



NewgenONE OmniDocs RMS

Configuration and Deployment Guide for AWS

Version: 4.0 SP1

[Newgen Software Technologies Ltd.](#)

This document contains propriety information of NSTL. No part of this document may be reproduced, stored, copied, or transmitted in any form or by any means of electronic, mechanical, photocopying, or otherwise, without the consent of NSTL.

Table of contents

1	Preface	3
1.1	Revision history	3
1.2	Intended audience	3
1.3	Documentation feedback	3
2	Configuring AWS Kubernetes Cluster	4
2.1	Creating an IAM user	4
2.2	Creating a VPC	6
2.3	Creating a subnets	8
2.4	Creating an internet gateway	11
2.5	Creating a route table	12
2.6	Creating an IAM role	13
2.7	Creating a security group	14
2.8	Creating an EKS cluster	14
2.9	Creating a key pair	16
2.10	Provisioning Kubernetes worker nodes using cloud formation	17
2.11	Adding inbound rule in EC2 instance	20
2.12	Enabling worker node to join EKS cluster	22
2.13	Running Kubectl from a local machine	24
2.14	Creating EFS	24
2.15	Mounting EFS to worker nodes	27
2.16	Configuring Kubernetes dashboard	27
2.17	Configuring AWS load balancer controller	27
2.18	Configuring AWS Elastic Redis Cache	30
2.19	Registering domain using route-53	35
2.20	Generating SSL certificate against the registered domain	37
2.21	Cluster AutoScaler	41
2.21.1	Node Group IAM Policy	41
2.21.2	Updating auto scaling group	42
2.21.3	Deploying cluster autoscaler	44
2.21.4	Viewing cluster autoscaler logs	45
2.22	Setting up CloudWatch container insights	45
3	Deploying OmniDocs and RMS containers	46
3.1	Prerequisites	46
3.2	Deliverables	46
3.2.1	Docker Images	47
3.2.2	Configuration files	47
3.2.3	YAML files	48
3.3	Changes in Product's YAML files	49
3.4	Changes in AWS load balancer controller YAML files	54
3.5	Changes in configuration files	56
3.5.1	Prerequisites	56
3.5.2	OmniDocs+RMS Web Changes	56
3.5.3	Wrapper changes	60

3.5.4	AlarmMailer changes.....	61
3.5.5	LDAP changes	62
3.5.6	SSO changes.....	69
3.5.7	Scheduler changes	69
3.5.8	ThumbnailManager changes	70
3.5.9	TEM changes.....	71
3.5.10	EasySearch changes.....	72
3.5.11	WOPI changes.....	78
3.5.12	OmniScanWeb changes	80
3.5.13	RMS SharePoint Adapter changes	82
3.5.14	Messaging Service changes	84
3.6	Deploying containers	86
3.7	Creating cabinet and data source	90
3.7.1	Getting started with OSA.....	90
3.7.2	Registering JTS server	92
3.7.3	Connecting OSA to JTS Server.....	94
3.7.4	Creating a cabinet.....	96
3.7.5	Associating a cabinet	101
3.7.6	Creating a data source.....	105
3.7.7	Registering a cabinet in OmniDocs	118
3.7.8	Registering a cabinet in RMS	120
3.7.9	Creating site and volume.....	121
3.8	EasySearch Post-Deployment Changes	126
3.9	OmniScanWeb: Registering a cabinet	127
3.10	Creating a Secret manager policy and secrets	129
4	Configuring AWS CodePipeline for container deployment on EKS	132
4.1	Overview	132
4.2	Architecture of CICD pipeline.....	132
4.3	Configuring AWS Elastic container registry	133
4.4	Push and Pull docker images to or from AWS ECR.....	135
4.5	Configuring AWS CodePipeline	138
4.5.1	Creating an IAM Policy and IAM Role	138
4.5.2	Creating AWS CodeCommit repository	141
4.5.3	Creating AWS CodeBuild project	143
4.5.4	Creating AWS CodePipeline.....	148
4.5.4.1	Configuring AWS CodePipeline for Dev stage	148
4.5.4.2	Configuring notification.....	157
4.5.4.3	Configuring AWS CodePipeline for UAT stage.....	162
4.5.4.4	Configuring AWS CodePipeline for production stage.....	174
Appendix		198

1 Preface

This guide describes the deployment and configuration of NewgenONE OmniDocs Record Management System (RMS) 4.0 SP1. It includes Docker images and their required configuration files on the AWS Elastic Kubernetes Service (EKS).

1.1 Revision history

Revision Date	Description
April 2024	Initial publication

1.2 Intended audience

This guide is intended for system administrators, developers, and all other users who are looking for information on the deployment of NewgenONE OmniDocs and RMS containers on AWS Kubernetes Services. The reader must have administrative rights on the machine.

1.3 Documentation feedback

To provide feedback or any improvement suggestions on technical documentation, write an email to docs.feedback@newgensoft.com.

To help capture your feedback effectively, share the following information in your email.

- Document Name:
- Version:
- Chapter, Topic, or Section:
- Feedback or Suggestions:

2 Configuring AWS Kubernetes Cluster

This chapter describes the configuration of AWS Kubernetes Service. For procedural details, refer to the below sections.

2.1 Creating an IAM user

For creating an IAM user, configure the AWS Kubernetes Cluster instead of using the Amazon Management Console root user.

Perform the below steps to create an IAM user:

1. Sign in to the AWS Management Console using the root user and open the IAM console in Services.
2. Select **Users** and then select **Add User** in the navigation panel.
3. Enter the **username** for the new user. It's a signed-in name for AWS.
4. Select the user's access type. You can select programmatic access or access to the AWS Management Console, and both.
 - Select **Programmatic access** to access the API, AWS CLI, or Tools for Windows PowerShell. This creates an access key for each new user. You can view or download the access keys once you reach the final page.
 - Select **AWS Management Console access** to access the AWS Management Console. This creates a password for each new user.
5. For the Console password, select any of the following:
 - **Auto-generated password:** It provides a randomly generated password to each user that meets the account password policy in effect (if any). Once complete, you can view or download the passwords.
 - **Custom password:** The password you entered is assigned to each user.

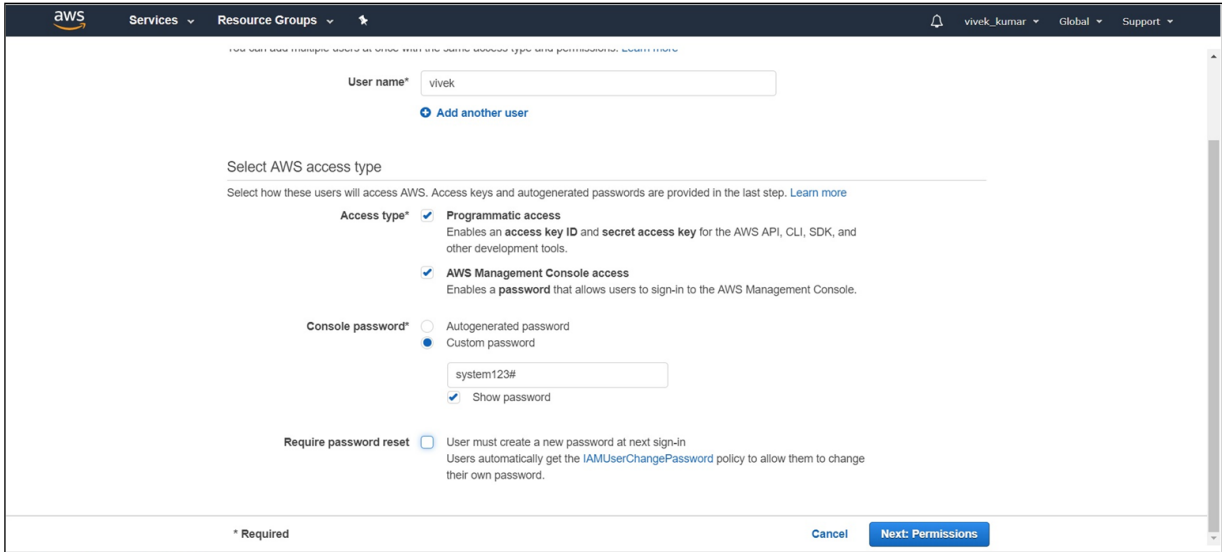


Figure 2.1

6. Click **Next: Permissions**. The Set Permissions screen appears.
7. Select the **Attach existing policies directly** and select the **Administrator Access** policy.

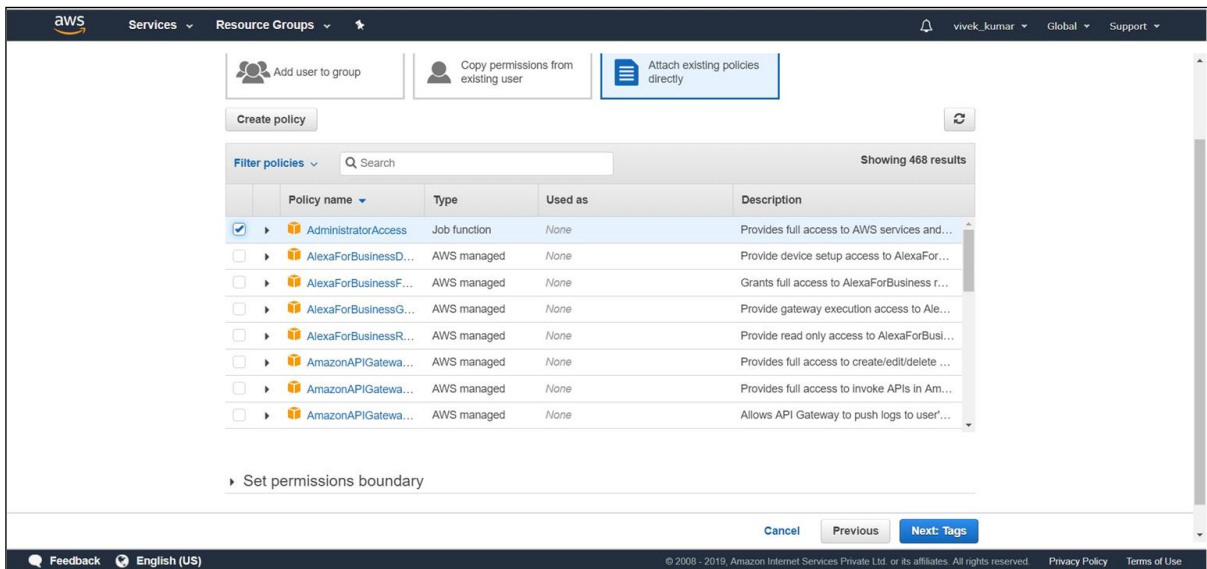


Figure 2.2

8. Click **Next**. The user is created successfully.

2.2 Creating a VPC

Perform the below steps to create VPC (Virtual Private Cloud):

1. Sign in to the AWS Management Console using the root user and open the VPC in Services.
2. Select **Your VPC** and click Create VPC in the navigation pane.
3. Select **Resources** to create **VPC Only** in the Create VPC.
4. Specify the user-defined VPC name in the Name Tag field.
5. Specify the **IPv4 CIDR block** as **10.0.0.0/16** and click **Create**.

VPC > Your VPCs > Create VPC

Create VPC Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

VPC settings

Resources to create Info
Create only the VPC resource or create VPC, subnets, etc.

VPC only VPC, subnets, etc.

Name tag - optional
Creates a tag with a key of 'Name' and a value that you specify.

amit-vpc

IPv4 CIDR block Info

IPv4 CIDR manual input IPAM-allocated IPv4 CIDR block

IPv4 CIDR

10.0.0.0/16

IPv6 CIDR block Info

No IPv6 CIDR block IPAM-allocated IPv6 CIDR block Amazon-provided IPv6 CIDR block IPv6 CIDR owned by me

Tenancy Info

Default

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key: Name Value - optional: amit-vpc

Remove

Add new tag

You can add 49 more tags.

Cancel Create VPC

Figure 2.3

6. In the VPC, go to **Action** and click **Edit DNS hostnames**.

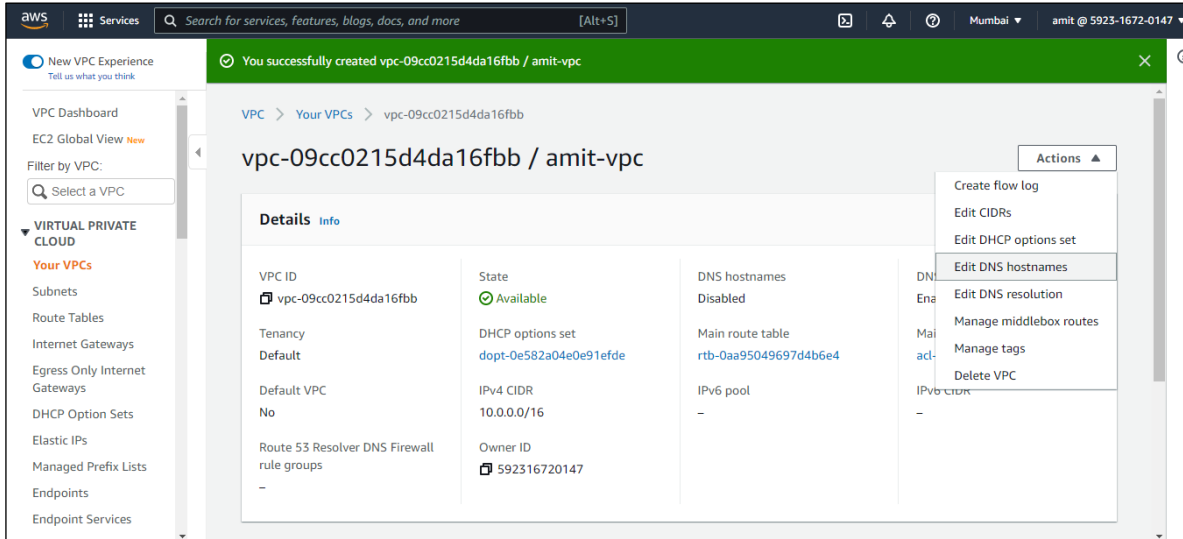


Figure 2.4

7. Select the Enable checkbox to enable DNS Hostnames.

8. Click **Save Changes**.



Figure 2.5

2.3 Creating a subnets

It contains information on creating three subnets for the availability zone in the Mumbai region for High Availability.

Perform the below steps to create subnets:

1. In VPC Dashboard, go to the **Subnets**.
2. Go to the **Create Subnet** and select the created VPC using the dropdown.
3. In Subnet settings, specify the user-defined subnet name in the **Subnet name** field.
4. Select the **ap-south-1a** in the Availability Zone.
5. Specify the **10.0.1.0/24** in the IPv4 CIDR block.
6. Create two more subnets for other availability zones: **ap-south-1b** and **ap-south-1c** by clicking **Add new subnet**.
7. Click **Create subnet**.

VPC > Subnets > Create subnet

Create subnet Info

VPC

VPC ID
Create subnets in this VPC.

vpc-09cc0215d4da16fbb (amit-vpc) ▼

Associated VPC CIDRs

IPv4 CIDRs
10.0.0.0/16

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 3

Subnet name
Create a tag with a key of 'Name' and a value that you specify.

amit-subnet-1A
The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

Asia Pacific (Mumbai) / ap-south-1a ▼

IPv4 CIDR block Info
10.0.1.0/24 ✕

▼ Tags - optional

Key	Value - optional	
Q Name ✕	Q amit-subnet-1A ✕	Remove

Add new tag
You can add 49 more tags.

Remove

Subnet 2 of 3

Subnet name
Create a tag with a key of 'Name' and a value that you specify.

amit-subnet-1B
The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

Asia Pacific (Mumbai) / ap-south-1b ▼

IPv4 CIDR block Info
10.0.2.0/24 ✕

▼ Tags - optional

Key	Value - optional	
Q Name ✕	Q amit-subnet-1B ✕	Remove

Add new tag
You can add 49 more tags.

Remove

Subnet 3 of 3

Subnet name
Create a tag with a key of 'Name' and a value that you specify.

amit-subnet-1C
The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

Asia Pacific (Mumbai) / ap-south-1c ▼

IPv4 CIDR block Info
10.0.3.0/24 ✕

▼ Tags - optional

Key	Value - optional	
Q Name ✕	Q amit-subnet-1C ✕	Remove

Add new tag
You can add 49 more tags.

Remove

Add new subnet

Cancel Create subnet

Figure 2.6

8. After creating subnets, Edit all subnet settings.
9. Select **one subnet**.
10. Go to the **Action** and Edit **subnet settings**.

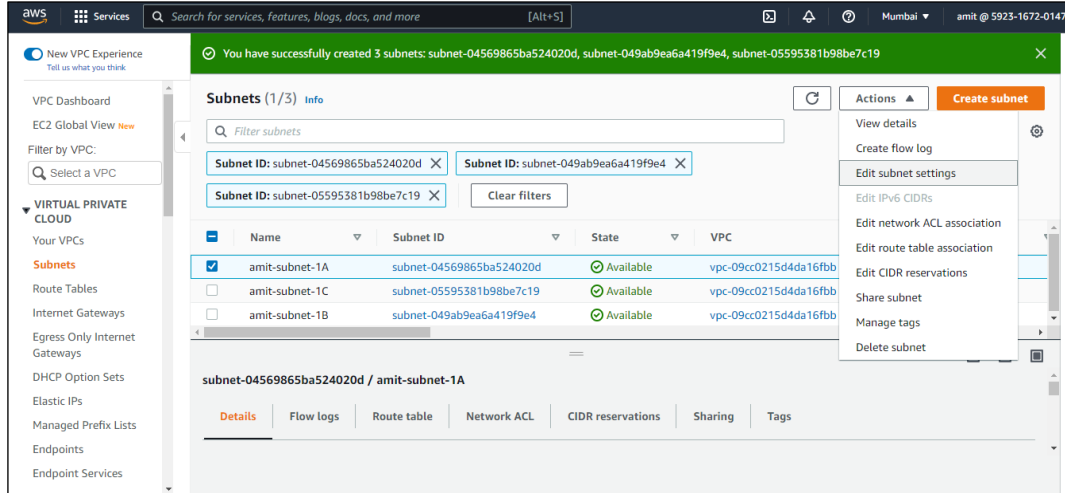


Figure 2.7

11. In the Auto-assign IP settings, **Enable auto-assign public IPv4 address**.
12. Click **Save** to save subnet settings.

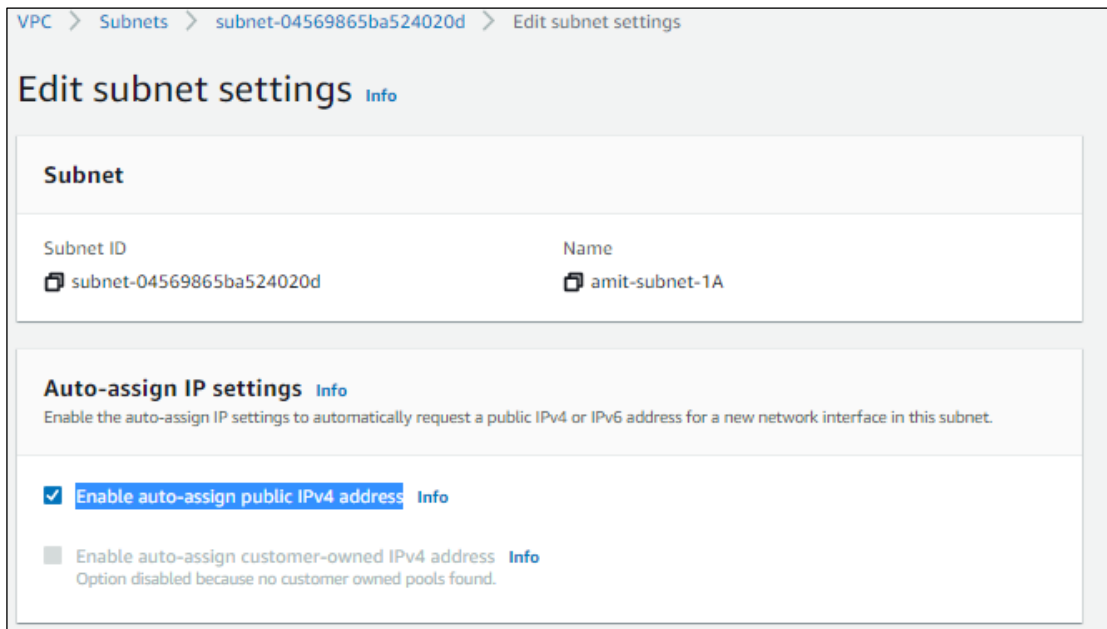


Figure 2.8

2.4 Creating an internet gateway

This section explains how to create an Internet Gateway for the **public Route Table**. The creation of the Route Table is described in the section [create a route table](#).

NOTE:

To use a private Route Table, you must create **Nat Gateway** which cost up to \$40.

Perform the below steps to create an Internet Gateway:

1. In VPC Dashboard, go to the **Internet Gateways** and click **Create internet gateway**.
2. Specify the user-defined name in the **Name tag** field and click **Create**.

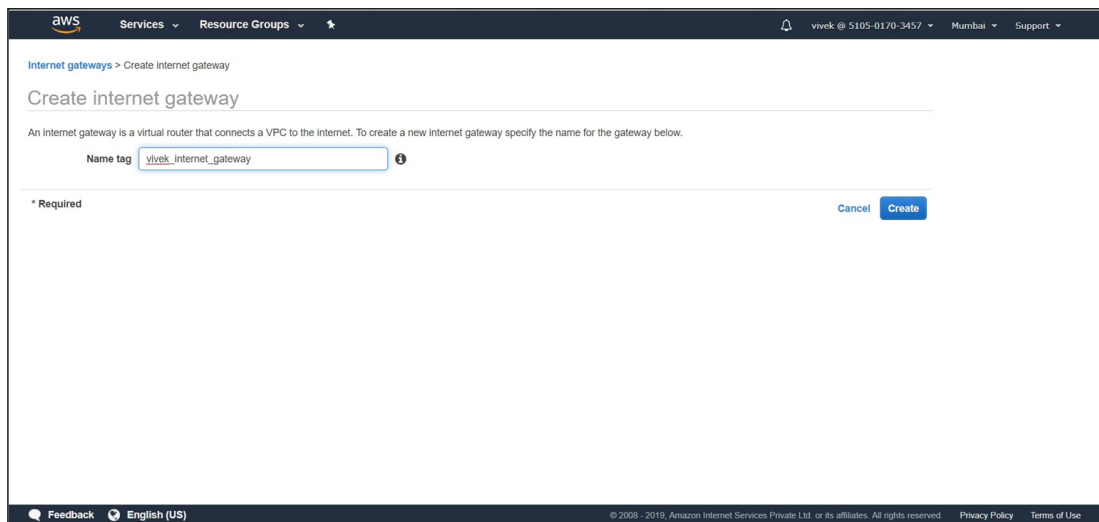


Figure 2.9

3. Select the created internet gateway.
4. Select the **Attach to VPC** option in the **Actions** menu.
5. Select the created VPC and click **Attach**.

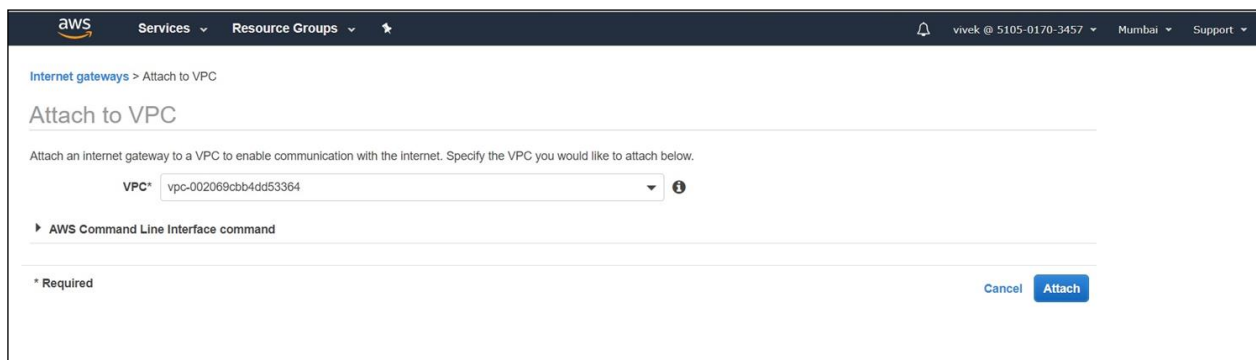


Figure 2.10

2.5 Creating a route table

Perform the below steps to create the Route table:

1. In VPC Dashboard, go to the **Route Tables** and click **Create route tables**.
2. Specify the user-defined route table name in the **Name tag** field.
3. Select the created VPC and click **Create**.

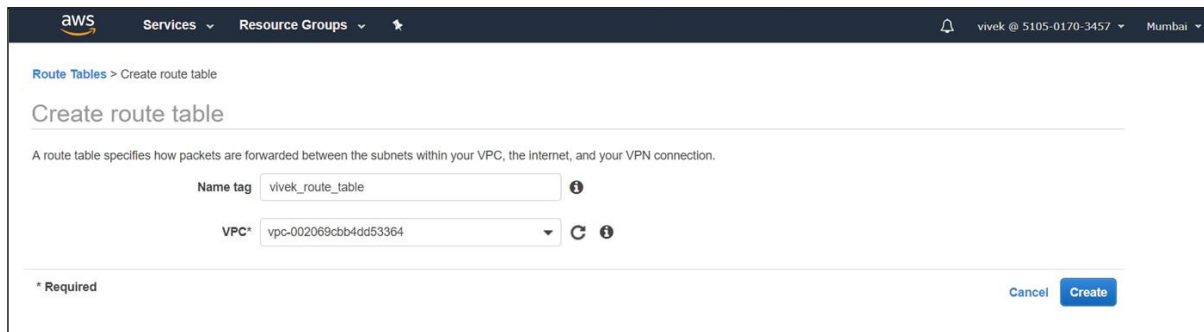


Figure 2.11

4. Select the created route table.
5. Go to the **Routes** tab and click **Edit Route**.
6. To provide internet access to a created route table, add a **new route** and specify **0.0.0.0/0** in the **Destination** field.
7. Select the created Internet gateway in the **Target** field.
8. Click **Save routes**.
9. Select the created route table.
10. Go to **Subnet Associations** and click **Edit subnet associations**.
11. Select the created subnets for all availability zones for the Mumbai region.

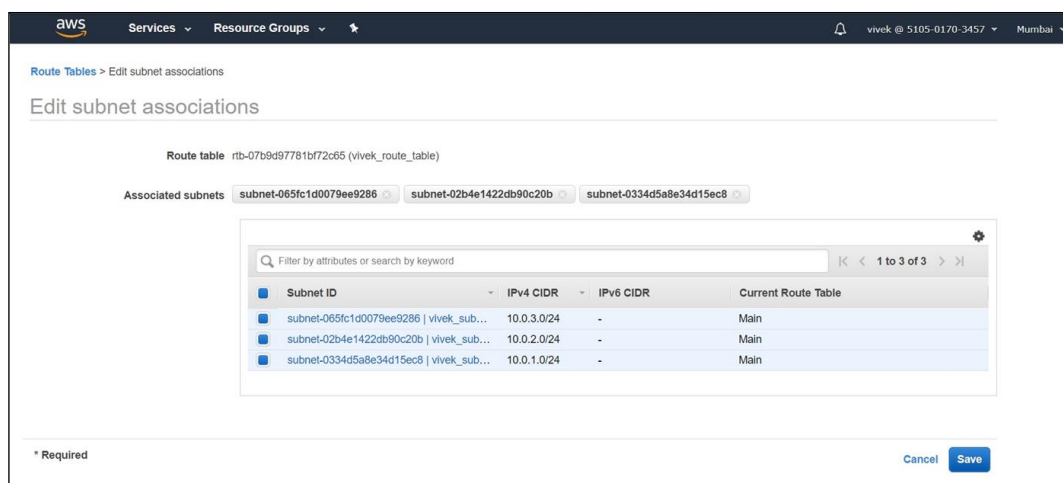


Figure 2.12

2.6 Creating an IAM role

Before creating a Kubernetes Cluster, you must create an IAM role that Kubernetes can assume to create AWS resources.

For example, when a load balancer is created, Kubernetes assumes the role to create an Elastic Load Balancing load balancer in your account. You can create this one time only and can be used for multiple EKS clusters.

Perform the below steps to create an IAM Role:

1. Go to IAM Dashboard.
2. Go to the **Roles** and click **Create role**.
3. Select **EKS** from the list of services.
4. Select **EKS Cluster** to **Allows access to other AWS service resources that are required to operate clusters managed by EKS** for your use case.
5. Click **Next: Permissions**.

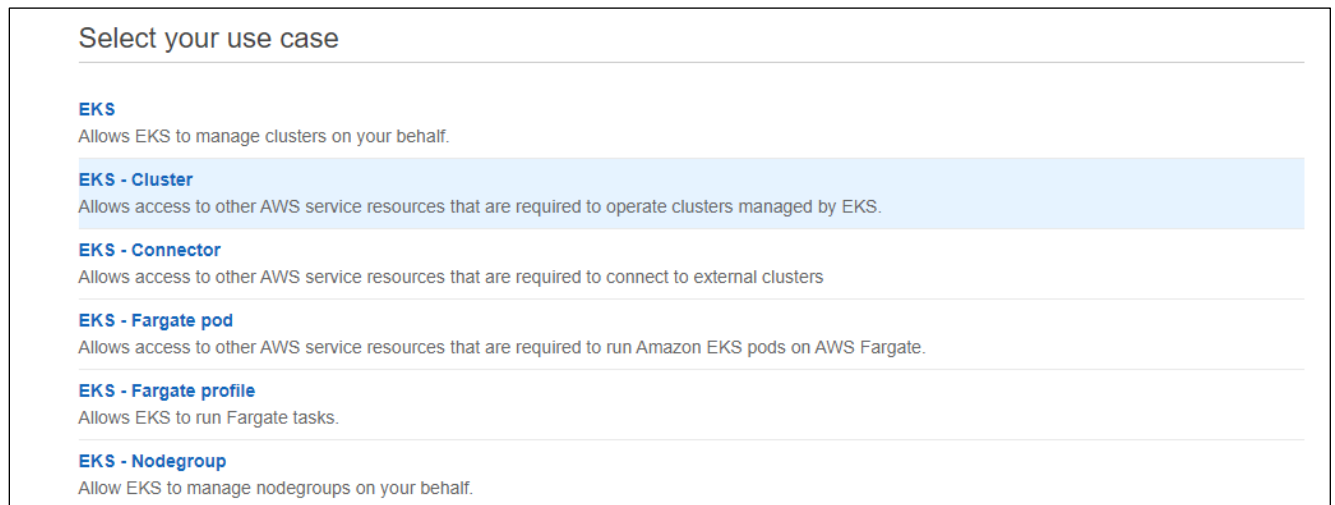


Figure 2.13

6. Click **Next: Tags**.
7. Click the **Next: Review**.
8. Specify the user-defined role name given under review and then click **Create role**.

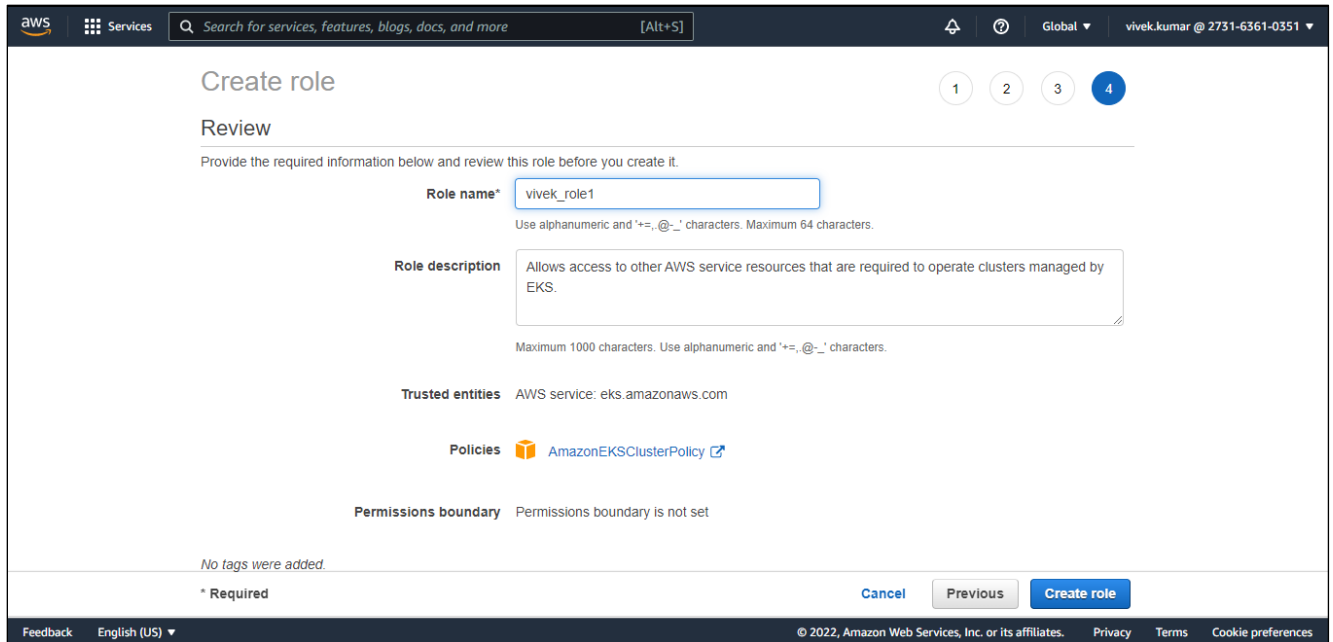


Figure 2.14

2.7 Creating a security group

Creating a security group is required for the EKS cluster.

Perform the below steps to create a Security Group:

1. In the VPC Dashboard, go to the **Security Groups** and click **Create security group**.
2. On the Create security group tab, specify the user-defined security group name and description.
3. Select the created VPC and click **Create**.

2.8 Creating an EKS cluster

Before creating the EKS Cluster, you must sign in to the AWS Management Console using an IAM user instead of using a root user for EKS Cluster creation.

Perform the below steps to create an AWS Kubernetes Cluster:

1. Go to **EKS Service** and click the **Next step**.



Figure 2.15

2. Specify the following in the Create Cluster:
 - **Cluster name:** Enter the User-defined name.
 - **Kubernetes version:** Select default that is, **1.21**.
 - **Role name:** Select the created IAM role.
3. Click **Next**.

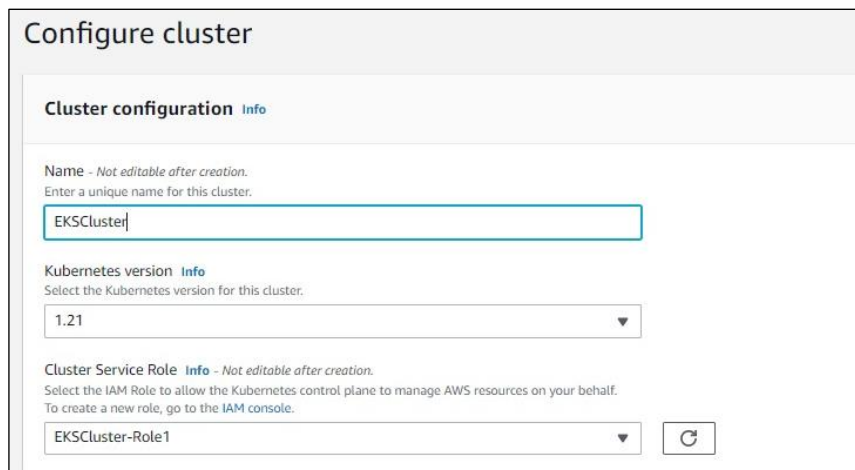


Figure 2.16

4. Specify the following in the Networking info:
 - **VPC:** Select the created VPC.
 - **Subnets:** Select all the subnets of the Mumbai region.
 - **Security groups:** Select the created security group.
 - **Cluster endpoint access:** Enable both Private access and Public access.
5. Once all the details are specified, click **Next**.

Networking Info
These properties cannot be changed after the cluster is created.

VPC Info
Select a VPC to use for your EKS Cluster resources.
To create a new VPC, go to the [VPC console](#).

vpc-002069cbb4dd53364 | vivek_vpc

Subnets Info
Choose the subnets in your VPC where the control plane may place elastic network interfaces (ENIs) to facilitate communication with your cluster.
To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets

subnet-065fc1d0079ee9286 X subnet-02b4e1422db90c20b X
subnet-0334d5a8e34d15ec8 X

Security groups Info
Choose the security groups to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets.
To create a new security group, go to the corresponding page in the [VPC console](#).

Select security groups

sg-049392491607b5e43 X

Cluster endpoint access Info
Configure access to the Kubernetes API server endpoint.

Public
The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.

Public and private
The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.

Private
The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.

▶ **Advanced Settings**

Cancel Previous **Next**

Figure 2.17

6. On the Configure tab, click **Next**.
7. In the Review tab, click **Create** and create a page.

NOTE:

Ensure that the cluster status is **ACTIVE**.

2.9 Creating a key pair

Perform the below steps to create a key pair:

1. Go to **EC2 Dashboard**.
2. Click **Key Pairs**.
3. Click **Create Key Pair**. The Create Key Pair dialog appears.
4. Specify the **Key pair name** and click **Create**. A `<KeyPair Name>.pem` gets downloaded.

NOTE:

For the SSH connection, you must keep the key pair name. Convert this `.pem` file to `.ppk` for SSH connection through the local machine.

2.10 Provisioning Kubernetes worker nodes using cloud formation

Perform the below steps to provide provision Kubernetes worker nodes using Cloud Formation:

1. Download the latest version of the AWS CloudFormation template.

```
curl -o amazon-eks-nodegroup.yaml https://raw.githubusercontent.com/aws-labs/amazon-eks-ami/master/amazon-eks-nodegroup.yaml
```

NOTE:

To download the latest YAML file, refer to the below link:

<https://docs.aws.amazon.com/eks/latest/userguide/launch-workers.html>

2. Open the **AWS CloudFormation** console.
3. Go to **Create Stack** under **With new resources (standard)**.
4. To **Specify a template**, select **Upload a template file** and then select **Select** file. Select the *amazon-eks-nodegroup.yaml* file that you downloaded earlier and then click **Next**.

The screenshot shows the 'Create stack' wizard in the AWS CloudFormation console. The 'Specify template' step is active, showing options for 'Template source' and 'Upload a template file'. The 'Upload a template file' option is selected, and a file named 'amazon-eks-nodegroup.yaml' is shown as uploaded. The 'S3 URL' field is populated with a long URL, and a 'View in Designer' button is visible next to it. At the bottom, there are 'Cancel' and 'Next' buttons.

Figure 2.18

5. On the Specify stack details, specify the following details:
 - **Stack name** - Select a stack name for the AWS CloudFormation stack.
 - **ClusterName** - Enter the name used for creating the Amazon EKS cluster. This name must exactly match the name as per the given name.
 - **ClusterControlPlaneSecurityGroup** - Select the SecurityGroups of EKS Cluster.

- **NodeGroupName** - Enter a name for your node group. Later, this name identifies the Auto Scaling node group created for your worker nodes.
- **NodeAutoScalingGroupMinSize** - Enter the minimum number of nodes that your worker node Auto Scaling group can scale in them.
- **NodeAutoScalingGroupDesiredCapacity** - Enter the desired number of nodes to scale your created stack.
- **NodeAutoScalingGroupMaxSize** - Enter the maximum number of nodes that your worker node Auto Scaling group can scale out in them.
- **NodeInstanceType** - Select an instance type for your worker nodes.
- **NodeImageIdSSMParam** - This is a pre-populated optimized Amazon Linux AMI ID for a Kubernetes version. Change the Kubernetes minor version supported with EKS Cluster. For example, earlier you had created an EKS Cluster with v1.21. You must use the same version here as shown below:
/aws/service/eks/optimized-ami/1.21/amazon-linux-2/recommended/image_id
- **NodeImageId** - This is an optional field. If you are using your custom AMI, then enter a node AMI ID otherwise leave it blank.
- **NodeVolumeSize** - Specify a node volume size for your nodes, in GiB.
- **KeyName** - Enter the name of an Amazon EC2 SSH key pair that you can use to connect using SSH into your worker nodes after they launch.

6. After specifying the above details, click **Next**.

Specify stack details

Stack name

Stack name

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.

EKS Cluster

ClusterName
The cluster name provided when the cluster was created. If it is incorrect, nodes will not be able to join the cluster.

ClusterControlPlaneSecurityGroup
The security group of the cluster control plane.

Worker Node Configuration

NodeGroupName
Unique identifier for the Node Group.

NodeAutoScalingGroupMinSize
Minimum size of Node Group ASG.

NodeAutoScalingGroupDesiredCapacity
Desired capacity of Node Group ASG.

NodeAutoScalingGroupMaxSize
Maximum size of Node Group ASG. Set to at least 1 greater than NodeAutoScalingGroupDesiredCapacity.

NodeInstanceType
EC2 instance type for the node instances

NodeImageIdSSMParam
AWS Systems Manager Parameter Store parameter of the AMI ID for the worker node instances. Change this value to match the version of Kubernetes you are using.

NodeImageId
(Optional) Specify your own custom image ID. This value overrides any AWS Systems Manager Parameter Store value specified above.

NodeVolumeSize
Node volume size

NodeVolumeType
EBS volume type for nodes

KeyName
The EC2 Key Pair to allow SSH access to the instances

BootstrapArguments
Arguments to pass to the bootstrap script. See files/bootstrap.sh in <https://github.com/awslabs/amazon-eks-ami>

DisableIMDSv1

Worker Network Configuration

VpcId
The VPC of the worker instances

Subnets
The subnets where workers can be created.

subnet-04569865ba524020d X subnet-05595381b98be7c19 X subnet-049ab9ea6a419f9e4 X

Cancel Previous **Next**

Figure 2.19

7. Click **Next** and configure stack options.
8. In the Review tab, review all the specified details and select the checkbox **I acknowledge that AWS CloudFormation might create IAM resources**.
9. Click **Create stack**. Ensure that the creation status becomes **CREATE_COMPLETE** now.

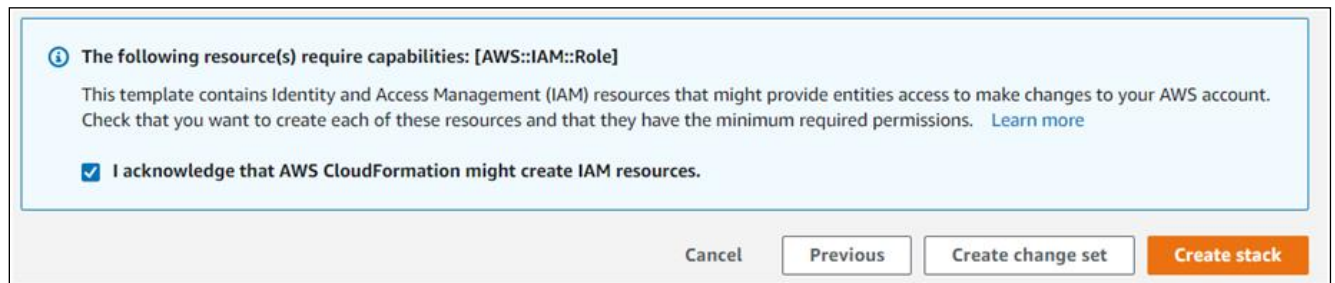


Figure 2.20

NOTE:

Once your stack creation is complete, select it on the console, and select the Outputs tab. Record the **NodeInstanceRole** for the node group created earlier. Keep this NodeInstanceRole to configure your Amazon EKS worker nodes.

2.11 Adding inbound rule in EC2 instance

Perform the below steps to add the Inbound rule in an EC2 instance:

1. Go to **VPC Dashboard**.
2. Select **Security Groups** and select the security group mapped with EKS Node/EC2 Instance.
3. Go to the **Inbound Rules** tab and click **Edit rules**.
4. Click **Add a new Rule** and specify 22 in the **Port Range** field.
5. Select **My IP** or any other IP range from where you want to access the ssh of worker nodes in the Source field.



Figure 2.21

6. Connect this EC2 instance from the local machine using the default user name **ec2-user** and SSH key pair as created earlier.

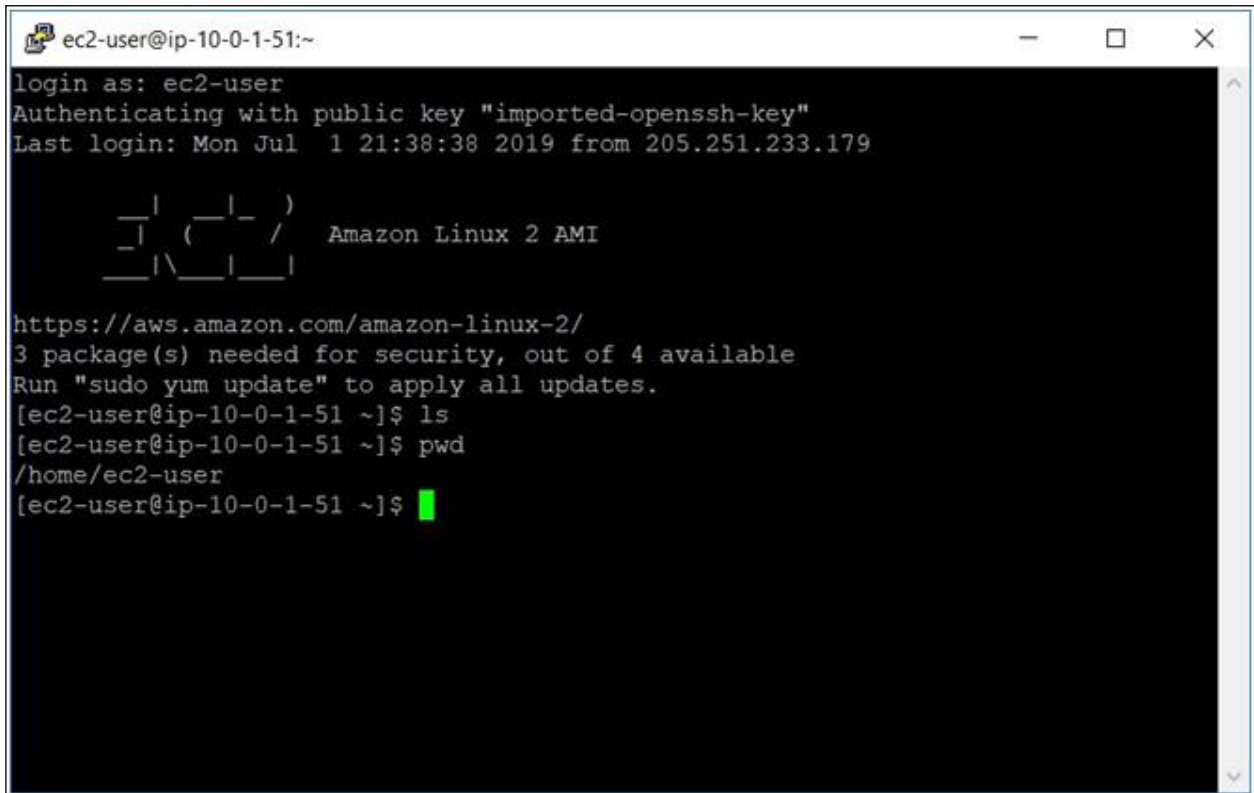


Figure 2.22

2.12 Enabling worker node to join EKS cluster

Perform the following to enable the Worker Node/EC2 instance to EKS Cluster:

1. Connect to the worker node through the Putty.
2. Install or Update AWS Cli on the worker node by using the below URL:
<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

NOTE:

AWS cli version 2.x is required. Check the AWS cli version using the below command:

```
aws --version
```

3. Install the Kubectl on the worker node by using the below URL:
<https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html>
4. Execute the below command to the worker node:

```
aws configure
```
5. After the above command is executed, specify the following details:
 - **Access Key ID:** Provide the **Access key ID** of the user which is used to create the EKS Cluster.
 - **Secret Access Key:** Provide the **SecretKey ID** of the user which is used to create the EKS Cluster.
 - **Region:** ap-south-1
 - **Output:** JSON

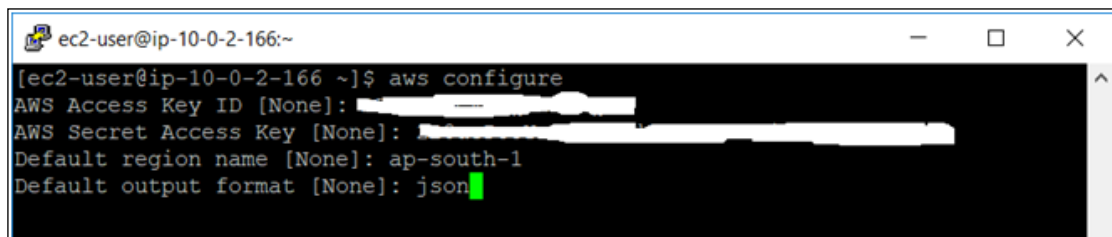


Figure 2.23

6. Now, execute the below command:

```
aws eks --region <RegionName> update-kubeconfig --name <ClusterName>
```

For example,

```
aws eks --region ap-south-1 update-kubeconfig --name EKSCluster
```

7. To download the configuration map, execute the below command:

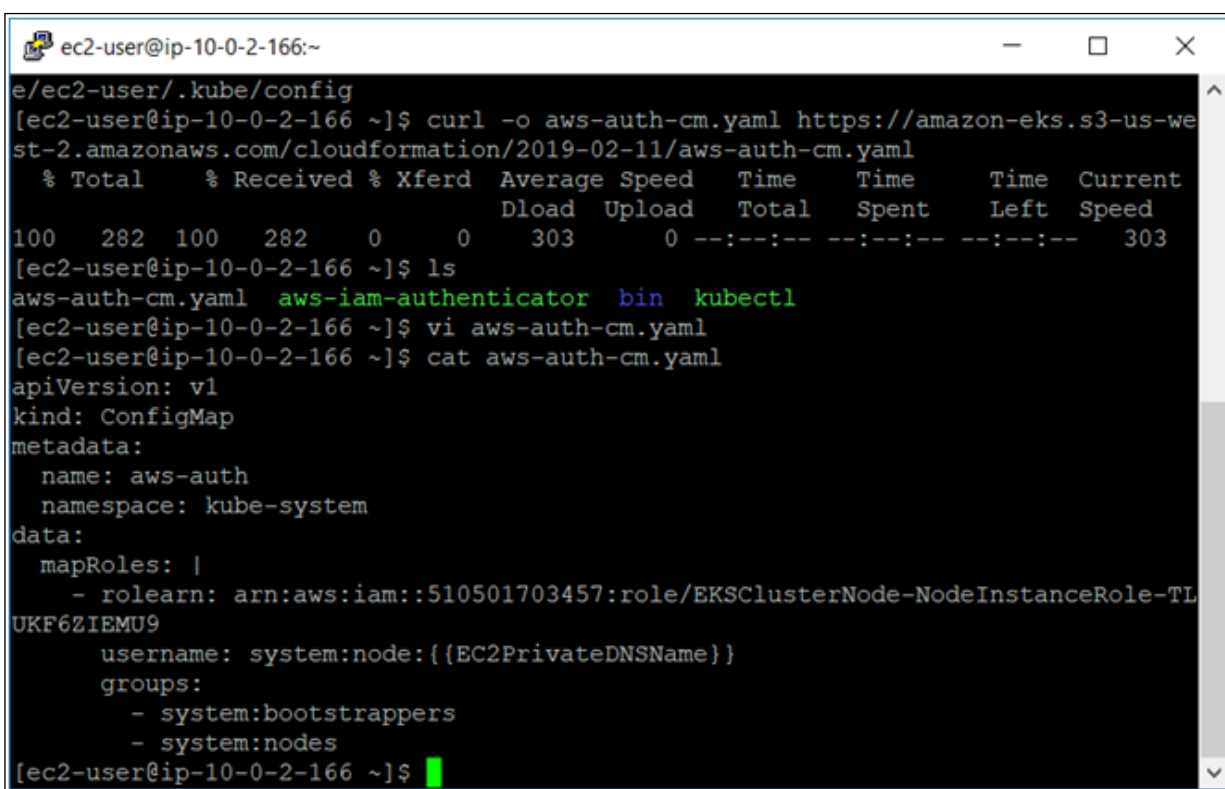
```
curl -o aws-auth-cm.yaml https://amazon-eks.s3.us-west-2.amazonaws.com/cloudformation/2020-10-29/aws-auth-cm.yaml
```

NOTE:

To download the latest S3 URL, refer to the below link:

<https://docs.aws.amazon.com/eks/latest/userguide/launch-workers.html>

8. After the above command gets executed, a file `aws-auth-cm.yaml` is downloaded to the worker node.
9. Open this file in the edit mode and replace the <ARN of instance role (not instance profile)> snippet with the **NodeInstanceRole** value recorded in the procedure.
10. Save the file.



```
ec2-user@ip-10-0-2-166:~
e/ec2-user/.kube/config
[ec2-user@ip-10-0-2-166 ~]$ curl -o aws-auth-cm.yaml https://amazon-eks.s3-us-we
st-2.amazonaws.com/cloudformation/2019-02-11/aws-auth-cm.yaml
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left     Speed
100  282  100  282    0    0   303      0  --:--:--  --:--:--  --:--:--  303
[ec2-user@ip-10-0-2-166 ~]$ ls
aws-auth-cm.yaml  aws-iam-authenticator  bin  kubectl
[ec2-user@ip-10-0-2-166 ~]$ vi aws-auth-cm.yaml
[ec2-user@ip-10-0-2-166 ~]$ cat aws-auth-cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: arn:aws:iam::510501703457:role/EKSClusterNode-NodeInstanceRole-TL
UKF6ZIEMU9
    username: system:node:{{EC2PrivateDNSName}}
  groups:
    - system:bootstrappers
    - system:nodes
[ec2-user@ip-10-0-2-166 ~]$
```

Figure 2.24

11. Execute the below command on the worker node.

```
kubectl apply -f aws-auth-cm.yaml
```

Check the status of your nodes and wait for them to reach the Ready status.

```
kubectl get nodes --watch
```


2.13 Running Kubectl from a local machine

Before running the kubectl commands from your local machine, ensure that you have the following prerequisites:

- kubectl: <https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html>.
- aws-cli: <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html>.
- Execute the below command to the worker node:

```
aws configure
```

- Once the above command gets executed, specify the following details:
 - **Access Key ID:** Provide the **Access key ID** of the user which is used to create the EKS Cluster.
 - **Secret Access Key:** Provide the **SecretKey ID** of the user which is used to create the EKS Cluster.
 - **Region:** ap-south-1
 - **output:** JSON
- Execute the below command:

```
aws eks --region <RegionName> update-kubeconfig --name <Cluster_Name>
```

For example,

```
aws eks --region ap-south-1 update-kubeconfig --name EKSCluster
```

- Execute kubectl commands from our machine.

For example,

```
kubectl get pods
```

2.14 Creating EFS

Perform the below steps to create an AWS Elastic File System (EFS) storage:

1. Go to **EFS Service** and select **Create file system**.
2. In the **Name - optional**, specify the user-defined name for your file system that is, **omnidocs-efs**.
3. In the **Virtual Private Cloud (VPC)**, select the created VNC for your EKS cluster.
4. In the **Availability and Durability**, select Regional.
5. Click **Create**.

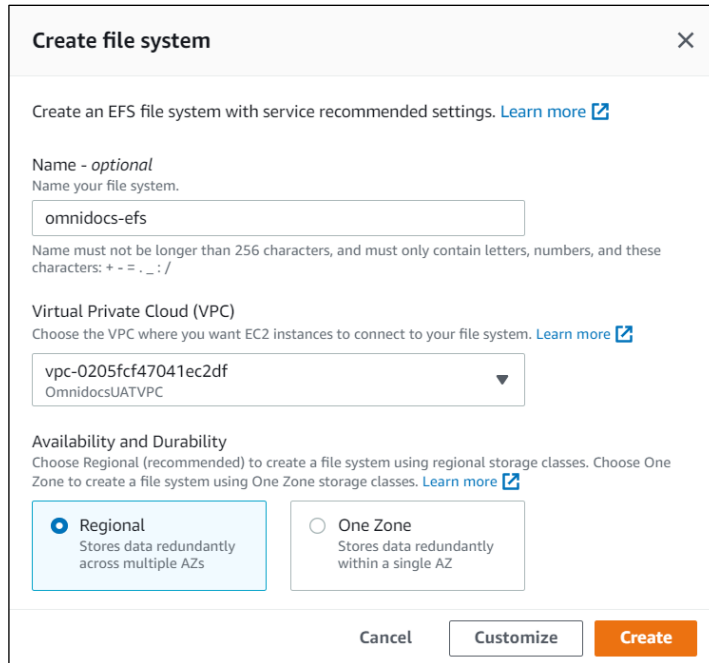


Figure 2.25

6. Open the created EFS and switch to the **Access Point** tab and select **Create access point**.

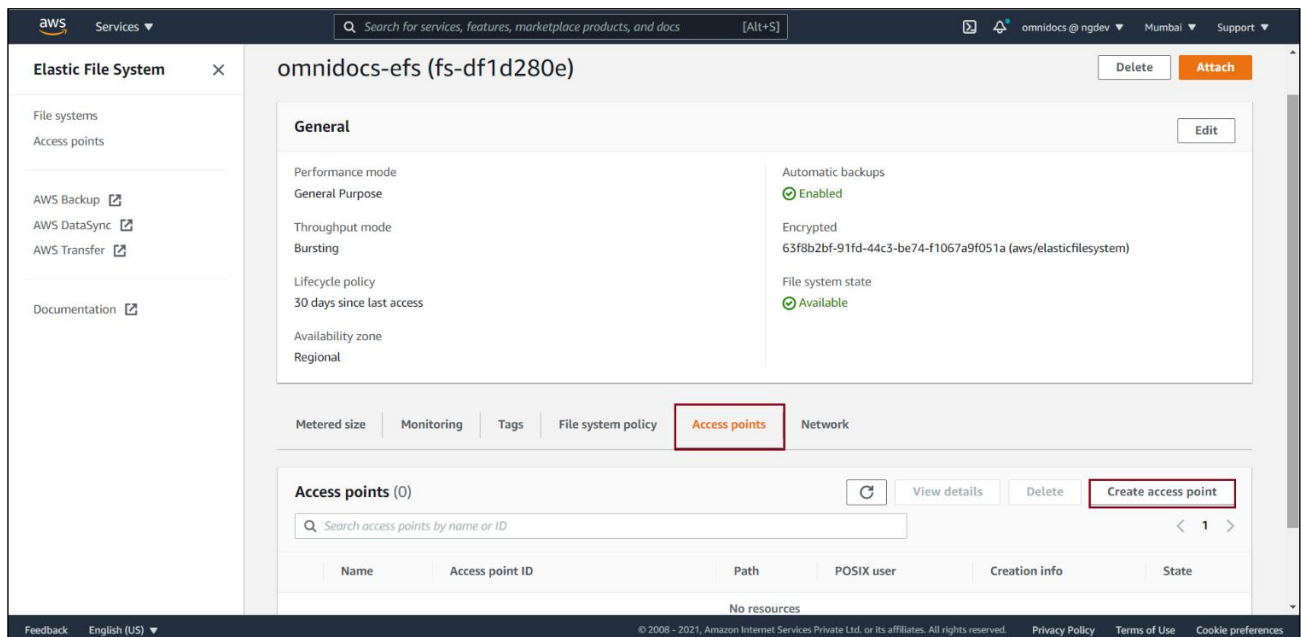


Figure 2.26

7. In the **Name – optional**, specify the user-defined name that is, **omnidocs-efs-accesspoint**.
8. In the **Root directory path**, use **/mnt/efs** [This directory path must exist on EC2 worker nodes, else, create it if does not exist].

9. In the **POSIX user –optional**, specify **1000** in the User ID, Group ID, and Secondary group IDs textboxes.
10. In the **Root directory creation permissions – optional**, specify **1000** in the Owner user ID and Owner group ID. And keep the default POSIX permission **0755**.
11. Click **Create access point**.

NOTE:

The Worker node’s Security group must be added to the EFS Network.

Create access point for fs-082ceed3f363f7872

An access point is an application-specific entry point into an EFS file system that makes it easier to manage application access to shared datasets. [Learn more](#)

Details

File system
Choose the file system to which your access point is associated.
fs-082ceed3f363f7872

Name - optional
omnidocs-efs-accesspoint
Maximum of 256 Unicode letters, whitespace, and numbers, plus + - = _ . /

Root directory path - optional
Connections use the specified path as the file system's virtual root directory. [Learn more](#)
/mnt/efs
Example: "/foo/bar"

POSIX user - optional
The full POSIX identity on the access point that is used for all file operations by NFS clients. [Learn more](#)

User ID
POSIX user ID used for all file system operations using this access point.
1000
Accepts values from 0 to 4294967295

Group ID
POSIX group ID used for all file system operations using this access point.
1000
Accepts values from 0 to 4294967295

Secondary group IDs
Secondary POSIX group IDs used for all file system operations using this access point.
1000
A comma-separated list of valid POSIX group IDs

Root directory creation permissions - optional
EFS will automatically create the specified root directory with these permissions if the directory does not already exist. [Learn more](#)

Owner user ID
Owner user ID for the access point's root directory, if the directory does not already exist.
1000
Accepts values from 0 to 4294967295

Owner group ID
Owner group ID for the access point's root directory, if the directory does not already exist.
1000
Accepts values from 0 to 4294967295

POSIX permissions to apply to the root directory path
755
An octal number representing the file's mode bits.

Tags - optional
Add tags to associate key-value pairs to your resource. [Learn more](#)

Tag key: Tag value - optional:

You can add 49 more tag(s)

Figure 2.27

2.15 Mounting EFS to worker nodes

For mounting the Elastic File System (EFS) with Worker nodes, install the **amazon-efs-utils** using the below command:

```
sudo yum install -y amazon-efs-utils
```

Add the below line to the */etc/fstab* file on each Worker Node:

```
fs-8241f853.efs.ap-south-1.amazonaws.com:/mnt/efs efs  
_netdev,tls,accesspoint=fsap-0bbac155fbd3ad350 0 0
```

Where,

fs-8241f853.efs.ap-south-1.amazonaws.com =Elastic File System DNS Name
/mnt/efs=Existing directory structure of EC2 instance [Create if does not exist]
fsap-0bbac155fbd3ad350= Attached Access Point to the EFS

Execute the below command:

```
sudo mount -a
```

NOTE:

You must mount EFS to all the running worker nodes.

2.16 Configuring Kubernetes dashboard

Use the below URL to configure the Kubernetes Dashboard:

<https://docs.aws.amazon.com/eks/latest/userguide/dashboard-tutorial.html>

Once Kubernetes Dashboard is configured, execute the below command:

```
kubectl proxy
```

Use the below URL to open the Kubernetes Dashboard:

<http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#!/login>

2.17 Configuring AWS load balancer controller

The AWS Load Balancer Controller manages AWS Elastic Load Balancers for a Kubernetes cluster. It creates an application load balancer when you create a Kubernetes ingress. The Ingress resource configures the ALB to route HTTP or HTTPS traffic to different pods within the cluster.

Perform the below steps to configure the AWS Load Balancer Controller:

1. Create an IAM OIDC provider and associate it with your cluster using the below commands:

```
eksctl utils associate-iam-oidc-provider \
  --region <region-code> \
  --cluster <cluster-name> \
  --approve
```

NOTE:

If you don't have the eksctl version 0.25.0 or later installed, then complete the installation using the below URL:

<https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html#installing-eksctl>

2. Download an IAM policy for the AWS Load Balancer Controller that allows it to make calls to AWS APIs on your behalf using the below command:

```
curl -o iam_policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.3.1/docs/install/iam_policy.json
```

NOTE:

To get the latest iam-policy, refer to the below link:

<https://docs.aws.amazon.com/eks/latest/userguide/aws-load-balancer-controller.html>

3. Create an IAM policy called **AWSLoadBalancerControllerIAMPolicy** using the downloaded policy.

```
aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy --
policy-document file://iam_policy.json
```

4. Create an IAM role for the AWS Load Balancer Controller and attach the role to the service account created in the further steps.

Perform the below steps to create the IAM role:

1. Open the IAM console and select **Create Roles**.
2. In the **Select type of trusted entity** section, select **Web identity**.
3. In the **Select a web identity provider**, specify the following:
 - i. In the **Identity provider**, select the URL for your cluster.
 - ii. In the **Audience**, select *sts.amazonaws.com*.
 - iii. Click **Next: Permissions**.
4. In the Attach Policy section, select the policy **AWSLoadBalancerControllerIAMPolicy**
5. Specify the role name as **AmazonEKSLoadBalancerControllerRole** and then select **Create Role**.
6. After the role is created, select the role in the console to open it for editing.
7. Select the **Trust relationships** tab and select the **Edit trust policy**.
 - i. Edit the OIDC provider suffix and change it from **aud** to: **sub**.
 - ii. Replace *sts.amazonaws.com* with the service account ID given in the quotes below. (This service account id is created in further steps).

```
"system:serviceaccount:kube-system:aws-load-balancer-controller"
```

- iii. The resulting line in policy must be as follows:

```
"oidc.eks.region-  
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B716D3041E:sub": "  
"system:serviceaccount:SERVICE_ACCOUNT_NAMESPACE:SERVICE_ACCOUNT_NAME"
```

For example,

```
"oidc.eks.ap-south-  
1.amazonaws.com/id/C9D4F2E6E31D3880DCE2BEFEA007C4CB:sub": "  
"system:serviceaccount:kube-system:aws-load-balancer-controller"
```

- iv. Select **Update policy** to finish.

NOTE:

Take note of the Role ARN of the newly created role **AmazonEKSLoadBalancerControllerRole**.

8. Create a Kubernetes service account named **aws-load-balancer-controller** in the **kube-system namespace**. To create the same, save the following contents to a file that's named as **aws-load-balancer-controller-service-account.yaml**, replacing the created **role ARN**.

```
apiVersion: v1  
kind: ServiceAccount  
metadata:  
  labels:  
    app.kubernetes.io/component: controller  
    app.kubernetes.io/name: aws-load-balancer-controller  
  name: aws-load-balancer-controller  
  namespace: kube-system  
  annotations:  
    eks.amazonaws.com/role-arn:  
arn:aws:iam::273163610351:role/AmazonEKSLoadBalancerControllerRole
```

NOTE:

To download the latest service account YAML contents, refer to the below link:

<https://docs.aws.amazon.com/eks/latest/userguide/aws-load-balancer-controller.html>

9. Execute the following command to create a Kubernetes Service Account:

```
kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

10. Install the **cert-manager** using the following command:

```
kubectl apply --validate=false -f https://github.com/jetstack/cert-  
manager/releases/download/v1.5.4/cert-manager.yaml
```

11. Download the controller specification using the below command:

```
curl -Lo v2_4_1_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.4.1/v2_4_1_full.yaml
```

12. Make the following edit in `v2_4_1_full.yaml` file.

- a. Delete the **kind: ServiceAccount** section of the file.
- b. Replace **your-cluster-name** in the **Deployment spec** section of the file with the name of your cluster.

For example,

```
spec:
  containers:
    - args:
      - --cluster-name=<ClusterName>
      - --ingress-class=alb
```

- c. Apply the file.

```
kubectl apply -f v2_4_1_full.yaml
```

NOTE:

If a user is facing issues like no matches for kind **IngressClassParams** in version **elbv2.k8s.aws/v1beta1** then execute the below command to fix this issue:

```
sudo yum install git -y
kubectl apply -k "github.com/aws/eks-charts/stable/aws-load-balancer-controller//crds?ref=master"
kubectl apply -f v2_4_1_full.yaml
```

13. Use the below command to verify the status of the AWS Load Balancer Controller:

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

14. Use the below command to check the logs of the AWS Load Balancer Controller:

```
kubectl logs deployment.apps/aws-load-balancer-controller -n kube-system
```

2.18 Configuring AWS Elastic Redis Cache

To configure the AWS Elastic Redis, perform the below:

1. Sign in to the AWS Management Console and open the ElastiCache console.
2. Select **Get Started Now**. If you already have an available cluster, select **Create**.
3. For the **Cluster engine**, select **Redis**.
4. Complete the **Redis settings** section as follows:
 - i. **Cluster creation method** – Configure and create a new cluster.
 - ii. **Cluster mode** – Disabled.

Cluster settings Info

Choose a cluster creation method
Choose one of the following options to create a new cluster.

Configure and create a new cluster
Set all of the configuration options for your new cluster.

Restore from backups
Use an existing backup or .rdb file to restore a cluster.

Cluster mode
Scale your cluster dynamically with no downtime.

Enabled
Cluster mode enables replication across multiple shards for enhanced scalability and availability.

Disabled
The Redis cluster will have a single shard (node group) with one primary node and up to 5 read replica.

If you choose cluster mode disabled you cannot change the number of shards. The configuration supports all Redis commands and functionality but limits maximum cache size and performance. [Learn more](#)

Figure 2.28

- iii. **Name** – Enter a user-defined name.
- iv. **Description (optional)** – Enter any description.
- v. **Location** – AWS Cloud
- vi. **Multi-AZ** – Enabled

Name

The name is required, can have up to 40 characters, and must begin with a letter. It should not end with a hyphen or contain two consecutive hyphens. Valid characters: A-Z, a-z, 0-9, and - (hyphen).

Description - optional

Location

Choose whether to host the cluster in the AWS Cloud or on premises.

Location

AWS Cloud

Use the AWS Cloud for your ElastiCache instances.

On premises

Create your ElastiCache instances on an Outpost (through AWS Outposts). You need to create a subnet ID on an Outpost first.

Multi-AZ

Enable

Multi-AZ provides enhanced high availability through automatic failover to a read replica, cross AZs, in case of a primary node failover.

Figure 2.29

- vii. **Engine version Compatibility** - Select the latest version.
- viii. **Port** - Accept the default port.
- ix. **Parameter group** - Accept the default parameter group.
- x. **Node type**: Select the node type that you want to use for this cluster.
- xi. **Number of replicas** - Select the number of nodes you want to provide as a provision for this cluster.

Cluster settings

Use the following options to configure the cluster.

Engine version
Version compatibility of the Redis engine that will run on your nodes.

6.2 ▼

Port
The port number that nodes accept connections on.

6379

Parameter groups
Parameter groups control the runtime properties of your nodes and clusters.

default.redis6.x ▼

Node type
The type of node to be deployed and its associated memory size.

cache.r5.large
13.07 GiB memory Up to 10 Gigabit network performance ▼

Number of replicas
Enter the number of replicas between 0 and 5. Zero replicas will not enable an enhanced cluster with primary/replica roles.

1

Figure 2.30

- xii. Create subnet group
 - a. **Name** - Type a unique name
 - b. **Description** - Type any description.
 - c. **VPC ID** - Select a VPC in which EKS cluster is created.
 - d. **Subnets** - Select all subnets.
 - e. **Availability zones placement:** Accept as default that is, No Preference.
 - f. Click **Next**.

Subnet group settings

A subnet group is a collection of subnets (typically private). Designate a subnet group for your clusters running in an Amazon Virtual Private Cloud (VPC) environment.

Subnet groups

Choose existing subnet group
 Create a new subnet group

Name

newgen-redis-subnet-group

The name is required, can have up to 255 characters, and must begin with a letter. It should not end with a hyphen or contain two consecutive hyphens. Valid characters: A-Z, a-z, 0-9, and - (hyphen).

Description - *optional*

newgen-redis-subnet-group

VPC ID

The identifier for the VPC environment where your cluster is to run.

vpc-0205fcf47041ec2df ▼ Create VPC [↗](#)

i For Multi-AZ high availability mode, choose IDs for at least two subnets from two Availability Zones in the table below.

Selected subnets (2) Manage

Availability Zone ▲	Subnet ID ▼	CIDR block ▼
ap-south-1a	subnet-0048aa992f3464134	10.0.2.0/24
ap-south-1b	subnet-0defa4d35f35b9415	10.0.1.0/24

► Tags for subnet group

Availability Zone placements

Use the following fields to configure placements for Availability Zones.

Availability Zone placements

HA mode - Globally, distribute AZs to maximize AZ spread across shard masters. At the second level, spread nodes within a shard across AZs for within-shard HA. Low latency mode - For fast writes, put all shard masters in the same AZ.

No preference ▼

Cancel Next

Figure 2.31

- xiii. Under the **Advanced settings** section as follows:
 - a. Disable the **Encryption at rest**.
 - b. Disable the **Encryption in transit**.
 - c. Select EKS Worker node's in security group and click **Next**.

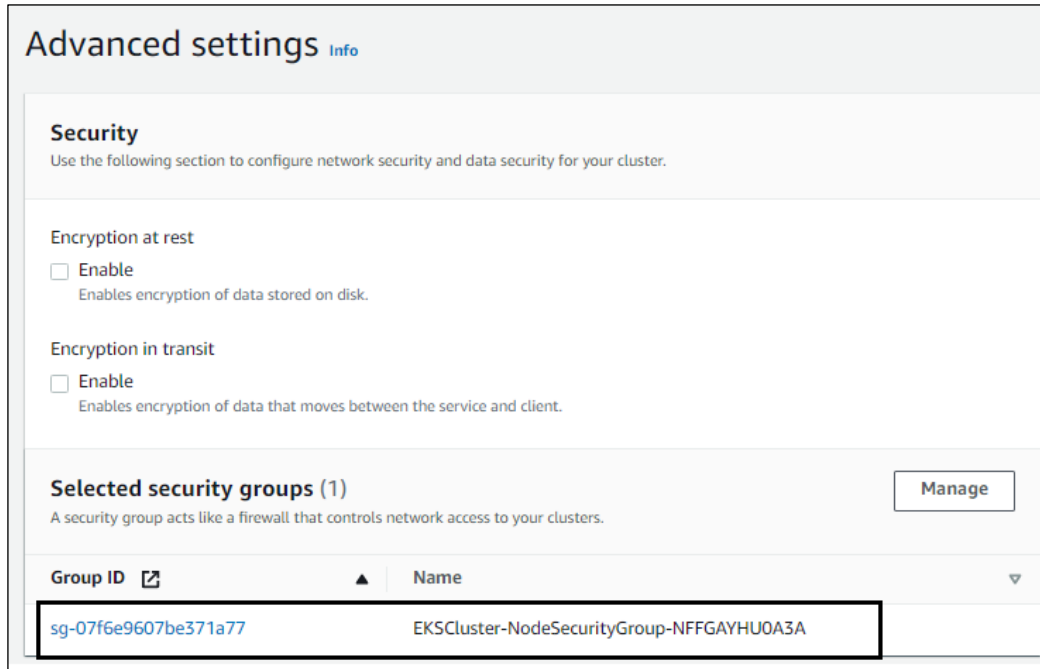


Figure 2.32

- 5. Review the settings and click **Create** to launch your Redis cluster.

2.19 Registering domain using route-53

AWS ALB Ingress Controller creates an Application Load Balancer and routes the incoming requests to the target Kubernetes services according to the host-based routing rules. Host-based routing is a capability of ALB that redirects the user requests to the right service based on the request-host header.

For example, we can set the rules as below:

- If URL is *ibpsportal.aws.co.in* then redirect to the iBPS Portal container.
- If URL is *ibpsbam.aws.co.in* then redirect to the iBPS BAM container.

To support host-based routing, you must register a domain and create a new record.

Perform the below steps to Record Set for each host path:

1. Register a domain using the AWS Route-53 service. Open the route53 service and go to **domain registration** in the Domain section.

- Once the domain is registered, it creates a Hosted Zone. Click the newly created Hosted Zone list and then click **Go to Record Sets**.

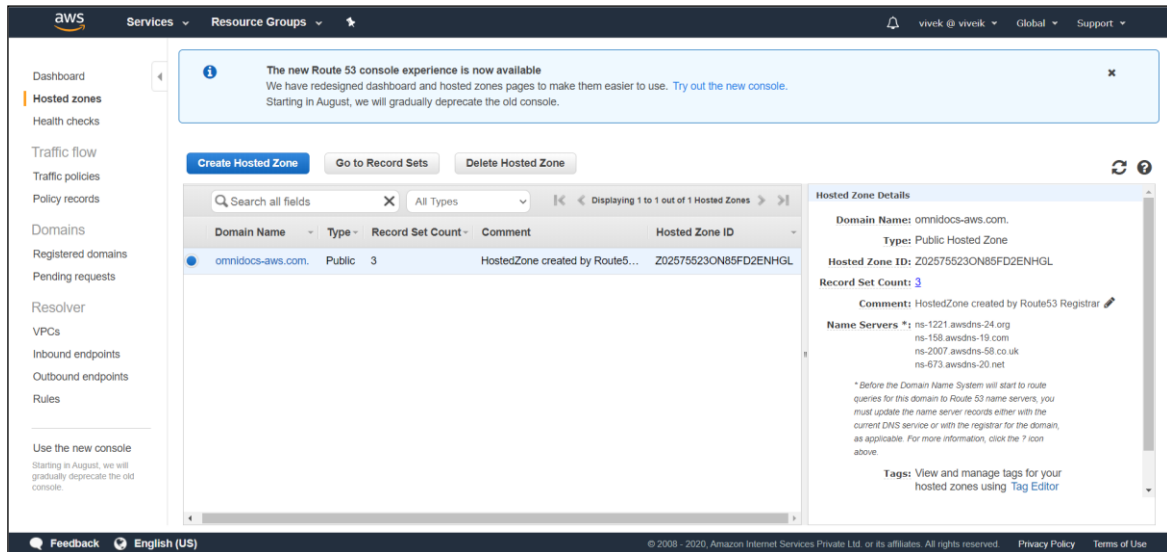


Figure 2.33

- Click **Create Record Set**. The Create Record Set dialog appears:

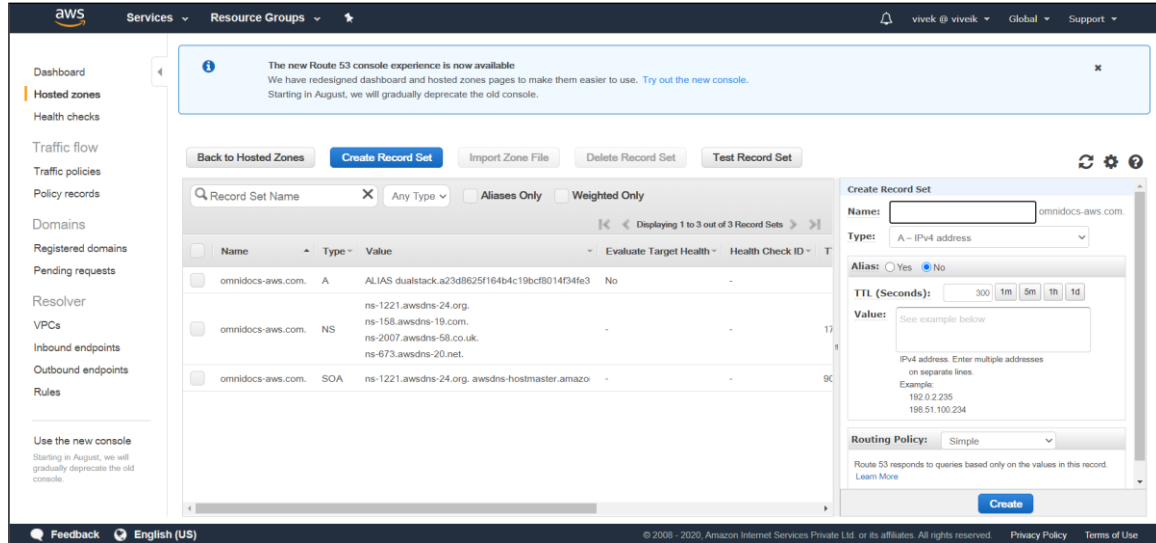


Figure 2.34

- Enter the following details in the Create Record Set to create a new RecordSet.
 - Name:** Enter the user-defined name.
 - Type:** Select type as *A – IPv4-address*.
 - Alias:** Select alias as **Yes**.

- iv. **Alias Target:** Select the alias target as **Load Balancer**.

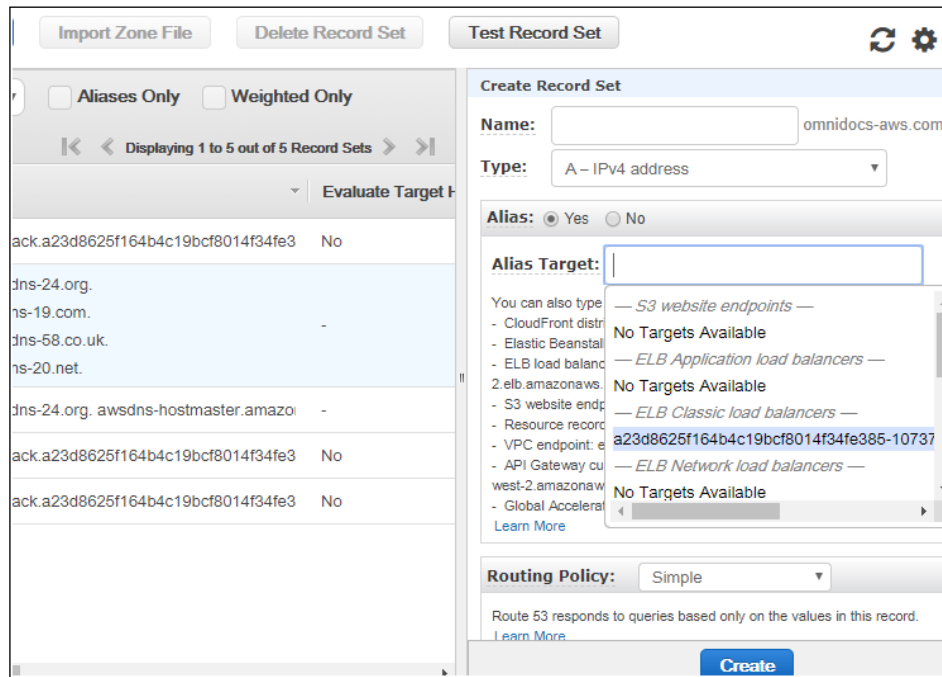


Figure 2.35

Use this RecordSet as a host path for Ingress Controller. Thus, ALB is registered with a Domain name.

2.20 Generating SSL certificate against the registered domain

This section explains how to generate an SSL certificate against the registered domain.

Prerequisite:

You must have a registered domain in Route53.

Perform the below steps to generate an SSL Certificate:

1. Go to the **Certificate Manager** given under the **Services**.
2. Click **Request a certificate**.

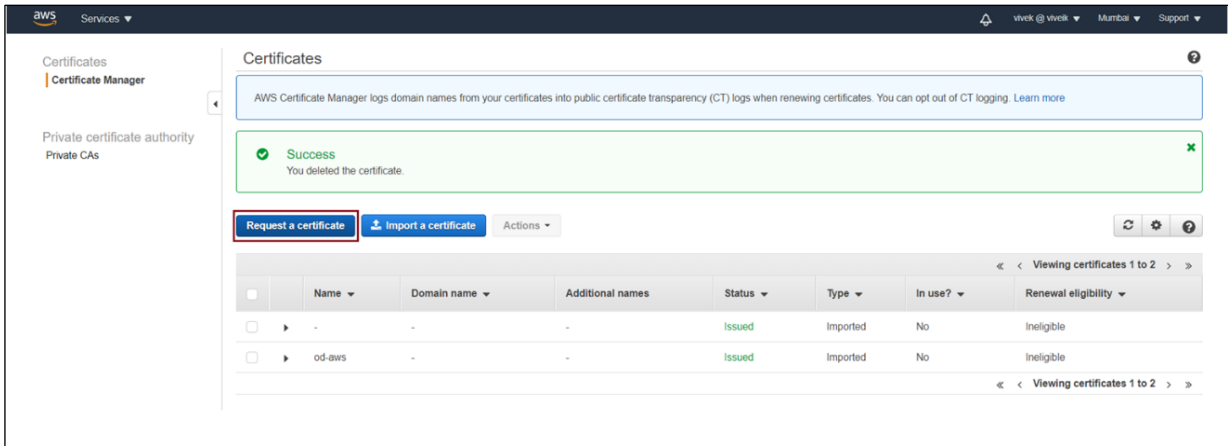


Figure 2.36

3. To **Request a public certificate**, select the type of certificate for ACM to provide.

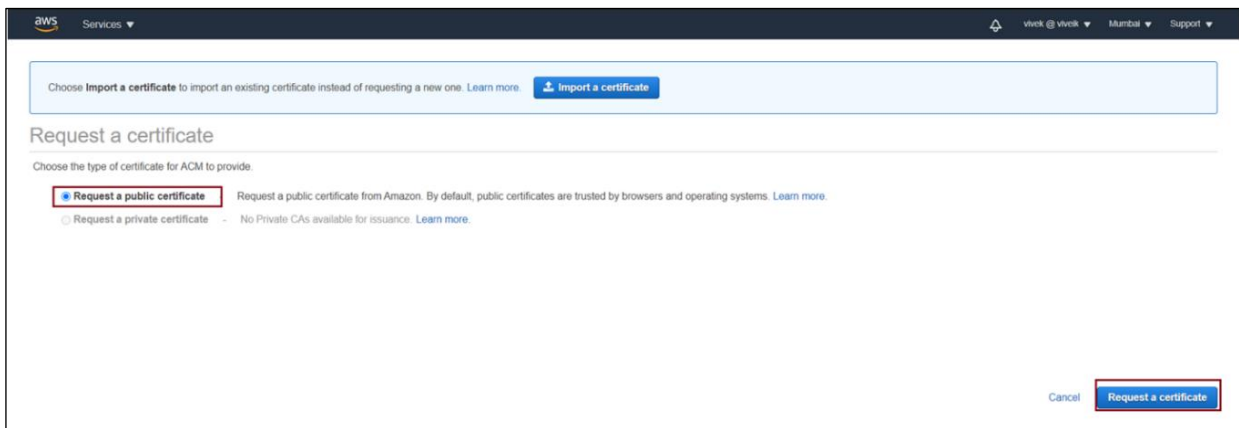


Figure 2.37

4. Add your registered domain name like *omnidocs-aws.com*.
5. Add another name to this certificate as **.<DOMAIN_NAME>* like **.omnidocs-aws.com*)

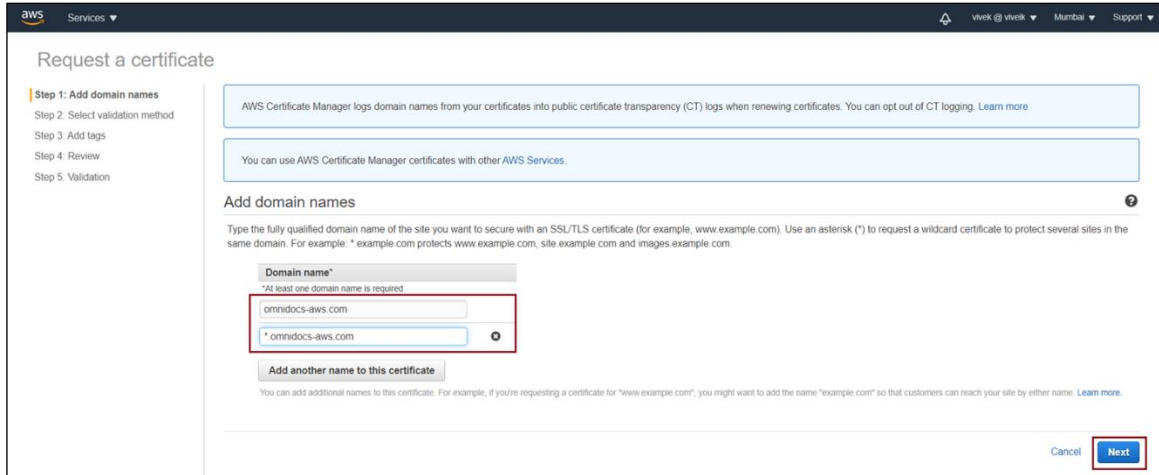


Figure 2.38

6. Select a validation method: **Email validation.**

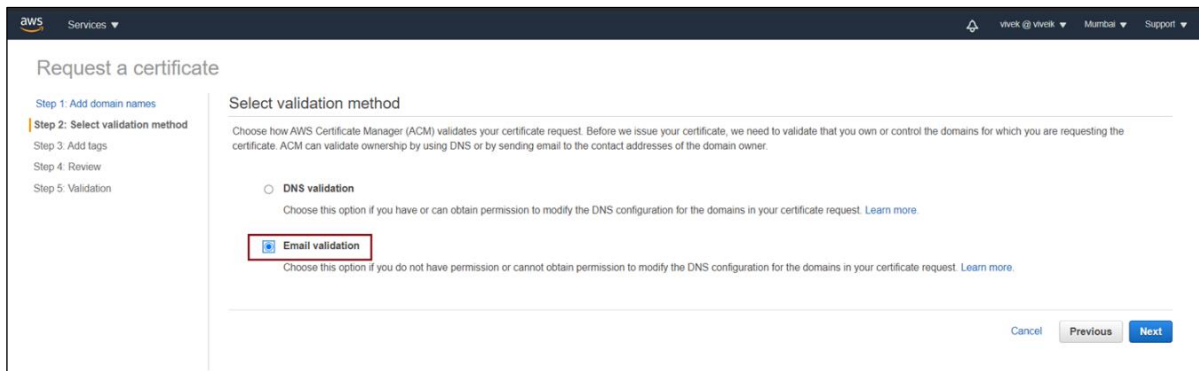


Figure 2.39

7. On the Add Tags, click **Review.**

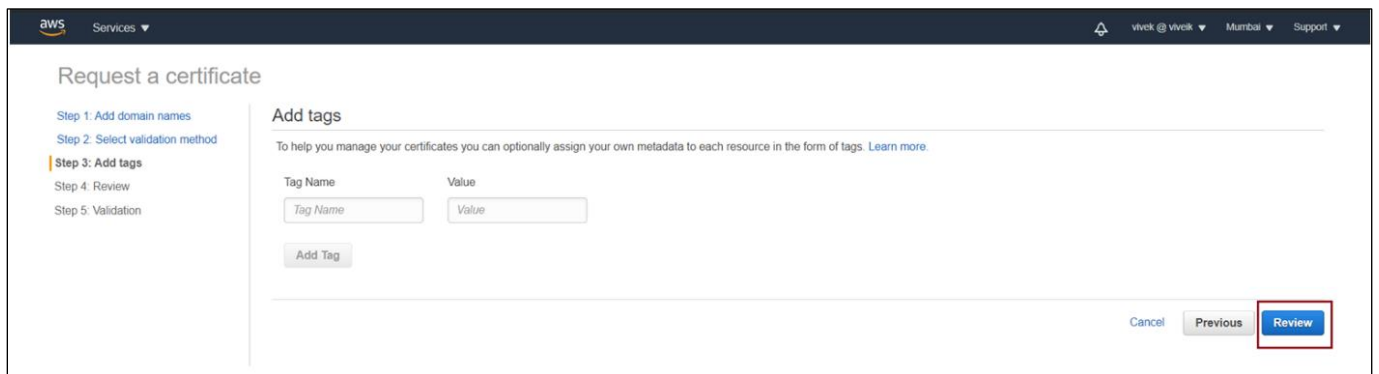


Figure 2.40

8. Click **Confirm and request**.

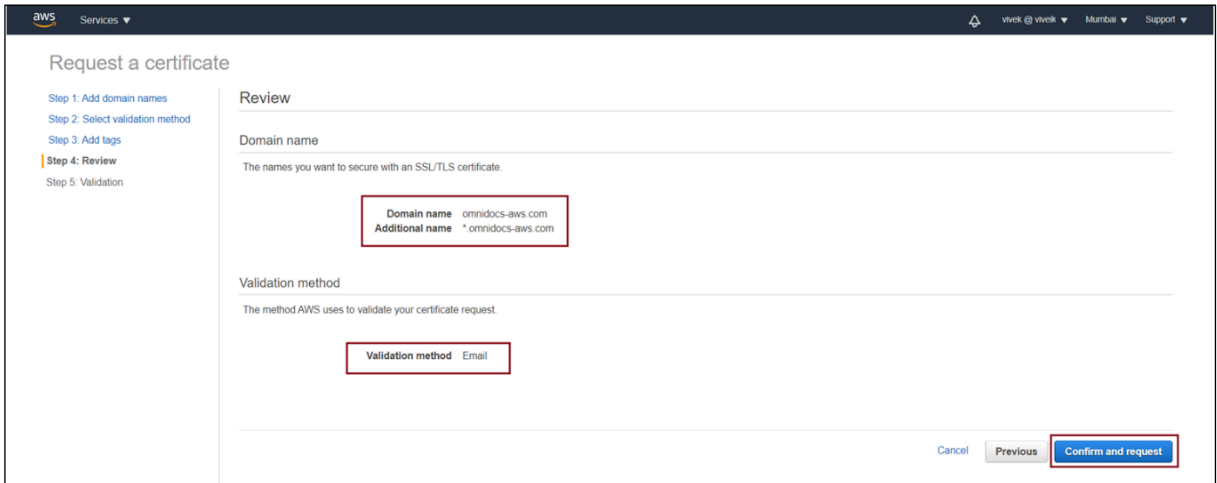


Figure 2.41

9. Click **Continue**. An approval mail is sent to the below recipients of the registered domain.

- Registrant Contact
- Administrative Contact
- Technical Contact

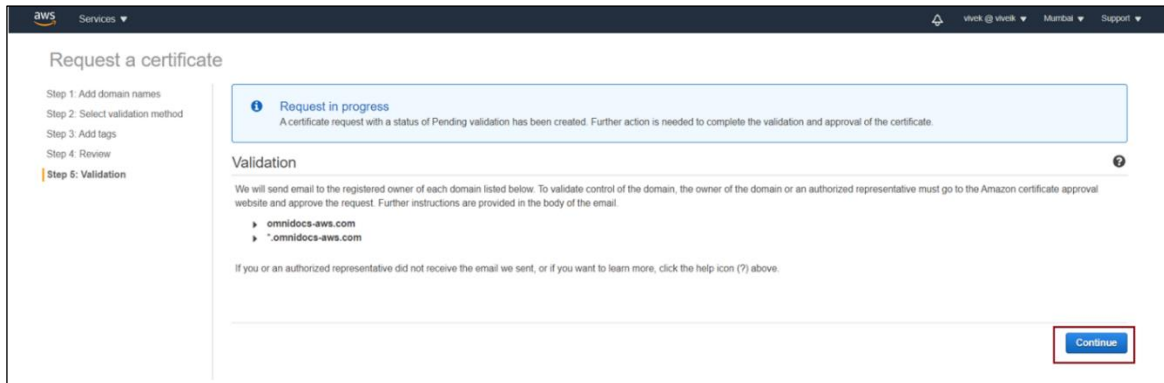


Figure 2.42

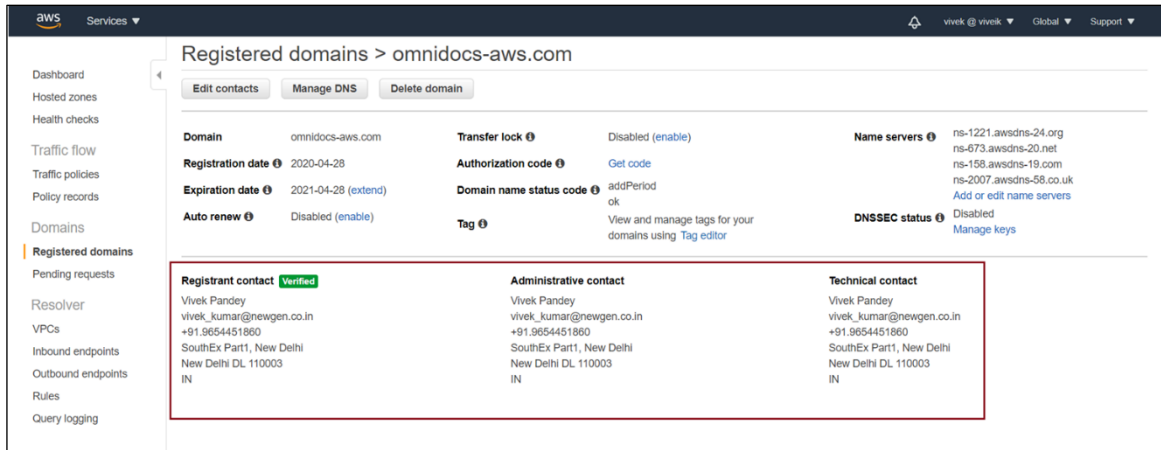


Figure 2.43

10. Once all the recipients approve the certificate, the certificate status gets changed from **Pending Validation** to **issue**.

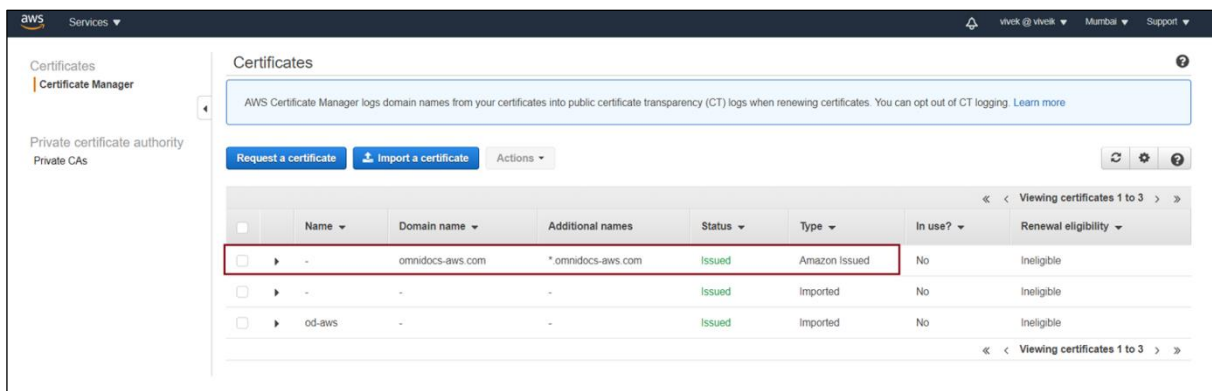


Figure 2.44

2.21 Cluster AutoScaler

The Cluster AutoScaler requires additional IAM and resource tagging considerations that are given in the following subsections:

2.21.1 Node Group IAM Policy

Create an IAM policy with the following JSON scripts and attach it to the Worker Node's IAM Role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",

```

```

    "autoscaling:DescribeAutoScalingInstances",
    "autoscaling:DescribeLaunchConfigurations",
    "autoscaling:DescribeTags",
    "autoscaling:SetDesiredCapacity",
    "autoscaling:TerminateInstanceInAutoScalingGroup",
    "ec2:DescribeLaunchTemplateVersions"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
]
}

```

2.21.2 Updating auto scaling group

Perform the below steps to update the Auto Scaling Group:

1. Create an **AMI** of any worker node.
2. Go to the **Auto Scaling Groups** and click the created autoscaling group for this EKS Cluster.

Name	Launch template/configuration	Instan...	Status	Desired capacity	M...	M...	Avail
EKSCluster-NodeGroup-1XY272UXQVEIE	NodeLaunchTemplate_3jfx1a4czh1W Ve	2	-	2	2	3	ap-sc
ecmsuit-NodeGroup-CANXJS8YBO3I	ecmsuit-NodeLaunchConfig-RVRM7...	2	-	2	1	5	ap-sc

Figure 2.36

3. Click the attached **Launch Template**.

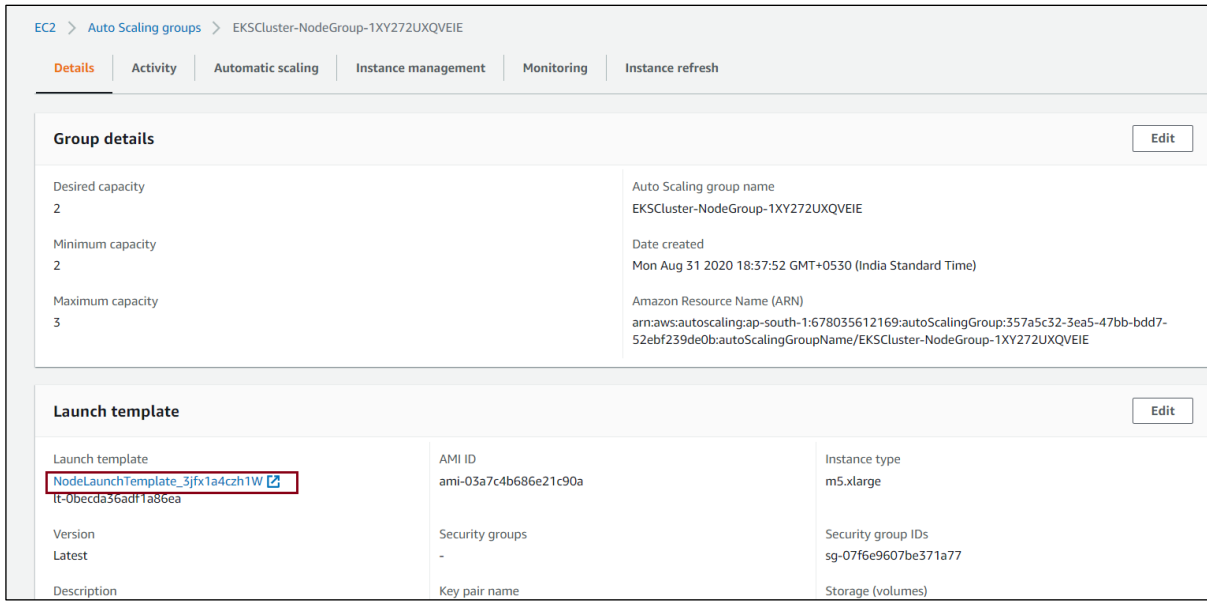


Figure 2.37

4. Select the created **Launch Template** and select the **Modify Template (Create new version)** option from the **Actions** menu.

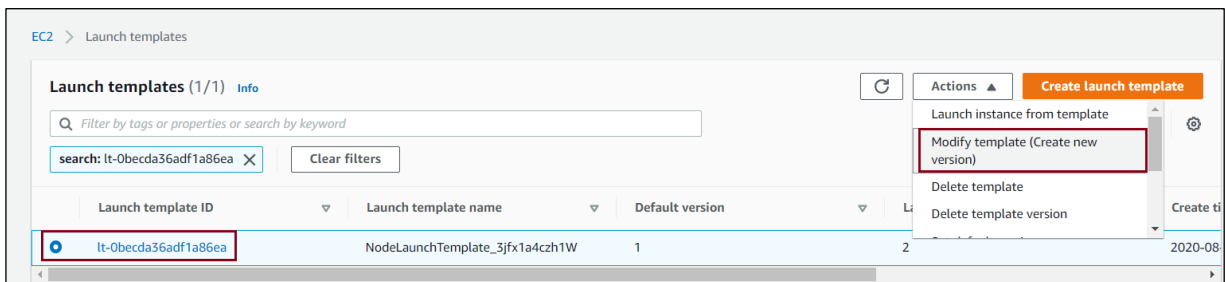


Figure 2.47

5. Select the created AMI and click **Create template version**.
6. Go back to the Auto Scaling Group and click **Edit** given in the right panel.



Figure 2.48

7. Select the **Latest** version and click **Update**.

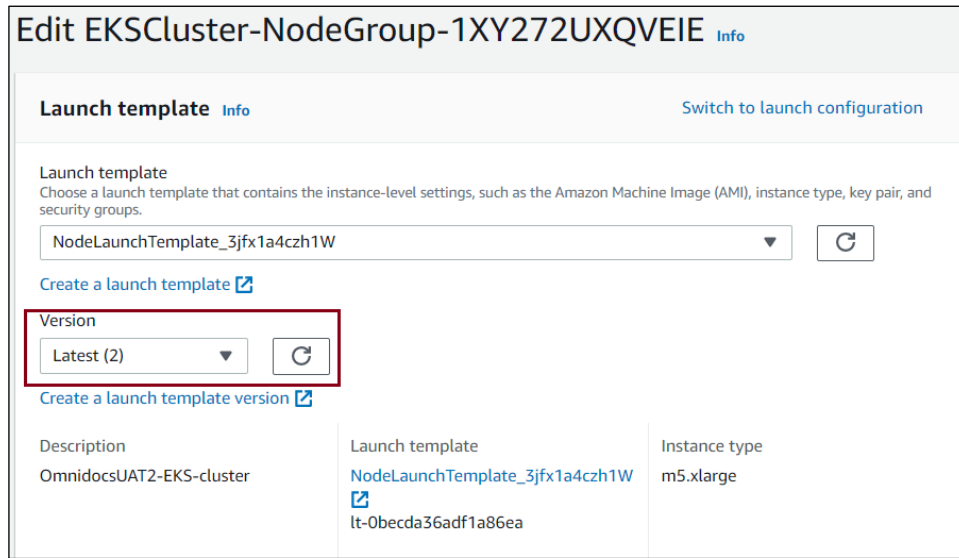


Figure 2.49

8. The Cluster AutoScaler requires the following tags on your node group that is Auto Scaling groups so it can be auto-discovered.

Key	Value
k8s.io/cluster-autoscaler/<cluster-name>	owned
k8s.io/cluster-autoscaler/enabled	True

2.21.3 Deploying cluster autoscaler

Perform the below steps to deploy the Cluster AutoScaler:

1. Deploy the Cluster Autoscaler to your cluster using the below command:

```
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/autoscaler/master/cluster-
autoscaler/cloudprovider/aws/examples/cluster-autoscaler-autodiscover.yaml
```

2. Add the cluster-autoscaler.kubernetes.io/safe-to-evict annotation to the deployment using the below command:

```
kubectl -n kube-system annotate deployment.apps/cluster-autoscaler cluster-
autoscaler.kubernetes.io/safe-to-evict="false"
```

3. Edit the Cluster AutoScaler deployment using the below command:

```
kubectl -n kube-system edit deployment.apps/cluster-autoscaler
```

- Edit the cluster-autoscaler container command to replace <YOUR CLUSTER NAME> with your cluster's name, and add the following options:

- --balance-similar-node-groups
- --skip-nodes-with-system-pods=false

4. Save and close the file to apply the changes.

For Example:

```
spec:
  containers:
    - command:
      - ./cluster-autoscaler
      - --v=4
      - --stderrthreshold=info
      - --cloud-provider=aws
      - --skip-nodes-with-local-storage=false
      - --expander=least-waste
      - --node-group-auto-discovery=asg:tag=k8s.io/cluster-
autoscaler/enabled,k8s.io/cluster-autoscaler/<YOUR CLUSTER NAME>
      - --balance-similar-node-groups
      - --skip-nodes-with-system-pods=false
```

5. Open the below URL in a web browser and find the latest Cluster AutoScaler version that matches your cluster's Kubernetes major and minor versions.

For example, if your cluster's Kubernetes version is 1.21 then find the latest Cluster AutoScaler release that begins with 1.21. Record the semantic version number (1.21.n) for that release to use in the further step.

<https://github.com/kubernetes/autoscaler/releases>.

6. Set the Cluster AutoScaler image tag to the version that you have recorded.

7. Use the below command and replace 1.21.n with your value.

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
autoscaler=us.gcr.io/k8s-artifacts-prod/autoscaling/cluster-
autoscaler:v1.21.n
```

2.21.4 Viewing cluster autoscaler logs

Once you have deployed the Cluster Autoscaler, you can view the logs and verify that it is monitoring your cluster load.

You can view your Cluster AutoScaler logs using the below command:

```
kubectl -n kube-system logs -f deployment.apps/cluster-autoscaler
```

2.22 Setting up CloudWatch container insights

Container Insights is a fully managed CloudWatch service and is used to collect, aggregate, and summarize metrics and logs of containerized applications deployed on ECS or EKS service. The metrics include the utilization of resources such as CPU, memory, disk, and network.

Container insights also provide diagnostic information, such as container restart failures, to help you isolate issues and resolve them quickly. The metrics that Container Insights collects are available in CloudWatch automatic dashboards

Perform the below steps to set up the CloudWatch container insights:

1. Attach the below policy to the IAM role of your worker nodes:
 - CloudWatchAgentServerPolicy
2. Execute the below command to deploy container insights on the EKS cluster:

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/{{cluster_name}}/CLUSTER_NAME/;s/{{region_name}}/REGION_NAME/" | kubectl apply -f -
```

In the above command, change the **CLUSTER_NAME** and **REGION_NAME** as required.

3 Deploying OmniDocs and RMS containers

This chapter describes the deployment of OmniDocs and RMS containers on AWS. Refer to the below sub-sections for procedural details.

3.1 Prerequisites

To deploy OmniDocs and RMS containers, the AWS Elastic Kubernetes Service must already be configured, and its Worker nodes must be in the ready state.

NOTE:

Refer to the [Configuration of AWS Kubernetes Cluster](#) for the configuration of AWS Elastic Kubernetes Service.

3.2 Deliverables

Newgen has isolated the product suite into multiple Docker containers to enable the independent scalability of each Docker container. This separation is done based on the product's usability. At a broad level, Web components and EJB components are isolated for deployment in separate container instances. Web components is deployed on the underlying web server JBoss WebServer 5.7.x. EJB components is deployed on the underlying application server JBoss EAP 7.4.x. Newgen has released multiple Docker images for the different product suites along with some configuration files for data persistence, YAML files for deployment, and some documentation for end-to-end configurations and deployments.

The followings are the list of deliverables:

- [Docker Images](#)
- [Configuration Files](#)
- [YAML Files](#)

3.2.1 Docker Images

The following 7 Docker images are delivered for the initial product deployment:

- OmniDocs+RMS Web Components
- OmniDocs Web Service Components
- OmniDocs+RMS EJB Components
- OmniDocs Add-on Services (Wrapper, AlarmMailer, Scheduler, ThumbnailManager and LDAP)
- Docker Images for EasySearch (Only one is required based on the infrastructure availability)
 - EasySearch (Apache Manifold and ElasticSearch [freeware software])
Or
 - EasySearch (Apache Manifold only [freeware software] with AWS managed ElasticSearch)
- Text Extraction Manager or Full-Text Search (TEM/FTS)
- OmniScan Web Components
- RMS SharePoint Adapter
- Messaging Service

NOTE:

These Docker images can be delivered to a private Docker repository like AWS ECR (Elastic Container Registry) or in the form of compressed files that can be shared over the FTP or similar kind of media.

3.2.2 Configuration files

Configuration files are dynamic in nature and data is written at runtime. Database details in configuration files such as *Server.xml* and *standalone.xml* are written at runtime. These types of files must be kept outside the container to persist the data. Here, AWS EFS is used to persist configuration files.

The following configuration files are shared for OmniDocs and RMS Docker images:

- OmniDocs11.0Web
- OmniDocs11.0Ejb
- OD11.0Services
- EasySearch11.0
- TEM11.0
- OmniscanWeb6.0
- RMS SharePoint Adapter
- MessagingService

3.2.3 YAML files

YAML files stands for “YAML Ain’t Markup Language”. It is a human-readable object configuration file that is used to deploy and manage the objects on the Kubernetes cluster. In other words, it is a manifest file that contains the deployment descriptor of Kubernetes containers. You can execute YAML files using “kubectl apply -f <YAMLFile>” or use these files in AWS CodePipeline to deploy the containers.

The following configuration files has shared for OmniDocs and RMS Docker images:

- OmniDocs11.0Web.yml
- OmniDocs11.0Web_Services.yml
- OmniDocs11.0EJB.yml
- OmniDocs11.0Services.yml
- EasySearch11.0.yml
- EasySearch11.0_ApacheOnly.yml
- OmniDocs11.0MessagingService.yml
- TEM11.0.yml
- OmniScanWeb6.0.yml
- AWS_ALB-IngressController.yml
- buildspec.yml
- buildspec_EasySearch.yml
- RMSSharePointAdapter.yml

Here’s an example of a YAML file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  #Name should not be more than 13 letters
  name: od110web
spec:
  replicas: 1
  selector:
    matchLabels:
      name: od110web
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    metadata:
      labels:
        name: od110web
    spec:
      containers:
        - name: od110web
          image: REGISTRY_ID.dkr.ecr.REGION.amazonaws.com/REPOSITORY_NAME:IMAGE_TAG
          imagePullPolicy: Always
          securityContext:
            runAsNonRoot: true
          resources:
            requests:
              memory: 2048Mi
              cpu: 800m
            limits:
              memory: 3072Mi
              cpu: 1000m
          volumeMounts:
            - name: omnidocsconfig
```

Figure 3.1

AWS_ALB-IngressController.yml is used for the ingress controller. An ingress controller is an object running inside the Kubernetes cluster that is used to manage the host-based routing rules. For

example, you can set the host-based routing rules like if the URL is *omnidocs.newgendocker.com* then the ingress controller redirects the user request to OmniDocs+RMS WEB containers and if the URL is *omniscan.newgendocker.com* then it redirects the user request to the OmniScanWEB containers.

Buildspec.yml is used in AWS CodePipeline for deploying the containers on AWS Elastic Kubernetes Service.

NOTE:

You can store the above YAML files in AWS CodeCommit Repo that is used by AWS CodePipeline.

3.3 Changes in Product's YAML files

The changes in the Product's YAML files are:

- **Name:** In the *OmniDocs11.0Web.yml* file, **od110web** is given as the default name of Kubernetes objects - deployment, replicas, container, and service. You can change this name as per your choice. While changing the name, ensure that this name is not more than 13 letters in length and must contain small letters only.

For example,

```
apiVersion: apps/v1
kind: Deployment
metadata:
  #Name should not be more than 13 letters
  name: od110web
spec:
  replicas: 1
  selector:
    matchLabels:
      name: od110web
  strategy:
    type: RollingUpdate
  rollingUpdate:
    maxSurge: 1
    maxUnavailable: 0
  template:
    metadata:
      labels:
        name: od110web
    spec:
      containers:
        - name: od110web
```

Figure 3.2

- **Replica:** In the *OmniDocs11.0Web.yml* file, the default replica is given as **1**. That means only one container is created after the deployment. You can increase this number as per our choice.
- **Image:** In the *OmniDocs11.0Web.yml* file, update the **image** location. By default, the below value is given:

```
image: REGISTRY_ID.dkr.ecr.REGION.amazonaws.com/REPOSITORY_NAME:IMAGE_TAG
```

Here:

- **REGISTRY_ID** is the AWS account ID where AWS ECR (Elastic Container Registry) is created.
- **REGION** is the AWS Region where AWS ECR (Elastic Container Registry) is created.
- **REPOSITORY_NAME** is the Omnidocs WEB Docker image name.
- **IMAGE_TAG** is a Docker image's tag name that you want to deploy.

As AWS CodePipeline is used to deploy Docker images, you do not need to update the image in the YAML file as AWS CodePipeline updates these values at runtime.

- **SecurityContext:** In the *OmniDocs11.0Web.yml* file, **SecurityContext [runAsNonRoot: true]** is defined. It means the OmniDocs11.0Web container can never be run with root privileges. If any container tries to run with the root user, then Kubernetes stops its deployments.

```
securityContext:
  runAsNonRoot: true
```

Figure 3.3

- **Resource request and limit:** In the *OmniDocs11.0Web.yml* file, resource request and resource limit parameters are defined. The **request** parameter specifies the minimum required resources to run the particular container and the **limit** parameter specifies the maximum resource limit that a container can use. In other words, a running container is not allowed to use more than the resource limit you set.

```
requests:
  memory: 1024Mi
  cpu: 800m
limits:
  memory: 2048Mi
  cpu: 1000m
```

Figure 3.4

Here, 1000m CPU = 1 Core CPU

The above-specified limit is the minimum required resource to run a container. If users are increasing, then you must increase the limit range accordingly.

- **VolumeMounts and Volume:** Volume mounts and volumes are used to persist the data outside the container so that whenever the container terminates due to any reason our data is always persisted. In the *OmniDocs11.0Web.yml* file, we have persisted configuration files or folders and log files.

```
volumeMounts:
- name: omnidocsconfig
  mountPath: /Newgen/jws-5.7/tomcat/bin/Newgen/NGConfig
- name: omniscanweb
  mountPath: /Newgen/jws-5.7/tomcat/bin/omniscanweb
- name: context-xml
  mountPath: /Newgen/jws-5.7/tomcat/conf/context.xml
- name: web-xml
  mountPath: /Newgen/jws-5.7/tomcat/conf/web.xml
- name: jboss-ejb-client-properties
  mountPath: /Newgen/jws-5.7/tomcat/lib/jboss-ejb-client.properties
- name: tomcat-logs
  mountPath: /Newgen/jws-5.7/tomcat/logs
  subPathExpr: $(POD_NAME)
- name: omnidocs-logs
  mountPath: /Newgen/jws-5.7/tomcat/bin/Newgen/NGLogs
  subPathExpr: $(POD_NAME)
- name: ng-temp
  mountPath: /Newgen/jws-5.7/tomcat/bin/Newgen/NGTemp
  subPathExpr: $(POD_NAME)
- name: system-report
  mountPath: /Newgen/jboss-eap-7.4/bin/SystemReports
```

Figure 3.5

In volumeMounts, **mountPath** is a path inside the container that is being mounted. Here, mountPath cannot be changed as this structure is predefined in a Docker container. The **name** is a user-defined name that must be matched with the name specified in volumes.

```

volumes:
- name: omnidocsconfig
  hostPath:
    path: /mnt/efs/OmniDocs11.0SP1/OmniDocs11.0Web/Newgen/NGConfig
    type: DirectoryOrCreate
- name: omniscanweb
  hostPath:
    path: /mnt/efs/OmniDocs11.0SP1/OmniDocs11.0Web/omniscanweb
    type: DirectoryOrCreate
- name: context-xml
  hostPath:
    path: /mnt/efs/OmniDocs11.0SP1/OmniDocs11.0Web/context.xml
    type: File
- name: web-xml
  hostPath:
    path: /mnt/efs/OmniDocs11.0SP1_Latest/OmniDocs11.0Web/web.xml
    type: File
- name: jboss-ejb-client-properties
  hostPath:
    path: /mnt/efs/OmniDocs11.0SP1/OmniDocs11.0Web/jboss-ejb-client.properties
    type: File
- name: tomcat-logs
  hostPath:
    path: /mnt/efs/OmniDocs11.0SP1/ProductLogs/odl10web-AWS-Release-Build_Pipeline_Number/tomcat_logs
    type: DirectoryOrCreate
- name: omnidocs-logs
  hostPath:
    path: /mnt/efs/OmniDocs11.0SP1/ProductLogs/odl10web-AWS-Release-Build_Pipeline_Number/NGLogs
    type: DirectoryOrCreate
- name: ng-temp
  hostPath:
    path: /mnt/efs/OmniDocs11.0SP1/ProductLogs/odl10web-AWS-Release-Build_Pipeline_Number/NGTemp
    type: DirectoryOrCreate
- name: system-report
  hostPath:
    path: /mnt/efs/OmniDocs11.0SP1/SystemReports
    type: DirectoryOrCreate

```

Figure 3.6

In volumes, the hostPath mounts a file or directory from the worker node's file system into the container. This path must exist in the worker node's file system. The hostPath can be a file path or folder path, you just need to define its type whether it is a File or Directory. In this YAML file, some hostPath contains dynamic values whose value gets updated at runtime.

- **Ports:** In the *OmniDocs11.0Web.yml* file, containerPort is specified as **8080**. That means only port 8080 is exposed outside the container and no other port is accessible from outside.

```

ports:
- name: http
  containerPort: 8080

```

Figure 3.7

- **ReadinessProbe:** The kubelet uses the readiness probe to know when a container is ready to start accepting traffic. Until unless the readiness probe is not succeeded, the container does not serve the user requests.

```

readinessProbe:
  httpGet:
    path: /omnidocs/web
    port: 8080
  # Intial delay is set to a high value to have a better
  # visibility of the ramped deployment
  initialDelaySeconds: 30
  periodSeconds: 5

```

Figure 3.8

Here, until unless `ip:port/omnidocs/web` is not accessible through a browser, the container does not accept the user request.

- **LivenessProbe:** Docker containers have healing power, if an application running inside the container gets down due to any reason or becomes unresponsive then Kubernetes restarts the application automatically inside the container. This feature is known as **LivenessProbe** in Kubernetes.

```

livenessProbe:
  httpGet:
    path: /omnidocs/web
    port: 8080
  initialDelaySeconds: 300
  failureThreshold: 5
  periodSeconds: 10

```

Figure 3.9

- **Environment variable:** In the `OmniDocs11.0Web.yml` file, the `JAVA_OPTS` parameter is defined that is used to set the heap size in the WEB container dynamically.

```

- name: JAVA_OPTS
  value: "-XX:+UseContainerSupport -XX:+DisableExplicitGC -XX:InitialRAMPercentage=75.0
        -XX:MaxRAMPercentage=75.0"

```

Figure 3.10

Ensure '`-XX:MaxRAMPercentage`' is a parameter through which you can provide the available memory to use as a max heap size to JVM. In the above example, 75% of total memory is allocated as heap size.

NOTE:

You can use the above guidelines to update other YAML Files that are as follows:

- OmniDocs11.0Web_Services.yml
 - OmniDocs11.0EJB.yml
 - OmniDocs11.0Services.yml
 - EasySearch11.0.yml
 - EasySearch11.0_ApacheOnly.yml
 - TEM11.0.yml
 - OmniScanWeb6.0.yml
 - RMSSharePointAdapter.yml
 - OmniDocs11.0MessagingService.yml
-

3.4 Changes in AWS load balancer controller YAML files

The AWS Load Balancer Controller creates an Application Load Balancer and routes the incoming requests to the target Kubernetes services according to the host-based routing rules. Host-based routing is a capability of ALB that redirects the user requests to the right service based on the request-host header

For example, you can set the rules as below:

- If URL is *omnidocs.newgendocker.com*, then redirect to the OmniDocs+RMS Web container.
- If URL is *omniscan.newgendocker.com*, then redirect to the OmniScanWeb container.

NOTE:

To support the host-based routing, we must register a domain, create a new RecordSet in Route-53 for each host-path and generate the SSL certificate against the registered domain. Refer to the [Configuration of AWS Kubernetes Cluster](#) section for the configuration of AWS ALB Ingress Controller, Route-53, and Certificate Manager.

- Once AWS ALB Ingress is configured, RecordSet is created in Route-53, and an SSL certificate is generated. You must deploy the Ingress controller along with its ruleset using the YAML file.
- Before deploying the same you need to update some settings in the *AWS_ALB-IngressController.yml* file.
- To access your application using both the HTTP and HTTPS protocols, ensure that the below annotation is added:
alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

```
annotations:
  kubernetes.io/ingress.class: alb
  alb.ingress.kubernetes.io/scheme: internet-facing
  alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'
  alb.ingress.kubernetes.io/certificate-arn: <AWS Certificate ARN>
  alb.ingress.kubernetes.io/ssl-policy: ELBSecurityPolicy-TLS-1-2-Ext-2018-06
```

Figure 3.11

- To access your application using the HTTPS protocol only, update the annotation as below:
alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}]'
- Update the **SSL certificate ARN** generated from AWS Certificate Manager.
- If you want to open both the HTTP and HTTPS protocols and whenever the calls come to the HTTP, it redirects to HTTPS, then make sure the below annotation is added:
alb.ingress.kubernetes.io/ssl-redirect: '443'
- Update the subnets and security groups associated with the Kubernetes Worker nodes.
alb.ingress.kubernetes.io/subnets: subnet-0a37ea6be439259a0, subnet-0e3a0a6a7d3887eca, subnet-09ee1bc2c393de555
alb.ingress.kubernetes.io/security-groups: sg-0f4f4504892233a90
- In the above step, there are multiple host-based rules defined.
 - **omnidocs.newgendoctr.com [Specified as a record set in Route-53]**
If the host URL is *omnidocs.newgendoctr.com*, then it redirects the user request to the **od110web** container's service which is running on port 8080. Here, od110web is the name of the OmniDocs+RMS Web container.
 - **omnidocswebservices.newgendoctr.com [Specified as a record set in Route-53]**
If the host URL is *omnidocswebservices.newgendoctr.com*, then it redirects the user request to the **od110websvc** container's service which is running on port 8080. Here, od110websvc is the name of the OmniDocs Web Service container.
 - **omnidocsconsole.newgendoctr.com [Specified as a record set in Route-53]**
If the host URL is *omnidocsconsole.newgendoctr.com*, then it redirects the user request to the **od110ejb** container's service which is running on port 9990. Here, od110ejb is the name of the OmniDocs+RMS EJB container.
 - **apachemanifold.newgendoctr.com [Specified as a record set in Route-53]**
If the host URL is *apachemanifold.newgendoctr.com* then it redirects the user request to the easysearch11 container's service which is running on port 8345. Here, easysearch11 is the name of the EasySearch container.
 - **omniscan.newgendoctr.com [Specified as a record set in Route-53]**
If the host URL is *omniscan.newgendoctr.com*, then it redirects the user request to the **omniscan web** container's service which is running on port 8080. Here, omniscanweb is the name of the OmniScan Web container.

➤ **Odmsgservice.newgendocker.com [Specified as a record set in Route-53]**

If the host URL is odmsgservice.newgendocker.com, then it redirects the user request to the odmessage container's service which is running on port 8080. Here, **odmsgservice** is the name of the OD Message Service container.

- In this YAML file, you can change the host URL, ServiceName, ServicePort, and the name **name: alb-ingress** as per your choice.
- After making the required changes as per our choice, you can deploy the Ingress controller by executing this YAML file using the below command:

```
kubectl apply -f AWS_ALB-IngressController.yml
```

NOTE:

To execute the above command, kubectl must be configured on your local server. Refer to the [Configuration of AWS Kubernetes Cluster](#) section to run kubectl from your local machine.

3.5 Changes in configuration files

The section includes the following sub-sections:

3.5.1 Prerequisites

The Prerequisites are as follows:

- All the configuration files must be copied to the worker node's hostPath location defined in the YAML files and that hostPath must be mounted to the AWS EFS.
- The RedisCache server is already configured.
- A valid wildcard certificate and the domain are already configured.
- SSL or TLS must be configured for the Application's URL.

NOTE:

- Refer to the [Mount EFS to Worker Nodes](#) section to mount the EFS to hostPath.
 - By default, the applications run on HTTPS only. If you want to run with HTTP protocol then some additional setting is required. For more information, refer to the Docker Troubleshooting Guide.
-

3.5.2 OmniDocs+RMS Web Changes

The changes in OmniDocs+RMS Web are:

- Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.OEJB.yml* file] in *NGOClientData.xml*, *RMAAppConfig.xml* and *RMClientData.xml* files in between the `<endPointURL></endPointURL>` tags located inside the *OmniDocs11.0Web\Newgen\NGConfig\ngdbini* folder at the mapped location on the Worker node.

```

<ClientInfo>
  <ProviderUrl></ProviderUrl>
  <jndiServerName></jndiServerName>
  <jndiServerPort></jndiServerPort>
  <ContextSuffix></ContextSuffix>
  <WildFlyUserName></WildFlyUserName>
  <WildFlyPassword></WildFlyPassword>
  <JndiContextFactory></JndiContextFactory>
  <ClientLookUpName>ejb:omnidocs_ejb/omnidocs_ejb/NGOClientServiceHandlerBean!com.newgen.omni.jts.txn.NGOClientServiceHandlerHome</ClientLookUpName>
  <AdminLookUpName>ejb:omnidocs_ejb/omnidocs_ejb/NGOAdminServiceHandlerBean!com.newgen.omni.jts.txn.NGOAdminServiceHandlerHome</AdminLookUpName>
  <UrlPackagePrefix></UrlPackagePrefix>
  <endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>
  <xmlParamName>InputXML</xmlParamName>
  <HeaderKey></HeaderKey>
  <HeaderValue></HeaderValue>
</ClientInfo>

```

Figure 3.12

Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.

- Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.0EJB.yml* file] in *IS.ini* file in between the `<endPointURL ></endPointURL >` tags located inside the *OmniDocs11.0Web\Newgen\NGConfig* folder at the mapped location on the Worker node. For example,

```

<LogPath>Replication.log</LogPath>
<SMSTimeOut>60000</SMSTimeOut>
<SMSReadInterval>30000</SMSReadInterval>
<SMSRetryCount>5</SMSRetryCount>
<SMSGenerateLog>true</SMSGenerateLog>
<IsJNDI>true</IsJNDI>
<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>
<xmlParamName>InputXML</xmlParamName>
<HeaderKey></HeaderKey>
<HeaderValue></HeaderValue>
<ProviderUrl></ProviderUrl>
<jndiServerName></jndiServerName>
<jndiServerPort></jndiServerPort>

```

Figure 3.13

- Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.0EJB.yml* file] in *jboss-ejb-client.properties* file located inside the *OmniDocs11.0Web* folder at the mapped location on the Worker node. For example,

```
remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=false
remote.connections=default
remote.connection.default.host=od110ejb
remote.connection.default.port = 8080
remote.connection.default.username=XXXXXXXXXX
remote.connection.default.password=XXXXXXXXXX
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOANONYMOUS=false
```

Figure 3.14

Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.

- Update the **RMSEnabled=Y** in *eworkstyle.ini* file located at *OmniDocs11.0Web/Newgen/NGConfig/ngdbini/odwebini* folder at mapped location.
- Update the AWS Elastic Redis cache's configuration endpoint in *redisson.yaml* file against the *singleServerConfig* or *clusterServersConfig*. If redis cache is SSL enabled, then use the *redis://<endpoint url>:port* and if SSL is not enabled, then use *redis://<endpoint url>:port*. This file *redisson.yaml* is located inside the *OmniDocs11.0Web* folder at the mapped location on the Worker node.

```

---
singleServerConfig:
  idleConnectionTimeout: 10000
  connectTimeout: 10000
  timeout: 3000
  retryAttempts: 3
  retryInterval: 1500
  password: null
  subscriptionsPerConnection: 5
  clientName: null
  address: "redis://[REDACTED]:6379"
  subscriptionConnectionMinimumIdleSize: 1
  subscriptionConnectionPoolSize: 50
  connectionMinimumIdleSize: 24
  connectionPoolSize: 64
  database: 0
  dnsMonitoringInterval: 5000
threads: 16
nettyThreads: 32
codec: !<org.redisson.codec.MarshallingCodec> {}
transportMode: "NIO"

#Reference: https://github.com/redisson/redisson/wiki/2.-Configuration#26-single-instance-mode

#CLUSTER ---
#CLUSTER clusterServersConfig:
#CLUSTER  idleConnectionTimeout: 10000
#CLUSTER  connectTimeout: 10000
#CLUSTER  timeout: 3000
#CLUSTER  retryAttempts: 3
#CLUSTER  retryInterval: 1500
#CLUSTER  failedSlaveReconnectionInterval: 3000
#CLUSTER  failedSlaveCheckInterval: 60000
#CLUSTER  password: null
#CLUSTER  subscriptionsPerConnection: 5
#CLUSTER  clientName: null
#CLUSTER  loadBalancer: !<org.redisson.connection.balancer.RoundRobinLoadBalancer> {}
#CLUSTER  subscriptionConnectionMinimumIdleSize: 1

```

Figure 3.15

- Open the *web.xml* file in edit mode located inside the OmniDocs11.0Web folder at the mapped location on the Worker node.
- Search for filter **httpHeaderSecurity** and update the `<param-value></param-value>` tag's value with OmniDocs URL without context name against `<param-name> antiClickJackingUri</param-name>`.

```

<filter>
  <filter-name>httpHeaderSecurity</filter-name>
  <filter-class>org.apache.catalina.filters.HttpHeaderSecurityFilter</filter-class>
  <async-supported>true</async-supported>
  <init-param>
    <param-name>antiClickJackingOption</param-name>
    <param-value>ALLOW-FROM</param-value>
  </init-param>
  <init-param>
    <param-name>antiClickJackingUri</param-name>
    <param-value>omnidocs.newgendocker.com</param-value>
  </init-param>
</filter>

```

Figure 3.16

- Open the `web_svc.xml` file in edit mode located inside the `OmniDocs11.0Web` folder at the mapped location on the Worker node.
- Search for `filter-class <filter-class>org.apache.catalina.filters.CorsFilter</filter-class>` and update the `<param-value></param-value>` tag's value with OmniDocs URL with protocol against `<param-name> antiClickJackingUri</param-name>`

```

<filter>
  <filter-name>CorsFilter</filter-name>
  <filter-class>org.apache.catalina.filters.CorsFilter</filter-class>
  <init-param>
    <param-name>cors.allowed.origins</param-name>
    <param-value>https://omnidocs.bpmsncloud.co.in</param-value>
  </init-param>
  <init-param>
    <param-name>cors.allowed.methods</param-name>
    <param-value>GET, POST, HEAD, OPTIONS, PUT</param-value>
  </init-param>
  <init-param>
    <param-name>cors.allowed.headers</param-name>
    <param-value>Content-Type, X-Requested-With, accept, Origin, Access-Control-Request
  </param-value>
  </init-param>
  <init-param>
    <param-name>cors.exposed.headers</param-name>
    <param-value>Access-Control-Allow-Origin</param-value>
  </init-param>
</filter>

```

Figure 3.17

3.5.3 Wrapper changes

Update the OmniDocs+RMS EJB container name [Defined in `OmniDocs11.0EJB.yml` file] in `NGOClientData.xml`, `RMAppConfig.xml` and `RMClientData.xml` files in between the `<endPointURL></endPointURL>` tags file located inside the `OD11.0Services/Wrapper/ngdbini` folder at the mapped location on the Worker node.

```

<ClientInfo>
  <ProviderUrl></ProviderUrl>
  <jndiServerName></jndiServerName>
  <jndiServerPort></jndiServerPort>
  <ContextSuffix></ContextSuffix>
  <WildFlyUserName></WildFlyUserName>
  <WildFlyPassword></WildFlyPassword>
  <JndiContextFactory></JndiContextFactory>
  <ClientLookUpName>ejb:omnidocs_ejb/omnidocs_ejb/NGOClientServiceHandlerBean!com.newgen.omni.jts.txn.NGOClientServiceHandlerHome</ClientLookUpName>
  <AdminLookUpName>ejb:omnidocs_ejb/omnidocs_ejb/NGOAdminServiceHandlerBean!com.newgen.omni.jts.txn.NGOAdminServiceHandlerHome</AdminLookUpName>
  <UrlPackagePrefix></UrlPackagePrefix>
  <endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>
  <xmlParamName>InputXML</xmlParamName>
  <HeaderKey></HeaderKey>
  <HeaderValue></HeaderValue>
</ClientInfo>

```

Figure 3.18

Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.

3.5.4 AlarmMailer changes

Prerequisite:

- The cabinet is created and associated with the running containers. If the cabinet is not created, then refer to [Cabinet and Data Source Creation](#) section.
- SecretManager policy must be created and attached to the Worker Node IAM role. If not created, then refer to the Creating secret manager policy and secrets section.

The changes in AlarmMailer are as follows:

1. Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.0EJB.yml* file] in *IS.ini* in between the `<endPointURL></endPointURL>` tags file located inside the *OD11.0Services* or *AlarmMailer* folder at the mapped location on the Worker node.

For example,

```
<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>
```

Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.

2. Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.0EJB.yml* file] in *NGOClientData.xml* in between the `<endPointURL></endPointURL>` tags file located inside the *OD11.0Services/AlarmMailer/ngdbini* folder at the mapped location on the Worker node.

For example,

```
<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>
```

Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.

3. Update the below settings in the *Alarm.ini* file located inside the *OD11.0Services/AlarmMailer* folder at the mapped location on the Worker node.

- i. Update the OmniDocs URL without context name in between the `<webservername></webservername>` tag.

For example, `<webservername>omnidocs.newgendocker.com</webservername>`

Here, *omnidocs.newgendocker.com* is the host path defined in the *AWS_ALB-IngressController.yml* file.

- ii. Leave the `WebServerPort` as blank if OmniDocsWEB URL does not contain a port.

For example, `<webserverport></webserverport>`

- iii. Update the OmniDocs cabinet name in between `<cabinetname></cabinetname>` tag.

For example, `<cabinetname>ecmsuite</cabinetname>`

Here, **ecmsuite** is the OmniDocs cabinet name that gets created.

- iv. Update the OmniDocs supervisor group's user in between the `<user></user>` tag.

For example, `<user>supervisor</user>`

- v. Update the OmniDocs supervisor group's user password in between the `<password></password>` tag. Ensure that this password must be in an encrypted format.

For example, <password>X-D;U:T-C;P-C;p5-C;b:d:</password>

4. If Secret Vault is used to store sensitive information, then refer below points:
 - i. Update the values in below tags that is located inside <general> tag in Alarm.ini located inside the OD11.0Services/AlarmMailer folder at the mapped location on the Worker node.

```
<keyvault>true</keyvault>
```

```
<secret>AlarmMailerPSequence</secret>
```

Here: **AlarmMailerPSequence** is Secret name that is already created.

- ii. Update base64 Encoded string of *https://<OmniDocs+RMS Web Host URL>/Security* [Present in *AWS_ALB-IngressController.yml* file] in *SecretManager.ini* file.

For Example,

If Security war context URL is as following:

```
https://rms40.newgendocker.com/Security
```

Then, EndPointURL property in *SecretManager.ini* must be as following:

```
EndPointURL=aHR0cHM6Ly9ybXM0MC5uZXdnZW5kb2NrZXIuY29tLlNlbnV3VyaXR5
```

3.5.5 LDAP changes

Prerequisite:

- The cabinet is created and associated with the running containers. If the cabinet is not created, then refer to the [Cabinet and Data Source Creation](#) section.
- SecretManager policy must be created and attached to the Worker Node IAM role. If not created, then refer to the Creating a Secret Manager policy and secrets section.

The changes in LDAP are as follows: (For On_Prem Active Directory)

- Ensure that the LDAP Domain server is configured, and a private tunnel is created between the Kubernetes worker nodes and the LDAP Domain server.
- Update the OmniDocs+RMS EJB container name [Defined in OmniDocs11.0EJB.yml file] in *NGOClientData.xml* in between the <endPointURL></endPointURL> tags file located inside the *OD11.0Services/ODAuthMgr/ngdbini* folder at the mapped location on the Worker node.

For example,

```
<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>
```

Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.

- Update the cabinet name and domain name in the *ldap.ini* and *Hook.ini* file located inside the *OD11.0Services/ODAuthMgr* folder at the mapped location.

```

#
#Tue Nov 26 11:34:40 IST 2013
DISPort=1999
DISIPAddress=127.0.0.1
Log4j_properties_file=jtshook_log4j.properties
Encoding=UTF-8
PROTOCOL=ldap
LOGOUTTIME=15000
DIRECTORYSERVICE=ActiveDS
REACTUI=true

# Default domain name to add user For multidomain LDAP
DEFAULTDOMAIN=eco.com
ecmsuite=eco.com

```

Figure 3.19

```

#
#Wed Dec 23 10:53:14 GMT+05:30 2009
DISPort=1999
DISIPAddress=127.0.0.1
Encoding=UTF-8
setEncoding=true
LogOutTime=15000
IsMakerChecker=N

# Default domain name to add user For multidomain LDAP
ecmsuite=eco.com

```

Figure 3.20

Here, **ecmsuite** is the cabinet name and *eco.com* is the domain name.

- Update the same cabinet name and domain name in the *ldap.ini* and *Hook.ini* file located inside the *OmniDocs11.0Web\Newgen\NGConfig* folder at the mapped location.
- Update the OD11.0Services container's service name [Defined in respective YAML file] in *ldap.ini* and *Hook.ini* file located inside the *OmniDocs11.0Web\Newgen\NGConfig* folder at the mapped location.


```

#
#Tue Nov 26 11:34:40 IST 2013
DISPort=1999
DISIPAddress=od110services
Log4j_properties_file=jtshook_log4j.properties
Encoding=UTF-8
PROTOCOL=ldap
LOGOUTTIME=15000
DIRECTORYSERVICE=ActiveDS
REACTUI=true

# Default domain name to add user For multidomain LDAP
DEFAULTDOMAIN=eco.com
ecmsuite=eco.com

```

Hook.ini

Figure 3.21

```

#
#Wed Dec 23 10:53:14 GMT+05:30 2009
DISPort=1999
DISIPAddress=od110services
Encoding=UTF-8
setEncoding=true
LogOutTime=15000
IsMakerChecker=N

# Default domain name to add user For multidomain LDAP
ecmsuite=eco.com

```

Ldap.ini

Figure 3.22

Here, **od110services** is the service name of the OD11.0Services container.

- Set **<Display>** as true for LDAP in *AdminMenuOptions.xml* located inside *OmniDocs11.0Web/Newgen/NGConfig/ngdbini/Custom/CABINETNAME* folder at mapped location.

```

<SSALink>
  <LinkName>Ldap</LinkName>
  <LinkDescription>LdapDescription</LinkDescription>
  <JspName>/ldap/config.jsp</JspName>
  <Display>true</Display>
  <IconURL></IconURL>
</SSALink>

```

Figure 3.23

5. If Secret Vault is used to store sensitive information, then refer below points:

- Update the values in below tags that are located inside `<ServerInfo>` tag in `Server.xml` located inside the `OmniDocs11.0Ejb/ngdbini` folder at the mapped location on the Worker node.
`<secretName>LDAP</secretName>`
`<secretManager>Y</secretManager>`
Here: **LDAP** is Secret name that is already created.
- Update **base64 Encoded** string of `https://<OmniDocs+RMS Web Host URL>/Security` [Present in `AWS_ALB-IngressController.yml` file] in `SecretManager.ini` file located inside `OmniDocs11.0Ejb/ngdbini` folder and `OmniDocs11.0Web/ngdbini` folder.
For Example,
If Security war context URL is as following:
`https://rms40.newgendocker.com/Security/`
Then, EndPointURL property in `SecretManager.ini` must be as following:
`EndPointURL=aHR0cHM6Ly9ybXM0MC5uZXdnZW5kb2NrZXIuY29tL1N1Y3VyaXR5`
- Update the below properties in `KeyVault.properties` file located inside `OmniDocs11.0Web/ngdbini` folder.
 - Region (Its value must be region of the AWS account. For example, ap-south-1)
 - KeyVaultType (Its value must be the key vault type. For example., AWS)

The changes in LDAP are as follows: (For Azure Active Directory)

- Update the OmniDocs+RMS EJB container name [Defined in `OmniDocs11.0EJB.yml` file] in `NGOClientData.xml` in between the `<endPointURL></endPointURL>` tags file located inside the `OD11.0Services/ODAuthMgr/ngdbini` folder at the mapped location on the Worker node.
For example,
`<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>`
Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.
- Update the cabinet name, domain name, and directory service as **AzureAD** in the `Hook.ini` file located inside the `OD11.0Services/ODAuthMgr` folder at the mapped location.

```

DISPort=1999
DISIPAddress=127.0.0.1
Log4j_properties_file=jtshook_log4j.properties
Encoding=UTF-8
PROTOCOL=ldap
LOGOUTTIME=15000
DIRECTORYSERVICE=AzureAD
REACTUI=true

# Default domain name to add user For multidomain LDAP
DEFAULTDOMAIN=eco.com
ecmsuite=eco.com

```

Hook.ini

Figure 3.24

- Update the cabinet name and domain name in the *ldap.ini* file located inside the *OD11.0Services* or *ODAuthMgr* folder at the mapped location.

```

#
#Wed Dec 23 10:53:14 GMT+05:30 2009
DISPort=1999
DISIPAddress=127.0.0.1
Encoding=UTF-8
setEncoding=true
LogOutTime=15000
IsMakerChecker=N

# Default domain name to add user For multidomain LDAP
ecmsuite=eco.com

```

LDAP.ini

Figure 3.25

Here, **ecmsuite** is the cabinet name and *eco.com* is the domain name.

- Update the directory service as **AzureAD** in the *DIS.xml* file located inside the **OD11.0Services** or **ODAuthMgr** folder at the mapped location.

```

<XML>
<Title>Cabinet Mapping Information</Title>
<Body>
<Debug>
<Log4jPropPath>dissynch_log4j.properties</Log4jPropPath>
<SynchLog>true</SynchLog>
<DISLog>>false</DISLog>
<changePass>>false</changePass>
<Language>english</Language>
<Encoding>UTF-8</Encoding>
<JTSIP>127.0.0.1</JTSIP>
<JTSPort>3333</JTSPort>
</Debug>
<RootDirectoryInformation>
<DirectoryService>AzureAD</DirectoryService>
</RootDirectoryInformation>
<NoOfCabinets>0</NoOfCabinets>
<doSynch>true</doSynch>
<GroupSynch>true</GroupSynch>
<ExistingUserAsNonDomain>>false</ExistingUserAsNonDomain>
<NonDomainSupport>true</NonDomainSupport>
<InactivateDeleteUser>true</InactivateDeleteUser>
<Protocol>ldap</Protocol>
<DISPort>1999</DISPort>
<CabinetList>
</CabinetList>
</Body>
</XML>

```

Figure 3.26

- Update the same cabinet name and domain name in the *ldap.ini* and *Hook.ini* file located inside the **OmniDocs11.0Web\Newgen\NGConfig** folder at the mapped location.
- Update the OD11.0Services container's service name [Defined in respective YAML file] in *ldap.ini* and *Hook.ini* file located inside the **OmniDocs11.0Web\Newgen\NGConfig** folder at the mapped location.
- Update the directory service as **AzureAD** in *Hook.ini* and *config.ini* located inside the **OmniDocs11.0Web\Newgen\NGConfig** folder at the mapped location.

```

DISPort=1999
DISIPAddress=od110services
Log4j_properties_file=jtshook_log4j.properties
Encoding=UTF-8
PROTOCOL=ldap
LOGOUTTIME=15000
DIRECTORYSERVICE=AzureAD
REACTUI=true

# Default domain name to add user For multidomain LDAP
DEFAULTDOMAIN=eco.com
ecmsuite=eco.com

```

Hook.ini

Figure 3.27

```

#
#Wed Dec 23 10:53:14 GMT+05:30 2009
DISPort=1999
DISIPAddress=od110services
Encoding=UTF-8
setEncoding=true
LogOutTime=15000
IsMakerChecker=N

# Default domain name to add user For multidomain LDAP
ecmsuite=eco.com

```

ldap.ini

Figure 3.28

```

DIRECTORYSERVICE=AzureAD
REACTUI=true

```

config.ini

Figure 3.29

Here, **od110services** is the service name of the OD11.0Services container.

- Set **<Display>** as true for ldap in *AdminMenuOptions.xml* located inside *OmniDocs11.0Web/Newgen/NGConfig/ngdbini/Custom/CABINETNAME* folder at mapped location.
For example,

```
<SSALink>
  <LinkName>ldap</LinkName>
  <LinkDescription>ldapDescription</LinkDescription>
  <JspName>/ldap/config.jsp</JspName>
  <Display>true</Display>
  <IconURL></IconURL>
</SSALink>
```

Figure 3.30

3.5.6 SSO changes

Prerequisite:

The cabinet is created and associated with the running containers. If the cabinet is not created, then refer to [Cabinet and Data Source Creation](#) section.

The changes in SSO are as follows:

- Update the *<Host-Path URL of OmniDocsWeb container>* at the place of *ibps5aurora.newgendocker.com* in *mapping.xml* file located inside the *OmniDocs11.0Web/Newgen/NGConfig/ngdbini/SSOConFig* folder.
- Update the **CabinetName** in *mapping.xml* file located inside the *OmniDocs11.0Web/Newgen/NGConfig/ngdbini/SSOConFig* folder.
- Configure the **CabinetName=DomainName** in *sso.ini* file located inside the *OmniDocs11.0Web/Newgen/NGConfig/ngdbini/SSOConFig* folder.
- *ecmsuite=eco.com*

3.5.7 Scheduler changes

Prerequisite:

The cabinet is created and associated with the running containers. If the cabinet is not created, then refer to [Cabinet and Data Source Creation](#) section.

The changes in Scheduler are as follows:

- Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.0EJB.yml* file] in *IS.ini* in between the *<endPointURL></endPointURL>* tags file located inside the **OD11.0Services** or **Scheduler** folder at the mapped location on the Worker node.
For example,
<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>
Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.

- Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.OEJB.yml* file] in *NGOClientData.xml*, *RMAAppConfig.xml* and *RMClientData.xml* in between the `<endPointURL></endPointURL>` tags file located inside the *OD11.0Services/Scheduler/ngdbini* folder at the mapped location on the Worker node.
For example,
`<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>`
Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.
- Update the OD11.0Services container's service name [Defined in respective YAML file] in *SchedulerConf.ini* file located at **OD11.0Services** or **Scheduler** folder at the mapped location.
For example: **schedulerIpAddress=od110services**
- Update the OD11.0Services container's service name [Defined in respective YAML file] in *eworkstyle.ini* file located at *OmniDocs11.0Web/Newgen/NGConfig/ngdbini/Custom/<CABINETNAME>* folder at mapped location.
For example: **schedulerLocation=od110services**

3.5.8 ThumbnailManager changes

Prerequisite:

The cabinet is created and associated with the running containers. If the cabinet is not created, then refer to [Cabinet and Data Source Creation](#) section.

The changes in ThumbnailManager are as follows:

- Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.OEJB.yml* file] in *IS.ini* in between the `<endPointURL></endPointURL>` tags file located inside the **OD11.0Services** or **ThumbnailManager** folder at the mapped location on the Worker node.
For example,
`<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>`
Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.
- Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.OEJB.yml* file] in *NGOClientData.xml* in between the `<endPointURL></endPointURL>` tags file located inside the *OD11.0Services/ThumbnailManager/ngdbini* folder at the mapped location on the Worker node.
For example,
`<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>`
Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.

- Update the cabinet name, supervisor group's user name, and password in *ThumbnailConfig.xml* located inside the **OD11.0Services** or **ThumbnailManager** folder at the mapped location on the Worker node.

```
<cabinets><cabinet><cabinetname>ecmsuite</cabinetname><jtsip>127.0.0.1
</jtsip><jtsport>3333</jtsport><user>supervisor</user><password>:X-D;U:T-C;P-C;p5-C;b:
</password><BatchSize>10</BatchSize><priority>1</priority><encoding>UTF-8
</encoding></cabinet></cabinets>
```

Figure 3.31

3.5.9 TEM changes

Prerequisite:

The cabinet is created and associated with the running containers. If the cabinet is not created, then refer to [Cabinet and Data Source Creation](#) section.

The changes in TEM are as follows:

- Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.0EJB.yml* file] in *IS.ini* and *NGOClientData.xml* in between the `<endPointURL></endPointURL>` tags file located inside the **TEM11.0** folder at the mapped location on the Worker node.
For example,
`<endPointURL>http://od110ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>`
Here, **od110ejb** is the name of the OmniDocs+RMS EJB container.
- Update the cabinet name in filename *FTSServer-CABINETNAME-1.properties*.
For example: *FTSServer-ecmsuite-1.properties* [ecmsuite is the cabinet name].
- Update the OmniDocs+RMS EJB container name [Defined in *OmniDocs11.0EJB.yml* file] in *FTSServer-ecmsuite-1.properties*.
- Update the OmniDocs supervisor group's user name.
- Update the OmniDocs supervisor group's user password. Ensure this password must be in an encrypted format.

```
ServerAddress=od110ejb
SiteId=1
UserName=supervisor
Password=:X-D;U:T-C;P-C;p5-C;b:d:
PollTime=10
OCRPath=tesseract
DocumentCount=1000
Language=eng
SleepTime=15
```

Figure 3.32

3.5.10 EasySearch changes

Prerequisite:

The cabinet is created and associated with the running containers. If the cabinet is not created, then refer to [Cabinet and Data Source Creation](#) section.

The changes in EasySearch are as follows:

- Docker Images for EasySearch (Only one is required based on the infrastructure availability)
 - EasySearch (Apache Manifold and ElasticSearch [freeware software])
 - Or
 - EasySearch (Apache Manifold only [freeware software] with AWS managed ElasticSearch)

The EasySearch container consists of Apache Manifold and ElasticSearch [both are freeware software]. You have another option as well where you can use ElasticSearch ManagedService on AWS. In such a case, the EasySearch container consists of Apache Manifold only [freeware software]. Hence, there are two Docker images for EasySearch:

- [EasySearch \(Apache Manifold and ElasticSearch\)](#)
- [EasySearch \(Apache Manifold only\)](#)

Based on the EasySearch Docker images, configuration changes are done accordingly.

EasySearch (Apache Manifold and ElasticSearch)

- Update the EasySearch container name [Defined in *EasySearch11.0.yml* file] against the **ESServerIPAddress** key in the *ESconfig.ini* file located inside the *EasySearch11.0\ESConfigurator\conf* folder at the mapped location on the Worker node. For example, **ESServerIPAddress=easysearch11** [Where easysearch11 is the container name].
- Update the Database details in the *ESconfig.ini* file located inside the *EasySearch11.0\ESConfigurator\conf* folder at the mapped location on the Worker node.
 - ESclusterName=CABINETNAME_cluster
 - OdDBIPAddress=DBIP
 - OdDBPort=DBPORT
 - OdCabinetName=CABINETNAME
 - OdDBUserName=DBUSER
 - OdDBPassword=DBPASSWORD in encrypted format
 - OdDBType=sqlserver | oracle | postgres

```

ESServerIPAddress=easysearch11
ESServerTCPPort=9300
ESServerHttpPort=9200
ESProtocol=http
ESClusterName=ecmsuite_cluster
OdDBIPAddress=omnidocs-aurorards-db-instance-1-restore
OdDBPort=5432
OdCabinetName=ecmsuite
OdDBUserName=postgres
OdDBPassword=:X-D;Y-D;L-C;N-C;VSJ-C;4T-C;r
OdDBType=postgres
MCFIPAddress=127.0.0.1

```

Figure 3.33

- Update the cabinet name in the **CrawlerConfig.xml** file located inside the **EasySearch11.0\apache-manifoldcf-2.19\example** folder at the mapped location on the Worker node.
- Update the OmniDocs supervisor group's user name.
- Update the OmniDocs supervisor group's user password. Ensure this password must be in an encrypted format.

```

<cabinets>
<cabinet>
<cabinetname>ecmsuite</cabinetname>
<jtsip>127.0.0.1</jtsip>
<jtsport>3333</jtsport>
<user>supervisor</user>
<password>:X-D;U:T-C;P-C;p5-C;b:d</password>
<StopPhraseFlag>N</StopPhraseFlag>
<StopPhrases>
<StopPhrase>Newgen Software Technologies</StopPhrase>
<StopPhrase>omnidocs 11.0</StopPhrase>
</StopPhrases>
<Pages>ALL</Pages>
</cabinet>
</cabinets>

```

Figure 3.34

- Update the cabinet name in the *elasticsearch.yml* file located inside the *EasySearch11.0\elasticsearch-7.17.4\config* folder at the mapped location on the Worker node. For example,

```
# Use a descriptive name for your cluster:
#
cluster.name: ecmsuite_cluster
#
```

Figure 3.35

Where, **ecmsuite** is the cabinet name.

- Update the cabinet name and EasySearch container name [Defined in *EasySearch11.0.yml* file] in the *EasySearch.xml* file located inside the *OmniDocs110Ejb\ngdbini* folder at the mapped location on the Worker node.

```
<ElasticSearch>
  <ESIPAddress>easysearch11</ESIPAddress> <!--Adress of ElasticSearch Server -->
  <ESPort>9200</ESPort> <!--Port ElasticSearch Server running on -->
  <ESClusterName>ecmsuite_cluster</ESClusterName> <!-- Name of your cluster -->
  <ESUserName></ESUserName> <!-- Mandatory value for basic authentication -->
  <ESSecret></ESSecret> <!--Mandatory value for basic authentication -->
  <ESProtocol>http</ESProtocol> <!-- Set https for SSL/TLS configuration -->
  <ESTrustStorePath></ESTrustStorePath> <!--Value optional for SSL/TLS configuration-->
  <ESTrustStoreSecret></ESTrustStoreSecret> <!--Value optional for SSL/TLS configuration-->
</ElasticSearch>
```

Figure 3.36

Here, **easysearch11** is the container name and **ecmsuite** is the cabinet name.

- Update the EnableEasySearch=Y and EasySearch container name [Defined in *EasySearch11.0.yml* file] in the *eworkstyle.ini* file located inside the *OmniDocs11.0Web\Newgen\NGConfig\ngdbini\Custom\CABINET_NAME* folder at the mapped location on the Worker node.

```
#For EasySearch
EnableEasySearch=Y
EnableEasySearchIndexDropDown=N
EasySearchIPAddress=easysearch11
EasySearchHttpPort=9200
```

Figure 3.37

Here, **easysearch11** is the container name.

EasySearch (Apache Manifold Only)

In the case of the EasySearch (Apache Manifold only) container, use the AWS ElasticSearch Managed Service. Also, ensure that the Elastic Managed Service is already created before further processing.

- Update the EasySearch container name [Defined in *EasySearch11.0_ApacheOnly.yml* file] against the **ESServerIPAddress** key in the *ESconfig.ini* file located inside the *EasySearch11.0\ESConfigurator\conf* folder at the mapped location on the Worker node. For example, **ESServerIPAddress=easysearch11** [Where easysearch11 is the container name].
- Update the Database details in the *ESconfig.ini* file located inside the *EasySearch11.0\ESConfigurator\conf* folder at the mapped location on the Worker node.
 - **ESClusterName=CABINETNAME_cluster**
 - **OdDBIPAddress=DBIP**
 - **OdDBPort=DBPORT**
 - **OdCabinetName=CABINETNAME**
 - **OdDBUserName=DBUSER**
 - **OdDBPassword=DBPASSWORD** in encrypted format
 - **OdDBType=sqlserver | oracle | postgres**

```
ESServerIPAddress=easysearch11
ESServerTCPPort=9300
ESServerHttpPort=9200
ESProtocol=http
ESClusterName=ecmsuite_cluster
OdDBIPAddress=omnidocs-aurorards-db-instance-1-restore
OdDBPort=5432
OdCabinetName=ecmsuite
OdDBUserName=postgres
OdDBPassword=:X-D;Y-D;L-C;N-C;VSJ-C;4T-C;r
OdDBType=postgres
MCFIPAddress=127.0.0.1
```

Figure 3.38

- Update the **UseAWSElasticSearch=true** in the *ESconfig.ini* file located inside the *EasySearch11.0\ESConfigurator\conf* folder at the mapped location on the Worker node.
- Update the **AWSesDomainURL**, **AWSAccessKey**, **AWSecretKey**, and **AWSRegion** in the *ESconfig.ini* file located inside the *EasySearch11.0\ESConfigurator\conf* folder at the mapped location on the Worker node.

For example,

```

UseAWSElasticSearch=true
AWSESDomainURL=https://vpc-testdomain1.ap-south-1.es.amazonaws.com
AWSAccessKey=AKIAJ3Q33111D6U57DM
AWSecretKey=IXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
AWSRegion=ap-south-1

```

Figure 3.39

- Update the cabinet name in the *CrawlerConfig.xml* file located inside the **EasySearch11.0\apache-manifoldcf-2.19\example** folder at the mapped location on the Worker node.

```

<cabinets>
<cabinet>
<cabinetname>ecmsuite</cabinetname>
<jtsip>127.0.0.1</jtsip>
<jtsport>3333</jtsport>
<user>supervisor</user>
<password>:X-D;U:T-C;P-C;p5-C;b:d</password>
<StopPhraseFlag>N</StopPhraseFlag>
<StopPhrases>
<StopPhrase>Newgen Software Technologies</StopPhrase>
<StopPhrase>omnidocs 11.0</StopPhrase>
</StopPhrases>
<Pages>ALL</Pages>
</cabinet>
</cabinets>

```

Figure 3.40

- Update the cabinet name and EasySearch container name [Defined in *EasySearch11.0_ApacheOnly.yml* file] in the **EasySearch.xml** file located inside the **OmniDocs11.0Ejb\ngdbini** folder at the mapped location on the Worker node.
For example,

```

<ElasticSearch>
  <ESIPAddress>easysearch11</ESIPAddress> <!--Adress of ElasticSearch Server -->
  <ESPort>9200</ESPort> <!--Port ElasticSearch Server running on -->
  <ESClusterName>ecmsuite_cluster</ESClusterName> <!-- Name of your cluster -->
  <ESUserName></ESUserName> <!-- Mandatory value for basic authentication -->
  <ESSecret></ESSecret> <!--Mandatory value for basic authentication -->
  <ESProtocol>http</ESProtocol> <!-- Set https for SSL/TLS configuration -->
  <ESTrustStorePath></ESTrustStorePath> <!--Value optional for SSL/TLS configuration-->
  <ESTrustStoreSecret></ESTrustStoreSecret> <!--Value optional for SSL/TLS configuration-->
</ElasticSearch>

```

Figure 3.41

Here, **easysearch11** is the container name and **ecmsuite** is the cabinet name.

- Update the `<AWSElasticSearch value= "true">` in the `EasySearch.xml` file located inside the `OmniDocs11.0Ejb\ngdbini` folder at the mapped location on the Worker node.
- Update the `AWSesDomainURL`, `AWSAccessKey`, `AWSsecretKey`, and `AWSRegion` in the `EasySearch.xml` file located inside the `OmniDocs11.0Ejb\ngdbini` folder at the mapped location on the Worker node.

For example,

```
<AWSElasticSearch value= "true">  
  <AWSesDomainURL>https://vpc-testdomain1.ap-south-1.es.amazonaws.com</AWSesDomainURL>  
  <AWSAccessKey>AKIAJ3726P7W464U572M</AWSAccessKey>  
  <AWSsecretKey>A3FQDQVQ733V6D623J23A33343323</AWSsecretKey>  
  <AWSRegion>ap-south-1</AWSRegion>  
</AWSElasticSearch>
```

Figure 3.42

- Update the `EnableEasySearch=Y` and EasySearch container name [Defined in `EasySearch11.0.yml` file] in the `eworkstyle.ini` file located inside the `OmniDocs11.0Web\Newgen\NGConfig\ngdbini\Custom\CABINET_NAME` folder at the mapped location on the Worker node.

For example,

```
#For EasySearch  
EnableEasySearch=Y  
EnableEasySearchIndexDropDown=N  
EasySearchIPAddress=easysearch11  
EasySearchHttpPort=9200
```

Figure 3.43

Here, **easysearch11** is the container name.

3.5.11 WOPI changes

- Update the OmniDocsEJB container name [Defined in *OmniDocs11.OEJB.yml* file] in *NGOClientData.xml* file in between the `<endPointURL></endPointURL>` tags located inside the *OmniDocs_WOPI\Newgen\NGConfig\ngdbini* folder at the mapped location on the Worker node.

```
<?xml version="1.0"?>
<ClientInfo>
  <ProviderUrl></ProviderUrl>
  <JndiServerName></JndiServerName>
  <JndiServerPort></JndiServerPort>
  <ContextSuffix></ContextSuffix>
  <WildFlyUserName></WildFlyUserName>
  <WildFlyPassword></WildFlyPassword>
  <JndiContextFactory></JndiContextFactory>
  <ClientLookUpName>ejb:omnidocs_ejb/omnidocs_ejb/NGOClientServiceHandlerBean!com.newgen.omni.jts.txn.NGOClientServiceHandlerHome</ClientLookUpName>
  <AdminLookUpName>ejb:omnidocs_ejb/omnidocs_ejb/NGOAdminServiceHandlerBean!com.newgen.omni.jts.txn.NGOAdminServiceHandlerHome</AdminLookUpName>
  <UrlPackagePrefix></UrlPackagePrefix>
  <endPointURL>http://rms40ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>
  <xmlParamName>InputXML</xmlParamName>
  <HeaderKey></HeaderKey>
  <HeaderValue></HeaderValue>
</ClientInfo>
```

Figure 3.44

Here, **rms40ejb** is the name of the OmniDocsEJB container.

- Update the OmniDocsEJB container name [Defined in *OmniDocs11.OEJB.yml* file] in *IS.ini* file in between the `<endPointURL ></endPointURL >` tags located inside the *OmniDocs_WOPI\Newgen\NGConfig* folder at the mapped location on the Worker node. For example,

```
<Encoding>UTF-8</Encoding>
<LogPath>Replication.log</LogPath>
<SMSTimeOut>60000</SMSTimeOut>
<SMSReadInterval>30000</SMSReadInterval>
<SMSRetryCount>5</SMSRetryCount>
<SMSGenerateLog>true</SMSGenerateLog>
<IsJNDI>true</IsJNDI>
<endPointURL>http://rms40ejb:8080/callbroker/execute/GenericCallBroker</endPointURL>
<xmlParamName>InputXML</xmlParamName>
<HeaderKey></HeaderKey>
<HeaderValue></HeaderValue>
<ProviderUrl></ProviderUrl>
```

Figure 3.45

- Update the WOPI_SOURCE, RMS_REDIRECTURL and CABINETNAME in *WOPIConfiguration.ini* file located inside the *OmniDocs_WOPI\Newgen\NGConfig\AddInsConfig* folder at the mapped location on the Worker node.

```

WOPI_SOURCE=https://wopi.newgendocker.com

#To enable custom app functionality and breadcrumb URL redirection,should incorporate the app at this location.
#OMNIDOCES_REDIRECTURL=https://rms40.newgendocker.com/omnidocs/wopi/redirect.html
RMS_REDIRECTURL=https://rms40.newgendocker.com/omnidocs/wopi/redirect.html

#MYAPP_REDIRECTURL=https://wopi.domain.com/Apps/redirect.html

#Cabinet Index If admin wants to configure multiple cabinet then need to add new cabinet with increment index .
#ngofficewopi_INDEX=1 This is example for ngofficewopi cabinetname and index 1
rmspostgres27feb_INDEX=1

```

Figure 3.46

Where,

https://wopi.newgendocker.com is host URL of WOPI container.

https://rms40.newgendocker.com is Host URL of RMS WEB container.

rmspostgres27feb is cabinet name.

- Open the *web.xml* file in edit mode located inside the *OmniDocs_WOPI* folder at the mapped location on the Worker node.
- Search for filter-class `<filter-class>org.apache.catalina.filters.CorsFilter</filter-class>` and update the `<param-value></param-value>` tag's value with OmniDocs URL against `<param-name>antiClickJackingUri</param-name>` and `*` against `<param-name>cors.allowed.origins</param-name>`

```

<filter>
  <filter-name>httpHeaderSecurity</filter-name>
  <filter-class>org.apache.catalina.filters.HttpHeaderSecurityFilter</filter-class>
  <async-supported>true</async-supported>
  <init-param>
    <param-name>antiClickJackingOption</param-name>
    <param-value>ALLOW-FROM</param-value>
  </init-param>
  <init-param>
    <param-name>antiClickJackingUri</param-name>
    <param-value>omnidocs11alpine2.newgendocker.com</param-value>
  </init-param>
</filter>

<filter>
  <filter-name>CorsFilter</filter-name>
  <filter-class>org.apache.catalina.filters.CorsFilter</filter-class>
  <init-param>
    <param-name>cors.allowed.origins</param-name>
    <param-value>*</param-value>
  </init-param>
  <init-param>
    <param-name>cors.allowed.methods</param-name>
    <param-value>GET,POST,HEAD,OPTIONS,PUT</param-value>
  </init-param>
  <init-param>
    <param-name>cors.allowed.headers</param-name>
    <param-value>Content-Type,X-Requested-With,accept,Origin,Access-Control-Request-Method,Access-Control-Request-Headers,Access-Control-Allow-Origin
  </param-value>
  </init-param>
</filter>

```

Figure 3.47

- Add the `CSPHeaderAllowedDomains` tag in the *eworkstyle.ini* file located inside the *OmniDocs11.0Web/Newgen/NGConfig/ngdbini/odwebini* folder at the mapped location on the Worker node.


```
CSPHeaderAllowedDomains=default-src * data: 'unsafe-inline' 'unsafe-eval';
```

- Add the WOPIOfficeExtensionSupport and WOPIOfficeExtensionSupportURL tag in the *eworkstyle.ini* file located inside the *OmniDocs11.0Web/Newgen/NGConfig/ngdbini/Custom/CABINET_NAME* folder at the mapped location on the Worker node.

```
WOPIOfficeExtensionSupport=doc,docx,DOCX,DOC,xls,xlsx,XLSX,XLS,ppt,pptx,PPTX,PPT,wopitest,WOPITEST,wopitestx,WOPITESTX  
WOPIOfficeExtensionSupportURL=https://wopi.newgendocker.com
```

3.5.12 OmniScanWeb changes

The changes in OmniScanWeb are as follows:

- Update the AWS Elastic Redis cache's configuration endpoint in *redisson.yaml* file against the *singleServerConfig* or *clusterServersConfig*. If redis cache is SSL enabled, then use the *redis://<endpoint url>:port* and if SSL is not enabled, then use *redis://<endpoint url>:port*. This *redisson.yaml* file is located inside the *OmniScanWeb6.0* folder at the mapped location on the Worker node.

```

--
singleServerConfig:
  idleConnectionTimeout: 10000
  connectTimeout: 10000
  timeout: 3000
  retryAttempts: 3
  retryInterval: 1500
  password: null
  subscriptionsPerConnection: 5
  clientName: null
  address: "redis://XXXXXXXXXXXXXXXXXXXX@amazonaws.com:6379"
  subscriptionConnectionMinimumIdleSize: 1
  subscriptionConnectionPoolSize: 50
  connectionMinimumIdleSize: 24
  connectionPoolSize: 64
  database: 0
  dnsMonitoringInterval: 5000
threads: 16
nettyThreads: 32
codec: !<org.redisson.codec.MarshallingCodec> {}
transportMode: "NIO"

#Reference: https://github.com/redisson/redisson/wiki/2.-Configuration#26-single-instance-mode

#CLUSTER ---
#CLUSTER clusterServersConfig:
#CLUSTER   idleConnectionTimeout: 10000
#CLUSTER   connectTimeout: 10000
#CLUSTER   timeout: 3000
#CLUSTER   retryAttempts: 3
#CLUSTER   retryInterval: 1500
#CLUSTER   failedSlaveReconnectionInterval: 3000
#CLUSTER   failedSlaveCheckInterval: 60000
#CLUSTER   password: null
#CLUSTER   subscriptionsPerConnection: 5
#CLUSTER   clientName: null
#CLUSTER   loadBalancer: !<org.redisson.connection.balancer.RoundRobinLoadBalancer> {}
#CLUSTER   subscriptionConnectionMinimumIdleSize: 1

```

Figure 3.48

3.5.13 RMS SharePoint Adapter changes

Prerequisite:

The cabinet must be created and associated with the running containers. To create a cabinet, refer to the [Creating cabinet and data source](#) section.

The changes in RMS SharePoint Adapter are as follows:

1. Update the cabinet name in filename *RMS-SPServer-CABINETNAME-1.properties* located inside *SharePointAdapter/properties* folder at the mapped location on the worker node.
For example, RMS-SPServer-ecmsuite-1.properties [**ecmsuite** is the cabinet name].
2. Change the required changes in *RMS-SPServer-CABINETNAME-1.properties* file.

Following is the sample of the Properties file:

```
AutoArchiveActive=Y
OnlineSharePointSite=Y
RMSCabinetName =spcab2
RMSServerAddress=omnidocs.newgendocker.com
RMSServerPort=443
RMSServerProtocols=https
RMSUserName=supervisor2
RMSPassword=:X-D;U:T-C;P-C;p5-C;b:d:u:SJDE
sharePointUserDomain=vwm55.onmicrosoft.com
sharePointHostIPAddress=vwm55.onmicrosoft.com
shraePointPort=443
sharePointProtocols=https
sharePointSiteAbsoluteURL=
sharePointUser=spadmin@vwm55.onmicrosoft.com
sharePointSequence=:F:0s-C;6-C;G-C;a:8:y-D;W:Z-C;4P-C;a-C;k:0
sharePointSite=newSite
SharePointReconciliationReports=Y
SharePointReportsLibrary=/sites/newSite/SharePointRMSReport/
MaxFailerCount=3
```

Description of the Properties keys:

Keys	Description
AutoArchiveActive	Y signifies this properties file is in used whereas N signifies this file is not in use.
OnlineSharePointSite	Y – Signifies online method is working. N – Signifies offline method is working.
RMSCabinetName	Name of the RMS cabinet.
RMSServerAddress	RMS web container host URL.
RMSServerPort	RMS web container port that is, 443 = if SSL enabled or 80 = if SSL Disabled.
RMSServerProtocols	RMS server protocol that is, https = if SSL enabled or http = if SSL Disabled.
RMSUserName	RMS supervisor’s group user name.
RMSPassword	RMS supervisor’s group user encrypt password from alarm mailer.
sharePointUserDomain	SharePoint domain name.
sharePointHostIPAddress	SharePoint site server address.
shraePointPort	SharePoint site server port.
sharePointProtocols	SharePoint site server protocols.
sharePointSiteAbsoluteURL	SharePoint site absolute URL.
sharePointUser	SharePoint admin user name.
sharePointSequence	SharePoint encrypt password from alarm mailer.
sharePointSite	SharePoint site name.
SharePointReconciliationReports	Y or N.
SharePointReportsLibrary	SharePoint Reconciliation Reports absolute path.
MaxFailerCount	Maximum failure count.

```

AutoArchiveActive=Y
OnlineSharePointSite=Y
RMSCabinetName=rms23dec
RMSServerAddress=omnidocs.newgendocker.com
RMSServerPort=443
RMSServerProtocols=https
RMSUserName=supervisor2
RMSPassword=:l37-C;q;J-D;lX-C;6
sharePointUserDomain=vwm55.onmicrosoft.com
sharePointHostIPAddress=vwm55.onmicrosoft.com
shraePointPort=443
sharePointProtocols=https
sharePointSiteAbsoluteURL=
sharePointUser=spadmin@vwm55.onmicrosoft.com
sharePointSequence=:X-D;Y-D;N-C;VSJ-C;4T-C;r
sharePointSite=SharpeointRMssite
SharePointReconciliationReports=Y
SharePointReportsLibrary=/site/SharpeointRMssite/SharePointRMSReport/
MaxFailerCount=3

```

Figure 3.49

- Update **site name**, **tenantID**, **ClientID** and **clientSecret** in file `SharePointConfig.ini` located inside the `OmniDocs11.0Web\Newgen\NGConfig\AddInsConfig` folder at the mapped location on the Worker node.

For Example,

```
SharpeointRMssite_tenantID=U-D;P/-[REDACTED];d:e:Q2C-D;A-C;l-D;X:B:H:tQ:[REDACTED];xJ-[REDACTED];j:f-C;c:V:N
SharpeointRMssite_clientID=Q-C;79-D;5-C;3:2tLa-C;I-D;[REDACTED];ksGqu:gB-D;L:V:MF-C;g-C;M:0-C;p:7:y-D;p-C;8Rv:s
SharpeointRMssite_clientSecret:[REDACTED];g:S-D;7ws-D;a-D;x-D;q:z:I:fHL-C;R-D;M:r4-D;IT-D;[REDACTED];g-C;o-D;ic7-D;3-D;9
```

Figure 3.50

NOTE:

The TenantID, ClientID and clientSecret must be in encrypted format.

3.5.14 Messaging Service changes

Prerequisite:

A cabinet must be created. To create a cabinet, refer to the [Creating cabinet and data source](#) section.

The changes in Messaging Service are as follows:

- Update the Message Service Hosted URL in **hostHeaderWhitelist** in the `application.properties` file located inside the `MessagingService` folder at the mapped location on the Worker node.
For Example –

```
#spring.mvc.converters.preferred-json-mapper=gson
spring.mvc.pathmatch.matching-strategy = ANTI_PATH_MATCHER
version=1.0
spring.mvc.dispatch-trace-request=true

server.port=8080
debugLog = false
hostHeaderWhitelist=["odmsgservice.newgendocker.com","localhost"]
hostHeaderCheck = Y
#server.ssl.enabled=true
#server.ssl.key-store: newgensoftware.net.jks
```

Figure 3.51

- Update the Database details in the `ServerInfo.json` file located inside the `MessagingService/omniconf` folder at the mapped location on the Worker node.
For Example –

```
"cabinet_name": {
  "userName": "postgres",
```

```

"password": ":X-D;Y-D;L-C-C;VSJ-C;4T-C;s",
"maxDBConnections": 50,
"minDBConnections": 5,
"querytimeout": 1000,
"cabinetType": "B",
"databaseType": "POSTGRES",
"port": 5432,
"service": "",
"comment": "postgres",
"driverName": "org.postgresql.Driver",
"driverURL": "jdbc:postgresql://auroxxx-xxxxxxxx-xx.cluster-
cvxxxxxxxxekwxu.ap-south-1.rds.amazonaws.com:5432/cabinet_name",
"tnsname": "",
"idleTimeout": 600000,
"maxLifetime": 1800000,
"connectionTimeout": 300000,
"autoCommit": true
}

```

Parameters	Description
cabinet_name	The name of the cabinet. Provide the cabinet name in lowercase letters.
userName	Database username.
Password	The password of the above user in encrypted form. Encrypt the password as follows: For Windows: Encrypt the password using the <i>encryptPassword.bat</i> file located at <i>OD11\Common Services for J2EE\AlarmMailer\encryptPassword.bat</i> . For Linux: Go to the <i>encryptPassword.sh</i> file located at <i>OD11\Common Services for J2EE\AlarmMailer\encryptPassword.sh</i> and provide full rights using the below command for password encryption: <code>chmod 777 encryptPassword.sh.</code>
maxDBConnections	The maximum number of allowed database connections.
minDBConnections	The minimum number of allowed database connections.
querytimeout	It defines the timeout period of the responding database in minutes. Example – 6
cabinetType	It defines the type of cabinet. Example - B or I or D
databaseType	It defines the type of database cabinet. Example – POSTGRES
port	Provide the port number of the database server.
service	NA
comment	It is used to provide related comments at the time of cabinet creation.
driverName	The name of the class that implements a protocol JDBC for a database connection. Example: "org.postgresql.Driver"

Parameters	Description
driverURL	It contains information about the database to connect and other configuration properties. Example: “jdbc:postgresql://<IP Address or host name of the database server>:<Port number of the database server>/<Cabinet name>”
tnsname	It is the configuration file that contains network service names mapped to connect descriptors for the local naming method, or net service names mapped to the listener protocol addresses.
idleTimeout	It controls the maximum amount of time that a connection is allowed to sit idle in the pool. The minimum allowed value is 10000ms (10 seconds). Default: 600000 (10 minutes).
maxLifetime	It controls the maximum lifetime of a connection in the pool. A value of 0 indicates no maximum lifetime (infinite lifetime), subject of course to the idleTimeout setting. The minimum allowed value is 30000ms (30 seconds). Default: 1800000 (30 minutes).
connectionTimeout	It controls the maximum number of milliseconds that a client will wait for a connection from the pool. The lowest acceptable connection timeout is 250 ms. Default: 30000 (30 seconds).
autoCommit	It controls the default auto-commit behavior of connections returned from the pool. It is a boolean value. Default: true.

3.6 Deploying containers

Perform the below steps to deploy the containers:

1. You can deploy the containers on AWS Elastic Kubernetes Service from our local machine by executing the below command or you can deploy them using AWS CodePipeline. However, it recommends deploying the containers using AWS CodePipeline for better traceability.

```
kubectl apply -f <YAML_File>
```

For example,

```
kubectl apply -f OmniDocs11.0Web.yml
```

NOTE:

- To execute the above command, kubectl must be configured on your local server. Refer to the [Configuration of AWS Kubernetes Cluster](#) section to run kubectl from your local machine.
 - To deploy the containers using AWS CodePipeline, AWS CodePipeline must be configured. Refer to the [Configuration of AWS CodePipeline](#) section.
-

2. In AWS CodePipeline, a separate Release pipeline is created for each Docker image like OmniDocs11.0Web, OmniDocs11.0WebService, OmniDocs11.0EJB, OmniDocs11.0Services, EasySearch11.0, TEM11.0, and OmniScanWeb6.0.

For Example

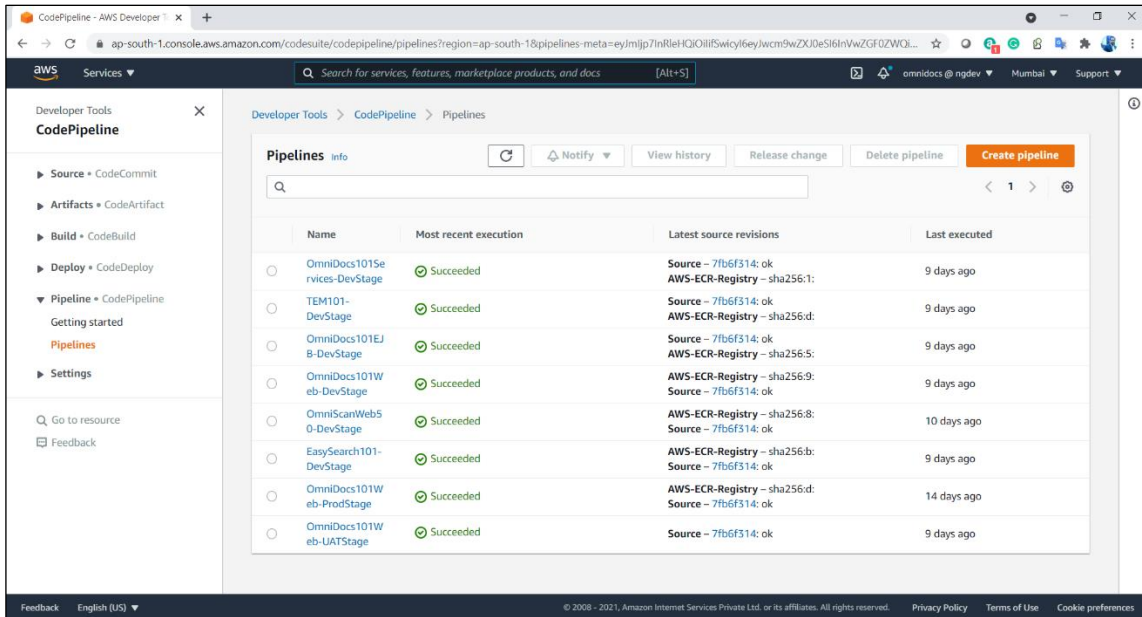


Figure 3.52

3. Trigger the Release Pipeline to deploy the required Docker containers.
4. Once the deployment is done, deployed containers can be visible from the Kubernetes Dashboard. Refer to the [Configuration of AWS Kubernetes Cluster](#) to configure the Kubernetes Dashboard.

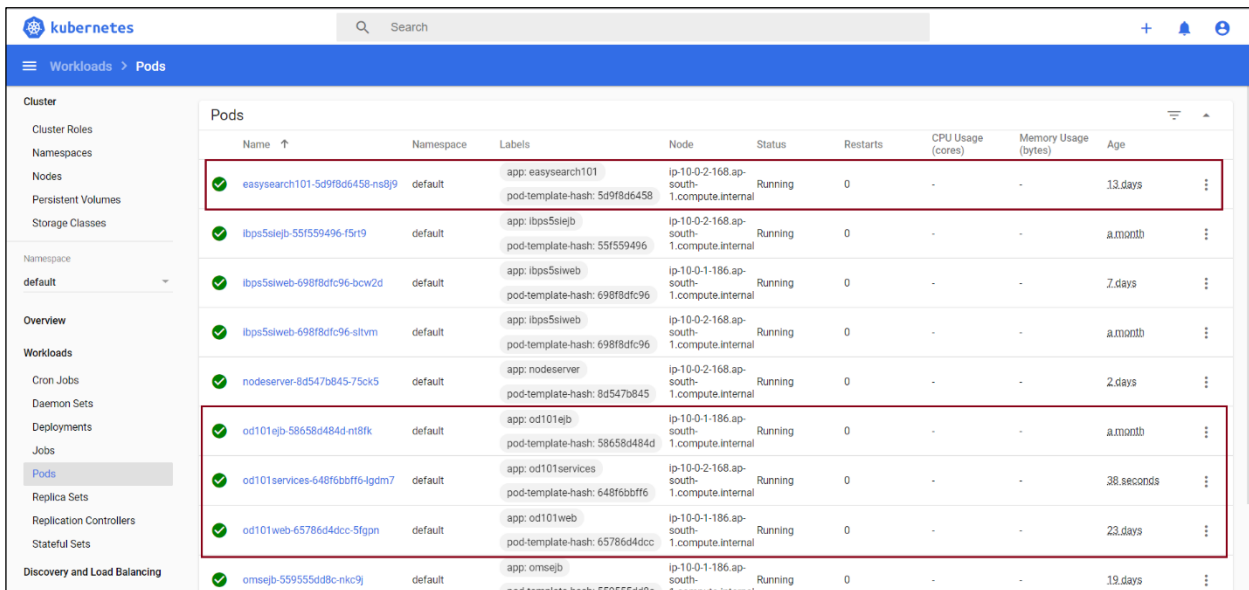


Figure 3.53

5. Update the container's replica set from 1 (default value) to any other number in YAML files, then that number of containers is listed in Kubernetes Dashboard.

- Increase the replica set from Kubernetes Dashboard from the **Deployments** menu on the left-panel.
For Example,

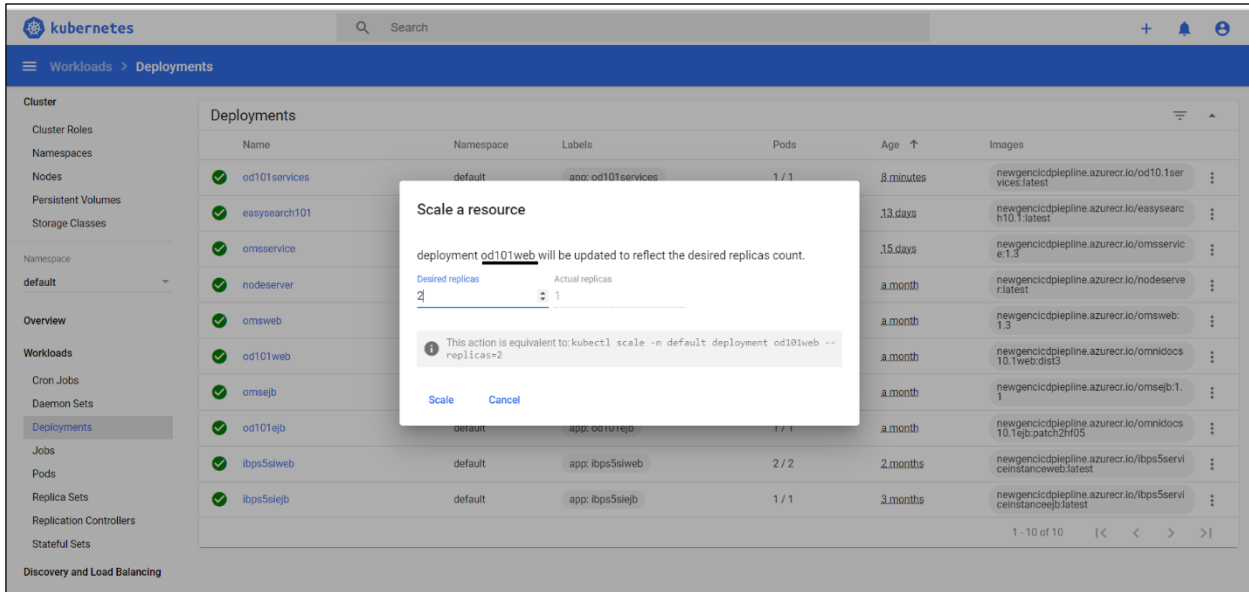


Figure 3.54

- But, when you redeploy the containers from AWS CodePipeline, the replica set increased from Kubernetes gets overwritten by the replica set defined in the YAML file.
- In any case, if you need to restart the container, then you have two options. Either redeploy the container from AWS CodePipeline which launches the new container by following up the rolling update feature of Kubernetes or execute the restart command from the Kubernetes pod's shell.
- To execute the restart command from the Kubernetes pod's shell, follow the below steps:
 - Open the **Kubernetes Dashboard** and list out all the deployed pods.
 - Click **Pod** that you want to restart.
 - Click **Exec into pod** icon given on upper-right panel.

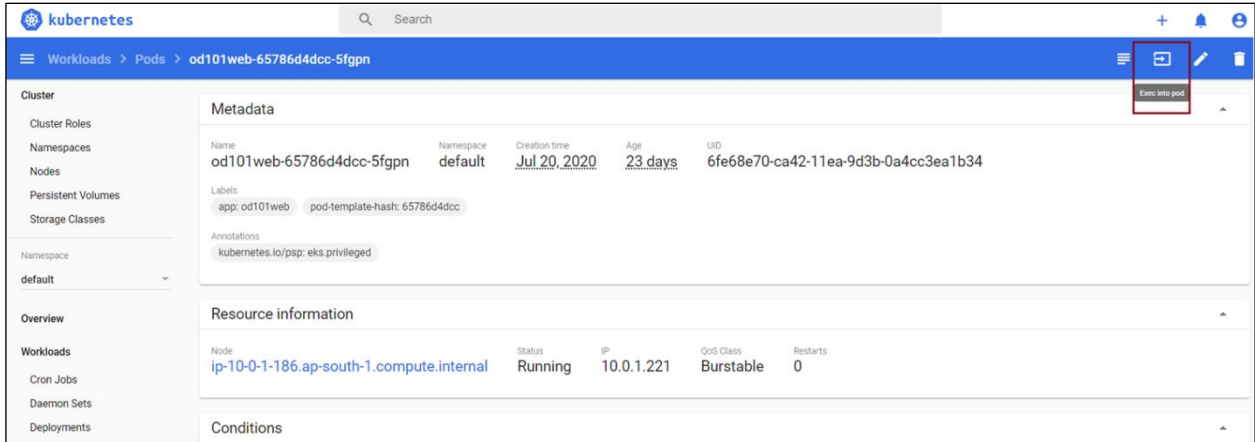


Figure 3.55

- After that pod's shell terminal opens.
- Execute the below command to restart the container:


```
restartjws.sh
```
- The restart command is different for each container. Refer to the below table:

Container Name	Restart Command
OmniDocs11.0Web, OmniDocs11.0WebService	restartjws.sh
OmniDocs11.0EJB	restartjboss.sh
OmniDocs11.0Services	restartalarm.sh, restartauthmgr.sh, restartscheduler.sh, restartthumbnail.sh, restartwrapper.sh
EasySearch11.0(With ElasticSearch)	restarteasysearch.sh
EasySearch11.0(Without ElasticSearch)	restartapache.sh
TEM11.0	restarttem.sh
OmniScanWeb6.0	restartjws.sh
RMSSharePointAdapter	Restartsharepointadapter.sh
Messaging Service	Restartservice.sh

10. Once the EasySearch11 container is deployed (Whether **With ElasticSearch** or **Without ElasticSearch**), execute the below command in Kubernetes pod's shell for the 1st time to configure the Apache Manifold jobs. After that in subsequent deployments, this execution is not required.

```
runESConfigurator.sh
```

3.7 Creating cabinet and data source

Prerequisites:

- OmniDocs+RMS Web, OmniDocs+RMS EJB, and OmniDocsServices are already deployed.
- ALB Ingress Controller is already configured and deployed using the *AWS_ALB-IngressController.yml* file.
- S3 bucket is already created to store the PN files. PN files are encrypted files that contain all the added, uploaded, and scanned documents by Newgen products.

Once the above prerequisites are fulfilled, refer the below sections to create the Cabinet and Data Source.

- [Getting started with OSA](#)
- [Registering JTS server](#)
- [Connecting OSA to JTS Server](#)
- [Creating a cabinet](#)
- [Associating a cabinet](#)
- [Creating a data source](#)
- [Registering a cabinet in OmniDocs](#)
- [Registering a cabinet in RMS](#)
- [Creating Site and Volume](#)

3.7.1 Getting started with OSA

Perform the below steps to start the OSA:

1. Since the container is a CLI-based deployment you can't launch any GUI-based application inside the container. But you must use the OSA to create a cabinet that is a GUI-based application. In such a case, deploy OSA to some GUI-based machine either on a local server or on an EC2 instance. Also, add an inbound rule in the Kubernetes worker node's security group to allow OSA to communicate with the OmniDocs11.0Services container deployed on that worker node.
2. Once OSA is deployed on a machine, navigate to the OSA folder on that machine and double click RunAdmin.bat (For Windows) or RunAdmin.sh (For Linux) to start OSA.
3. When the application is launched. The Login dialog box appears.

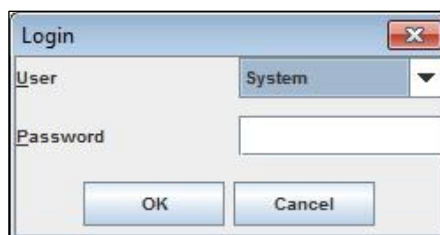


Figure 3.56

4. Select the user as **System** and specify the password as **system**.
5. Click **OK** to log in. After the successful login, the OSA screen appears displaying the list of registered services.

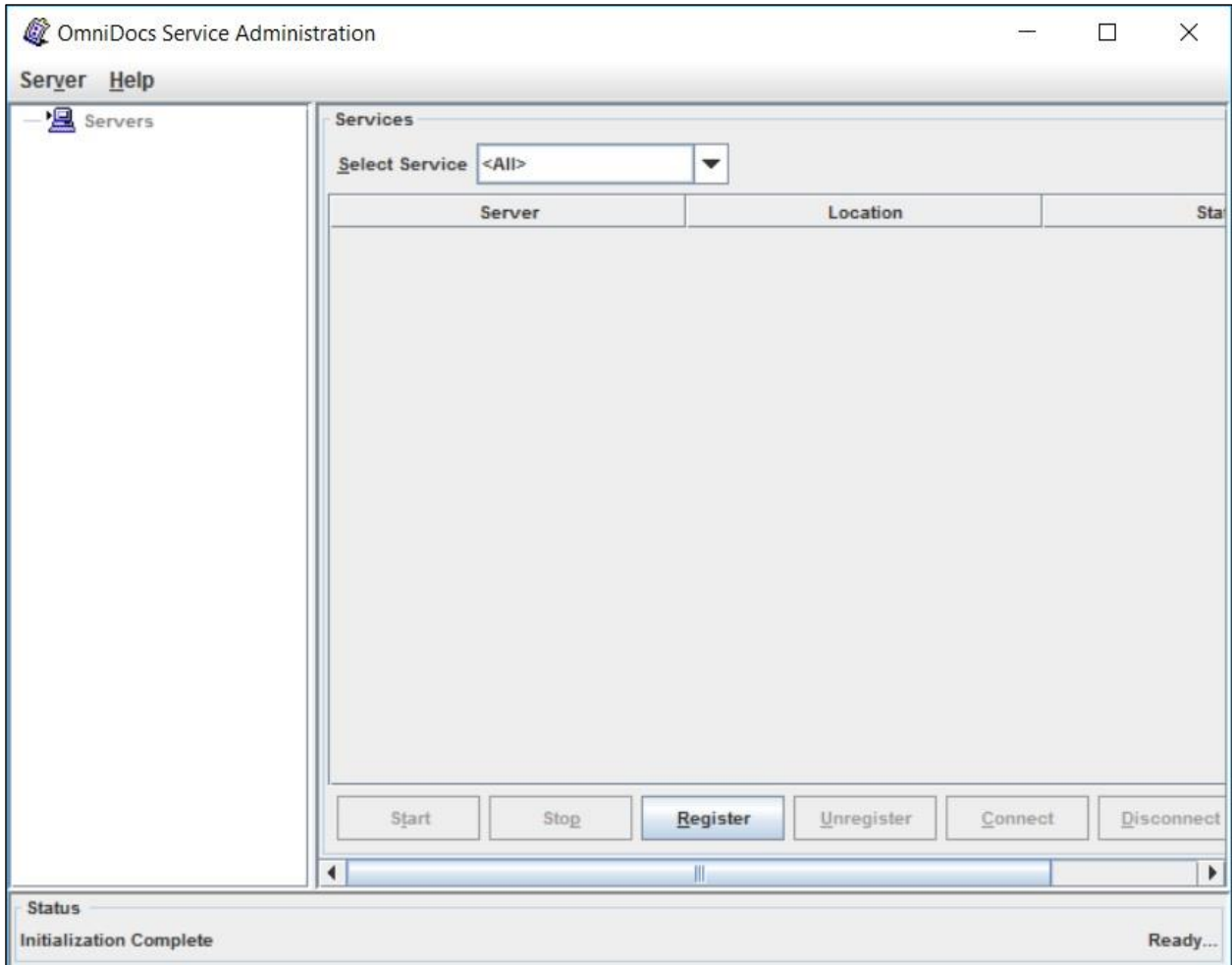


Figure 3.57

3.7.2 Registering JTS server

Perform the below steps to register the JTS Server:

1. To register the JTS server, click **Register** button. The **Register New Server** dialog box appears.

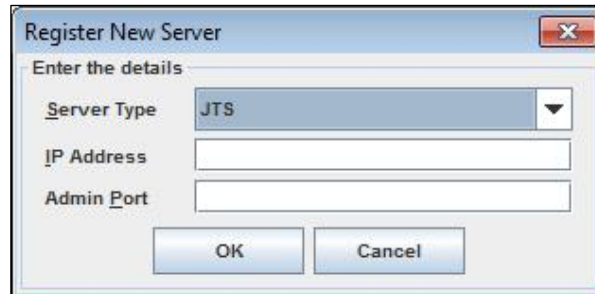


Figure 3.58

2. Select the JTS and specify the public IP address of the Kubernetes Worker node on which the OmniDocs11.0Services (Wrapper, AlarmMailer, THN, and so on) container is deployed. For example, suppose there are two worker nodes in the Kubernetes cluster and after deploying the OmniDocs11.0Services container, it gets deployed to the 1st worker node then specify the IP address of the 1st worker node. But in a case, 2 replicas are deployed on the OmniDocs11.0Services container, one on each worker node, in that case, specify the IP address of any worker node.
3. Specify the Admin port of Wrapper service running inside the OmniDocs11.0Services container. Since Wrapper is running inside the container with Admin port 9996 but that Admin port cannot be accessed directly. Kubernetes generates a random port (aka NodePort) for each port running inside the container that is exposed outside the container for public use. To get this NodePort either from Kubernetes Dashboard or by executing the below command from your local machine:

```
kubectl get svc <OmniDocs11.0Services container name>
```

For example,

```
C:\WINDOWS\system32\cmd.exe
C:\Users\vivek_kumar>kubectl get svc od101services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
od101services NodePort      172.20.43.136 <none>         3333:30846/TCP  9996:31370/TCP  1999:31477/TCP  3h49m
C:\Users\vivek_kumar>
```

Figure 3.59

Here, **Wrapper Admin port 9990** is exposed outside the container and Kubernetes has generated a random port 31370 as a NodePort. This NodePort keeps changing whenever you redeploy the container.

Figure 3.60

- 4. Click **OK** to register the JTS Server.

3.7.3 Connecting OSA to JTS Server

Perform the below steps to connect the OSA to the JTS Server:

1. Once the JTS Server is registered, it is displayed in the list in a disconnected state.

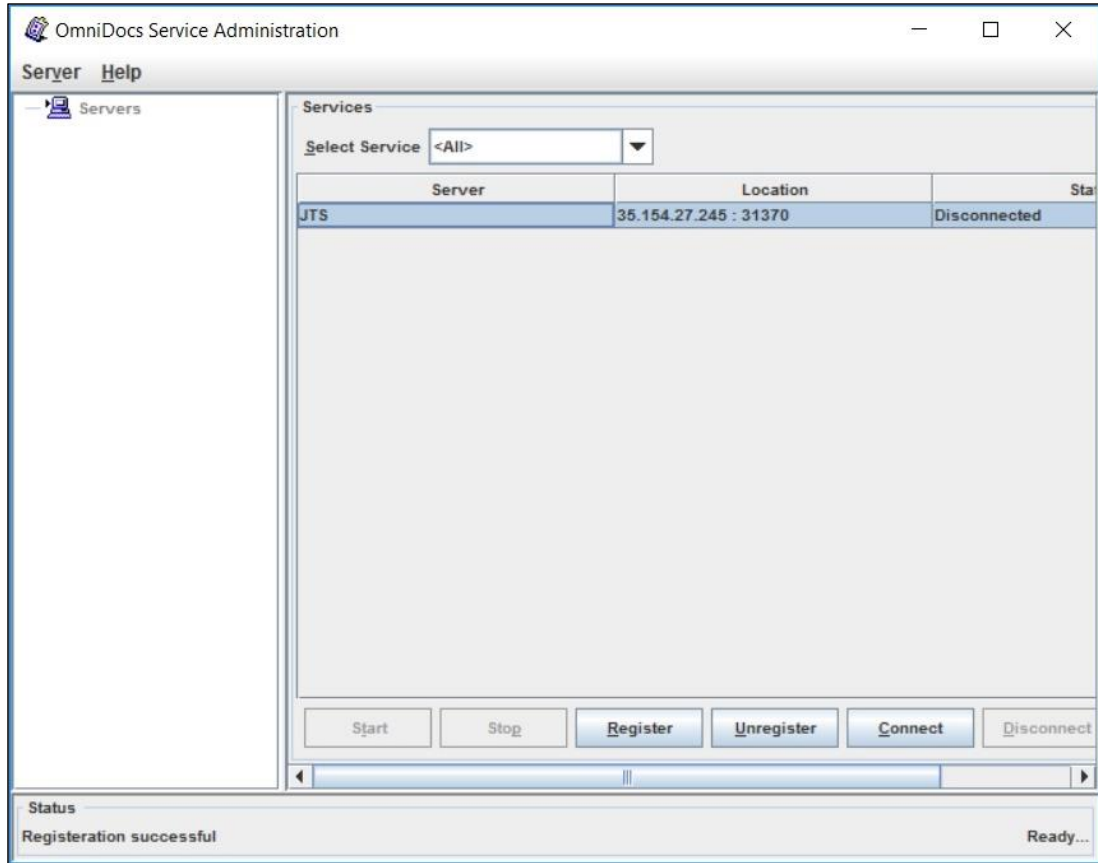


Figure 3.61

2. Select the registered JTS Server and click **Connect**. Once JTS is connected, the **Manage** button gets enabled.
3. Click **Manage** button, after clicking on the Manage button, an entry of the connected JTS server along with its IP Address is displayed on the upper-left panel in the repository view.
4. Select the JTS from the repository view. The list of already created and associated cabinets, appears.

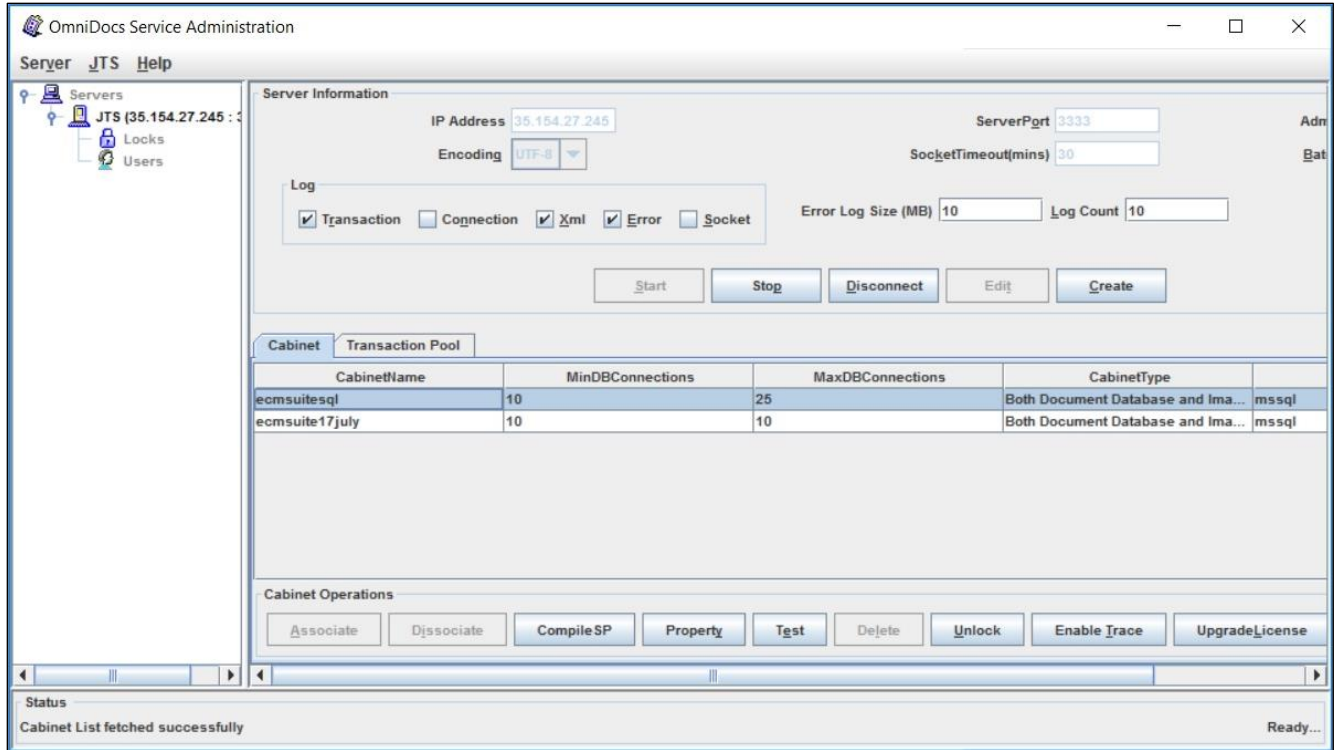


Figure 3.62

3.7.4 Creating a cabinet

Perform the below steps to create a cabinet:

For MSSQL:

1. Click **Create**. The Create Cabinet dialog box appears.

Create Cabinet (35.154.27.245 : 31370)

Cabinet Type

Document database Image Server database Both

Database Type

MSSQL / Amazon RDS Oracle Postgres Azure

MSSQL Information

Device Size (MB) Log Size (MB)

Cabinet information

Cabinet Name

Server Name

User name

Password

Database Path

CD Key

Security Level

Password Algorithm

Enable ETS

Status

OK Cancel

Figure 3.63

2. Select the cabinet type that needs to be created from the Cabinet Type area. The Cabinet can be a **Document database**, an **Image server database**, or both.
3. Select the database option from the Database Type section.
4. Specify the initial database size in the **Device Size** textbox and specify the initial log size in the **Log Size** textbox. Else, continue with the default values.
5. Specify the following cabinet information:
 - Specify the cabinet name in the **Cabinet Name** textbox.
 - Specify the server name (name of the machine where the MS SQL server is running) in the **Server I.P.** textbox.
 - Specify the username in the **User name** textbox.
 - Specify the password in the **Password** textbox.
 - Specify the CD key in the **CD Key** textbox.
 - Select the **Enable FTS** checkbox.

NOTE:

In the case of MSSQL if the Database port is not equal to 1433 (Default port) update the database port in the *DatabaseDriver.xml* file located inside the OmniDocs11.0Ejb/ngdbini folder at the mapped location on the worker node before creating the cabinet.

Figure 3.64

6. Click **OK** to create the cabinet. The Cabinet created successfully dialog appears.

For Aurora PostgreSQL:

1. Click **Create**. The Create Cabinet dialog box appears.

Figure 3.65

2. Select the cabinet type that needs to be created from the Cabinet Type area. The Cabinet can be a **Document database**, an **Image server database**, or both.
3. Select the database option from the Database Type section.
4. Specify the port number if default port 5432 is not used.
5. Specify the following cabinet information:
 - Specify the cabinet name in the **Cabinet Name** textbox.
 - Specify the Aurora PostgreSQL server name in the **Server I.P.** textbox.
 - Specify the username in the **User name** textbox.
 - Specify the password in the **Password** textbox.
 - Specify the CD key in the **CD Key** textbox.

Create Cabinet (13.127.66.151 : 30896) [X]

Cabinet Type

Document database
 Image Server database
 Both

Database Type

MSSQL / Amazon RDS
 Oracle
 Postgres
 Azure
 OracleRAC

Postgres Information

Port:

Cabinet information

Cabinet Name	<input type="text" value="ibpsaurora12dec"/>
Server I.P.	<input type="text" value="uster-cv4updtekwxu.ap-south-1.rds.amazonaws.com"/>
User name	<input type="text" value="postgres"/>
Password	<input type="password" value="••••••••"/>
Database Path	<input type="text" value="PGDATA"/>
CD Key	<input type="text" value="CK1E8v4m1507A P44H622v 7005ZnZ WwYiuQ1Z"/>
Security Level	<input type="text" value="Object Level"/> ▼
Password Algorithm	<input type="text" value="PC1"/> ▼

Status

OK Cancel

Figure 3.66

- Click **OK** to create the cabinet. The Cabinet created successfully dialog box appears.

3.7.5 Associating a cabinet

Perform the below steps to associate the cabinet:

For MSSQL:

1. Click **Stop** to enable the Associate button.
2. Click **Associate**. The Associate a Cabinet dialog appears with the following tabs:
 - i. **Database tab:** Select the database type.
 - ii. **Cabinet properties tab:** Specify the cabinet details that you have specified during cabinet creation.

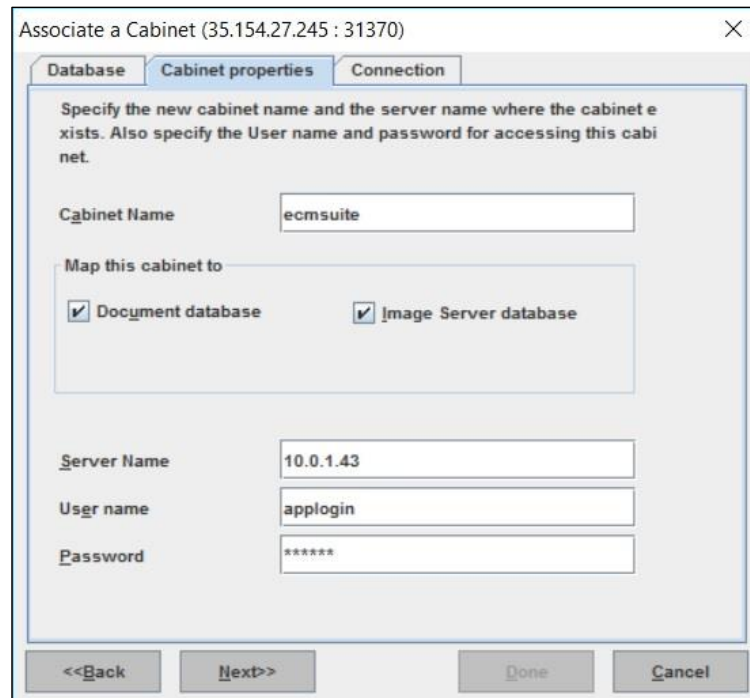


Figure 3.67

- iii. **Connection tab:** Specify the **maximum** and the **minimum** number of connections that the JTS must maintain with the database, specify the **query time** out for the selected cabinet in the Query timeout text box and specify the **refresh interval** time for connection.

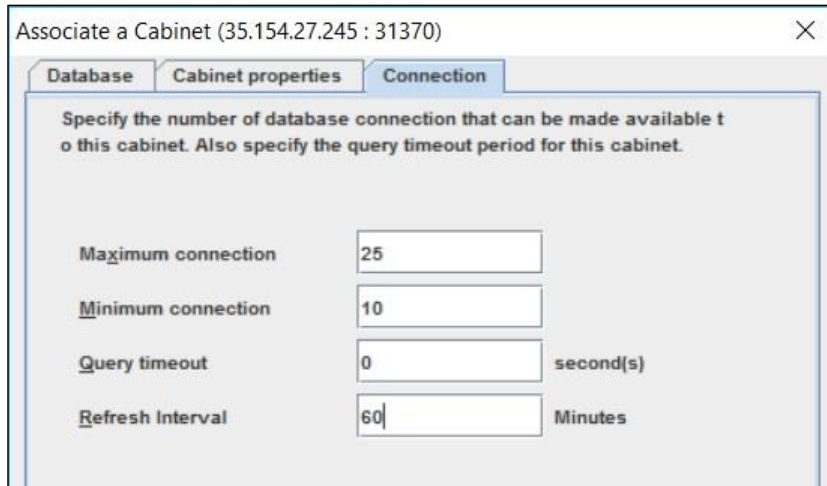


Figure 3.68

3. Click **Done** to associate the selected cabinet. Once the cabinet is associated successfully, it appears with the list.

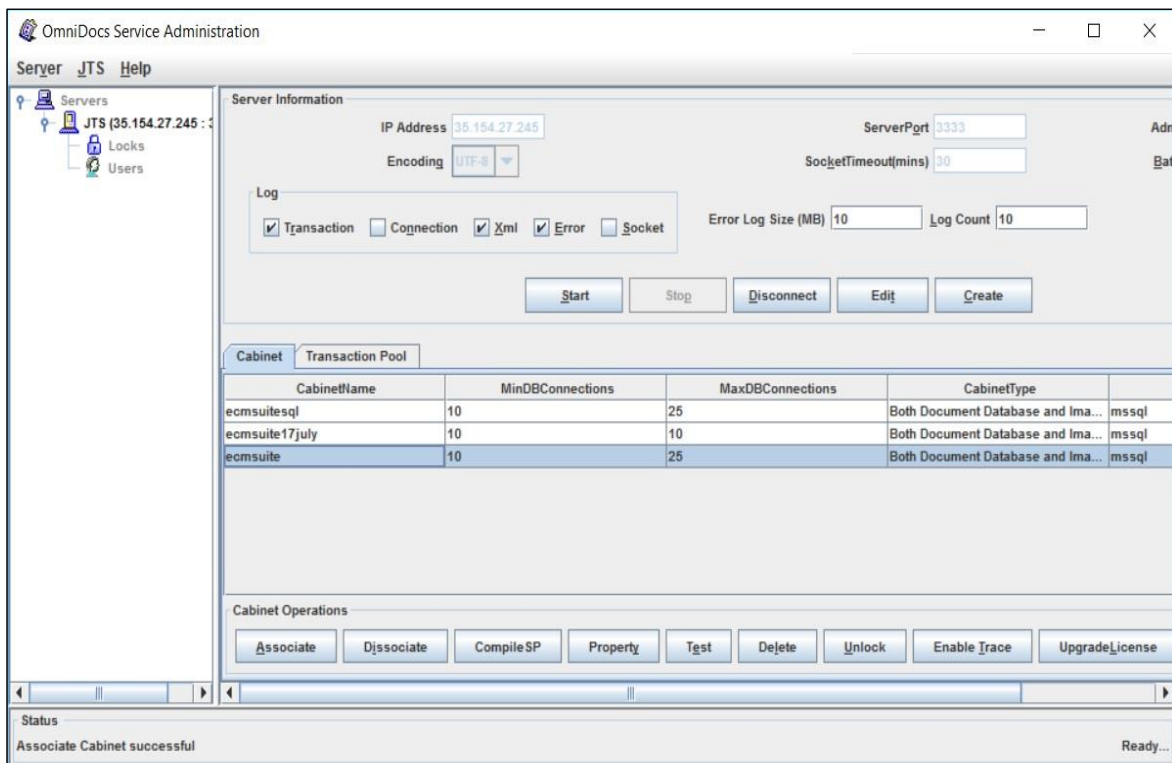


Figure 3.69

For Aurora PostgreSQL:

1. Click **Stop** to enable the Associate button.
2. Click **Associate**. The Associate a Cabinet dialog appears with the following tabs:

- i. **Database tab:** Select the database type.
- ii. **Cabinet properties tab:** Specify the cabinet details that you have specified during cabinet creation.

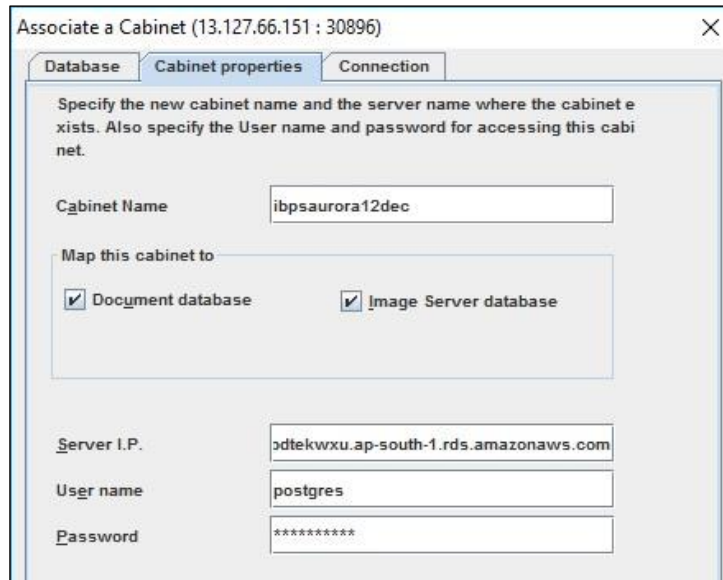


Figure 3.70

- iii. **Connection tab:** Specify the **maximum** and the **minimum** number of connections that the JTS must maintain with the database. Also, specify the **query time** out for the selected cabinet in the Query timeout text box and specify the **refresh interval** time for connection.

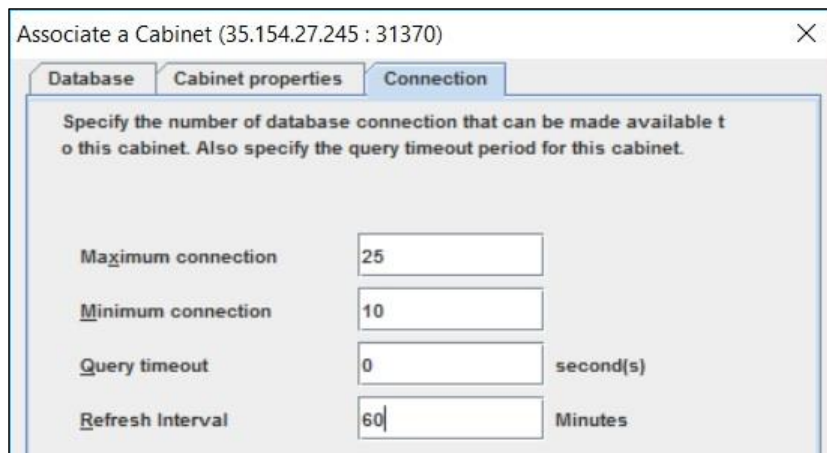


Figure 3.71

- 3. Click **Done** to associate the selected cabinet. Once the cabinet is associated successfully, it appears with the list.

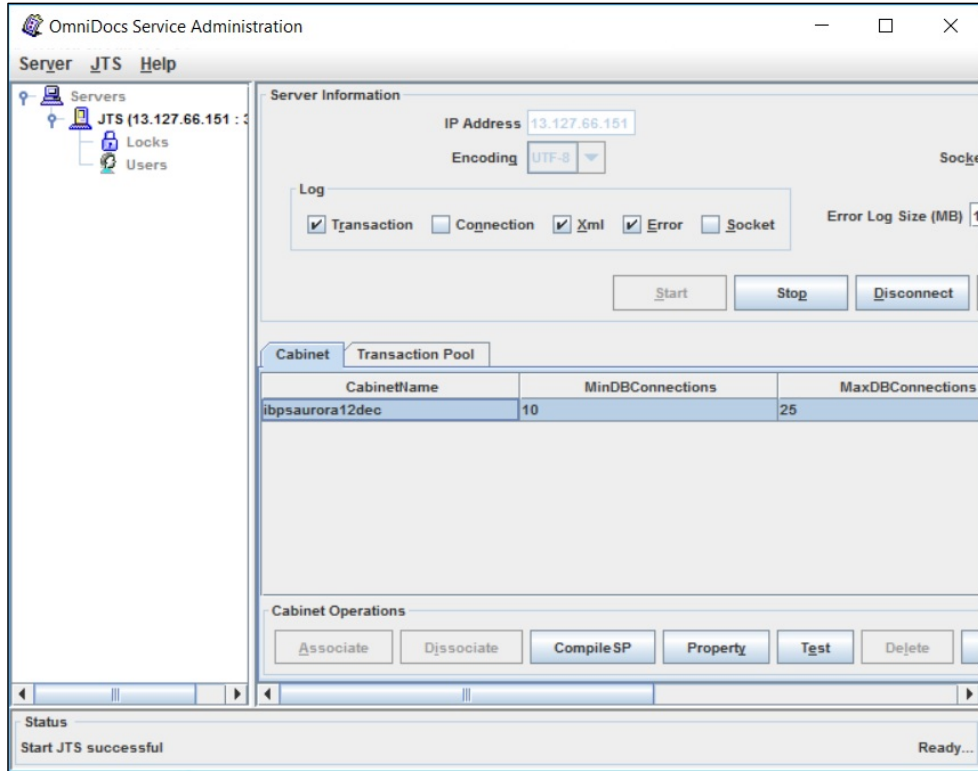


Figure 3.72

3.7.6 Creating a data source

Perform the below steps to create the data source:

For MSSQL:

1. Open the <Host-Path URL of OmniDocs+RMS EJB container> like *http://ecmsuiteconsole.aws.co.in* as defined in the *AWS_ALB-IngressController.yml* file. It automatically redirects to the JBoss EAP 7.4 Admin console.
2. Enter the newgen as username and password system123# respectively to login to the Admin console. After a successful login, the Red Hat JBoss Enterprise Application Platform screen appears.

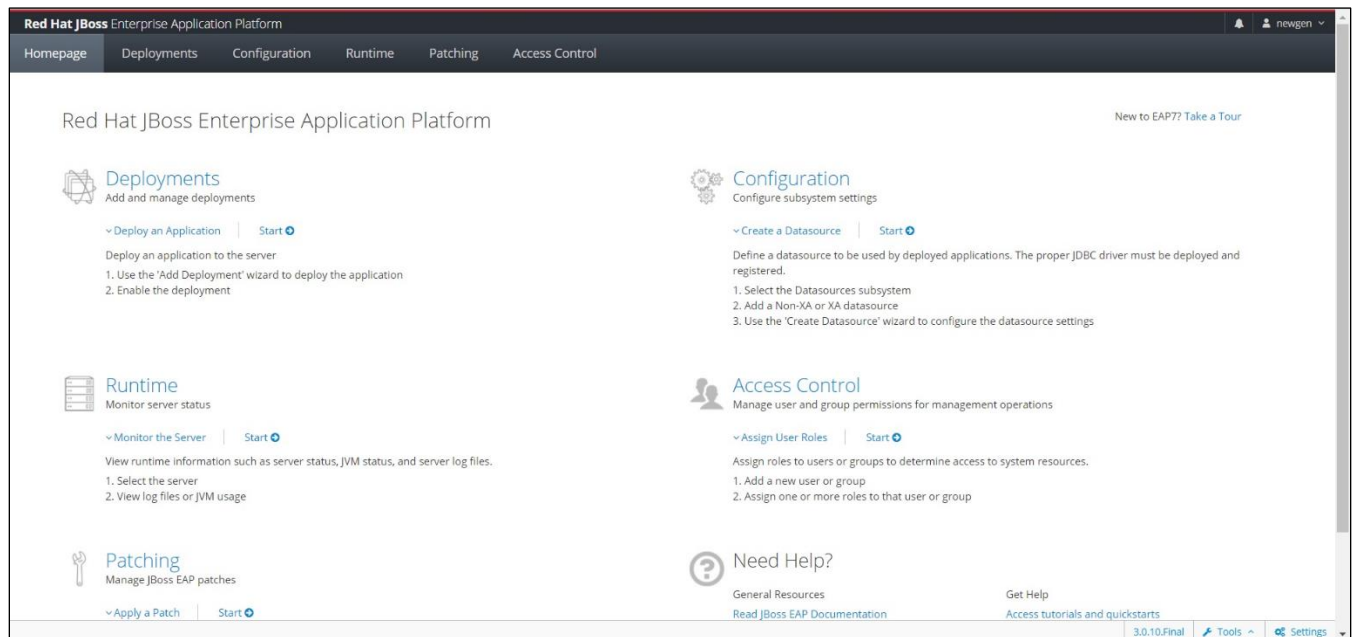


Figure 3.73

3. Go to the **Subsystems** in the Configuration tab.
4. Go to the **Datasources and Drivers**. Then, click Datasources.

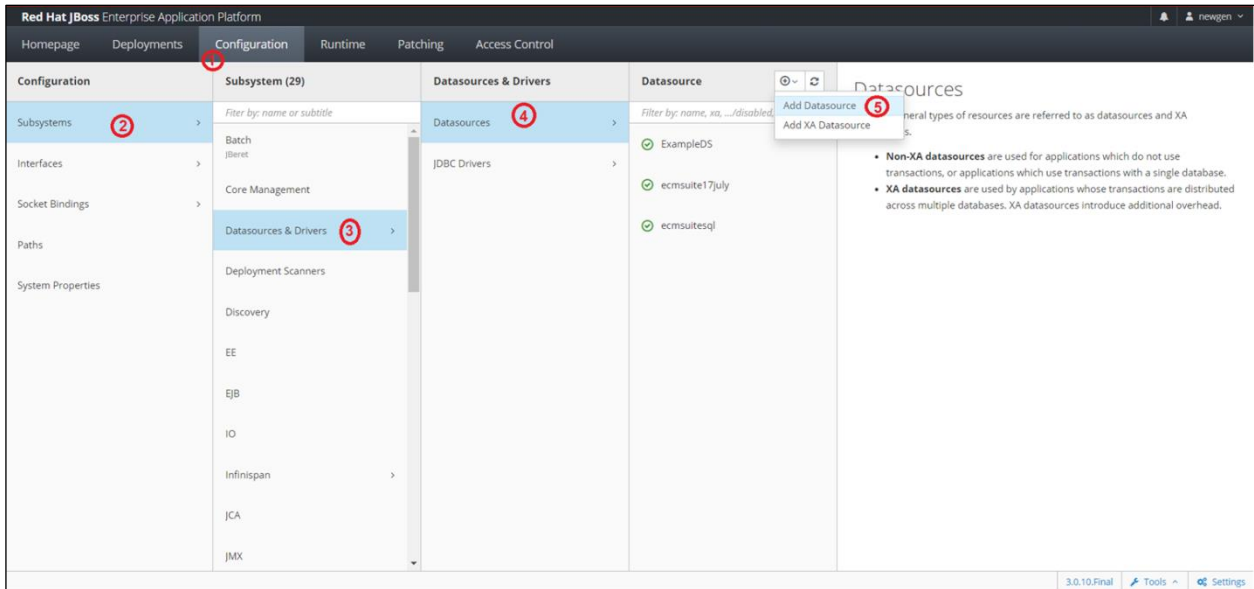


Figure 3.74

5. Click Plus + icon and select **Add Datasource**. The Add Datasource dialog appears.
6. For MSSQL Database Server, select **Microsoft SQLServer** and click **Next**.

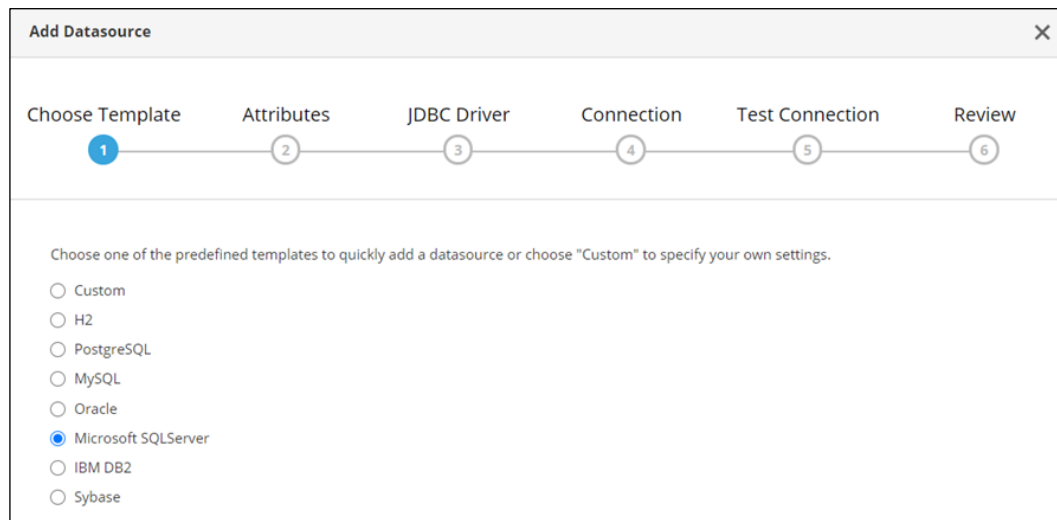


Figure 3.75

7. Provide a DataSource Name and JNDI Name.
 - **Name:** Enter the OmniDocs cabinet name that is cabinet name.
 - **JNDI Name:** java:/same as OmniDocs cabinet name
8. Click **Next**.

Figure 3.76

9. Select JDBC Driver Name.
10. For MSSQL, select **sqljdbc42.jar**.
11. Clear **Drive Module Name** and **Driver Class Name** textboxes.
12. Click **Next**.

Figure 3.77

13. Provide the following Connection Setting details and click **Next**:
 - **Connection URL:**
jdbc:sqlserver://MSSQL_Server_IP:MSSQL_Server_Port;databaseName=CABINET_NAME
 - **UserName:** Enter the SQL Server User Name
 - **Password:** Enter the SQL Server Password
 - **Security Domain:** Keep this blank.

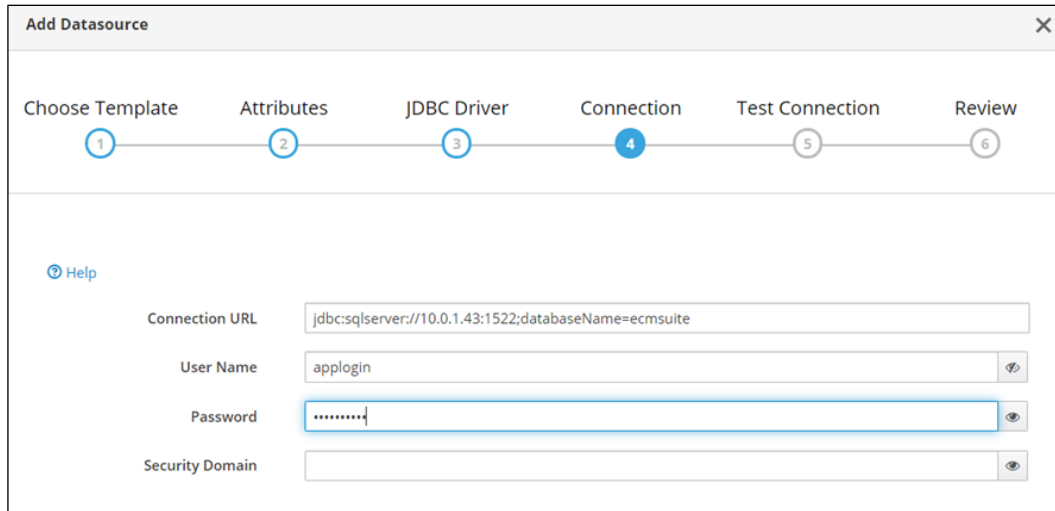


Figure 3.78

14. Click **Next** on the **Test Connection** page.

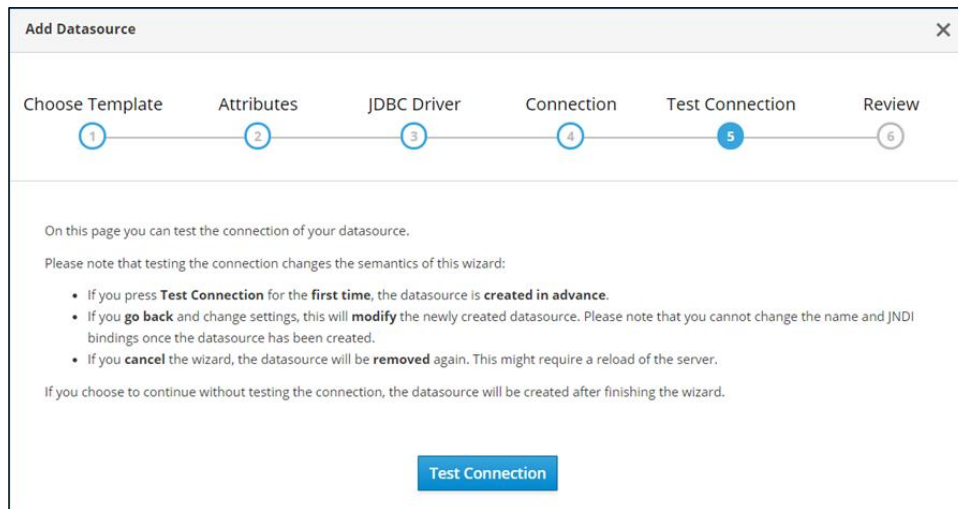


Figure 3.79

15. Click **Finish**. After the creation of the datasource, a success message appears.

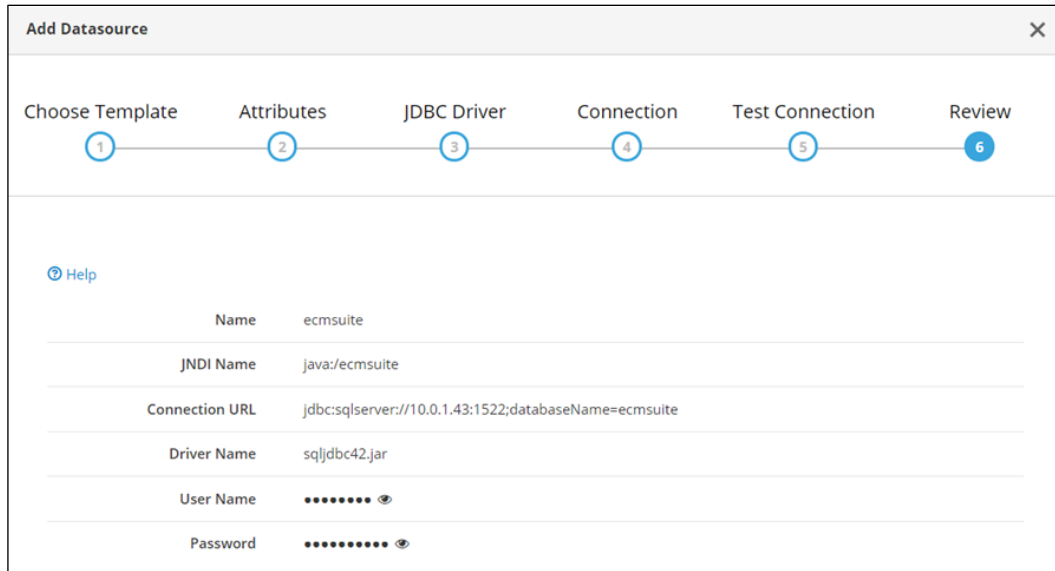


Figure 3.80

16. Click **View Datasource** to view the created datasource. The created datasource appears in the list of **Datasource**.
17. Click **View** against the datasource. A screen appears with the attributes of the datasource appears.
18. Click **Edit** link.

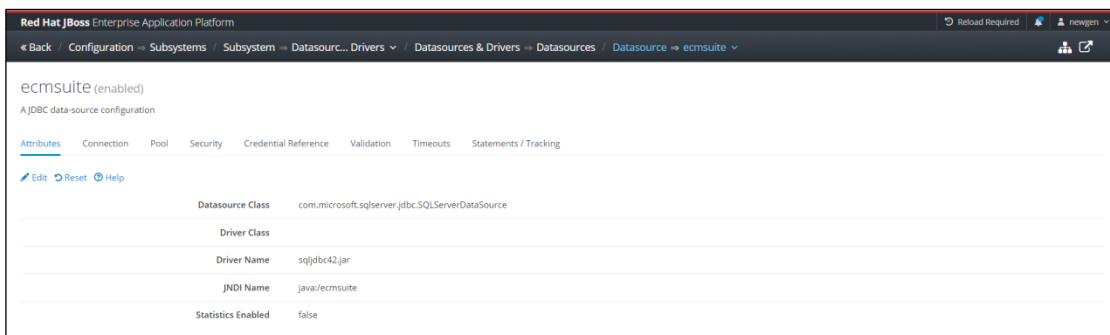


Figure 3.81

19. Clear the **Datasource Class** textbox and click **Save**.

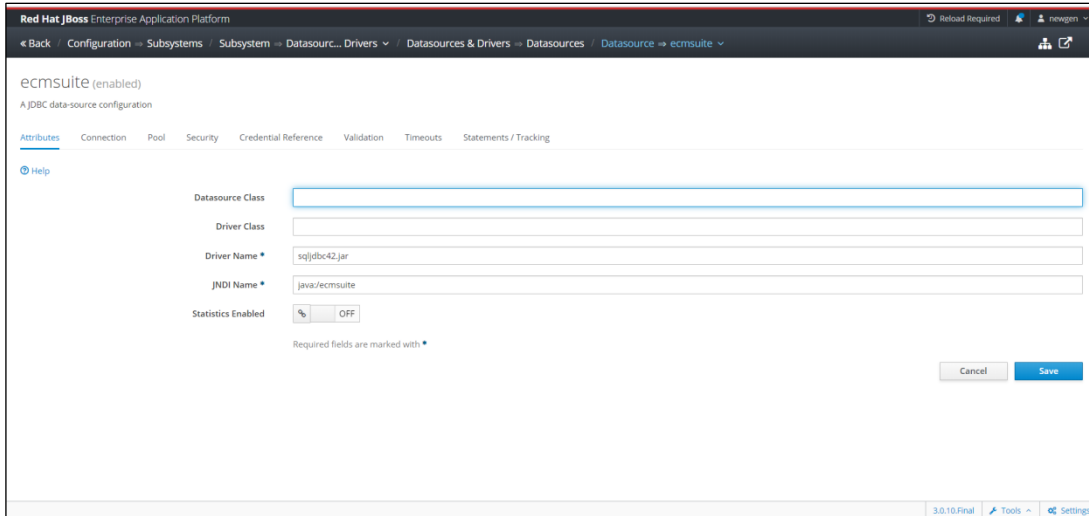


Figure 3.82

20. After that restart the OmniDocs+RMS EJB container.
21. Once the OmniDocs+RMS EJB container is restarted, open the JBossEAP Admin console once again.
22. Go to the **Subsystems** in the Configuration tab.
23. Go to the **Datasources and Drivers**. Then, click Datasources.
24. Select the created data source and click **Test connection** from the dropdown list.

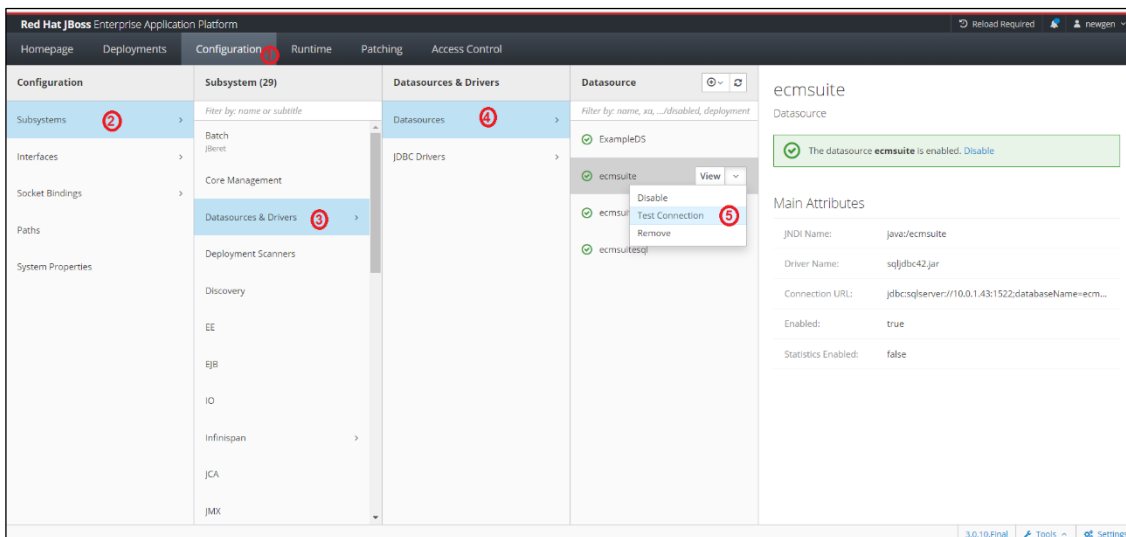


Figure 3.83

On the successful data connection, a success message appears.

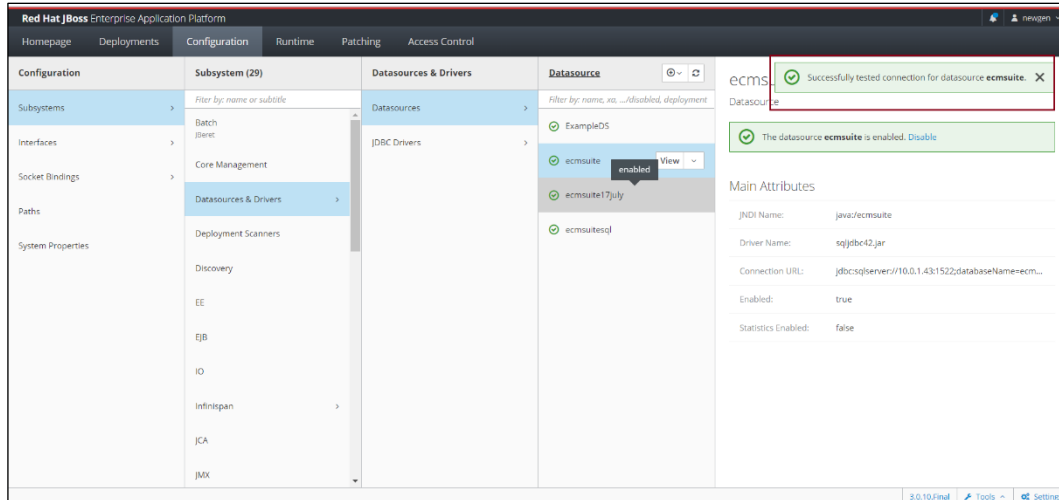


Figure 3.84

25. Add the below connection pool setting and idle-connection-timeout setting inside the created DataSource in *standalone.xml* file located inside the **OmniDocs11.0Ejb** or **configuration** folder at the mapped location on the worker node.

```

<pool>
  <min-pool-size>100</min-pool-size>
  <initial-pool-size>100</initial-pool-size>
  <max-pool-size>600</max-pool-size>
  <flush-strategy>Gracefully</flush-strategy>
</pool>

<timeout>
  <idle-timeout-minutes>5</idle-timeout-minutes>
</timeout>

```

For example,

```

<datasource jndi-name="java:/auroraod23oct1" pool-name="auroraod23oct1">
  <connection-url>jdbc:
    postgresql://omnidocs-auroraod23oct1-jca-aws-ec2-ore-cluster-cluster-cv4updkwku-ap-south-1-rds.amazonaws.com:5432/auroraod23oct1
  </connection-url>
  <driver>postgresql-42.2.18.jar</driver>
  <pool>
    <min-pool-size>0</min-pool-size>
    <initial-pool-size>0</initial-pool-size>
    <max-pool-size>600</max-pool-size>
    <flush-strategy>Gracefully</flush-strategy>
  </pool>
  <security>
    <user-name>postgres</user-name>
    <password>[REDACTED]</password>
  </security>
  <validation>
    <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker"/>
    <background-validation>true</background-validation>
    <background-validation-millis>1000</background-validation-millis>
    <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter"/>
  </validation>
  <timeout>
    <idle-timeout-minutes>5</idle-timeout-minutes>
  </timeout>
</datasource>

```

Figure 3.85

26. Restart the **OmniDocs+RMS EJB** container once again.

For Aurora PostgreSQL:

1. Open the *<Host-Path URL of OmniDocs+RMS EJB container>* like *http://ecmsuiteconsole.aws.co.in* as defined in the *AWS_ALB-IngressController.yml* file. It automatically redirects to the JBossEAP 7.4 Admin console.
2. Specify the newgen as username and system123# as password respectively to login to the Admin console.

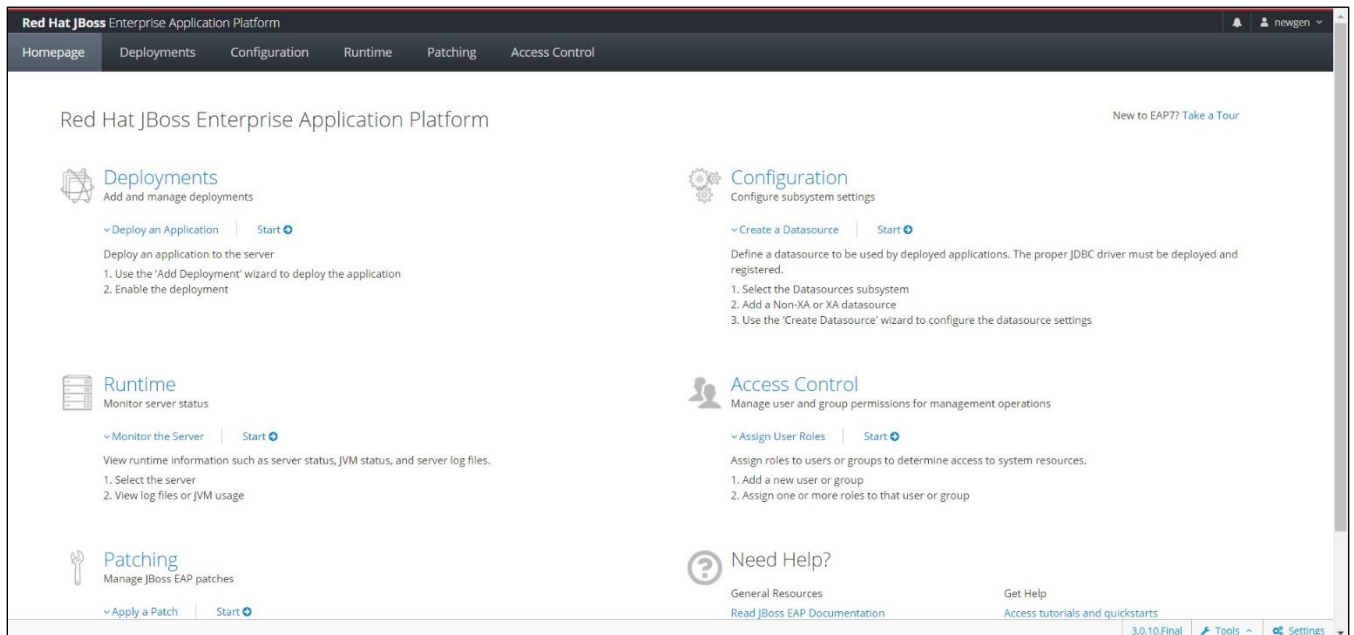


Figure 3.86

3. Go to the **Subsystems** in the Configuration tab.
4. Go to the **Datasources and Drivers**. Then, click Datasources.
5. Click Plus **+** icon and select **Add Datasource**. The Add Datasource dialog appears.

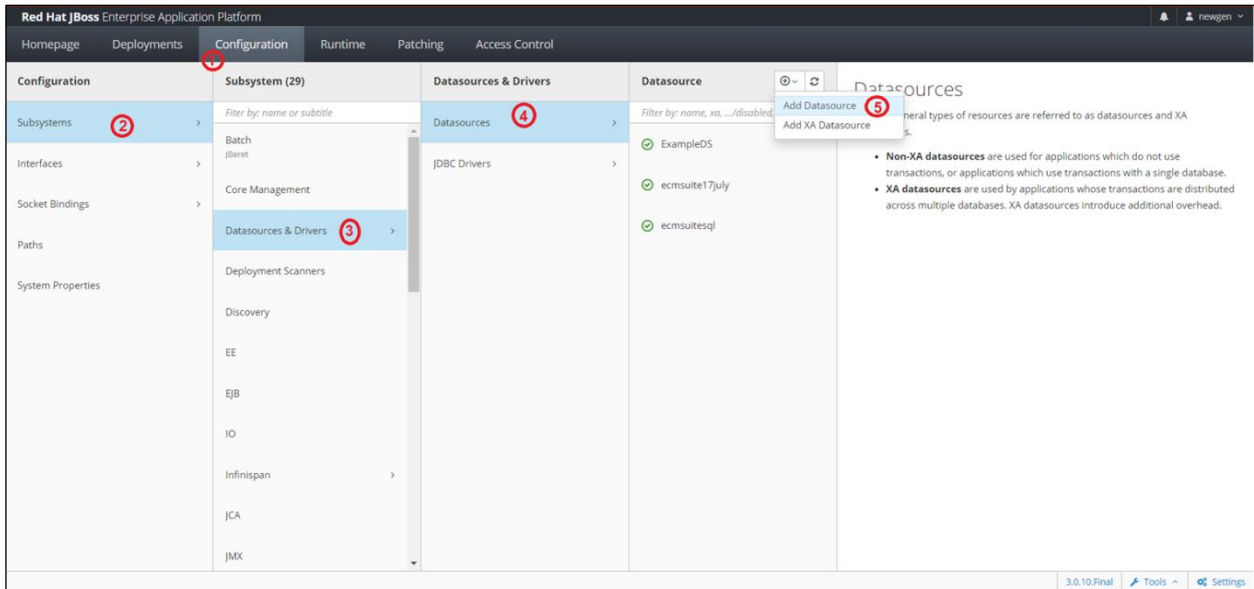


Figure 3.87

6. For the Aurora PostgreSQL Database Server, select **PostgreSQL** and click **Next**.

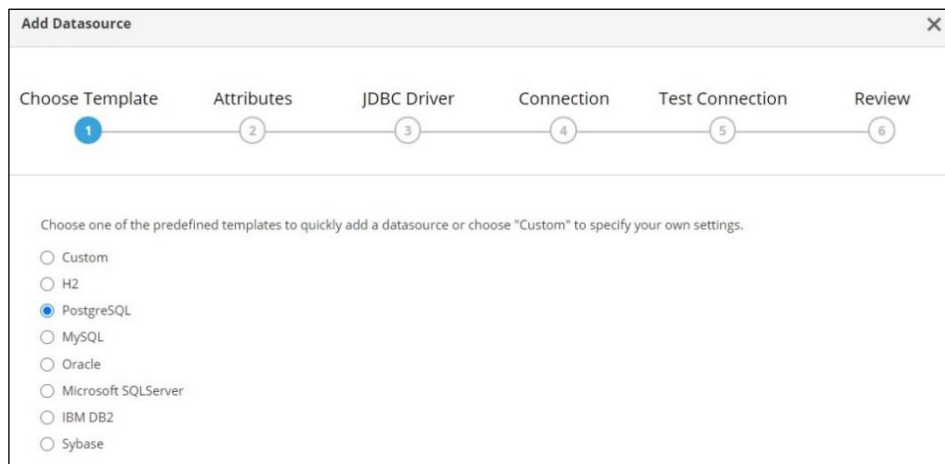


Figure 3.88

7. Provide a DataSource Name and JNDI Name.

- **Name:** Enter same as OmniDocs cabinet name.
- **JNDI Name:** java:/same as OmniDocs cabinet name.

8. Click **Next**.

9. Select JDBC Driver Name.

10. For Aurora PostgreSQL, select **postgresql-42.5.0.jar**.

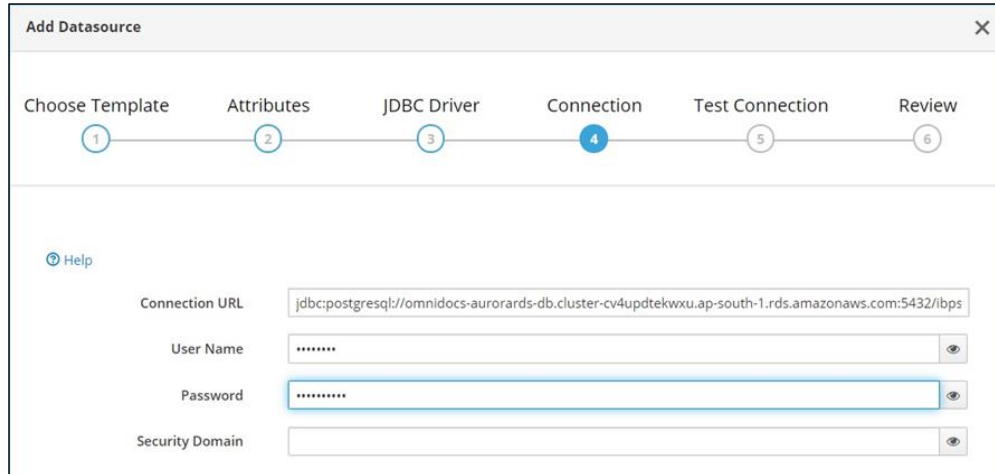
11. Clear **Driver Module Name** and **Driver Class Name** textboxes and click **Next**.

12. Provide the following Connection Setting details and click **Next**:

- **Connection URL:**

jdbc:postgresql://AuroraPostgreSQL_Server_IP:AuroraPostgreSQL_Server_Port/CABINET_NAME

- **UserName:** AuroraPostgreSQL Server User Name
- **Password:** AuroraPostgreSQL Server Password
- **Security Domain:** Keep this blank.



The screenshot shows a wizard window titled "Add Datasource" with a close button (X) in the top right corner. At the top, there is a progress bar with six steps: "Choose Template" (1), "Attributes" (2), "JDBC Driver" (3), "Connection" (4), "Test Connection" (5), and "Review" (6). Step 4, "Connection", is currently selected and highlighted in blue. Below the progress bar, there is a "Help" icon and a list of fields for configuration:

- Connection URL:** jdbc:postgresql://omnidocs-aurorards-db.cluster-cv4updtekwxu.ap-south-1.rds.amazonaws.com:5432/lbps
- User Name:** A text input field containing seven asterisks (password masked).
- Password:** A text input field containing seven asterisks (password masked).
- Security Domain:** An empty text input field.

Figure 3.89

13. Click **Next** on the **Test Connection** page.
14. Click **Finish**. After the creation of the datasource, a success message appears.

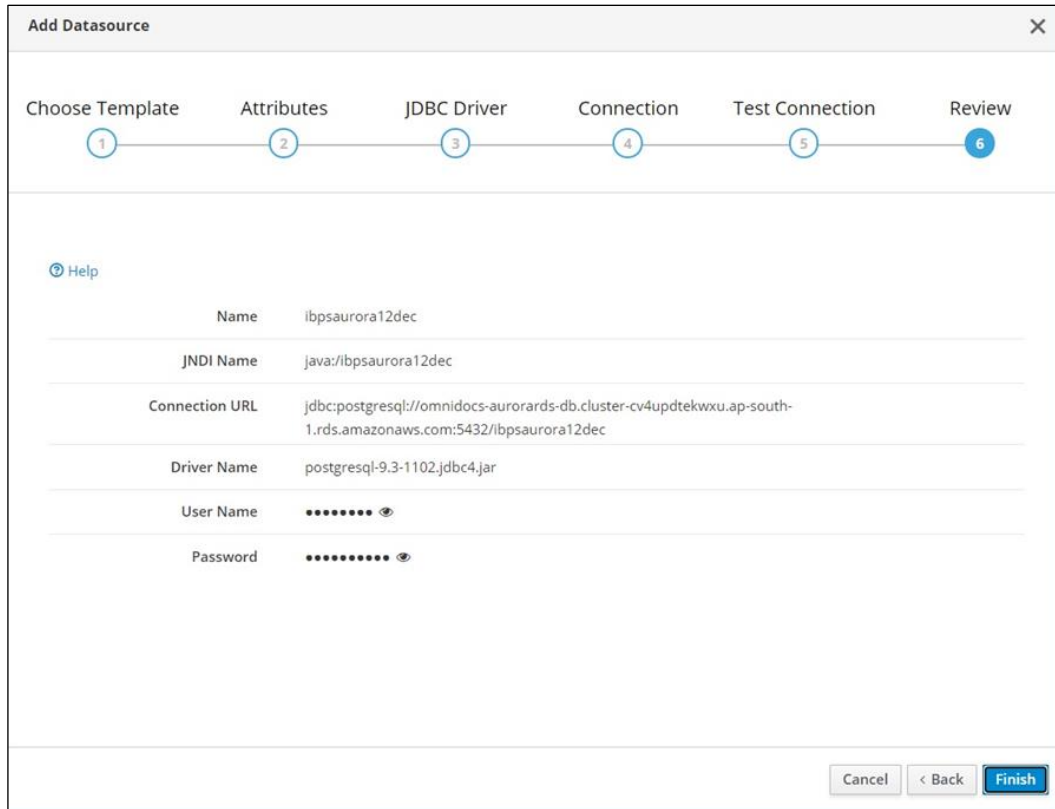


Figure 3.90

15. Click **View Datasource** to view the created datasource. The created datasource appears in the list of **Datasource**.
16. Click **View** against the datasource. A screen appears with the attributes of the datasource.
17. Click **Edit** link and clear the **Datasource Class** textbox.

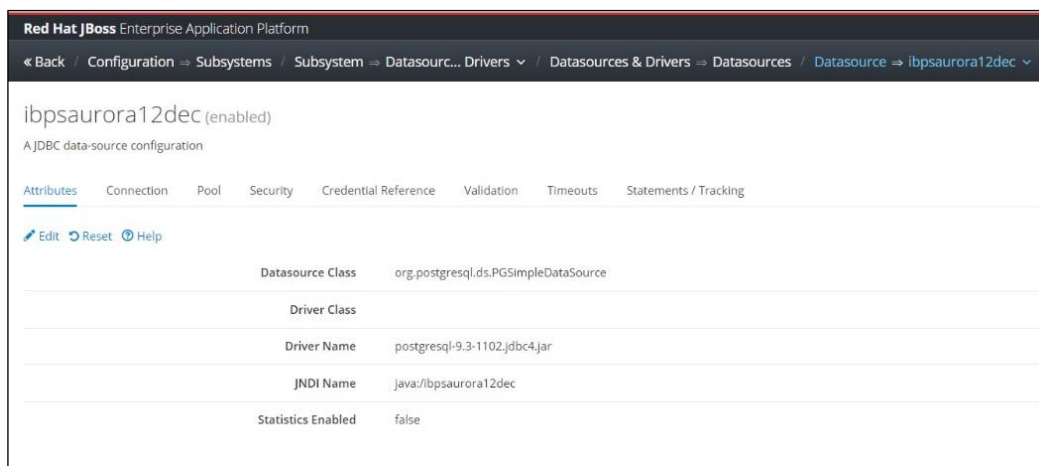


Figure 3.91

18. Click **Save**. After that restart the OmniDocs+RMS EJB container.

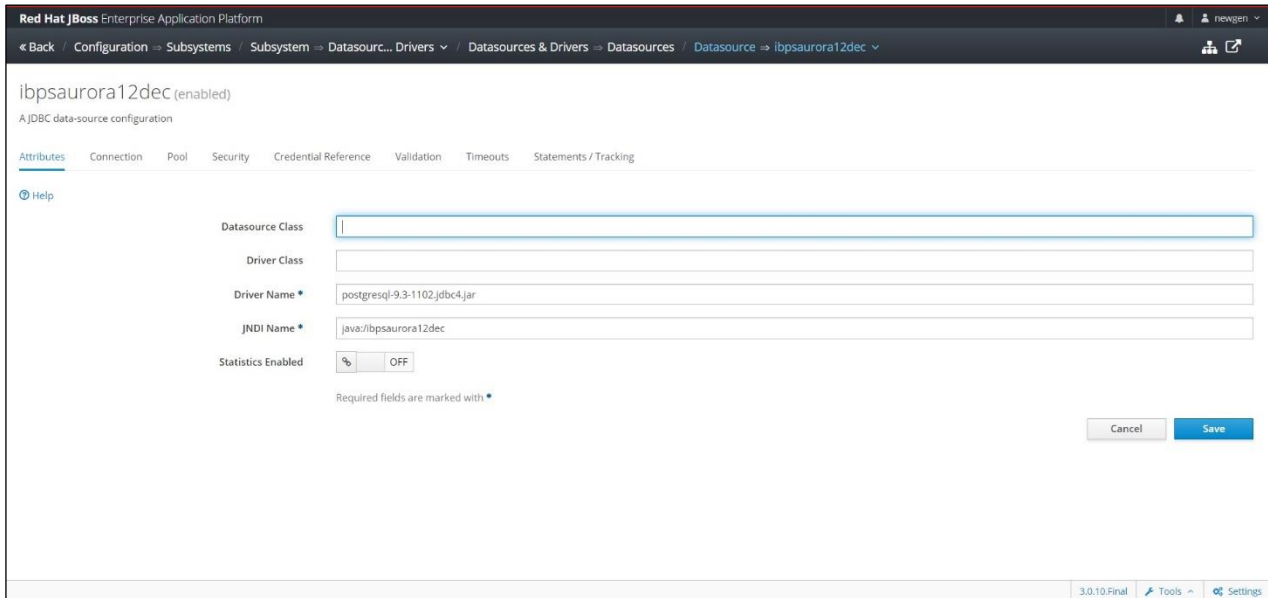


Figure 3.92

19. Once the OmniDocs+RMS EJB container is restarted, open the JBossEAP Admin console once again.

20. Go to the **Subsystems** in the Configuration tab.

21. Go to the **Datasources and Drivers**. Then, click Datasources.

22. Select the created data source and click **Test connection** from the dropdown list.

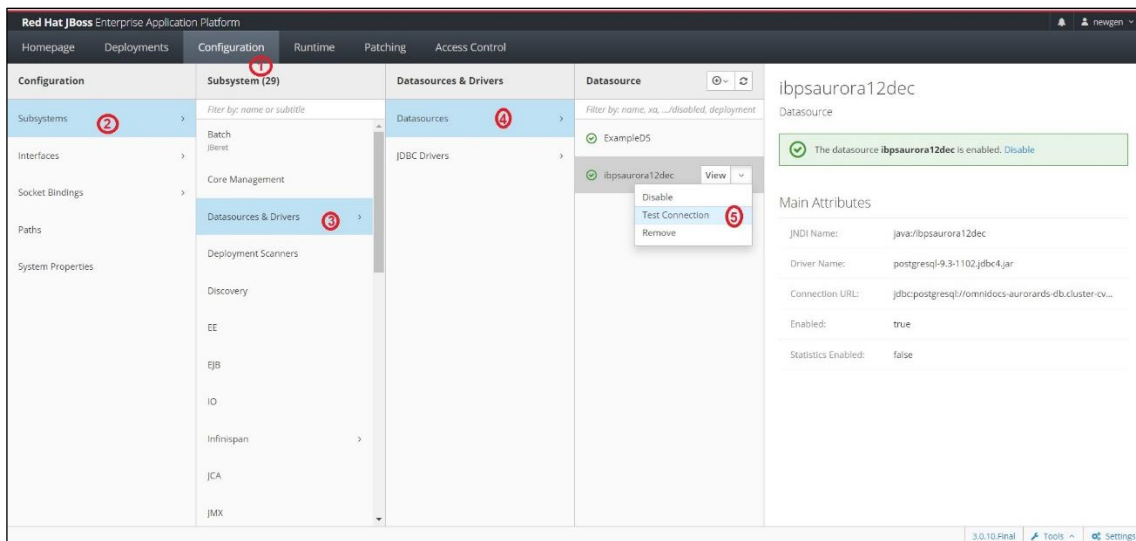


Figure 3.93

On a successful data connection, a success message appears.

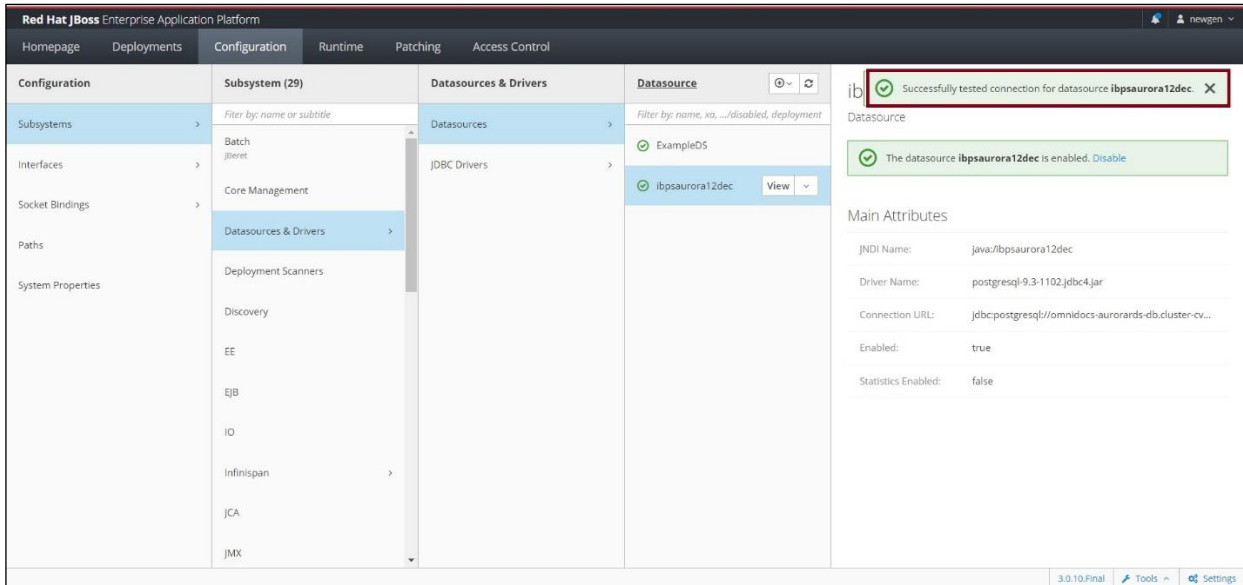


Figure 3.94

23. Add the below connection pool setting and idle-connection-timeout setting inside the created DataSource in *standalone.xml* file located inside the **OmniDocs11.0Ejb** or **configuration** folder at the mapped location on the worker node.

```
<pool>
    <min-pool-size>0</min-pool-size>
    <initial-pool-size>0</initial-pool-size>
    <max-pool-size>600</max-pool-size>
    <flush-strategy>Gracefully</flush-strategy>
</pool>

<timeout>
    <idle-timeout-minutes>5</idle-timeout-minutes>
</timeout>
```

For example,

```

<datasource jndi-name="java:/auroraod23oct1" pool-name="auroraod23oct1">
  <connection-url>jdbc:
  postgresql://omnidocs-auroraod23oct1-jdbcpool-1-ore-cluster.cluster-cv4updtekwxu.ap-south-1.rds.amazonaws.com:5432/auroraod23oct1
  </connection-url>
  <driver>postgresql-42.2.18.jar</driver>
  <pool>
    <min-pool-size>0</min-pool-size>
    <initial-pool-size>0</initial-pool-size>
    <max-pool-size>600</max-pool-size>
    <flush-strategy>Gracefully</flush-strategy>
  </pool>
  <security>
    <user-name>postgres</user-name>
    <password>postgres</password>
  </security>
  <validation>
    <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker"/>
    <background-validation>true</background-validation>
    <background-validation-millis>10000</background-validation-millis>
    <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter"/>
  </validation>
  <timeout>
    <idle-timeout-minutes>5</idle-timeout-minutes>
  </timeout>
</datasource>

```

Figure 3.95

24. Restart the **OmniDocs+RMS EJB** container once again.

3.7.7 Registering a cabinet in OmniDocs

Perform the below steps to register a cabinet:

1. Register the cabinet for OmniDocs Admin using the following URL:

[http://<Host-Path URL of OmniDocsWeb](http://<Host-Path URL of OmniDocsWeb container>/omnidocs/admin/main/registration/registration.jsp)

[container>/omnidocs/admin/main/registration/registration.jsp](http://<Host-Path URL of OmniDocsWeb container>/omnidocs/admin/main/registration/registration.jsp)

For example,

<http://ecmsuite.aws.co.in/omnidocs/admin/main/registration/registration.jsp>

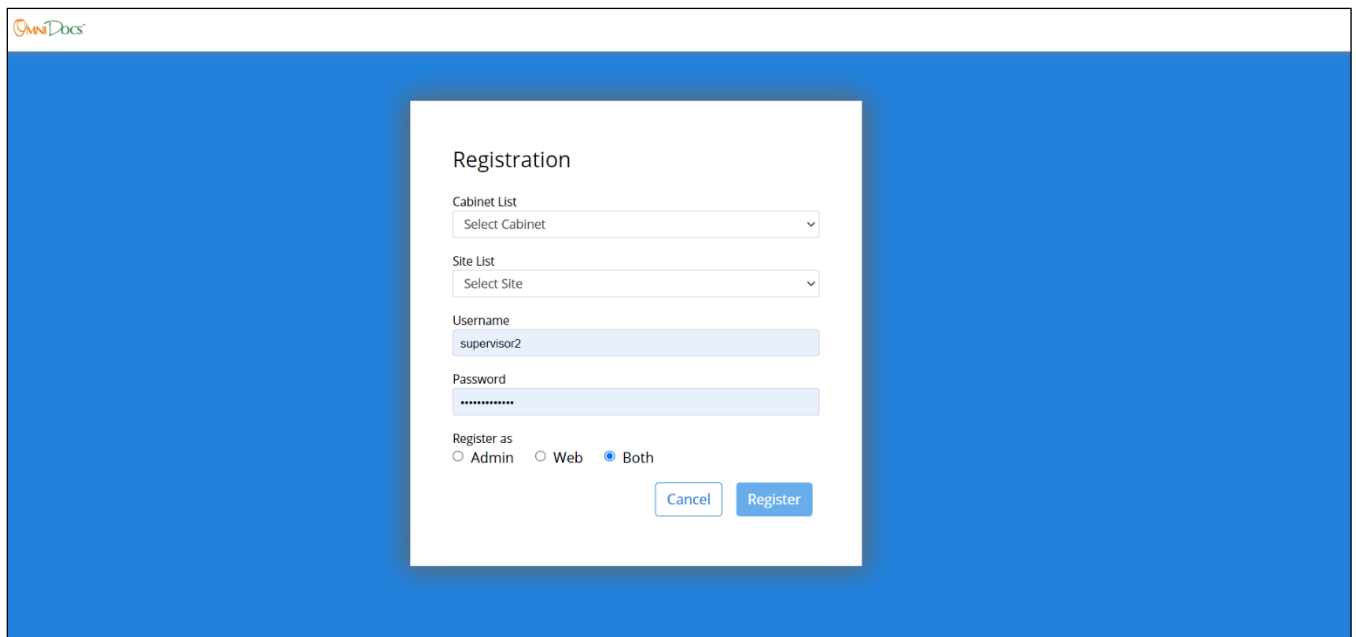


Figure 3.96

All the created cabinets get auto populated in the **Cabinet List** dropdown list.

2. Select the required cabinet, select the associated site, and specify the **Username** and **Password**.
3. Select the Register as **Both** and click **Register**. After successful registration, a confirmation message appears.

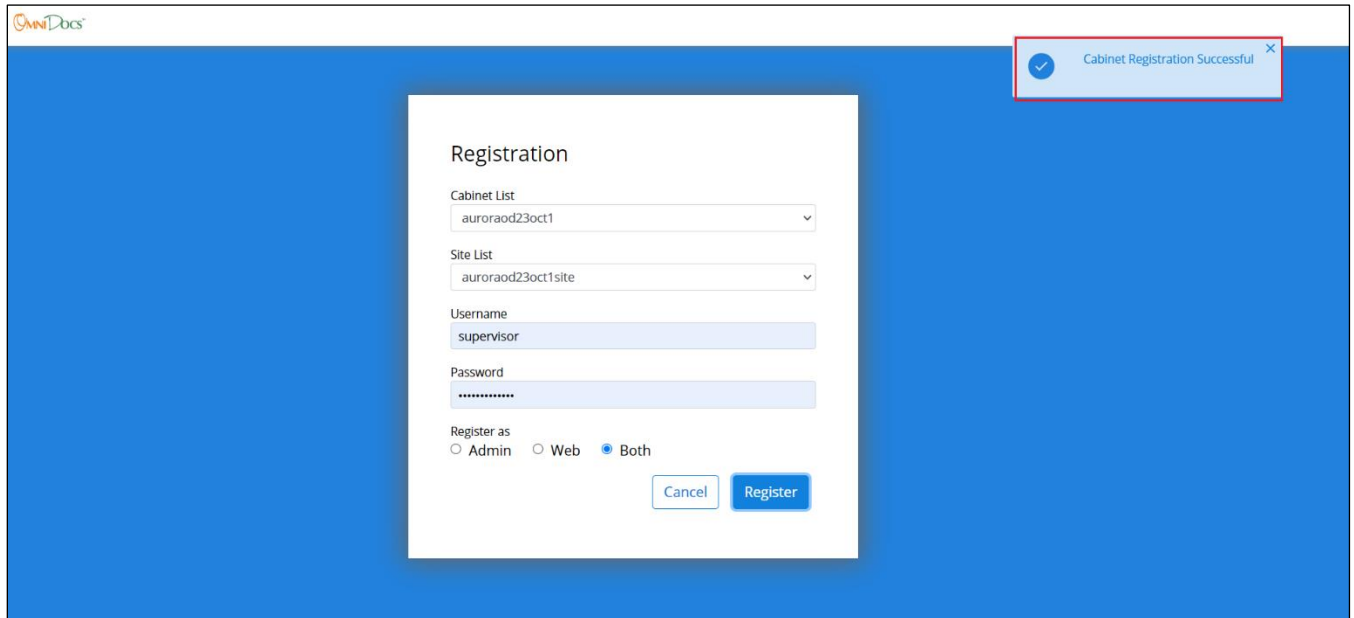
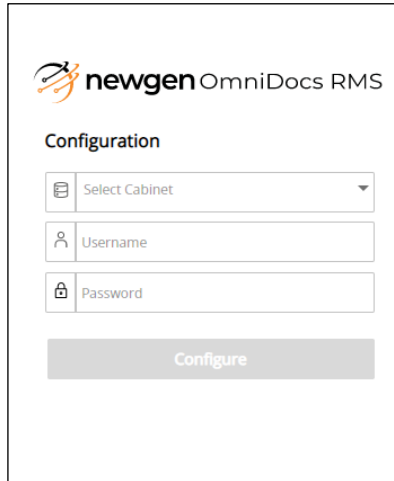


Figure 3.97

3.7.8 Registering a cabinet in RMS

Perform the below steps to register a cabinet in RMS:

1. Configure the cabinet for RMS using the following URL:
http://<Host-Path URL of OmniDocsWeb container>/rms/config
For example, *http://ecmsuite.aws.co.in/rms/config*

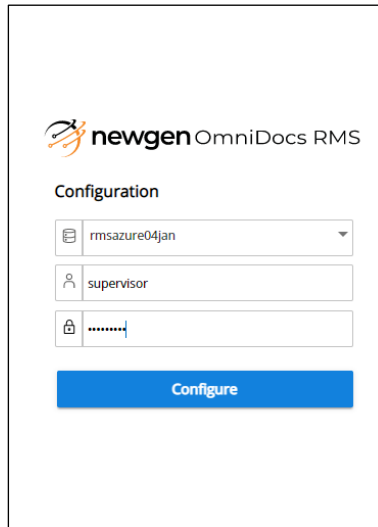


The screenshot shows the 'newgen OmniDocs RMS' configuration interface. Under the 'Configuration' heading, there are three input fields: a dropdown menu labeled 'Select Cabinet', a text field labeled 'Username', and a password field labeled 'Password'. A grey 'Configure' button is positioned below these fields.

Figure 3.98

All the created cabinets get auto populated in the Cabinet List dropdown list.

2. Select the required cabinet, specify the **Username** and **Password** and click **Configure**.



The screenshot shows the same RMS configuration interface as Figure 3.98, but with the fields populated. The 'Select Cabinet' dropdown now shows 'rmsazure04jan'. The 'Username' field contains 'supervisor' and the 'Password' field contains a masked password '.....'. The 'Configure' button is now blue.

Figure 3.99

After successful registration of cabinet, a confirmation message appears.

3.7.9 Creating site and volume

Perform the below steps to create site and volume:

1. Login to the OmniDocs Admin using the following URL:

`http://<Host-Path URL of OmniDocsWeb container>/omnidocs/admin`

For example,

`http://ecmsuite.aws.co.in/omnidocs/admin`

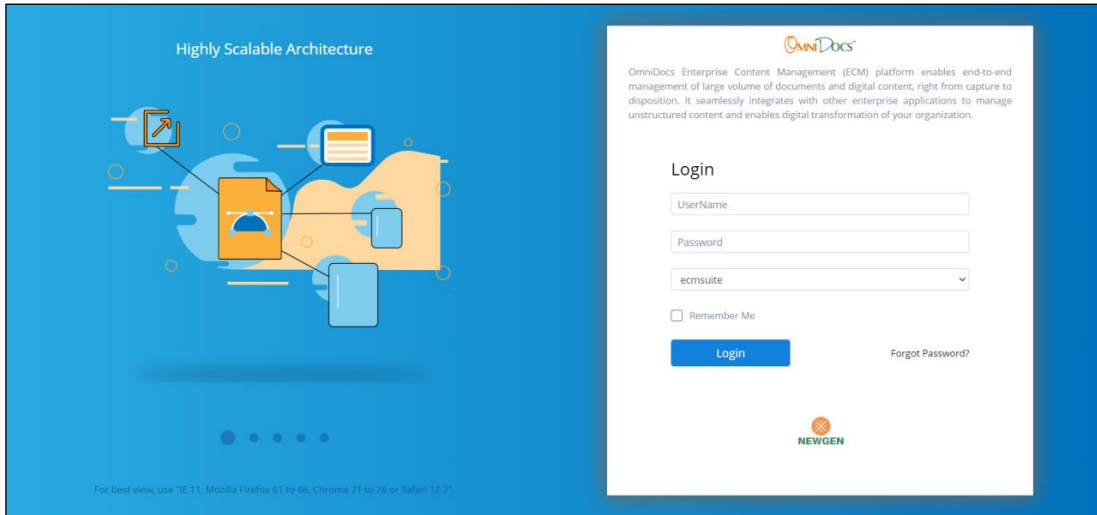


Figure 3.100

2. After a successful login, click **Sites** link under **Administration**.

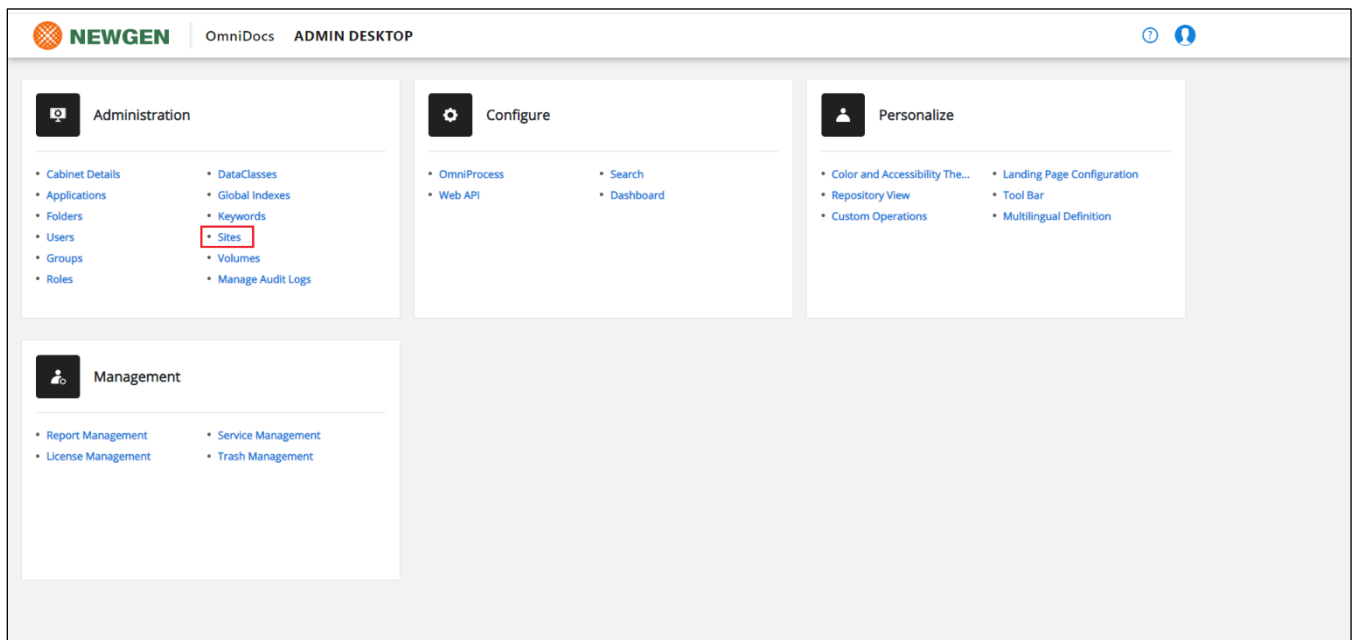


Figure 3.101

3. Click **+Add**. The Add Site dialog appears.

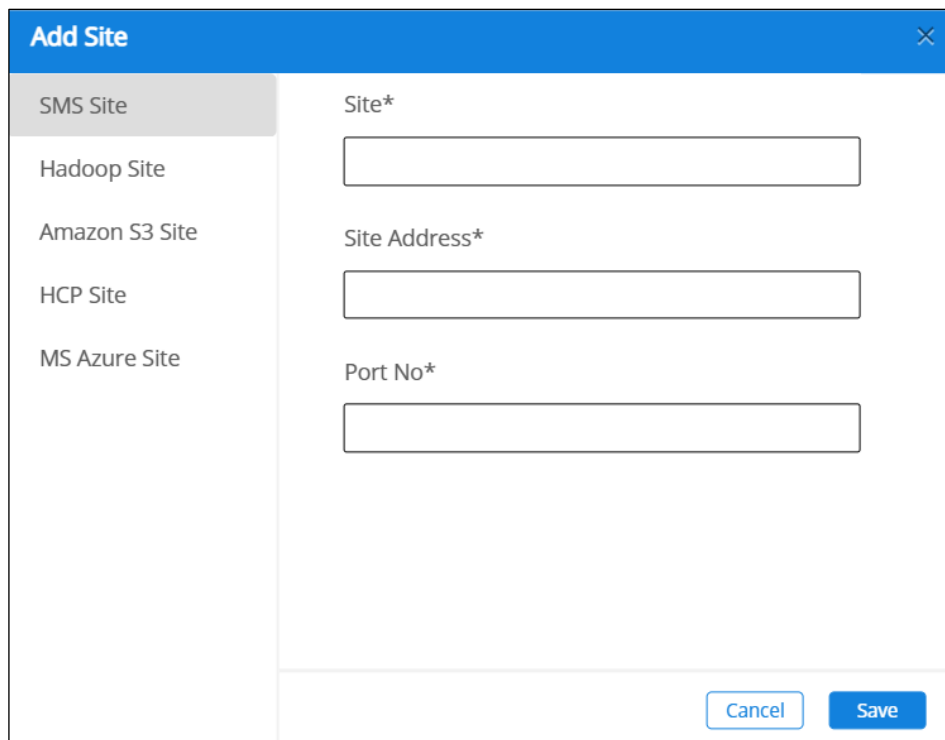


Figure 3.102

4. Click **Amazon S3 Site**.
5. Specify the user-defined site name, **Access Key**, and **Secret Key** that have rights to the S3 bucket.
6. Click **Save**.

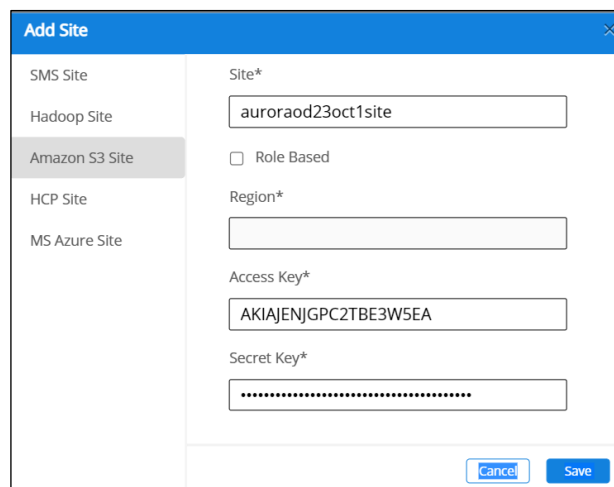


Figure 3.103

The added Site appears under Sites in the left pane.

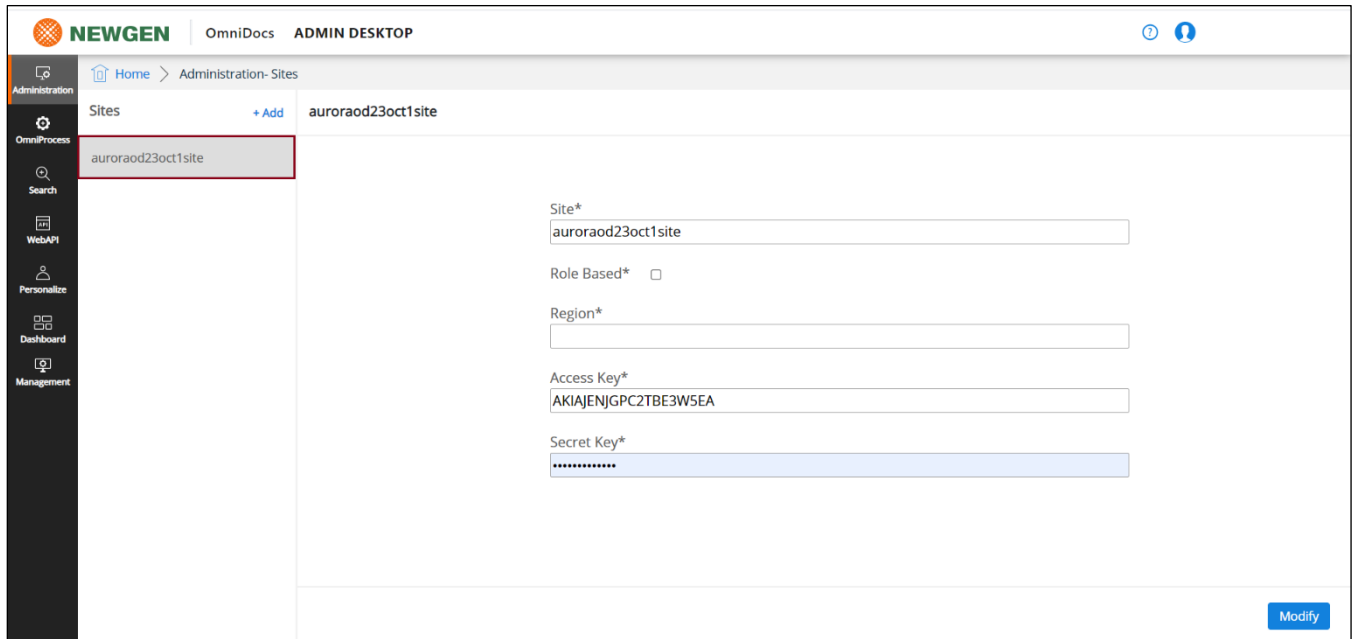


Figure 3.104

7. Go back to the **Home** page.

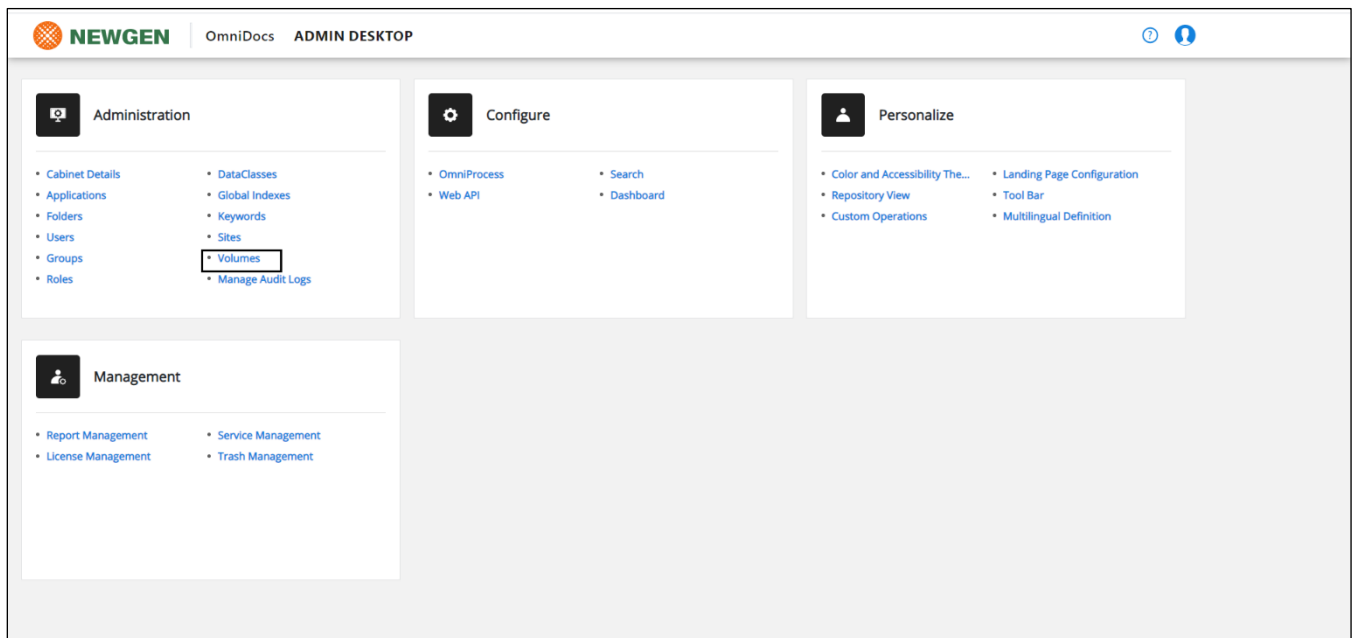


Figure 3.105

8. Select **Volumes**. The Volumes screen appears.

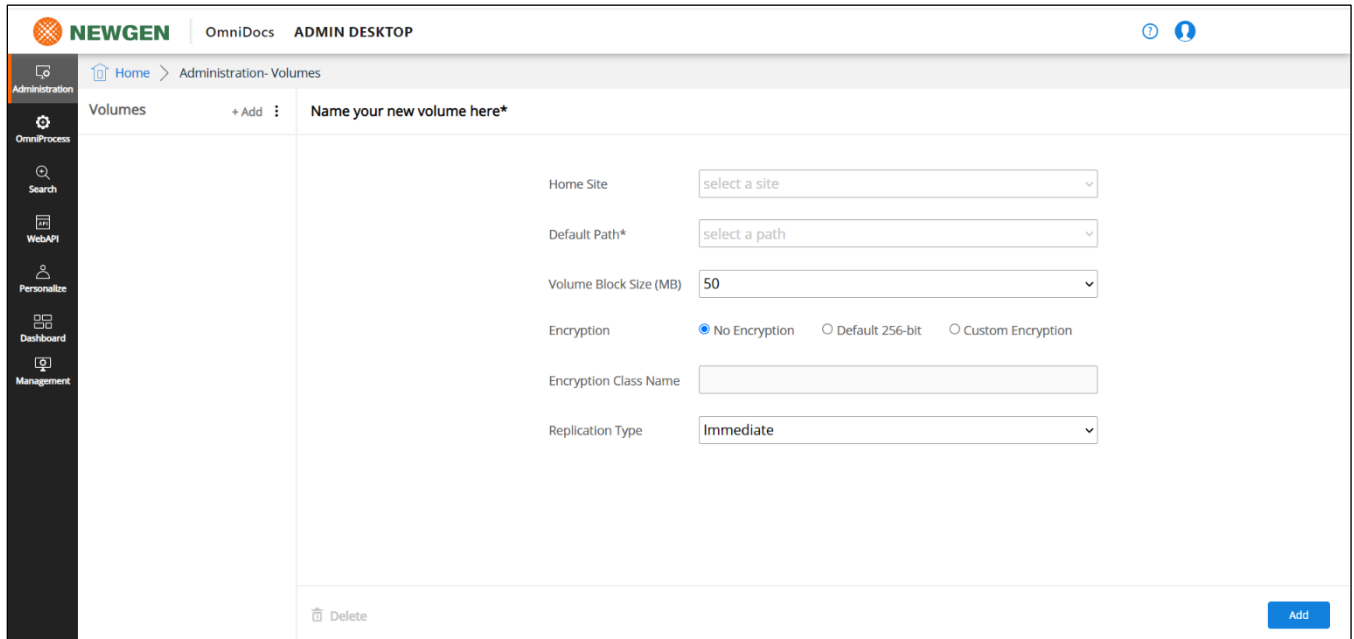


Figure 3.106

9. Specify the following details:

- **Home Site:** Select the newly created Site name.
- **Default Path:** Select the S3 bucket in which you want to store PN files.
- **Volume Name:** Specify the user-defined volume name.

10. Click **Add**.

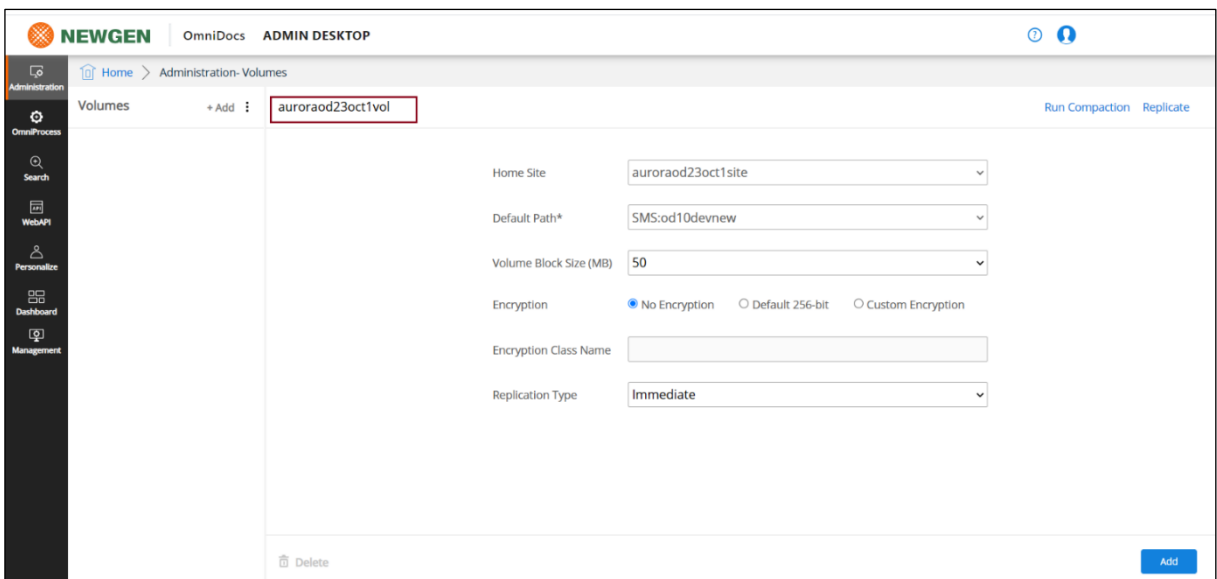


Figure 3.107

The added volume appears under **Image Volumes** in the left panel.

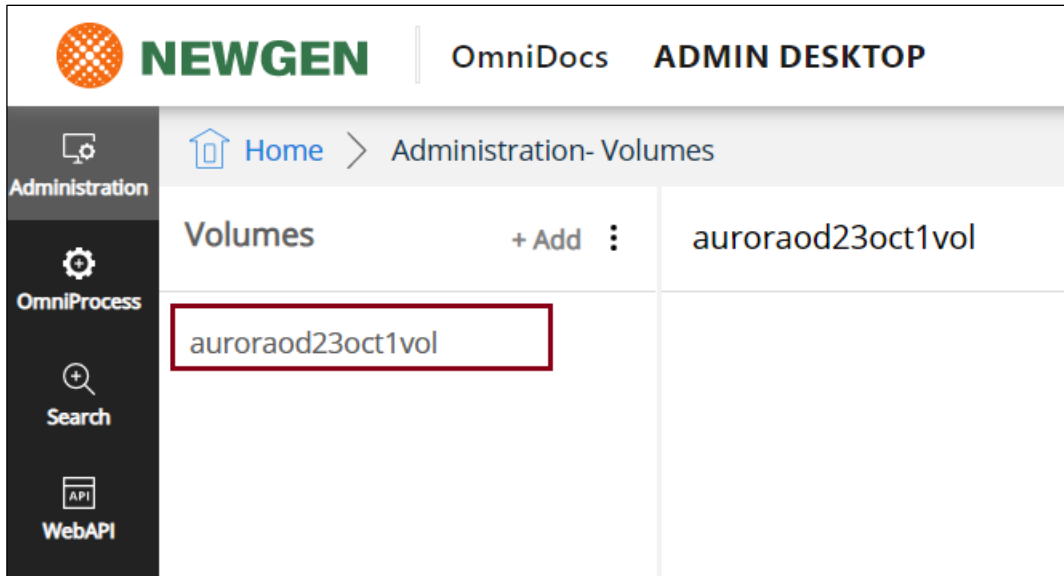


Figure 3.108

11. Go back to the **Home** screen.

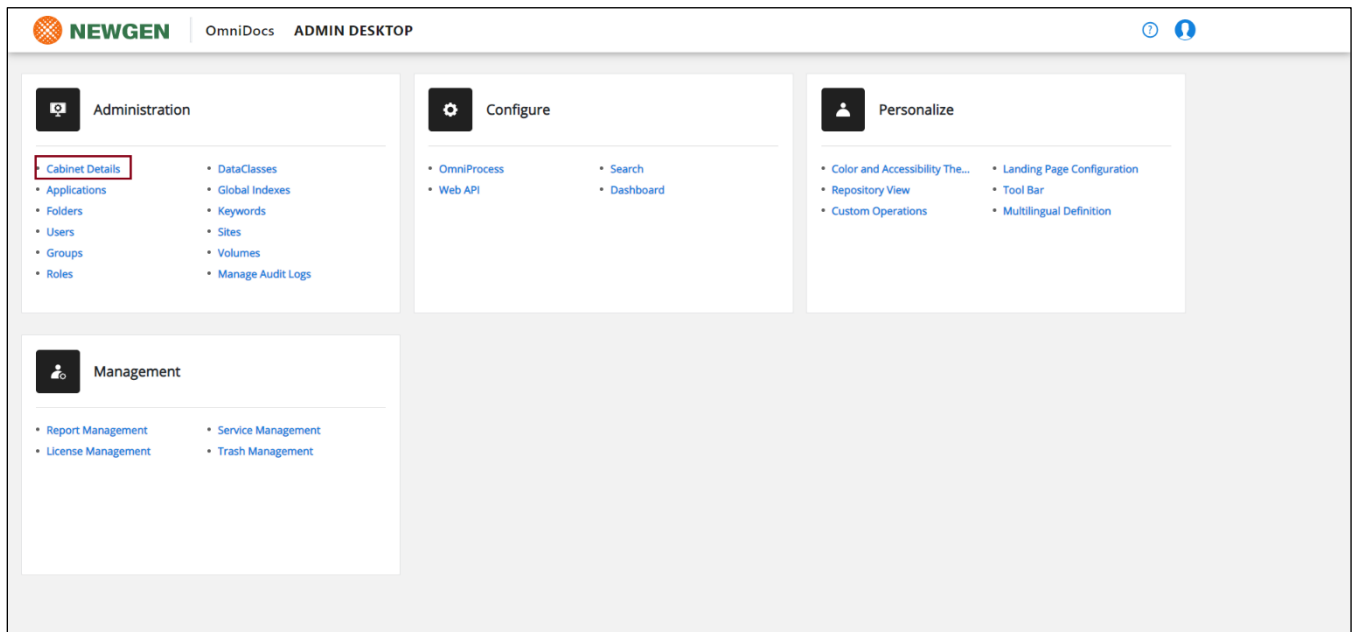


Figure 3.109

12. Click **Cabinet Details**.

13. Select the added volume from the **Default Image Volume** using the dropdown

14. Click **Save**. The Site and Volume are now created successfully.

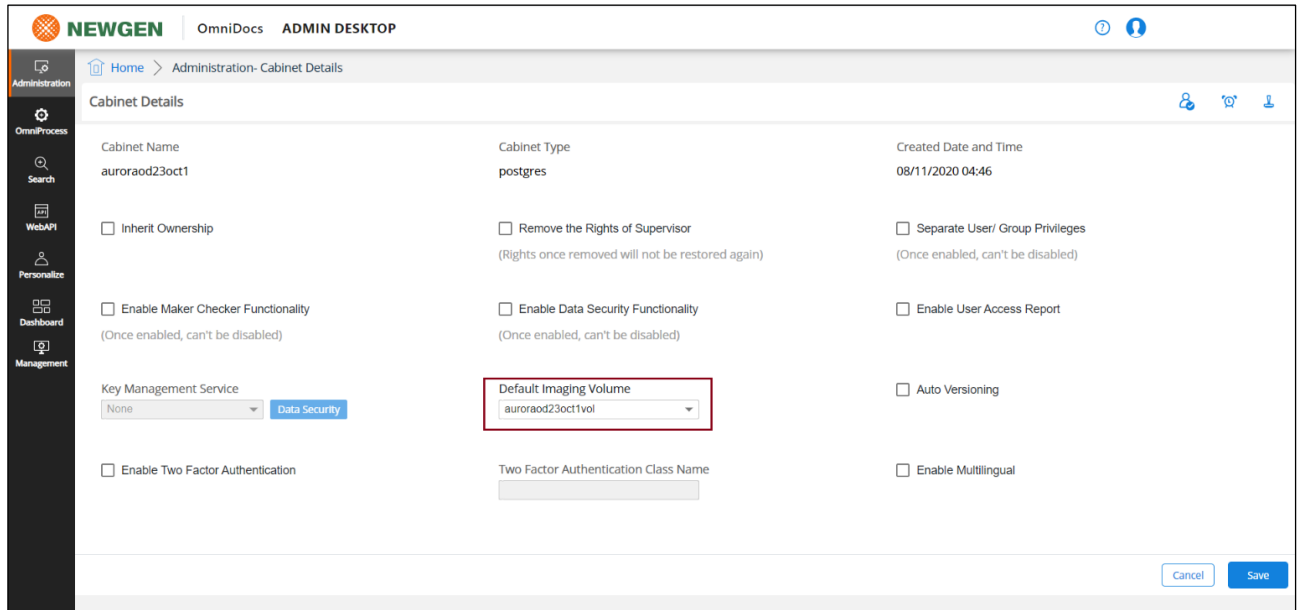


Figure 3.110

15. Log in to the OmniDocs Web using the below URL to start.

http://<Host-Path URL of OmniDocsWeb container>/omnidocs/web

For example: *http://ecmsuite.aws.co.in/omnidocs/web*

3.8 EasySearch Post-Deployment Changes

Perform the below steps to do EasySearch post-deployment changes:

1. Login to the ApacheManifold Admin using the following URL:

<Host-Path URL of ApacheManifold>/mcf-crawler-ui/login.jsp

For example,

http://ecmsuiteapache.aws.co.in/mcf-crawler-ui/login.jsp



Figure 3.111

2. Log in with the following credentials:
 - **User ID:** admin
 - **Password:** admin
3. After a successful login, click **Jobs** tree showing in the left panel.
4. Click **Status and Job Management**. The below job list appears:
 - <CABINET_NAME>_Document
 - <CABINET_NAME>_Folder
5. Start both the jobs.
6. Once both the jobs started, the Job's status appears as **Running**.

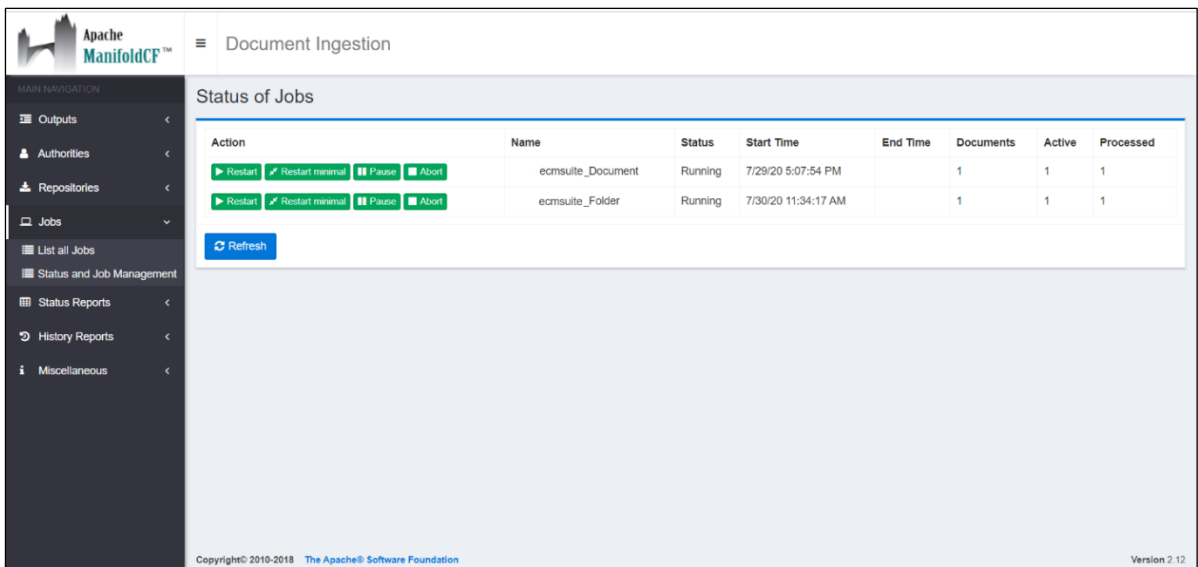


Figure 3.112

3.9 OmniScanWeb: Registering a cabinet

Perform the below steps to register the cabinet in OmniScanWeb:

1. Open the OmniScanWeb using the following URL:

http://<Host-Path URL of OmniScanWeb container>/omniscanweb

 For example,

<https://omniscan.newgendocker.com/omniscanweb>
2. Click **Register New Cabinet** link on the OmniScan Web login screen.
3. Specify the Server URL as given below:

http://<Host-Path URL of OmniDocsWeb container>/NGServlet/servlet/ExternalServlet

 For example,

<https://omnidocs.newgendocker.com/NGServlet/servlet/ExternalServlet>

- Specify the **OmniDocs EJB** container name for AppServer IP or Server URL, 8080 for AppServer Port, and JBOSSSEAP for AppServer Type.

← Login Register Cabinet

1 Connect 2 Register

Server URL

AppServer IP

AppServer Port

AppServer Type

Connect

newgen OmniScan
© Powered by Newgen Softwares

Figure 3.113

- Click **Connect**.
- Select the **Cabinet Name**, **Site ID**, and **Volume ID** from the list.

← Login Register Cabinet

1 Connect 2 Register

Cabinet Name

Site ID

Volume ID

Register

newgen OmniScan
© Powered by Newgen Softwares

Figure 3.114

7. Click **Register**. The registered cabinet appears in the **Cabinet Name** list on the login screen. Now you can log into OmniScan Web.

NOTE:

Ensure that the **OmniScan_Template_Repository** folder is already created in OmniDocs before logging into OmniScan Web.

3.10 Creating a Secret manager policy and secrets

To create IAM Policy and Role, perform the below steps:

1. Create an IAM Policy with the name **SecretMgr_Policy** with the following permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:DescribeSecret",
        "secretsmanager:PutSecretValue",
        "secretsmanager:CreateSecret",
        "secretsmanager>DeleteSecret",
        "secretsmanager:CancelRotateSecret",
        "secretsmanager:ListSecretVersionIds",
        "secretsmanager:UpdateSecret",
        "secretsmanager:GetRandomPassword",
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:StopReplicationToReplica",
        "secretsmanager:ReplicateSecretToRegions",
        "secretsmanager:RestoreSecret",
        "secretsmanager:RotateSecret",
        "secretsmanager:UpdateSecretVersionStage",
        "secretsmanager:RemoveRegionsFromReplication",
        "secretsmanager:ListSecrets"
      ],
      "Resource": "*"
    }
  ]
}
```

2. Add this policy to the **Worker node IAM Role**.
3. Create a Secret for **Alarm Mailer**

Refer to the below steps to create Secrets for Alarm Mailer:

- Open the AWS Secret Manager console.
- Click **Store a new secret**.
- Select **Other type of secret**.

- Add Key/ value mentioned below:
 - a. CabinetName_Username
 - b. CabinetName_Password

NOTE:

Update the **CabinetName** with your Cabinet Name.

Here values are:

CabinetName_Username : supervisor group’s username

CabinetName_Password : supervisor group’s password

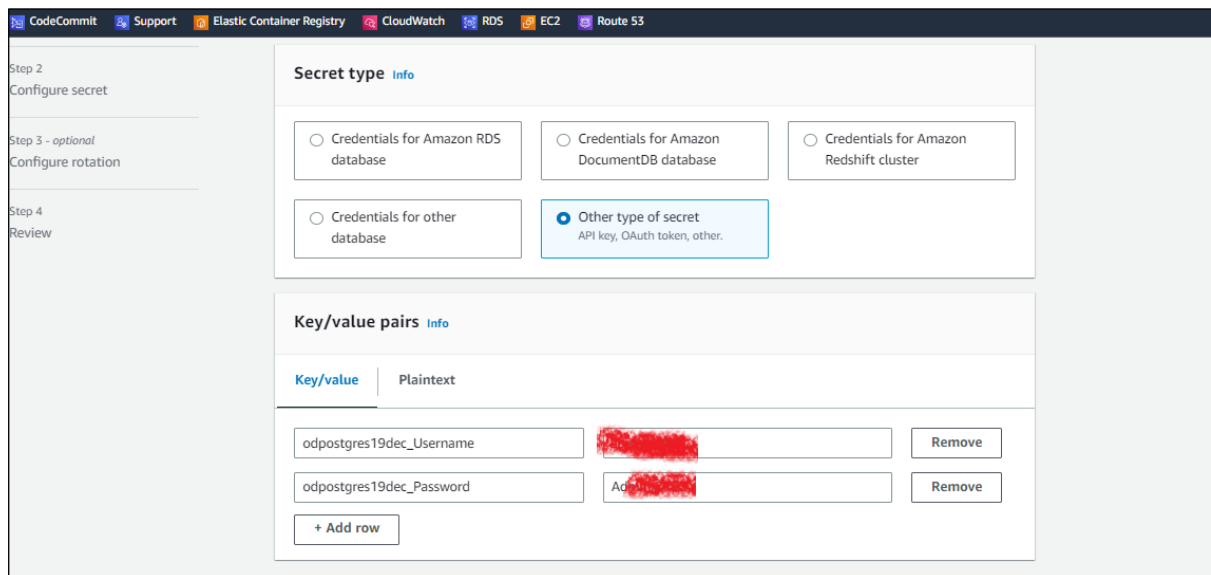


Figure 3.115

- Click **Next**.
- Enter Secret name is **AlarmMailerPSequence**.
- Click **Next** and **Store** to save this Secret.

4. Create Secret for LDAP:

Follow Below steps to create Secrets for LDAP

- Open the AWS Secret Manager console.
- Click on **Store a new secret**.
- Select **Other type of secret**.
- Add Key/ value mentioned below:
 - a. ODUsername
 - b. ODPassWord

- c. DomainUserName
- d. DomainPassword
- e. DistinguishedName

Here values are:

ODUsername: supervisor group's username

ODPassword: supervisor group's password

DomainUserName: Active Directory Domain username

DomainPassword: Active Directory Domain password

DistinguishedName: Active Directory Distinguished username

The screenshot shows the AWS IAM console interface for creating a secret. At the top, there are four radio button options for secret types: 'Credentials for Amazon RDS database', 'Credentials for Amazon DocumentDB database', 'Credentials for Amazon Redshift cluster', and 'Other type of secret'. The 'Other type of secret' option is selected and highlighted in blue. Below this, there is a section for 'Key/value pairs' with an 'Info' link. The 'Key/value' tab is active, showing a table with five rows of key-value pairs. Each row has a 'Key' column, a 'Value' column (with redacted content), and a 'Remove' button. At the bottom left of the table is a '+ Add row' button.

Key/value	Plaintext	
ODUsername	sc[REDACTED]	Remove
ODPassword	[REDACTED]	Remove
DomainUserName	a[REDACTED]	Remove
DomainPassword	sys[REDACTED]	Remove
DistinguishedName	adm[REDACTED]@gen[REDACTED].net	Remove

Figure 3.116

- Click **Next**.
- Enter Secret name is **LDAP**.
- Click **Next** and **Store** to save this Secret.

4 Configuring AWS CodePipeline for container deployment on EKS

deployment on EKS

This chapter describes the configuration of AWS CodePipeline for container deployment on Elastic Kubernetes Service (EKS).

4.1 Overview

The Build Pipeline and Release Pipeline are separated into two parts. Build Pipeline is done through the Jenkins server which can be installed on an on-premises machine or a cloud machine. Using the AWS CodePipeline cloud service, you can manage the Release pipeline. In this architecture, three stages are created that is, Dev, UAT, and Production and in each stage, deployment is quite different. You can have some more stages depending on the requirements. This document describes the configuration of the AWS CodePipeline for container deployment on EKS.

4.2 Architecture of CI/CD pipeline

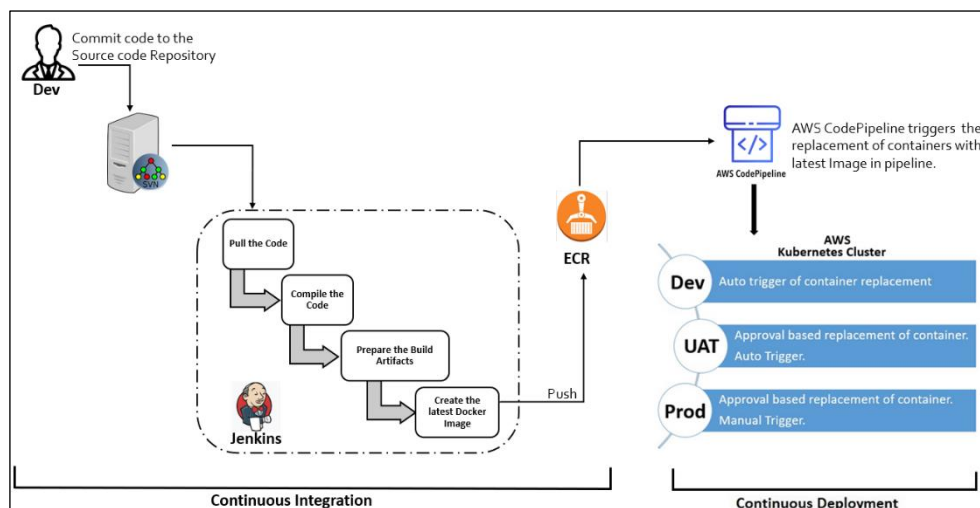


Figure 4.1

- The Newgen representative builds the product’s base Docker images on the company’s on-premises servers using Jenkins.
- As soon as the Dev team commits the code to the source code repository, the Jenkins pipeline gets triggered. It pulls the code > compiles them > prepares the build artifacts > creates Docker images and pushes the newly created Docker images to the AWS Elastic Container Registry.

- As soon as any Docker image is pushed to the AWS Elastic Container Registry, AWS CodePipeline triggers the deployment to the Dev environment. Here, you can configure the performance testing as well as security testing of the application. In Addition, you can perform manual testing as required.
- UAT and Production deployments are based on approval and are available on-demand. To deploy to the UAT environment, you need to trigger the UAT deployment. Upon deployment trigger, an approval mail is sent to the project manager or the concerned team. As soon as the project manager approves the go-ahead, UAT deployment gets started automatically.
- Production deployment is also based on approval, but it is multi-level approval. To deploy a production environment, you require the approval of all stakeholders, and the production environment doesn't get triggered automatically on receiving all the approvals. A manual intervention mail is sent to the engineer who is supposed to deploy to production with a checklist. During deployment, all the checklist points get verified before performing the production deployment. In case any point of the checklist is not covered, then deployment to the production gets rejected.

4.3 Configuring AWS Elastic container registry

This section explains how to configure the AWS Elastic Container Registry.

Perform the below steps to configure AWS Elastic Container Registry:

1. Open the Amazon ECR console at <https://console.aws.amazon.com/ecr/repositories>.
2. From the navigation bar, select the Region to create your repository.
3. In the navigation pane, select **Repositories**.
4. On the Repositories page, select **Create repository**.
5. Select **Private** in the Visibility settings.
6. Enter a unique name for a repository that is, **omnidocs10.1 web** in the **Repository Name**.
7. For **Tag immutability**, do not enable the tag mutability setting for the repository. Repositories are configured with immutable tags that prevent image tags from getting overwritten.
8. Enable the image scanning setting for the repository for Scan on push. Repositories that are configured to scan on push start an image scan whenever an image is pushed.
9. Keep the **Encryption settings** as default.
10. Select **Create repository**.

ECR > Repositories > Create repository

Create repository

General settings

Visibility settings [Info](#)
Choose the visibility setting for the repository.

Private
Access is managed by IAM and repository policy permissions.

Public
Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.

678035612169.dkr.ecr.ap-south-1.amazonaws.com/

15 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, and forward slashes.

Tag immutability [Info](#)
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

Disabled

ⓘ Once a repository is created, the visibility setting of the repository can't be changed.

Image scan settings

Scan on push
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.

Enabled

Encryption settings

KMS encryption
You can use *AWS Key Management Service (KMS)* to encrypt images stored in this repository, instead of using the default encryption settings.

Disabled

ⓘ The KMS encryption settings cannot be changed or disabled after the repository is created.

[Cancel](#) [Create repository](#)

Figure 4.2

NOTE:

AWS ECR repositories can also be created while pushing Docker images to the AWS ECR using the AWS CLI.

4.4 Push and Pull docker images to or from AWS ECR

This section describes how to push and pull docker images from AWS ECR.

Prerequisites: Ensure that you have installed the latest version of the **AWS CLI** and **Docker**.

Following are the steps to push, and pull Docker images to or from AWS ECR:

- [Authentication](#)
- [Push](#)
- [Pull](#)

Authentication:

1. Before you push or pull the Docker images, you need to authenticate the Docker client to the AWS ECR.
2. Execute the below command to configure the AWS AccessKey and AWS SecretKey of the IAM user that has the rights to push or pull Docker images to AWS ECR:

```
aws configure set aws_access_key_id <AWS_AccessKey>
aws configure set aws_secret_access_key <AWS_SecretKey>
```

3. Execute the below command to retrieve the authentication token and authenticate the Docker client:

```
aws ecr get-login-password --region <AWS_Region> | docker login --username AWS
--password-stdin <AWS_AccountID>.dkr.ecr.<AWS_Region>.amazonaws.com
```

Push:

1. Before pushing the docker images to AWS ECR, you must create a repository to store them in.
2. Execute the below command to create a new repository if already not created:

```
aws ecr describe-repositories --repository-names <RepositoryName> || aws ecr
create-repository --repository-name <RepositoryName> --image-scanning-
configuration scanOnPush=true
```

3. Execute the below command to push the Docker images from your local machine to AWS ECR:

```
Docker tag <ImageName>:<ImageTag>
<AWS_AccountID>.dkr.ecr.<AWS_Region>.amazonaws.com/<ImageName>:<ImageTag>
docker push
<AWS_AccountID>.dkr.ecr.<AWS_Region>.amazonaws.com/<ImageName>:<ImageTag>
```

NOTE:

- Docker images might be shared in the form of a compressed tar file. As compressed Docker images cannot be used directly, first you need to decompress them in a Docker image form, and then you can use it. In such a case, the client needs to perform the following:
 - Download the compressed Docker image file.
 - Convert the compressed file into a Docker image using the Docker Load command.
Example: `docker load -i C:\DockerImages\omnidocs110web.tar`
 - Re-tag the images with your own registry and push them up.
 - To push any local Docker images to a repository, it is mandatory to first tag that image. We can also configure these commands in Jenkins to execute them automatically.
-

4. Use the below `batch` scripts to configure the 'Push Docker images to AWS ECR' in Jenkins:

```
@echo off
set AWS_AccessKey= AKIAJENJGXXXXXXXXXXXXXXXXXXXXX
set AWS_SecretKey= G4tbmZNlv64EO5475G4XXXXXXXXXXXXXXXXX
set AWS_AccountID=678035612169
set AWS_Region=ap-south-1
set ImageName=omnidocs11.0web
set ImageTag=sp1
set BuildNumber=%ImageTag%-build-%BUILD_NUMBER%

aws configure set aws_access_key_id %AWS_AccessKey%
aws configure set aws_secret_access_key %AWS_SecretKey%

aws ecr get-login-password --region %AWS_Region% | docker login --username AWS
--password-stdin %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com

aws ecr describe-repositories --repository-names %ImageName% || aws ecr
create-repository --repository-name %ImageName% --image-scanning-configuration
scanOnPush=true

docker tag %ImageName%:%ImageTag%
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%ImageTag%
docker push
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%ImageTag%

docker tag %ImageName%:%ImageTag%
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%BuildNumber%
docker push
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%BuildNumber%
```

```

@echo off
set AWS_AccessKey= AKIAJENJGXXXXXXXXXXXXXXXXXXXXX
set AWS_SecretKey= G4tbmZn1v64EO5475G4XXXXXXXXXXXXXXXXXXXXX
set AWS_AccountID=678035612169
set AWS_Region=ap-south-1
set ImageName=omnidocs11.0web
set ImageTag=spl
set BuildNumber=%ImageTag%-build-%BUILD_NUMBER%
aws configure set aws_access_key_id %AWS_AccessKey%
aws configure set aws_secret_access_key %AWS_SecretKey%
aws ecr get-login-password --region %AWS_Region% | docker login --username AWS--password-stdin %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com
aws ecr describe-repositories --repository-names %ImageName% || aws ecr create-repository --repository-name %ImageName% --image-scanning-configuration scanOnPush=true
docker tag %ImageName%:%ImageTag% %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%ImageTag%
docker push %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%ImageTag%
docker tag %ImageName%:%ImageTag% %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%BuildNumber%
docker push %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%BuildNumber%

```

Figure 4.3

Pull:

1. Execute the below command to pull the Docker images from AWS ECR:

```

docker pull
<AWS_AccountID>.dkr.ecr.<AWS_Region>.amazonaws.com/<ImageName>:<ImageTag>

```

2. Use the below **batch** scripts to configure the **Pull Docker images from AWS ECR** in Jenkins:

```

@echo off
set AWS_AccessKey=AKIAJENJGXXXXXXXXXXXXXXXXXXXXX
set AWS_SecretKey=G4tbmZn1v64EO5475G4XXXXXXXXXXXXXXXXXXXXX
set AWS_AccountID=678035612169
set AWS_Region=ap-south-1
set ImageName=omnidocs11.0web
set ImageTag=spl

aws configure set aws_access_key_id %AWS_AccessKey%
aws configure set aws_secret_access_key %AWS_SecretKey%
aws ecr get-login-password --region %AWS_Region% | docker login --username AWS
--password-stdin %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com

docker pull
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%ImageTag%

```

```

@echo off
set AWS_AccessKey=AKIAJENJGXXXXXXXXXXXXXXXXXXXXX
set AWS_SecretKey=G4tbmZn1v64EO5475G4XXXXXXXXXXXXXXXXXXXXX
set AWS_AccountID=678035612169
set AWS_Region=ap-south-1
set ImageName=omnidocs11.0web
set ImageTag=spl
aws configure set aws_access_key_id %AWS_AccessKey%
aws configure set aws_secret_access_key %AWS_SecretKey%
aws ecr get-login-password --region %AWS_Region% | docker login --username AWS
--password-stdin %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com
docker pull
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%ImageTag%

```

Figure 4.4

4.5 Configuring AWS CodePipeline

To configure the AWS CodePipeline, follow the below subsections:

- [Creation of IAM Policy and IAM Role](#)
- [Creation of AWS CodeCommit Repository](#)
- [Creation of AWS CodeBuild Project](#)
- [Creation of AWS CodePipeline](#)

4.5.1 Creating an IAM Policy and IAM Role

Perform the below steps to create IAM Policy and Role:

1. Create an IAM policy with the name **EKS-cluster-access** with the following permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "eks:DescribeCluster",
      "Resource": "*"
    }
  ]
}
```

2. Create another IAM policy with the name **code-build-service-policy** with the following permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:logs:ap-south-1:678035612169:log-group:*"
      ],
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ]
    },
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::codepipeline-ap-south-1-*"
      ],
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ]
    }
  ]
}
```

```

    ],
    },
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:codecommit:ap-south-1:678035612169:*"
      ],
      "Action": [
        "codecommit:GitPull"
      ]
    }
  ]
}

```

NOTE:

The policy **code-build-service-policy** is created for the AWS Region **ap-south-1** only. If you want to create this IAM role for other regions, then update the region in the JSON policy file. Use your AWS account ID as the place of **678035612169** in the above JSON policy file.

3. Create an IAM role with the name **genesis-codebuild-eks** and attach the policy **EKS-cluster-access** and **code-build-service-policy** created in the previous step. It must be applied for the **codebuild** service. This is required for the CodeBuild role to authenticate with the EKS cluster.

NOTE:

CodeBuild role has permission to authenticate the cluster but doesn't have the requisite RBAC access to do any other action on the cluster. Due to the reason that when an Amazon EKS cluster is created, the IAM entity user or role that creates the cluster is automatically granted system masters permissions in the cluster's RBAC configuration. To grant additional AWS users or roles the ability to interact with your cluster, you must edit the **aws-auth ConfigMap** within Kubernetes.

4. Execute the below command to open the **aws-auth ConfigMap** in edit mode:

```
kubectl edit configmap aws-auth -n kube-system
```

5. Add the following under **data.mapRoles**:

```

- rolearn: <ARN of the created IAM role for CodeBuild>
  username: <Name of the created IAM role for CodeBuild>
  groups:
    - system:masters

```

For example,

```

- rolearn: arn:aws:iam::678035612169:role/genesis-codebuild-eks
  username: genesis-codebuild-eks
  groups:
    - system:masters

```

6. The final **aws-auth ConfigMap** must look somewhat like this:

```

apiVersion: v1
kind: ConfigMap
metadata:

```

```

name: aws-auth
namespace: kube-system
data:
  mapRoles: |
    - rolearn: arn:aws:iam::678035612169:role/EKScluster-NodeInstanceRole-1FRJ044RCC90Q
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::678035612169:role/genesis-codebuild-eks
      username: genesis-codebuild-eks
      groups:
        - system:masters

```

```

ec2-user@ip-10-0-1-108:~
Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::678035612169:role/EKScluster-NodeInstanceRole-1FRJ044RCC90Q
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::678035612169:role/genesis-codebuild-eks
      username: genesis-codebuild-eks
      groups:
        - system:masters
kind: ConfigMap
metadata:
  annotations:
    kubectrl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"mapRoles":"- rolearn: arn:aws:iam::678035612169:role/EKScluster-NodeInstanceRole-1FRJ044RCC90Q\n username: system:node:{{EC2PrivateDNSName}}\n groups:\n - system:bootstrappers\n - system:nodes\n"},"kind":"ConfigMap","metadata":{"annotations":{},"name":"aws-auth","namespace":"kube-system"}}
    creationTimestamp: "2020-08-31T13:14:45Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "82131456"

```

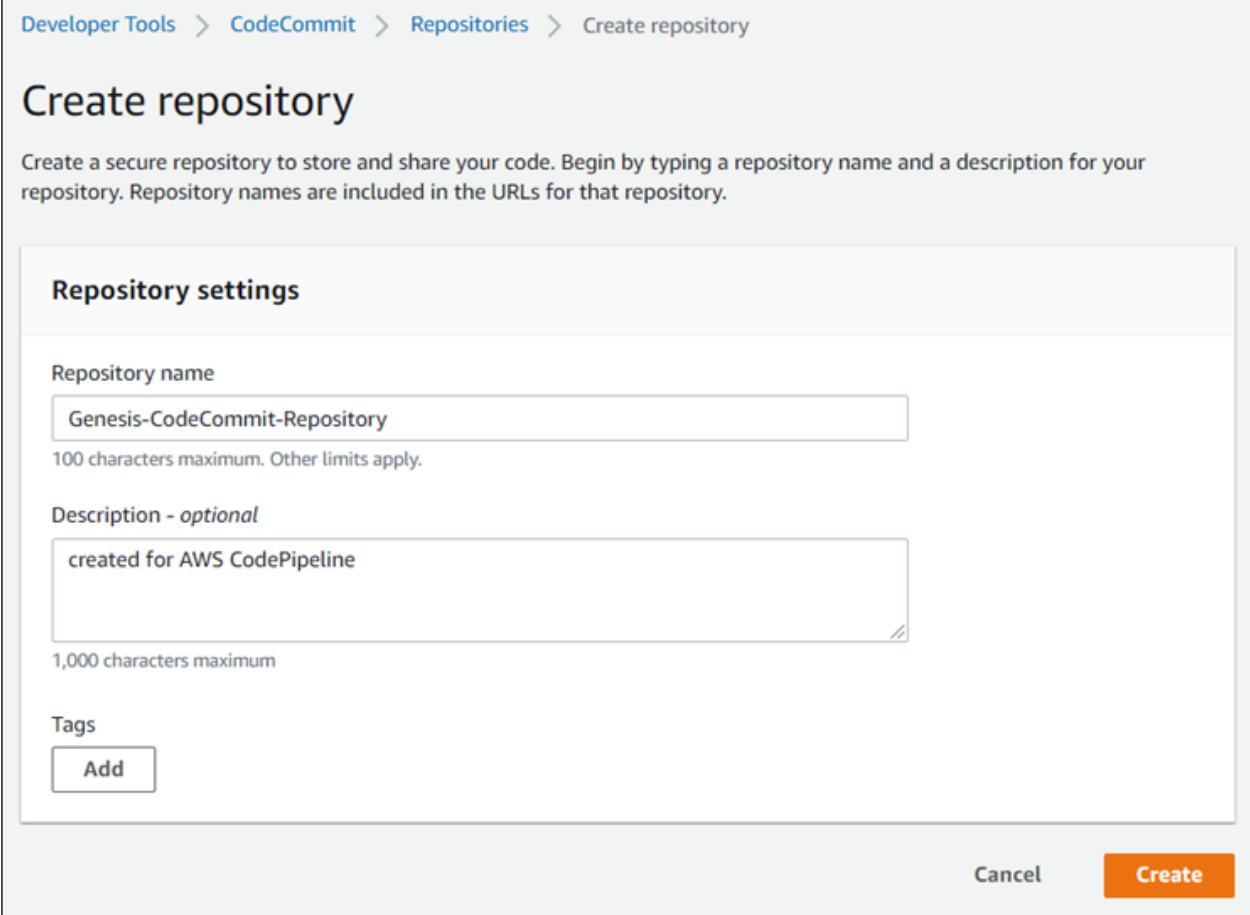
Figure 4.5

The CodeBuild role has the requisite RBAC access.

4.5.2 Creating AWS CodeCommit repository

Perform the below steps to create AWS CodeCommit Repository:

1. Open the AWS CodeCommit console at:
<http://console.aws.amazon.com/codesuite/codecommit/home>
2. Select **Create repository**.
3. For the **Repository name**, enter a unique name for your repository, that is, **Genesis-CodeCommit-Repository**.
4. Select **Create**.



Developer Tools > CodeCommit > Repositories > Create repository

Create repository

Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository.

Repository settings

Repository name
Genesis-CodeCommit-Repository
100 characters maximum. Other limits apply.

Description - optional
created for AWS CodePipeline
1,000 characters maximum

Tags
Add

Cancel Create

Figure 4.6

5. Upload all the YAML files shared with the Release Package to the created AWS CodeCommit repository:
 - AWS_ALB-IngressController.yml
 - buildspec.yml
 - buildspec_EasySearch.yml
 - OmniDocs11.0Web.yml
 - OmniDocs11.0Web_Services.yml

- OmniDocs11.0EJB.yml
- OmniDocs11.0Services.yml
- EasySearch11.0.yml
- EasySearch11.0_ApacheOnly.yml
- TEM11.0.yml
- OmniScanWeb6.0.yml
- RMSSharePointAdapter.yml
- OmniDocs11.0MessagingService.yml

NOTE:

The YAML file is a human-readable object configuration file that is used to deploy and manage the objects on the Kubernetes cluster. In other words, it is a manifest file that contains the deployment descriptor of Docker images.

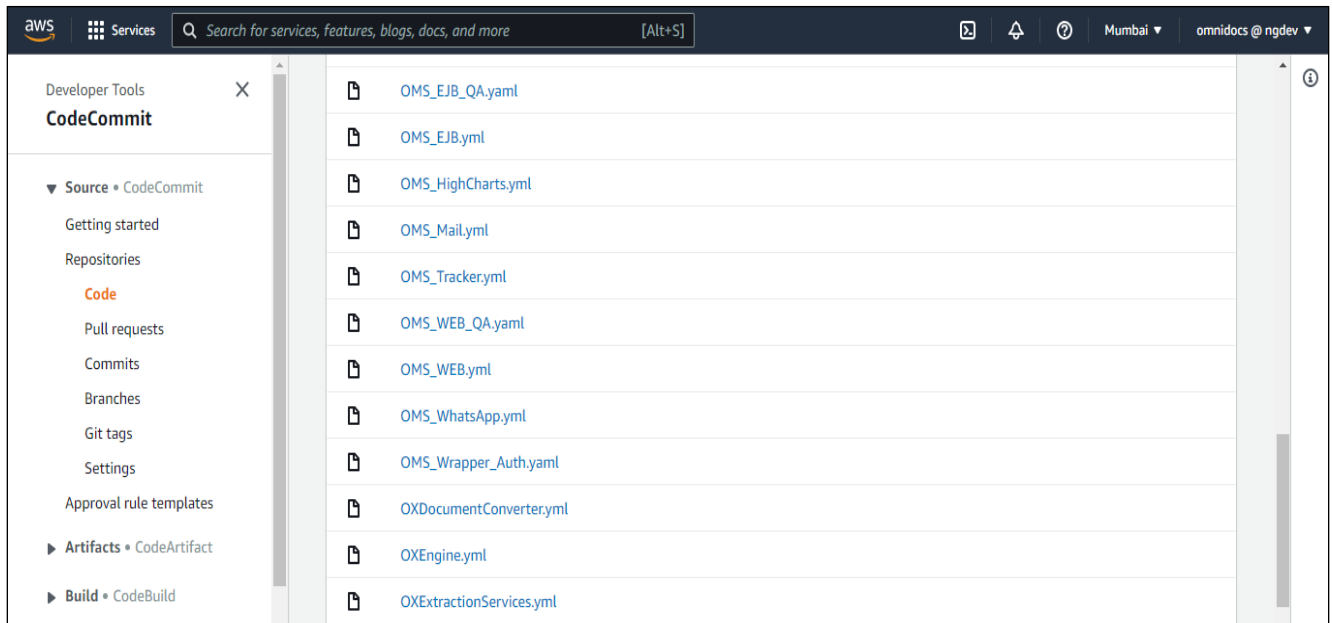


Figure 4.7

4.5.3 Creating AWS CodeBuild project

This section explains how to create AWS Code Build Project.

NOTE:

Use the upcoming steps as a reference to configure the Release Pipeline for the below Docker Images:

- OmniDocs 11.0 Web
 - OmniDocs 11.0 Web_Services
 - OmniDocs 11.0 EJB
 - OmniDocs 11.0Services
 - EasySearch11.0
 - TEM11.0
 - OmniScanWeb6.0
 - RMS SharePoint Adapter
 - Messaging Service
-

Perform the below steps to create AWS Code Build Project:

1. Open the AWS CodeBuild console at:

<http://console.aws.amazon.com/codesuite/codebuild/home>

2. Select **Create build project**.

Specify the following. Once done, select **Create build project**.

3. Specify the following in the Project configuration.

- i. Enter a unique name for your CodeBuild project that is, **OmniDocs101Web** in the **Project Name**.
- ii. (Optional) Enter a description of the build project to help other users understand the project.
- iii. (Optional) Select **Enable Build badge**. Build badge provides an embeddable, dynamically generated image (badge) that displays the status of the latest build for a project.
- iv. Restrict the **concurrent build limit** to start the project to **1**.
- v. Keep the other settings as default.


Project configuration

Project name

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.

Description - *optional*

Created for EKS Deployment



Build badge - *optional*

Enable build badge

Enable concurrent build limit - *optional*
Limit the number of allowed concurrent builds for this project.

Restrict number of concurrent builds this project can start

Concurrent build limit

The concurrent build limit must be greater than 0 and less than the account build limit.

▶ **Additional configuration**
tags

Figure 4.8

4. Specify the following in the **Source**.
 - i. Select the AWS CodeCommit in the **Source provider**.
 - ii. Select the existing AWS CodeCommit repository **Genesis-CodeCommit-Repository** created in the **Creation of AWS CodeCommit Repository**.
 - iii. Select **Branch** as main.
 - iv. Keep the other settings as default.

Source
Add source

Source 1 - Primary

Source provider

AWS CodeCommit
▼

Repository

Q Genesis-CodeCommit-Repository
✕

Reference type

Choose the source version reference type that contains your source code.

Branch

Git tag

Commit ID

Branch

Choose a branch that contains the code to build.

main
▼

Commit ID - *optional*

Choose a commit ID. This can shorten the duration of your build.

Q

Source version [Info](#)

refs/heads/main

7fb6f314 ok

▶ **Additional configuration**

Git clone depth, Git submodules

Figure 4.9

5. Specify the following in the **Environment**:

- i. Select the **Managed image** in the **Environment image**.
- ii. Select **Amazon Linux 2** in the **Operating system**.
- iii. Select **Standard** in the **Runtime(s)**.
- iv. For **Image**, select 'aws/codebuild/amazonlinux2-x86_64-standard:3.0'.
- v. Select **Always use the latest image for this runtime image for the image** version.
- vi. Select the Existing service role **genesis-codebuild-eks** created in the **Creation of IAM Policy and IAM Role** in the **Service role**.

NOTE:

Don't select the checkbox **Allow AWS CodeBuild to modify this service role so it can be used with this build project**. Also, keep the other settings as default.

Environment

Environment image

Managed image
Use an image managed by AWS CodeBuild

Custom image
Specify a Docker image

Operating system

Amazon Linux 2

i The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See [Docker Images Provided by CodeBuild for details](#).

Runtime(s)

Standard

Image

aws/codebuild/amazonlinux2-x86_64-standard:3.0

Image version

Always use the latest image for this runtime version

Privileged

Enable this flag if you want to build Docker images or want your builds to get elevated privileges

Service role

New service role
Create a service role in your account

Existing service role
Choose an existing service role from your account

Role ARN

Allow AWS CodeBuild to modify this service role so it can be used with this build project

▶ **Additional configuration**
Timeout, certificate, VPC, compute type, environment variables, file systems

Figure 4.10

6. Specify the following in the **Buildspec**.
 - i. Select the **Use a buildspec file** in the **Build specification**.
 - ii. Specify *buildspec.yml* in the **Buildspec name - optional**.

NOTE:

By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

Buildspec

Build specifications

Use a buildspec file
Store build commands in a YAML-formatted buildspec file

Insert build commands
Store build commands as build project configuration

Buildspec name - *optional*
By default, CodeBuild looks for a file named buildspec.yaml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yaml or configuration/buildspec.yaml).

buildspec.yaml

Figure 4.11

7. **Batch configuration:** Leave with default settings.
8. **Artifacts:** Leave with default settings.
9. Specify the following in the **Logs:**
 - i. Select the CloudWatch logs – optional.
 - ii. Specify the Group name the same as the CodeBuild project name that is., **OmniDocs101Web**.
 - iii. For the **Stream name**, specify the codebuild.

Logs

CloudWatch

CloudWatch logs - *optional*
Checking this option will upload build output logs to CloudWatch.

Group name
OmniDocs101Web

Stream name
codebuild

S3

S3 logs - *optional*
Checking this option will upload build output logs to S3.

Cancel **Create build project**

Figure 4.12

10. Select **Create build project**.

NOTE:

The same CodeBuild project can be used for all types of stages like Dev, UAT, and Production or any other stage as per the business requirement.

4.5.4 Creating AWS CodePipeline

The three stages are configured that is, Dev, UAT, and Production, and at each stage, the deployment is quite different. You can have some more stages depending on the requirements.

NOTE:

Use the following steps as a reference to configure the Release Pipeline for the below Docker Images.

- OmniDocs11.0Web
 - OmniDocs11.0Web_Services
 - OmniDocs11.0EJB
 - OmniDocs11.0Services
 - EasySearch11.0
 - TEM11.0
 - OmniScanWeb6.0
 - RMS SharePoint Adapter
 - Messaging Service
-

Dev Stage: As soon as any Docker Image is pushed to the AWS Elastic Container Registry, AWS CodePipeline triggers the deployment to the Dev environment.

UAT Stage: UAT and Production deployments are approval based and they are called on-demand. To deploy to the UAT environment, triggers the UAT deployment. Once deployment is triggered, an approval mail to the concerned team. Upon receiving approval, UAT deployment gets started automatically.

Production Stage: Production Deployment is also based on approval based but it is multi-level approval. To deploy to a production environment, you require the approval of multiple stakeholders but deployment for the production environment is not get triggered automatically. A manual intervention mail is sent to the engineer who is supposed to deploy to production with a checklist. During the process, if the checklist points are not covered then the deployment to production gets rejected.

4.5.4.1 Configuring AWS CodePipeline for Dev stage

Perform the below steps to configure the Dev Stage:

1. Open the AWS CodePipeline console at:
<http://console.aws.amazon.com/codesuite/codepipeline/home>
2. Select the **Create pipeline**.

Specify the required details in the following steps. Once complete, select **Create pipeline** at the Review step:

3. **Select pipeline settings:**

- i. Enter a unique name for your pipeline, that is, **OmniDocs101Web-DevStage** for the Pipeline name.
- ii. Select **New service role** for Service role.
- iii. Select the checkbox **Allow AWS CodePipeline to create a service role so it can be used with this new pipeline.**
- iv. Keep the other settings as default and click **Next**.

The screenshot shows the 'Choose pipeline settings' dialog box. The 'Pipeline settings' section includes a text input for 'Pipeline name' containing 'OmniDocs101Web-DevStage'. Below it, the 'Service role' section has two radio buttons: 'New service role' (selected) and 'Existing service role'. The 'Role name' section has a text input containing 'AWSCodePipelineServiceRole-ap-south-1-OmniDocs101Web-DevStage'. A checkbox labeled 'Allow AWS CodePipeline to create a service role so it can be used with this new pipeline' is checked. At the bottom, there is an 'Advanced settings' section with a right-pointing arrow, and 'Cancel' and 'Next' buttons.

Figure 4.13

4. Specify the following in the Add source stage:

- i. Select the **AWS CodeCommit** for the Source provider.
- ii. Select the existing **AWS CodeCommit repository** 'Genesis-CodeCommit-Repository' created in the Creation of AWS CodeCommit Repository for the **Repository name**.
- iii. Select **Main** for the **Branch name**.
- iv. Select the recommended option **Amazon CloudWatch Events** for **Change detection options**.

NOTE:

Amazon CloudWatch Events creates a CloudWatch event rule. As soon as the changes are done in the integrated AWS CodeCommit repository, it triggers the pipeline. But do not trigger the AWS CodePipeline whenever there is a change in

the CodeCommit repository. The pipeline must be triggered whenever you push a new image to the container registry like AWS ECR.

Refer to the following sections for the configuration of AWS ECR with CodePipeline to disable the CloudWatch event rule once the pipeline is created.

- v. Keep the other settings as default and click **Next**.

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit

Repository name
Choose a repository that you have already created where you have pushed your source code.

Genesis-CodeCommit-Repository

Branch name
Choose a branch of the repository

main

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

Amazon CloudWatch Events (recommended)
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

AWS CodePipeline
Use AWS CodePipeline to check periodically for changes

Output artifact format
Choose the output artifact format.

CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include git metadata about the repository.

Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full git clone. Only supported for AWS CodeBuild actions.

Cancel Previous Next

Figure 4.14

5. Specify the following in the Add build stage:
 - i. Select the **AWS CodeBuild** in the Build provider.
 - ii. Select a **region** in which you want to create your pipeline.
 - iii. Select the existing CodeBuild project **OmniDocs101Web** created in the **Creation of AWS CodeBuild Project**.
 - iv. Create the below environment variables for Environment variables (optional):

Name	Value	Type
AWS_DEFAULT_REGION	ap-south-1	Plaintext
AWS_CLUSTER_NAME	OmniDocs-uat2	Plaintext

YAML_FILE	OmniDocs11.0Web.yml	Plaintext
CODE_PIPELINE_EXECUTION_ID	{codepipeline.PipelineExecutionId}	Plaintext

- **AWS_DEFAULT_REGION:** Specify the region where the AWS EKS cluster is created.
 - **AWS_CLUSTER_NAME:** Specify the name of the EKS cluster for the Dev stage.
 - **YAML_FILE:** Specify the name of the YAML file that is stored in the AWS CodeCommit repository and that is used to deploy the **OmniDocs10.1Web** container for the Dev stage.
 - **CODE_PIPELINE_EXECUTION_ID:** This variable is just created for logging purposes so that you can track the build-id and its initiated pipeline.
- v. For **Build type**, select **Single build** and Click **Next**.

Figure 4.15

6. In the **Add deploy stage:** Skip the deploy stage.

Figure 4.16

7. In the **Review**, select **Create pipeline**.

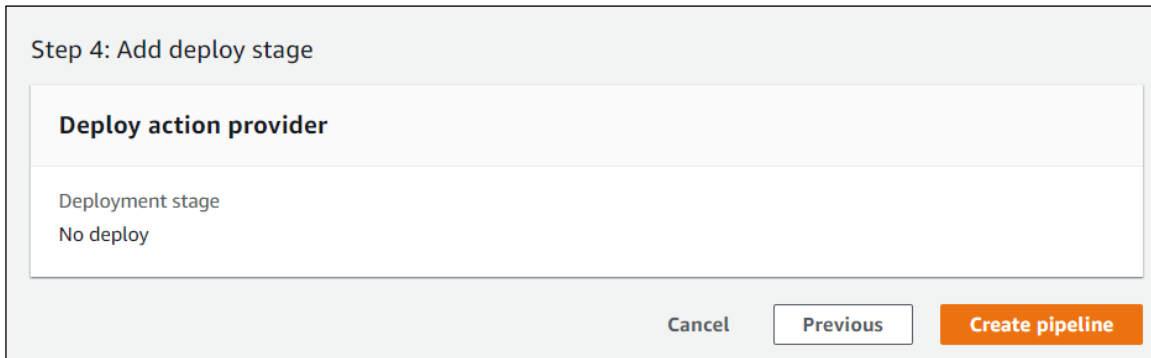


Figure 4.17

NOTE:

As soon as you create the pipeline, it starts the first pipeline execution. This execution failed as expected if you have not yet integrated the AWS ECR into the pipeline. You need to do the same.

Perform the below steps to integrate AWS ECR into the AWS CodePipeline:

1. Open the created pipeline **OmniDocs101Web-DevStage** in **Edit** mode.
2. Select the **Edit source stage**.
3. Select **+ Add action**.

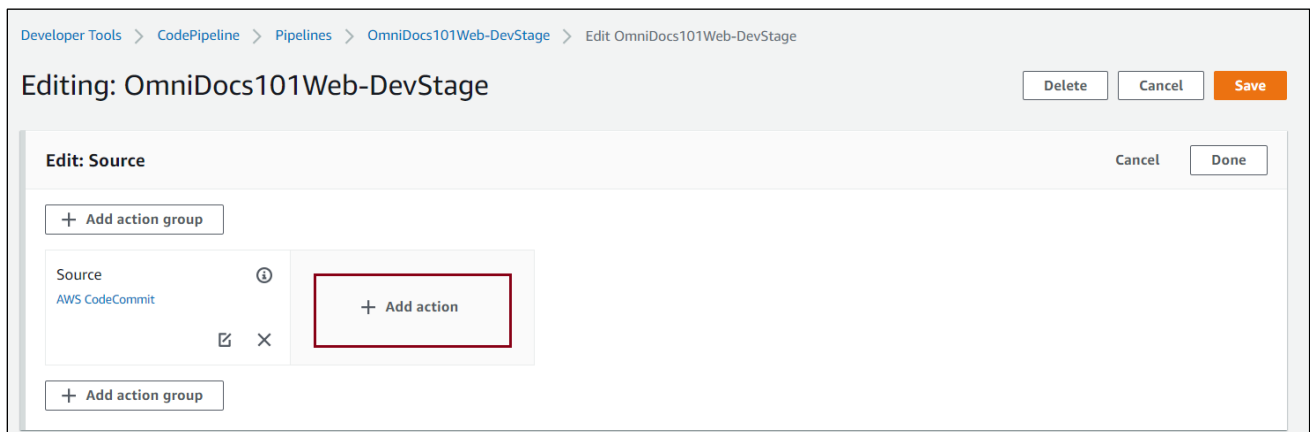


Figure 4.18

4. In the **Edit action** panel, specify the unique **Action name**. that is., AWS-ECR-Registry
5. Select the Amazon ECR in **Action provider**.
6. Select the **omnidocs10.1web** Docker image that needs to deploy to the Dev stage in the Repository name.

7. In **Image tag – optional**, select the image tag that you want to use to set up the continuous deployment trigger.
8. In **Variable namespace – optional**, specify the unique namespace, that is, **AWS-ECR**. This is required to use its output variable in the following sections.
9. In **Output artifacts**, specify the unique variable name that is, **SourceArtifact1**. SourceArtifact is already used by AWS CodeCommit action.
10. Click **Done**.

Edit action

Action name
Choose a name for your action
AWS-ECR-Registry
No more than 100 characters

Action provider
Amazon ECR

Repository name
Choose an Amazon ECR repository as the source location.
omnidocs10.1web

Image tag - *optional*
Choose the image tag that triggers your pipeline when a change occurs in the image repository.
patch3hf19-alpine-openjdk
If an image tag is not selected, defaults to latest

Variable namespace - *optional*
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)
AWS-ECR

Output artifacts
Choose a name for the output of this action.
SourceArtifact1
No more than 100 characters

Cancel Done

Figure 4.19

11. Click **Done** on the Edit source stage.
12. Select the **Edit build stage**.
13. Click **Edit** in the AWS CodeBuild action.

Edit: Build Cancel Delete Done

+ Add action group

Build
AWS CodeBuild
+ Add action

+ Add action group

Figure 4.20

14. Add three new environment variables as given in the table below:

Name	Value	Type
IMAGE_REGISTRY_ID	#{AWS-ECR.RegistryId}	Plaintext
IMAGE_REPOSITORY_NAME	#{AWS-ECR.RepositoryName}	Plaintext
IMAGE_TAG	#{AWS-ECR.ImageTag}	Plaintext

Here, **AWS-ECR** is the name of the variable namespace, created in the Amazon ACR action.

Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Name	Value	Type	
AWS_DEFAULT_REGION	ap-south-1	Plaintext	Remove
AWS_CLUSTER_NAME	Omnidocs-uat2	Plaintext	Remove
YAML_FILE	Omnidocs10.1Web.yml	Plaintext	Remove
CODE_PIPELINE_EXECUTION_ID	#{codepipeline.PipelineExecutionId}	Plaintext	Remove
IMAGE_REGISTRY_ID	#{AWS-ECR.RegistryId}	Plaintext	Remove
IMAGE_REPOSITORY_NAME	#{AWS-ECR.RepositoryName}	Plaintext	Remove
IMAGE_TAG	#{AWS-ECR.ImageTag}	Plaintext	Remove

Add environment variable

Figure 4.21

15. Click **Done** on the edit action panel.

16. In the Edit build stage, click **Done**.

17. Click **Save** to save the pipeline.

Developer Tools > CodePipeline > Pipelines > Omnidocs101Web-DevStage > Edit Omnidocs101Web-DevStage

Editing: Omnidocs101Web-DevStage Delete Cancel Save

Edit: Source Edit stage

Source + AWS-ECR-Registry -
AWS CodeCommit Amazon ECR

Figure 4.22

NOTE:

Once the AWS ECR is integrated into the pipeline, it creates a new CloudWatch event rule that acts as a deployment trigger. Now, whenever you push the new Docker image with the same image tag name that is defined in the Amazon ECR action in the source stage, it triggers the pipeline.

As described in the **Add Source Stage**, the AWS CodeCommit action creates a new CloudWatch event rule, and it triggers the AWS CodePipeline whenever there is a change in the CodeCommit repository in which the CloudWatch event rule is disabled once the pipeline is created.

Perform the below steps to disable the CloudWatch event rule created against the AWS CodeCommit action:

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>
2. In the **Events** tab, click **Rules** on the navigation panel.
3. Search the rule created against the AWS CodeCommit repository **Genesis-CodeCommit-Repository** created in the **Creation of AWS CodeCommit Repository**.

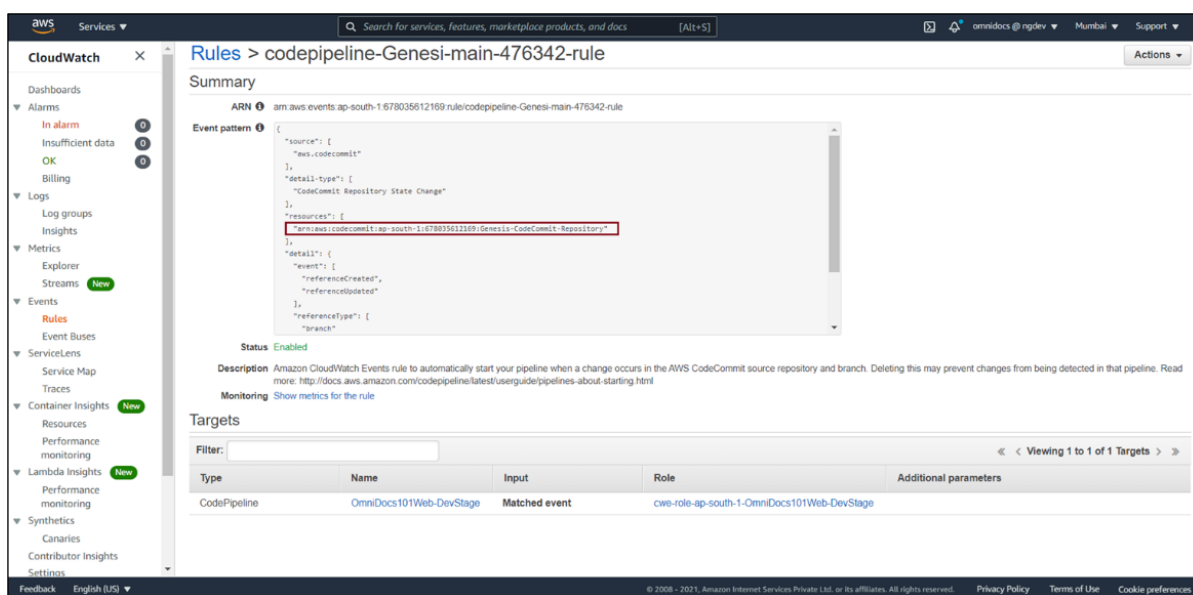


Figure 4.23

4. Select the rule, go to the **Actions** menu, and select **Disable** or **Delete**. It does not trigger the pipeline whenever any change is done in the AWS CodeCommit repository.

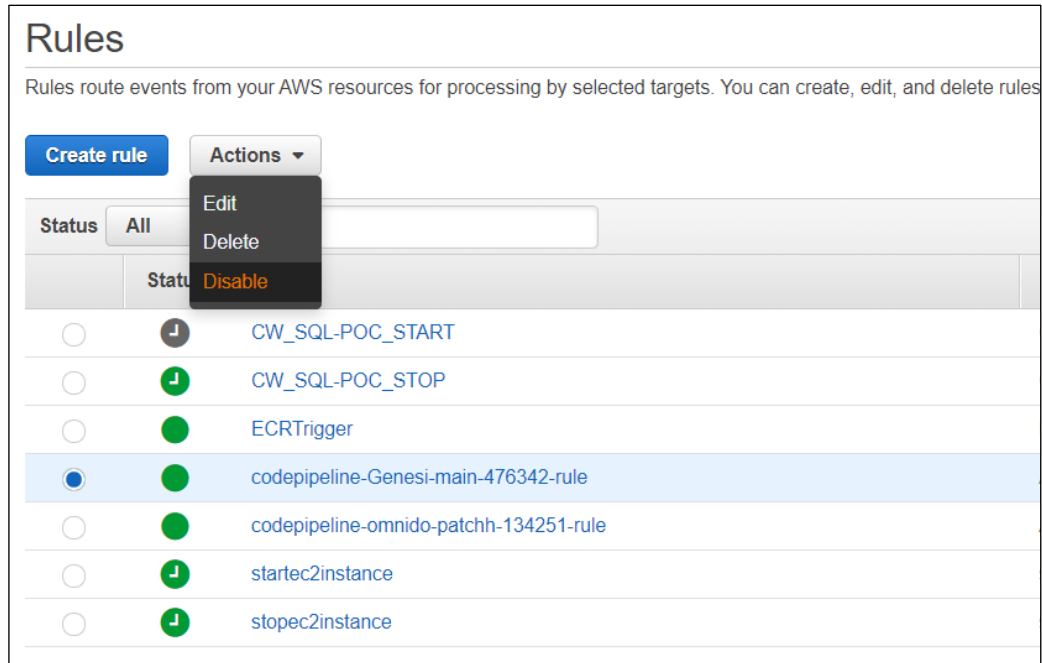


Figure 4.24

5. Click **Release change** to trigger the pipeline manually.

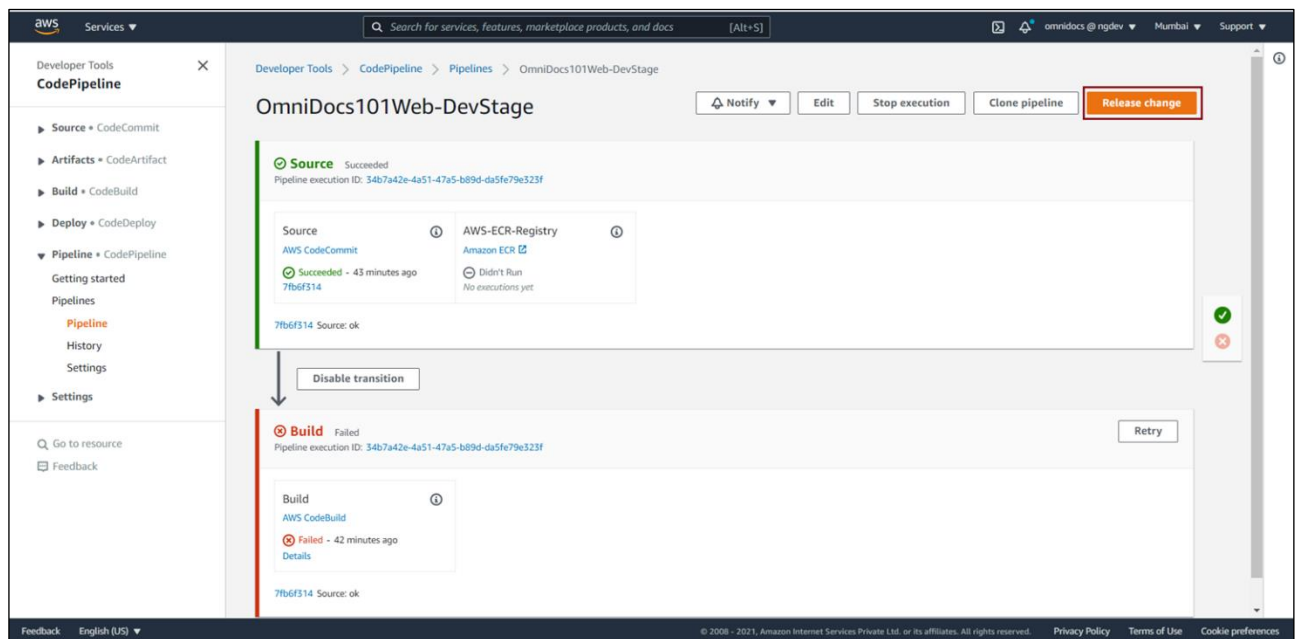


Figure 4.25

4.5.4.2 Configuring notification

This section contains the configuration of notification using the AWS **SNS topic**, to notify the recipient(s) about the pipeline execution status whether it succeeded or failed. The following are the steps to configure notifications:

1. **Create an SNS topic**
2. **Create a subscription to the SNS topic**
3. **Create a Lambda function**
4. **Create a CloudWatch event rule**

Perform the below steps to create an SNS topic:

1. Sign in to the Amazon SNS console <https://console.aws.amazon.com/sns/home>
2. Select the **Region** to create your repository on the navigation panel.
3. Select **Topics** in the left navigation panel.
4. Select **Create topic**.
5. By default, the console creates a FIFO topic, select **Standard**.
6. Enter the Name for the topic such as **SNSTopic1** in the **Details** section.
7. In **Display name – optional**, use display name such as **DevOps Admin**.
8. Select **Create topic**.

The screenshot shows the 'Create topic' page in the AWS SNS console. The 'Details' section is active, showing the 'Type' dropdown menu. The 'Standard' option is selected, which is highlighted in light blue. The 'Name' field contains 'SNSTopic1' and the 'Display name - optional' field contains 'DevOps Admin'. The 'FIFO' option is unselected and has a white background.

Figure 4.26

Perform the below steps to create a subscription to the SNS topic:

1. In the left navigation pane, select **Subscriptions**.
2. Click **Create subscription**. The Create Subscription screen appears.

3. Select the **Topic ARN**.
4. Select **Email** for **Protocol**.
5. In **Endpoint**, enter an email address to receive notifications.
6. Select the **Create subscription**.

The screenshot shows the 'Create subscription' form in the AWS console. The form is titled 'Create subscription' and has a 'Details' section. In the 'Details' section, there are three input fields: 'Topic ARN' with the value 'arn:aws:sns:us-east-1:678035612169:SNSTopic1', 'Protocol' set to 'Email', and 'Endpoint' with the value 'vivek_kumar@newgen.co.in'. Below these fields is a blue information box that says 'After your subscription is created, you must confirm it. Info'. There are two expandable sections: 'Subscription filter policy - optional' and 'Redrive policy (dead-letter queue) - optional'. At the bottom right of the form are 'Cancel' and 'Create subscription' buttons.

Figure 4.27

7. Check your email inbox and select **Confirm subscription** in the email from AWS Notifications. The sender ID is usually no-reply@sns.amazonaws.com. The Amazon SNS opens in a web browser and displays a subscription confirmation with your subscription ID.
8. Create more subscriptions and attached them to the same topic that you created to send emails to multiple recipients.
9. Once your subscription is created, click **Confirm**.

Perform the below steps to create a Lambda function:

1. Open the function page on the lambda console:
<https://console.aws.amazon.com/lambda/home>
2. Select **Create function**.
3. In the **Function name**, specify the unique function name such as **lambda-fuction1**.
4. In **Runtime**, select **python 3.8** or the latest version.
5. Keep the other settings as default and select the **Create function**.

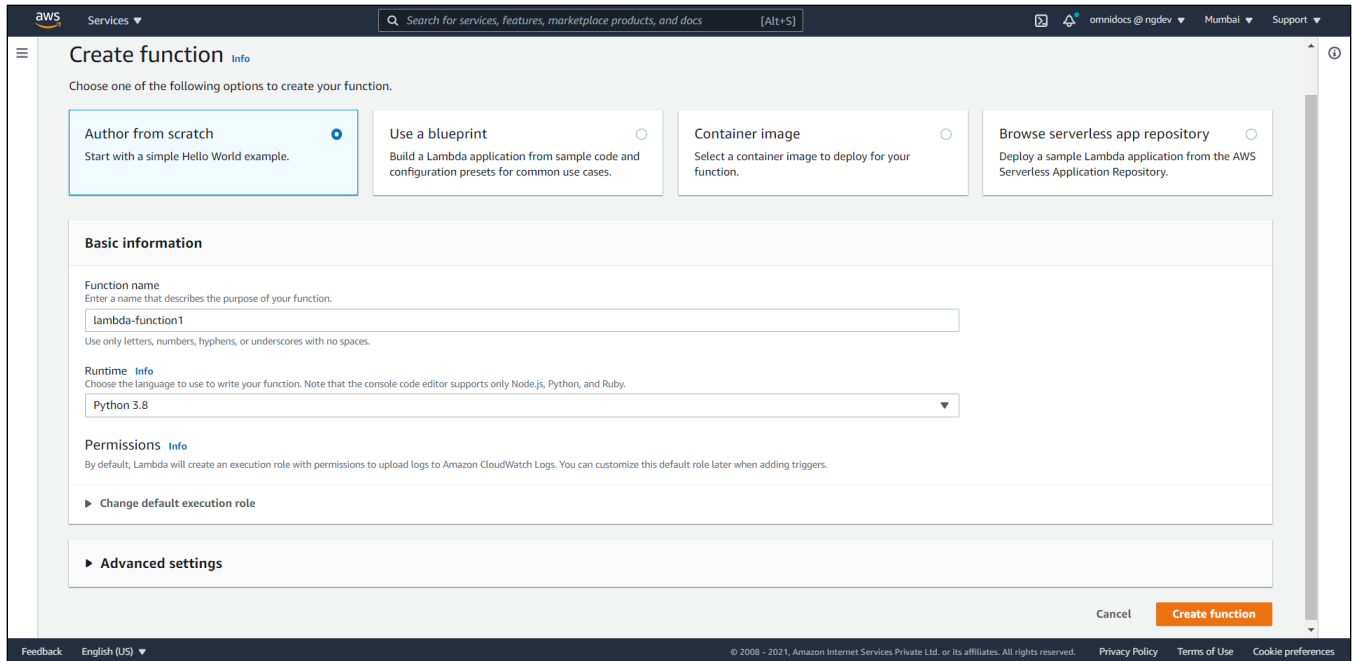


Figure 4.28

6. Under the **Code** tab, select **lambda_function.py**.
7. Replace the default code snippet using the below code snippet and select **Deploy**:

```
import json
import boto3
sns = boto3.client('sns')

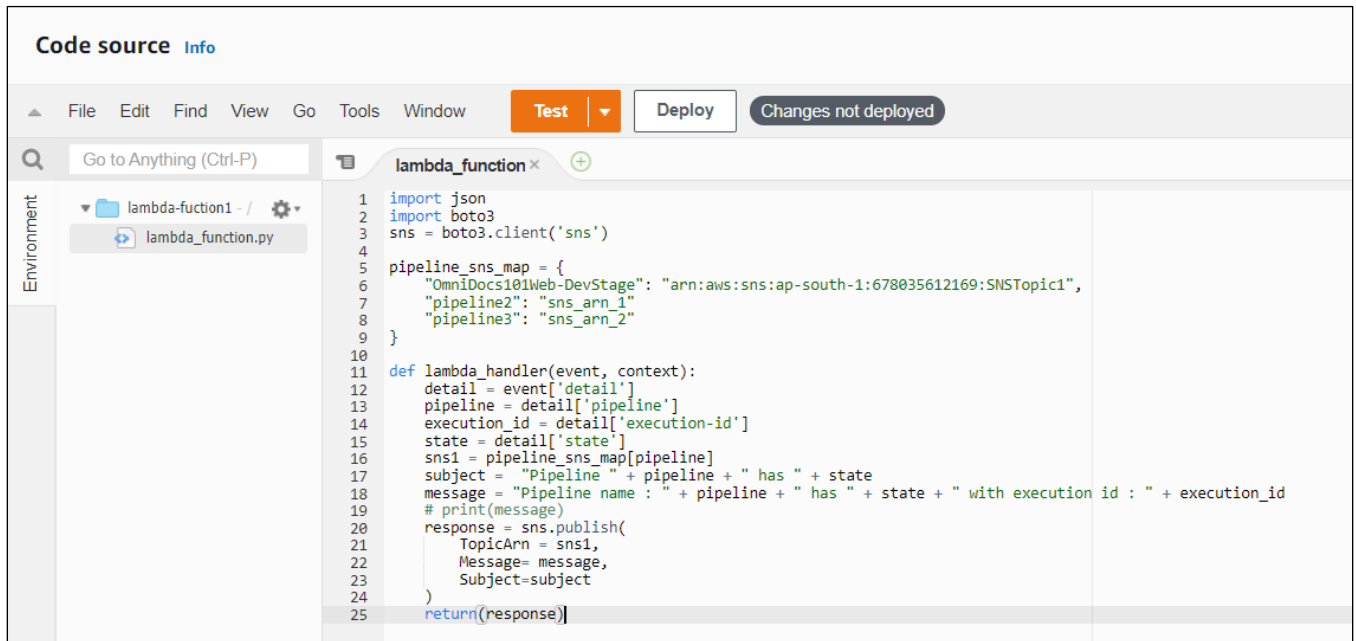
pipeline_sns_map = {
    "pipeline1": "sns_arn_1",
    "pipeline2": "sns_arn_1"
    "pipeline3": "sns_arn_2"
}

def lambda_handler(event, context):
    detail = event['detail']
    pipeline = detail['pipeline']
    execution_id = detail['execution-id']
    state = detail['state']
    sns1 = pipeline_sns_map[pipeline]
    subject = "Pipeline " + pipeline + " has " + state
    message = "Pipeline name : " + pipeline + " has " + state + " with
    execution id : " + execution_id
    # print(message)
    response = sns.publish(
        TopicArn = sns1,
        Message= message,
        Subject=subject
    )
    return(response)
```


- In the above code snippet, update **pipeline name(s)** in pipeline1, pipeline2, and pipeline3 as well as update the **SNS topic(s)** ARN at the place of sns_arn_1, sns_arn_2.

NOTE:

You can use the same SNS topic for all the pipelines or different SNS topics for each pipeline. Add an entry for each newly created pipeline and its associated SNS topic to use them.



```
Code source Info
File Edit Find View Go Tools Window Test Deploy Changes not deployed
Go to Anything (Ctrl-P)
Environment
  lambda-fuction1 - /
  lambda_function.py
1 import json
2 import boto3
3 sns = boto3.client('sns')
4
5 pipeline_sns_map = {
6   "OmniDocs101Web-DevStage": "arn:aws:sns:ap-south-1:678035612169:SNSTopic1",
7   "pipeline2": "sns_arn_1",
8   "pipeline3": "sns_arn_2"
9 }
10
11 def lambda_handler(event, context):
12     detail = event['detail']
13     pipeline = detail['pipeline']
14     execution_id = detail['execution-id']
15     state = detail['state']
16     sns1 = pipeline_sns_map[pipeline]
17     subject = "Pipeline " + pipeline + " has " + state
18     message = "Pipeline name : " + pipeline + " has " + state + " with execution id : " + execution_id
19     # print(message)
20     response = sns.publish(
21         TopicArn = sns1,
22         Message = message,
23         Subject = subject
24     )
25     return(response)]
```

Figure 4.29

- Go to the **Configuration** tab and select **Permissions**.
- Select the created **IAM role** for this lambda function. The IAM role **Summary** screen appears.

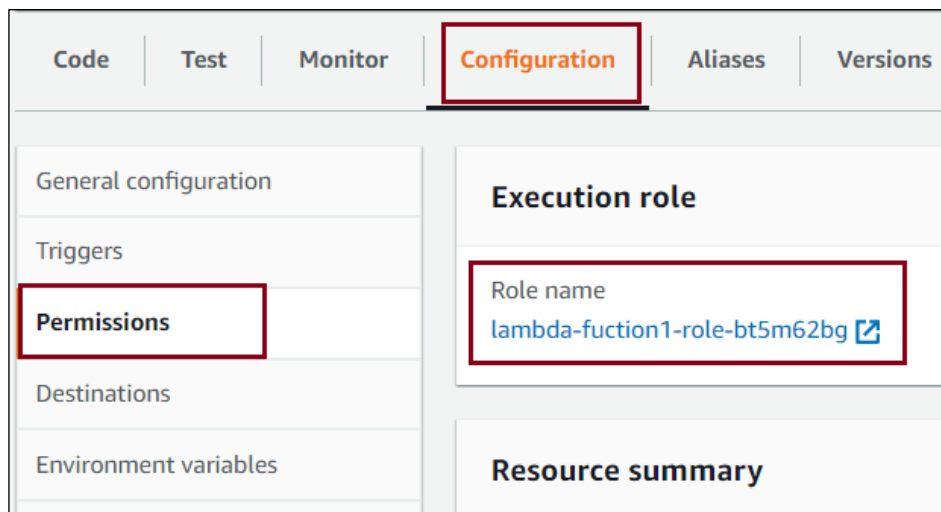


Figure 4.30

11. Select the **Add inline policy**.
12. In **Service**, select the **SNS**.
13. In **Actions**, select **Publish**.
14. Select **All resources** in **Resources**.
15. Select the **Review policy**.

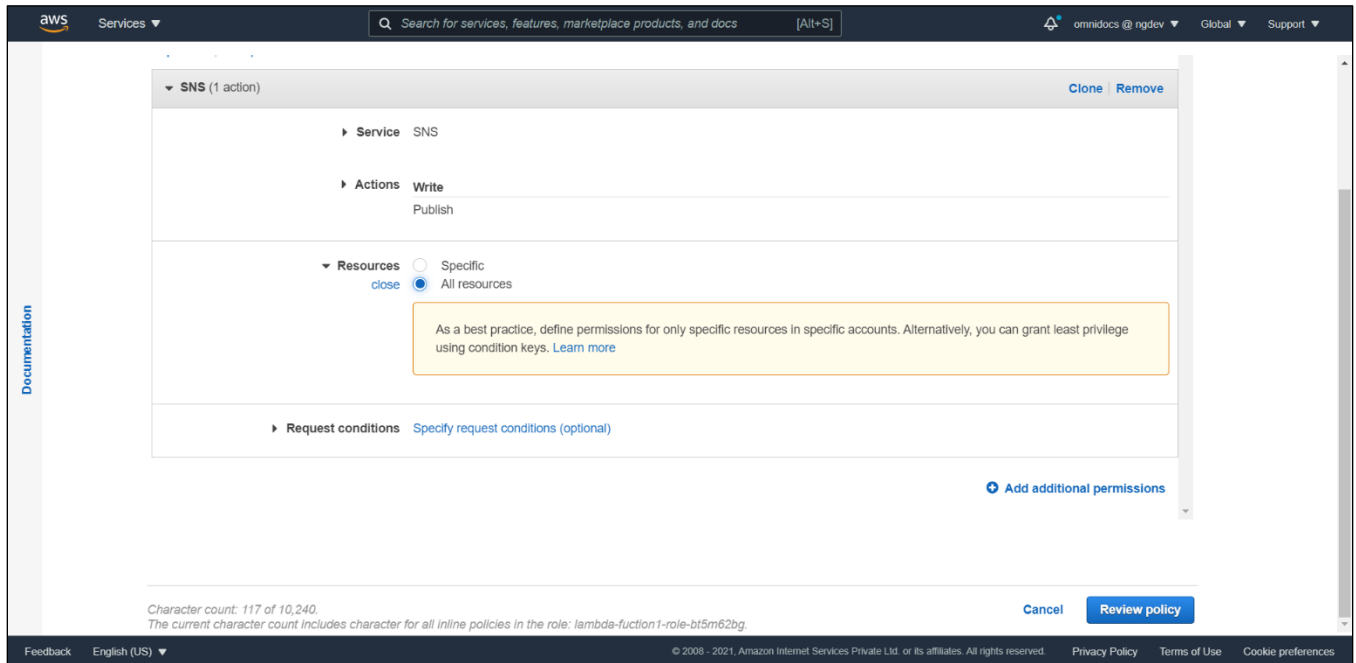


Figure 4.31

16. Specify the policy name such as **sns-lambda-policy**.
17. Select **Create policy**.

Perform the below steps to create a CloudWatch event rule:

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>
2. In the Events tab, select the **Rules** on the navigation pane.
3. Select the **Create rule**.
4. Select the **Event Pattern** radio button in the Event Source.
5. Select the **Service Name** as **CodePipeline** using the dropdown.
6. Select the **Event Type** as **CodePipeline Pipeline Execution State Change** using the dropdown.
7. Select the **Specified state(s)** and select **FAILED** and **SUCCEEDED** states.
8. Click **+Add target*** given on the upper-right.
9. Select the **Lambda function** as a target.
10. For **Function***, select the existing function name **lambda-function1**.
11. Select the **Configure details** given at the lower right.

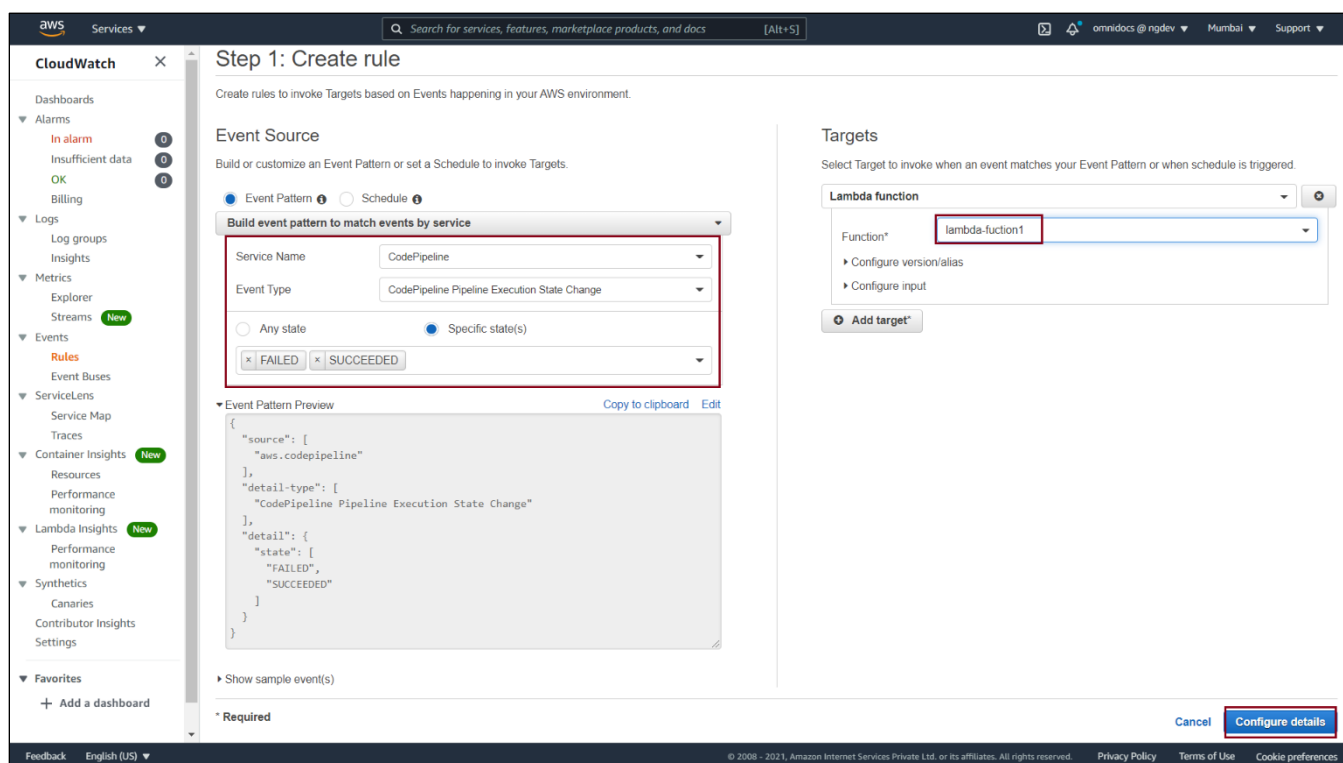


Figure 4.32

12. Specify the **Unique Rule Name** and **Description** on the **Rule definition** tab.

4.5.4.3 Configuring AWS CodePipeline for UAT stage

UAT deployments are based on approval and are available on-demand. To deploy to the UAT environment, you need to trigger the UAT deployment. Once deployment is triggered, an approval mail is sent to the project manager or the concerned team. Upon receiving the approval, the UAT deployment gets started automatically.

Perform the below steps to configure the UAT Stage:

1. Open the AWS CodePipeline console at: <http://console.aws.amazon.com/codesuite/codepipeline/home>
2. On the **Welcome** tab, select the **Create pipeline**.
3. Specify the required details in the following steps. Once complete, select **Create pipeline** at the Review step:
 - Select the pipeline settings and specify the following:
 - i. Enter a unique name for your pipeline that is, **OmniDocs101Web-UATStage** in the **Pipeline name**.
 - ii. Select the **New service** role in the Service role.

- iii. Select the checkbox **Allow AWS CodePipeline to create a service role so it can be used with this new pipeline.**
- iv. Keep the other settings as default and click **Next.**

The screenshot shows the 'Choose pipeline settings' dialog box. It has a title bar with 'Choose pipeline settings' and an 'Info' link. Below the title bar is a section titled 'Pipeline settings'. Inside this section, there are three main areas: 1. 'Pipeline name' with a text input field containing 'OmniDocs101Web-UATStage' and a note 'No more than 100 characters'. 2. 'Service role' with two radio button options: 'New service role' (selected) and 'Existing service role'. 3. 'Role name' with a text input field containing 'AWSCodePipelineServiceRole-ap-south-1-OmniDocs101Web-UATStage' and a checked checkbox labeled 'Allow AWS CodePipeline to create a service role so it can be used with this new pipeline'. At the bottom of the dialog, there is a 'Cancel' button and a 'Next' button.

Figure 4.33

- 4. Add source stage and specify the following:
 - i. Select the **AWS CodeCommit** in the **Source provider**.
 - ii. Select the existing AWS CodeCommit repository **Genesis-CodeCommit-Repository** created in **Creation of AWS CodeCommit Repository** in **Repository name**.
 - iii. Select the **Main** in the **Branch name**.
 - iv. Select the recommended option **Amazon CloudWatch Events** in the **Change detection options**.

NOTE:

Amazon CloudWatch Events creates a CloudWatch event rule. Once any changes are done in the integrated AWS CodeCommit repository, it triggers the pipeline. But do not trigger the AWS CodePipeline whenever there is a change in the CodeCommit repository. The pipeline must be triggered whenever you push a new image to the container registry like AWS ECR. Refer to the following sections for the configuration of AWS ECR with CodePipeline to disable the CloudWatch event rule once the pipeline is created.

- v. Keep the other settings as default and click **Next**.

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit

Repository name
Choose a repository that you have already created where you have pushed your source code.

Genesis-CodeCommit-Repository

Branch name
Choose a branch of the repository

main

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

Amazon CloudWatch Events (recommended)
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

AWS CodePipeline
Use AWS CodePipeline to check periodically for changes

Output artifact format
Choose the output artifact format.

CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include git metadata about the repository.

Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full git clone. Only supported for AWS CodeBuild actions.

Cancel Previous Next

Figure 4.34

5. Add the build stage and specify the following:
 - i. Select the **AWS CodeBuild** in the **Build provider** using the dropdown.
 - ii. Select a **Region** in which you want to create your pipeline using the dropdown.
 - iii. Select the existing CodeBuild project **OmniDocs101Web** created in the **Creation of AWS CodeBuild Project**.
 - iv. For **Environment variables – optional**, create the below environment variables:

Name	Value	Type
AWS_DEFAULT_REGION	ap-south-1	Plaintext
AWS_CLUSTER_NAME	OmniDocs-uat2	Plaintext
YAML_FILE	OmniDocs10.1Web.yml	Plaintext
CODE_PIPELINE_EXECUTION_ID	{codepipeline.PipelineExecutionId}	Plaintext

- **AWS_DEFAULT_REGION:** Specify the region where the AWS EKS cluster is created.
 - **AWS_CLUSTER_NAME:** Specify the EKS cluster name created for the UAT stage.
 - **YAML_FILE:** Specify the YAML file name that is stored in the AWS CodeCommit repository and that is used to deploy the OmniDocs10.1Web container for UAT Stage.
 - **CODE_PIPELINE_EXECUTION_ID:** This variable is just created for logging purposes so that you can track the build-ID and its initiated pipeline.
- v. In **Build type**, select the Single build and click **Next**.

Build - optional

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

Region
Asia Pacific (Mumbai)

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

OmniDocs101Web or Create project

Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. Learn more

Name	Value	Type	
AWS_DEFAULT_REGION	ap-south-1	Plaintext	Remove
AWS_CLUSTER_NAME	OmniDocs-uat2	Plaintext	Remove
YAML_FILE	OmniDocs10.1Web.yml	Plaintext	Remove
CODE_PIPELINE_EXECUTION_ID	#codepipeline.Pipeline	Plaintext	Remove

Add environment variable

Build type

Single build
Triggers a single build.

Batch build
Triggers multiple builds as a single execution.

Cancel Previous Skip build stage Next

Figure 4.35

6. In the **Add deploy stage**, skip the deploy stage.

Deploy - optional

Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Cancel Previous Skip deploy stage Next

Figure 4.36

7. In the **Review**, select **Create pipeline**.

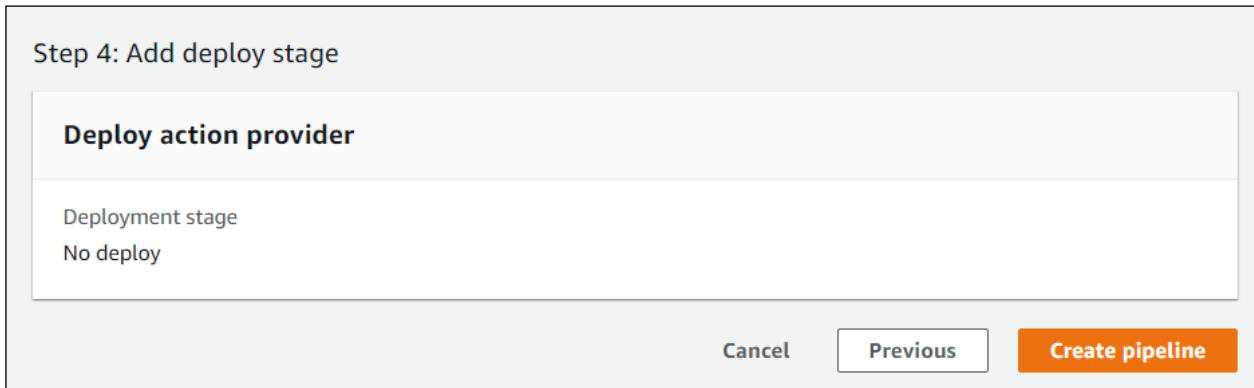


Figure 4.37

NOTE:

Once the pipeline is created, it starts its first pipeline execution. This execution fails as expected when you have not yet integrated the AWS ECR into the pipeline. You must perform the same.

Perform the below steps to integrate AWS ECR into the AWS CodePipeline:

1. Open the created pipeline **OmniDocs101Web-UATStage** in **Edit** mode.
2. Select the **Edit source stage**.
3. Select **+ Add action**.

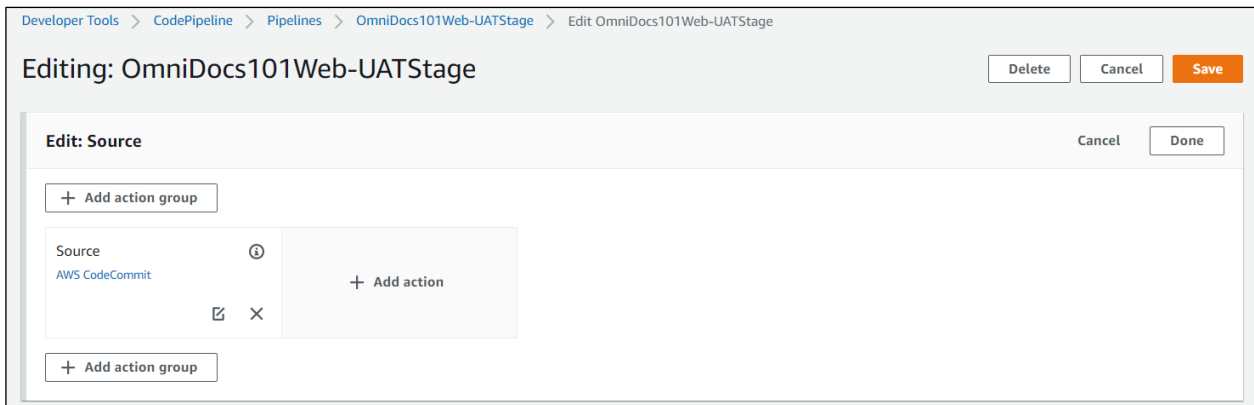


Figure 4.38

4. In the **Edit action** panel, specify the unique **Action name**. that is, AWS-ECR-Registry.
5. Select the **Amazon ECR** in the **Action provider**.
6. Select the **omnidocs10.1web** in the **Repository name**. It enables the Docker image to deploy to the Dev stage.

7. In **Image tag – optional**, select the image tag that you want to use to set up the continuous deployment trigger.
8. In **Variable namespace – optional**, specify the unique namespace that is, AWS-ECR. This is required while using its output variable in the following sections.
9. In **Output artifacts**, specify the unique variable name. that is, **SourceArtifact1**. SourceArtifact is already used by AWS CodeCommit action.
10. Click **Done**.

The screenshot shows the 'Edit action' dialog box with the following fields and values:

- Action name:** AWS-ECR-Registry
- Action provider:** Amazon ECR
- Repository name:** omnidocs10.1web
- Image tag - optional:** patch3hf19-alpine-openjdk-uat
- Variable namespace - optional:** AWS-ECR
- Output artifacts:** SourceArtifact1

Buttons for 'Cancel' and 'Done' are visible at the bottom right.

Figure 4.39

11. Click **Done** on the Edit source stage.
12. Select the **Edit build stage**.
13. Click the **Edit** icon for AWS CodeBuild action.

The screenshot shows the 'Edit: Build' dialog box with the following elements:

- Stage:** Build
- Action:** AWS CodeBuild
- Action icon:** Edit icon (highlighted with a red box)
- Buttons:** Cancel, Delete, Done

Figure 4.40

14. Add the three new environment variables given in the table below:

Name	Value	Type
IMAGE_REGISTRY_ID	#{AWS-ECR.RegistryId}	Plaintext
IMAGE_REPOSITORY_NAME	#{AWS-ECR.RepositoryName}	Plaintext
IMAGE_TAG	#{AWS-ECR.ImageTag}	Plaintext

Where, **AWS-ECR** is the name of the variable namespace, created in Amazon ACR action.

Figure 4.41

15. On the Edit action panel, click **Done**.

16. Click **Done** on the Edit build stage. Since this is the UAT stage and it must be an **approval-based** pipeline.

17. Select the **+ Add stage** in between the Source stage and Build stage.

Figure 4.42

18. Specify the Stage name such as **Approval**.
19. Click **+Add action group** under the **Approval** stage.



Figure 4.43

20. Specify the unique action name such as **Approval-for-UAT** in the **Action name**.
21. Select the **Manual approval** in the **Action provider**.
22. Select the ARN of SNSTopic1 created in **Configuration of Notification** in the **SNS topic ARN – optional**.
23. Specify a comment to display for the reviewer in the email notifications or the console in **Comments-optional**. For Example, provide your approval for UAT deployment.
24. Keep the other settings as default and click **Done**.

Figure 4.44

25. Click **Done** on the Approval stage.
26. Click **Save** in the upper-right to save the pipeline.

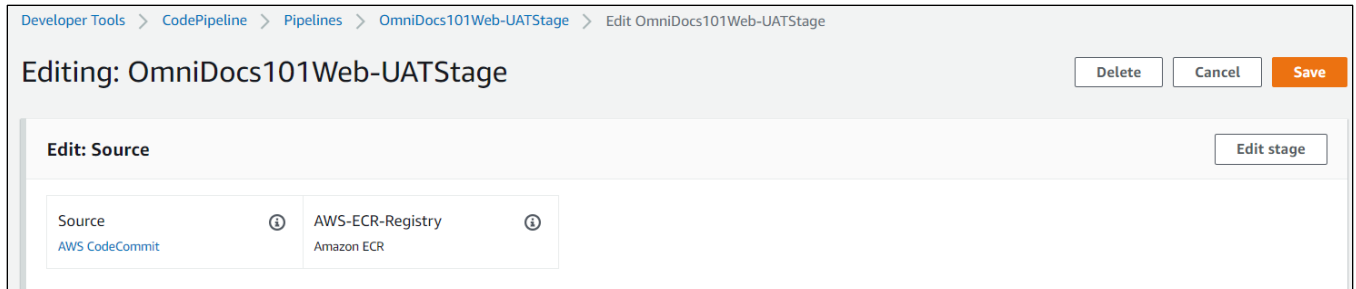


Figure 4.45

NOTE:

Upon integration of the AWS ECR into the pipeline, this adds the target into the existing CloudWatch event rule that acts as a deployment trigger. Now, whenever you push the new Docker image with the same image tag name that is defined in the Amazon ECR action in the source stage, it triggers the pipeline. But as per the UAT deployment specification, UAT deployments are based on approval and are available on demand. Disable the continuous deployment of the UAT pipeline.

Perform the below steps to remove the target from the existing CloudWatch event rule created against AWS ECR action:

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>
2. Select **Rules** in the **Events**. Search the rule created against the Amazon ECR with the same image tag name defined in the source stage.

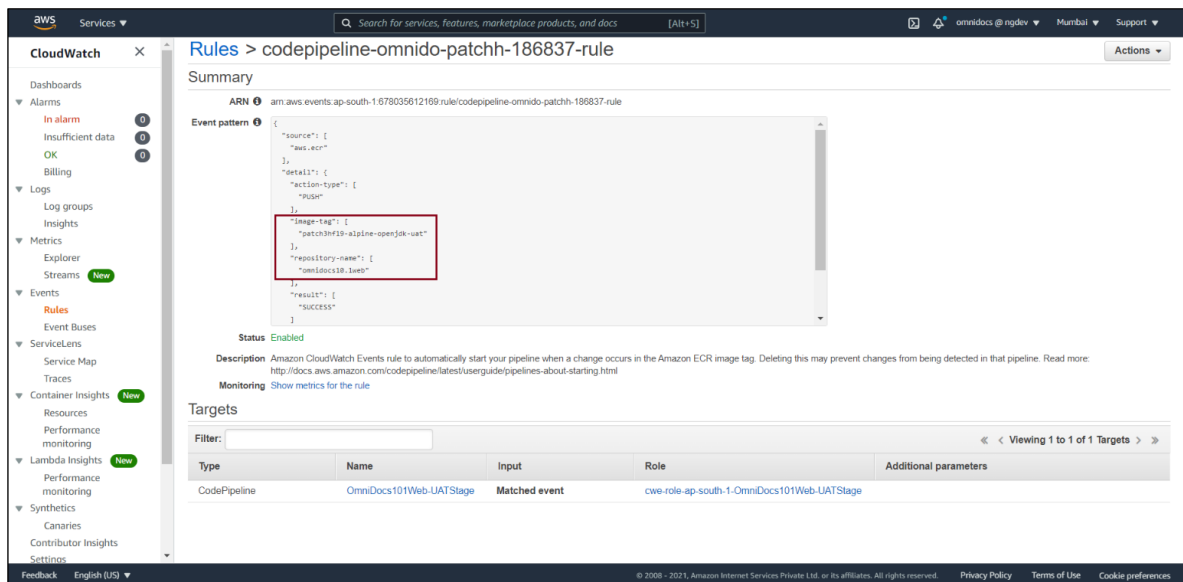


Figure 4.46

3. Select the rule, go to the **Actions** menu and select **Disable/Delete**.

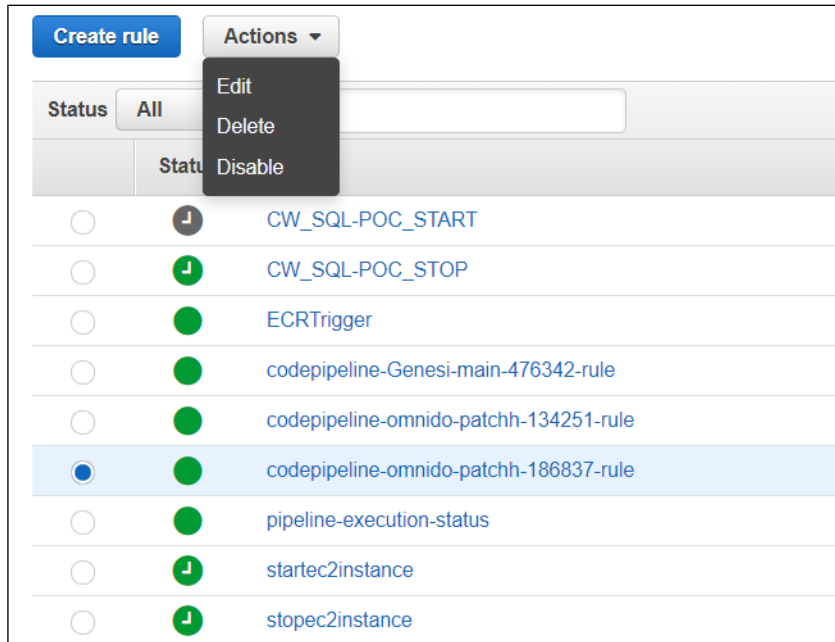


Figure 4.47

Now, it does not trigger the pipeline whenever you push the Docker image to the AWS ECR.

NOTE:

As described in the Add source stage, the AWS CodeCommit action creates a new CloudWatch event rule, and it triggers the AWS CodePipeline whenever there is a change in the CodeCommit repository. You can disable the CloudWatch event rule once the pipeline is created.

Perform the below steps to disable the CloudWatch event rule created against AWS CodeCommit action:

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>
2. Select Rules under the Events tab given on the navigation panel.
3. Search the rule created against the AWS CodeCommit repository **Genesis-CodeCommit-Repository** created in the **creation of the AWS CodeCommit Repository**.

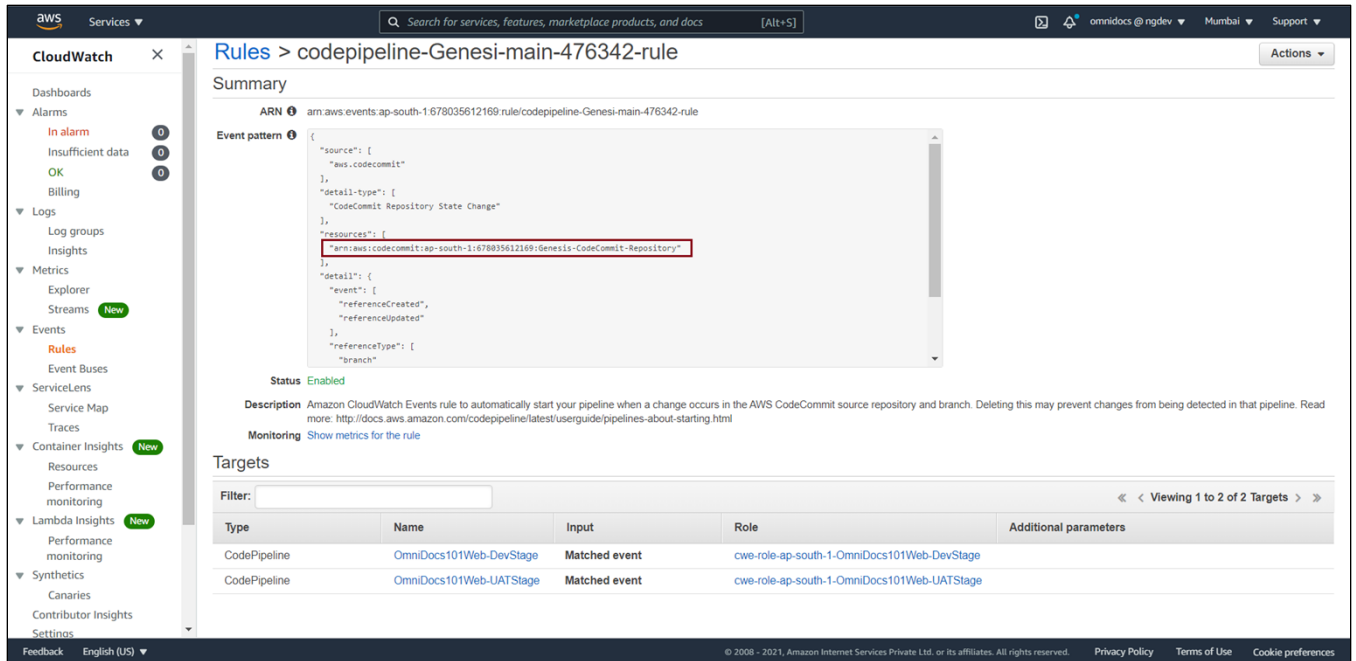


Figure 4.48

4. Select the rule, go to the **Actions** menu, and select **Disable** or **Delete**.

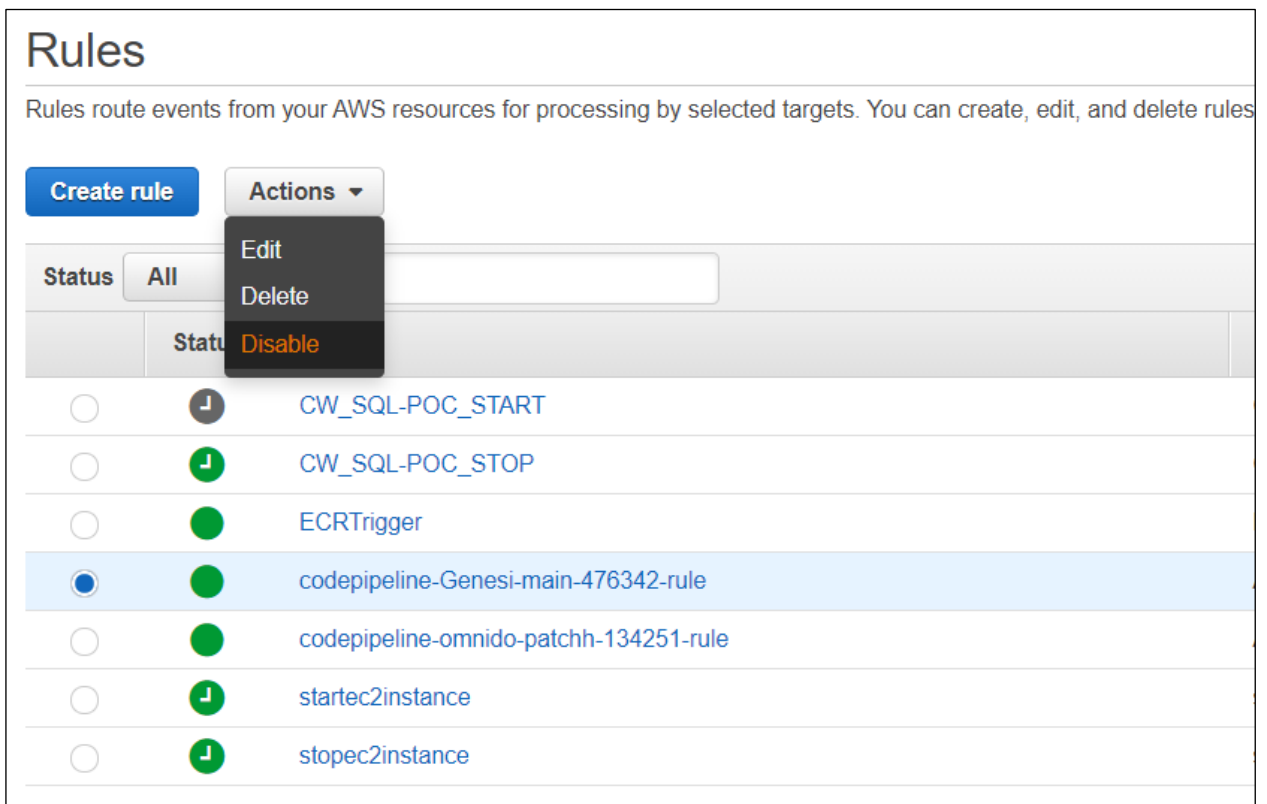


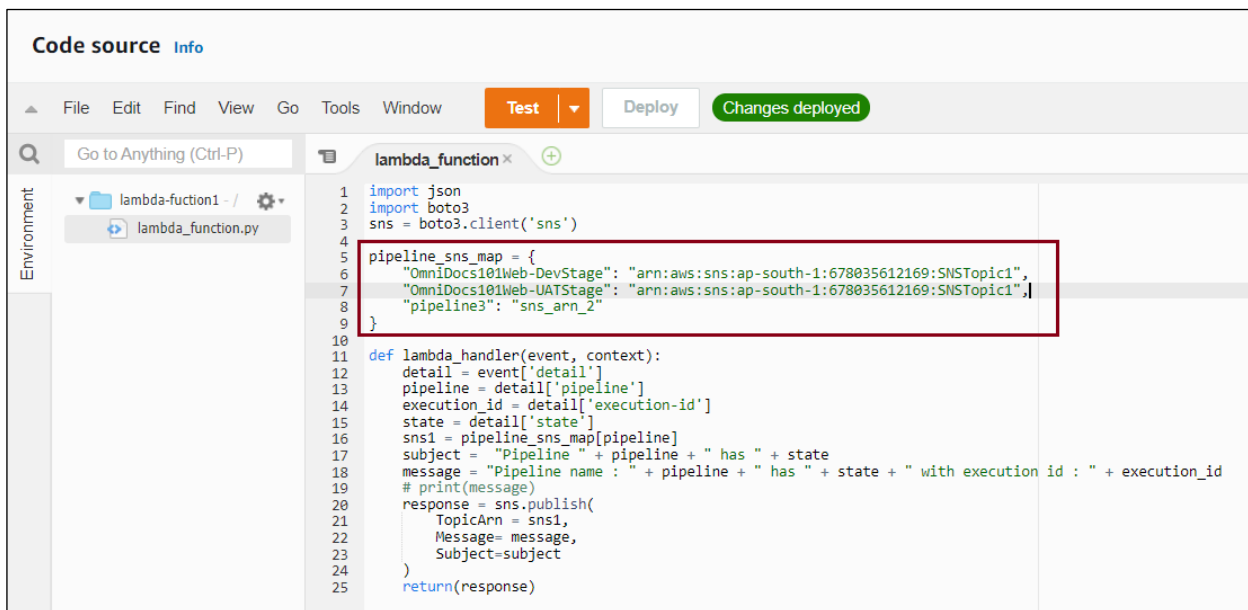
Figure 4.49

Now, it does not trigger the pipeline whenever you make any changes in the AWS CodeCommit repository.

NOTE:

Add an entry in the Lambda function `lambda_function1` created in **Configuration of AWS CodePipeline for UAT Stage**, for each newly created pipeline and its associated SNS topic that you can use. This is required to notify the recipient(s) about the pipeline execution status whether it succeeded or failed.

5. Add an entry of the pipeline **OmniDocs101Web-UATStage** in the lambda function **lambda_function1**.



```
Code source Info
File Edit Find View Go Tools Window Test Deploy Changes deployed
Go to Anything (Ctrl-P) lambda_function
Environment
  lambda-fuction1 - /
  lambda_function.py
1 import json
2 import boto3
3 sns = boto3.client('sns')
4
5 pipeline_sns_map = {
6   "OmniDocs101Web-DevStage": "arn:aws:sns:ap-south-1:678035612169:SNSTopic1",
7   "OmniDocs101Web-UATStage": "arn:aws:sns:ap-south-1:678035612169:SNSTopic1",
8   "pipeline3": "sns_arn_2"
9 }
10
11 def lambda_handler(event, context):
12     detail = event['detail']
13     pipeline = detail['pipeline']
14     execution_id = detail['execution-id']
15     state = detail['state']
16     sns1 = pipeline_sns_map[pipeline]
17     subject = "Pipeline " + pipeline + " has " + state
18     message = "Pipeline name : " + pipeline + " has " + state + " with execution id : " + execution_id
19     # print(message)
20     response = sns.publish(
21         TopicArn = sns1,
22         Message= message,
23         Subject=subject
24     )
25     return(response)
```

Figure 4.50

6. Click **Release change** to trigger the pipeline manually.

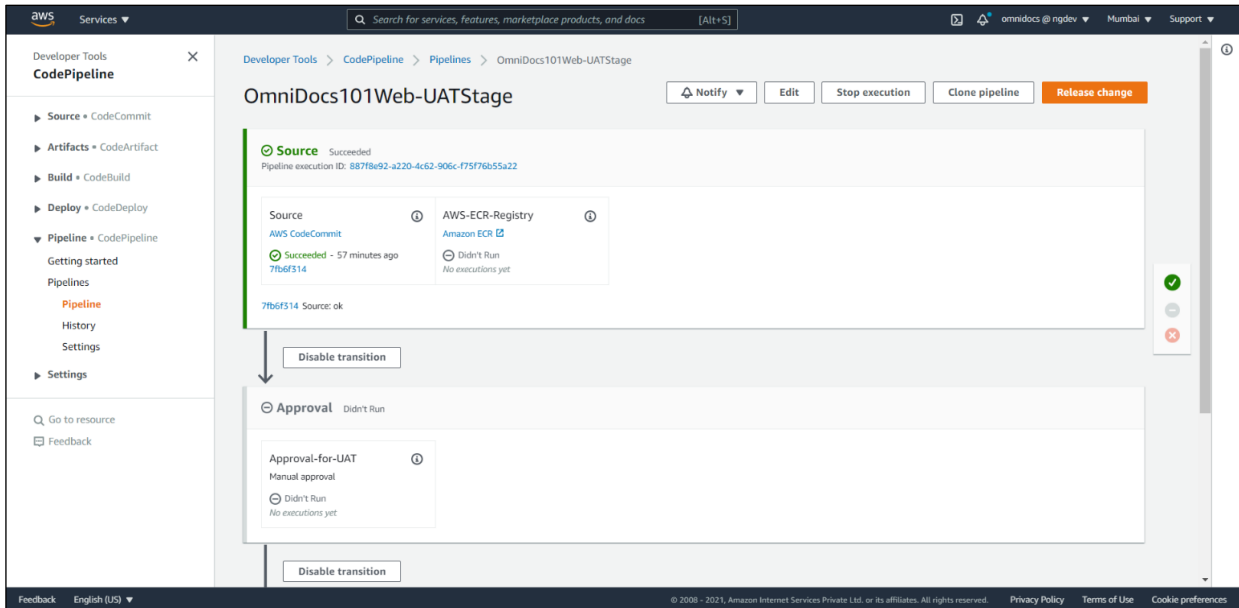


Figure 4.51

4.5.4.4 Configuring AWS CodePipeline for production stage

The production deployment is based on approval, but it has a multi-level approval system. To deploy a production environment, you require the approval of all stakeholders. Once approval is received from all the stakeholders, the deployment to the production environment is not triggered automatically. Manual intervention mail is sent to the engineer who is supposed to deploy to production with a checklist. If all the checklist points are not covered, then the deployment to production gets rejected.

Production deployment has a multi-level approval system. To support this multi-level approval, you must create multiple SNS topics in reference to **create an SNS topic** and **create a subscription to the SNS topic** with different subscriptions.

Perform the below steps to configure the Production Stage

1. Open the AWS CodePipeline console at: <http://console.aws.amazon.com/codesuite/codepipeline/home>
2. Click **Create pipeline** given on the **Welcome** tab.
Specify the required details in the following steps. Once complete, click Create pipeline at the Review step:
3. Specify the following on the **Select pipeline settings**:
 - i. Enter a unique name for your pipeline that is, **OmniDocs101Web-ProdStage** in the **Pipeline name**.
 - ii. Select the New Service Role in the **Service role**.

- iii. Select the checkbox **Allow AWS CodePipeline to create a service role so it can be used with this new pipeline.**
- iv. Keep the other settings as default and click **Next.**

The screenshot shows the 'Choose pipeline settings' dialog box. It has a title bar with 'Choose pipeline settings' and an 'Info' link. Below the title bar is a section titled 'Pipeline settings'. Inside this section, there are three main areas: 'Pipeline name' with a text input field containing 'OmniDocs101Web-ProdStage' and a note 'No more than 100 characters'; 'Service role' with two radio button options: 'New service role' (selected) and 'Existing service role'; and 'Role name' with a text input field containing 'AWSCodePipelineServiceRole-ap-south-1-OmniDocs101Web-ProdStage' and a note 'Type your service role name'. Below the role name field is a checked checkbox with the text 'Allow AWS CodePipeline to create a service role so it can be used with this new pipeline'. At the bottom of the dialog, there is a section for 'Advanced settings' with a right-pointing arrow, and two buttons: 'Cancel' and 'Next'.

Figure 4.52

4. Specify the following on the **Add source stage**:
 - i. Select the **AWS CodeCommit** in the **Source provider**.
 - ii. Select the existing AWS CodeCommit repository **Genesis-CodeCommit-Repository** created in the Configuration of Notification in the **Repository name**.
 - iii. Select **main** in the **Branch name**.
 - iv. In **Change detection options**, select the recommended option **Amazon CloudWatch Events**.

NOTE:

Amazon CloudWatch Events creates a CloudWatch event rule. As soon as you make any change in the integrated AWS CodeCommit repository, it triggers the pipeline. But you do not want to trigger the AWS CodePipeline whenever there is a change in the CodeCommit repository. The pipeline must be triggered whenever you push a new image to the container registry like AWS ECR. You can see the configuration of AWS ECR with CodePipeline in the following sections. So, you can disable the CloudWatch event rule once the pipeline is created.

- v. Keep the other settings as default and click **Next**.

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit

Repository name
Choose a repository that you have already created where you have pushed your source code.

Genesis-CodeCommit-Repository

Branch name
Choose a branch of the repository

main

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

Amazon CloudWatch Events (recommended)
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

AWS CodePipeline
Use AWS CodePipeline to check periodically for changes

Output artifact format
Choose the output artifact format.

CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include git metadata about the repository.

Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full git clone. Only supported for AWS CodeBuild actions.

Cancel Previous Next

Figure 4.53

- Specify the following in the **Add build stage**:
 - In the **Build provider**, select **AWS CodeBuild**.
 - In the **Region**, select a region in which you want to create your pipeline.
 - In the Project name, select the existing CodeBuild project **OmniDocs101Web** created in the **creation of the AWS CodeBuild Project**.
 - In the **Environment variables – optional**, create the below environment variables:

Name	Value	Type
AWS_DEFAULT_REGION	ap-south-1	Plaintext
AWS_CLUSTER_NAME	OmniDocs-uat2	Plaintext
YAML_FILE	OmniDocs10.1Web.yml	Plaintext
CODE_PIPELINE_EXECUTION_ID	#{codepipeline.PipelineExecutionId}	Plaintext

- **AWS_DEFAULT_REGION**: Specify the region where the AWS EKS cluster is created.
- **AWS_CLUSTER_NAME**: Specify the name of the EKS cluster created for the PROD stage.
- **YAML_FILE**: Specify the name of the YAML file that is stored in the AWS CodeCommit repository and is used to deploy the OmniDocs10.1Web container for the PROD Stage.
- **CODE_PIPELINE_EXECUTION_ID**: This variable is just created for logging purposes so that you can track the build-id and its initiated pipeline.

6. For **Build type**, select the Single build and click **Next**.

Build - optional

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild ▼

Region

Asia Pacific (Mumbai) ▼

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

OmniDocs101Web × or [Create project](#)

Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Name	Value	Type	
AWS_DEFAULT_REGION	ap-south-1	Plaintext ▼	Remove
AWS_CLUSTER_NAME	OmniDocs-uat2	Plaintext ▼	Remove
YAML_FILE	OmniDocs10.1Web.yml	Plaintext ▼	Remove
CODE_PIPELINE_EXECU'	#{codepipeline.Pipeline}	Plaintext ▼	Remove

[Add environment variable](#)

Build type

Single build
Triggers a single build.

Batch build
Triggers multiple builds as a single execution.

[Cancel](#)
[Previous](#)
[Skip build stage](#)
[Next](#)

Figure 4.54

7. In the **Add deploy stage**, skip the deploy stage.

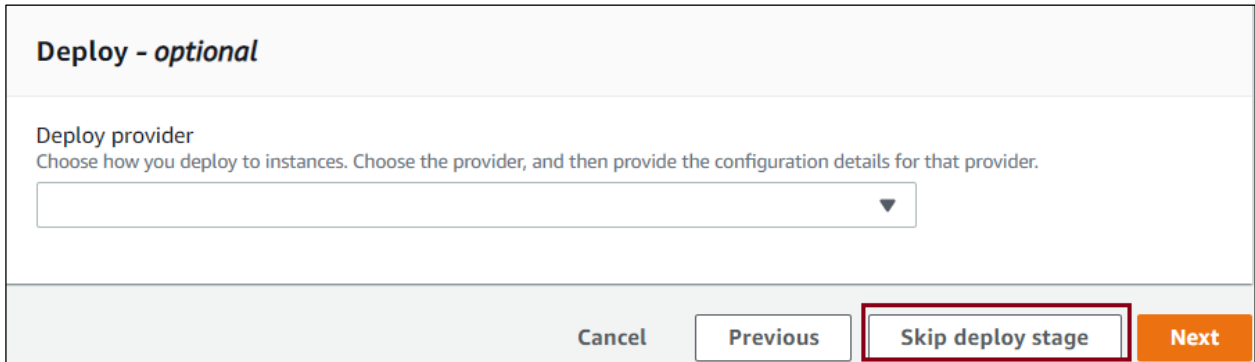


Figure 4.55

8. In the Review, click **Create pipeline**.

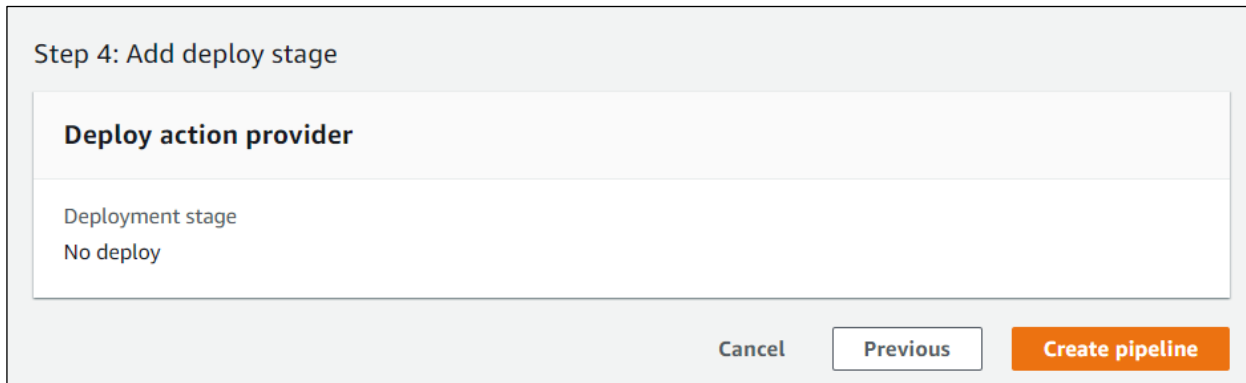


Figure 4.56

NOTE:

As soon as you create the pipeline, it starts its first pipeline execution. This execution failed as expected when you have not yet integrated the AWS ECR into the pipeline. You need to do the same.

9. Specify the following to integrate the AWS ECR into the AWS CodePipeline:
 - i. Open the created pipeline **OmniDocs101Web-ProdStage** in **Edit** mode.
 - ii. Select the **Edit source stage**.
 - iii. Select **+ Add action**.

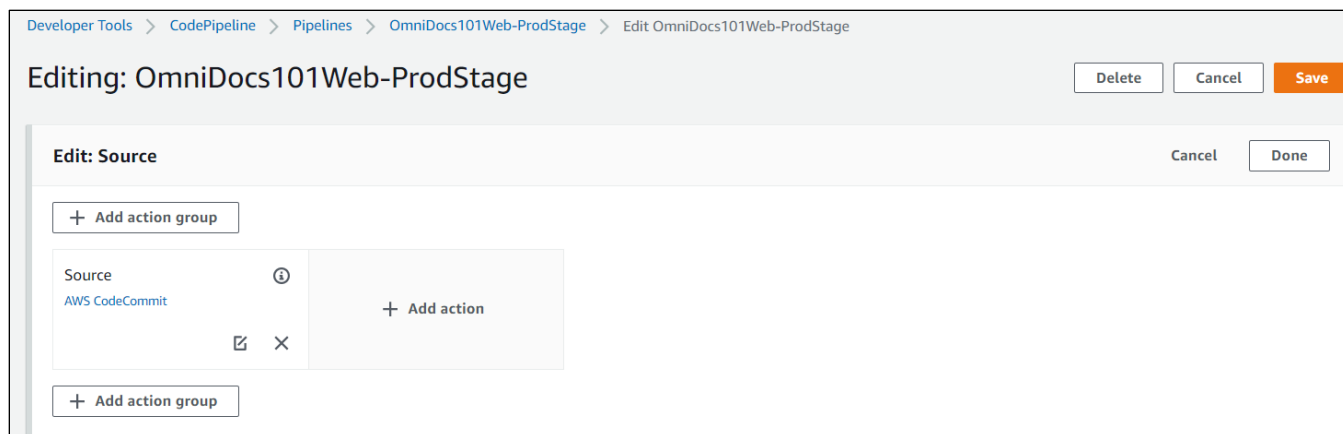


Figure 4.57

- iv. In the **Edit action** panel, specify the unique **Action name**, that is, **AWS-ECR-Registry**
- v. In the **Action provider**, select the **Amazon ECR**.
- vi. In the **Repository name**, select the **omnidocs10.1web** Docker image that needs to deploy to the Prod stage.
- vii. In the **Image tag – optional**, select the image tag that you want to use to set up the continuous deployment trigger.
- viii. In the **Variable namespace – optional**, specify the unique namespace, that is, **AWS-ECR**. This is required to use its output variable in the following sections.
- ix. In the **Output artifacts**, specify the unique variable name, that is, **SourceArtifact1**. SourceArtifact is already used by AWS CodeCommit action.
- x. Click **Done**.

Edit action

Action name
Choose a name for your action

No more than 100 characters

Action provider

Repository name
Choose an Amazon ECR repository as the source location.

Image tag - optional
Choose the image tag that triggers your pipeline when a change occurs in the image repository.

If an image tag is not selected, defaults to latest

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Output artifacts
Choose a name for the output of this action.

No more than 100 characters

Figure 4.58

- xi. Click **Done** on the Edit source stage.
- xii. Select the **Edit build stage**.
- xiii. Click the **Edit** icon for AWS CodeBuild action.

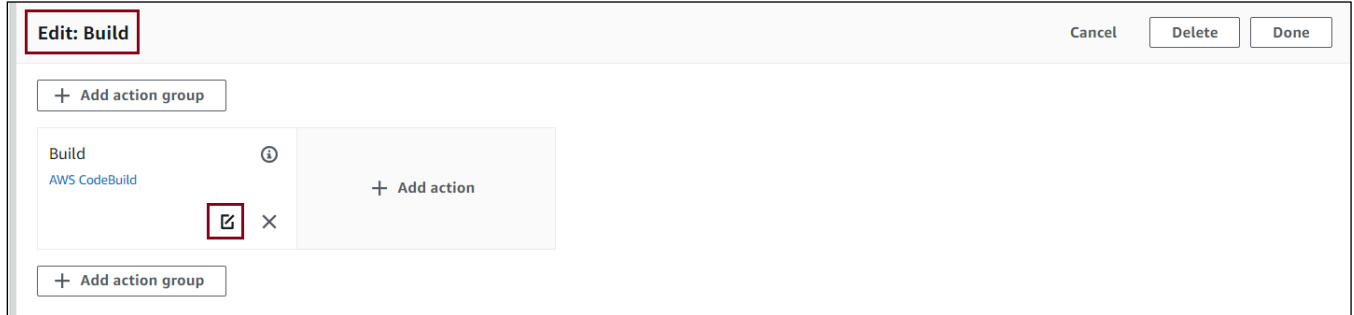


Figure 4.59

10. Add the three new environment variables as given in the table below:

Name	Value	Type
IMAGE_REGISTRY_ID	#{AWS-ECR.RegistryId}	Plaintext
IMAGE_REPOSITORY_NAME	#{AWS-ECR.RepositoryName}	Plaintext
IMAGE_TAG	#{AWS-ECR.ImageTag}	Plaintext

Where, **AWS-ECR** is the name of the variable namespace, created in Amazon ACR action.

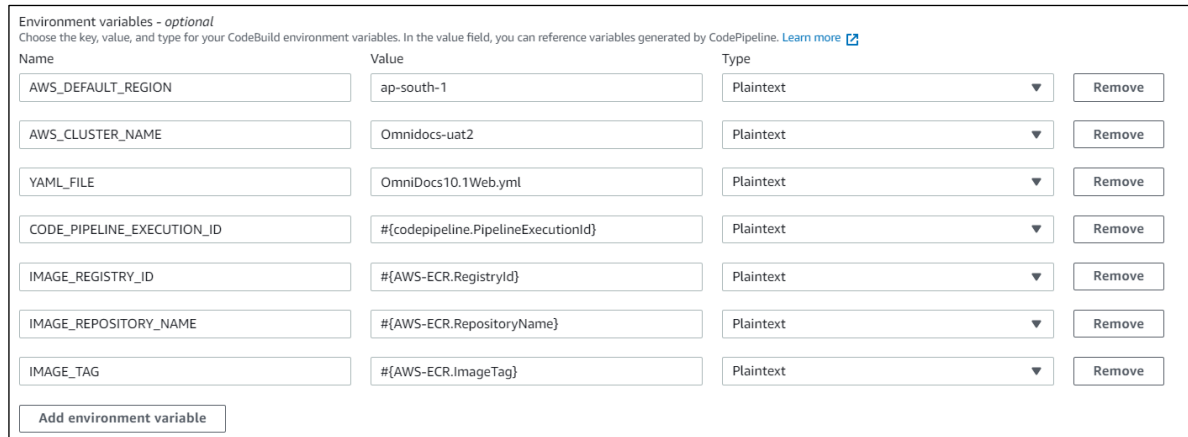


Figure 4.60

- 11. On the Edit Action Panel, click **Done**.
- 12. On the Edit build stage, click Done. Since this is the PROD stage and it must be an **approval-based** pipeline then select the **+ Add stage**.

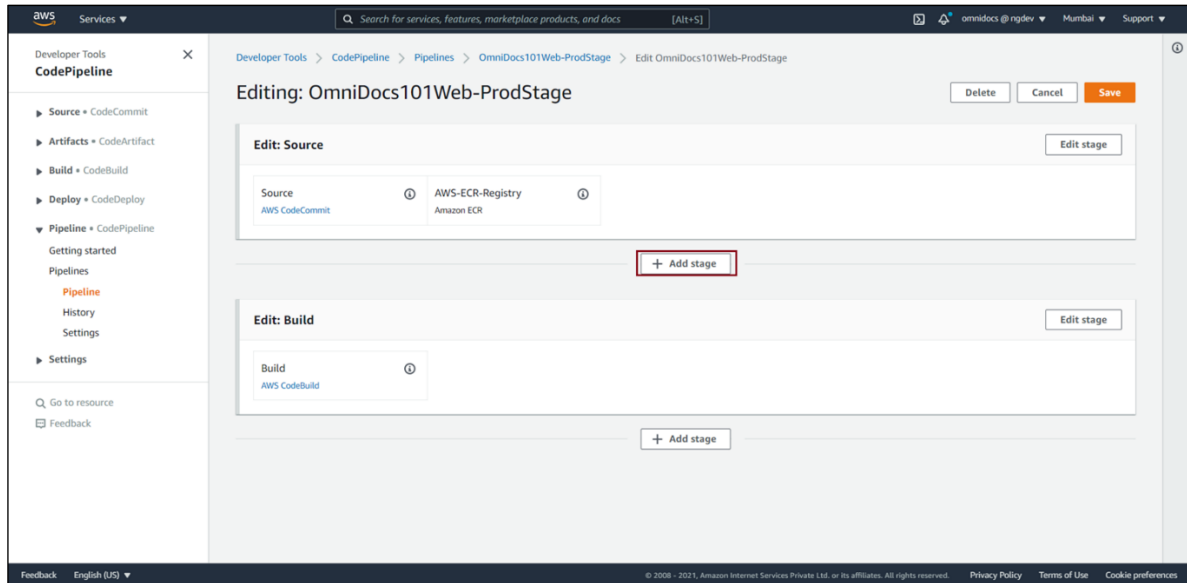


Figure 4.61

13. Specify the Stage name such as **Approval**.

- i. Click **+Add action group** under the **Approval** stage.



Figure 4.62

- ii. Specify the unique action name such as **Approval-for-PROD** in the **Action name**.
- iii. Select the **Manual approval** in the **Action provider**.
- iv. Select the ARN of SNSTopic1 created in Configuration of Notification in the **SNS topic ARN – optional**.
- v. Specify a comment to display for the reviewer in the email notifications or the console in Comments-optional. For Example, provide your approval for **PROD** deployment.
- vi. Keep the other settings as default and click **Done**.

Edit action [X]

Action name
Choose a name for your action
Approval-for-PROD
No more than 100 characters

Action provider
Manual approval

Configure the approval request.

SNS topic ARN - optional
arn:aws:sns:ap-south-1:678035612169:SNSTopic1

URL for review - optional
Type the URL you want to provide to the reviewer as part of the approval request. The URL must begin with 'http://' or 'https://'.

Comments - optional
Comments you type here display for the reviewer in email notifications or the console.
Please provide your approval for PROD deployment.

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Cancel Done

Figure 4.63

14. The pipeline has a multi-level approval system, you need to add multiple Approval actions with different SNS topics.

15. Click **+ Add action** under the **Approval** stage and specify the following:

Edit: Approval Cancel Delete Done

+ Add action group

Approval-for-PROD ⓘ
Manual approval
✎ ✕

+ Add action

+ Add action group

Figure 4.64

- i. Specify the unique action name such as **Approval-for-PROD-2** in the Action name.
- ii. Select **Manual approval** in the **Action provider**.
- iii. In the **SNS topic ARN – optional**, select the ARN of **SNSTopic2** created in the **configuration of notification**.
- iv. For **Comments – optional**, specify the comment to display for the reviewer in email notifications or the console, for example, provide your approval for PROD deployment.
- v. Keep the other settings as default and click **Done**.

Edit action [Close]

Action name
Choose a name for your action
Approval-for-PROD-2
No more than 100 characters

Action provider
Manual approval

Configure the approval request.

SNS topic ARN - *optional*
arn:aws:sns:ap-south-1:678035612169:SNSTopic2

URL for review - *optional*
Type the URL you want to provide to the reviewer as part of the approval request. The URL must begin with 'http://' or 'https://'.

Comments - *optional*
Comments you type here display for the reviewer in email notifications or the console.
Please provide your approval for PROD deployment.

Variable namespace - *optional*
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Cancel Done

Figure 4.65

- vi. Click **Done**.
- vii. Click **Save** to save the pipeline.

Developer Tools > CodePipeline > Pipelines > OmniDocs101Web-ProdStage > Edit OmniDocs101Web-ProdStage

Editing: OmniDocs101Web-ProdStage [Delete] [Cancel] [Save]

Edit: Source [Edit stage]

Source AWS CodeCommit	AWS-ECR-Registry Amazon ECR
--------------------------	--------------------------------

+ Add stage

Edit: Approval [Edit stage]

Approval-for-PROD Manual approval	Approval-for-PROD-2 Manual approval
--------------------------------------	--

Figure 4.66

NOTE:

As soon as you integrate the AWS ECR into the pipeline, it adds to the existing CloudWatch event rule that acts as a deployment trigger. Now, whenever you push the new Docker image with the same image tag name that is defined in the Amazon ECR action in the source stage, it triggers the pipeline. But as per the PROD deployment specification, PROD deployments are approval based and are available on-demand. So, you need to disable the continuous deployment for the PROD pipeline.

Perform the below steps to Remove the target from the existing CloudWatch event rule created against AWS ECR action:

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Go to the **Rules** given under **Events** in the navigation pane.
3. Search the rule created against the Amazon ECR with the same image tag name that is defined in the source stage.

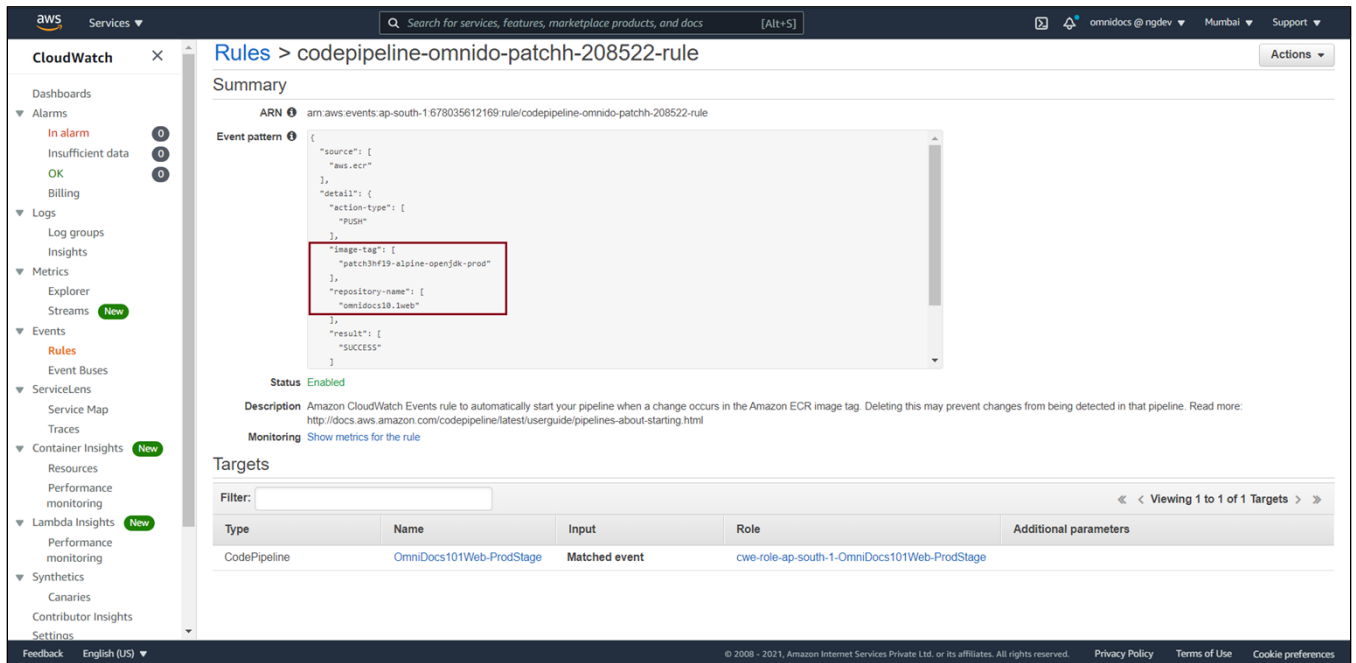


Figure 4.67

4. Select the rule, go to the **Actions** menu, and select **Disable** or **Delete**.

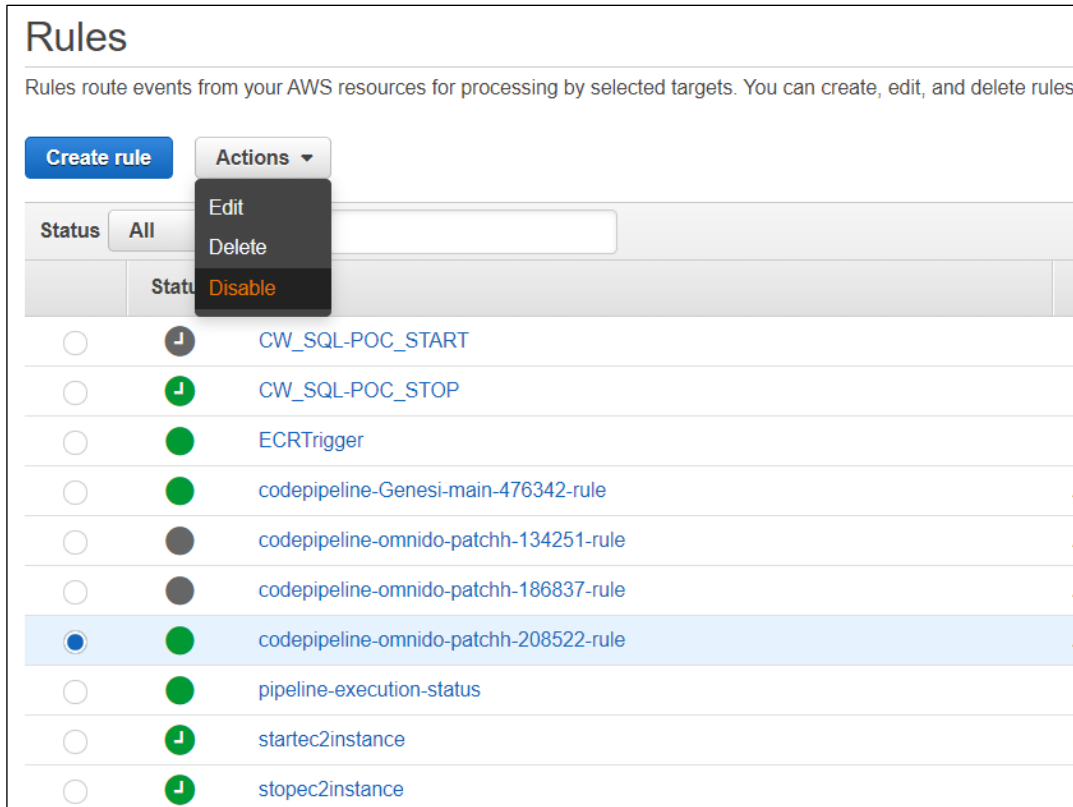


Figure 4.68

Now, it does not trigger the pipeline whenever you push the Docker image to the AWS ECR.

NOTE:

As discussed in the **Add source stage**, the AWS CodeCommit action creates a new CloudWatch event rule, and it triggers the AWS CodePipeline whenever there is a change in the CodeCommit repository. You can disable the CloudWatch event rule once the pipeline is created.

Perform the below steps to disable the CloudWatch event rule created against AWS CodeCommit action:

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Go to the **Rules** under **Events** in the navigation pane.
3. Search the rule created against the AWS CodeCommit repository **Genesis-CodeCommit-Repository**.

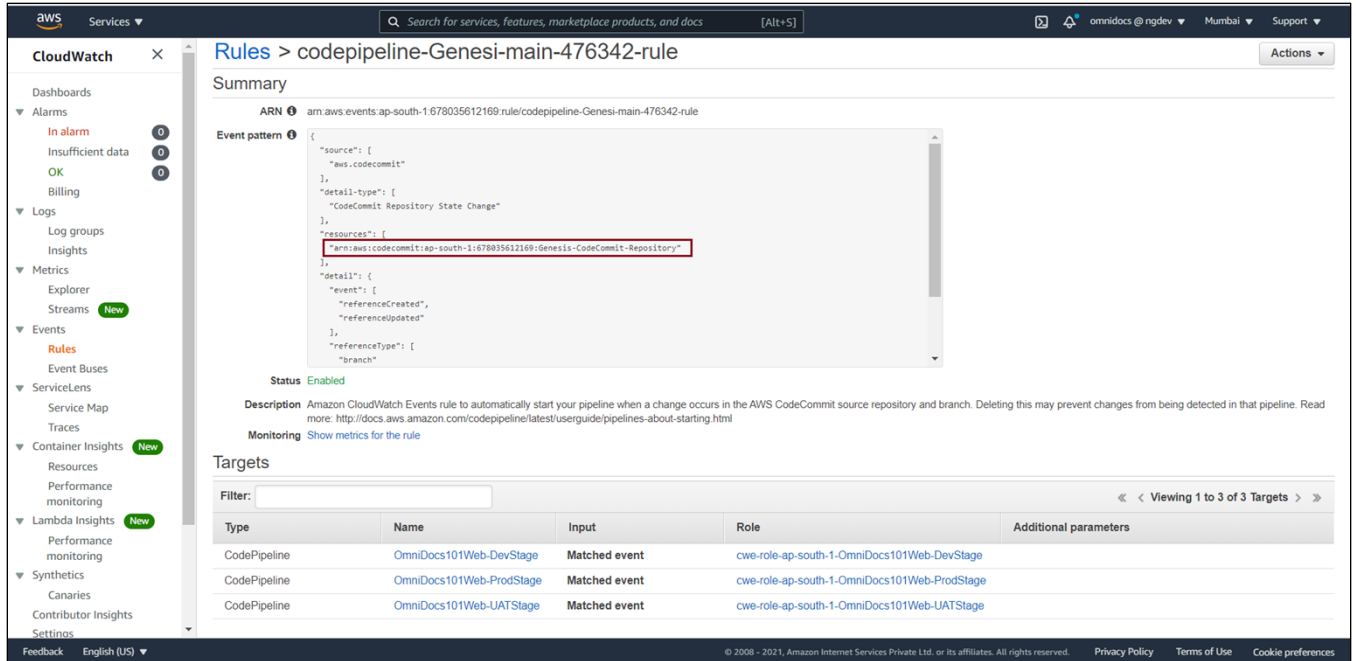


Figure 4.69

4. Select the rule, go to the **Actions** menu, and select **Disable** or **Delete**.

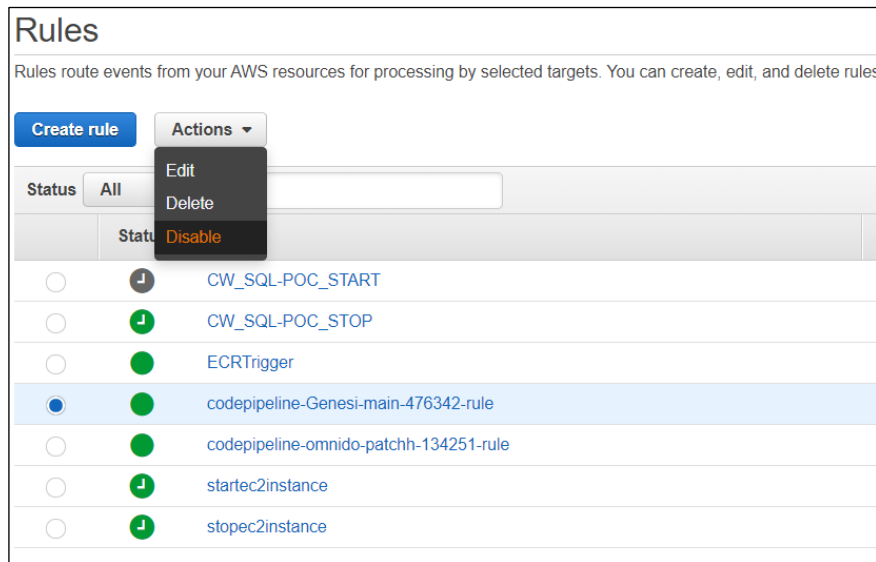


Figure 4.70

Now, it does not trigger the pipeline whenever you make any change in the AWS CodeCommit repository.

NOTE:

As per the production deployment specification, even after taking approvals from all stakeholders, the deployment to the production environment is not triggered automatically. A manual intervention mail is sent to the engineer who is supposed to deploy to production with a checklist.

Configuration of Manual Intervention Mail:

To configure the manual intervention mail, follow the below steps:

- **Create an SES identity**
- **Create a Lambda function**
- **Add a stage to the PROD pipeline**

Create an SES identity:

In AWS SES, an identity is an email address or domain that you use to send an email. Before sending/receiving an email using AWS SES, you must verify each identity that you are going to use as a sender or recipient. In other words, you can say that a verified identity is an email address or domain that you have proven that you own.

To create a verified SES identity, follow the below steps:

1. Sign in to the Amazon SES console <https://console.aws.amazon.com/ses/home>
2. In the console, use the region selector to select the **AWS Region** where you want to verify the email address.
3. Select the Verified identities under Configuration in the navigation pane.
4. Select the **Create identity**.
5. On the **Create identity** page, select Identity type as **Email Address**.
6. Specify the email address that you want to use as sender or recipient.
7. Click **Create identity**.

Create identity

A *verified identity* is a domain, subdomain, or email address you use to send email through Amazon SES. Identity verification at the domain level extends to all email addresses under one verified domain identity.

Identity details [Info](#)

Identity type

Domain
 To verify ownership of a domain, you must have access to its DNS settings to add the necessary records.

Email address
 To verify ownership of an email address, you must have access to its inbox to open the verification email.

Email address

Email address can contain up to 320 characters, including plus signs (+), equals signs (=) and underscores (_).

Assign a default configuration set
 Enabling this option ensures that the assigned configuration set is applied to messages sent from this identity by default whenever a configuration set isn't specified at the time of sending.

Tags - optional [Info](#)

You can add one or more tags to help manage and organize your resources, including identities.

No tags associated with the resource.

You can add 50 more tags.

Figure 4.71

8. Select the inbox for the email address for verification and receives a message with the following subject line: **Amazon Web Services - Email Address Verification Request in region **RegionName****, where **RegionName** is the name of the AWS Region.
9. Click the **link** in the message. A confirmation message appears **You have successfully verified an email address**.

Create a Lambda function

1. Open the function page on the lambda console:
<https://console.aws.amazon.com/lambda/home>
2. Select **Create function**.
3. In the **Function name**, specify the unique function name such as **lambda-fuction-ses**.
4. In the **Runtime**, select **python 3.8** or the latest version. Keep the other settings as default.
5. Select **Create function**.

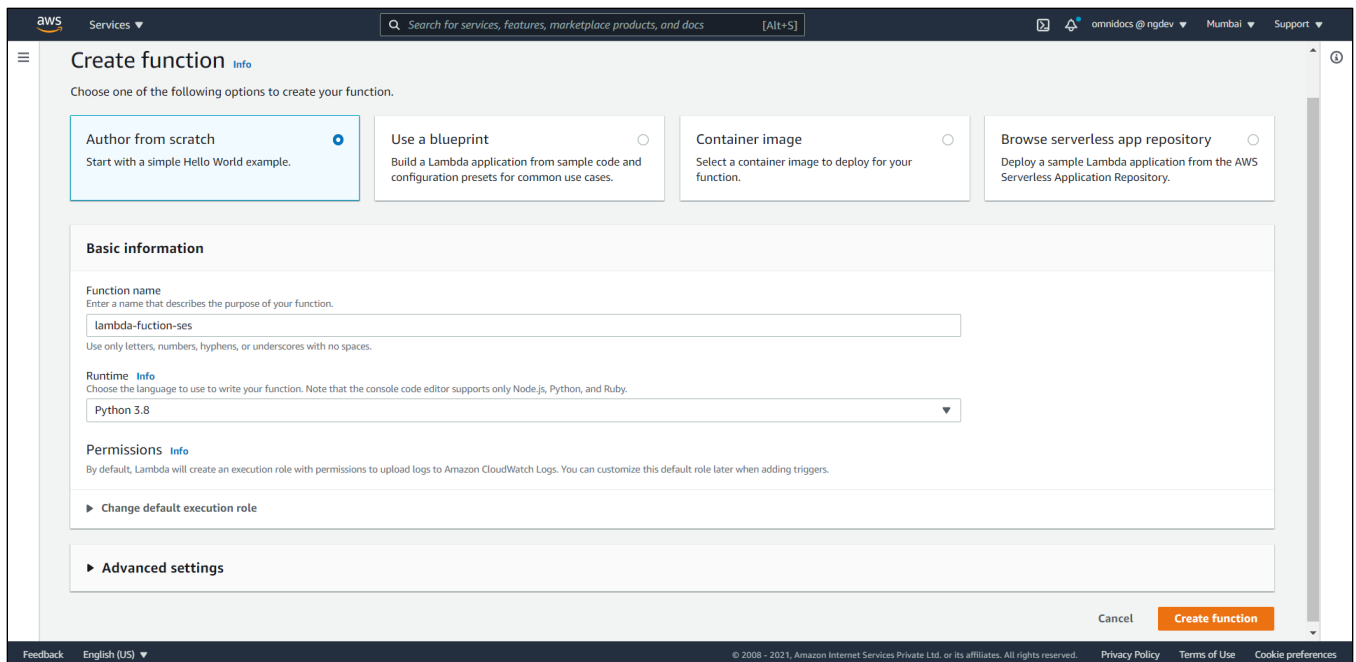


Figure 4.72

6. Under the **Code** tab, select **lambda_function.py**.
7. Replace the default code snippet using the below code snippet, and select **Deploy**:

```
import boto3
from botocore.exceptions import ClientError

# Create a new Codepipeline event to set the Job status
code_pipeline = boto3.client('codepipeline')

def lambda_handler(event, context):
    #Getting JobID of the pipeline
    JobId=event['CodePipeline.job']['id']

    execution_id=event['CodePipeline.job']['data']['actionConfiguration']['configuration']['UserParameters']

    # Update vivek_kumar@abc.co.in with your "From" address. This address must
    # be verified with Amazon SES.
    SENDER = "DevOps Admin <vivek_kumar@abc.co.in>"

    #Update ToRecipients, CcRecipients, and BccRecipients addresses. If your
    # account is still in the sandbox, this address must be verified.
    ToRecipients = ["vivek_kumar@abc.co.in","vivekkumarpandey185@gmail.com"]
    #CcRecipients = ["vivek_kumar@abc.co.in"]
    #BccRecipients = ["vivek_kumar@abc.co.in"]

    # If necessary, replace ap-south-1 with the AWS Region you're using for
    # Amazon SES.
    AWS_REGION = "ap-south-1"

    # The subject line for the email.
```

```

SUBJECT = "Manual Intervention Mail for the "+execution_id

# The HTML body of the email.
BODY_HTML = ""<html>
<head></head>
<body>
<pre><p style="color:#1F4E79">Dear Recipient,
Before deploying to the production, make sure that the below checklist points
are completed:
  • All the Major and Catastrophic bugs must be fixed.
  • The latest images must be thoroughly tested on the Dev and UAT stages.
  • Approval has been taken from all stakeholders.
  • Deployment downtime has been taken from the client.
Regards:</BR>DevOps Admin
</p></pre>
</body>
</html>""

# The character encoding for the email.
CHARSET = "UTF-8"

# Create a new SES resource and specify a region.
client = boto3.client('ses',region_name=AWS_REGION)

# Try to send the email.
try:
    #Provide the contents of the email.
    response = client.send_email(
        Destination={'ToAddresses':ToRecipients},

#Destination={'ToAddresses':ToRecipients,'CcAddresses':CcRecipients},

#Destination={'ToAddresses':ToRecipients,'CcAddresses':CcRecipients,'BccAddresses':BccRecipients},
        Message={
            'Body': {
                'Html': {
                    'Charset': CHARSET,
                    'Data': BODY_HTML,
                },
            },
            'Subject': {
                'Charset': CHARSET,
                'Data': SUBJECT,
            },
        },
        Source=SENDER,
    )

# Display an error if something goes wrong.
except ClientError as e:
    print("Email is not sent!"),
    print(e.response['Error']['Message']),
    put_job_failure(JobId, 'Unable to send mail')
else:
    print("Email sent! with below Message ID:"),
    print(response['MessageId']),

```

```

        put_job_success(JobId, "Mail sent successfully")

def put_job_success(job, message):
    print('Putting job success')
    print(message)
    code_pipeline.put_job_success_result(jobId=job)

def put_job_failure(job, message):
    print('Putting job failure')
    print(message)
    code_pipeline.put_job_failure_result(jobId=job, failureDetails={'message':
message, 'type': 'JobFailed'})

```

8. In the above lambda function, you can update the following:

- **SENDER:** You can update your email ID in the From address. This address must be verified with Amazon SES.
- **ToRecipients:** Specify the multiple ToAddresses separated by a comma (,). These email addresses must be verified.
- **CcRecipients and BccRecipients:** By-default, CcRecipients and BccRecipients are commented. If you want to use these then you just uncomment at two places.
- **AWS_REGION:** If required, replace ap-south-1 with the AWS Region you are using for Amazon SES.
- **SUBJECT:** Update the email subject line as per your requirement.
- **BODY_HTML:** Update the mail body in HTML form.

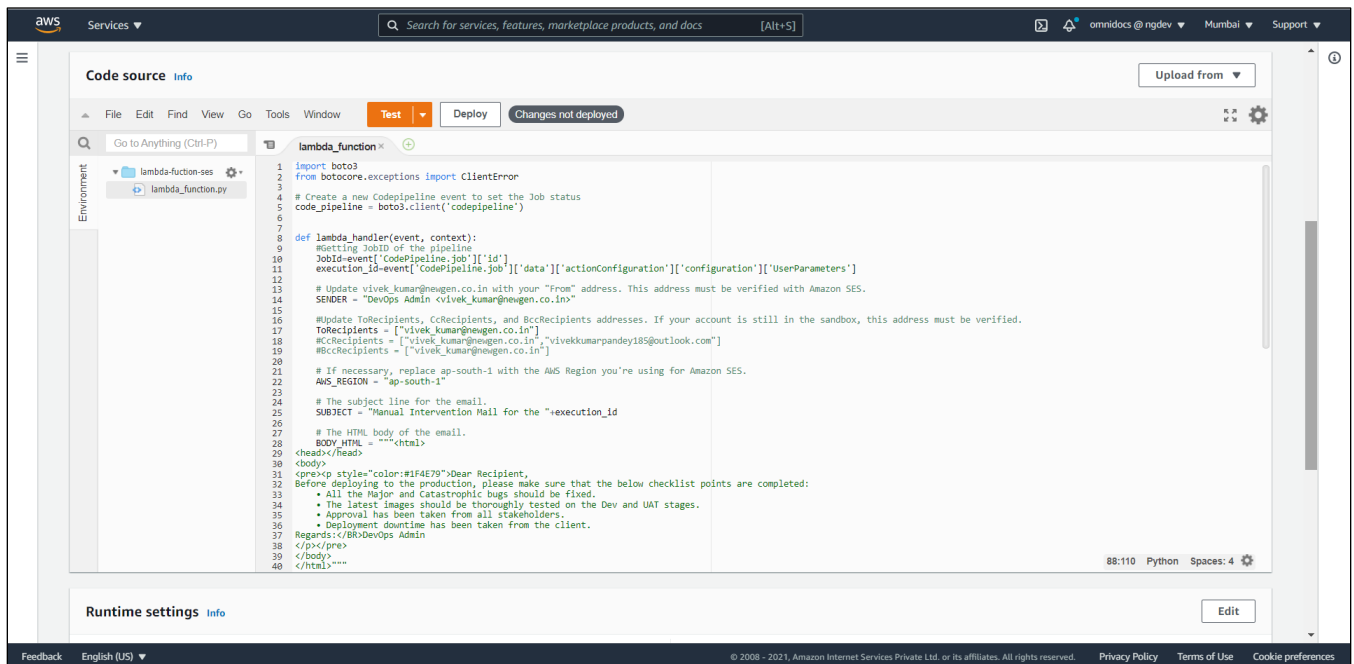


Figure 4.73

9. Go to the **Permissions** under the **Configuration** tab.
10. Select the created IAM role for this lambda function. The IAM role Summary screen appears.

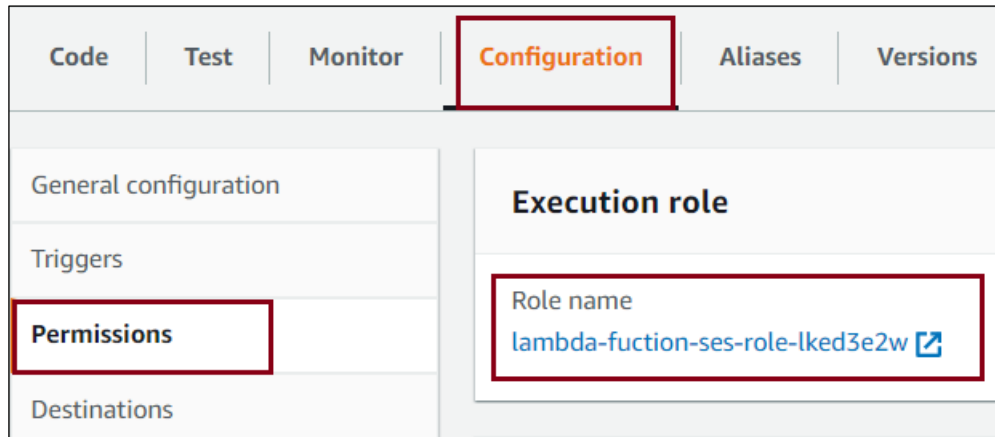


Figure 4.74

11. Select **Add inline policy**.
12. Select SES in the **Service**.
13. Select **SendEmail** and **SendRawEmail** in the **Actions**.
14. Select **All resources** in the **Resources**.
15. Click **Review policy**.

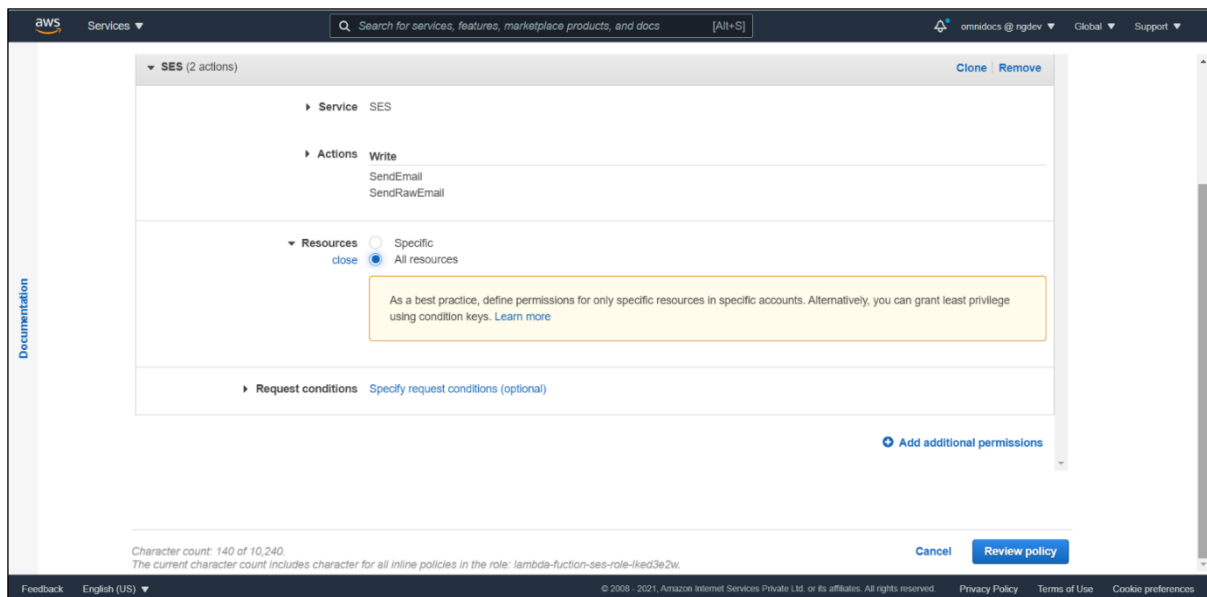


Figure 4.75

16. Specify the policy name such as **ses-lambda-policy**.
17. Select the **Create policy**. Add another inline policy by selecting **Add inline policy**.

18. Select the **CodePipeline** in the **Service**.

19. Select the **PutJobSuccessResult** and **PutJobFailureResult** in the **Actions**.

By default, the above actions support all resources so there is no need to select.

20. Select **Review policy**.

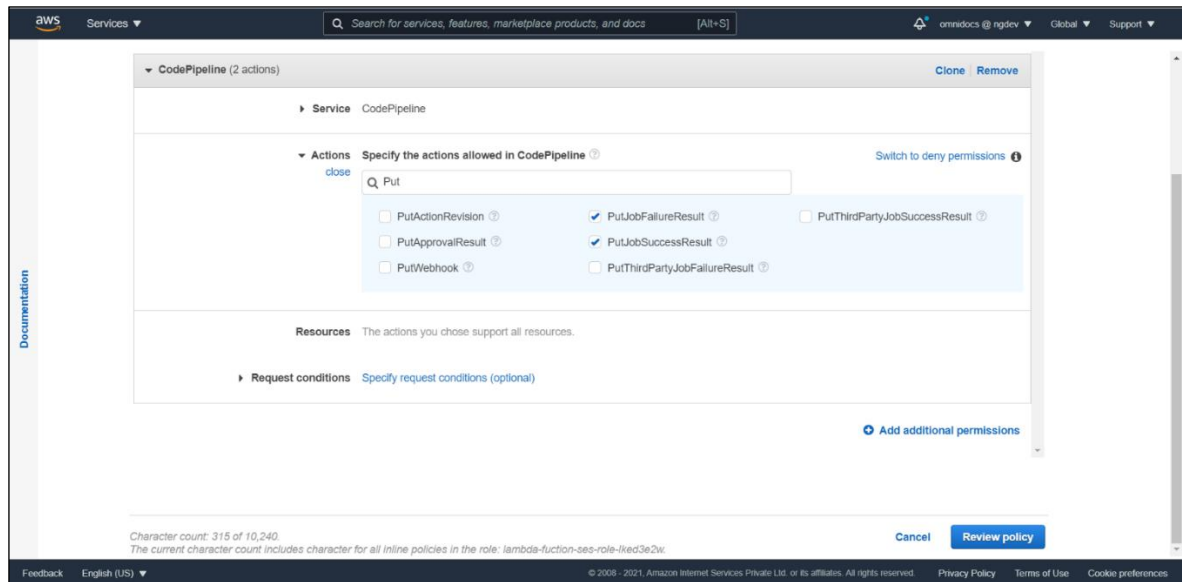


Figure 4.76

21. Specify the policy name such as **codepipelie-lambda-policy** and select **Create policy**.

Add a stage to the PROD pipeline:

NOTE:

As per the production deployment specification, even after taking approvals from all stakeholders, the deployment to the production environment is not triggered automatically. A manual intervention mail is sent to the engineer who is supposed to deploy to production with a checklist. If all the checklist points are covered or not, then the deployment to the production gets rejected. To handle this use case, you need to add a stage just after the Approval stage and add 2 actions: Firstly, execute the Lambda function, and secondly, Manual approval action without the SNS topic.

To add a stage to the PROD pipeline, follow the below steps:

1. Open the created pipeline **OmniDocs101Web-ProdStage** in **Edit** mode.
2. Select **+ Add stage** and specify the stage name such as **Manual-Intervention**.

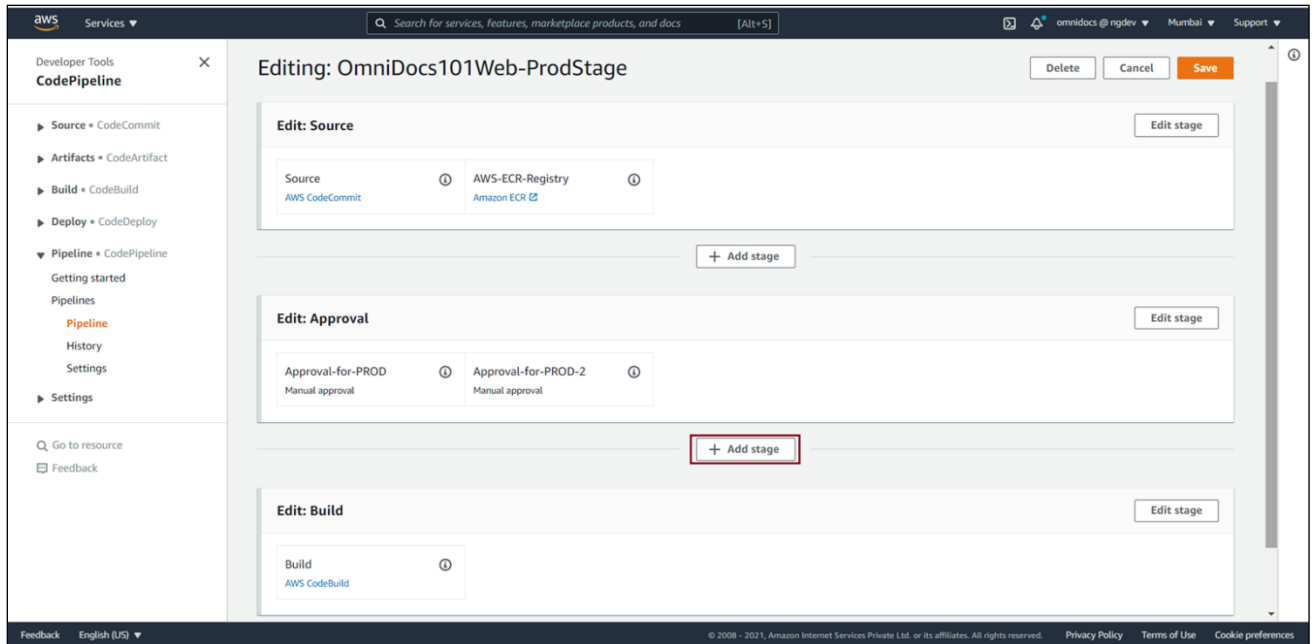


Figure 4.77

3. Specify the following in the **Manual-Intervention** stage:
 - i. Click **+Add action group** under the **Manual-Intervention** stage.



Figure 4.78

- ii. Specify the action name such as **Execute-Lambda** in the **Action** name.
- iii. Select **AWS Lambda** in the **Action provider**.
- iv. Select the AWS region where the Lambda function is created in the Region.
- v. Select lambda function lambda-function-ses in the Function name.
- vi. In the User parameters – optional, specify the parameters as given below:

Pipeline "<Name of the Pipeline>" with Execution Id
"#{codepipeline.PipelineExecutionId}"

For example,
 Pipeline "OmniDocs101Web-ProdStage" with Execution Id
"#{codepipeline.PipelineExecutionId}"
- vii. Keep the other settings as default.
- viii. Click **Done**.

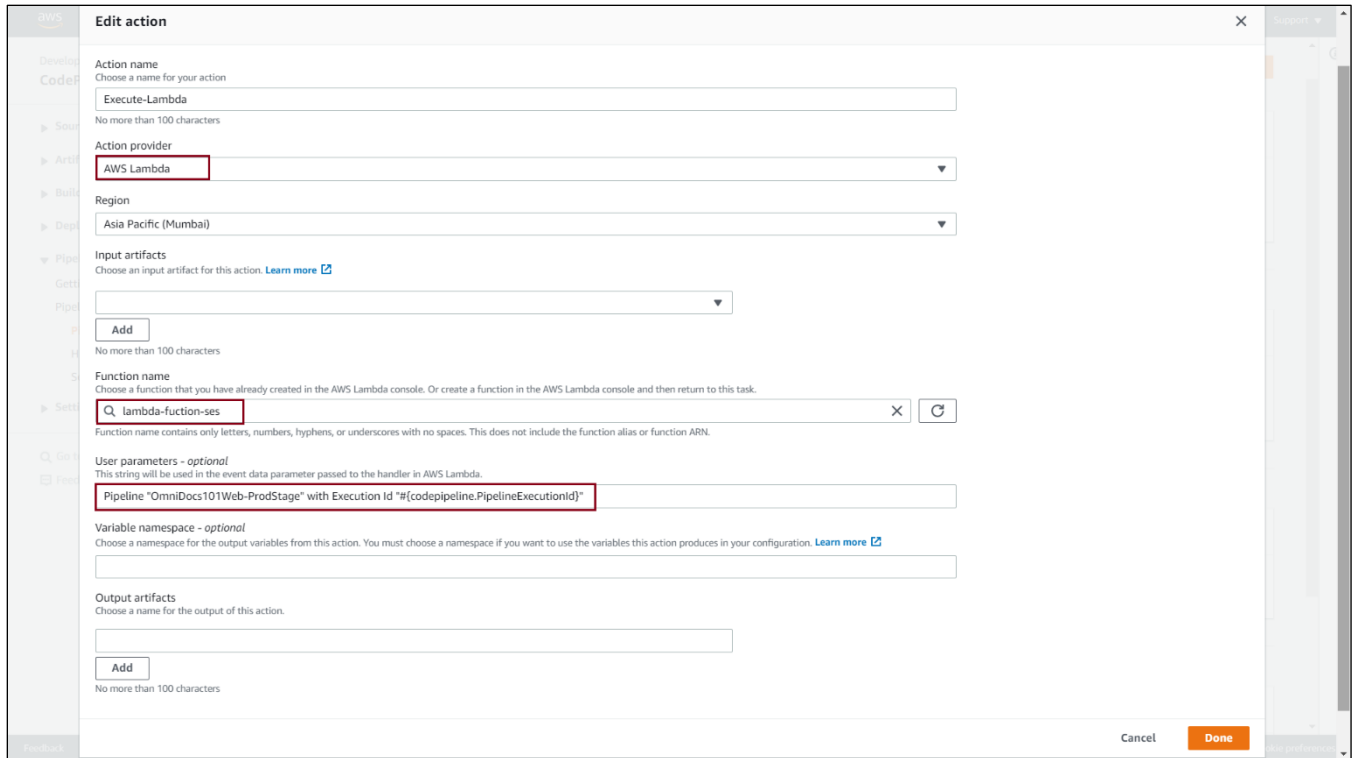


Figure 4.79

4. To add another action for the Manual approval, click **+Add action** under the **Manual-Intervention** stage and specify the following:



Figure 4.80

- i. Specify the **Action name** that is, as **Manual-Approval**.
- ii. Select **Manual approval** in the **Action provider**.
- iii. For **Comments – optional**, specify the comment to display for the reviewer in email notifications or the console.

Ensure that all the checklist points shared over the mail are completed for the Pipeline " OmniDocs101Web-ProdStage " with Execution Id "#{codepipeline.PipelineExecutionId}". Where **OmniDocs101Web-ProdStage** is the name of the pipeline.

- iv. Keep the other settings as default and click **Done**.

Edit action

Action name
Choose a name for your action
Manual-Approval
No more than 100 characters

Action provider
Manual approval

Configure the approval request.

SNS topic ARN - optional
Q

URL for review - optional
Type the URL you want to provide to the reviewer as part of the approval request. The URL must begin with 'http://' or 'https://'.

Comments - optional
Comments you type here display for the reviewer in email notifications or the console.
Please make sure all the checklist points shared over the mail are completed for the Pipeline " OmniDocs101Web-ProdStage" with Execution Id "#{codepipeline.PipelineE

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Cancel Done

Figure 4.81

- v. Click **Done** on the **Manual-Intervention** stage.
- vi. Click **Save** in the upper-right to save the pipeline.

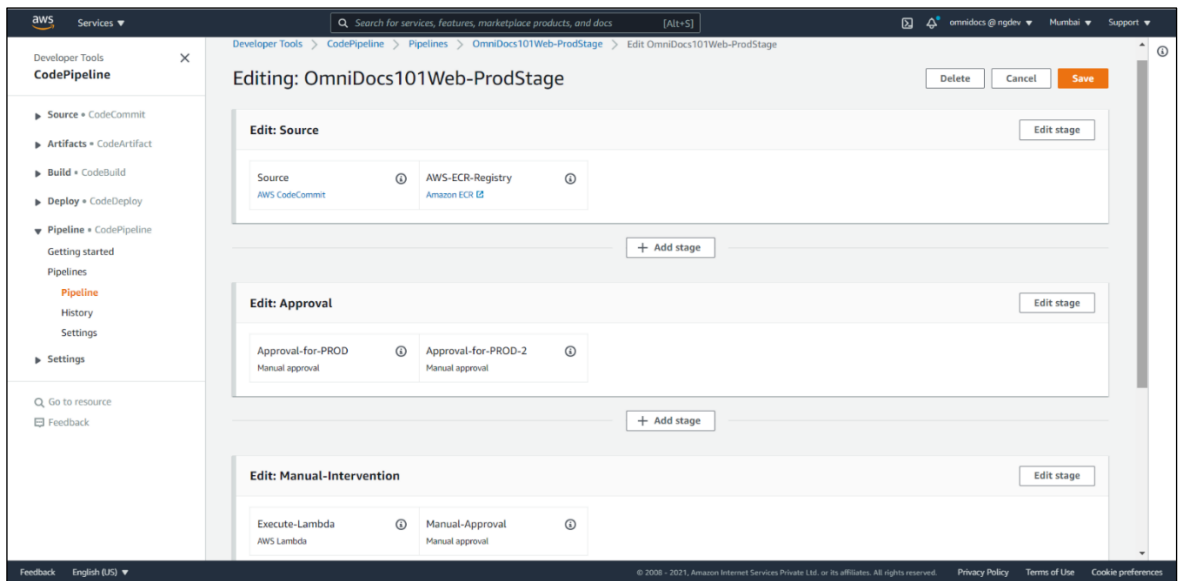
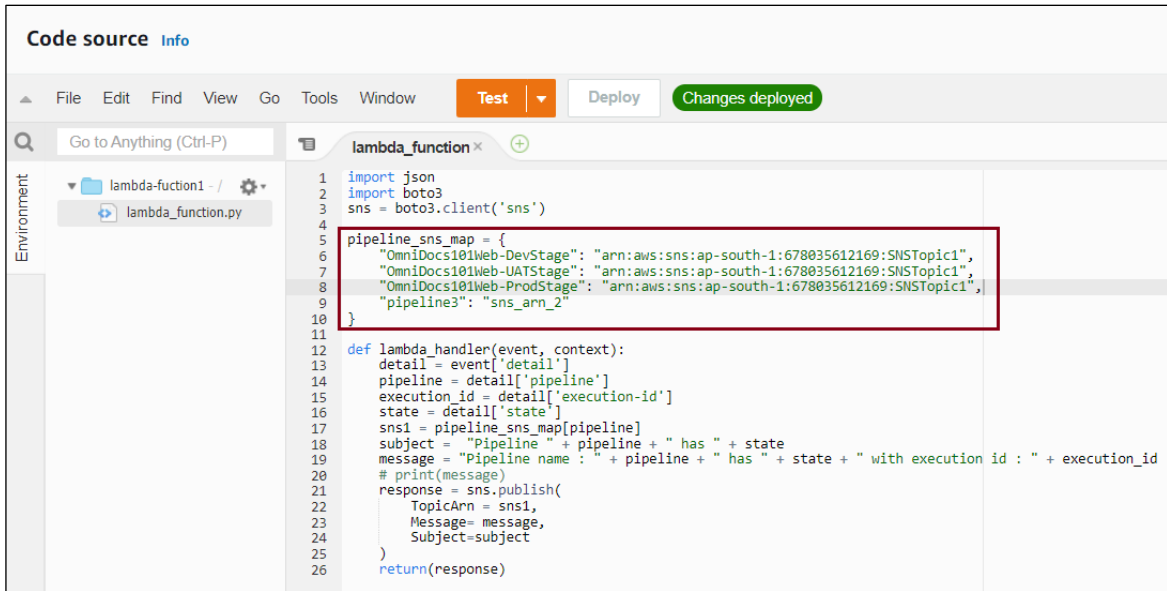


Figure 4.82

NOTE:

To add an entry in the Lambda function `lambda_function1` created in **Creation of AWS CodeCommit Repository** for each newly created pipeline and its associated SNS topic that you can use. This is required to notify the recipient(s) about the pipeline execution status whether it is succeeded or failed.

5. Add an entry of the pipeline 'OmniDocs101Web-ProdStage' in the lambda function `lambda_function1`.



```
1 import json
2 import boto3
3 sns = boto3.client('sns')
4
5
6 pipeline_sns_map = {
7     "OmniDocs101Web-DevStage": "arn:aws:sns:ap-south-1:678035612169:SNSTopic1",
8     "OmniDocs101Web-UATStage": "arn:aws:sns:ap-south-1:678035612169:SNSTopic1",
9     "OmniDocs101Web-ProdStage": "arn:aws:sns:ap-south-1:678035612169:SNSTopic1",
10 }
11
12 def lambda_handler(event, context):
13     detail = event['detail']
14     pipeline = detail['pipeline']
15     execution_id = detail['execution-id']
16     state = detail['state']
17     sns1 = pipeline_sns_map[pipeline]
18     subject = "Pipeline " + pipeline + " has " + state
19     message = "Pipeline name : " + pipeline + " has " + state + " with execution id : " + execution_id
20     # print(message)
21     response = sns.publish(
22         TopicArn = sns1,
23         Message= message,
24         Subject=subject,
25     )
26     return(response)
```

Figure 4.83

6. Trigger the pipeline manually by clicking on **Release change**.

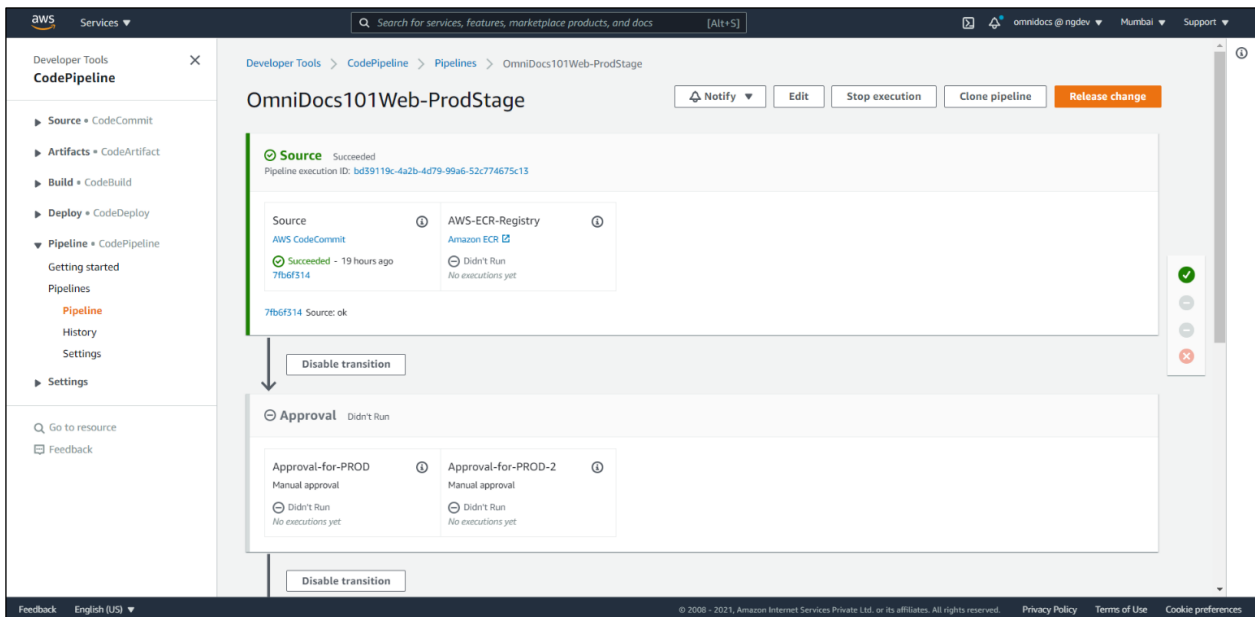


Figure 4.84

Appendix

This guide contains third-party product information about configuring Amazon Web Services (AWS) CodePipeline for Container Deployment on EKS and AWS Kubernetes Cluster. Newgen Software Technologies Ltd does not claim any ownership on such third-party content. This information is shared in this guide only for convenience of our users and could be an excerpt from the AWS documentation. For latest information on configuring the AWS Kubernetes Cluster and AWS CodePipeline refer to the AWS documentation.