Activity 3

In this activity, we are enhancing a CloudFormation template to make it more dynamic and reusable by introducing parameters for various AWS resources. The template creates a VPC with public and private subnets, route tables, an Internet gateway, and an EC2 instance with an attached EBS volume and security group. The goal is to allow customization of resource properties, such as instance type, image ID, volume size, key pair, CIDR blocks, and availability zones, using parameters. These parameters will ensure flexibility and reusability of the template across different deployments. Once the modifications are complete, the template will be tested by creating a stack, verifying resources, and cleaning up afterward.

Key Features:

1. **Parameterization of EC2 Instance Properties:**
- Define parameters for EC2 instance properties like InstanceType, ImageId, EBS VolumeSize, and KeyName.
- Restrict allowed values for InstanceType to 't2.nano', 't2.micro', and 't2.small'.

2. **VPC and Subnet Customization:**
- Create a CommaDelimitedList parameter for CIDR blocks to define the VPC and subnet IP ranges.
- Use Fn::Select to reference specific CIDR blocks for VPC, public subnet, and private subnet.

3. **Availability Zone Parameterization:**
- Define a parameter for Availability Zones and reference it in subnet configurations.

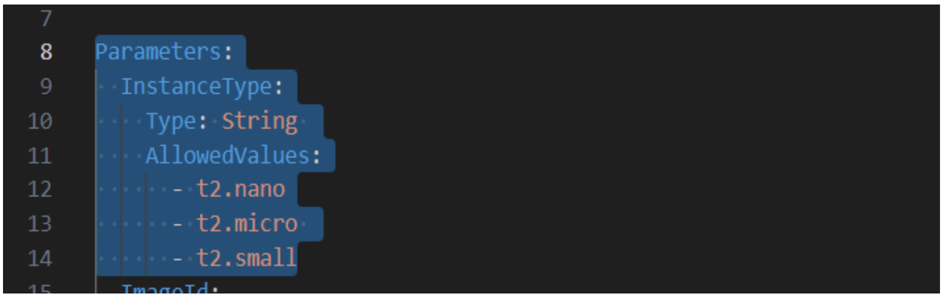
4. **Dynamic Naming with Fn::Sub:**
- Modify the VPC name tag by appending "-vpc" to the stack name using the Fn::Sub function and the AWS::StackName pseudo parameter.

5. **Enhanced Template Reusability:**
- Parameters increase the flexibility and reusability of the template for different use cases.

6. **Stack Testing and Validation:**
- Create a stack using the updated template, verify that resources are created with correct values, and clean up afterward.

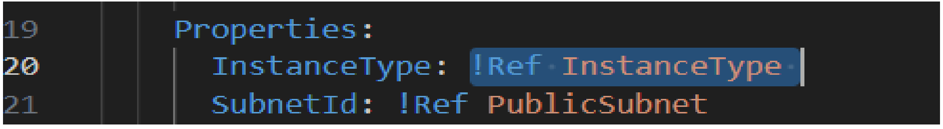
Activity

1. Find the template files in our GitHub repository under the same name as the heading for easy access and edits. Find and Save the attached template locally, open it in VS Code for edits.Add a Parameters section to the CloudFormation template. Define an InstanceType parameter (String type) for the EC2 instance with allowed values ('t2.nano', 't2.micro', 't2.small') and change the Image Id, Subnet id and Vpc id according to your region.



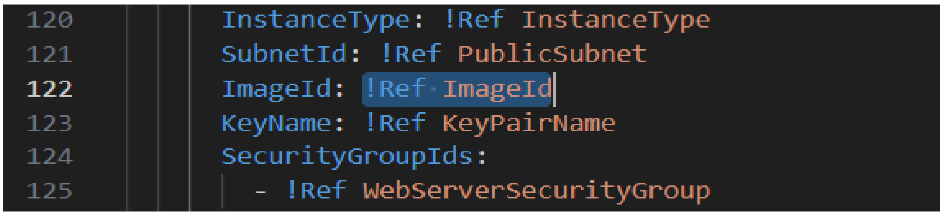
```
Parameters:
  InstanceType:
    Type: String
    AllowedValues:
      - t2.nano
      - t2.micro
      - t2.small
  ImageId:
```

2. Reference the InstanceType parameter in the WebServerInstance resource's InstanceType attribute.



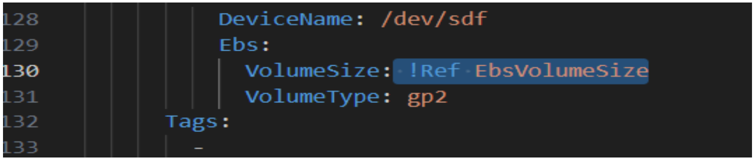
```
    Properties:
      InstanceType: !Ref InstanceType
      SubnetId: !Ref PublicSubnet
```

3. Define an ImageId parameter (String type) with the current value as the default and reference it in the ImageId attribute of WebServerInstance.

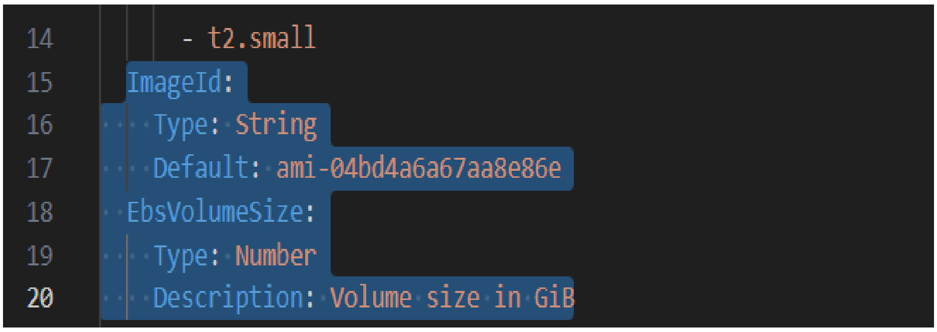


```
      InstanceType: !Ref InstanceType
      SubnetId: !Ref PublicSubnet
      ImageId: !Ref ImageId
      KeyName: !Ref KeyPairName
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
```

4. Define a VolumeSize parameter (Number type) for the EBS volume size with a description like "Volume size in Gigabytes" and reference it in BlockDeviceMappings.

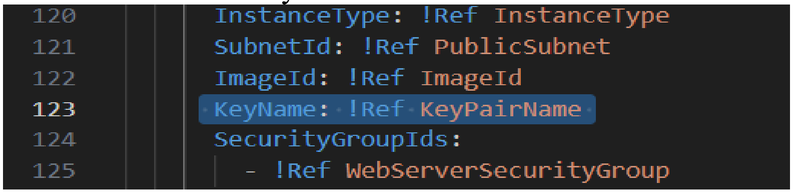


```
          DeviceName: /dev/sdf
          Ebs:
            VolumeSize: !Ref EbsVolumeSize
            VolumeType: gp2
      Tags:
        -
```

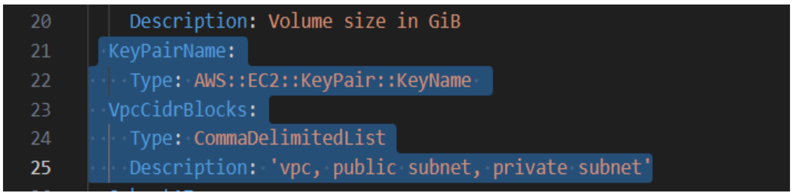


```
      - t2.small
  ImageId:
    Type: String
    Default: ami-04bd4a6a67aa8e86e
  EbsVolumeSize:
    Type: Number
    Description: Volume size in GiB
```

5. Define an EC2 KeyPair parameter with AWS-specific EC2 key-pair names and reference it in the KeyName attribute of WebServerInstance.

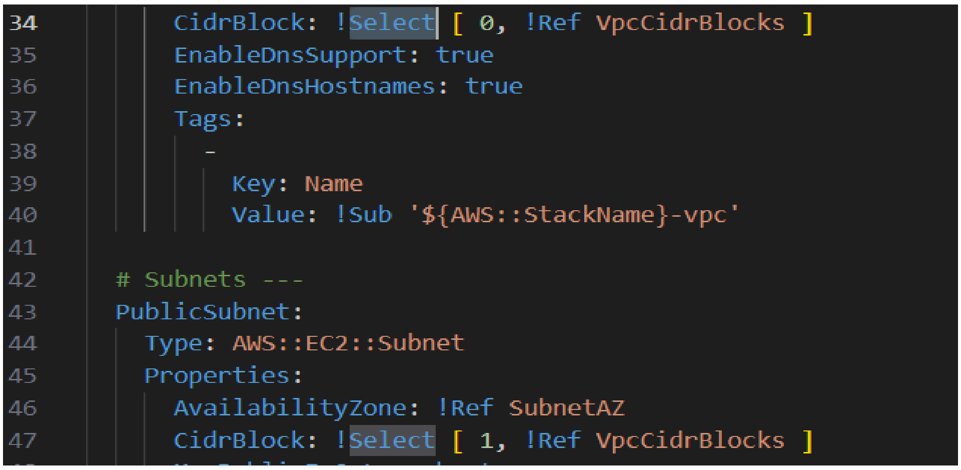


```
      InstanceType: !Ref InstanceType
      SubnetId: !Ref PublicSubnet
      ImageId: !Ref ImageId
      KeyName: !Ref KeyPairName
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
```



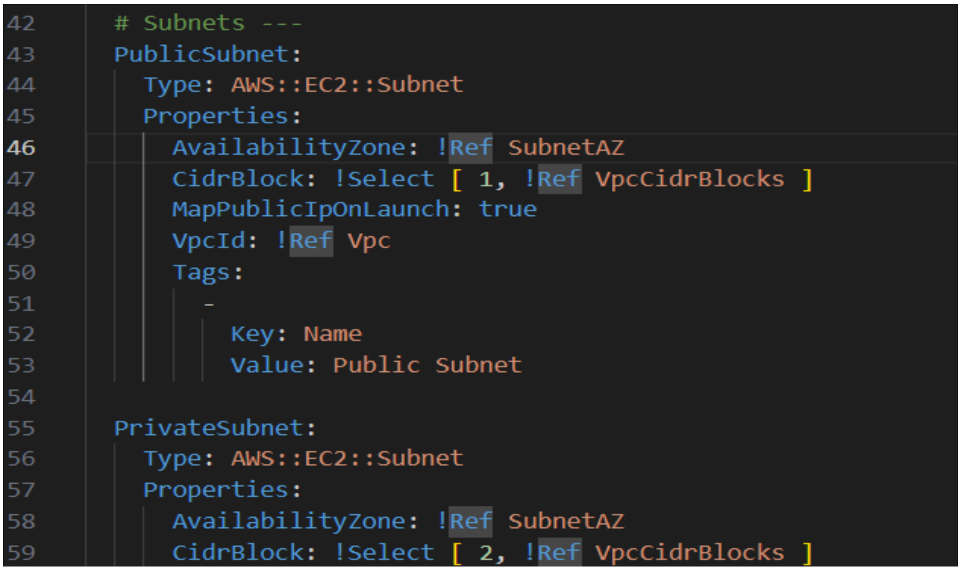
```
    Description: Volume size in GiB
  KeyPairName:
    Type: AWS::EC2::KeyPair::KeyName
  VpcCidrBlocks:
    Type: CommaDelimitedList
    Description: 'vpc, public subnet, private subnet'
```

6. Define a VpcCidrBlocks parameter (CommaDelimitedList type) for the VPC and subnet CIDR blocks, and reference it in Vpc, PublicSubnet, and PrivateSubnet resources.



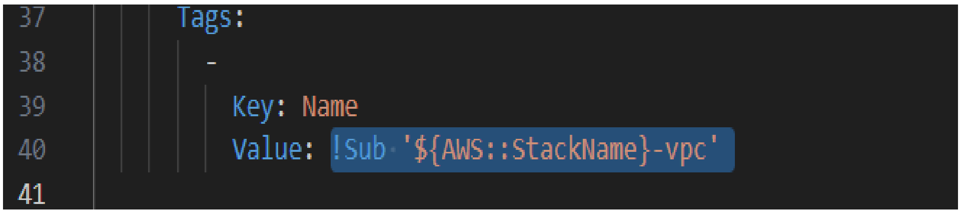
```
      CidrBlock: !Select [ 0, !Ref VpcCidrBlocks ]
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        -
          Key: Name
          Value: !Sub '${AWS::StackName}-vpc'

  # Subnets ---
  PublicSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Ref SubnetAZ
      CidrBlock: !Select [ 1, !Ref VpcCidrBlocks ]
```

7. Define a SubnetAZ parameter (AWS-specific Availability Zone type) and reference it in the PublicSubnet and PrivateSubnet resources.



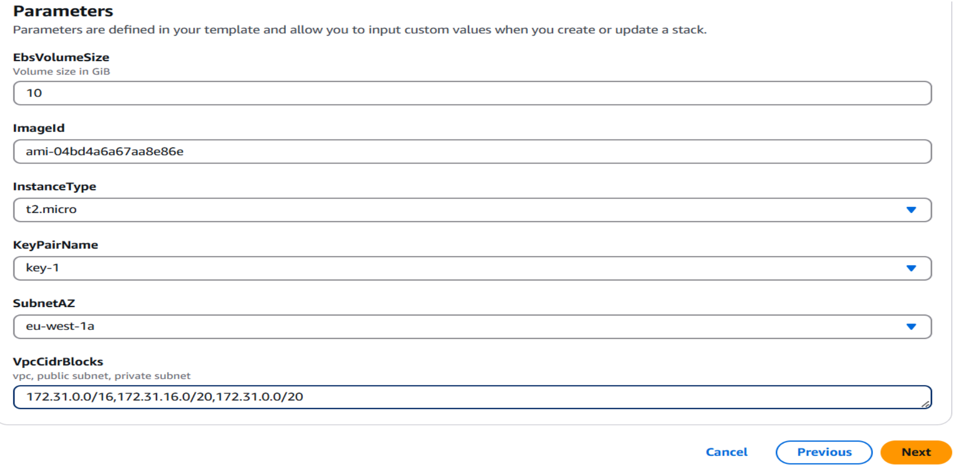
```
  # Subnets ---
  PublicSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Ref SubnetAZ
      CidrBlock: !Select [ 1, !Ref VpcCidrBlocks ]
      MapPublicIpOnLaunch: true
      VpcId: !Ref Vpc
      Tags:
        -
          Key: Name
          Value: Public Subnet

  PrivateSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Ref SubnetAZ
      CidrBlock: !Select [ 2, !Ref VpcCidrBlocks ]
```

8. Edit the Vpc name tag using Fn::Sub to append "-vpc" to the stack name.



```
      Tags:
        -
          Key: Name
          Value: !Sub '${AWS::StackName}-vpc'
```

9. Save the template('section-3-activity-solution-template') and upload it to the AWS CloudFormation Console to create a new stack(you can take reference to create a stack from the previous documents).
10. During stack creation, provide values for parameters such as EbsVolumeSize, InstanceType, ImageId, KeyName, SubnetAZ, and VpcCidrBlocks.



Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

EbsVolumeSize
Volume size in GiB
10

ImageId
ami-04bd4a6a67aa8e86e

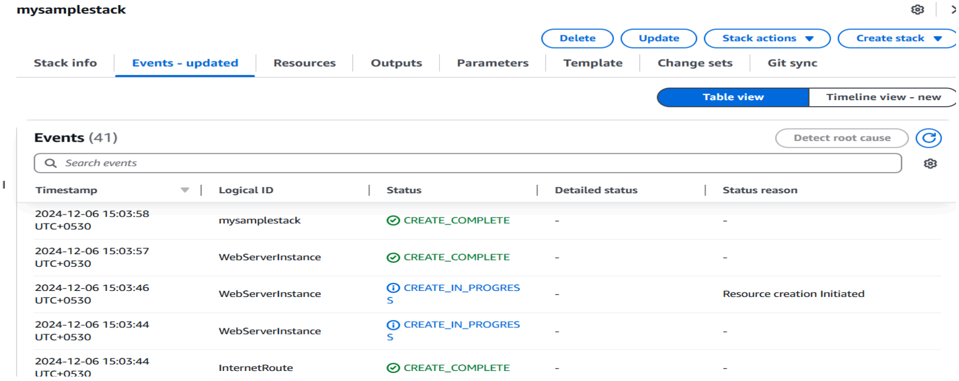
InstanceType
t2.micro

KeyPairName
key-1

SubnetAZ
eu-west-1a

VpcCidrBlocks
vpc, public subnet, private subnet
172.31.0.0/16,172.31.16.0/20,172.31.0.0/20

11. Verify the parameter values on the review page and create the stack.
12. Wait for the stack to finish creating successfully.

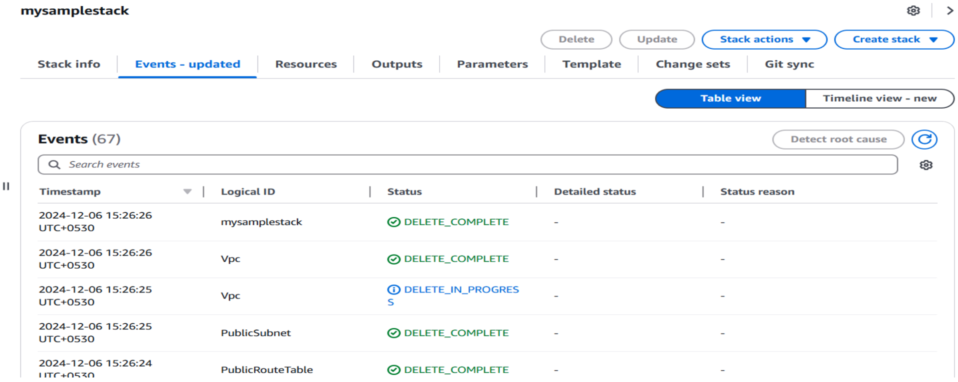


mysamplestack

Events (41)

Timestamp	Logical ID	Status	Detailed status	Status reason
2024-12-06 15:03:58 UTC+0530	mysamplestack	✓ CREATE_COMPLETE	-	-
2024-12-06 15:03:57 UTC+0530	WebServerInstance	✓ CREATE_COMPLETE	-	-
2024-12-06 15:03:46 UTC+0530	WebServerInstance	ⓘ CREATE_IN_PROGRESS	-	Resource creation Initiated
2024-12-06 15:03:44 UTC+0530	WebServerInstance	ⓘ CREATE_IN_PROGRESS	-	-
2024-12-06 15:03:44 UTC+0530	InternetRoute	✓ CREATE_COMPLETE	-	-

13. After completing the task, Clean up by deleting the stack.



mysamplestack

Events (67)

Timestamp	Logical ID	Status	Detailed status	Status reason
2024-12-06 15:26:26 UTC+0530	mysamplestack	✓ DELETE_COMPLETE	-	-
2024-12-06 15:26:26 UTC+0530	Vpc	✓ DELETE_COMPLETE	-	-
2024-12-06 15:26:25 UTC+0530	Vpc	ⓘ DELETE_IN_PROGRESS	-	-
2024-12-06 15:26:25 UTC+0530	PublicSubnet	✓ DELETE_COMPLETE	-	-
2024-12-06 15:26:24 UTC+0530	PublicRouteTable	✓ DELETE_COMPLETE	-	-