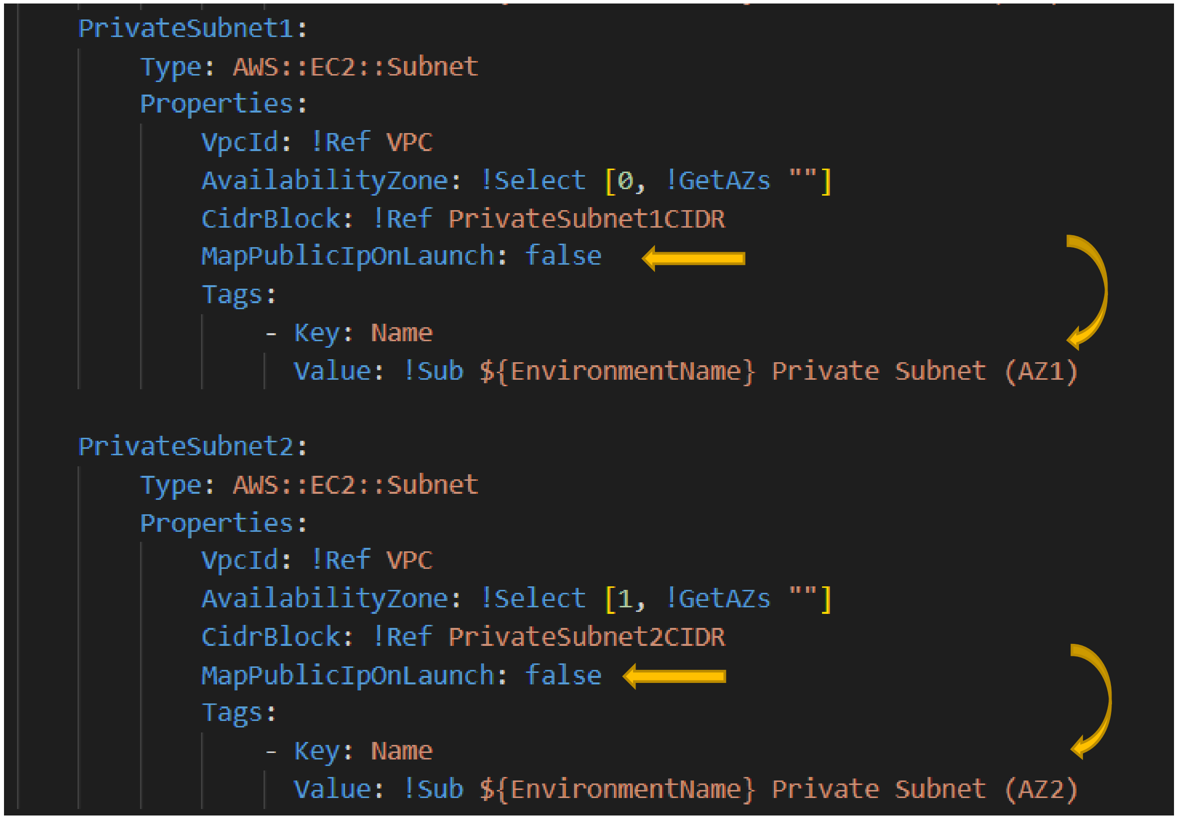
```yaml
#create VPC

    VPC:
        Type: AWS::EC2::VPC
        Properties:
            CidrBlock: !Ref VpcCIDR
            EnableDnsHostnames: true
            Tags:
                - Key: Name
                  Value: !Ref EnvironmentName
 #create Internet Gateway

    InternetGateway:
        Type: AWS::EC2::InternetGateway
        Properties:
            Tags:
                - Key: Name
                  Value: !Ref EnvironmentName
 # Attachment of Internet Gateway to VPC

    InternetGatewayAttachment:
        Type: AWS::EC2::VPCGatewayAttachment
        Properties:
            InternetGatewayId: !Ref InternetGateway
            VpcId: !Ref VPC
```

- Now we have 2 availability zone we create it and need to create 4 subnets (2puplic _2 private)
- We start to create 2 subnets public but we will put them in deferent zone
- And there map public it will be true



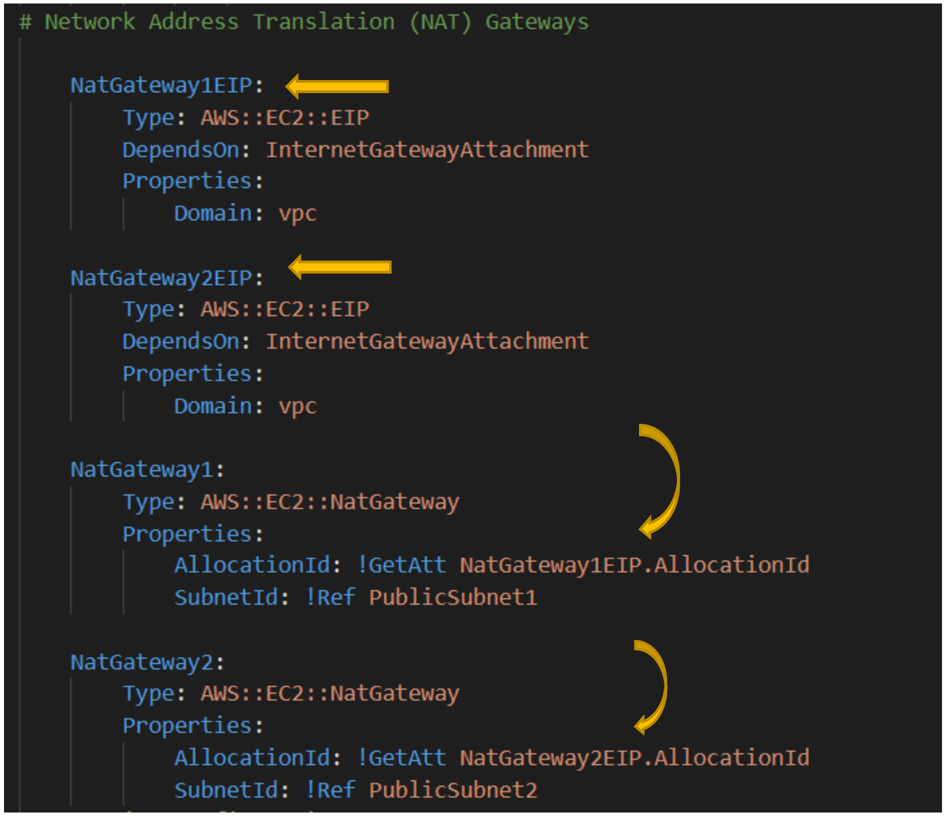
```
# Subnets for project

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs ""]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)
  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs ""]
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ2)
```

- We start to create 2 subnets private but we will put them in deferent zone
- And there map public it will be false



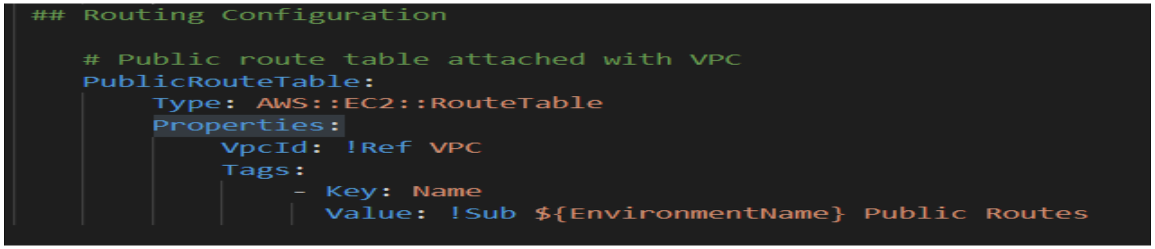
```
  PrivateSubnet1:
      Type: AWS::EC2::Subnet
      Properties:
          VpcId: !Ref VPC
          AvailabilityZone: !Select [0, !GetAZs ""]
          CidrBlock: !Ref PrivateSubnet1CIDR
          MapPublicIpOnLaunch: false
          Tags:
              - Key: Name
                Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

  PrivateSubnet2:
      Type: AWS::EC2::Subnet
      Properties:
          VpcId: !Ref VPC
          AvailabilityZone: !Select [1, !GetAZs ""]
          CidrBlock: !Ref PrivateSubnet2CIDR
          MapPublicIpOnLaunch: false
          Tags:
              - Key: Name
                Value: !Sub ${EnvironmentName} Private Subnet (AZ2)
```

- Now we will start to create networking
- We have 2 NAT gateway and it connect with 2 elastic ip address
- So we will create 2 net gateway EIP and the traffic will start with them
- And also create 2 net gateway will get data from them to another steps (Route table)



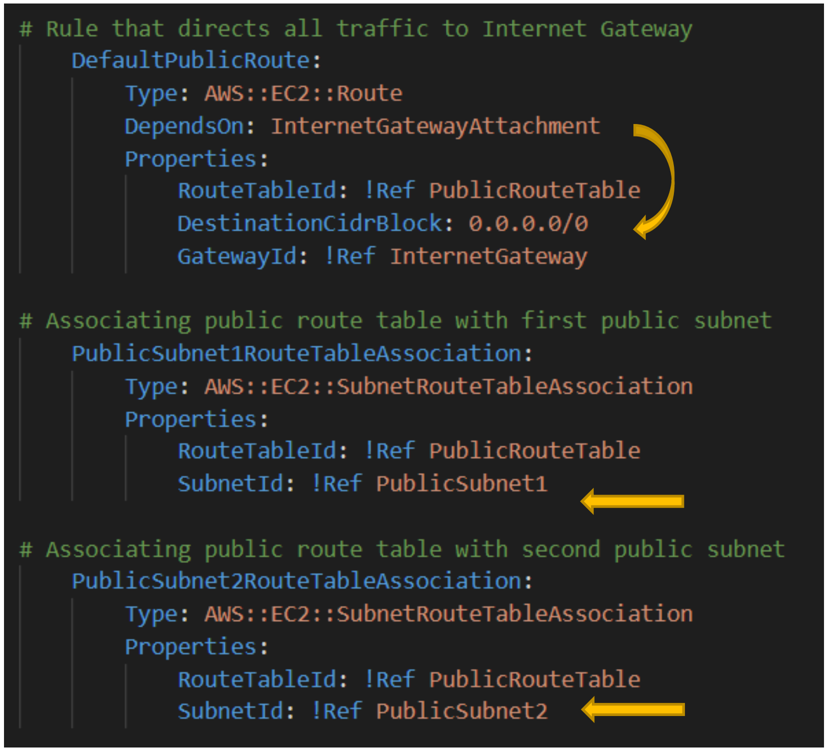
```
# Network Address Translation (NAT) Gateways

    NatGateway1EIP:
        Type: AWS::EC2::EIP
        DependsOn: InternetGatewayAttachment
        Properties:
            Domain: vpc

    NatGateway2EIP:
        Type: AWS::EC2::EIP
        DependsOn: InternetGatewayAttachment
        Properties:
            Domain: vpc

    NatGateway1:
        Type: AWS::EC2::NatGateway
        Properties:
            AllocationId: !GetAtt NatGateway1EIP.AllocationId
            SubnetId: !Ref PublicSubnet1

    NatGateway2:
        Type: AWS::EC2::NatGateway
        Properties:
            AllocationId: !GetAtt NatGateway2EIP.AllocationId
            SubnetId: !Ref PublicSubnet2
```

- Now we will start to create Routing Configuration by default



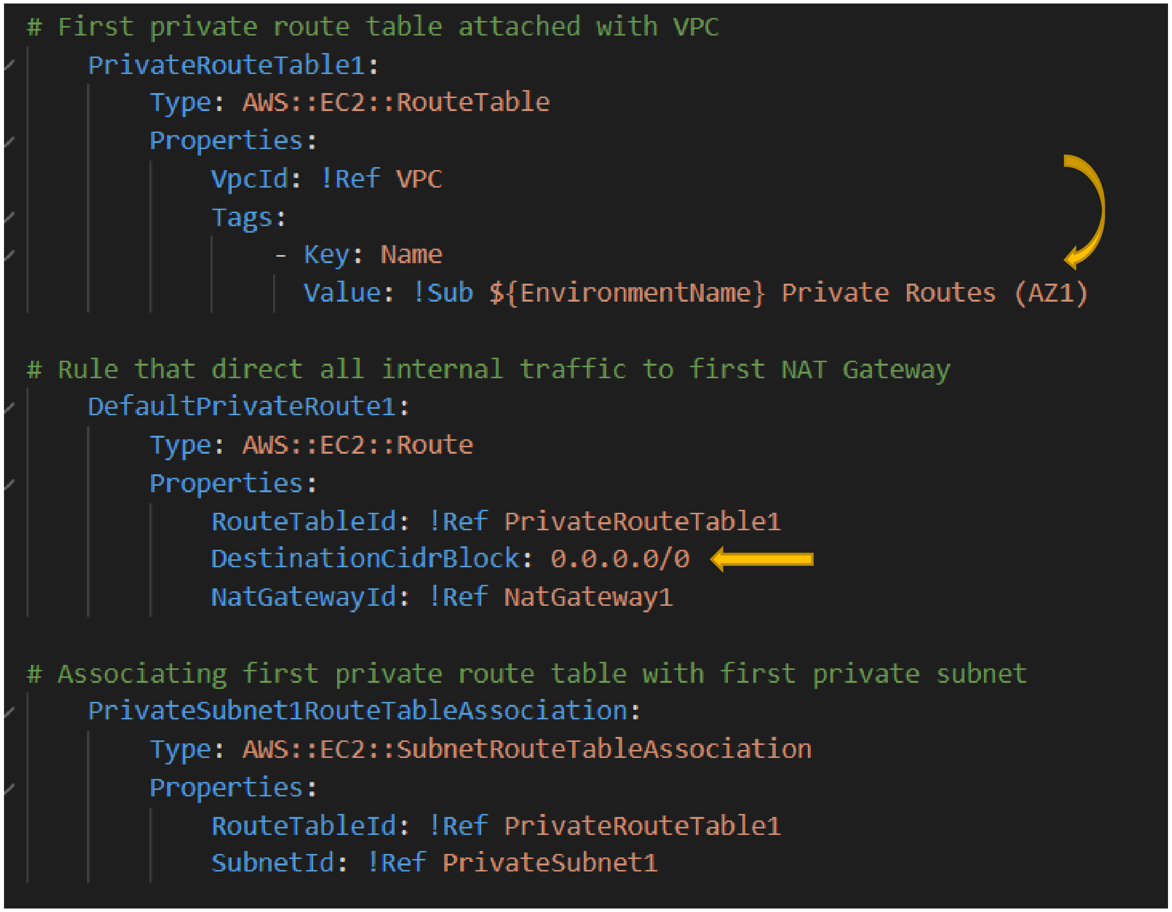
```
## Routing Configuration

    # Public route table attached with VPC
    PublicRouteTable:
        Type: AWS::EC2::RouteTable
        Properties:
            VpcId: !Ref VPC
            Tags:
                - Key: Name
                  Value: !Sub ${EnvironmentName} Public Routes
```

- We will now will start public route table and it will send data public from 2 subnets to route table



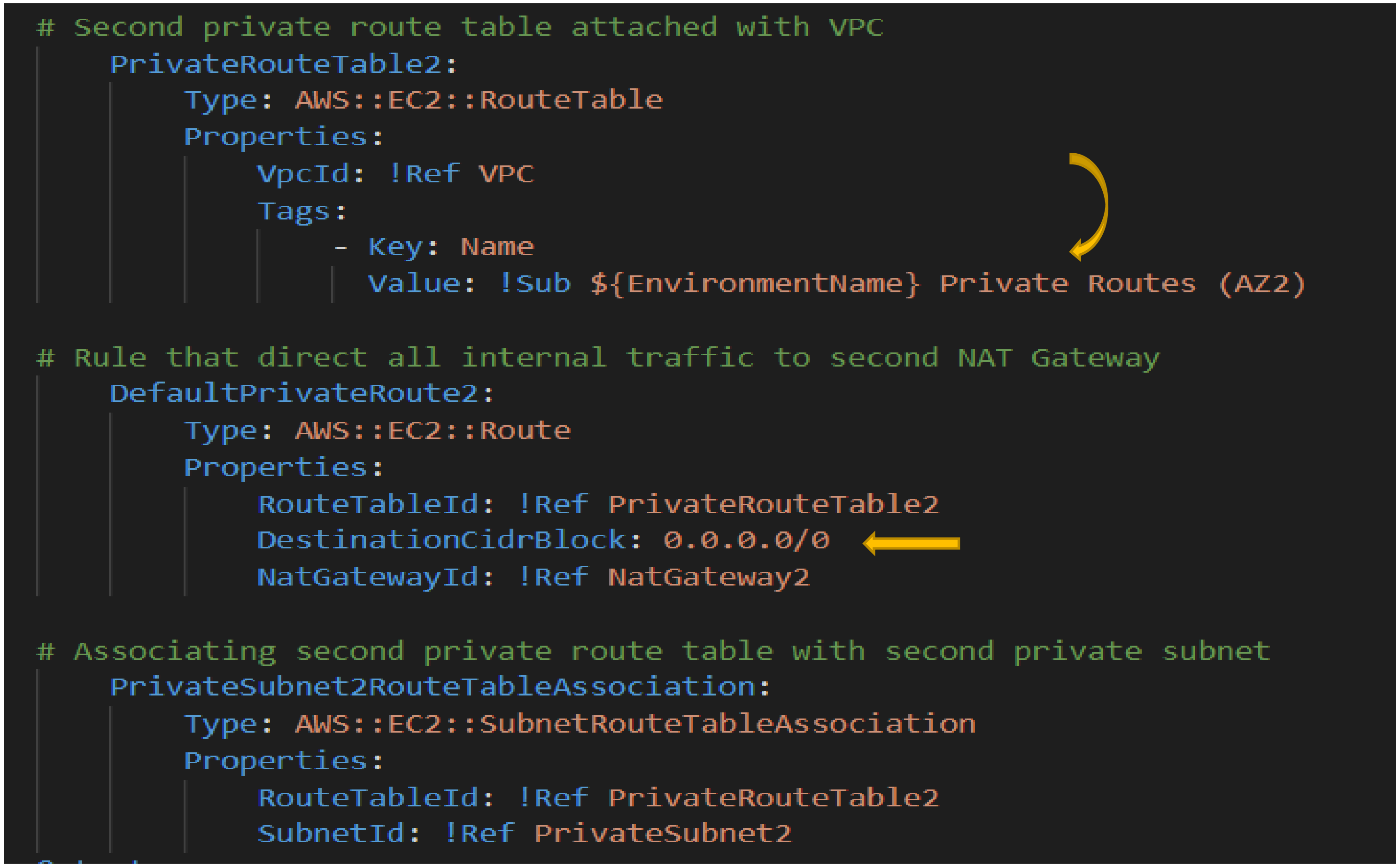
```
# Rule that directs all traffic to Internet Gateway
    DefaultPublicRoute:
        Type: AWS::EC2::Route
        DependsOn: InternetGatewayAttachment
        Properties:
            RouteTableId: !Ref PublicRouteTable
            DestinationCidrBlock: 0.0.0.0/0
            GatewayId: !Ref InternetGateway

# Associating public route table with first public subnet
    PublicSubnet1RouteTableAssociation:
        Type: AWS::EC2::SubnetRouteTableAssociation
        Properties:
            RouteTableId: !Ref PublicRouteTable
            SubnetId: !Ref PublicSubnet1

# Associating public route table with second public subnet
    PublicSubnet2RouteTableAssociation:
        Type: AWS::EC2::SubnetRouteTableAssociation
        Properties:
            RouteTableId: !Ref PublicRouteTable
            SubnetId: !Ref PublicSubnet2
```

- We now will start First private route table and it will send data public from 2 subnets to route table in (az1)



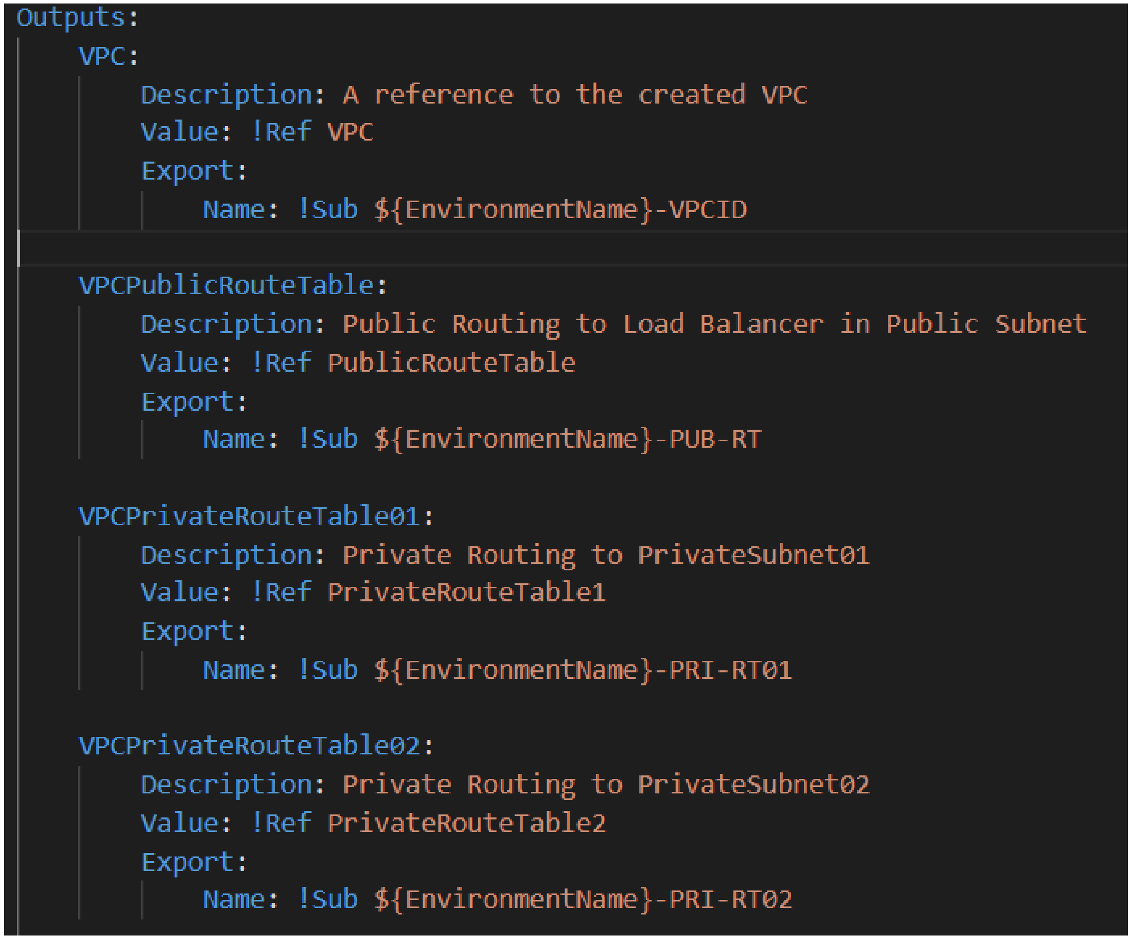
```
# First private route table attached with VPC
    PrivateRouteTable1:
        Type: AWS::EC2::RouteTable
        Properties:
            VpcId: !Ref VPC
            Tags:
                - Key: Name
                  Value: !Sub ${EnvironmentName} Private Routes (AZ1)

# Rule that direct all internal traffic to first NAT Gateway
    DefaultPrivateRoute1:
        Type: AWS::EC2::Route
        Properties:
            RouteTableId: !Ref PrivateRouteTable1
            DestinationCidrBlock: 0.0.0.0/0
            NatGatewayId: !Ref NatGateway1

# Associating first private route table with first private subnet
    PrivateSubnet1RouteTableAssociation:
        Type: AWS::EC2::SubnetRouteTableAssociation
        Properties:
            RouteTableId: !Ref PrivateRouteTable1
            SubnetId: !Ref PrivateSubnet1
```

- We now will start second private route table and it will send data public from 2 subnets to route table (az2)



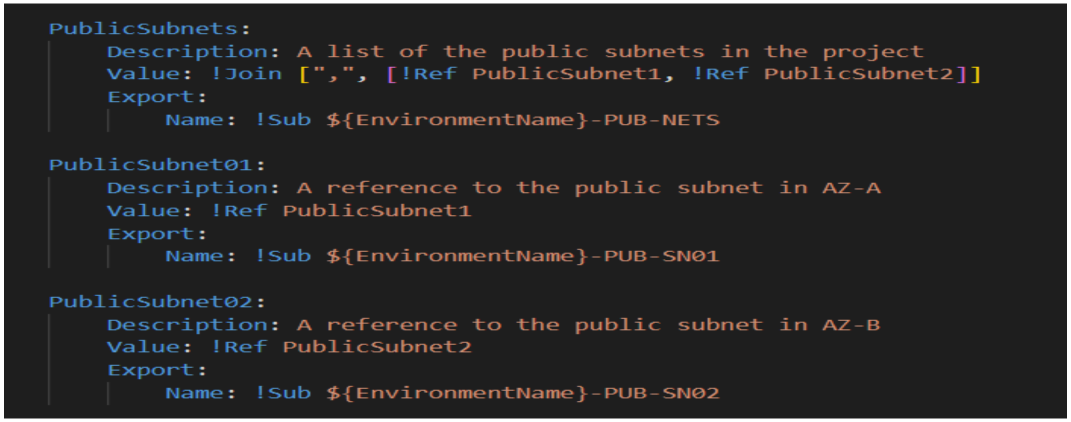
```
# Second private route table attached with VPC
    PrivateRouteTable2:
        Type: AWS::EC2::RouteTable
        Properties:
            VpcId: !Ref VPC
            Tags:
                - Key: Name
                  Value: !Sub ${EnvironmentName} Private Routes (AZ2)

# Rule that direct all internal traffic to second NAT Gateway
    DefaultPrivateRoute2:
        Type: AWS::EC2::Route
        Properties:
            RouteTableId: !Ref PrivateRouteTable2
            DestinationCidrBlock: 0.0.0.0/0
            NatGatewayId: !Ref NatGateway2

# Associating second private route table with second private subnet
    PrivateSubnet2RouteTableAssociation:
        Type: AWS::EC2::SubnetRouteTableAssociation
        Properties:
            RouteTableId: !Ref PrivateRouteTable2
            SubnetId: !Ref PrivateSubnet2
```

- Now create outputs for vpcs



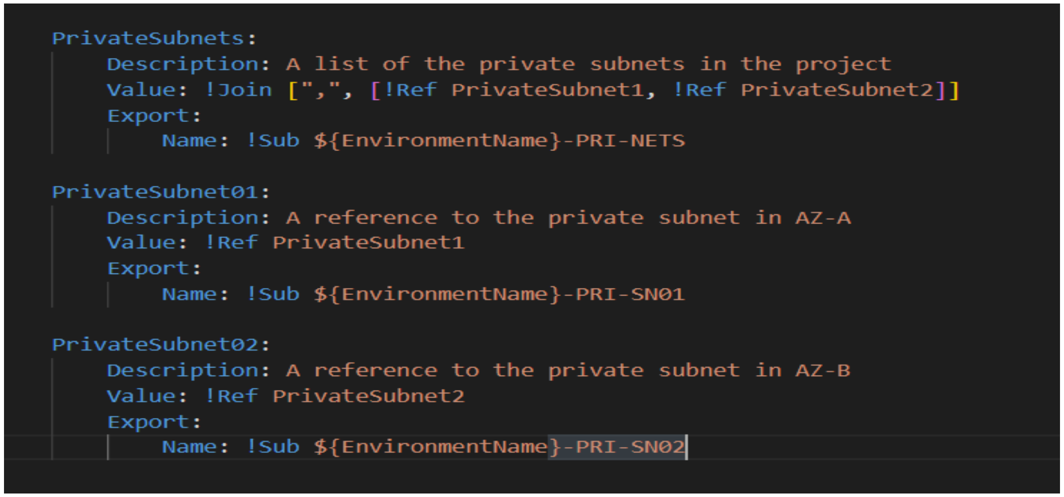
```
Outputs:
    VPC:
        Description: A reference to the created VPC
        Value: !Ref VPC
        Export:
            Name: !Sub ${EnvironmentName}-VPCID

    VPCPublicRouteTable:
        Description: Public Routing to Load Balancer in Public Subnet
        Value: !Ref PublicRouteTable
        Export:
            Name: !Sub ${EnvironmentName}-PUB-RT

    VPCPrivateRouteTable01:
        Description: Private Routing to PrivateSubnet01
        Value: !Ref PrivateRouteTable1
        Export:
            Name: !Sub ${EnvironmentName}-PRI-RT01

    VPCPrivateRouteTable02:
        Description: Private Routing to PrivateSubnet02
        Value: !Ref PrivateRouteTable2
        Export:
            Name: !Sub ${EnvironmentName}-PRI-RT02
```

- And we will create outputs Public Subnets



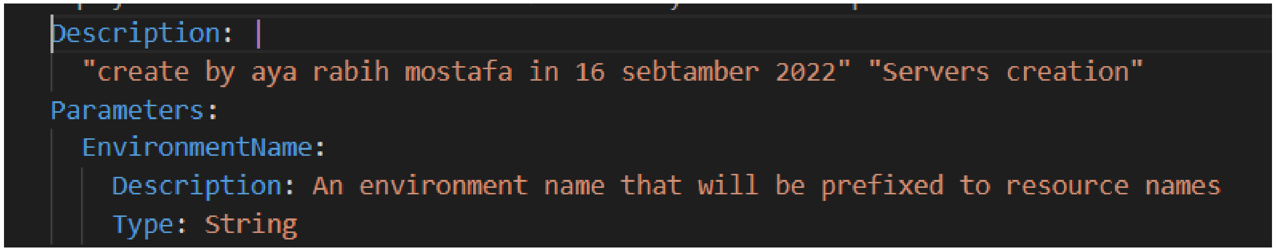
```
PublicSubnets:
    Description: A list of the public subnets in the project
    Value: !Join [",", [!Ref PublicSubnet1, !Ref PublicSubnet2]]
    Export:
        Name: !Sub ${EnvironmentName}-PUB-NETS

PublicSubnet01:
    Description: A reference to the public subnet in AZ-A
    Value: !Ref PublicSubnet1
    Export:
        Name: !Sub ${EnvironmentName}-PUB-SN01

PublicSubnet02:
    Description: A reference to the public subnet in AZ-B
    Value: !Ref PublicSubnet2
    Export:
        Name: !Sub ${EnvironmentName}-PUB-SN02
```

- And we will create outputs private Subnets



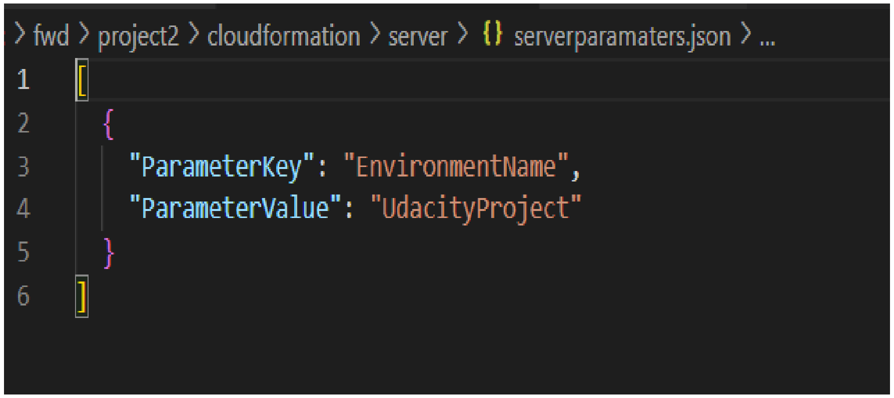
```
PrivateSubnets:
    Description: A list of the private subnets in the project
    Value: !Join [",", [!Ref PrivateSubnet1, !Ref PrivateSubnet2]]
    Export:
        Name: !Sub ${EnvironmentName}-PRI-NETS

PrivateSubnet01:
    Description: A reference to the private subnet in AZ-A
    Value: !Ref PrivateSubnet1
    Export:
        Name: !Sub ${EnvironmentName}-PRI-SN01

PrivateSubnet02:
    Description: A reference to the private subnet in AZ-B
    Value: !Ref PrivateSubnet2
    Export:
        Name: !Sub ${EnvironmentName}-PRI-SN02
```

- Recommended to start your project with describe your project and more details about it like that



```
Description: |
  "create by aya rabih mostafa in 16 sebtamber 2022" "Servers creation"
Parameters:
  EnvironmentName:
    Description: An environment name that will be prefixed to resource names
    Type: String
```

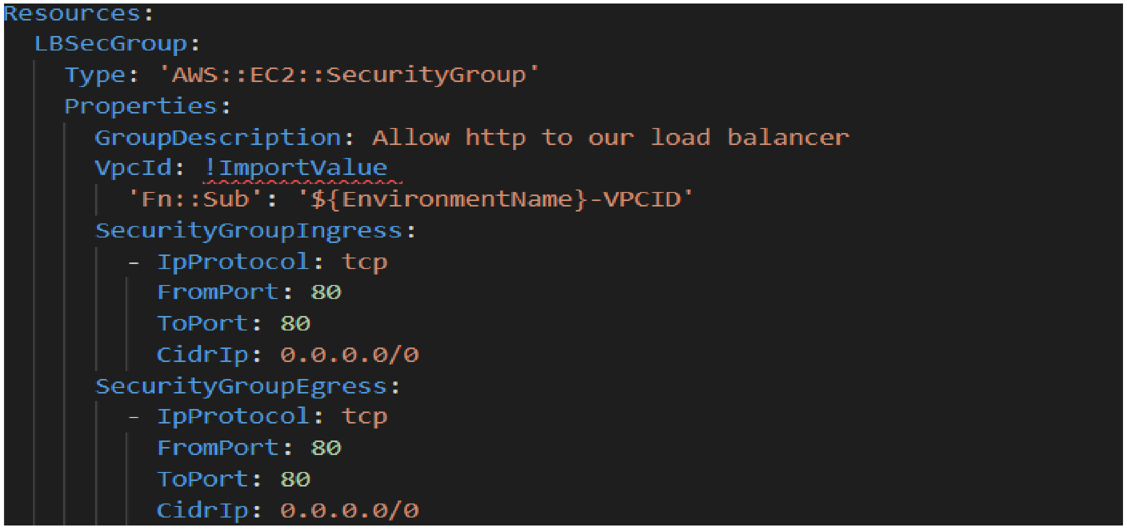
- Will create parameters with json.file :-
- It is parameters key we will used it in yaml file second steps



fwd › project2 › cloudformation › server › {} serverparamaters.json › ...

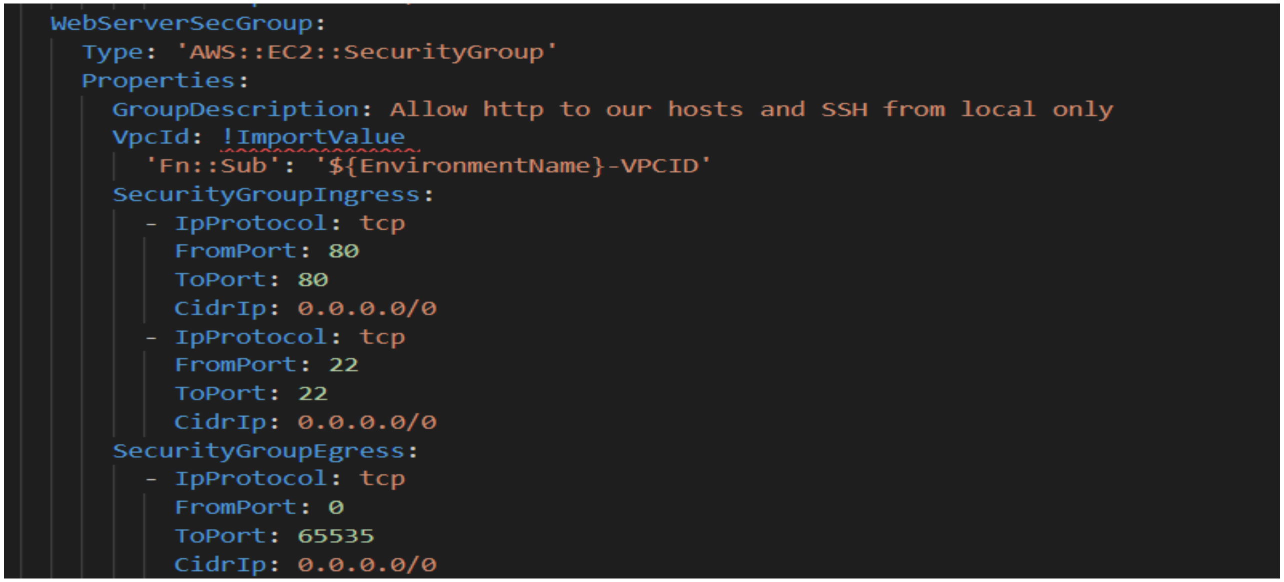
```
[
  {
    "ParameterKey": "EnvironmentName",
    "ParameterValue": "UdacityProject"
  }
]
```

- We will now start to create Security Group for Load Balancer



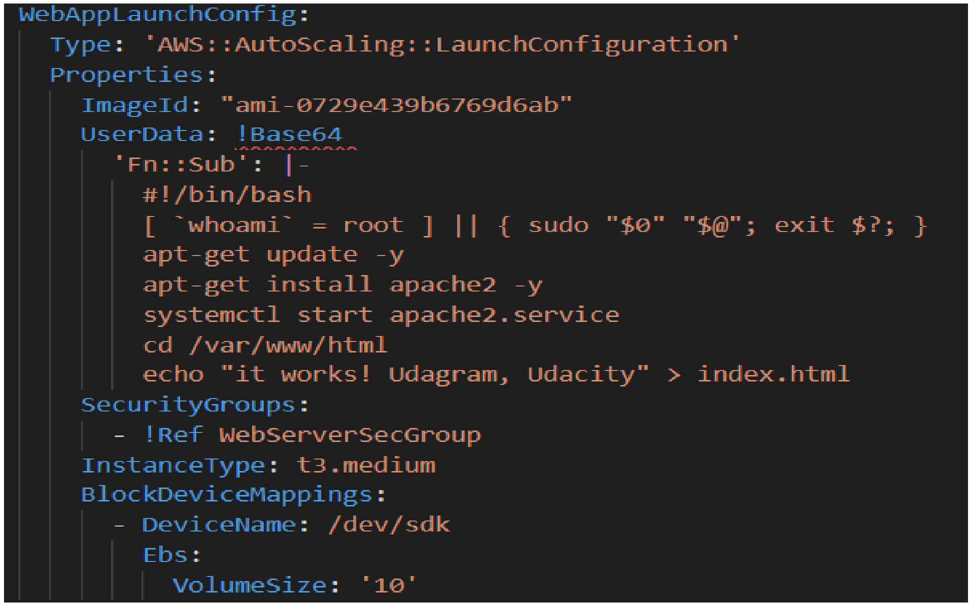
```yaml
Resources:
  LBSecGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Allow http to our load balancer
      VpcId: !ImportValue
        'Fn::Sub': '${EnvironmentName}-VPCID'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
```

- We will now start to create web server Security group



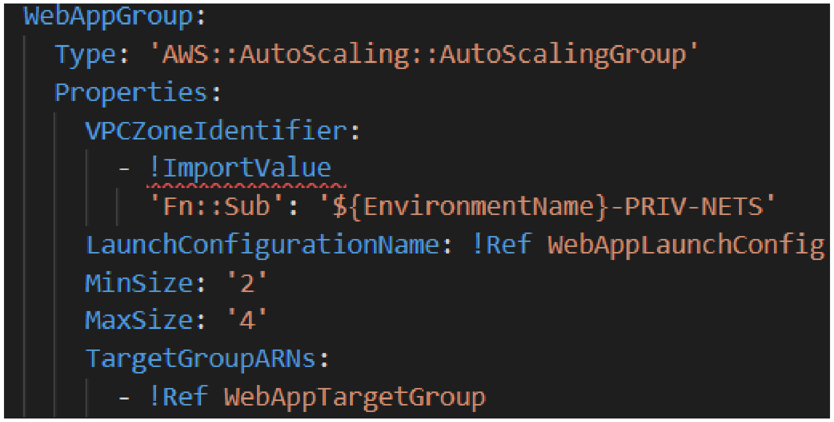
```yaml
  WebServerSecGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Allow http to our hosts and SSH from local only
      VpcId: !ImportValue
        'Fn::Sub': '${EnvironmentName}-VPCID'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
        - IpProtocol: tcp
          FromPort: 0
          ToPort: 65535
          CidrIp: 0.0.0.0/0
```

- Create Launch Configuration



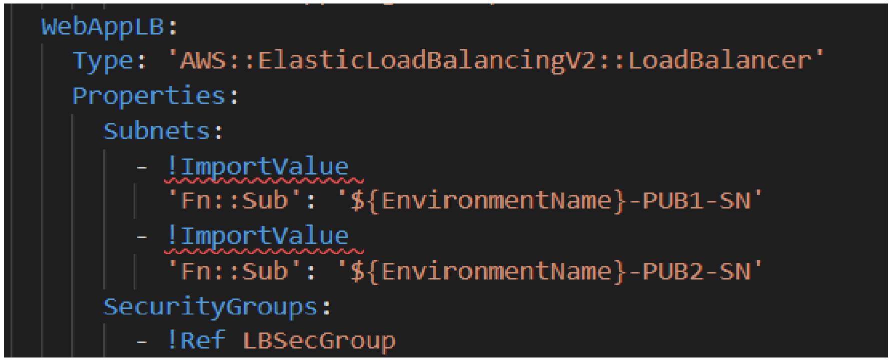
```yaml
WebAppLaunchConfig:
  Type: 'AWS::AutoScaling::LaunchConfiguration'
  Properties:
    ImageId: "ami-0729e439b6769d6ab"
    UserData: !Base64
      'Fn::Sub': |-
        #!/bin/bash
        [ `whoami` = root ] || { sudo "$0" "$@"; exit $?; }
        apt-get update -y
        apt-get install apache2 -y
        systemctl start apache2.service
        cd /var/www/html
        echo "it works! Udagram, Udacity" > index.html
    SecurityGroups:
      - !Ref WebServerSecGroup
    InstanceType: t3.medium
    BlockDeviceMappings:
      - DeviceName: /dev/sdk
        Ebs:
          VolumeSize: '10'
```

- Create webapp group



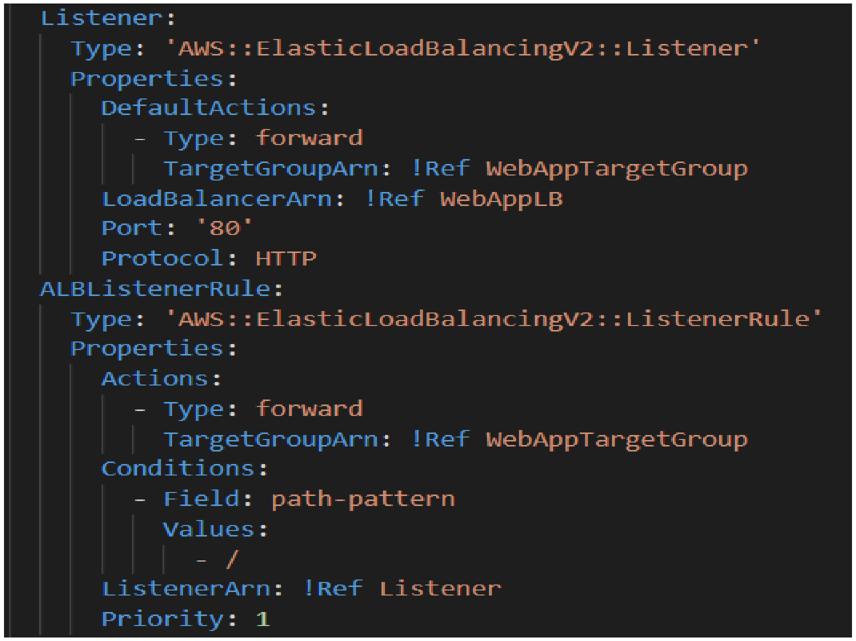
```yaml
WebAppGroup:
  Type: 'AWS::AutoScaling::AutoScalingGroup'
  Properties:
    VPCZoneIdentifier:
      - !ImportValue
        'Fn::Sub': '${EnvironmentName}-PRIV-NETS'
    LaunchConfigurationName: !Ref WebAppLaunchConfig
    MinSize: '2'
    MaxSize: '4'
    TargetGroupARNs:
      - !Ref WebAppTargetGroup
```

- Create webAppLB



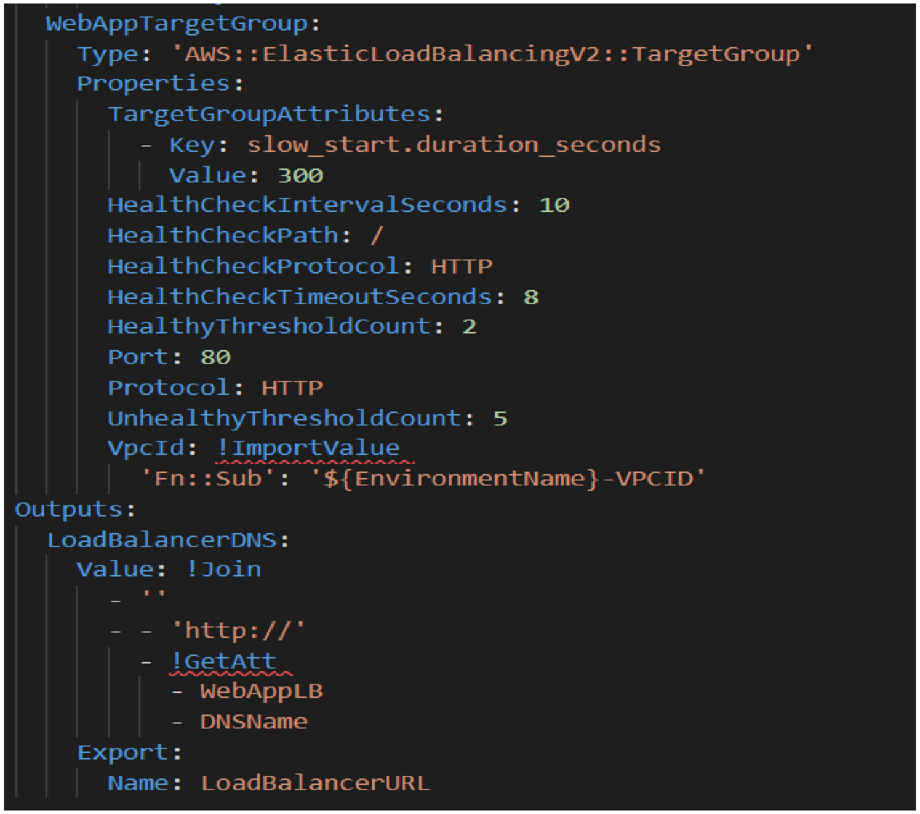
```
WebAppLB:
  Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
  Properties:
    Subnets:
      - !ImportValue
        'Fn::Sub': '${EnvironmentName}-PUB1-SN'
      - !ImportValue
        'Fn::Sub': '${EnvironmentName}-PUB2-SN'
    SecurityGroups:
      - !Ref LBSecGroup
```

- Create listener



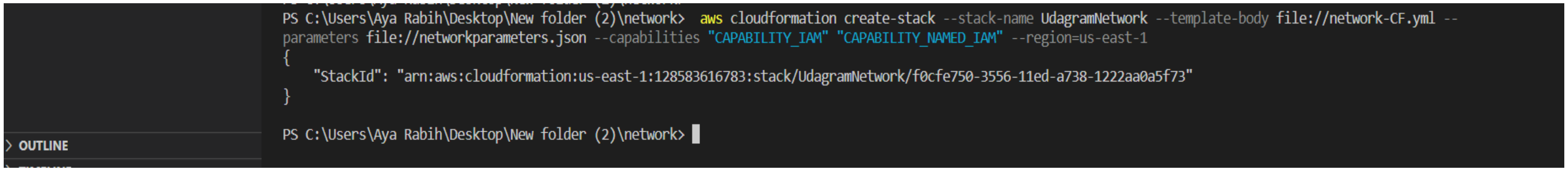
```
Listener:
  Type: 'AWS::ElasticLoadBalancingV2::Listener'
  Properties:
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref WebAppTargetGroup
    LoadBalancerArn: !Ref WebAppLB
    Port: '80'
    Protocol: HTTP
ALBListenerRule:
  Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
  Properties:
    Actions:
      - Type: forward
        TargetGroupArn: !Ref WebAppTargetGroup
    Conditions:
      - Field: path-pattern
        Values:
          - /
    ListenerArn: !Ref Listener
    Priority: 1
```

- Create targetgroup

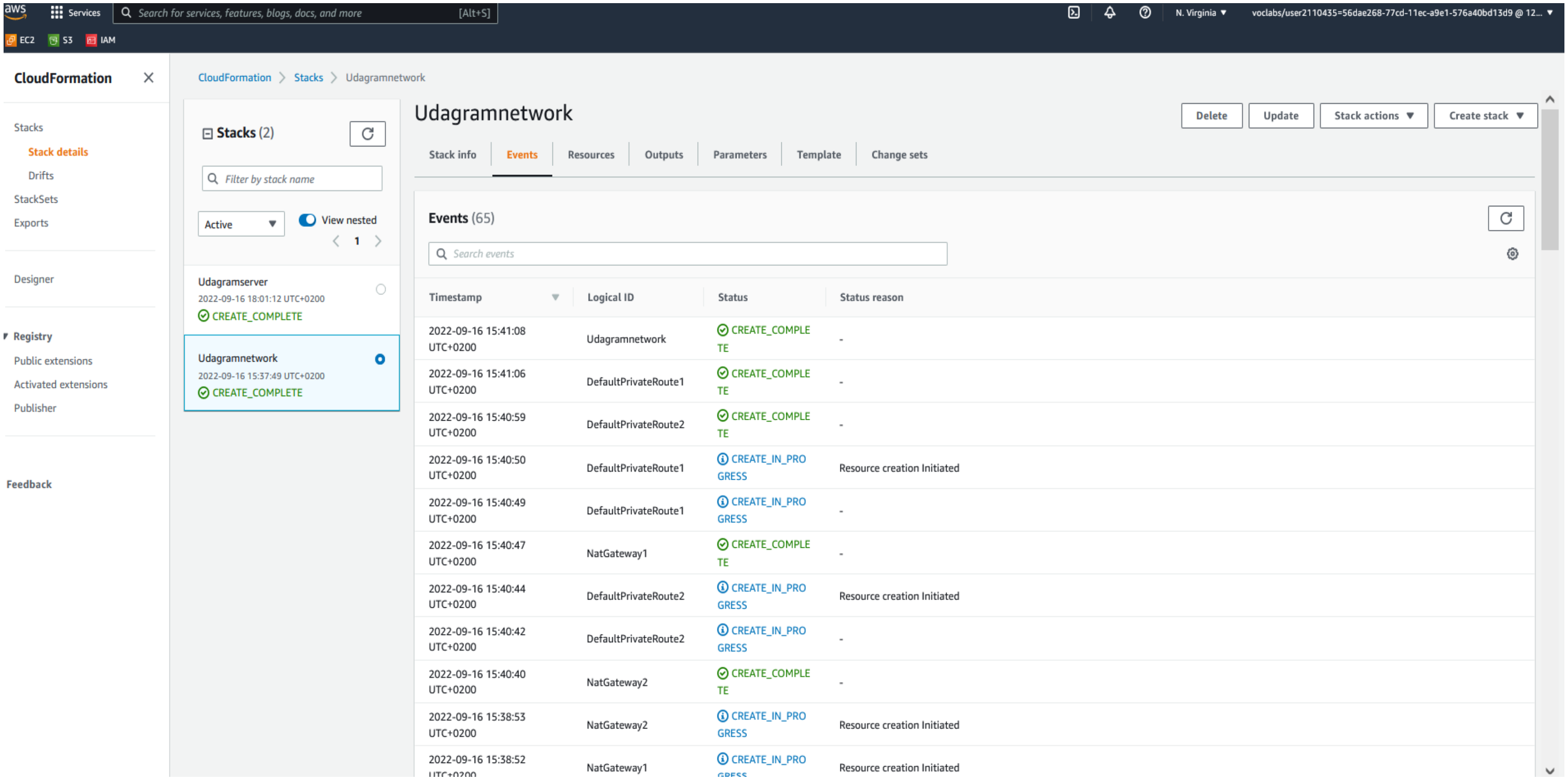


```
  WebAppTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      TargetGroupAttributes:
        - Key: slow_start.duration_seconds
          Value: 300
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: /
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 8
      HealthyThresholdCount: 2
      Port: 80
      Protocol: HTTP
      UnhealthyThresholdCount: 5
      VpcId: !ImportValue
        'Fn::Sub': '${EnvironmentName}-VPCID'
Outputs:
  LoadBalancerDNS:
    Value: !Join
      - ''
      - - 'http://'
        - !GetAtt
          - WebAppLB
          - DNSName
    Export:
      Name: LoadBalancerURL
```

- Now we run code and that output for network



```
PS C:\Users\Aya Rabih\Desktop\New folder (2)\network>  aws cloudformation create-stack --stack-name UdagramNetwork --template-body file://network-CF.yml --
parameters file://networkparameters.json --capabilities "CAPABILITY_IAM" "CAPABILITY_NAMED_IAM" --region=us-east-1
{
    "StackId": "arn:aws:cloudformation:us-east-1:128583616783:stack/UdagramNetwork/f0cfe750-3556-11ed-a738-1222aa0a5f73"
}

PS C:\Users\Aya Rabih\Desktop\New folder (2)\network> 
```



Udagramserver
2022-09-16 18:01:12 UTC+0200
CREATE_COMPLETE

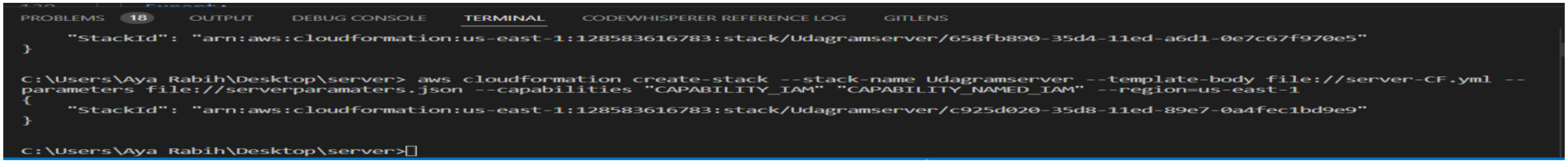
Udagramnetwork
2022-09-16 15:37:49 UTC+0200
CREATE_COMPLETE

Udagramnetwork

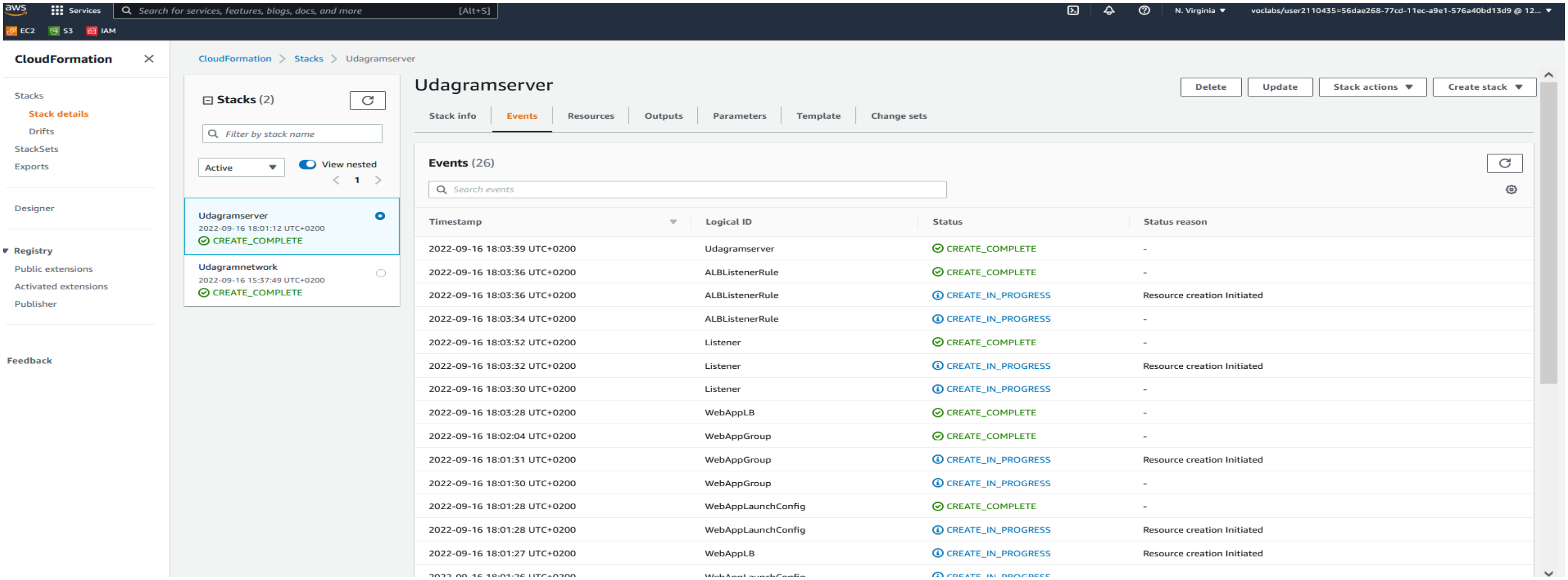
Events (65)

Timestamp	Logical ID	Status	Status reason
2022-09-16 15:41:08 UTC+0200	Udagramnetwork	CREATE_COMPLETE	-
2022-09-16 15:41:06 UTC+0200	DefaultPrivateRoute1	CREATE_COMPLETE	-
2022-09-16 15:40:59 UTC+0200	DefaultPrivateRoute2	CREATE_COMPLETE	-
2022-09-16 15:40:50 UTC+0200	DefaultPrivateRoute1	CREATE_IN_PROGRESS	Resource creation Initiated
2022-09-16 15:40:49 UTC+0200	DefaultPrivateRoute1	CREATE_IN_PROGRESS	-
2022-09-16 15:40:47 UTC+0200	NatGateway1	CREATE_COMPLETE	-
2022-09-16 15:40:44 UTC+0200	DefaultPrivateRoute2	CREATE_IN_PROGRESS	Resource creation Initiated
2022-09-16 15:40:42 UTC+0200	DefaultPrivateRoute2	CREATE_IN_PROGRESS	-
2022-09-16 15:40:40 UTC+0200	NatGateway2	CREATE_COMPLETE	-
2022-09-16 15:38:53 UTC+0200	NatGateway2	CREATE_IN_PROGRESS	Resource creation Initiated
2022-09-16 15:38:52 UTC+0200	NatGateway1	CREATE_IN_PROGRESS	Resource creation Initiated

- Now we run code and that output for server



```
    "StackId": "arn:aws:cloudformation:us-east-1:128583616783:stack/Udagramserver/658fb890-35d4-11ed-a6d1-0e7c67f970e5"
}

C:\Users\Aya Rabih\Desktop\server> aws cloudformation create-stack --stack-name Udagramserver --template-body file://server-CF.yml --
parameters file://serverparamaters.json --capabilities "CAPABILITY_IAM" "CAPABILITY_NAMED_IAM" --region=us-east-1
{
    "StackId": "arn:aws:cloudformation:us-east-1:128583616783:stack/Udagramserver/c925d020-35d8-11ed-89e7-0a4fec1bd9e9"
}

C:\Users\Aya Rabih\Desktop\server>
```



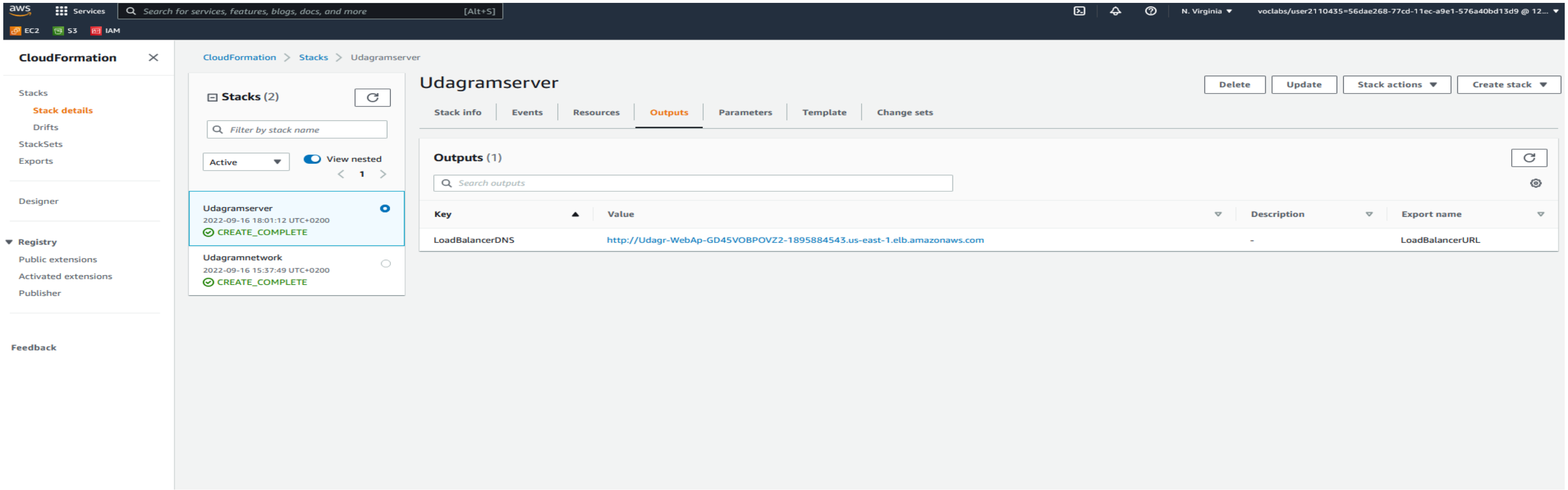
Udagramserver
2022-09-16 18:01:12 UTC+0200
CREATE_COMPLETE

Udagramnetwork
2022-09-16 15:37:49 UTC+0200
CREATE_COMPLETE

Udagramserver

Events (26)

Timestamp	Logical ID	Status	Status reason
2022-09-16 18:03:39 UTC+0200	Udagramserver	CREATE_COMPLETE	-
2022-09-16 18:03:36 UTC+0200	ALBListenerRule	CREATE_COMPLETE	-
2022-09-16 18:03:36 UTC+0200	ALBListenerRule	CREATE_IN_PROGRESS	Resource creation Initiated
2022-09-16 18:03:34 UTC+0200	ALBListenerRule	CREATE_IN_PROGRESS	-
2022-09-16 18:03:32 UTC+0200	Listener	CREATE_COMPLETE	-
2022-09-16 18:03:32 UTC+0200	Listener	CREATE_IN_PROGRESS	Resource creation Initiated
2022-09-16 18:03:30 UTC+0200	Listener	CREATE_IN_PROGRESS	-
2022-09-16 18:03:28 UTC+0200	WebAppLB	CREATE_COMPLETE	-
2022-09-16 18:02:04 UTC+0200	WebAppGroup	CREATE_COMPLETE	-
2022-09-16 18:01:31 UTC+0200	WebAppGroup	CREATE_IN_PROGRESS	Resource creation Initiated
2022-09-16 18:01:30 UTC+0200	WebAppGroup	CREATE_IN_PROGRESS	-
2022-09-16 18:01:28 UTC+0200	WebAppLaunchConfig	CREATE_COMPLETE	-
2022-09-16 18:01:28 UTC+0200	WebAppLaunchConfig	CREATE_IN_PROGRESS	Resource creation Initiated
2022-09-16 18:01:27 UTC+0200	WebAppLB	CREATE_IN_PROGRESS	Resource creation Initiated
2022-09-16 18:01:26 UTC+0200	WebAppLaunchConfig	CREATE_IN_PROGRESS	



Stacks (2)

- Udagramserver — 2022-09-16 18:01:12 UTC+0200 — CREATE_COMPLETE
- Udagramnetwork — 2022-09-16 15:37:49 UTC+0200 — CREATE_COMPLETE

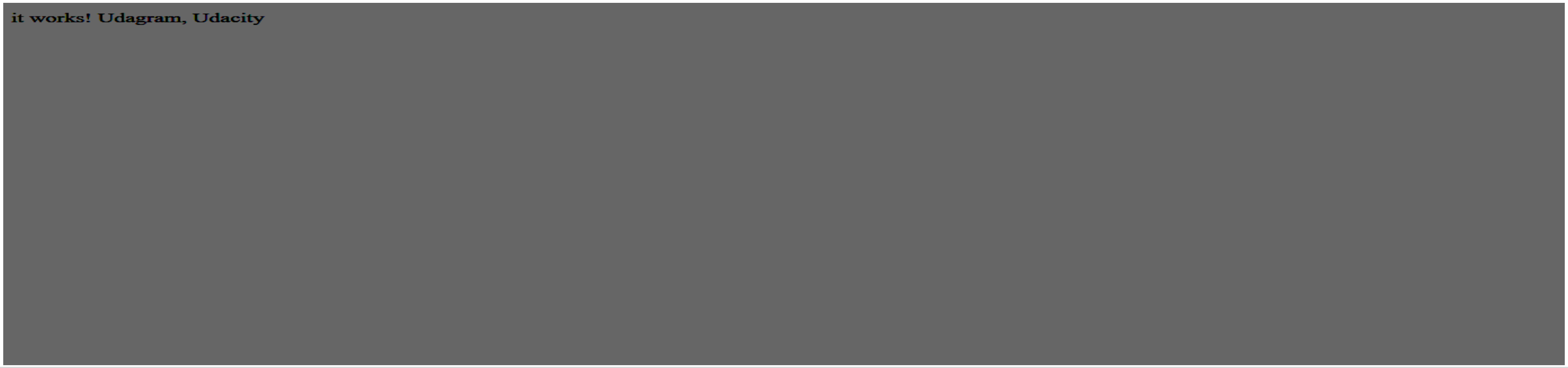
Udagramserver

Outputs (1)

Key	Value	Description	Export name
LoadBalancerDNS	http://Udagr-WebAp-GD45VOBPOVZ2-1895884543.us-east-1.elb.amazonaws.com	-	LoadBalancerURL

And when we run site open ourserver

Done!



it works! Udagram, Udacity