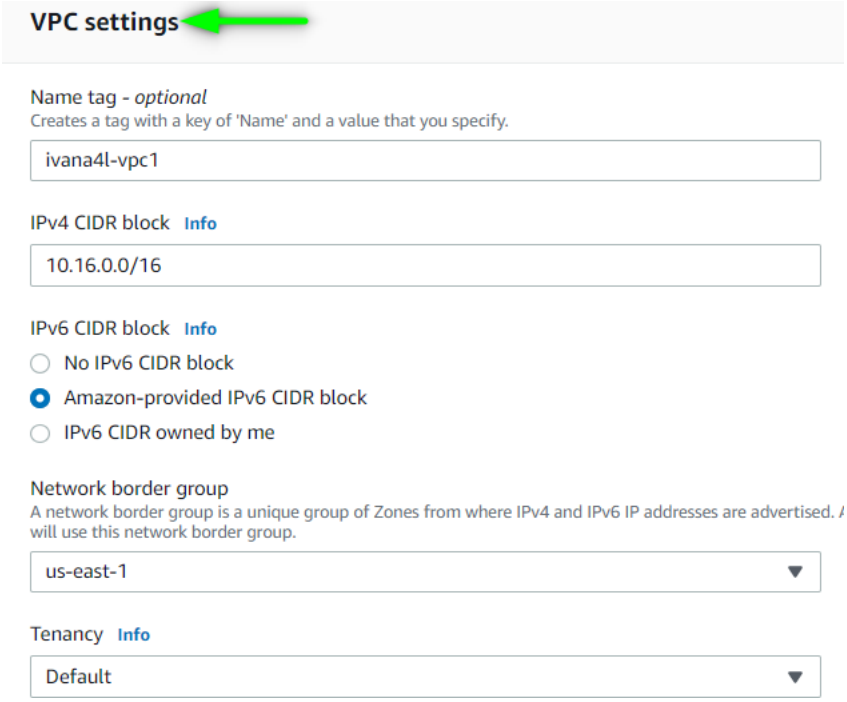
SUMMARY:

This Lab covers VPC fundamentals: VPC shell, mult-tier VPC's, EBS-snapshots-instance stores, NAT

Contents

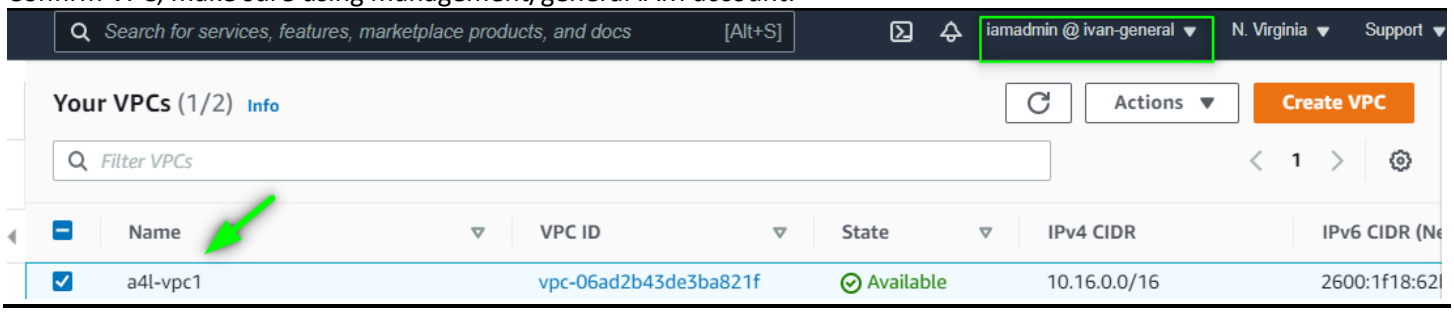
1. Create VPC Shell

implement the VPC shell for the Animals4life (A4L) organization in our accounts:
VPC settings:



Enable hostnames:

Confirm VPC, make sure using management/general IAM account:

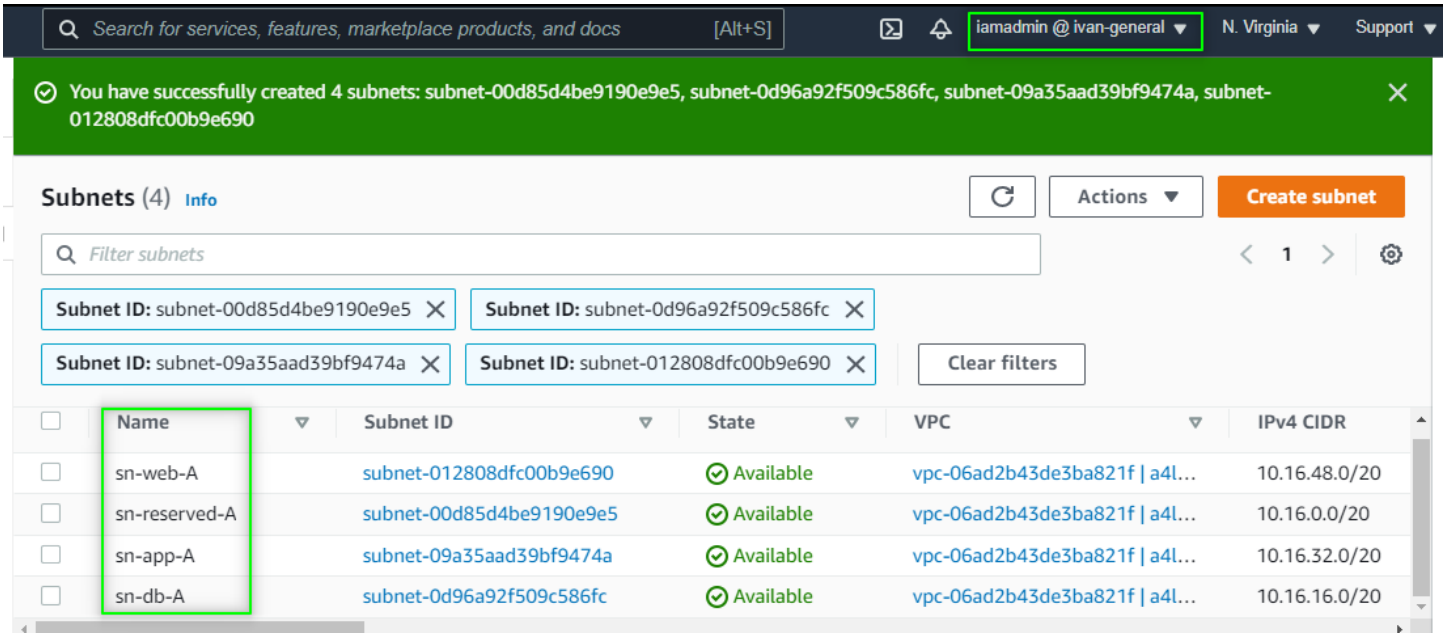


2. Implement Multi-tier VPC Subnets

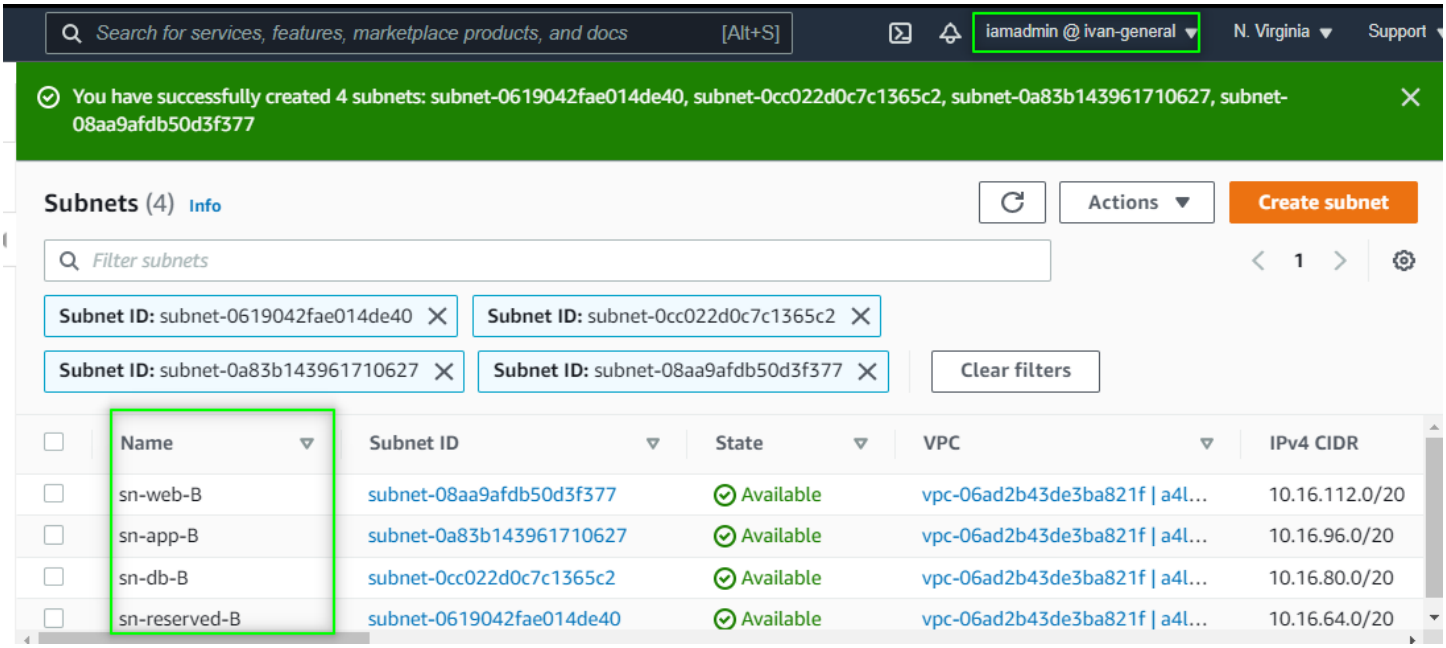
Subnet list for creation – creating 12 subnets, splitting into 3 availability zones (A-B-C), 1 reserved, 1 for database, 1 for web, 1 for app:

NAME CIDR AZ CustomIPv6Value
sn-reserved-A 10.16.0.0/20 AZA IPv6 00
sn-db-A 10.16.16.0/20 AZA IPv6 01
sn-app-A 10.16.32.0/20 AZA IPv6 02
sn-web-A 10.16.48.0/20 AZA IPv6 03
sn-reserved-B 10.16.64.0/20 AZB IPv6 04
sn-db-B 10.16.80.0/20 AZB IPv6 05
sn-app-B 10.16.96.0/20 AZB IPv6 06
sn-web-B 10.16.112.0/20 AZB IPv6 07
sn-reserved-C 10.16.128.0/20 AZC IPv6 08
sn-db-C 10.16.144.0/20 AZC IPv6 09
sn-app-C 10.16.160.0/20 AZC IPv6 0A
sn-web-C 10.16.176.0/20 AZC IPv6 0B

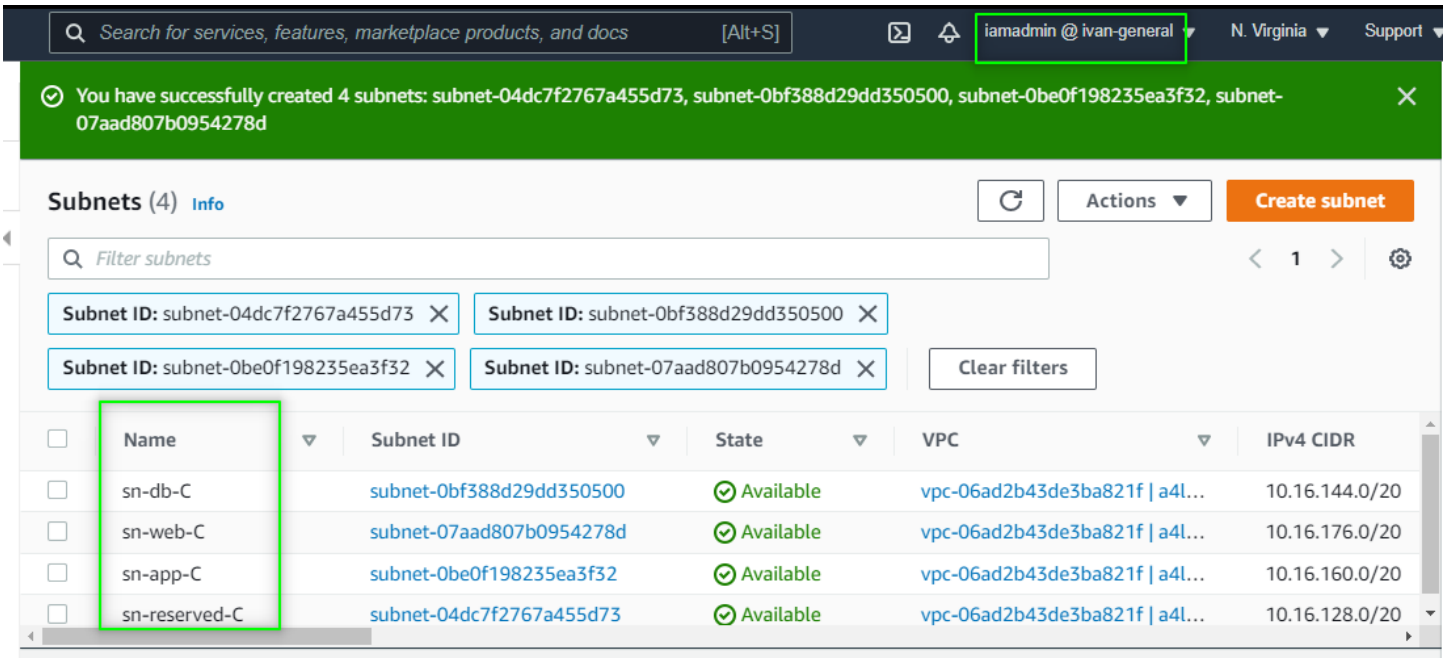
Subnets for AZ – A:



Subnets for AZ – B:



Subnets for AZ – C:



Confirm total list, should be 12:

sn-app-A	subnet-09a35aad39bf9474a	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.32.0/20
sn-app-B	subnet-0a83b143961710627	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.96.0/20
sn-app-C	subnet-0be0f198235ea3f32	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.160.0/2(
sn-db-A	subnet-0d96a92f509c586fc	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.16.0/20
sn-db-B	subnet-0cc022d0c7c1365c2	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.80.0/20
sn-db-C	subnet-0bf388d29dd350500	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.144.0/2(
sn-reserved-A	subnet-00d85d4be9190e9e5	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.0.0/20
sn-reserved-B	subnet-0619042fae014de40	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.64.0/20
sn-reserved-C	subnet-04dc7f2767a455d73	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.128.0/2(
sn-web-A	subnet-012808dfc00b9e690	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.48.0/20
sn-web-B	subnet-08aa9afdb50d3f377	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.112.0/2(
sn-web-C	subnet-07aad807b0954278d	Available	vpc-06ad2b43de3ba821f | a4l...	10.16.176.0/2(

Enable Auto IPv6 addresses for all:

***This can be automated, but it is important to know how to manually create a VPC subnet.

3. EBS, snapshots, and instance store volumes

In this [DEMO] lesson you get a chance to interact with EBS, Instance Store Volumes and EC2

- Create an EBS Volume
- Mount it to an EC2 instance

- Create and Mount a file system
- Generate a test file
- Migrate the volume to another EC2 instance in the same AZ
- Verify the file system and file are intact
- Create a EBS SNapshot from the volume
- Create a new EBS Volume in AZ-B
- Verify the filesystem and file are intact
- Copy the snapshot to another region
- Create an EC2 instance with instance store volumes
- Create a filesystem and test file
- Restart instance and verify the file system is intact
- Stop and Start the instance
- Verify the file system is no longer present - new EC2 Host.

Create CFN stack, VPC CFN file used:

```
Description:  Animals4Life base VPC Template
  It will be used anywhere where AWS product and service [DEMO] lessons require a VPC to be i
nplace and functional
  Optional additions to the VPC Template (to save costs)
  A4L_BastionHost - Deploys a bastionHost to access the VPC Private Resources (Can be removed
 to save costs)
  A4L_NatGateways - Deploys 3 NatGateways (this has a base cost, can be omitted if no private
 subnet network access is required - or removed when not studying to save costs)
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.16.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: a4l-vpc1
  IPv6CidrBlock:
    Type: AWS::EC2::VPCCidrBlock
    Properties:
      VpcId: !Ref VPC
      AmazonProvidedIpv6CidrBlock: true
  InternetGateway:
    Type: 'AWS::EC2::InternetGateway'
    Properties:
```

```
    Tags:
    - Key: Name
      Value: A4L-vpc1-igw
InternetGatewayAttachment:
  Type: 'AWS::EC2::VPCGatewayAttachment'
  Properties:
    VpcId: !Ref VPC
    InternetGatewayId: !Ref InternetGateway
RouteTableWeb:
  Type: 'AWS::EC2::RouteTable'
  Properties:
    VpcId: !Ref VPC
    Tags:
    - Key: Name
      Value: A4L-vpc1-rt-web
RouteTableWebDefaultIPv4:
  Type: 'AWS::EC2::Route'
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId:
      Ref: RouteTableWeb
    DestinationCidrBlock: '0.0.0.0/0'
    GatewayId:
      Ref: InternetGateway
RouteTableWebDefaultIPv6:
  Type: 'AWS::EC2::Route'
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId:
      Ref: RouteTableWeb
    DestinationIpv6CidrBlock: '::/0'
    GatewayId:
      Ref: InternetGateway
RouteTableAssociationWebA:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    SubnetId: !Ref SubnetWEBA
    RouteTableId:
      Ref: RouteTableWeb
RouteTableAssociationWebB:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    SubnetId: !Ref SubnetWEBB
    RouteTableId:
      Ref: RouteTableWeb
RouteTableAssociationWebC:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    SubnetId: !Ref SubnetWEBC
    RouteTableId:
      Ref: RouteTableWeb
SubnetReservedA:
```

```
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: 10.16.0.0/20
      AssignIpv6AddressOnCreation: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '00::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-reserved-A
  SubnetReservedB:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs '' ]
      CidrBlock: 10.16.64.0/20
      AssignIpv6AddressOnCreation: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '04::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-reserved-B
  SubnetReservedC:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 2, !GetAZs '' ]
      CidrBlock: 10.16.128.0/20
      AssignIpv6AddressOnCreation: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '08::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-reserved-C
  SubnetDBA:
    Type: AWS::EC2::Subnet
```

```
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: 10.16.16.0/20
      AssignIpv6AddressOnCreation: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '01::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-db-A
  SubnetDBB:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs '' ]
      CidrBlock: 10.16.80.0/20
      AssignIpv6AddressOnCreation: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '05::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-db-B
  SubnetDBC:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 2, !GetAZs '' ]
      CidrBlock: 10.16.144.0/20
      AssignIpv6AddressOnCreation: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '09::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-db-C
  SubnetAPPA:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
```

```
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: 10.16.32.0/20
      AssignIpv6AddressOnCreation: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '02::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-app-A
  SubnetAPPB:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs '' ]
      CidrBlock: 10.16.96.0/20
      AssignIpv6AddressOnCreation: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '06::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-app-B
  SubnetAPPC:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 2, !GetAZs '' ]
      CidrBlock: 10.16.160.0/20
      AssignIpv6AddressOnCreation: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '0A::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-app-C
  SubnetWEBA:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
```

```
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: 10.16.48.0/20
      MapPublicIpOnLaunch: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '03::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-web-A
  SubnetWEBB:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs '' ]
      CidrBlock: 10.16.112.0/20
      MapPublicIpOnLaunch: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '07::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-web-B
  SubnetWEBC:
    Type: AWS::EC2::Subnet
    DependsOn: IPv6CidrBlock
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 2, !GetAZs '' ]
      CidrBlock: 10.16.176.0/20
      MapPublicIpOnLaunch: true
      Ipv6CidrBlock:
        Fn::Sub:
          - "${VpcPart}${SubnetPart}"
          - SubnetPart: '0B::/64'
            VpcPart: !Select [ 0, !Split [ '00::/56', !Select [ 0, !GetAtt VPC.Ipv6CidrBlocks
]]]
      Tags:
        - Key: Name
          Value: sn-web-C
  IPv6WorkaroundSubnetWEBA:
    Type: Custom::SubnetModify
    Properties:
      ServiceToken: !GetAtt IPv6WorkaroundLambda.Arn
      SubnetId: !Ref SubnetWEBA
```

```
  IPv6WorkaroundSubnetWEBB:
    Type: Custom::SubnetModify
    Properties:
      ServiceToken: !GetAtt IPv6WorkaroundLambda.Arn
      SubnetId: !Ref SubnetWEBB
  IPv6WorkaroundSubnetWEBC:
    Type: Custom::SubnetModify
    Properties:
      ServiceToken: !GetAtt IPv6WorkaroundLambda.Arn
      SubnetId: !Ref SubnetWEBC
  IPv6WorkaroundRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: "/"
      Policies:
        - PolicyName: !Sub "ipv6-fix-logs-${AWS::StackName}"
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
            - Effect: Allow
              Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
              Resource: arn:aws:logs:*:*:*
        - PolicyName: !Sub "ipv6-fix-modify-${AWS::StackName}"
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
            - Effect: Allow
              Action:
              - ec2:ModifySubnetAttribute
              Resource: "*"
  IPv6WorkaroundLambda:
    Type: AWS::Lambda::Function
    Properties:
      Handler: "index.lambda_handler"
      Code: #import cfnresponse below required to send respose back to CFN
        ZipFile:
          Fn::Sub: |
            import cfnresponse
            import boto3
```

```
            def lambda_handler(event, context):
                if event['RequestType'] is 'Delete':
                  cfnresponse.send(event, context, cfnresponse.SUCCESS)
                  return

                responseValue = event['ResourceProperties']['SubnetId']
                ec2 = boto3.client('ec2', region_name='${AWS::Region}')
                ec2.modify_subnet_attribute(AssignIpv6AddressOnCreation={
                                                 'Value': True
                                              },
                                              SubnetId=responseValue)
                responseData = {}
                responseData['SubnetId'] = responseValue
                cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData, "CustomRe
sourcePhysicalID")
      Runtime: python2.7
      Role: !GetAtt IPv6WorkaroundRole.Arn
      Timeout: 30
Outputs:
  a4lvpc1:
    Description: Animals4Life VPC1_ID
    Value: !Ref VPC
    Export:
      Name: a4l-vpc1
  a4lvpc1subnetweba:
    Description: Animals4Life VPC1 SubnetWEBA
    Value: !Ref SubnetWEBA
    Export:
      Name: a4l-vpc1-subnet-weba
  a4lvpc1subnetwebb:
    Description: Animals4Life VPC1 SubnetWEBB
    Value: !Ref SubnetWEBB
    Export:
      Name: a4l-vpc1-subnet-webb
  a4lvpc1subnetwebc:
    Description: Animals4Life VPC1 SubnetWEBC
    Value: !Ref SubnetWEBC
    Export:
      Name: a4l-vpc1-subnet-webc
  a4lvpc1subnetappa:
    Description: Animals4Life VPC1 SubnetAPPA
    Value: !Ref SubnetAPPA
    Export:
      Name: a4l-vpc1-subnet-appa
  a4lvpc1subnetappb:
    Description: Animals4Life VPC1 SubnetAPPB
    Value: !Ref SubnetAPPB
    Export:
      Name: a4l-vpc1-subnet-appb
  a4lvpc1subnetappc:
    Description: Animals4Life VPC1 SubnetAPPC
    Value: !Ref SubnetAPPC
```

```
    Export:
      Name: a4l-vpc1-subnet-appc
  a4lvpc1subnetdba:
    Description: Animals4Life VPC1 SubnetDBA
    Value: !Ref SubnetDBA
    Export:
      Name: a4l-vpc1-subnet-dba
  a4lvpc1subnetdbb:
    Description: Animals4Life VPC1 SubnetDBB
    Value: !Ref SubnetDBB
    Export:
      Name: a4l-vpc1-subnet-dbb
  a4lvpc1subnetdbc:
    Description: Animals4Life VPC1 SubnetDBC
    Value: !Ref SubnetDBC
    Export:
      Name: a4l-vpc1-subnet-dbc
  a4lvpc1subnetreserveda:
    Description: Animals4Life VPC1 SubnetReservedA
    Value: !Ref SubnetReservedA
    Export:
      Name: a4l-vpc1-subnet-reserveda
  a4lvpc1subnetreservedb:
    Description: Animals4Life VPC1 SubnetReservedB
    Value: !Ref SubnetReservedB
    Export:
      Name: a4l-vpc1-subnet-reservedb
  a4lvpc1subnetreservedc:
    Description: Animals4Life VPC1 SubnetReservedC
    Value: !Ref SubnetReservedC
    Export:
      Name: a4l-vpc1-subnet-reservedc
```

EBS Instance CFN used:

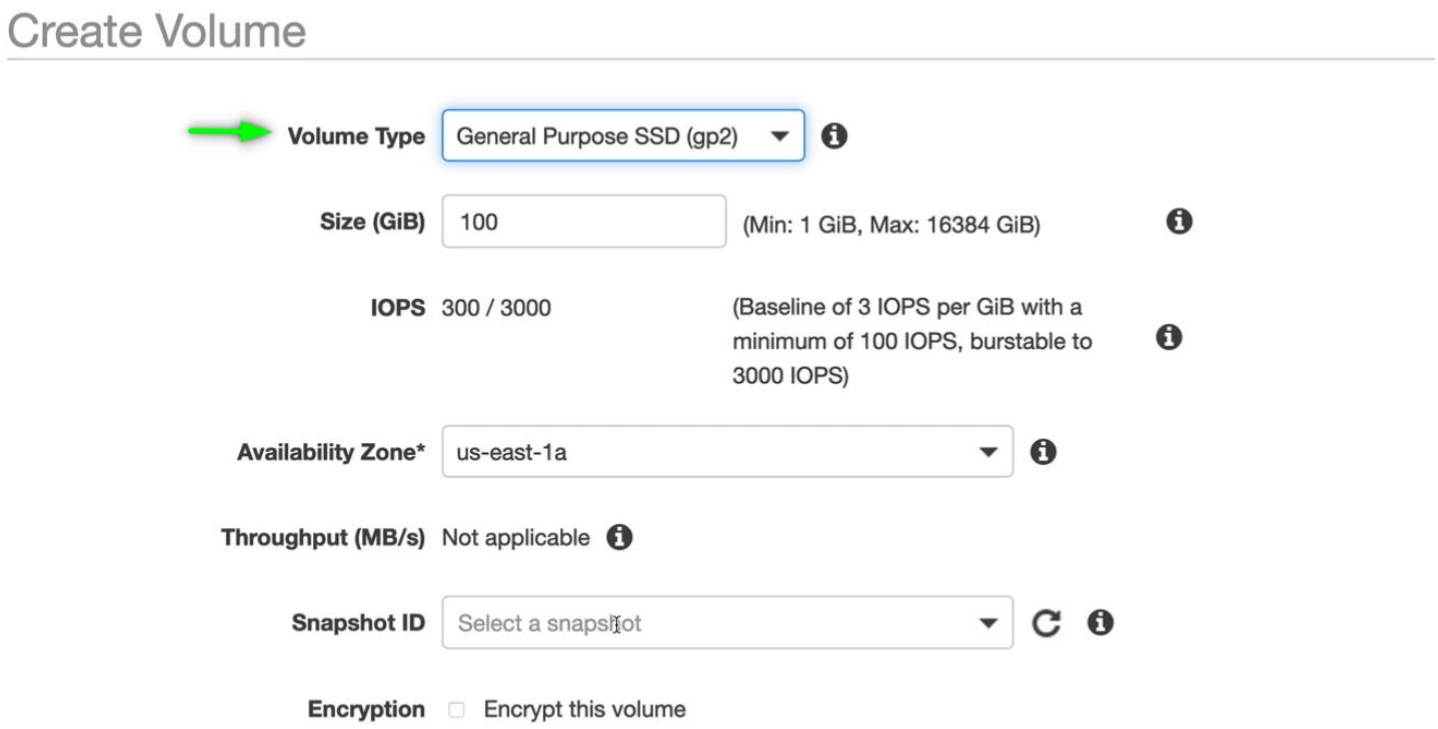
```
Description:  Create two instances in AZ-A and one in AZ-B for ebs_demo
Parameters:
  LatestAmiId:
    Description: AMI for Bastion Host (default is latest AmaLinux2)
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'
  KeyName:
    Type: AWS::EC2::KeyPair::KeyName
    Description: "Name of an existing SSH Keypair to access the instance"
Resources:
  Instance1:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: "t2.micro"
      ImageId: !Ref LatestAmiId
```

```
    IamInstanceProfile: !Ref SessionManagerInstanceProfile
    SubnetId: !ImportValue a4l-vpc1-subnet-weba
    SecurityGroupIds:
      - !Ref InstanceSecurityGroup
    Tags:
      - Key: Name
        Value: A4L-EBS-INSTANCE1-AZA
  Instance2:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: "t2.micro"
      ImageId: !Ref LatestAmiId
      IamInstanceProfile: !Ref SessionManagerInstanceProfile
      SubnetId: !ImportValue a4l-vpc1-subnet-weba
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
      Tags:
        - Key: Name
          Value: A4L-EBS-INSTANCE2-AZA
  Instance3:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: "t2.micro"
      ImageId: !Ref LatestAmiId
      IamInstanceProfile: !Ref SessionManagerInstanceProfile
      SubnetId: !ImportValue a4l-vpc1-subnet-webb
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
      Tags:
        - Key: Name
          Value: A4L-EBS-INSTANCE3-AZB
  InstanceSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      VpcId: !ImportValue a4l-vpc1
      GroupDescription: Enable SSH access via port 22 IPv4 & v6
      SecurityGroupIngress:
        - Description: 'Allow SSH IPv4 IN'
          IpProtocol: tcp
          FromPort: '22'
          ToPort: '22'
          CidrIp: '0.0.0.0/0'
        - Description: 'Allow SSH IPv6 IN'
          IpProtocol: tcp
          FromPort: '22'
          ToPort: '22'
          CidrIpv6: ::/0
  SessionManagerRole:
    Type: 'AWS::IAM::Role'
    Properties:
```

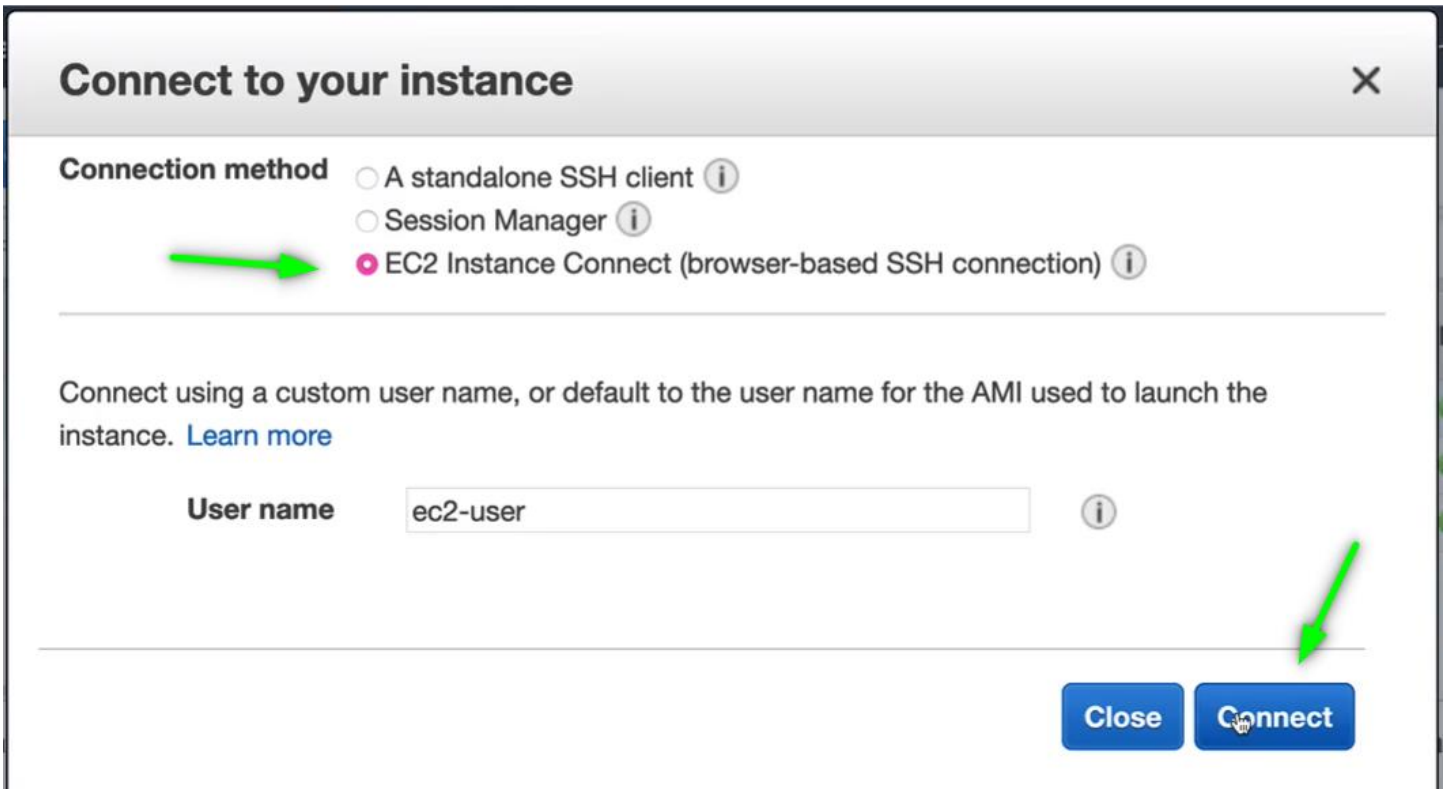
```
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
          Action:
            - 'sts:AssumeRole'
    Path: /
    Policies:
      - PolicyName: root
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Effect: Allow
              Action:
                - 'ssm:DescribeAssociation'
                - 'ssm:GetDeployablePatchSnapshotForInstance'
                - 'ssm:GetDocument'
                - 'ssm:DescribeDocument'
                - 'ssm:GetManifest'
                - 'ssm:GetParameter'
                - 'ssm:GetParameters'
                - 'ssm:ListAssociations'
                - 'ssm:ListInstanceAssociations'
                - 'ssm:PutInventory'
                - 'ssm:PutComplianceItems'
                - 'ssm:PutConfigurePackageResult'
                - 'ssm:UpdateAssociationStatus'
                - 'ssm:UpdateInstanceAssociationStatus'
                - 'ssm:UpdateInstanceInformation'
              Resource: '*'
            - Effect: Allow
              Action:
                - 'ssmmessages:CreateControlChannel'
                - 'ssmmessages:CreateDataChannel'
                - 'ssmmessages:OpenControlChannel'
                - 'ssmmessages:OpenDataChannel'
              Resource: '*'
            - Effect: Allow
              Action:
                - 'ec2messages:AcknowledgeMessage'
                - 'ec2messages:DeleteMessage'
                - 'ec2messages:FailMessage'
                - 'ec2messages:GetEndpoint'
                - 'ec2messages:GetMessages'
                - 'ec2messages:SendReply'
              Resource: '*'
  SessionManagerInstanceProfile:
    Type: 'AWS::IAM::InstanceProfile'
    Properties:
```

```
      Path: /
      Roles:
        - !Ref SessionManagerRole
```

Create volume (s)



Connect to ec2 instance:



Locate and add UUID:

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 26 available
Run "sudo yum update" to apply all updates.

[ec2-user@ip-10-16-60-135 ~]$
[ec2-user@ip-10-16-60-135 ~]$ df -k
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          485472       0    485472   0% /dev
tmpfs             503484       0    503484   0% /dev/shm
tmpfs             503484     436    503048   1% /run
tmpfs             503484       0    503484   0% /sys/fs/cgroup
/dev/xvda1       8376300 1308112   7068188  16% /
tmpfs             100700       0    100700   0% /run/user/1000
[ec2-user@ip-10-16-60-135 ~]$ sudo blkid
/dev/xvda1: LABEL="/" UUID="7a487823-831a-47e0-b9c5-97a7edc90077" TYPE="xfs" PARTLABEL="Linux" PARTUUID="a30d26f6-9e8c-4293-87da-62
1d64876e9e"
/dev/xvdf: UUID="534c326e-12cb-410d-ba99-f26f543fcbe3" TYPE="xfs"
[ec2-user@ip-10-16-60-135 ~]$ sudo nano /etc/fstab
```

Verity volume appears:

```
[ec2-user@ip-10-16-60-135 ~]$ sudo mount -a
[ec2-user@ip-10-16-60-135 ~]$ df -k
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          485472       0    485472   0% /dev
tmpfs             503484       0    503484   0% /dev/shm
tmpfs             503484     440    503044   1% /run
tmpfs             503484       0    503484   0% /sys/fs/cgroup
/dev/xvda1       8376300 1307776   7068524  16% /
tmpfs             100700       0    100700   0% /run/user/1000
/dev/xvdf       10475520   43476  10432044   1% /ebstest
[ec2-user@ip-10-16-60-135 ~]$
```

Create snapshot:

Snapshots > Create Volume

Create Volume

Snapshot ID snap-05604b30999dfdd75

Volume Type General Purpose SSD (gp2)

Size (GiB) 10 (Min: 1 GiB, Max: 16384 GiB)

IOPS 100 / 3000 (Baseline of 3 IOPS per GiB with a minimum of 100 IOPS, burstable to 3000 IOPS)

Availability Zone* us-east-1b

Fast Snapshot Restore Not enabled

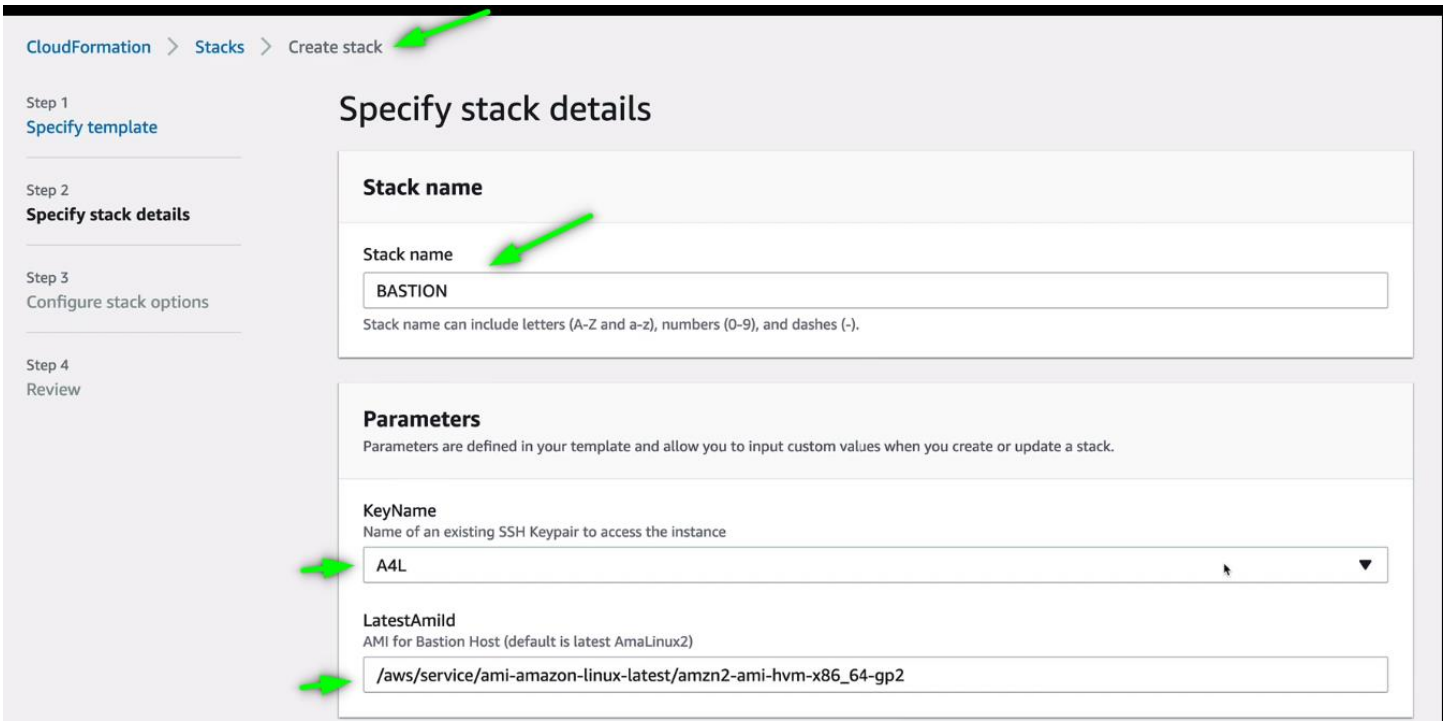
---

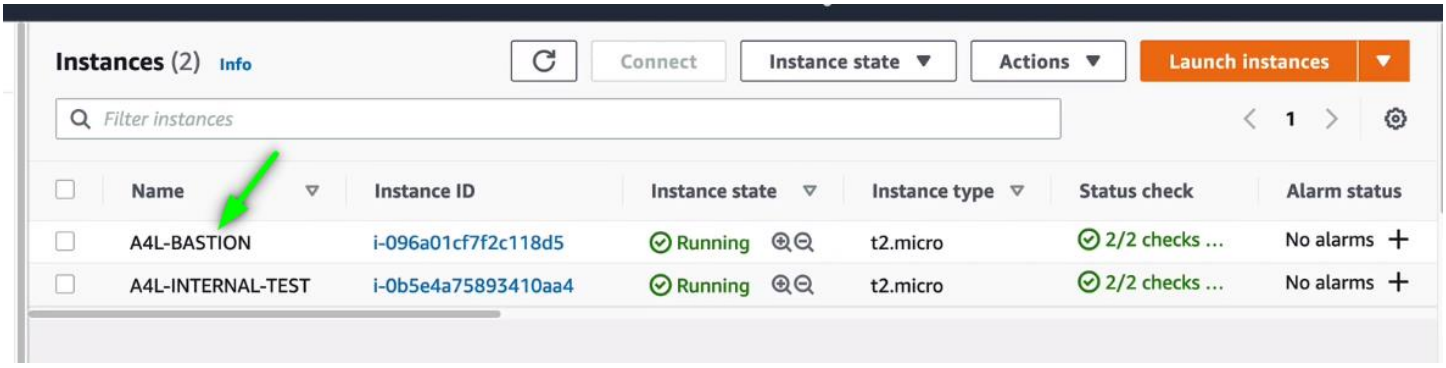
4. Implement private internet access using NAT

- In this [DEMO] lesson you will implement a highly-available regionally resilient NAT Gateway solution within the Animals4life VPC.
- In the first part - you will setup the demo and be ready to get started in PART2
- You will apply the VPC template, Bastion Template and create an internet test instance within the Animals for life VPC.
- In part two you will create three Nat Gateways

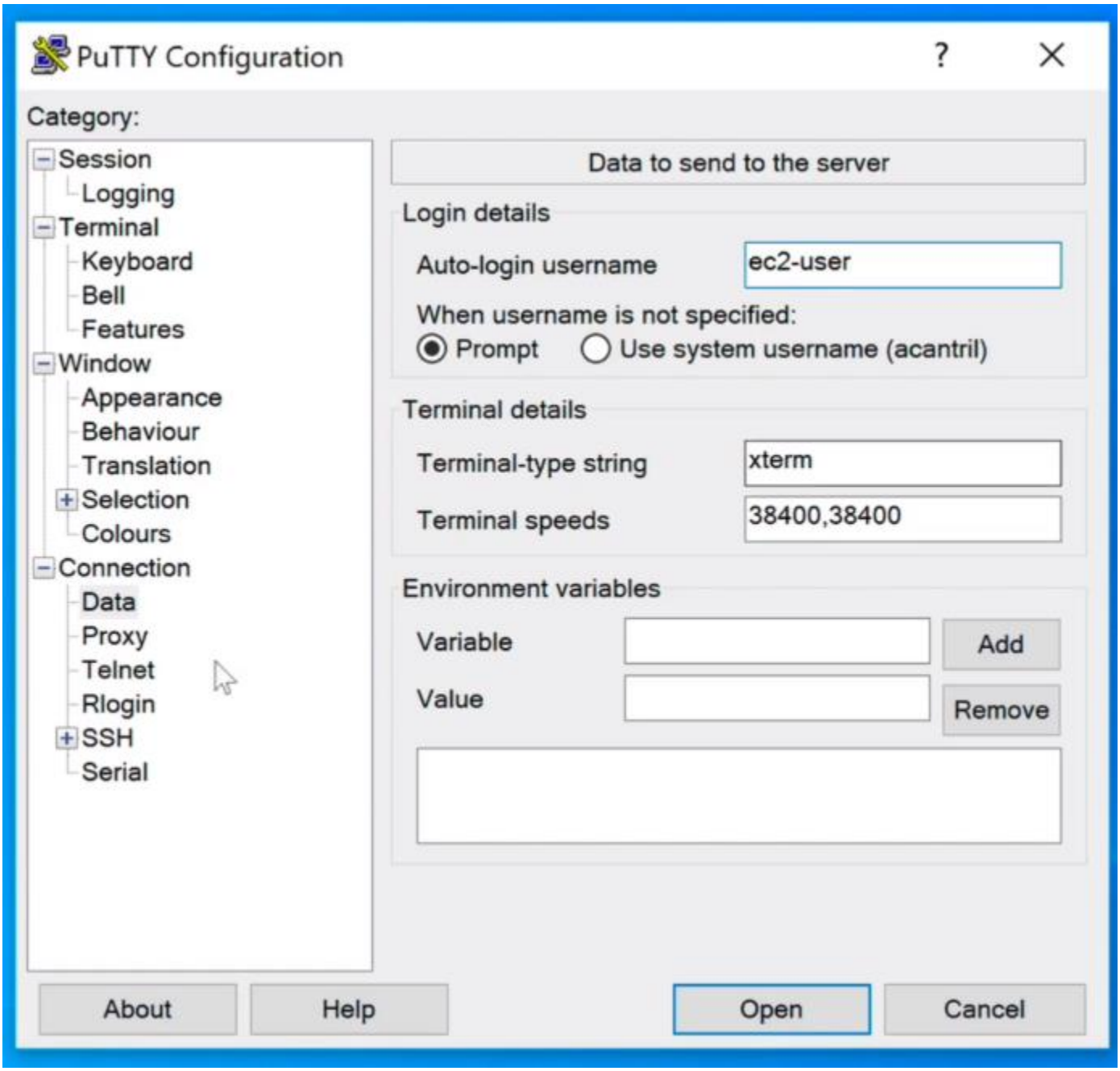
- Create Route tables and Default Routes with the Nat Gateway as a target and finally associate those Route Tables with the Reserved, App and DB subnets in AZ A, B and C before testing the solution using the internet instance.
- If you are using windows or linux I've included extra steps in the video with instructions

Create bastion host stack:

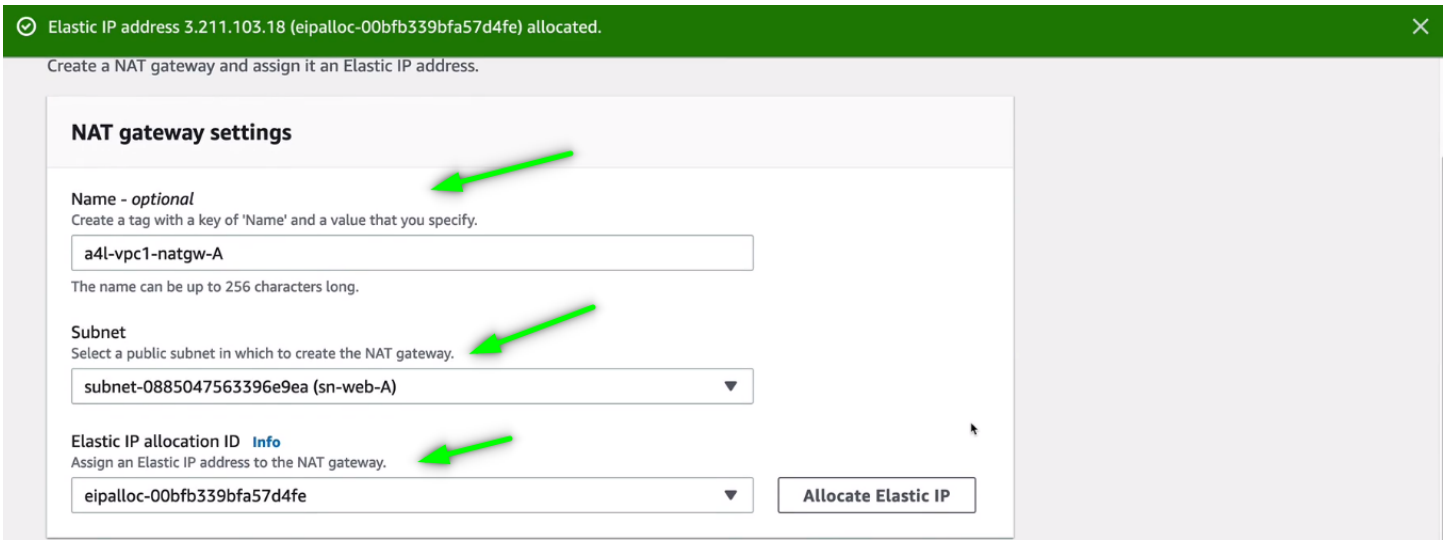


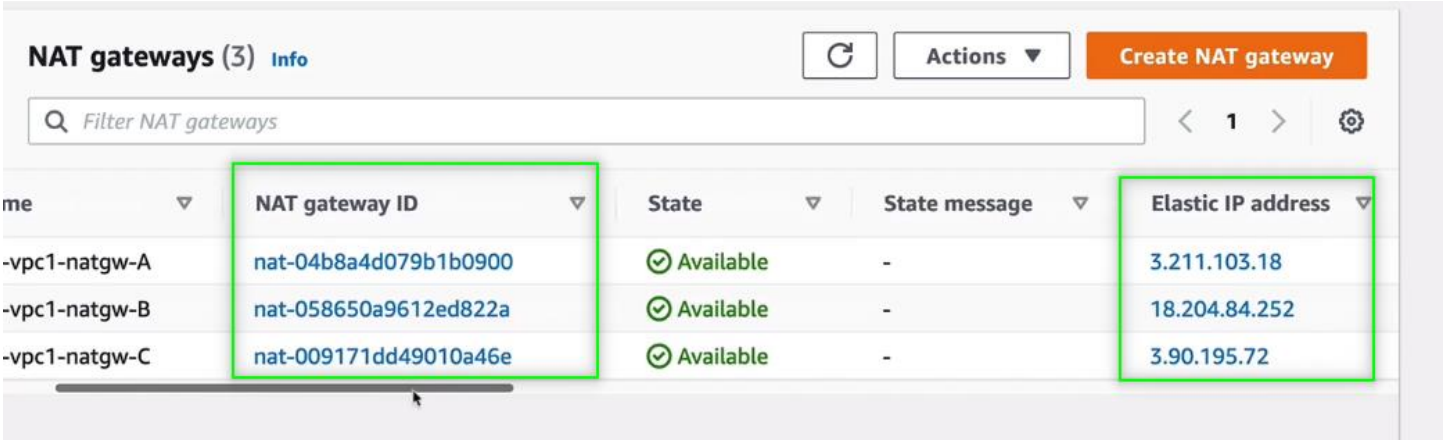


Setup putty for ssh connection:

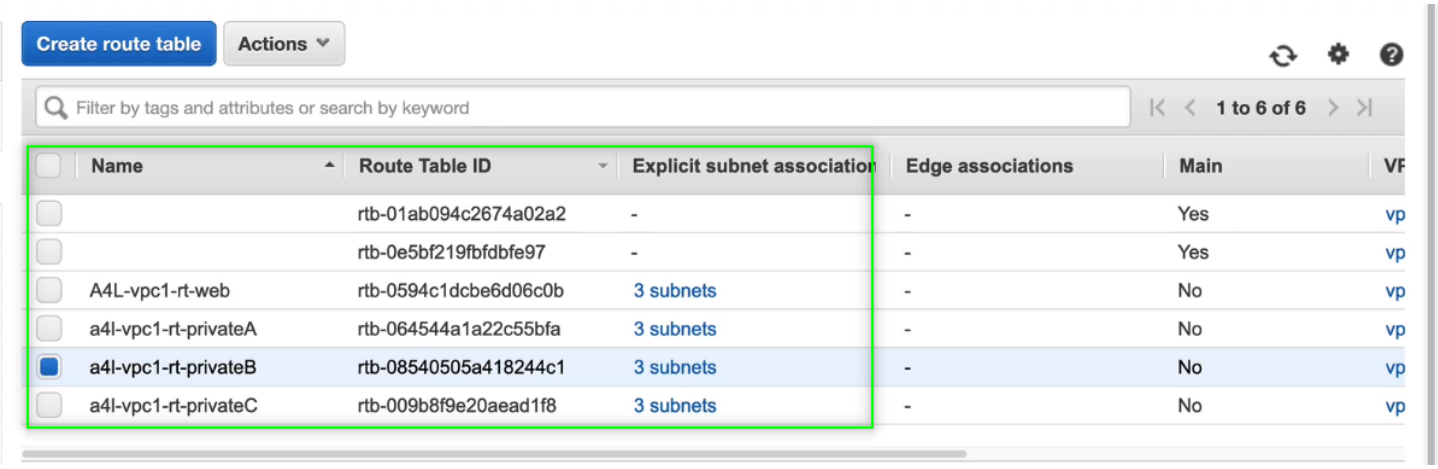


Create NAT gateways





Configure routing tables:



Test connection (ping):

```
10 packets transmitted, 0 received, 100% packet loss, time 9209ms

[ec2-user@ip-10-16-97-152 ~]$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=228 ttl=53 time=1.53 ms
64 bytes from 1.1.1.1: icmp_seq=229 ttl=53 time=1.01 ms
64 bytes from 1.1.1.1: icmp_seq=230 ttl=53 time=0.950 ms
64 bytes from 1.1.1.1: icmp_seq=231 ttl=53 time=0.846 ms
64 bytes from 1.1.1.1: icmp_seq=232 ttl=53 time=0.888 ms
64 bytes from 1.1.1.1: icmp_seq=233 ttl=53 time=0.895 ms
64 bytes from 1.1.1.1: icmp_seq=234 ttl=53 time=0.903 ms
64 bytes from 1.1.1.1: icmp_seq=235 ttl=53 time=0.856 ms
64 bytes from 1.1.1.1: icmp_seq=236 ttl=53 time=0.859 ms
64 bytes from 1.1.1.1: icmp_seq=237 ttl=53 time=0.902 ms
64 bytes from 1.1.1.1: icmp_seq=238 ttl=53 time=0.819 ms
64 bytes from 1.1.1.1: icmp_seq=239 ttl=53 time=0.896 ms
64 bytes from 1.1.1.1: icmp_seq=240 ttl=53 time=0.899 ms
64 bytes from 1.1.1.1: icmp_seq=241 ttl=53 time=0.938 ms
64 bytes from 1.1.1.1: icmp_seq=242 ttl=53 time=0.920 ms
64 bytes from 1.1.1.1: icmp_seq=243 ttl=53 time=0.895 ms
64 bytes from 1.1.1.1: icmp_seq=244 ttl=53 time=0.868 ms
64 bytes from 1.1.1.1: icmp_seq=245 ttl=53 time=0.961 ms
64 bytes from 1.1.1.1: icmp_seq=246 ttl=53 time=0.931 ms
64 bytes from 1.1.1.1: icmp_seq=247 ttl=53 time=0.958 ms
64 bytes from 1.1.1.1: icmp_seq=248 ttl=53 time=0.908 ms
64 bytes from 1.1.1.1: icmp_seq=249 ttl=53 time=0.904 ms
64 bytes from 1.1.1.1: icmp_seq=250 ttl=53 time=0.898 ms
64 bytes from 1.1.1.1: icmp_seq=251 ttl=53 time=0.884 ms
64 bytes from 1.1.1.1: icmp_seq=252 ttl=53 time=0.978 ms
```