SUMMARY:

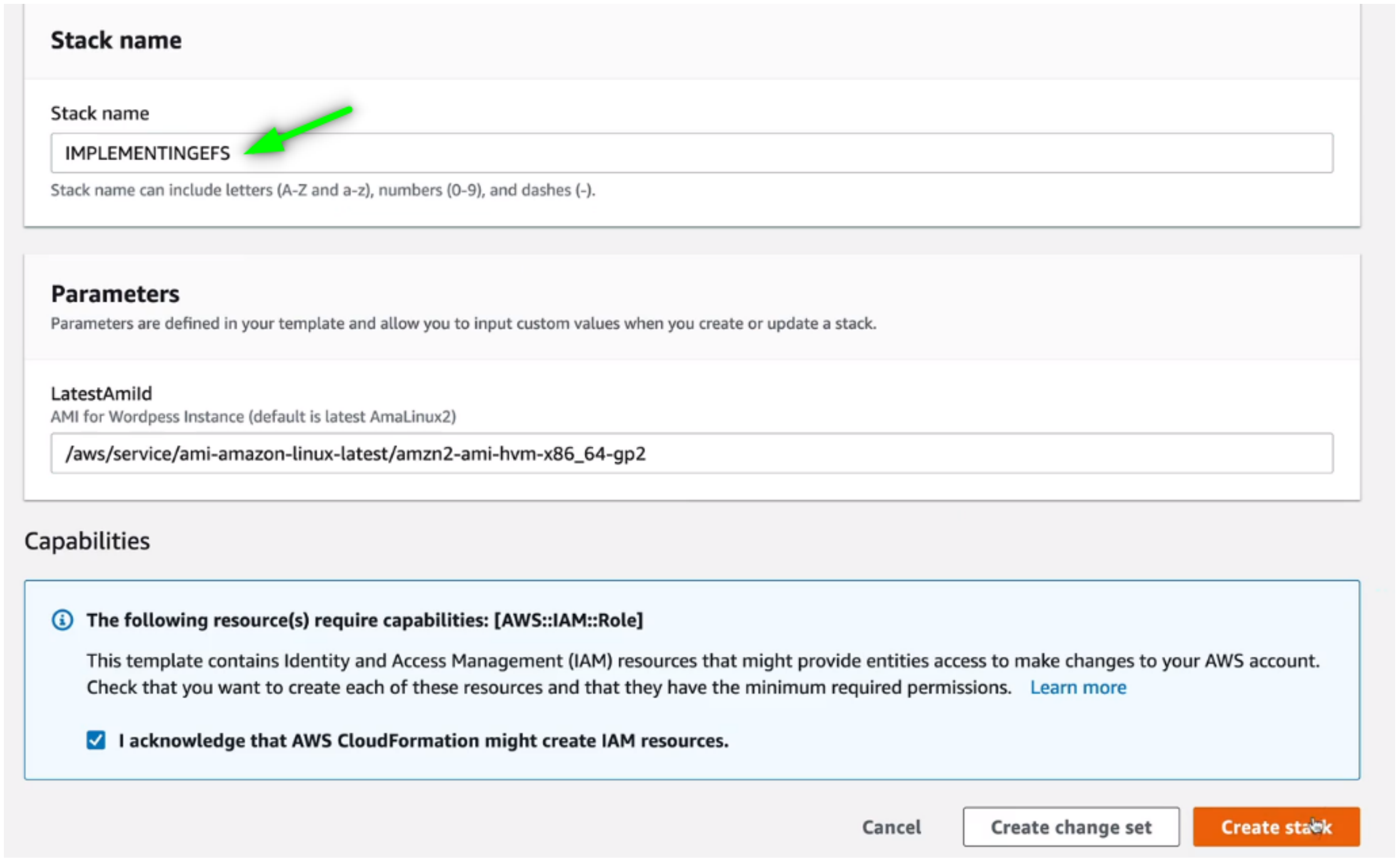
This Lab covers EFS – Implementing EFS, using EFS with Wordpress through CFN yaml

Contents

1. Implementing EFS

In this demo lesson you will implement a simple EFS file system in a VPC, configure mount targets and configure two EC2 instances to mount the file system within a mount point.

Create EFS stack


Stack name

Stack name: IMPLEMENTINGEFS

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

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

LatestAmiId

AMI for Wordpess Instance (default is latest AmaLinux2)

/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2

Capabilities

The following resource(s) require capabilities: [AWS::IAM::Role]

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. Learn more

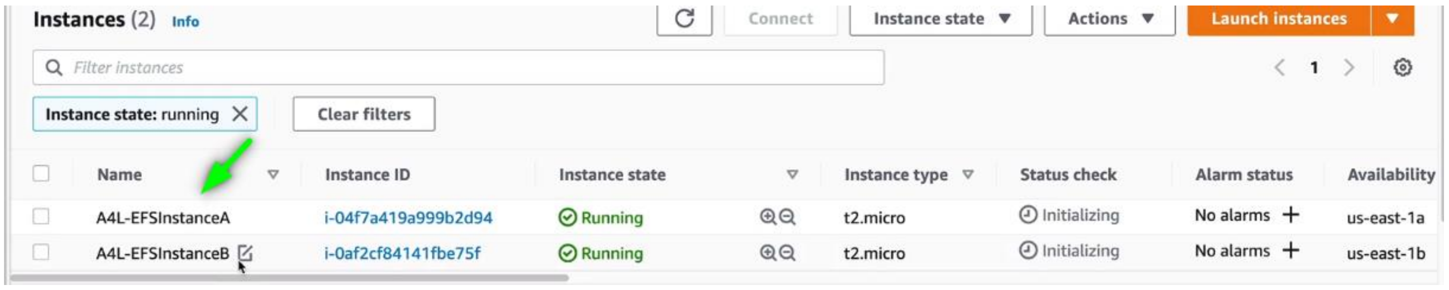
✓ I acknowledge that AWS CloudFormation might create IAM resources.

Commands used:

```
# INSTANCE A

yum -y install amazon-efs-utils

sudo ls -la /
df -k
```

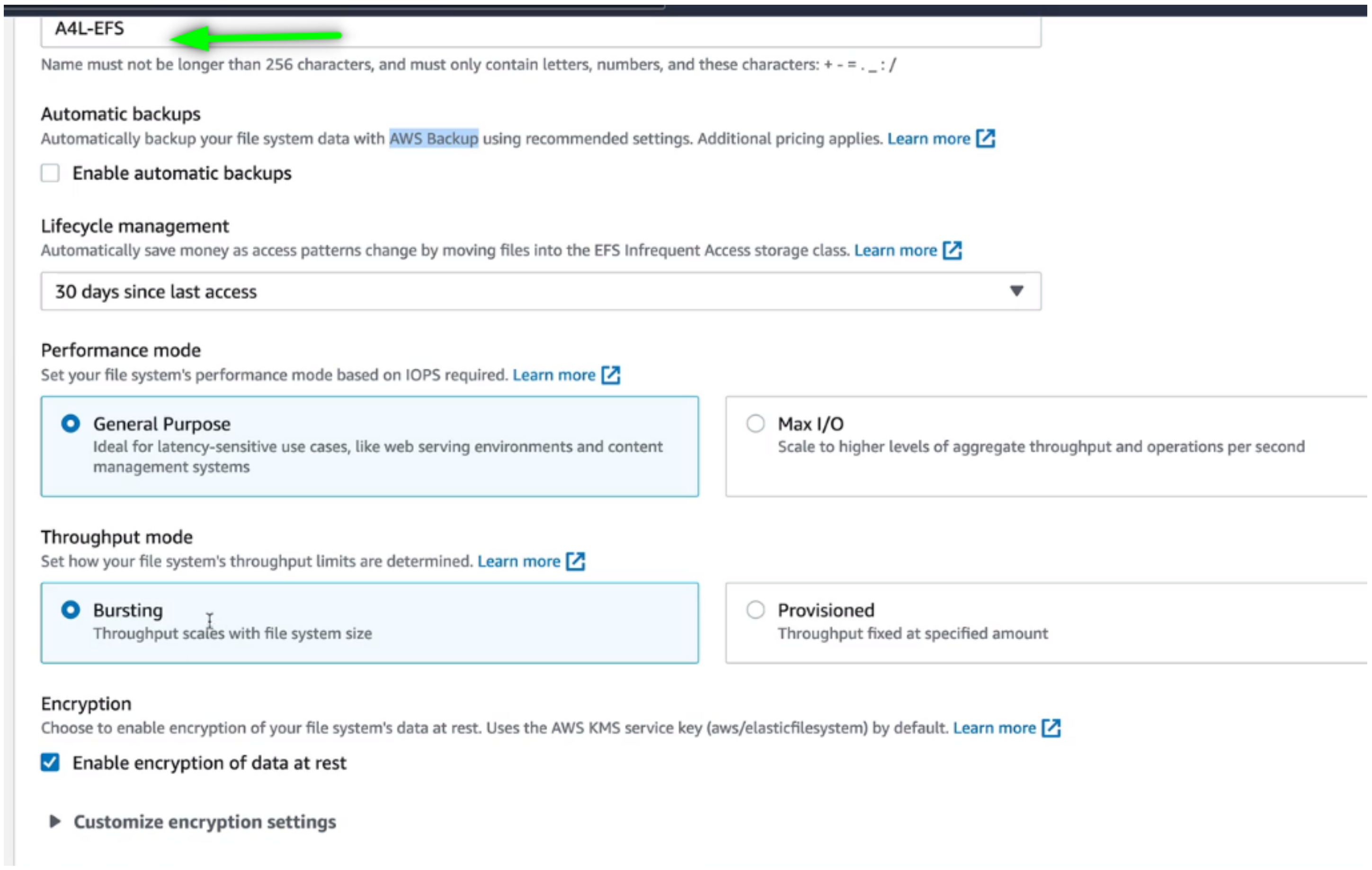
```
sudo mkdir -p /efs/wp-content
sudo nano /etc/fstab

file-system-id:/ /efs/wp-content efs _netdev,tls,iam 0 0

sudo mount /efs/wp-content
df -k
cd /efs/wp-content
sudo touch amazingtestfile.txt

# INSTANCE B

sudo mkdir -p /efs/wp-content
sudo nano /etc/fstab
file-system-id:/ /efs/wp-content efs _netdev,tls,iam 0 0
sudo mount /efs/wp-content
ls -la
```

Instanced created from stack, we will be pairing EFS's to:



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
A4L-EFSInstanceA	i-04f7a419a999b2d94	Running	t2.micro	Initializing	No alarms	us-east-1a
A4L-EFSInstanceB	i-0af2cf84141fbe75f	Running	t2.micro	Initializing	No alarms	us-east-1b

Configure EFS


A4L-EFS

Name must not be longer than 256 characters, and must only contain letters, numbers, and these characters: + - = . _ : /

Automatic backups

Automatically backup your file system data with AWS Backup using recommended settings. Additional pricing applies. Learn more

☐ Enable automatic backups

Lifecycle management

Automatically save money as access patterns change by moving files into the EFS Infrequent Access storage class. Learn more

30 days since last access

Performance mode

Set your file system's performance mode based on IOPS required. Learn more

[✓ General Purpose] Ideal for latency-sensitive use cases, like web serving environments and content management systems

Max I/O Scale to higher levels of aggregate throughput and operations per second

Throughput mode

Set how your file system's throughput limits are determined. Learn more

[✓ Bursting] Throughput scales with file system size

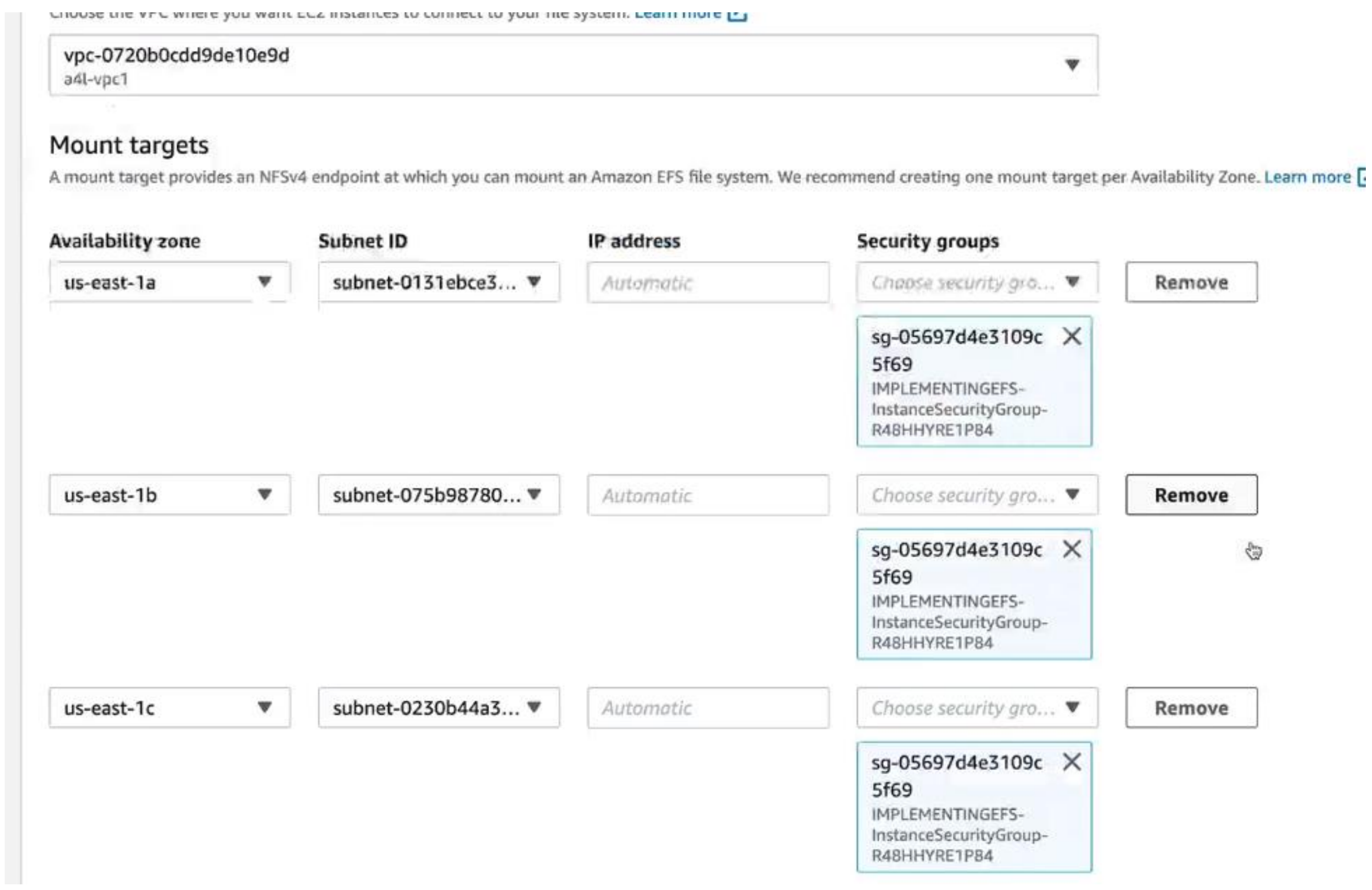
Provisioned Throughput fixed at specified amount

Encryption

Choose to enable encryption of your file system's data at rest. Uses the AWS KMS service key (aws/elasticfilesystem) by default. Learn more

☑ Enable encryption of data at rest

▶ Customize encryption settings



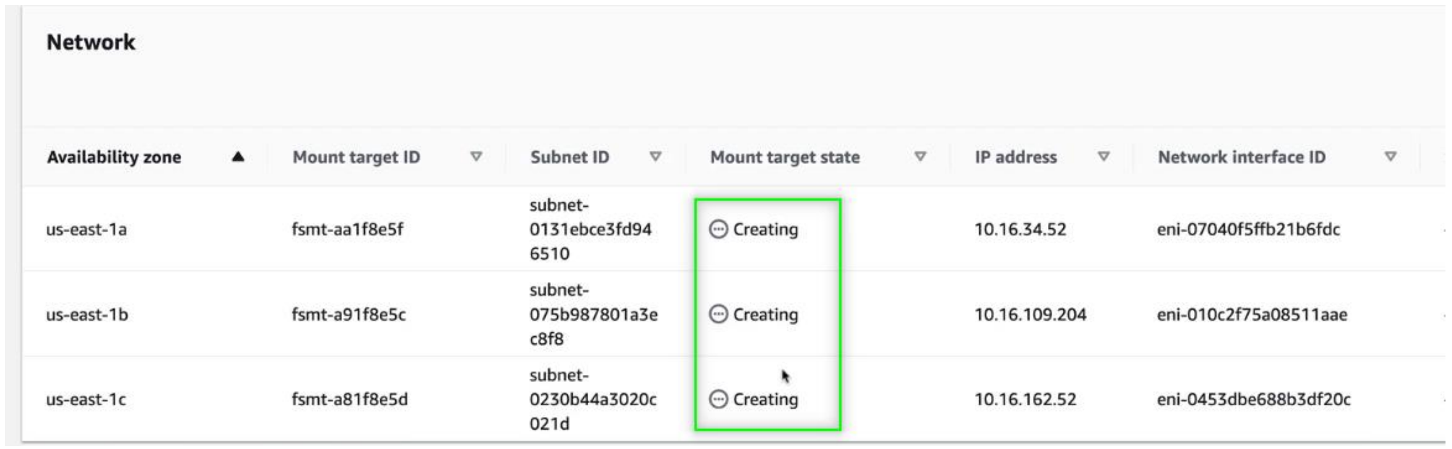
Choose the VPC where you want EC2 instances to connect to your file system. Learn more

vpc-0720b0cdd9de10e9d
a4l-vpc1

Mount targets

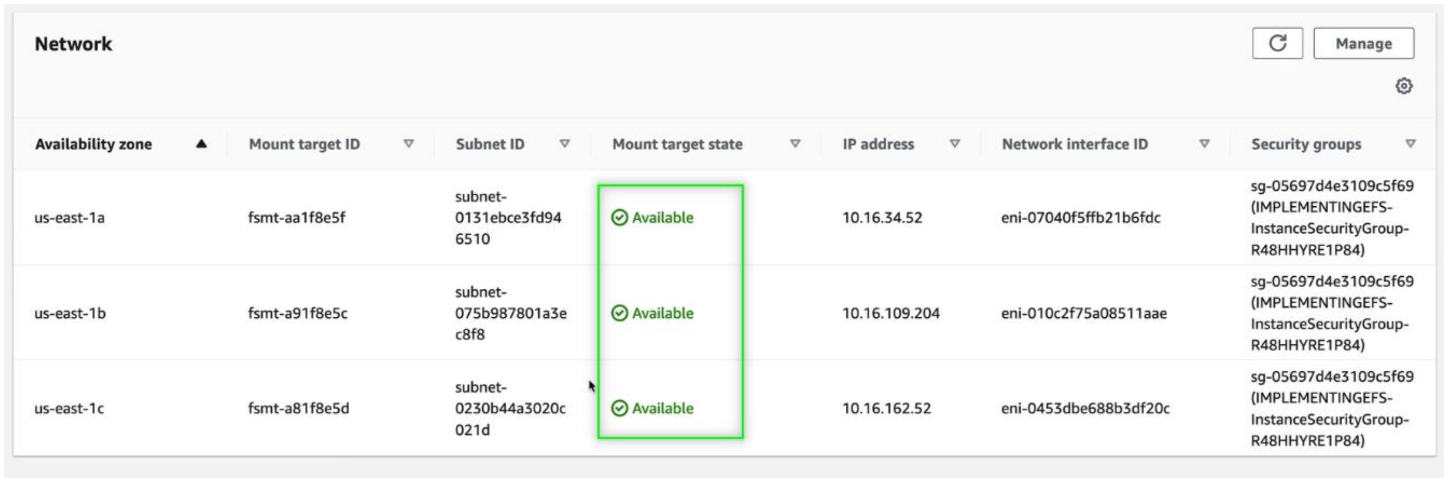
A mount target provides an NFSv4 endpoint at which you can mount an Amazon EFS file system. We recommend creating one mount target per Availability Zone. Learn more

Availability zone	Subnet ID	IP address	Security groups	
us-east-1a	subnet-0131ebce3…	Automatic	sg-05697d4e3109c5f69 IMPLEMENTINGEFS-InstanceSecurityGroup-R48HHYRE1P84	Remove
us-east-1b	subnet-075b98780…	Automatic	sg-05697d4e3109c5f69 IMPLEMENTINGEFS-InstanceSecurityGroup-R48HHYRE1P84	Remove
us-east-1c	subnet-0230b44a3…	Automatic	sg-05697d4e3109c5f69 IMPLEMENTINGEFS-InstanceSecurityGroup-R48HHYRE1P84	Remove



Network

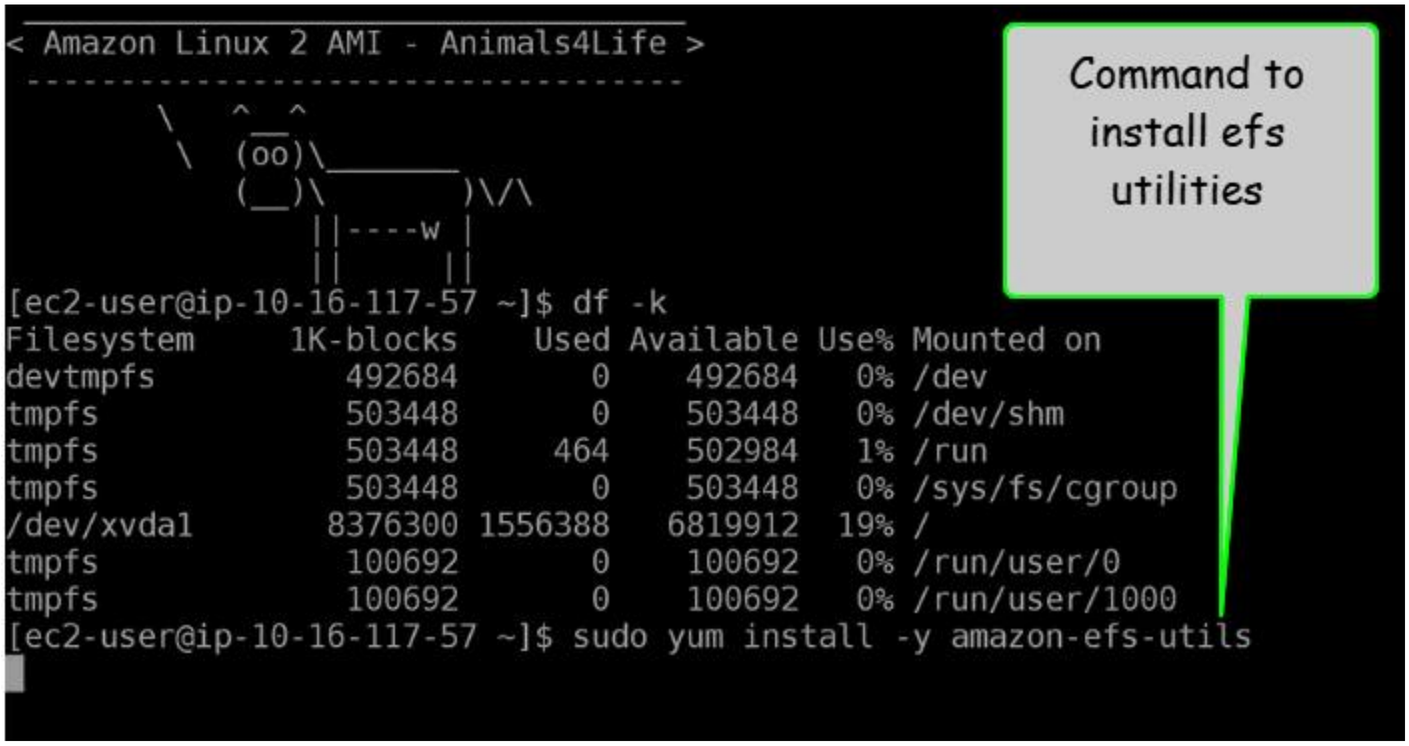
Availability zone	Mount target ID	Subnet ID	Mount target state	IP address	Network interface ID
us-east-1a	fsmt-aa1f8e5f	subnet-0131ebce3fd946510	Creating	10.16.34.52	eni-07040f5ffb21b6fdc
us-east-1b	fsmt-a91f8e5c	subnet-075b987801a3ec8f8	Creating	10.16.109.204	eni-010c2f75a08511aae
us-east-1c	fsmt-a81f8e5d	subnet-0230b44a3020c021d	Creating	10.16.162.52	eni-0453dbe688b3df20c



Network

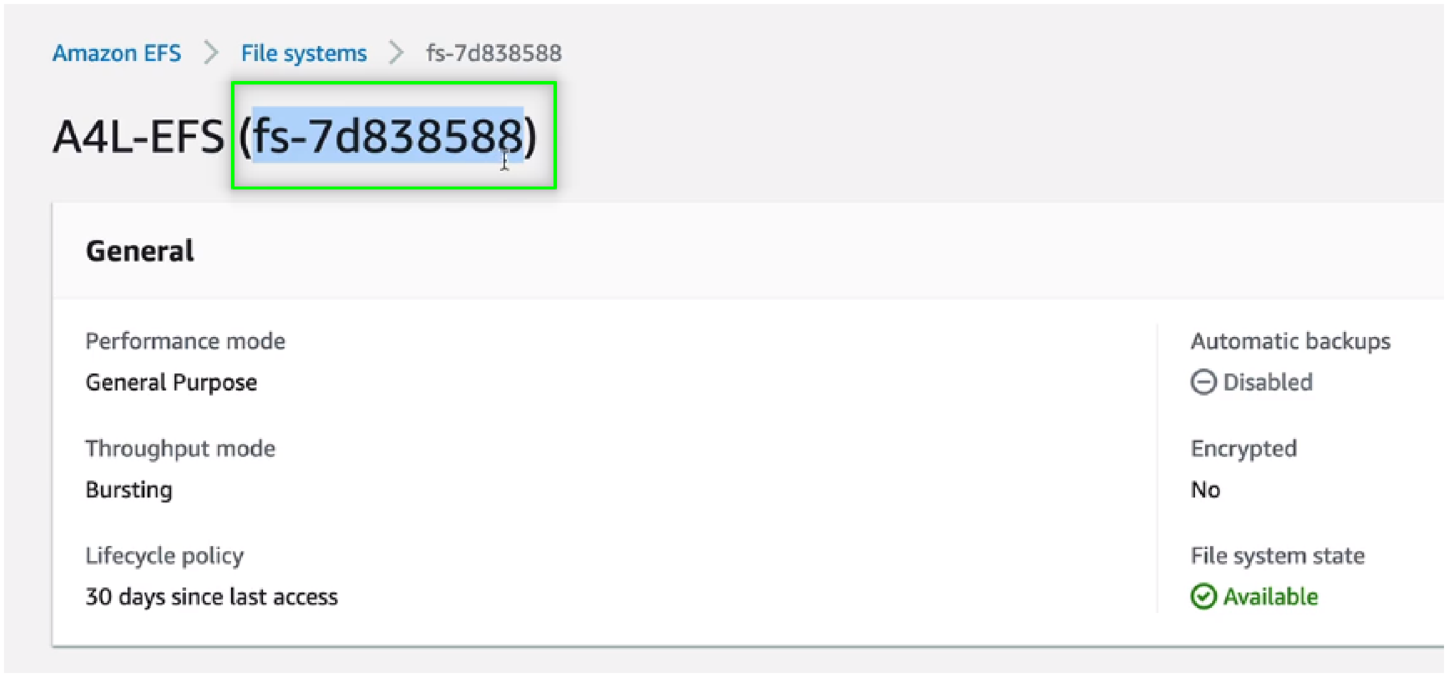
Availability zone	Mount target ID	Subnet ID	Mount target state	IP address	Network interface ID	Security groups
us-east-1a	fsmt-aa1f8e5f	subnet-0131ebce3fd946510	✓ Available	10.16.34.52	eni-07040f5ffb21b6fdc	sg-05697d4e3109c5f69 (IMPLEMENTINGEFS-InstanceSecurityGroup-R48HHYRE1P84)
us-east-1b	fsmt-a91f8e5c	subnet-075b987801a3ec8f8	✓ Available	10.16.109.204	eni-010c2f75a08511aae	sg-05697d4e3109c5f69 (IMPLEMENTINGEFS-InstanceSecurityGroup-R48HHYRE1P84)
us-east-1c	fsmt-a81f8e5d	subnet-0230b44a3020c021d	✓ Available	10.16.162.52	eni-0453dbe688b3df20c	sg-05697d4e3109c5f69 (IMPLEMENTINGEFS-InstanceSecurityGroup-R48HHYRE1P84)

Connect to EC2, can see that EFS is not yet mounted. Install EFS util:



```
 __________________________________
< Amazon Linux 2 AMI - Animals4Life >
 ----------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
[ec2-user@ip-10-16-117-57 ~]$ df -k
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          492684       0    492684   0% /dev
tmpfs             503448       0    503448   0% /dev/shm
tmpfs             503448     464    502984   1% /run
tmpfs             503448       0    503448   0% /sys/fs/cgroup
/dev/xvda1       8376300 1556388   6819912  19% /
tmpfs             100692       0    100692   0% /run/user/0
tmpfs             100692       0    100692   0% /run/user/1000
[ec2-user@ip-10-16-117-57 ~]$ sudo yum install -y amazon-efs-utils
```

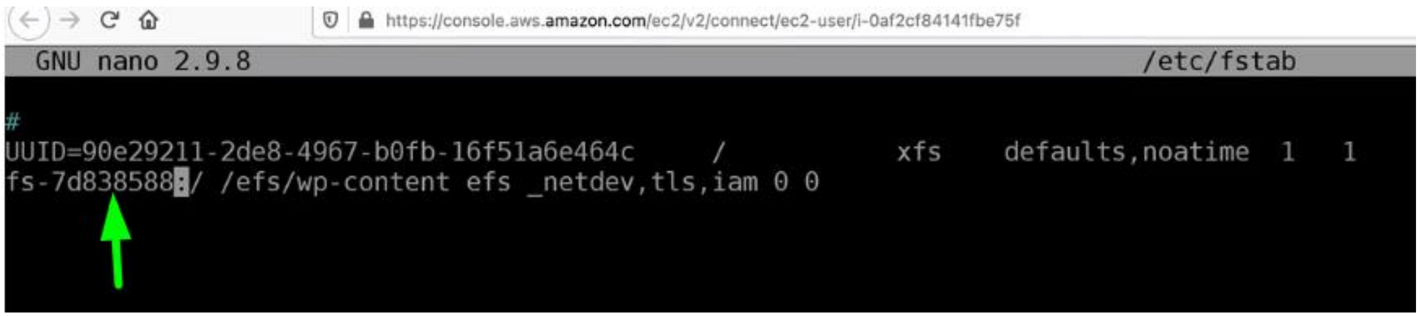
Obtain efs name from console and paste into cli to map:



A4L-EFS (fs-7d838588)

General

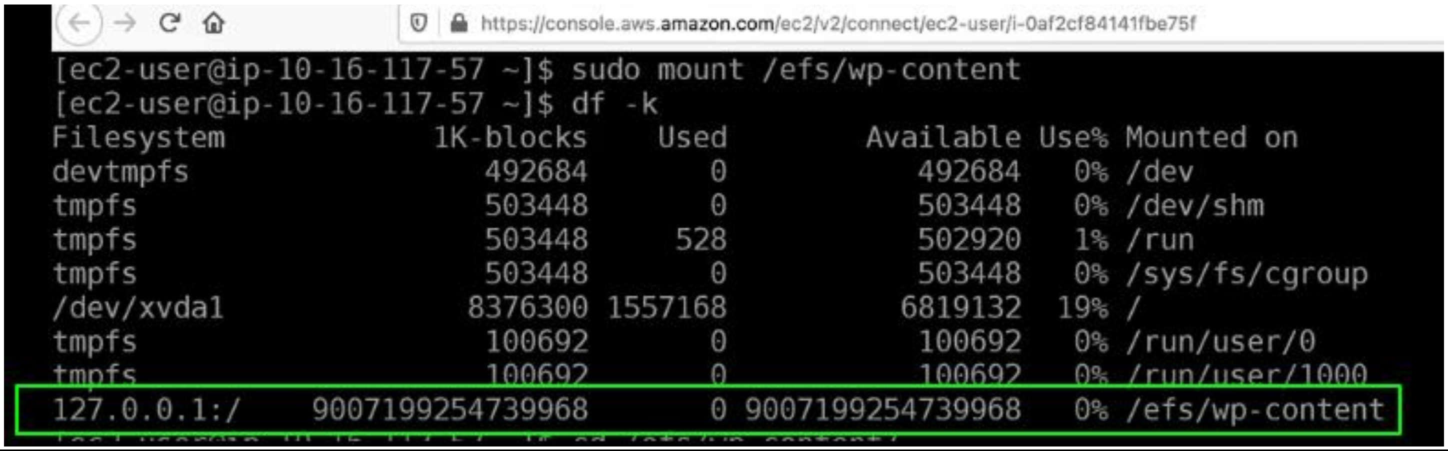
Performance mode	Automatic backups
General Purpose	Disabled
Throughput mode	Encrypted
Bursting	No
Lifecycle policy	File system state
30 days since last access	Available



https://console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0af2cf84141fbe75f

```
  GNU nano 2.9.8                                                     /etc/fstab

#
UUID=90e29211-2de8-4967-b0fb-16f51a6e464c     /           xfs    defaults,noatime  1   1
fs-7d838588:/ /efs/wp-content efs _netdev,tls,iam 0 0
```

Confirm EFS is mounted accordingly:



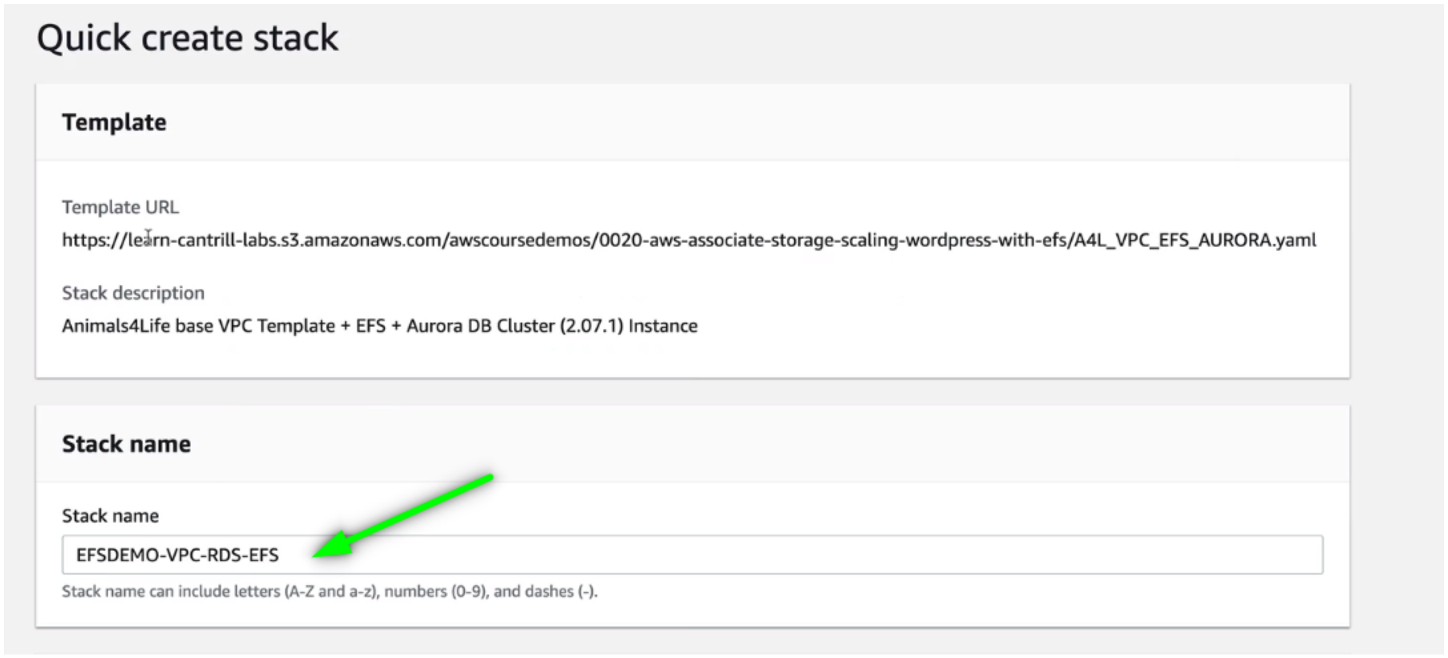
https://console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0af2cf84141fbe75f

```
[ec2-user@ip-10-16-117-57 ~]$ sudo mount /efs/wp-content
[ec2-user@ip-10-16-117-57 ~]$ df -k
Filesystem            1K-blocks    Used        Available Use% Mounted on
devtmpfs                 492684       0           492684   0% /dev
tmpfs                    503448       0           503448   0% /dev/shm
tmpfs                    503448     528           502920   1% /run
tmpfs                    503448       0           503448   0% /sys/fs/cgroup
/dev/xvda1              8376300 1557168          6819132  19% /
tmpfs                    100692       0           100692   0% /run/user/0
tmpfs                    100692       0           100692   0% /run/user/1000
127.0.0.1:/     9007199254739968       0 9007199254739968   0% /efs/wp-content
```

2. Using EFS with Wordpress

In this [DEMO] lesson you will evolve the Animals4life wordpress architecture by moving the wp-content (static media store) from the EC2 instance to an EFS based filesystem. This will allow Wordpress to scale - we can move to an architecture where more than one Wordpress instance runs at once. The experience you gain in this lesson while simple - will be the same experience used in much larger projects.

*use one-click deployments from lesson to create stacks. Wordpress stack + Aurora DB stack:



Quick create stack

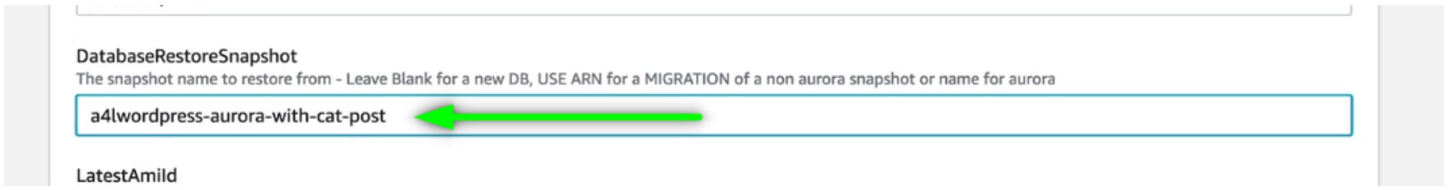
Template

Template URL
https://learn-cantrill-labs.s3.amazonaws.com/awscoursedemos/0020-aws-associate-storage-scaling-wordpress-with-efs/A4L_VPC_EFS_AURORA.yaml

Stack description
Animals4Life base VPC Template + EFS + Aurora DB Cluster (2.07.1) Instance

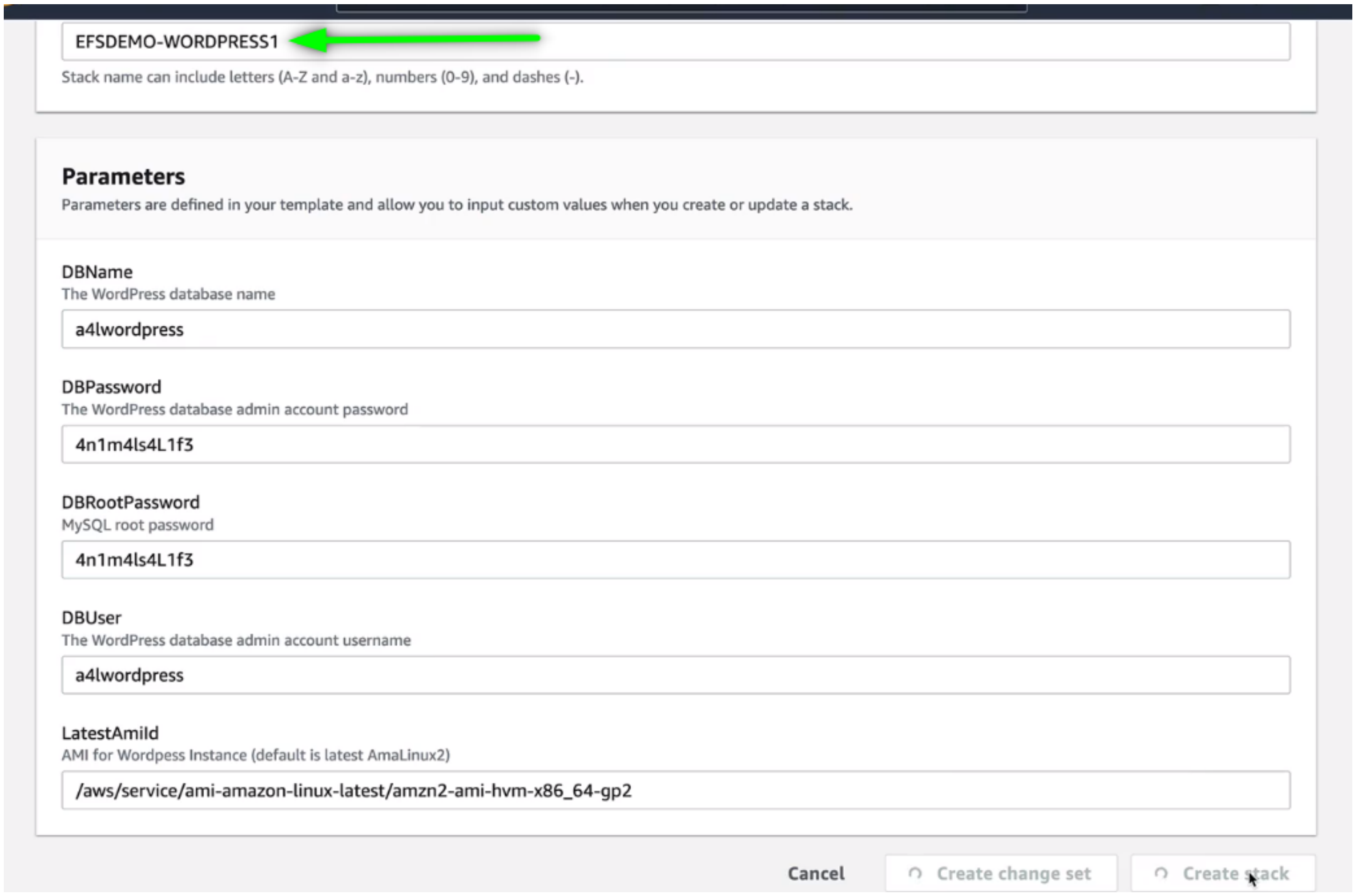
Stack name

Stack name
EFSDEMO-VPC-RDS-EFS
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).



DatabaseRestoreSnapshot
The snapshot name to restore from - Leave Blank for a new DB, USE ARN for a MIGRATION of a non aurora snapshot or name for aurora
a4lwordpress-aurora-with-cat-post

LatestAmiId



EFSDEMO-WORDPRESS1

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

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

DBName
The WordPress database name
a4lwordpress

DBPassword
The WordPress database admin account password
4n1m4ls4L1f3

DBRootPassword
MySQL root password
4n1m4ls4L1f3

DBUser
The WordPress database admin account username
a4lwordpress

LatestAmiId
AMI for Wordpess Instance (default is latest AmaLinux2)
/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2

Adrian explains the Stacks CFN files in the lesson, YAML file with EFS below:

```yaml
Description:  Animals4Life base VPC Template + EFS + Aurora DB Cluster (2.07.1) Instance
Parameters:
  LatestAmiId:
    Description: AMI for Wordpess Instance (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'
  DBName:
    AllowedPattern: '[a-zA-Z][a-zA-Z0-9]*'
    ConstraintDescription: must begin with a letter and contain only alphanumeric characters.
    Default: 'a4lwordpress'
    Description: The WordPress database name
    MaxLength: '64'
    MinLength: '1'
    Type: String
  DBPassword:
    ConstraintDescription: must contain only alphanumeric characters.
    Description: The WordPress database admin account password
    MaxLength: '41'
    MinLength: '8'
    Type: String
    Default: '4n1m4ls4L1f3'
  DBRootPassword:
    ConstraintDescription: must contain only alphanumeric characters.
    Description: MySQL root password
    MaxLength: '41'
    MinLength: '8'
    Type: String
    Default: '4n1m4ls4L1f3'
  DBUser:
    ConstraintDescription: must begin with a letter and contain only alphanumeric characters.
    Description: The WordPress database admin account username
    Default: 'a4lwordpress'
    MaxLength: '16'
    MinLength: '1'
    Type: String
  DatabaseRestoreSnapshot:
    Description: The snapshot name to restore from - Leave Blank for a new DB, USE ARN for a 
MIGRATION of a non aurora snapshot or name for aurora
    Type: String
Conditions:
  NoSnapshot:
    !Equals ['', !Ref DatabaseRestoreSnapshot]
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, "CustomReso
urcePhysicalID")
      Runtime: python2.7
      Role: !GetAtt IPv6WorkaroundRole.Arn
      Timeout: 30
  RDSSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId: !Ref VPC
      GroupDescription: "Ingress control for RDS instance"
      SecurityGroupIngress:
        - Description: 'Allow MySQL IPv4 IN'
          IpProtocol: tcp
          FromPort: '3306'
          ToPort: '3306'
          SourceSecurityGroupId: !Ref InstanceSecurityGroup
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: A4L Aurora subnet group
      SubnetIds:
        - !Ref SubnetDBA
        - !Ref SubnetDBB
        - !Ref SubnetDBC
  DBCluster:
    Type: "AWS::RDS::DBCluster"
    DeletionPolicy: Delete
    Properties:
      DBSubnetGroupName: !Ref DBSubnetGroup
      DatabaseName: !If [NoSnapshot,  !Ref DBName, !Ref 'AWS::NoValue' ]
      Engine: 'aurora-mysql'
      EngineMode: 'provisioned'
      EngineVersion: "5.7.mysql_aurora.2.07.1"
      MasterUserPassword: !If [NoSnapshot, !Ref DBPassword, !Ref 'AWS::NoValue' ]
      MasterUsername: !If [NoSnapshot, !Ref DBUser, !Ref 'AWS::NoValue' ]
      SnapshotIdentifier: !If [ NoSnapshot, !Ref 'AWS::NoValue', !Ref DatabaseRestoreSnapshot
]
```

```
      Tags:
        -
          Key: Name
          Value: "A4L-Aurora-Cluster"
      VpcSecurityGroupIds:
        - !Ref RDSSecurityGroup
  AuroraInstance1:
    Type: AWS::RDS::DBInstance
    DeletionPolicy: Delete
    Properties:
      AllowMajorVersionUpgrade: false
      AutoMinorVersionUpgrade: true
      DBClusterIdentifier: !Ref DBCluster
      DBInstanceClass: db.t3.small
      DBSubnetGroupName: !Ref DBSubnetGroup
      Engine: 'aurora-mysql'
      Tags:
        - Key: Name
          Value: !Join [ '', [ 'WordPress / ', !Ref 'AWS::StackName' ] ]
  AuroraInstance2:
    Type: AWS::RDS::DBInstance
    DeletionPolicy: Delete
    Properties:
      AllowMajorVersionUpgrade: false
      AutoMinorVersionUpgrade: true
      DBClusterIdentifier: !Ref DBCluster
      DBInstanceClass: db.t3.small
      DBSubnetGroupName: !Ref DBSubnetGroup
      Engine: 'aurora-mysql'
      Tags:
        - Key: Name
          Value: !Join [ '', [ 'WordPress / ', !Ref 'AWS::StackName' ] ]
  InstanceSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      VpcId: !Ref VPC
      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 HTTP IPv4 IN'
          IpProtocol: tcp
          FromPort: '80'
          ToPort: '80'
          CidrIp: '0.0.0.0/0'
        - Description: 'Allow SSH IPv6 IN'
          IpProtocol: tcp
          FromPort: '22'
          ToPort: '22'
          CidrIpv6: ::/0
  InstanceSecurityGroupSelfReferenceRule:
    Type: "AWS::EC2::SecurityGroupIngress"
    Properties:
      GroupId: !Ref InstanceSecurityGroup
      IpProtocol: 'tcp'
      FromPort: '0'
      ToPort: '65535'
      SourceSecurityGroupId: !Ref InstanceSecurityGroup
  WordpressRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
              - ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      ManagedPolicyArns:
        - "arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy"
        - "arn:aws:iam::aws:policy/AmazonSSMFullAccess"
        - "arn:aws:iam::aws:policy/AmazonElasticFileSystemClientFullAccess"
  WordpressInstanceProfile:
    Type: 'AWS::IAM::InstanceProfile'
    Properties:
      Path: /
      Roles:
        - !Ref WordpressRole
  CWAgentConfig:
    Type: AWS::SSM::Parameter
    Properties:
      Type: 'String'
      Value: |
        {
          "agent": {
            "metrics_collection_interval": 60,
            "run_as_user": "root"
          },
          "logs": {
            "logs_collected": {
              "files": {
                "collect_list": [
                  {
                    "file_path": "/var/log/secure",
                    "log_group_name": "/var/log/secure",
                    "log_stream_name": "{instance_id}"
                  },
                  {
                    "file_path": "/var/log/httpd/access_log",
                    "log_group_name": "/var/log/httpd/access_log",
                    "log_stream_name": "{instance_id}"
                  },
                  {
                    "file_path": "/var/log/httpd/error_log",
                    "log_group_name": "/var/log/httpd/error_log",
                    "log_stream_name": "{instance_id}"
                  }
                ]
              }
            }
          },
          "metrics": {
            "append_dimensions": {
              "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
              "ImageId": "${aws:ImageId}",
              "InstanceId": "${aws:InstanceId}",
              "InstanceType": "${aws:InstanceType}"
            },
            "metrics_collected": {
              "collectd": {
                "metrics_aggregation_interval": 60
              },
              "cpu": {
                "measurement": [
                  "cpu_usage_idle",
                  "cpu_usage_iowait",
                  "cpu_usage_user",
                  "cpu_usage_system"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                  "*"
                ],
                "totalcpu": false
              },
              "disk": {
                "measurement": [
                  "used_percent",
                  "inodes_free"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                  "*"
                ]
              },
              "diskio": {
                "measurement": [
                  "io_time",
                  "write_bytes",
                  "read_bytes",
                  "writes",
                  "reads"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                  "*"
                ]
              },
              "mem": {
                "measurement": [
                  "mem_used_percent"
                ],
                "metrics_collection_interval": 60
              },
              "netstat": {
                "measurement": [
                  "tcp_established",
                  "tcp_time_wait"
                ],
                "metrics_collection_interval": 60
              },
              "statsd": {
                "metrics_aggregation_interval": 60,
                "metrics_collection_interval": 10,
                "service_address": ":8125"
              },
              "swap": {
                "measurement": [
                  "swap_used_percent"
                ],
                "metrics_collection_interval": 60
              }
            }
          }
        }
  ElasticFileSystem:
    Type: AWS::EFS::FileSystem
    Properties:
      FileSystemTags:
        - Key: Name
          Value: !Join [ '', [ 'A4L-EFS-WordPress / ', !Ref 'AWS::StackName' ] ]
  ElasticFileSystemMountTarget0:
    Type: AWS::EFS::MountTarget
    Properties:
      FileSystemId: !Ref "ElasticFileSystem"
      SecurityGroups:
        - !Ref "InstanceSecurityGroup"
      SubnetId: !Ref "SubnetAPPA"
  ElasticFileSystemMountTarget1:
    Type: AWS::EFS::MountTarget
    Properties:
      FileSystemId: !Ref "ElasticFileSystem"
      SecurityGroups:
        - !Ref "InstanceSecurityGroup"
      SubnetId: !Ref "SubnetAPPB"
  ElasticFileSystemMountTarget2:
    Type: AWS::EFS::MountTarget
    Properties:
      FileSystemId: !Ref "ElasticFileSystem"
      SecurityGroups:
        - !Ref "InstanceSecurityGroup"
      SubnetId: !Ref "SubnetAPPC"
Outputs:
  instanceprofile:
    Description: "Default Instance Profile"
    Value: !Ref WordpressInstanceProfile
    Export:
      Name: "A4L-WordpressInstanceProfile"
  subnetweba:
    Description: "Web A (Public) Subnet"
    Value: !Ref SubnetWEBA
    Export:
      Name: "A4L-SubnetWEBA"
  InstanceSecurityGroup:
    Description: "A4L Default Instance SG"
    Value: !Ref InstanceSecurityGroup
    Export:
      Name: "A4L-InstanceSecurityGroup"
  dbendpoint:
    Description: "RDS Endpoint Address"
    Value: !GetAtt DBCluster.Endpoint.Address
    Export:
      Name: "A4L-DBENDPOINT"
  ElasticFileSystem:
    Value: !Ref ElasticFileSystem
    Export:
      Name: A4L-EFS
  cwagentconfig:
    Description: "CWAgent Config Parameter"
    Value: !Ref "CWAgentConfig"
    Export:
      Name: "A4L-CWAGENT-CONFIG"
```