MarkLogic Server

MarkLogic Server on Amazon Web Services (AWS) Guide

MarkLogic 10
May, 2019

Last Revised: 10.0-7, June, 2021

Table of Contents

MarkLogic Server on Amazon Web Services (AWS) Guide

1.0 Overview of MarkLogic Server on AWS

This chapter provides an overview of MarkLogic Server on Amazon Web Services (AWS) using a MarkLogic Amazon Machine Image (AMI), as well as how to create an AWS account and order a MarkLogic Server for AWS AMI. This chapter includes the following sections:

For more detailed information on AWS, see the Amazon documentation located at the following URL:

http://aws.amazon.com/documentation/

1.1 STOP: Before you do Anything!

There are multiple ways to launch a MarkLogic AMI to create a MarkLogic cluster or a single MarkLogic instance in the AWS environment. However, before you explore any alternatives, it is recommended that you first launch your MarkLogic AMI using a CloudFormation template and follow the procedures described in this guide. For details on how to launch a MarkLogic AMI using a CloudFormation template, see “Deploying MarkLogic on EC2 Using CloudFormation” on page 41. The MarkLogic CloudFormation templates are available from http://developer.marklogic.com/products/cloud/aws.

> **Note:** MarkLogic now supports the 1-Click Launch option in AWS Marketplace. Because of this, the published MarkLogic AMIs will have data volume predefined.

Should you later choose not to launch your MarkLogic AMI by means of a CloudFormation template, you will not have automatic access to the Managed Cluster features described in “The Managed Cluster Feature” on page 8. You can still launch an AMI, but you will need to follow the steps outlined in “Launching a MarkLogic AMI outside of CloudFormation” on page 10.

1.2 Understanding MarkLogic Server for AWS

MarkLogic provides pre-packaged AMIs containing Amazon Linux/Linux 2 and MarkLogic Server. MarkLogic has included scripts on these AMIs that simplify the steps necessary to get your MarkLogic Server instances up and running.

This section describes:

- Launching a MarkLogic AMI outside of CloudFormation

1.2.1 AWS Terminology

The following are the definitions for the terms used in this guide:

Amazon Web Services (AWS) is the Amazon Cloud Computing service. For details, see http://aws.amazon.com/.

Elastic Compute Cloud (EC2) is an AWS service that enables you to launch and manage server instances in Amazon's data centers using APIs or available tools and utilities. The AWS EC2 website is available at: http://aws.amazon.com/ec2/.

Virtual Private Cloud (VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network defined by you. For details, see https://aws.amazon.com/vpc/.

CentOS is a community-supported, free and open source operating system based on Red Hat Enterprise Linux.

A *Load Balancer* serves as the single point of contact for between your client and the cloud, distributing incoming application traffic across multiple targets, in multiple Availability Zones.

Availability zones are regions, physical locations around the world where AWS data centers are located.

An *Elastic Load Balancer* (ELB) is a service that automatically distributes and balances application traffic among multiple EC2 instances. For details, see http://docs.aws.amazon.com/gettingstarted/latest/wah/getting-started-create-lb.html.

Elastic Network Interface (ENI) is a virtual network interface that you can attach to an instance in a VPC. Network interfaces are available only for instances running in a VPC. For details, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html.

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume; there is no charge when your code is not running. For details, see https://aws.amazon.com/lambda/.

Amazon Machine Image (AMI) is an encrypted machine image that contains all information necessary to boot instances of software. Instances of MarkLogic Server are created from the stock Amazon Linux and Linux 2 AMI and have been pre-installed with MarkLogic and the necessary dependencies.

Type	Pricing
Enterprise	Per-hour AWS premium charged
Bring Your Own License (BYOL)	No additional charge

Elastic Block Store (EBS) is a type of storage designed specifically for Amazon EC2 instances. Amazon EBS allows you to create volumes that can be mounted as devices by Amazon EC2 instances. Amazon EBS volumes behave like raw unformatted external block devices. They are attached to user-specified block devices and provide a block device interface. You can load a file system on top of Amazon EBS volumes, or use them just as you would use a block device. Amazon EBS volumes exist separately from the actual instances and persist until you delete them. This allows you to store your data without leaving an Amazon EC2 instance running. Each Amazon EBS volume can be up to 16 TiB in size.

An *Instance* is the running system after an AMI is launched. Instances remain running unless they fail or are terminated. When this happens, the data on the instance is no longer available. Once launched, an instance looks very much like a traditional host.

An *Instance Type* defines the size of an Amazon EC2 instance. The MarkLogic Server instance types are shown in the table at the end of Step 5 in "Creating a CloudFormation Stack using the AWS Console" on page 47.

An *Instance Store* (sometimes referred to as *Ephemeral Storage*) is a fixed amount of storage space for an instance. An instance store is not designed to be a permanent storage solution. If an instance reboots, either intentionally or unintentionally, the data on the instance store will survive. If the underlying drive fails or the instance is terminated, the data will be lost.

AWS Cloud Storage (S3) is an Amazon web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. For details, see "Configuring MarkLogic for Amazon Simple Storage Service (S3)" on page 91 and https://aws.amazon.com/s3/.

Cloud Formation (CF) is the AWS Cloud Formation service for provisioning startup of AWS resources. For details, see "Deploying MarkLogic on EC2 Using CloudFormation" on page 41 and http://aws.amazon.com/cloudformation/. The MarkLogic CloudFormation templates are available from http://developer.marklogic.com/products/cloud/aws.

A *Classic Load Balancer* (CLB) runs at the application layer (layer seven) and the transport layer (layer 4) in the Open Systems Interconnection (OSI) model. The CloudFormation template will create a CLB if you deploy to one zone.

Application Load Balancer (ALB) is type of load balancer used in Elastic Load Balancing. It runs at the application layer (layer seven in the Open Systems Interconnection (OSI) model). The load balancer receives a request, evaluates the listener rules in priority order, determines which rule to apply, and selects a target from the target group.

> **Note:** With an ALB, you will be unable to use an ODBC connection for use with business intelligence (BI) tools. To use an ODBC connection with BI tools, you can create a separate Network Load Balancer for ODBC connections

Auto Scaling Groups (ASG) are used with a load balancer and target groups, enabiling you to scale out instances, or scale back in and de-register instances. For more information, see https://docs.aws.amazon.com/autoscaling/ec2/userguide/attach-load-balancer-asg.html in the Amazon EC2 Auto Scaling User Guide.

Target groups route requests to one or more registered targets using specified protocol and port numbers. A target may be registered with multiple target groups.

Listeners check for connection requests from clients, using the configured protocol and port. The rules defined for a listener determine how the load balancer routes requests to its registered targets.

Key Management Service (KMS) is a an AWS service that provides secure location, known as a keystore, where the encryption keys used to encrypt data are stored. This AWS KMS is not to be confused with the internal MarkLogic KMS described in Overview of MarkLogic Server on AWS in the *Security Guide*. The AWS KMS can be used as an external KMS. To access AWS KMS, MarkLogic must be configured to use AWS Credentials, as described in “Configure AWS Credentials” on page 93. A MarkLogic cluster inside a VPC (with or without a public DNS) will work with AWS KMS. For details on AWS KMS, see https://aws.amazon.com/kms/.

Managed Clusters is a MarkLogic feature that works with AWS features to automatically create and provision the necessary AWS resources and provide MarkLogic with the information needed to manage your cluster. For details, see “The Managed Cluster Feature” on page 8.

MarketPlace is the AWS service for publishing pay-per-use and free (no extra charge) public AMI's on amazon. For details, see https://aws.amazon.com/marketplace.

An *EC2 Compute Unit (ECU)* provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

Metadata Database is the database that stores and indexes all of the configuration data required to manage a cluster of one or more MarkLogic Servers. For AWS, the DynamoDB service is used to implement the Metadata Database. For details, see http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStartedDynamoDB.html.

1.2.2 The Managed Cluster Feature

Running MarkLogic Server in AWS has some challenges that you may not experience in traditional IT data centers. The Managed Clusters feature helps you mitigate these challenges with support for reliability, scalability and high availability, as well as with tools that automatically handle some of the more problematic issues. It is highly recommended that you use the CloudFormation templates and follow the procedures in this guide to launch your MarkLogic AMI in AWS as they are provided to help you leverage AWS and MarkLogic features especially designed for a reliable and easy cloud deployment.

> **Note:** The Managed Cluster feature now supports SSL-enabled clusters. For details, see Configuring SSL on App Servers in the *Security Guide*.

On AWS, the following should be considered before deploying MarkLogic nodes or clusters:

- Instance Hostnames — An EC2 instance starts life with a unique hostname. If this instance is stopped for any reason (such as a hardware/software failure or manual stopping to avoid EC2 charges), when it restarts it will start with a new hostname. This causes configuration problems, especially in clusters, but also in single-node installations. If you stop a cluster then bring it back online without reconfiguring it, the nodes will not rejoin the cluster because all the hostnames are different.

  The Managed Cluster feature automatically detects hostname changes and propagates the changes throughout the cluster.

- Transient Data — When an EC2 instance is terminated, the root volume is released and all of the data on it is lost. This includes any pre-installed software or OS configurations. MarkLogic installations should not be put on the root volume, but rather on an attached EBS volume.

  The Managed Cluster feature automatically keeps track of each EBS volume, along with its related EC2 instance and mount directory. When you restart your EC2 instances, the Managed Cluster feature automatically re-attached and mounts your volumes to the appropriate locations to ensure that your Forests and Databases are intact.

- AWS Security — If you want to access any AWS services from within an EC2 node you need authentication. One way to do this is to provide your AWS credentials and store them on the EC2 instance. However, this is both inconvenient and a security risk. The Managed Cluster feature and CloudFormation templates use IAM Roles so that you never need to expose your AWS Credentials.
- Securing your Infrastructure — EC2 instances by default are open to the world. You need to consider your security requirements before launching an EC2 instance and putting

sensitive data on it. The Cloud Formation Templates create baseline security groups to allow easy startup. You can edit these security groups to your particular security needs.

- Volatile Instances — EC2 instances can fail at any time. This is true of on-premise hardware as well, but in the cloud you are not there to fix this yourself. Instead you need to rely on AWS features to restart failed instances to keep your database up and running at all times. The CloudFormation templates create a suggested topology with AutoScaling groups across zones as needed that distribute your cluster and restart unhealthy instances automatically. The Managed Cluster support makes sure that restarted instances rejoin the cluster.
- Datacenter Failures — Datacenters can and do fail. Often do to unpredictable issues, like hurricanes or earthquakes. If you want high availability you need to plan for failure by distributing your cluster across datacenters, so that any single failure does not take your cluster offline. The CloudFormation templates create a suggested topology with AutoScaling groups distributed across zones in a single region for maximum fault tolerance.
- Load Balancing and Routing — Your MarkLogic cluster should only be accessible when it is healthy. Use of AWS ELBs is highly recommended even for single nodes, but especially when running a cluster. The ELBs not only balance traffic across healthy nodes, but automatically notify the AutoScaling groups when a node is unhealthy, even if the hardware and OS are running fine, so that traffic is diverted to healthy nodes and the unhealthy nodes are terminated and restarted.

  The Managed Cluster feature provides a Health Check application on each server that is used by the Load Balancer to detect if your MarkLogic instance is healthy.

  Note: If you are using a XCC client (such as mlcp) with MarkLogic running on AWS, you must enable the `xcc.httpcompliant` setting to work with AWS ELBs. For details, see Using a Load Balancer or Proxy Server with an XCC Application in the *XCC Developer's Guide*.

- Scaling Up and Down — Sometimes you want more capacity sometimes you want less. Cloud Computing provides the raw tools to enable scaling up and down, but your software needs to understand and integrate with provisioning changes. The CloudFormation templates allow an easy one-step process for scaling your cluster capacity up and down.
- License Application on a Cluster — When launching a cluster, you traditionally need to apply your license to each node manually after they come up for the first time. The Managed Cluster feature automatically applies your license keys to all of the nodes in the cluster.
- Cluster Formation — When running a MarkLogic cluster, you need to configure the first node with your Administrator credentials, then configure remaining nodes to connect to the cluster. The Managed Cluster feature automatically handles this for you so clusters (even clusters of one) come up ready to run without further manual intervention.

- Hibernating Clusters — A great benefit of Cloud Computing is that you only pay for what you use. If you are running a development server, or a site that doesn't need to be running 24/7, you can hibernate your entire cluster so you don't incur EC2 charges while it is not running. The Managed Cluster feature along with CloudFormation enable you to quickly restart the cluster and have your resources re-attach to all of your data, so that your cluster will be up and running where left off.

CloudFormation is not required to make use of the Managed Clusters feature, instead you can choose to manually or programmatically configure the AWS resources using other tools, but it is a challenging task without strong cloud orchestration and management tools. CloudFormation allows you to both document and implement a managed cluster configuration using a simple declarative template that can grow with your needs.

Running MarkLogic without the Managed Clusters feature is also supported (with or without our provided AMI's) and is the simplest configuration. However it is also the least reliable and is not recommended.

1.2.3 Launching a MarkLogic AMI outside of CloudFormation

This section describes the minimum steps you need to take should you insist on running MarkLogic without either CloudFormation or your own or third party cloud management tools. These steps do not enable the Managed Cluster feature and are not recommended.

- Launch an EC2 AMI, either one we provide in MarketPlace or your own AMI on which you have pre-installed MarkLogic, following the OS-specific installation instructions.
- Create a Key Pair, as described in “Creating a Key Pair” on page 28. You will need the Key Pair name to launch the instance.
- Create a Security Group that opens at port 22 (for ssh) and ports (8000-8002) for the Admin API. You may also need to configure the Security Group to open additional ports for your own applications or for foreign clusters.
- Create an EBS volume for your data and attach it to the logical device, `/dev/sdf`. On startup, MarkLogic will detect this volume (or wait for it to be attached) then create a filesystem and mount it as `/var/opt/MarkLogic`. The MarkLogic AMI will have the data volume pre-defined and it must be associated with `/dev/sdf`.

When your instance is started, you can browse to the Admin UI on your host’s port 8001. Log in with user “admin”; the password is set to the value of the instance ID.

1.3 HealthCheck App Server

The Elastic Load Balancer (ELB) periodically sends a heartbeat to each of its instances to monitor their health. Each instance of MarkLogic Server has a HealthCheck app server on port 7997. The ELB cannot be configured with authentication, so the URL for the HealthCheck App Server does not require authentication.

1.4 Elastic Network Interface

The Elastic Network Interface separates the network interface from an EC2 instance so you can attach and detach the instance from MarkLogic nodes. The ENI is assigned a private IPv4 addresses, so that attaching and detaching the ENI won't affect the assigned IPv4 address. This alleviates the burden to manage the hostname change when instances fail.

AWS enforces a default limitation of ENIs per Region. However, AWS can increase the limit upon request. For details about the default limitation of ENIs per Region, see https://docs.aws.amazon.com/vpc/latest/userguide/amazon-vpc-limits.html#vpc-limits-enis.

The MarkLogic CloudFormation templates creates ENIs based on the total nodes in the cluster (by Managed ENI stack). When a node is launched in a group, the Node Manager will attach ENIs to the new node.

Note that the ENI created and attached to instances will be the secondary network interface of the instance. The Managed Cluster locates the secondary network interface and uses its private DNS name or private IP address as the hostname for the new MarkLogic node. Because the ENI's private DNS name and IP address are static, the hostname for a MarkLogic node is also static, even if an instance fails and a new instance kicks in to assume the role in the cluster.

If you chose to set up the cluster manually or with your own templates, it is recommended that you create ENIs. Otherwise your private IPv4 address will be released when the node is terminated.

1.5 Typical Architecture

This section describes some of the typical configurations of a MarkLogic cluster in an AWS environment.

As described in the *Scalability, Availability, and Failover Guide*, Evaluator Nodes (E-Nodes) perform data processing operations including aggregates, computations (including user defined functions). Data Nodes (D-Nodes) manage the forest data operations. E-Nodes can be grouped separately from D-Nodes in a security group, which might be preferable for some deployments.

End-user or app-level queries should be routed to the E-Nodes through a load balancer. You can add E-Nodes to scale up a cluster to handle more queries, more users and more computation.

To ensure high availability, place D-Nodes in different availability zones in the same region and configure them for local-disk failover to ensure each transaction is written to one or more replicas. Put configured D-Nodes in different zones from the masters, protecting against zone failure. In AWS, the latency between zones in the same region is low (approximately two milliseconds). For optimum availability, D-Nodes and E-Nodes should be split evenly between three availability zones. For disaster recovery, you can place D-Nodes in different regions and use database replication between the D-Nodes in each region, protecting against region failure. The MarkLogic cluster in the different region should be similarly configured for high availability across the availability zones of that region.

Warning Two clusters can work in the same region (with or without DR). As a best practice for setting up DR for a typical architecture, they should be in different regions. Only then can they serve the purpose of protection against a region failure.

Note: Use high availability to protect against zone failure, and use disaster recovery to protect against region failure.

The recommended storage resources are EBS volumes for forests and S3 for backups. All volumes should be formatted with 16K blocks. This is optimized for MarkLogic's sequential IO profile. Each instance of MarkLogic Server should be configured with a small number of forests (2-4) per volume. Volumes should have a minimum of 3,000 IOPS (gp3 volume of any size or gp2 volume of 1 TB or larger).

2.0 Getting Started with MarkLogic Server on AWS

This chapter describes how to launch a MarkLogic Server AMI and access the MarkLogic Server Admin interface. This chapter includes the following sections:

2.1 Security

Access to MarkLogic server is controlled by the mechanisms described in the *Security Guide*. Within the AWS environment, access to EC2 instances is controlled by three mechanisms:

- Key Pairs, as described in “Creating a Key Pair” on page 28.
- AWS Identity and Access Management (IAM), as described in “Creating an IAM Role” on page 16.
- Security Groups, which are created by the CloudFormation template described in “Deploying MarkLogic on EC2 Using CloudFormation” on page 41.

Note: Amazon periodically updates its security resources. Each time you create a new instance of MarkLogic Server, the latest security updates are applied to that instance. Your older instances are not automatically updated and must be manually updated in order to obtain uniform and up-to-date security across your cluster. You can optionally disable automatic security updates for new instances. For details on security updates, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonLinuxAMIBasics.html#security-updates.

2.2 Separate MarkLogic Converters

Starting with MarkLogic 9.0-4, the MarkLogic converters/filters are offered as a package (called MarkLogic Converters package) separate from the MarkLogic Server package. For EC2, the converter installer/package is located in your default user home directory. There is a `README.txt` file in the package describing what the package is for, and pointing to the MarkLogic documentation for more information. See MarkLogic Converters Installation Changes Starting at Release 9.0-4 in the *Installation Guide* for more details.

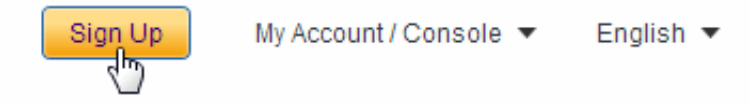
2.3 Summary of Deployment Procedures

The following is a summary of the procedures for deploying MarkLogic Server on EC2.

Procedure	For Details See
If you don't already have an AWS account, create one.	"Creating an AWS Account" on page 14
Enable a MarkLogic Server AMI.	"Enabling a MarkLogic Server for EC2 AMI" on page 15
Open the Amazon AWS Management Console.	"Accessing the AWS Management Console" on page 15
Create an IAM role.	"Creating an IAM Role" on page 16
If you don't already have a key pair, create one.	"Creating a Key Pair" on page 28
Create a Simple Notification Service (SNS) Topic.	"Creating a Simple Notification Service (SNS) Topic" on page 30
Create CloudFormation stack from a CloudFormation template.	"Deploying MarkLogic on EC2 Using CloudFormation" on page 41
Open the MarkLogic Server Admin interface.	"Accessing a MarkLogic Server Instance" on page 85

2.4 Creating an AWS Account

Before you can order a MarkLogic Server for EC2 AMI, you must set up an AWS account. To set up an AWS account, go to http://aws.amazon.com and click Sign Up for AWS:



Then follow the directions to create a new account. You will need to provide email and mail addresses, create a password, and provide credit card information.

2.5 Enabling a MarkLogic Server for EC2 AMI

You can use a MarkLogic-supplied AMI or build your own custom AMI using standard Amazon tools. This guide focuses on the MarkLogic-supplied AMIs that are available in AWS MarketPlace.

To enable your MarkLogic AMI, do the following:

- Go to https://aws.amazon.com/marketplace.
- Search for MarkLogic.
- In the MarkLogic product page, click the Accept Terms button.

Warning Unless, you plan to deploy your MarkLogic cluster manually, rather than use the recommended CloudFormation procedure, do not click on any of the Launch EC2 Instance buttons.

2.6 Initial Setup Procedures

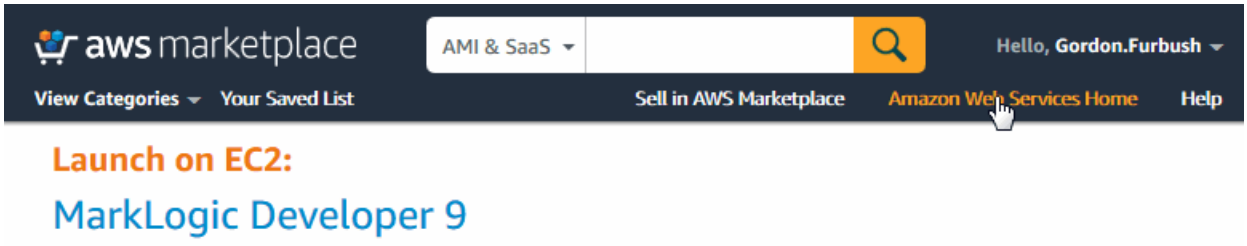
This section describes how to access the AWS management console and create a security group and key pair. Typically, you will create your security groups and key pairs once and reuse them for each instance you create. The topics in this section are:

- Accessing the AWS Management Console
- Creating an IAM Role
- Creating a Key Pair
- Creating a Simple Notification Service (SNS) Topic

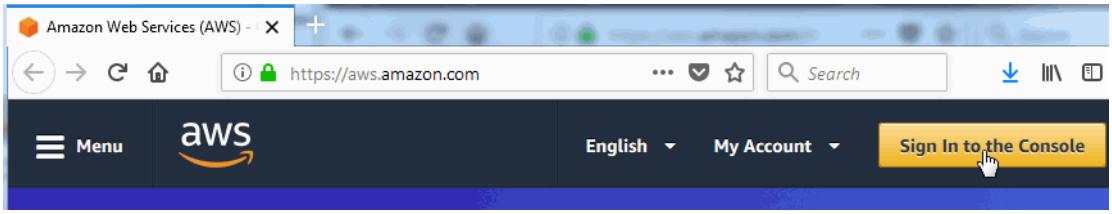
2.6.1 Accessing the AWS Management Console

This section describes how to access the Amazon AWS Management Console.

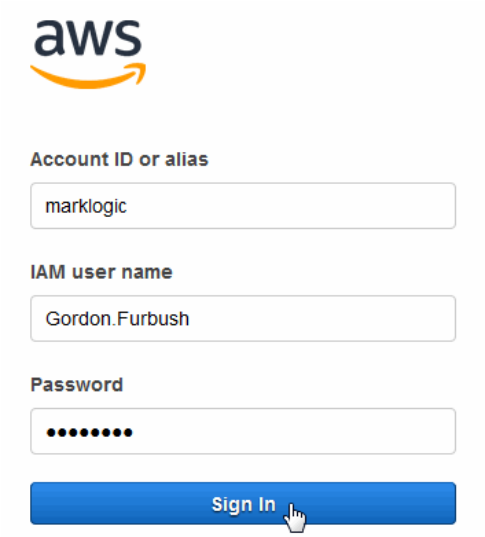
1. From the AWS Marketplace, click on `Amazon Web Services Home`.



2. In the Amazon Web Services Home page, click on `Sign In to the Console`.



3. Enter your AWS login credentials and click `Sign In`. The AWS Services page will appear.

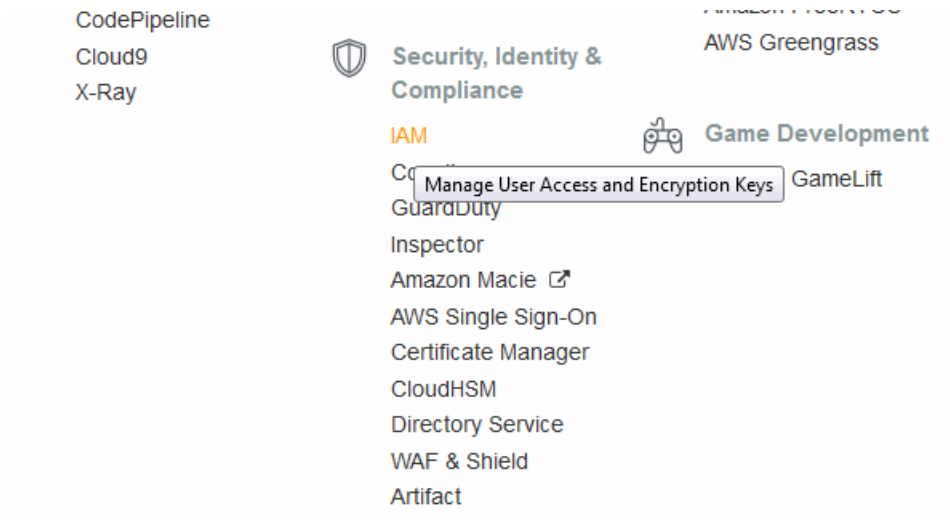


2.6.2 Creating an IAM Role

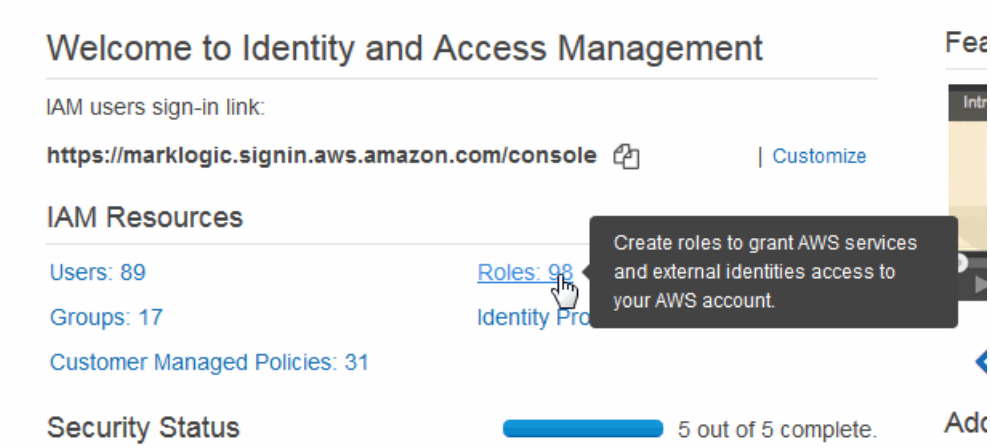
AWS Identity and Access Management (IAM) is a web service that enables you to manage users and user permissions in AWS. The service is targeted at organizations with multiple users or systems that use Amazon EC2, Amazon DynamoDB, and the AWS Management Console. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control user access to AWS resources.

This section describes how to create an IAM role. This section describes each step in the procedure, but does not discuss all of the options for each step. For more details, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html.

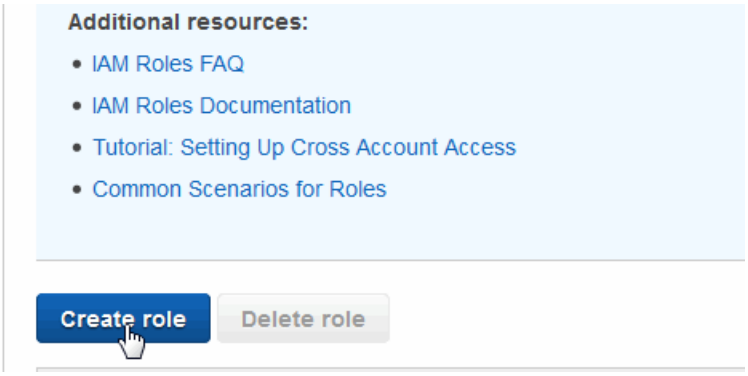
1. In the Security, Identity & Compliance section of the Amazon Web Services page, click on `IAM`:



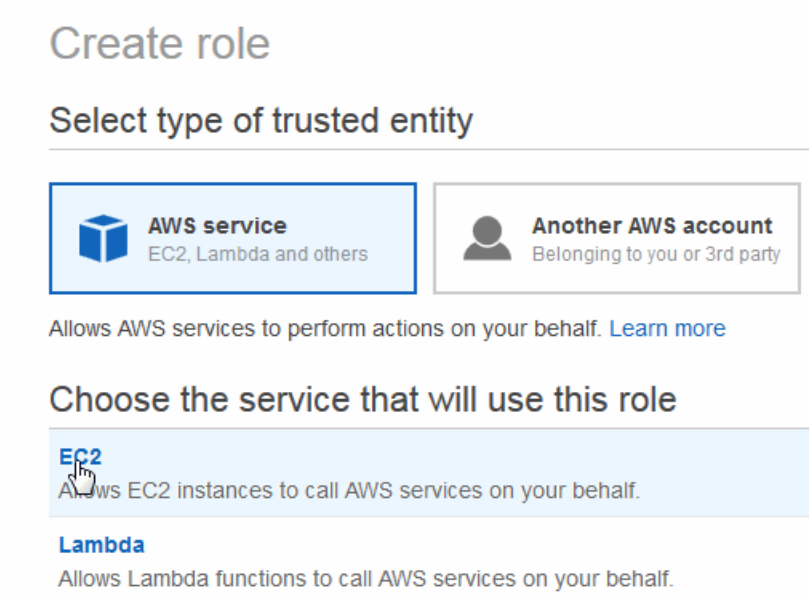
2. In the IAM Resources section of the Getting Started page, click Roles:



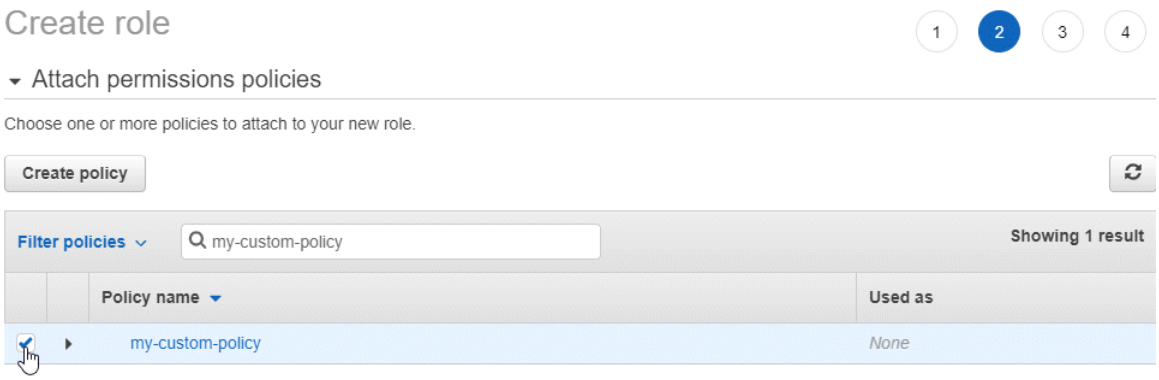
3. In the Roles page, click Create Role:



4. Select the AWS Service box and EC2. At the lower right hand portion of the page, click Next: Permissions.



5. In the Set Permissions window, select the access policy for the role. For details on IAM policies, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-policies-for-amazon-ec2.html.



The minimal privileges needed to launch a MarkLogic CloudFormation template, as tested, are as follows:

Note: The following set of permissions are the minimum required permissions to create and delete a MarkLogic CloudFormation stack. You will need additional permissions for S3 backups and KMS. The permissions below are quoted because they are in JSON format.

Note: MarkLogic recommends that you follow AWS best practices for controlling access to your AWS resources. For details, see https://docs.aws.amazon.com/IAM/latest/UserGuide/access_tags.html.

Amazon has changed the pattern for the ARN (https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html). The ARN now needs to specify the region and account details. This can be all regions and all accounts, which can be indicated using a wildcard "*"as in this example:

```
arn:aws:autoscaling:*:*:launchConfiguration::launchConfigurationName/*
```

Otherwise, the ARN should be specific to region, account, and the specific service, as in this example:

```
arn:aws:autoscaling:ap-southeast-
2:758929958593:autoScalingGroup:*:autoScalingGroupName/*
```

CloudFormation

- "cloudformation:CreateUploadBucket"
- "cloudformation:DeleteStackInstances"
- "cloudformation:ListExports"

- "cloudformation:ListStackSetOperations"
- "cloudformation:ListStackInstances"
- "cloudformation:DescribeStackResource"
- "cloudformation:CreateStackInstances"
- "cloudformation:ListStackSetOperationResults"
- "cloudformation:DescribeStackEvents"
- "cloudformation:DescribeStackSetOperation"
- "cloudformation:DescribeChangeSet"
- "cloudformation:ListStackResources"
- "cloudformation:ListStacks"
- "cloudformation:ListImports"
- "cloudformation:DescribeStackInstance"
- "cloudformation:DescribeStackResources"
- "cloudformation:GetTemplateSummary"
- "cloudformation:DescribeStacks"
- "cloudformation:GetStackPolicy"
- "cloudformation:DescribeStackSet"
- "cloudformation:ListStackSets"
- "cloudformation:CreateStack"
- "cloudformation:GetTemplate"
- "cloudformation:DeleteStack"
- "cloudformation:ValidateTemplate"
- "cloudformation:ListChangeSets"

- "Resource": "*"

DynamoDB

- "dynamodb:DeleteTable"
- "dynamodb:CreateTable"
- "dynamodb:DescribeTable"

- "Resource": "arn:aws:dynamodb:*:*:table/*MarkLogicDDBTable*"

EC2

- "ec2:DisassociateAddress"
- "ec2:DeleteSubnet"
- "ec2:ModifyVolumeAttribute"
- "ec2:DescribeAddresses"
- "ec2:CreateNatGateway"
- "ec2:CreateVpc"
- "ec2:AttachInternetGateway"
- "ec2:AssociateRouteTable"
- "ec2:DescribeInternetGateways"
- "ec2:DescribeAvailabilityZones"
- "ec2:CreateInternetGateway"
- "ec2:CreateSecurityGroup"
- "ec2:DescribeVolumes"
- "ec2:ModifyVpcAttribute"
- "ec2:DescribeRouteTables"
- "ec2:ReleaseAddress"
- "ec2:CreateRouteTable"
- "ec2:DetachInternetGateway"
- "ec2:DescribeNatGateways"
- "ec2:DisassociateRouteTable"
- "ec2:AllocateAddress"
- "ec2:DescribeSecurityGroups"
- "ec2:DescribeVpcs"
- "ec2:DeleteNatGateway"
- "ec2:DescribeVpcEndpoints"
- "ec2:DeleteVpc"
- "ec2:CreateSubnet"
- "ec2:DescribeSubnets"

- "Resource": "*"

- "ec2:RevokeSecurityGroupIngress"
- "ec2:DeleteRoute"
- "ec2:AuthorizeSecurityGroupIngress"
- "ec2:DeleteVpcEndpoints"
- "ec2:DeleteRouteTable"
- "ec2:CreateTags"
- "ec2:CreateVolume"
- "ec2:DeleteVolume"
- "ec2:DeleteInternetGateway"
- "ec2:DeleteSecurityGroup"
- "ec2:CreateRoute"
- "ec2:DeleteVpcEndpoints"
- "ec2:CreateVpcEndpoint"

- "Resources":
- "arn:aws:ec2:*:*:internet-gateway/*"
- "arn:aws:ec2:*:*:volume/*"
- "arn:aws:ec2:*:*:subnet/*"
- "arn:aws:ec2:*:*:route-table/*"
- "arn:aws:ec2:*:*:vpc-endpoint/*"
- "arn:aws:ec2:*:*:security-group/*"
- "arn:aws:ec2:*:*:vpc/*"
- "arn:aws:ec2:*:*:security-group/*"
- "arn:aws:ec2:*:*:route-table/*"
- "arn:aws:ec2:*:*:vpc-endpoint/*"
- "arn:aws:ec2:*:*:route-table/*"
- "arn:aws:ec2:*:*:security-group/*"
- "arn:aws:ec2:*:*:vpc-endpoint/*"
- "arn:aws:ec2:*:*:security-group/*"

- "arn:aws:ec2:*:*:route-table/*"

- "ec2:CreateTag"

- "Resource": "*"

ElasticLoadBalancing

- "elasticloadbalancing:DescribeLoadBalancers"

- "Resource": "*"

- "elasticloadbalancing:DeleteLoadBalancerPolicy"
- "elasticloadbalancing:DeleteLoadBalancer"
- "elasticloadbalancing:CreateLoadBalancer"
- "elasticloadbalancing:ModifyLoadBalancerAttributes"
- "elasticloadbalancing:CreateLoadBalancer"
- "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
- "elasticloadbalancing:CreateLoadBalancer"
- "elasticloadbalancing:CreateLoadBalancerPolicy"
- "elasticloadbalancing:ConfigureHealthCheck"
- "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
- "elasticloadbalancing:CreateLoadBalancerPolicy"
- "elasticloadbalancing:DeleteLoadBalancerPolicy"
- "elasticloadbalancing:ConfigureHealthCheck"
- "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
- "elasticloadbalancing:CreateLoadBalancer"
- "elasticloadbalancing:CreateLoadBalancer"
- "elasticloadbalancing:CreateLoadBalancer"
- "elasticloadbalancing:DeleteLoadBalancer"
- "elasticloadbalancing:CreateLoadBalancer"
- "elasticloadbalancing:ModifyLoadBalancerAttributes"

- "elasticloadbalancing:ConfigureHealthCheck"
- "elasticloadbalancing:CreateLoadBalancerPolicy"

- "Resources": "arn:aws:elasticloadbalancing:*:*:loadbalancer/*"

- "elasticloadbalancing:AddTags"
- "elasticloadbalancing:RemoveTags"

- "Resource": "*"

AutoScaling

- "autoscaling:DescribeLaunchConfigurations"
- "autoscaling:DescribeScalingActivities"
- "autoscaling:DescribeAutoScalingGroups"

- "Resource": "*"

- "autoscaling:CreateLaunchConfiguration"
- "autoscaling:DeleteLaunchConfiguration
- "autoscaling:DeleteAutoScalingGroup"
- "autoscaling:CreateAutoScalingGroup"
- "autoscaling:UpdateAutoScalingGroup"

- "Resources":
- "arn:aws:autoscaling:*:*:launchConfiguration::launchConfigurationName/*"
- "arn:aws:autoscaling:*:*:autoScalingGroup::autoScalingGroupName/*"

SNS

- "sns:ListSubscriptionsByTopic"
- "sns:Publish"
- "sns:GetTopicAttributes"

- "sns:DeleteTopic"
- "sns:CreateTopic"
- "sns:ConfirmSubscription"
- "sns:SetTopicAttributes"
- "sns:Subscribe"
- "sns:ListEndpointsByPlatformApplication"
- "sns:Unsubscribe"
- "sns:ListTopics"
- "sns:ListSubscriptions"
- "sns:ListPlatformApplications"

- "Resource": "*"

IAM

- "iam:GetRole"
- "iam:PassRole"
- "iam:DeleteRolePolicy"
- "iam:CreateRole"
- "iam:DeleteRole"
- "iam:PutRolePolicy"

- "Resource": "*"

Lambda

- "lambda:CreateFunction"
- "lambda:AddPermission"
- "lambda:InvokeFunction"
- "lambda:GetFunctionConfiguration"
- "lambda:DeleteFunction"
- "lambda:RemovePermission"
- "lambda:PutFunctionConcurrency"

- "Resource": "arn:aws:lambda:*:*:function:*"

S3

- "s3:PutObject"
- "s3:GetObjectAcl"
- "s3:GetObject"
- "s3:CreateBucket"
- "s3:GetObjectTagging"
- "s3:GetBucketAcl"
- "s3:GetBucketPolicy"

- "Resource": "*"

- "s3:PutBucketTagging"

- "Resource": "*"

Note: The following set of permissions are needed in a role that MarkLogic CloudFormation stack passes as an instance profile role. The permissions below are quoted because they are in JSON format.

DynamoDB

- "dynamodb:PutItem"
- "dynamodb:DescribeTable"
- "dynamodb:GetItem"
- "dynamodb:Scan"
- "dynamodb:UpdateItem"

- "Resources": "arn:aws:dynamodb:*:*:table/*MarkLogicDDBTable*"

EC2

- "ec2:AttachVolume"

- "ec2:CreateVolume"

- "Resources":
- "arn:aws:ec2:*:*:volume/*"
- "arn:aws:ec2:*:*:instance/*"

- "ec2:DescribeInstances"

- "Resource": "*"

SSM

- "ssm:UpdateInstanceInformation"
- "ssm:ListInstanceAssociations"
- "ssm:ListAssociations"
- "ssm:PutInventory"
- "ssm:UpdateInstanceAssociationStatus"

- "Resource": "*"

EC2Messages

- "ec2messages:GetMessages"

- "Resource": "*"

SSMMessages

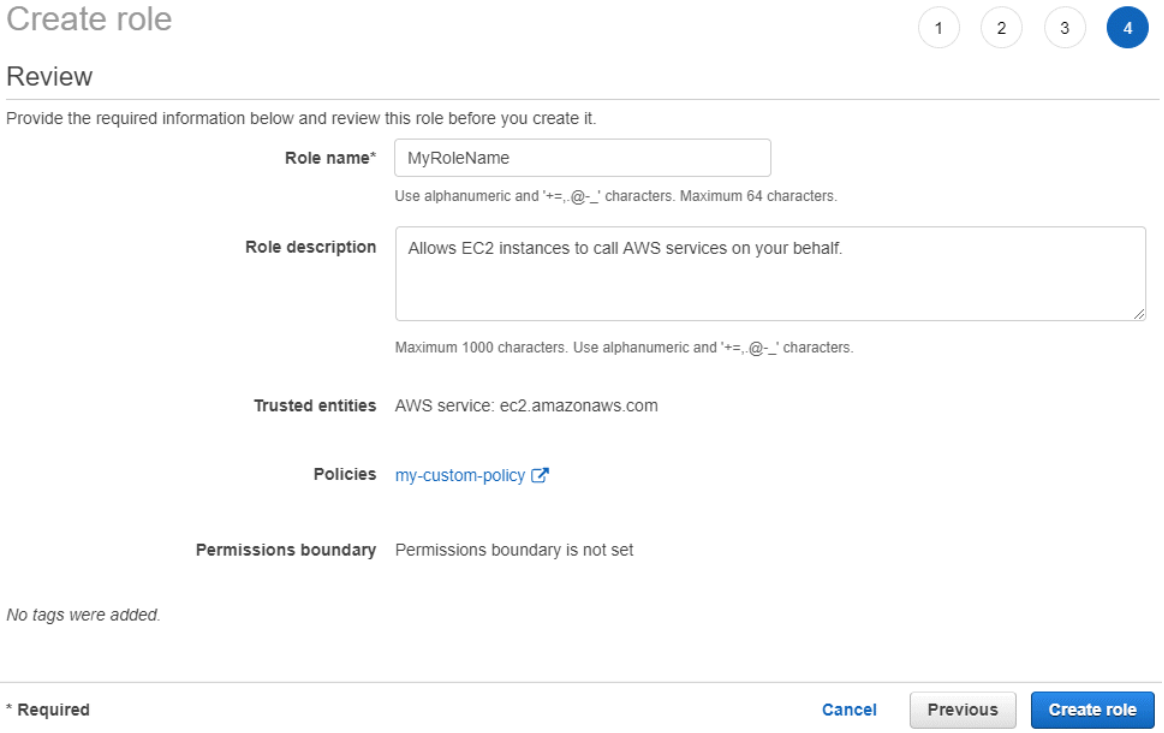
- "ssmmessages:OpenControlChannel"
- "ssmmessages:CreateControlChannel"

- "Resource": "*"

You may be able to use less privileges. For details on how to determine the least privileges to the IAM role, see https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.

6. At the lower right hand portion of the page, click Next: Tags. Enter any optional tag information for the IAM role, then click Next: Review.

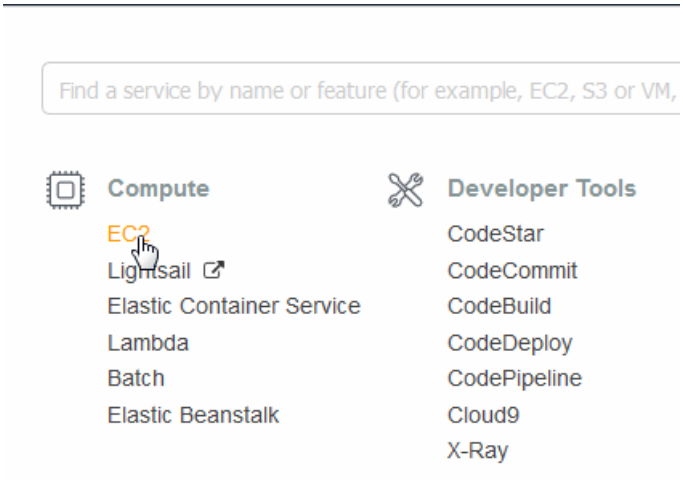
7. In the Review window, enter the name of the new role, review your settings and edit if you want to make changes. When done, click Create Role.



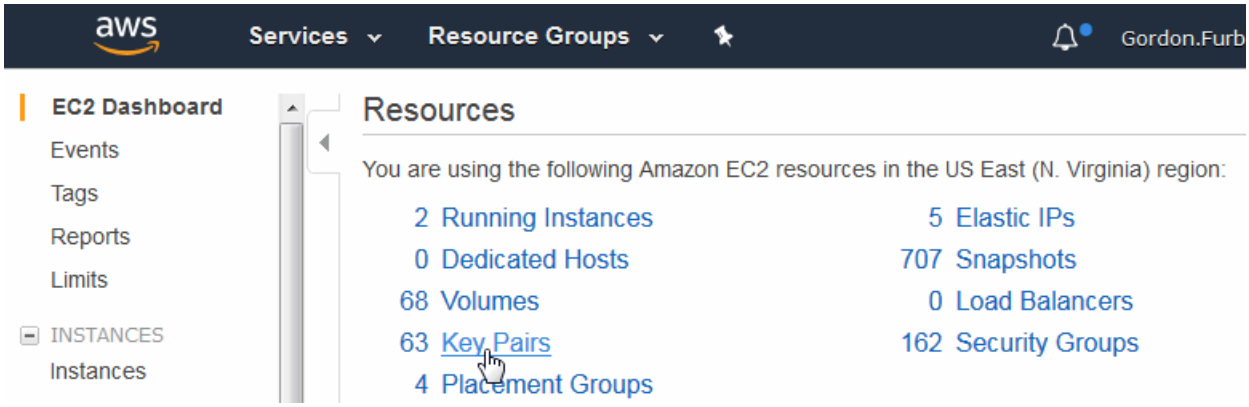
2.6.3 Creating a Key Pair

A key pair ensures that only you have access to your instances. You can create one or more Amazon EC2 key pairs. You can use a key pair to SSH to your instance.

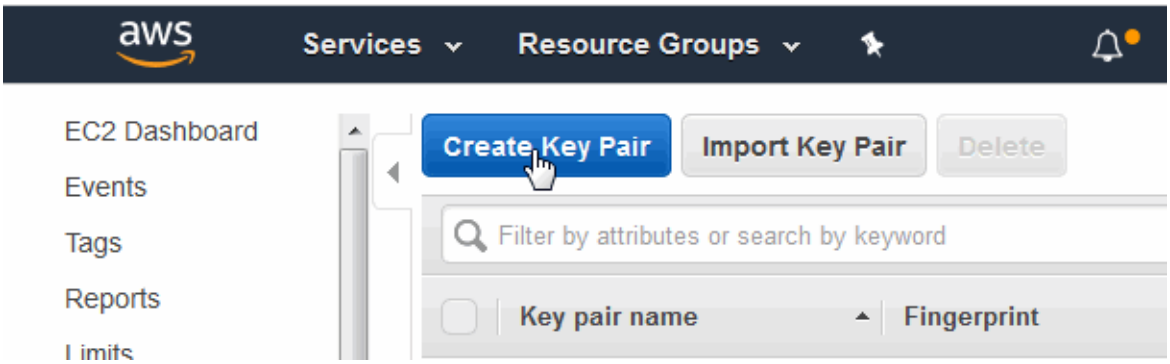
1. From the AWS Services page, select EC2 to open the EC2 Dashboard:



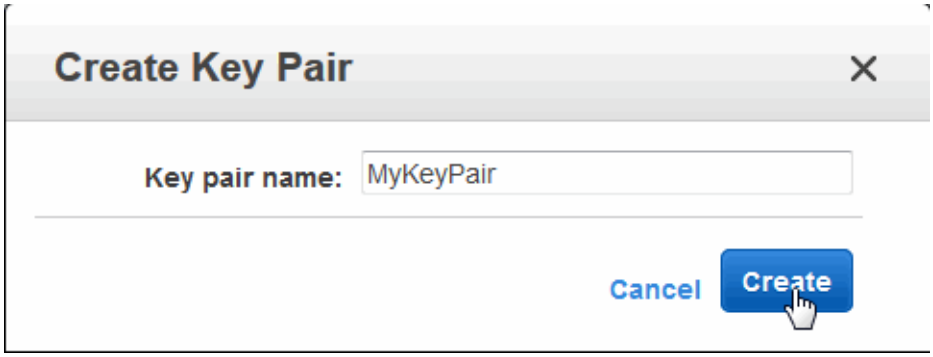
2. In the EC2 Dashboard, select Key Pairs:



3. In the Key Pairs page, click Create Key Pair:



4. Enter a name for your key pair and click Create:



5. Your key pair will be downloaded to your local system. When the download of the key pair completes, click Save File.

   Note: You will need to remember the location of the downloaded key pair on your local system should you need to create an SSH connection to your MarkLogic Server instance, as described in “Accessing an EC2 Instance” on page 86.

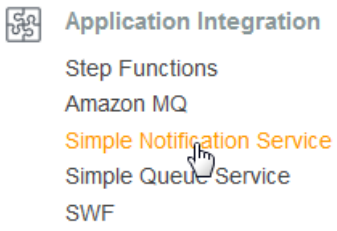
2.6.4 Creating a Simple Notification Service (SNS) Topic

The Amazon Simple Queue Service (SQS) is a queue system that enables you to queue messages generated by your EC2 Instances. In order to capture messages from your Instances, you must create a Simple Notification Service (SNS) Topic and specify it as part of your User Data in the CloudFormation Template.

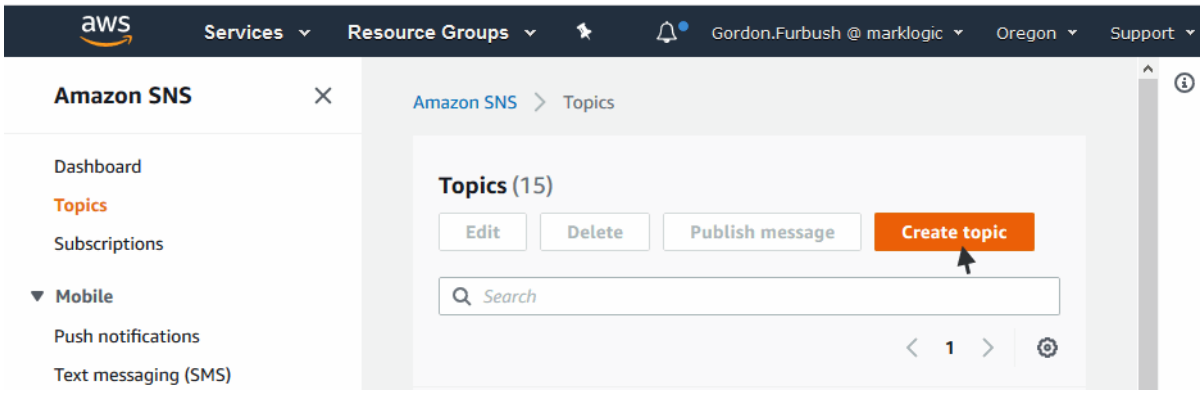
For details on the SQS queue system and creating an SNS topic, see http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqssubscribe.html.

There are a number of ways to create an SNS topic. One way is described below.

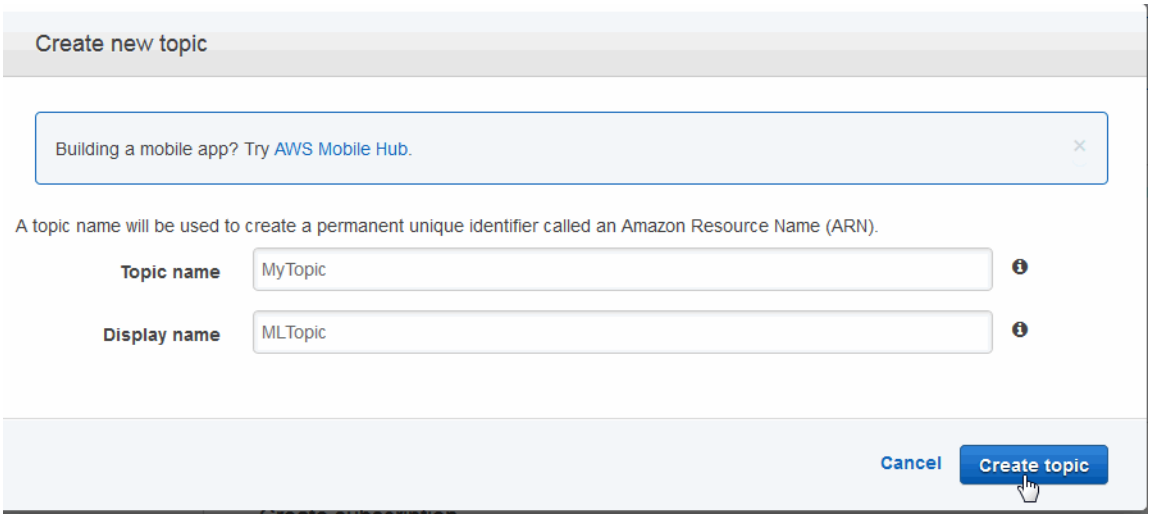
1. In the Services page, click on Simple Notification Service to open the SNS Dashboard.



2. Select Topics in the left menu and click Create Topic.

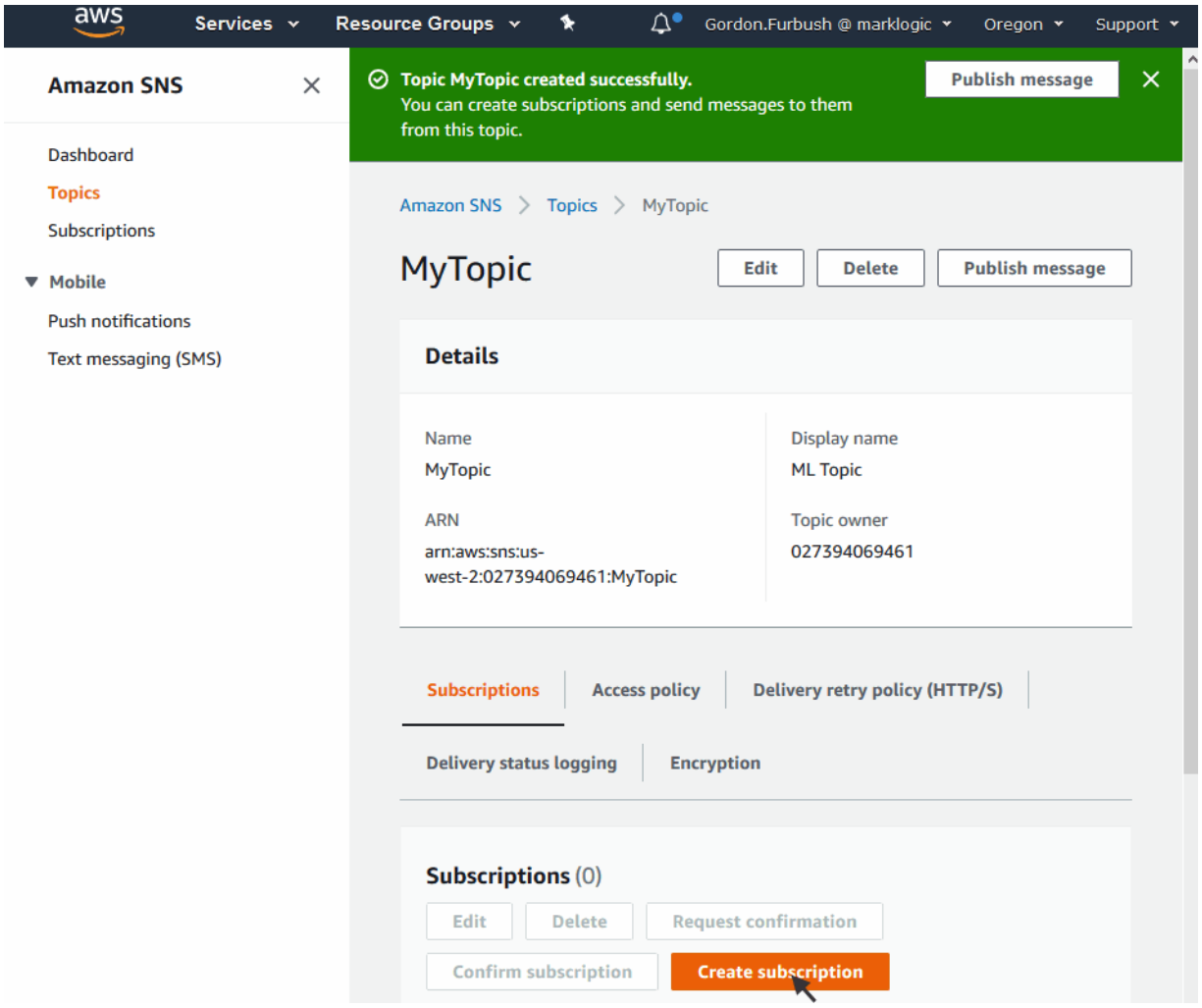


3. Enter a Topic Name and an optional Display Name. Click Create Topic.



4. In the Topic Details window, note the Topic ARN. This is what you will enter for the LogSNS field when you create your stack, as described in “Creating a CloudFormation Stack using the AWS Console” on page 47.

5. You must subscribe to an SNS Topic to view the messages. To subscribe to the topic, click Create Subscription in the Topic Details page. There are a number of ways to subscribe to an SNS Topic, as described in http://docs.aws.amazon.com/sns/latest/dg/welcome.html.



2.7 AWS Configuration Variables

On startup, MarkLogic is customizable by a set of environment variables. This applies to all configurations from single nodes managed externally to large distributed clusters using the full Cluster Management features.

These variables can be specified using any method that guarantees the values are present and consistent in the environment, regardless of what method is used to start the server and when the server is started. The variables related to Managed Cluster support also need to be configured properly on a per-instance basis. A simple and reliable method that allows reuse of the same AMI for all instances and doesn't require customizing the AMI itself is to pass the values as EC2 “User Data.” An alternative is to place the variable assignments in `/etc/marklogic.conf` either during the initial boot or built into a custom AMI dedicated for each equivalent node in the cluster.

When using CloudFormation, the `AWS::CloudFormation::Init` resource (and the helper `cfn-init` commands) are recommended for deployment and configuration. For details, see http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-init.html.

If not using CloudFormation, the lower-level `cloud-init` service can be used directly. For details, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html.

Other methods can be used to configure the environment as well, but must be carefully considered and tested due to differences in how the system configures the global root environment during boot, run-level changes and manual service operations (start/stop/restart).

Depending on the deployment tools used to initialize the system and the process and ordering of RPM installation, system configuration and startup, different methods of configuration may be needed to make sure the system is configured correctly before the first launch of MarkLogic on that instance, and that all instances in the group have consistent configuration.

The sample Cloud Formation templates implement an architecture and strategy that is well defined and tested. They are a good model to follow as a design pattern regardless of the tools used for implementation.

The following environment variables are recognized on startup of MarkLogic, or are automatically set from several configuration locations. Some values must be the same across all nodes in a cluster and some may vary for each instance. The sample templates and reference architecture use the Auto Scaling Group (ASG) Launch Configuration for initializing instance variables. One ASG per zone is used so that each zone can have different configurations, but within each zone (or ASG) the same values can be used.

- MARKLOGIC_EC2_HOST — If set to 0 then all EC2 all of AWS related features are disabled. MarkLogic will not access instance metadata by any means and the rest of the variables below are unused. By default, this variable is set to 1 (enable).

  This is useful for when you want to manage MarkLogic externally.

- MARKLOGIC_MANAGED_NODE — Controls the managed cluster feature. If set to 0 (disabled), MarkLogic will not automatically mount volumes, report instance status to DynamoDB, or automatically join a cluster. By default, this flag is set to 1 (enable).

  If you only want to use the IAM role, set MARKLOGIC_EC2_HOST=1 and MARKLOGIC_MANAGED_NODE=0.

- MARKLOGIC_BOOT_WAIT — If set, then the value is a number in seconds (default = 30) as the maximum time to wait for the initial data volume (`$MARKLOGIC_EBS, default /dev/sdf`) to come online. This is only used when MARKLOGIC_EBS_VOLUME is not specified and MarkLogic is waiting for a volume to be attached manually or from an external process.

  If the timeout is reached without a volume attached then startup aborts.

- MARKLOGIC_LICENSE_KEY — A license key to use for this MarkLogic instance. This license key is only valid for a Bring Your Own License (BYOL) AMI or a user-created AMI.

Note: A License key is not necessary to enable standard features.

- MARKLOGIC_LICENSEE — The Licensee corresponding to MARKLOGIC_LICENSE_KEY.
- MARKLOGIC_AWS_ACCESS_KEY — An AWS Access Key to be used when accessing the AWS Key Management Services (KWS) and the Simple Storage Service (S3). For details, see “Configure AWS Credentials” on page 93.
- MARKLOGIC_AWS_SECRET_KEY — An AWS Secret Key to be used when accessing the AWS Key Management Services (KWS) and the Simple Storage Service (S3). This variable must be explicitly set by the `export` keyword. For details, see “Configure AWS Credentials” on page 93.
- MARKLOGIC_AWS_SESSION_TOKEN — An optional AWS session token to be used when accessing the AWS Key Management Services (KWS) and the Simple Storage Service (S3). This variable must be explicitly set by the `export` keyword. For details, see “Configure AWS Credentials” on page 93.
- MARKLOGIC_CLUSTER_NAME — The MarkLogic cluster name used to auto-configure instances and clusters. For SimpleDB this corresponds to the "Domain" used for simpleDB (V8.0.3 and prior). For DynamoDB, this corresponds to the DynamoDB table name (V8.0.4+). This cluster name is required for any of the managed cluster features, including a single node cluster.
- MARKLOGIC_CLUSTER_MASTER — Must be set and equal to "1" for exactly one node in the cluster. The master node will create the initial databases and become the cluster bootstrap host.

  Can be set to 1 for multiple nodes named the same ending in "#" (See MARKLOGIC_NODE_NAME) in which case only the resolved name that ends in "1" will take on the role of cluster master.

- MARKLOGIC_NODE_NAME — A distinct name of a node within a cluster. Required if MARKLOGIC_CLUSTER_NAME is specified. May end in a "#". If the node name ends with a "#" such as "MyNode-#" this is taken as a variable node name. For more information see the discussion of `/sbin/service` in “Deployment and Startup” on page 44.
- MARKLOGIC_ADMIN_USERNAME — The MarkLogic Administrator username used for initial installations.
- MARKLOGIC_ADMIN_PASSWORD — The MarkLogic Administrator password used for initial installations.

  EC2 user data is not an AWS 'secure location' and cannot be cleared while the instance is running. Variables set in EC2 user data are evaluated as string literals, unlike values in

`/etc/marklogic.conf`, which are parsed as shell 'source' so are always 'plain text' (or base64 encoded).

The recommended location for configuration variables is `/etc/marklogic.conf`. For examples of using a secure store for MarkLogic credentials, see “Configuration Security Considerations” on page 39 .

- MARKLOGIC_EBS_VOLUME — The volume specification for the primary EBS volume. This volume will be attached to the logical device `/dev/sdf`, a filesystem is created, if needed, and mounted on `/var/opt/MarkLogic`. The format for this value is of the form `volspec[,volspec ...]` where `volspec` is one of:
  - vol-xxxx Attach to an existing EBS volume
  - snap-xxxx An AWS snapshot which will be used to create a volume.
  - <number> An integer from 1 to 1024 which indicates the size of the volume in GB. A fresh volume will be created.
  - <specification string> A volume specification string in the format compatible with the V1 EC2 CLI tools. This format is currently only supported by using EC2 user data or `/etc/marklogic.conf`.
  - [snapshot-id]:[volume-size]:[delete-on-termination]:[volume-type[:iops]]

  Where:

Parameter	Description
snapshot-id	an existing snapshot to use as the source of the volume
volume-size	the volume size in GB
delete-on-termination	< ignored >
volume-type	The EBS volume type, one of "standard" , "gp2" ,"io1"
iops	The Provisioned IOP (PIOP) - only allowed for volume types "iops"

Examples:

:20::gp2:true - a 20 GB volume with encryption and D storage type

snap-abcde:200::: - Create volume from snapshot "snap-abcde" and change the size to 200GB. Default gp2 volume type.

:1000::io1:2000: - A 1000 GB PIOP volume with 2000 PIOP

Notes:

- only some values are valid in combination, see the EC2 EBS documentation for details.
- One of snapshot-id or volume-size is required.
- Encrypted is only allowed with snapshot-id if the snapshot is also encrypted.
- iops is only allowed for volume type "io1"
- The default volume type if not specified is "gp2"
- For the 2nd or more specs this indicates to repeat the previous volspec. E.g. "10,20,*" indicates to create a 10 GB volume for the first node, a 20 GB volume for the 2nd and further nodes of the same name.

- MARKLOGIC_EBS_VOLUME1 ... MARKLOGIC_EBS_VOLUME9 — Up to 9 more EBS volumes in the same format as MARKLOGIC_EBS_VOLUME. These will be initialized, attached, filesystems created and mounted.
- MARKLOGIC_LOG_SNS — The Simple Notification Service (SNS) topic to be used to capture messages from the Simple Queue Service (SQS). Enter the full ARN for the SNS log topic, such as `arn:aws:sns:us-east-1:123456789012345 6:mytopic`.
- MARKLOGIC_EBS_KEY — A custom key for EBS Volumes that support encryption. The key used to encrypt the volume must be in the same region. When MarkLogic clusters are created using a CloudFormation template, the same encryption key is used to encrypt all EBS volumes in the cluster. EBS Encryption is only supported by some EC2 instance types, mostly the new generation. A value of `default` indicates the AWS default EBS key. If an empty value or no value is provided, EBS Encryption will be disabled.
- MARKLOGIC_LOG_SQS — An alternative to MARKLOGIC_LOG_SNS, The endpoint of an AWS SQS queue to post startup messages. May be used to monitor the startup progress of a cluster. If not present, empty, or set to "none" then it is not used.
- MARKLOGIC_ADMIN_AUTOCREATE — If set and cluster management is not configured, then the value is used as an EC2 metadata key, the metadata value is used for initial password for the Auto Create feature. On MarketPlace AMI's this is pre-configured to default to "instance-id."
- MARKLOGIC_AWS_SWAP_SIZE — The swap space size that is automatically configured under root volume during the system startup process. By default, swap space

size is set to 32GB and root volume size is set to 40GB. You can change the default swap space size through the CloudFormation template. If you change the default swap space size, MarkLogic reserves at least 8GB in the root volume for OS. If the root volume size is less than 8GB, swap space will not configure.

- MARKLOGIC_FEDRAMP — If set to "true", data encryption will be permanently set to "force" and configuration encryption will be permanently set to "on" in the `keystore.xml` file. If set to "true", host, port, and key IDs must be provided. If set to "true" and host, port, and key IDs are not provided, a p11-driver-path must be provided.
- MARKLOGIC_KMS_HOST — The KMS hostname to provide encryption and decryption operations for MarkLogic.
- MARKLOGIC_KMS_PORT — The port number used to communicate with KMS.
- MARKLOGIC_KMS_DATA_KEY — Identifies the key in the KMS used to encrypt data.
- MARKLOGIC_KMS_CONFIG_KEY — Identifies the key in the KMS used to encrypt configuration files.
- MARKLOGIC_KMS_LOGS_KEY — Identifies the key in the KMS used to encrypt log files.
- MARKLOGIC_P11_DRIVER_PATH — The path to a shared library supporting the PKCS #11 API.

2.8 EC2 User Data

A simple configuration method is to place all variables in the EC2 UserData. This method requires no additional software or infrastructure and can be entered using the AWS Console GUI, command line tools, AWS SDK, CloudFormation, and most third party deployment tools. However EC2 UserData is not a secure data store, so it should only be used for non-sensitive data.

Making use of the CloudInit feature in CloudFormation allows you to place a minimal 'stub' configuration in EC2 User data and the remaining data in a resource MetaData section in the template. This is significantly more secure and flexible.

In the MarkLogic startup (`/sbin/service MarkLogic <command>`), the EC2 UserData is read as lines of text, and if the line starts with "`MARKLOGIC_`" it is parsed as a name=value pair. Each of the name=value pairs is exported to the environment as <name>=<value>. For example, the `MARKLOGIC_CLUSTER_NAME` user data variable becomes `MARKLOGIC_CLUSTER_NAME` shell environment variable, but `MYNAME=MYVALUE` is ignored. Use of the `MARKLOGIC_` prefix is a security precaution to avoid users passing in arbitrary system environment variables, such as PATH. Similarly the `UserData` is parsed and the environment variables explicitly created rather than the text being eval'd so that arbitrary code injection cannot occur.

Any UserData line not starting with `MARKLOGIC_` is ignored so users are free to pass in additional name=value pairs in UserData, or to use it in its entirety for other purposes as long as lines do not start with `MARKLOGIC_`.

2.9 Configuration using the /etc/marklogic.conf File

If, for some reason, you cannot use a CloudFormation template to configure the UserData with the MarkLogic configuration variables described on "AWS Configuration Variables" on page 32, an alternative is to create an `/etc/marklogic.conf` file, which will be read by the MarkLogic on startup. This file is not provided on the AMI or in the RPM explicitly so that customizations will not be overwritten on upgrades of either the AMI or RPM. If you create and populate this file before the initial startup of MarkLogic, then it is sourced (evaluated by the shell invoking `/etc/sysconfig/MarkLogic`). Any of the supported configuration environment variables set as the result of sourcing `/etc/marklogic.conf` are exported and evaluated in the order and precedence described in "Deployment and Startup" on page 44.

As described in "AWS Configuration Variables" on page 32, by adding `MARKLOGIC_EC2_HOST=0` to the `/etc/marklogic.conf` file, the startup and management features are disabled.

> **Note:** See "Configuration Security Considerations" on page 39 for a recommended method to provide secure credentials.

The `/etc/marklogic.conf` file can be useful for building custom AMI's, integrating with deployment tools that make use of EC2 UserData difficult, and manual customization. The file can be created prior to installing the MarkLogic RPM and will not be deleted when you uninstall the RPM.

The following is an example `/etc/marklogic.conf` file. Most of the `MARKLOGIC` variables are exported (meaning set) by default. However, the `export` keyword is required for variables to be used by AWS, as shown below.

> **Note:** Always use `export` when setting environment variables in the `marklogic.conf` file.

```
export MARKLOGIC_HDFS_KERBEROS_KEYTAB=/space/jsolis/b9_0/qa/ldap/keytab/servic
es.keytab_builder_bad
export MARKLOGIC_HDFS_KERBEROS_PRINCIPAL=HTTP/builder@MLTEST1.LOCAL_bad
export MARKLOGIC_KEYTAB=/space/jsolis/b9_0/qa/ldap/keytab/services.keytab_buil
der
export MARKLOGIC_PRINCIPAL=HTTP/builder@MLTEST1.LOCAL
export JAVA_HOME=/home/builder/java/jdk1.8.0_72/
export MARKLOGIC_AWS_ACCESS_KEY=HD888DJ@92KDDjdjUDUDD
export MARKLOGIC_AWS_SECRET_KEY=@kddkKidiJndk7DDD
```

2.10 Other Configuration Methods

Other configuration methods, such as modifying the global profile (`/etc/profile`), root startup scripts, or editing /`etc/sysconfig/MarkLogic` are possible, but are not recommended. It is not guaranteed that changes to these files will survive updates to the OS or MarkLogic or that, even if untouched, that they will function the same at a later time. OS upgrades frequently modify the configuration of the root or init environment, changing the set of exported variables in effect during startup. Scripts that invoke `/sbin/service MarkLogic <command>` directly need to have the same environment as the init environment.

2.11 Configuration Security Considerations

In order to provide credentials for automated creation of the initial admin user, the variables MARAKLOGIC_ADMIN_USERNAME and MARKLOGIC_ADMIN_PASSWORD need to be set during the startup process described in “Deployment and Startup” on page 44. This is necessary for the initial installation and for rejoining the cluster in the event of a node termination and restart. The password is only used in the initial startup process and not exported to the MarkLogic process or stored on disk.

In order to provide a known password to the system securely, a plain text password should not be stored in `/etc/marklogic.conf` and passed in EC2 UserData. One simple method recommended by AWS is to make use of a private S3 bucket with encrypted storage and data transmission and in combination with a AMI Role that grants read-only access to the EC2 instances in the cluster. Using the AWS CLI, the password can be securely retrieved and passed to MarkLogic on demand. This command should be placed in `/etc/marklogic.conf` as the MARKLOGIC_ADMIN_PASSWORD variable.

See the AWS CLI for details: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-using.html.

The following is an example of a complete `/etc/marklogic.conf` file that securely retrieves credentials from S3:

```
export MARKLOGIC_CLUSTER_NAME=JOE-CFN-JOESecure5x-MarkLogicDDBTable-
164OK8LD6ARMY
export MARKLOGIC_EBS_VOLUME=vol-1111111
export MARKLOGIC_NODE_NAME=NodeA#
export MARKLOGIC_ADMIN_USERNAME=admin
##
export MARKLOGIC_ADMIN_PASSWORD=\
  $(aws s3 --region us-east-1 cp s3://marklogic.joesbucket/secret-password - )
##
export MARKLOGIC_CLUSTER_MASTER=1
export MARKLOGIC_LICENSEE=none
export MARKLOGIC_LICENSE_KEY=none
export MARKLOGIC_LOG_SNS=arn:aws:sns:us-east-1:02344343341:JOE-LOG-NOTIFY
```

> **Note:** Variables containing spaces must appear in quotes. For example: `MARKLOGIC_LICENSEE="Carp Corporation".`

For multiple zone clusters, since EC2 instances are created by the AutoScalingGroup, which uses a single LaunchConfiguration per ASG, the environment is identical for every EC2 instance created in that zone. The configuration variables are designed to allow for the nodes in each zone to have identical configuration values. The same concept is used to allow a variable number of nodes per zone. The configuration in the preceding example can be used for all nodes in a single zone. For each additional zone, the following three values need to be different, but the rest must be identical:

```
# ... Same as Zone  except for ...
export MARKLOGIC_EBS_VOLUME=vol-2222222
```

```
export MARKLOGIC_NODE_NAME=NodeB#
export MARKLOGIC_CLUSTER_MASTER=0
#....
```

Similar mechanisms can be used, such as connecting to a secure key manager to decrypt an encrypted password stored on disk.

The `/etc/marklogic.conf` file must be created before the first startup of MarkLogic for the host. If the username and password are changed externally, the password retrieved by `/etc/marklogic.conf` must return the current password or the node will fail to rejoin the cluster when restarted.

For an example of creating `/etc/marklogic.conf` with CloudFormation, see “Using CloudFormation with Secure Credentials” on page 81.

3.0 Deploying MarkLogic on EC2 Using CloudFormation

This chapter describes how to deploy MarkLogic Server using a CloudFormation Template.

3.1 What CloudFormation Template Version to Use

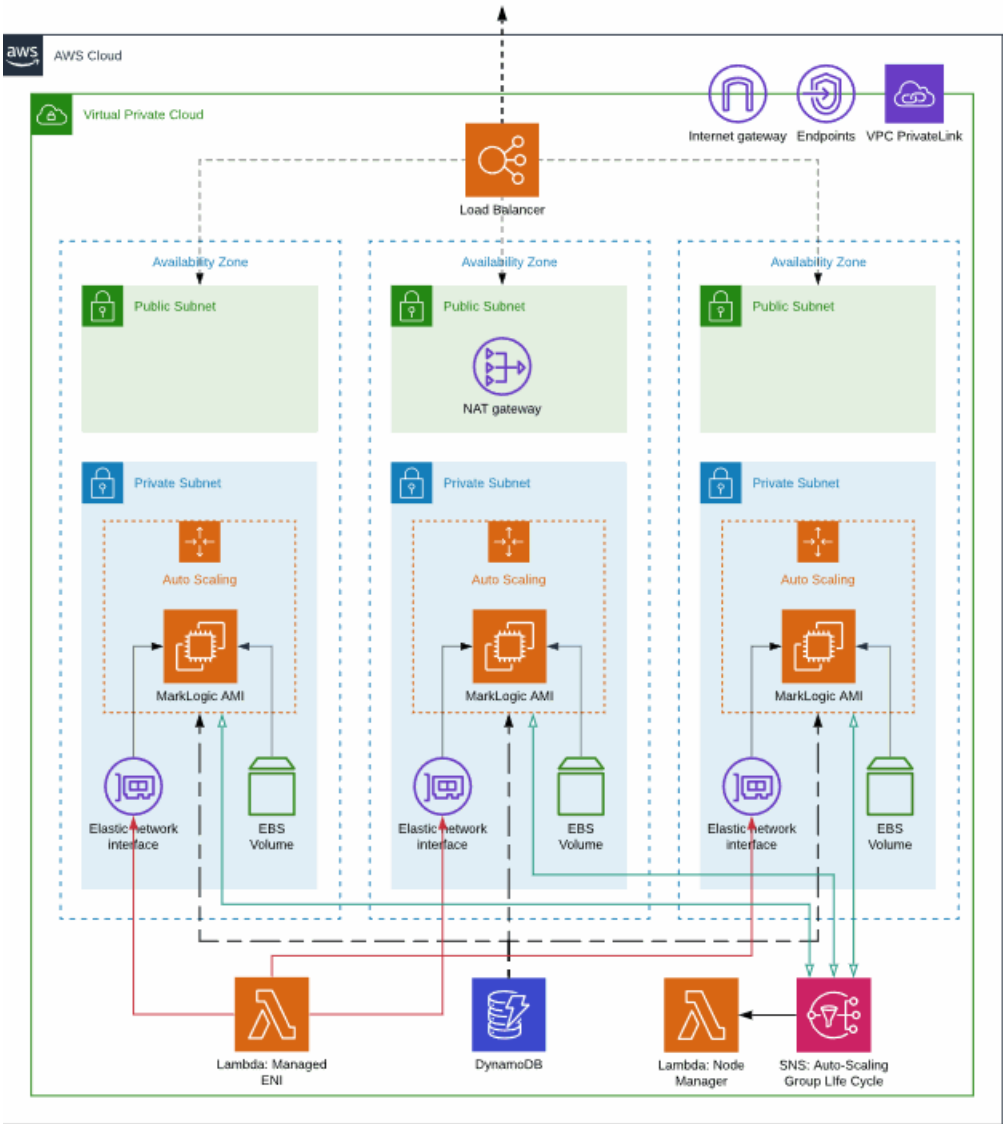
There are two basic versions of the MarkLogic CloudFormation Template. One template will launch a MarkLogic cluster on AWS with a new VPC. The other template will launch a MarkLogic cluster in an existing VPC on AWS. Both templates allow you to specify parameter values at startup to configure the cluster.

Both templates enable you to launch clusters with a Classic Load Balancer (CLB) or Application Load Balancer (ALB) based on the Number of Zones, Elastic Block Storage, Auto Scaling Group, and so on. Your cluster can be in either one Availability Zone or three Availability Zones. Multiple nodes can be placed within each Availability Zone.

> **Note:** The examples in this chapter are based on the MarkLogic AWS template that creates a new VPC.

3.2 Overview of a MarkLogic Cluster on AWS

A Managed Cluster is automatically initialized and pre-configured with recommended topology, such as the one illustrated below. This diagram shows a typical architecture of MarkLogic Cluster on AWS. Elastic Network Interfaces are attached to each MarkLogic node and CloudWatch is used to monitor the EC2 instance health. AWS Lambda is used to managed Elastic Network Interface in VPCs.



The sample CloudFormation templates implement a simple example of this reference architecture and makes use of the Managed Cluster feature. Regardless of how the cluster is created, the necessary components need to be created, configured and deployed in a controlled fashion.

Cloud Formation is an AWS Technology that allows you to specify the set of components necessary for creating a *Stack*. You can use one of the provided Amazon Cloud Formation templates to create a Managed Cluster. The Managed Cluster templates create:

- IAM Roles necessary for running AWS services without needing to pass in security credentials
- Security groups to control the incoming network traffic delivered to the instances.
- AutoScaling groups one per node
- Launch Configuration for the AutoScaling Groups
- Load balancer fronting all of the nodes
- EBS Volumes for each node

When using the Cloud Formation templates there are parameters that must be filled in (either via the AWS Console or any 3rd party command line tool that can launch a cloud formation stack). These parameters include:

- What Zone each node will run in
- The admin user and password for initially creating the security database
- The SSL Key name (Used to login to the instances once they are started)
- The size and EBS type of the volumes (in GB) to create for the initial data volume /var/opt/MarkLogic
- The EC2 instance type of the created instance.
- Optional: The Simple Notification Service (SNS) topic to be used to capture messages from the AutoScaling Groups and Managed Cluster Support startup procedure.

When launched, the Cloud Formation creates all the necessary resources. On startup, the Amazon EC2 nodes recognize that they are part of a Managed Cluster and perform the following actions without user intervention:

- Attach any volumes associated with this node
- Create a filesystem, if needed
- Mount the filesystem
- Start MarkLogic
- Apply and accept the EC2 license
- Either create the initial node (master) and set the admin username and password or attach to the cluster
- Associate the node with the Load Balancer

The Load Balancer detects proper running of MarkLogic via the HealthCheck App Server on port 7997 and will only direct traffic to that node if it has verified that the MarkLogic instance is up and running.

Each AutoScaling Group (ASG) detects system stability and will terminate and restart the node if the operating system is having problems. At any time you can hibernate the cluster by setting the ASG `NodesPerZone` value to 0. You can then restart the node by resetting the `NodesPerZone` to the previously set value. On restart, either by resuming from hibernate or restarting from the ASG detecting faults and restarting the server, the system will automatically do the following:

- Detect any previously attached volumes and re-attach them
- Detect if the hostname has changed since the previous start and, if so, rename the host to the new hostname in the MarkLogic cluster
- Re-attach to the cluster

3.3 Deployment and Startup

MarkLogic is started as either a system service (from `/sbin/service`) or manually (for example, service MarkLogic start). The standard install starts MarkLogic on the next reboot after install, however it may be started via a script or system configuration at any point.

Any customization to the startup environment must be completely in place before MarkLogic starts the first time after an install so that it properly configures its role (single, cluster master, cluster joiner), detects the correct data volumes, Java JVM, paths, and other configurable information. This section describes the AWS-specific configuration variables.

MarkLogic is typically configured to start on boot, but also may be started manually. All startup paths should be configured to inherit the same environment so that behavior is consistent. The biggest variation depends on whether or not MarkLogic is pre-installed on the AMI.

During the init process, the interaction and dependency between MarkLogic services and other services may need to be considered especially if using an AMI without MarkLogic pre-installed and configured.

The following table shows the typical startup ordering of services on an AWS Linux system.

Order	Service
02	lvm2-monitor
08	ip6tables
08	iptables
10	network
11	auditd
12	rsyslog

Order	Service
58	ntpd
80	sendmail
85	MarkLogic (Version 7)
86	tomcat-jsvc
98[c]	cloud-final (All User defined upstart and cloud-init scripts)
98[M]	MarkLogic (Version 8)
99	local (/etc/rc.local)

Note that `cloud-init` has several components, you can arrange using very low level configurations for file and config data to be populated in cloud-config state (52) but deployment tools use this for their own purposes. Most common is 'user scripts' which are run in 'cloud-final' (98[c]).

In Version 8, MarkLogic was moved to the LSB init configuration format which adds a dependency to run after `cloud-final`. This allows user configuration to be applied before MarkLogic whether or not it was pre-installed.

When MarkLogic is started, the following process runs:

1. `/sbin/service MarkLogic` is invoked . This runs via init (e.g `/etc/rc5.d/S98MarkLogic`), manually (e.g. `service MarkLogic start`)

2. `/etc/sysconfig/MarkLogic` is sourced (performing the following)

3. Default values for core env vars are defaulted

4. `/etc/marklogic.conf` is sourced (if it exists). This can modify or add variable.

5. If MARKLOGIC_EC2_HOST !=1, no additional EC2 specific processing is performed.

6. MARKLOGIC_HOSTNAME is calculated if not defined by using EC2 metadata in order

    - public-hostname
    - public-ipv4
    - local-hostname
    - local-ipv4
    - hostname

7. MARKLOGIC_AWS_ROLE is fetched from the IAM Role associated with the instance.

8. MARKLOGIC_EBS is set to `/dev/sdf` if not already set.

9. MARKLOGIC_AWS_SWAP_SIZE configures 32GB as the default swap space size value.

10. If MARKLOGIC_EC2_USERDATA != 0, then EC2 user data is read and parsed. Any name/value pairs overwrite existing settings.

11. If MARKLOGIC_CLUSTER_NAME, MARKLOGIC_NODE_NAME and MARKLOGIC_CLUSTER_MASTER is defined then the Managed Cluster logic is performed.

    - Forming or joining a cluster
    - Creating / attaching data volumes
    - Resolving hostname changes
    - Updating cluster configuration

    Note: This process is repeated on every boot and service start.

12. If Step 11 is performed, all resolved variables are cached by writing to `/var/local/mlcmd.conf` to avoid the overhead of recalculating the values on a restart.

13. If Step 11 is not performed, the following occurs:

- If MARKLOGIC_ADMIN_AUTOCREATE is set and not empty:
    - MARKLOGIC_ADMIN_PASSWORD is set to the value of the EC2 metadata who's key is $MARKLOGIC_ADMIN_AUTOCREATE. This overwrites any previous setting of MARKLOGIC_ADMIN_PASSWORD
    - If MARKLOGIC_ADMIN_PASSWORD is not empty and MARKLOGIC_ADMIN_USERNAME is empty then set MARKLOGIC_ADMIN_USERNAME="admin"
    - If MARKLOGIC_ADMIN_PASSWORD and if MARKLOGIC_ADMIN _USERNAME are both not empty then:
        - Initialize and mount any volumes specified in MARKLOGIC_EBS_VOLUME(s) configuration
        - Start the MarkLogic server
        - Create the initial admin user and initialize the security database.
        - Wait for the server to restart and validate the login (retry, timeouts as currently implemented in the Managed Cluster startup)

- Log the success or failure to the system log and console.

3.4 Creating a CloudFormation Stack using the AWS Console

This section describes how to use the AWS Console to create a CloudFormation Stack from a template. This section describes each step in the procedure, but does not discuss all of the options for each step. For more details, see:

http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-create-stack.html.

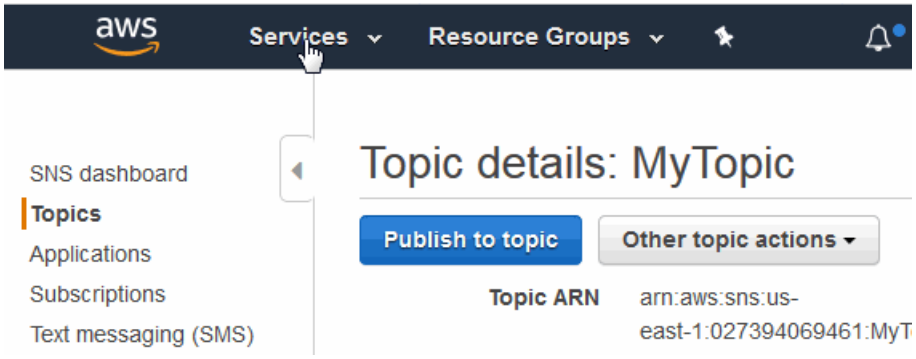
> **Note:** As described in https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpce-interface.html#vpce-interface-limitations, the services used by the MarkLogic CloudFormation templates may not be available in all Availability Zones. If a service is not supported, you will get a `UTC-0700 CREATE_FAILED` error when you attempt to create a stack.

Before you can create a CloudFormation Stack, you will need the following:

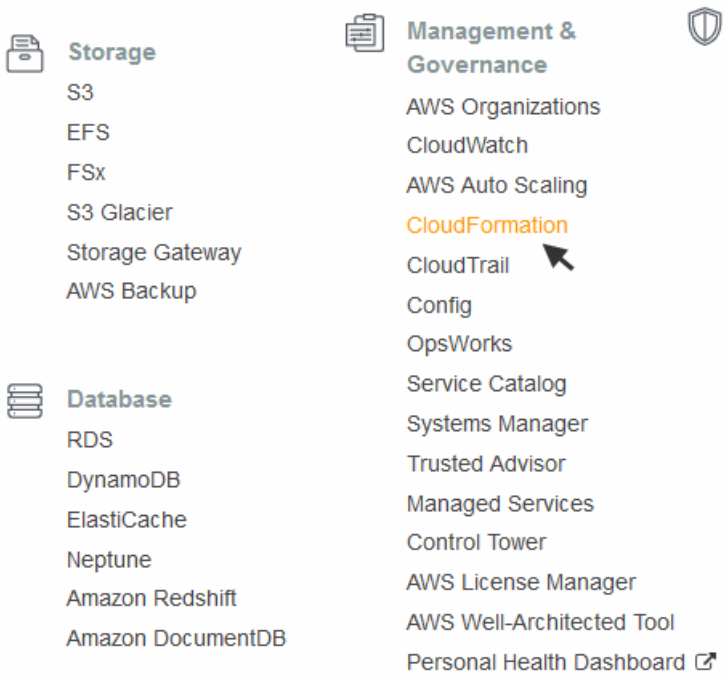
- Purchase an AMI from Amazon MarketPlace or create your own AMI (if you have an active maintenance contract, you can contact MarkLogic Technical Support for help). The sample CloudFormation templates have the latest MarketPlace AMIs embedded in them, you will need to edit these with the appropriate AMI IDs.
- A CloudFormation template. You can either obtain a template from MarkLogic or create your own, as described in “CloudFormation Template Overview” on page 61. The MarkLogic CloudFormation templates are available from http://developer.marklogic.com/products/cloud/aws.
- An IAM Role, as described in “Creating an IAM Role” on page 16
- A Key Pair, as described in “Creating a Key Pair” on page 28
- An SNS Topic, as described in “Creating a Simple Notification Service (SNS) Topic” on page 30

The following procedure describes how to create a CloudFormation Stack from a template:

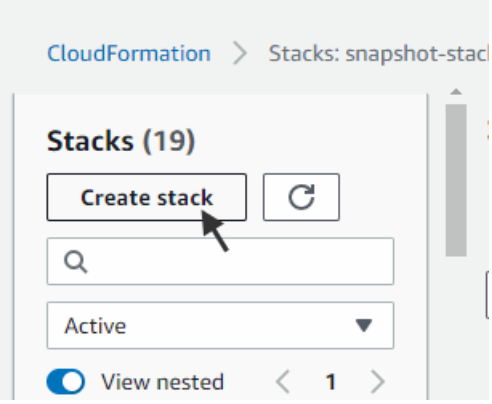
1. Click on Services in the upper left-hand portion of the AWS page to access the Amazon Web Services home page:



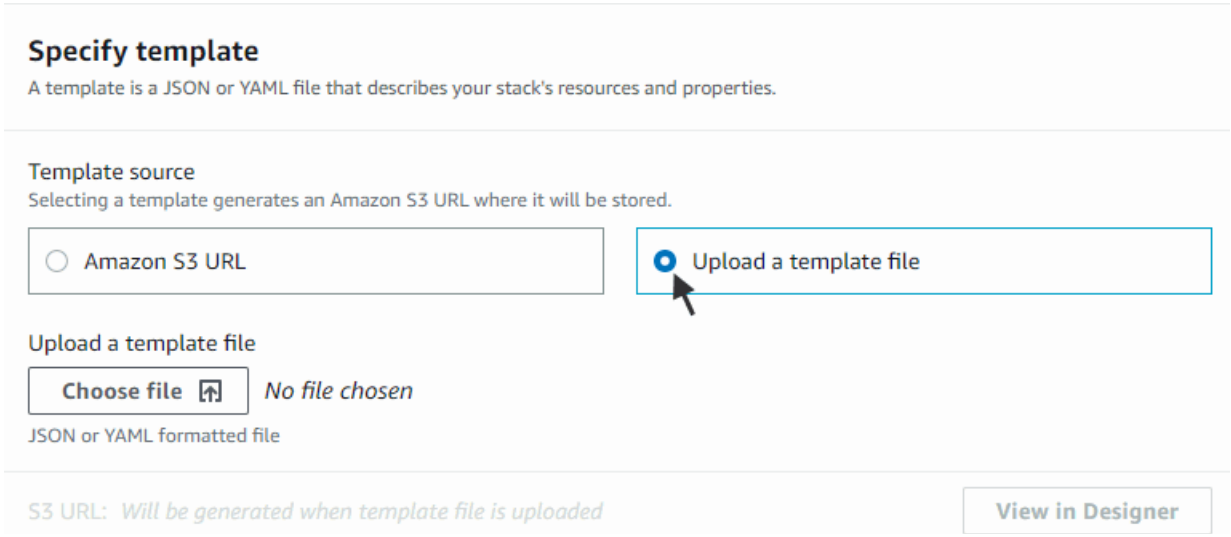
2. In the Amazon Web Services home page, click on CloudFormation:



3. In the CloudFormation Stacks page, click Create Stack.



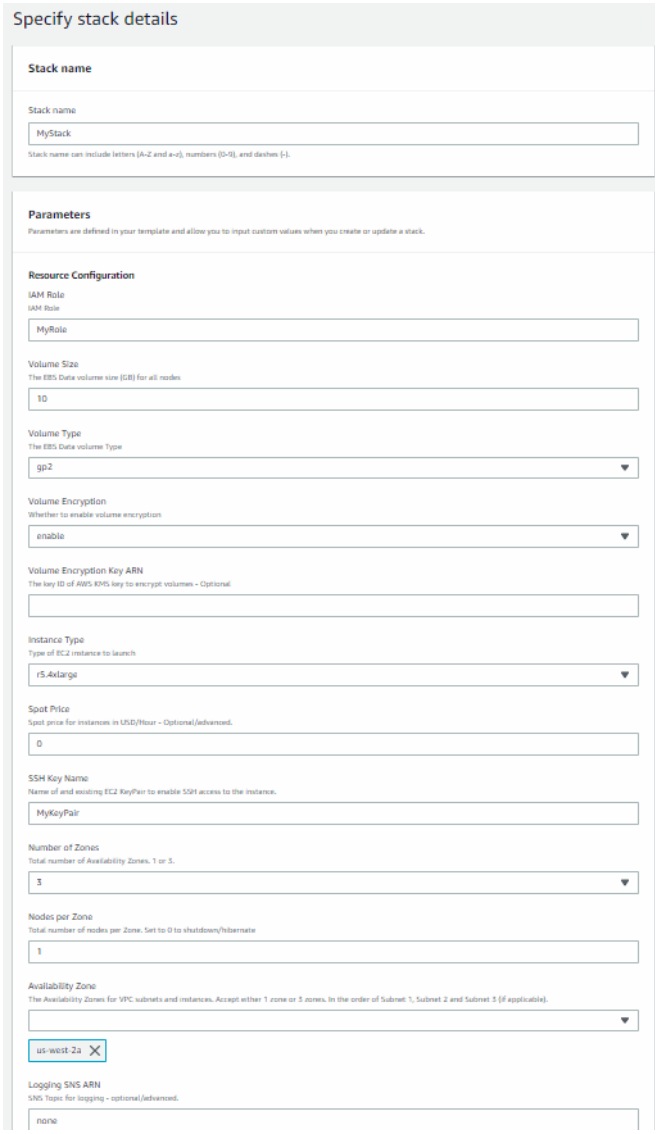
4. In the Select Template window, click Upload a template to Amazon S3 and select the CloudFormation template you downloaded from http://developer.marklogic.com/products/cloud/aws. When done, click Next.



The screens and descriptions below are for a AWS CloudFormation template that creates a new VPC.

5. In the Specify Details window, enter the name of the stack and information shown in the table below. Your Stack Name is used to identify all of the resources for your stack, including the names of your EBS volumes. It is a best practice to name your stack with an easily identifiable name, such as your user name. The EBS volumes for all but the first

node in each zone are not removed when you delete the stack, so you will want to be able to easily identify those volumes should you want to remove them after deleting your stack.



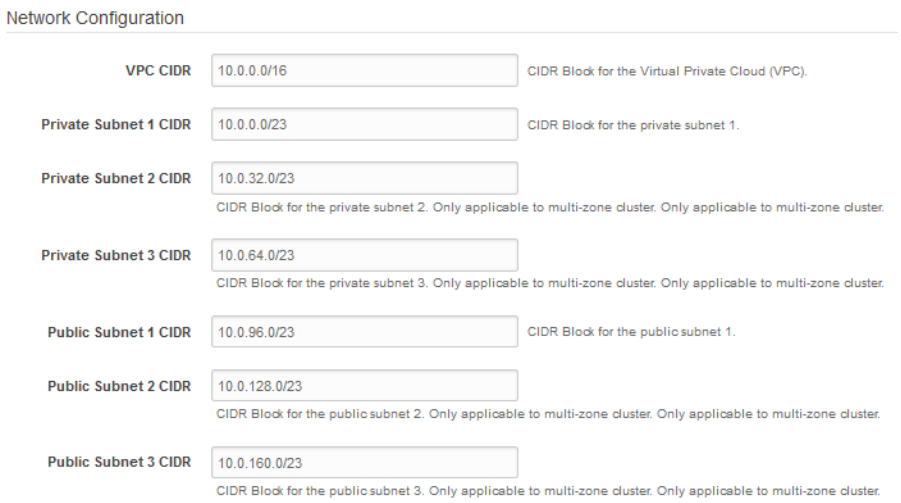
The Resource Configuration parameters are described in the following table. CloudFormation does not have real time validation of parameter values. The following assumptions are made when using CloudFormation templates to deploy clusters:

Parameter Name	Default	Description
IAMRole	Requires Input	The name of the IAM Role you created in “Creating an IAM Role” on page 16.

Parameter Name	Default	Description
Volume Size	10	The initial EBS volume size (GB). The range of valid values are 10 - 1000.
Volume Type	gp2	The EBS Data volume Type. Allowed Values: `standard` or `gp2`
Volume Encryption	enable	Whether to enable volume encryption. Select either `enable` or `disable`.
Volume Encryption Key ARN		The optional key ID of AWS KMS key to encrypt volumes.
InstanceType	r5.4xlarge	The type of EC2 instance to launch. These vary by release, product type, zone, region, and availability. There is no one single instance type that works for all MarkLogic deployments. Do note, however, that MarkLogic deployments generally have higher memory and storage I/O bandwidth requirements than legacy RDBMS deployments - so you'll likely want to start with Memory Optimized, Storage Optimized, or General Purpose instance types. The best instance type for your deployment will depend on your application code, workload, networking/system/cluster configurations, storage options, cloud architecture, etc. We recommend doing extensive testing in lower environments before using a specific instance type in production. Refer to http://developer.marklogic.com/products/cloud/aws for the current supported values for these fields. For details on each instance type, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html. **Note:** Only HVM instance types are now supported for Marketplace AMI's, PVM types may be used with custom AMIs. MarkLogic AMIs will not run on micro instances.
SpotPrice	0	Spot price for instances in USD/Hour. Allowed values are: 0 - 2. If not 0, then the amount given is a spot request for the instances is used instead of on-demand.
SSH Key Name	Requires Input	The name of the Key Pair you created in “Creating a Key Pair” on page 28.

Parameter Name	Default	Description
Number of Zones	3	Total number of Availability Zones in a region. Allowed values are: 1 or 3.
Nodes Per Zone	1	The number of nodes (hosts) to create for each zone. Allowed values are: 0 to 20. For example, a value of `1` will create one node for each zone, a total of three nodes for the cluster. A value of `0` will shutdown/hibernate all nodes.
Availability Zone	Requires Input	The Availability Zones for subnets. Accept either 1 zone or 3 zones. In the order of Subnet 1, Subnet 2 and Subnet 3 (if applicable). Each zone in your cluster should be in the same region, such as `us-east` or `us-west`. The values of the `Availability Zone` and `Number of Zones` parameters must match.
Logging SNS ARN	none	The Simple Notification Service (SNS) needed for logging. Enter the entire Topic ARN as it appears in the SNS Dashboard (for example, `arn:aws:sns:us-east-1:1234567890123456:mytopic`). For details on how to obtain an SNS Topic, see “Creating a Simple Notification Service (SNS) Topic” on page 30.

6. In the Network Configuration portion of the window are the parameters for the new VPC.



The Network Configuration parameters for the new VPC are described in the following table.

Parameter Name	Default	Description
VPC CIDR*	10.0.0.0/16	CIDR Block for the Virtual Private Cloud (VPC).
Private Subnet 1 CIDR*	10.0.0.0/23	CIDR Block for the private subnet 1.
Private Subnet 2 CIDR*	10.0.32.0/23	CIDR Block for the private subnet 2. Only applicable to multi-zone cluster.
Private Subnet 3 CIDR*	10.0.64.0/23	CIDR Block for the private subnet 3. Only applicable to multi-zone cluster.
Public Subnet 1 CIDR*	10.0.96.0/23	CIDR Block for the public subnet 1.
Public Subnet 2 CIDR*	10.0.128.0/23	CIDR Block for the public subnet 2. Only applicable to multi-zone cluster.
Public Subnet 3 CIDR*	10.0.160.0/23	CIDR Block for the public subnet 3. Only applicable to multi-zone cluster.

If you are using the MarkLogic AWS template that uses an existing VPC, you will see the following parameters:

Parameter Label	Default	Description
VPC	Requires Input	ID of existing Virtual Private Cloud. When deploying to an existing VPC, the Subnets must be in the specified VPC.
Public Sub net 1	Requires Input	The public subnet 1 in the VPC. This subnet must reside within the first selected Availability Zone (AZ). You must provide values for all three public subnets. If you only select one AZ, the second and third subnets will be ignored.
Public Sub net 2	Requires Input	The public subnet 2 in the VPC. This subnet must reside within the second selected Availability Zone (AZ). You must provide values for all three public subnets. If you only select one AZ, the second and third subnets will be ignored.
Public Sub net 3	Requires Input	The public subnet 3 in the VPC. This subnet must reside within the third selected Availability Zone (AZ). You must provide values for all three public subnets. If you only select one AZ, the second and third subnets will be ignored.
Private Sub net 1	Requires Input	The private subnet 1 in the VPC. This subnet must reside within the first selected Availability Zone (AZ). You must provide values for all three private subnets. If you only select one AZ, the second and third subnets will be ignored.
Private Sub net 2	Requires Input	The private subnet 2 in the VPC. This subnet must reside within the second selected Availability Zone (AZ). You must provide values for all three private subnets. If you only select one AZ, the second and third subnets will be ignored.
Private Sub net 3	Requires Input	The private subnet 3 in the VPC. This subnet must reside within the third selected Availability Zone (AZ). You must provide values for all three private subnets. If you only select one AZ, the second and third subnets will be ignored.

7. In the MarkLogic Configuration portion of the window are the parameters for the accessing MarkLogic Server. Click Next at the bottom when done.

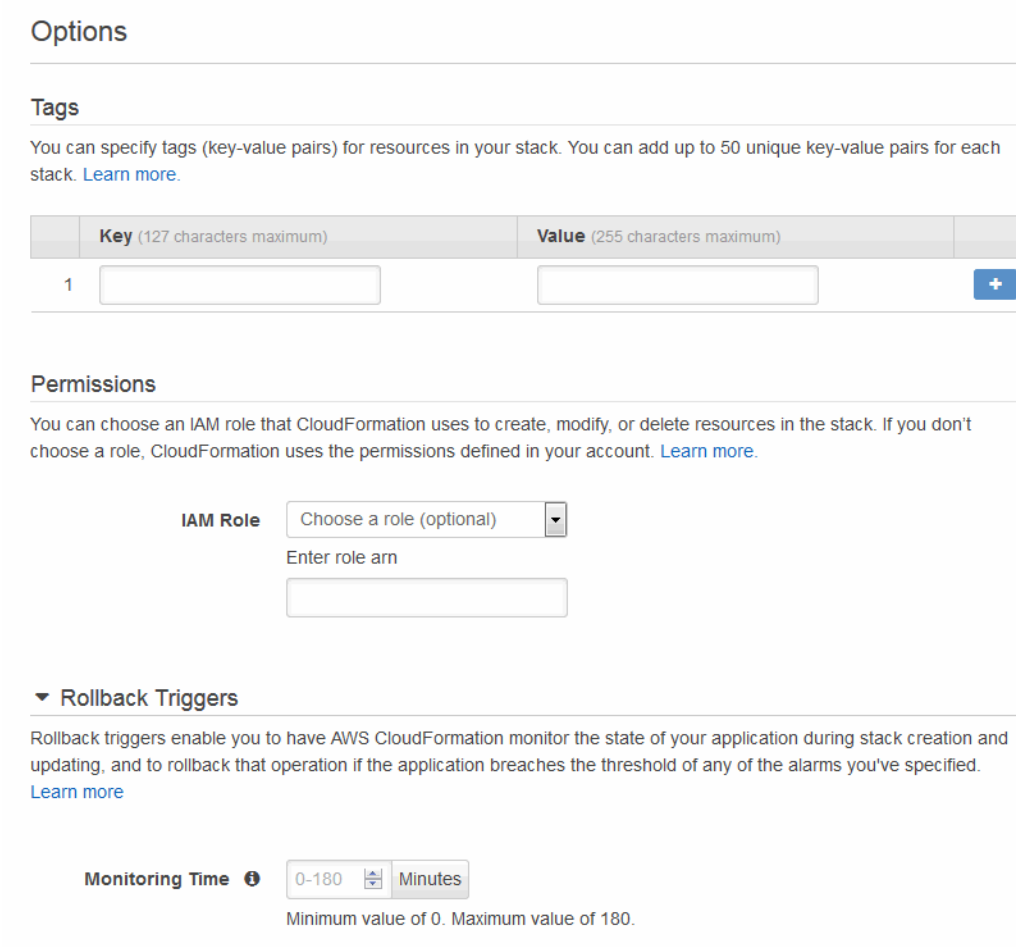
The MarkLogic Configuration parameters are described in the following table.

Parameter Name	Default	Description
AdminUser	Requires Input	The username you want to use to log in as the MarkLogic Administrator.
AdminPass	Requires Input	The password you want to use to log in as the MarkLogic Administrator.
Licensee	none	The name of the licensee obtained from your MarkLogic representative. Enter `none` if you plan to enter the license information later.
LicenseKey	none	The license key obtained from your MarkLogic representative. Enter `none` if you plan to enter the license information later.

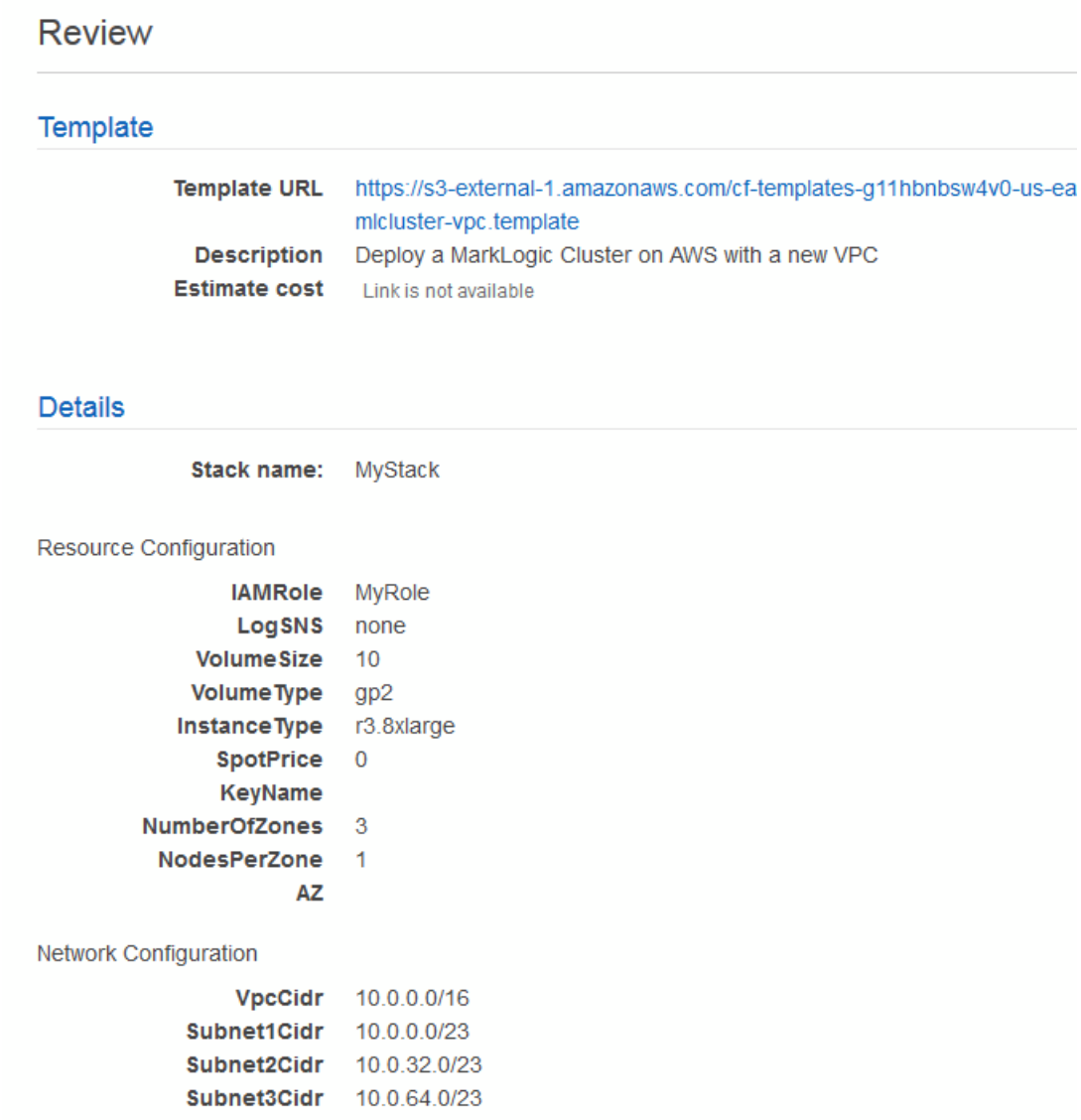
Note: If you want to use the BYOL (bring your own license) option, you must enter the License information in the Licensee and LicenseKey fields. If you don't provide any values for those fields, or leave the default `none`, the template automatically chooses the "pay as you go instances" option without any notification.

8. In the Options window, enter any tags for your stack. The tag(s) you provide identify your EC2 resources in the EC2 dashboard. For example, if you identify the Key as `Name`, the given Value (`Test Stack`, for example) will appear in the Name column of the Instance list in the EC2 dashboard. For details on tags, see http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-add-tags.html. Enter the role-back triggers, as described in https://docs.aws.amazon.com/AWSCloudFormation/latest/APIReference/API_RollbackConfiguration.html. When done, click Next.

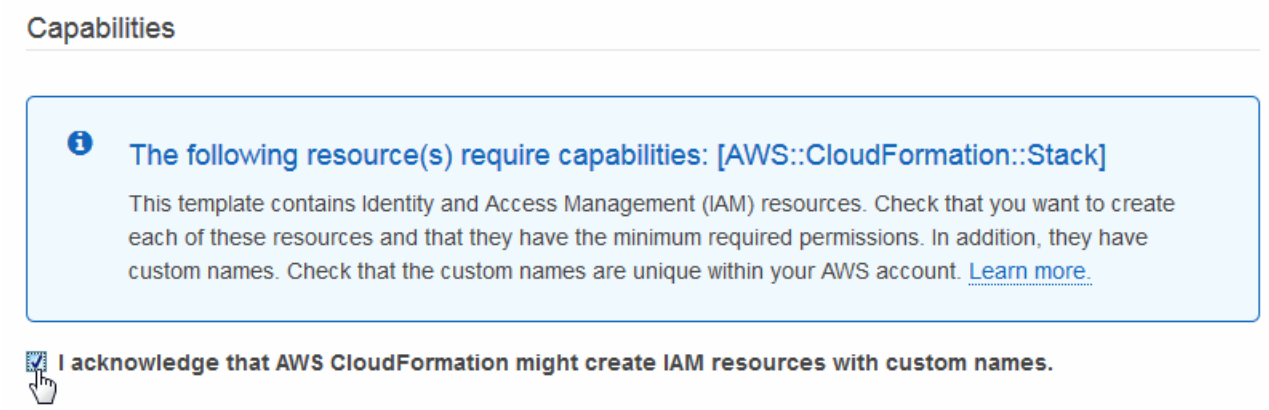
Note: Do not select an IAM Role under Permissions.



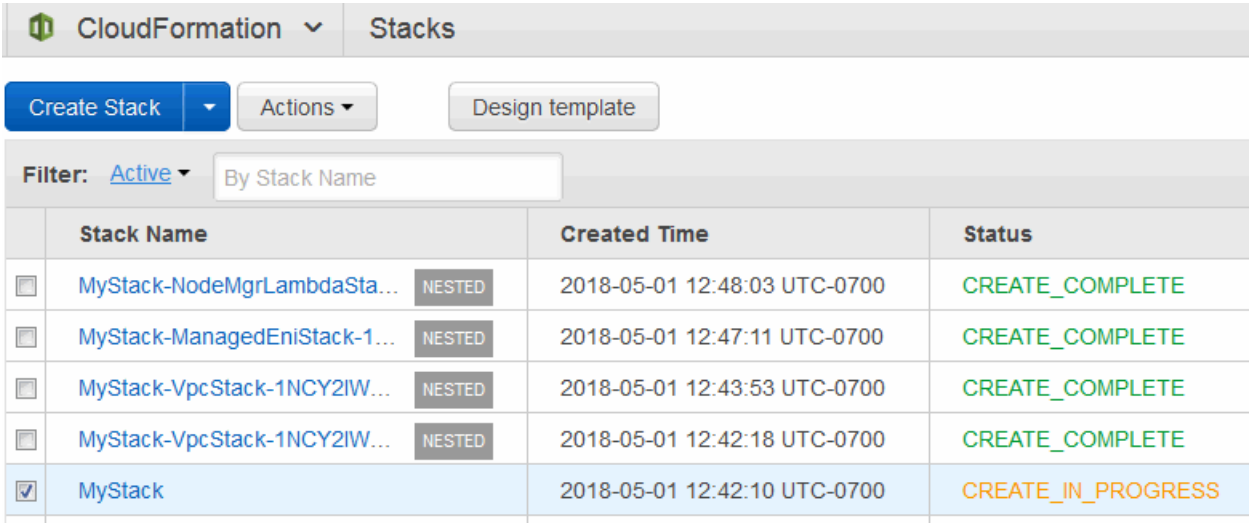
9. In the Review window, review the settings. Click Previous to make any changes. When done, click Create.



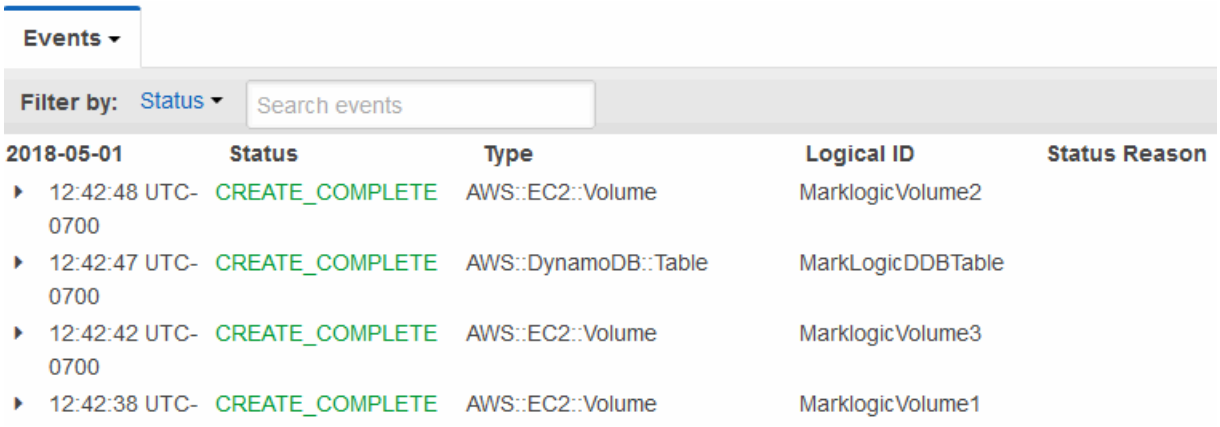
10. Click on the “I acknowledge” prompt. Skipping this step will result in a failed stack.



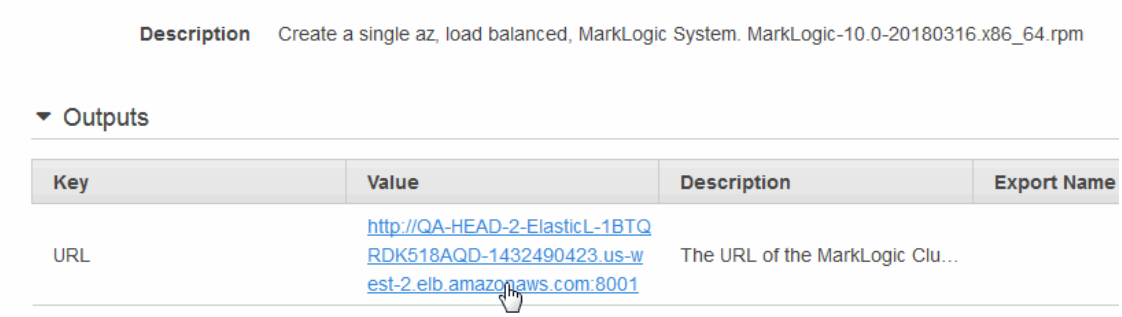
11. You will be notified that the stack is being created. The name, create date, and status of your stack will appear at the top of the page.



12. It takes a few minutes depending on the speed of AWS and the number of resources you are creating in the stack. You can Use the Events tab in the bottom portion of the page to view the progress of your stack creation. Click Refresh to see the latest status.



13. A status of CREATE_COMPLETE indicates that your AutoScaling groups have been created. Wait approximately 5-10 minutes for your EC2 instances to boot up before opening your Stack Detail page, navigating to the Outputs section, and clicking the Load Balancer URL in the Value column. This will open the MarkLogic Admin Interface on an available instance.



Note: If the URL in the Outputs tab does not work, wait another 5-10 minutes and try again.

14. Log in using the administrator username and password you specified in Step 5.

Note: Do not make any changes in the Administrator Interface until all of the hosts have been created and joined the cluster. If in doubt about the status of your stack, check the logs from the SNS topic described in “Creating a Simple Notification Service (SNS) Topic” on page 30.

3.5 Creating a CloudFormation Stack using the AWS Command-Line Interface

In addition to using the AWS CloudFormation console, you can use the AWS CloudFormation command line interface (CLI) to create a CloudFormation stack. The AWS CloudFormation CLI is described in http://aws.amazon.com/cli/.

Note: The AWS command line tools do not work with spaces for CloudFormation parameter values. Any parameter values containing a space will result in an error.

The list of CLI commands are documented in http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/CFN_CMD.html.

The following is a summary on how to create a stack using the AWS CloudFormation CLI:

1. Install and configure AWS CloudFormation CLI environment for your system, as described in http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-installing-cli.html.

2. Call the cfn-create-stack function with similar parameters as shown in "Creating a CloudFormation Stack using the AWS Console" on page 47. In addition, you must include the parameter: `--capabilities "CAPABILITY_IAM"`, as described in https://aws.amazon.com/cloudformation/resources/templates/govcloud-us/.

3. The cfn-create-stack function runs asynchronously, so it will return an id for the stack before the stack is created. You can use the cfn-describe-stack-events command with the stack id to check the status of your stack.

4. Once the stack is created, you can use the cfn-describe-stacks function to obtain the URL to the MarkLogic Admin Interface.

3.6 CloudFormation Template Overview

CloudFormation Templates consist of JSON code that is used to create a collection of AWS resources known as a *stack*. CloudFormation Templates are described in detail in http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-guide.html. This section describes the CloudFormation Template used to create a stack that consists of a three-plus node MarkLogic cluster and creates a new VPN.

Note: Amazon Linux 2 is the recommended base image for a customed MarkLogic image.

The Sample Templates available from http://developer.marklogic.com/products/cloud/aws are designed to demonstrate the architecture and IT requirements for the managed cluster feature and be useable out of the box as an example only. A production template will likely need to be customized to accommodate your specific IT requirements and may hard code many of the values exposed as parameters and mappings in these examples. For example, if you will only run in one region, there is no need for a mapping table of Region to AMI ID.

Note: Before attempting to modify this template, it is a best practice to run the unmodified template, as described in "Creating a CloudFormation Stack using the AWS Console" on page 47, to become familiar with the procedures for building a cloud stack.

The Sample Templates call sub-templates and wait for their completion. There are four sub-templates:

- VPC Stack
- Managed ENI Stack
- Node Manager Stack
- Endpoint Stack

Each of the sub-templates can be used separately. For example, you can use the VPC stack template to create a VPC and use the master template for an existing VPC to launch a MarkLogic cluster.

3.6.1 VPC Stack

The VPC Stack is only applicable to the template that creates a MarkLogic cluster with new VPCs. The following resources are created with this stack:

- VPC
- Subnet 1
- Subnet 2 (if applicable)
- Subnet 3 (if applicable)
- VPC Route Table
- VPC Route
- Internet Gateway

The Internet Gateway, VPC Route and Route Table are configured so that each node in the cluster can have access to the internet.

3.6.2 Managed ENI Stack

The Managed ENI Stack deploys a Lambda function to define a custom resource in Cloud Formation template called the Managed ENI. The Lambda function uses the AWS Python SDK (boto3) to define CloudFormation lifecycle hook to manage the Elastic Network Interface.

Upon launch of the stack, the AWS Lambda function creates an Elastic Network Interface based on the node count, subnets, and security group. The Network Interfaces created are tagged with a stack identifiers.

Upon deletion of the stack, the AMS Lambda function deletes the Elastic Network Interfaces that were tagged with the stack identifiers mentioned above.

The Managed ENI Stack defines a new IAM role with the following policies:

- ec2:CreateNetworkInterface
- ec2:DeleteNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:CreateTags
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

3.6.3 Node Manager Stack

The Node Manager Stack deploys a Lambda function (using AWS Python SDK boto3) that is hooked up with Auto Scaling Group's life cycle event and manages each cluster node. The following resources will be created by the stack:

- Lambda Function
- IAM Role
- SNS Topic
- Lambda Permission (to invoke)

The Node Manager Stack defines a new IAM role with the following policies:

- ec2:DescribeNetworkInterfaces
- ec2:AttachNetworkInterface
- ec2:DescribeInstances
- autoscaling:CompleteLifecycleAction
- sns:Publish
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

3.6.4 Endpoint Stack

The Endpoint Stack sub-template is invoked by the VPC Stack sub-template to create AWS Interface Endpoints for the VPC. Endpoint Stack creates VPC endpoints for EC2, KMS and ELB in the same region of the parent stack. The following resources are created by Endpoint Stack:

- Lambda Function
- IAM Role

- EC2 Interface Endpoint
- ELB Interface Endpoint
- KMS Interface Endpoint

The Endpoint Stack defines a new IAM role with the following policies:

- ec2:CreateVpcEndpoint
- ec2:DescribeVpcEndpoints
- ec2:DeleteVpcEndpoints
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

Note: You must have the IAM privilege to create IAM role, otherwise the deployment will fail.

3.7 Anatomy of a CloudFormation Template

CloudFormation does not have real time validation of parameter values. The following are some of the behaviors to be aware of when using CloudFormation templates to deploy clusters.

- All of the template parameters must have values.
- The parameter values of `Availability Zone` and `Number of Zones` must match.
- When deploying to an existing VPC, the parameter values of the `Availability Zone` must be consistent with the parameter values of `Subnets`. For example, if the `Availability Zone` values are `us-west-2a`, `us-west-2b`, and `us-west-2c`, then Subnets must provide IDs of subnets located in the zones in above order.
- When deploying to an existing VPC, the Subnets must be in the specified VPC.

Note: Load balancer type depends on the number of zones selected. Select three zones for an Application Load Balancer (ALB), or select one zone for a Classic Load Balancer (CLB).

The main sections of the CloudFormation Template are as follows:

- Metadata
- Parameters Declaration
- Conditions Declaration
- Mappings Declaration
- Resources Declaration

- Outputs Declaration

These sample templates create a load balancer as well as enable a public IP for each MarkLogic Server. The Load Balancer type depends on the number of zones selected. Select three zones for an Application Load Balancer (ALB), or select one zone for Classic Load Balancer (CLB). The output of the stack lists the URL of the Load Balancer.

When the Instance Public IP address is enabled, you are able to directly access each host (port 8000 for example) and SSH (when a public DNS is configured as described in "Accessing a MarkLogic Server Instance" on page 85). Otherwise, you cannot directly access the hosts. It is a best practice to not enable the public IP address.

> **Note:** The Instance Public IP address must be enabled to use SNS topic described in "Creating a Simple Notification Service (SNS) Topic" on page 30.

Most applications need to use the load balancer as their endpoint. XCC applications, such as mlcp, need to set the `xcc.httpcompliant=true` mode to connect through the ELB regardless of session affinity issues. For details, see Using a Load Balancer or Proxy Server with an XCC Application in the *XCC Developer's Guide*.

3.7.1 Metadata

The Managed Cluster Feature uses an external metadata store (a DynamoDB table) to save the configuration information for the cluster. Whenever a cluster event happens, the metadata store is updated with latest cluster node information to ensure that the cluster remains available and reliable in different kinds of cloud service failure events.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Deploy a MarkLogic Cluster on AWS with a new VPC
Metadata:
  version: 9.0-20180427
  binary: MarkLogic-9.0-20180427.x86_64.rpm
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: "Resource Configuration"
        Parameters:
          - IAMRole
          - LogSNS
          - VolumeSize
          - VolumeType
          - InstanceType
          - SpotPrice
          - KeyName
          - NumberOfZones
          - NodesPerZone
          - AZ
      - Label:
          default: "Network Configuration"
        Parameters:
```

```
        - VPC
        - PublicSubnet1
        - PublicSubnet2
        - PublicSubnet3
        - PrivateSubnet1
        - PrivateSubnet2
        - PrivateSubnet3
    - Label:
        default: "MarkLogic Configuration"
      Parameters:
        - AdminUser
        - AdminPass
        - Licensee
        - LicenseKey
  ParameterLabels:
    AdminUser:
      default: Admin User
    AdminPass:
      default: Admin password
    Licensee:
      default: Licensee
    LicenseKey:
      default: License Key
    IAMRole:
      default: IAM Role
    LogSNS:
      default: Logging SNS ARN
    VolumeSize:
      default: Volume Size
    VolumeType:
      default: Volume Type
    InstanceType:
      default: Instance Type
    SpotPrice:
      default: Spot Price
    KeyName:
      default: SSH Key Name
    NumberOfZones:
      default: Number of Zones
    NodesPerZone:
      default: Nodes per Zone
    AZ:
      default: Availability Zone
    VPC:
      default: VPC
    PublicSubnet1:
      default: Public Subnet 1
    PublicSubnet2:
      default: Public Subnet 2
    PublicSubnet3:
      default: Public Subnet 3
    PrivateSubnet1:
      default: Private Subnet 1
    PrivateSubnet2:
```

```
      default: Private Subnet 2
    PrivateSubnet3:
      default: Private Subnet 3
```

3.7.2 Parameters Declaration

The `Parameters` portion of the template defines the parameters necessary to build your MarkLogic cluster. The three zones define the hosted zones on which the servers in cluster are to be created. All of the zones should be in the same region, as described in http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html.

For a description of each parameter, see the table at the end of Step 5 in “Creating a CloudFormation Stack using the AWS Console” on page 47.

The parameters used to configure the resources are shown below.

```
Parameters:
  # resource configuration
  IAMRole:
    Description: IAM Role
    Type: String
  VolumeSize:
    Description: The EBS Data volume size (GB) for all nodes
    Type: Number
    MinValue: '10'
    MaxValue: '1000'
    Default: '10'
  VolumeType:
    Description: The EBS Data volume Type
    Type: String
    AllowedValues:
      - standard
      - gp2
    Default: gp2
  InstanceType:
    Description: Type of EC2 instance to launch
    Type: String
    Default: r5.4xlarge
    AllowedValues:
      - ---- Essential Enterprise and Bring-Your-Own-License ----
      allowed instance types ......
      - -------------- Bring-Your-Own-License Only --------------
      allowed instance types ......
  SpotPrice:
    Description: Spot price for instances in USD/Hour -
Optional/advanced.
    Type: Number
    MinValue: '0'
    MaxValue: '2'
    Default: '0'
  KeyName:
    Description: Name of and existing EC2 KeyPair to enable SSH access
```

```
to the instance.
    Type: String
  NumberOfZones:
    Description: Total number of Availability Zones. 1 or 3.
    Type: Number
    AllowedValues:
      - 1
      - 3
    Default: 3
  NodesPerZone:
    Description: Total number of nodes per Zone. Set to 0 to
shutdown/hibernate
    Type: Number
    MinValue: '0'
    MaxValue: '20'
    Default: '1'
```

The parameters used to configure the network are shown below.

The cluster can be in either one Availability Zone or three Availability Zones. Multiple nodes can be placed within an Availability Zone. The Availability Zones for subnets. Accept either 1 zone or 3 zones. In the order of Subnet 1, Subnet 2 and Subnet 3 (if applicable).

```
  AZ:
    Description: The Availability Zones for VPC subnets. Accept either
1 zone or 3 zones. In the order of Subnet 1, Subnet 2 and Subnet 3 (if
applicable).
    Type: 'List<AWS::EC2::AvailabilityZone::Name>'
  LogSNS:
    Description: SNS Topic for logging - optional/advanced. Requires
instance public IP enabled.
    Type: String
    Default: none
  # network configuration
     Parameters for VPC and subnets ........
```

The parameters used to configure MarkLogic Server are shown below.

```
  # marklogic configuration
  AdminUser:
    Description: The MarkLogic administrator username
    Type: String
  AdminPass:
    Description: The MarkLogic administrator password
    Type: String
    NoEcho: 'true'
  Licensee:
    Description: The MarkLogic Licensee or 'none'
    Type: String
    Default: none
  LicenseKey:
    Description: The MarkLogic License Key or 'none'
```

```
      Type: String
      Default: none
```

3.7.3 Conditions Declaration

The Conditions Declaration specifies the conditions under which portions of the template are used. For example, if `NumberOfZones` is not set to 1, the `MultiZone` condition enables the template to create three Availability Zones.

```
Conditions:
  UseLogSNS:
    - !Not [!Equals [!Ref LogSNS, "none"]]
  UseSpot: !Not
    - !Equals
      - !Ref SpotPrice
      - 0
  MultiZone: !Not [!Equals [!Ref NumberOfZones, 1]]
  EssentialEnterprise:
    !And [!Equals [!Ref LicenseKey, ''], !Equals [!Ref Licensee, '']]
  UseVolumeEncryption: !Equals [!Ref VolumeEncryption, 'enable']
  HasCustomEBSKey: !Not [!Equals [!Ref VolumeEncryptionKey, '']]
```

3.7.4 Mappings Declaration

The `Mappings` portion of the template provides a way of looking up values from a table.

The `LicenseRegion2AMI` map defines the values for all of the possible instance types. The `LicenseRegion2AMI` map defines the AMIs for each region. Each region has both a `Enterprise` and `BYOL` (Bring Your Own License) AMI. For details on AMIs, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html.

Note: You can set `LambdaPackageBucket` to point to your own private bucket.

```
Mappings:
  Variable:
    LambdaPackageBucket:
      base: 'marklogic-lambda-'
    TemplateUrl:
      base: 'https://s3.amazonaws.com/marklogic-releases'
    S3Directory:
      base: '9.0-9'
  LicenseRegion2AMI:
      us-east-1:
        Enterprise: ami-0c59ddcc7544fed1c
        BYOL: ami-0721c0f5c63ecd7c1
      All other supported regions .......
```

3.7.5 Resources Declaration

The `Resources` portion of the template defines all of the AWS resources created for your stack by this template. Each resource is defined as a specific AWS type. The details of each resource type are described in http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html.

These resources defined in this template include:

- VpcStack — See “VPC Stack” on page 62 for details.
- ManagedEniStack — See “Managed ENI Stack” on page 62 for details.
- NodeMgrLambdaStack — See “Node Manager Stack” on page 63 for details.
- Elastic Block Store (EBS) volumes
- DynamoDB Table (DynamoDB is the Amazon implementation of the Metadata Database)
- AutoScaling Groups (ASG). For each ASG, there are the following resources:
    - Security Group
    - Instance Type
    - Identity and Access Management (IAM) Instance Profile
    - Launch Configuration
    - UserData
    - Load Balancer
- Load Balancer ports
- Health Check values
- Security Group for each EC2 Instance

Because ENI is not managed by the CloudFormation stack directly, the Managed ENI Lambda function needs to identify the ENIs created in order to have the ability to update or clean them up. All ENIs created by the Lambda function are tagged with stack information.

The Availability Zones for VPC subnets. Accept either 1 zone or 3 zones. In the order of Subnet 1, Subnet 2 and Subnet 3 (if applicable). The Subnets in the VPC. You must provide values for all three public and private subnets for successful deployment. The order must be same as Availability Zone(s) selected. If you only select one Availability Zone, the second and third subnets will be ignored.

Upon launch of the `ManagedEniStack` stack, the AWS Lambda function creates an Elastic Network Interface based on the node count, subnets, and security group. The Network Interfaces created are tagged with a stack identifier.

```
ManagedEniStack:
    Type: AWS::CloudFormation::Stack
```

```
    DependsOn:
      - VpcStack
      - InstanceSecurityGroup
    Properties:
      NotificationARNs:
        - !If
          - UseLogSNS
          - !Ref LogSNS
          - !Ref 'AWS::NoValue'
      Parameters:
        S3Bucket: !Join [ "", [!FindInMap
[Variable,"LambdaPackageBucket","base"], !Ref 'AWS::Region']]
        S3Directory: !FindInMap [Variable,"S3Directory","base"]
        NodesPerZone: !Ref NodesPerZone
        NumberOfZones: !Ref NumberOfZones
        Subnets: !If
          - MultiZone
          - !Join
            - ','
            - - !GetAtt [VpcStack, Outputs.PrivateSubnet1Id]
              - !GetAtt [VpcStack, Outputs.PrivateSubnet2Id]
              - !GetAtt [VpcStack, Outputs.PrivateSubnet3Id]
          - !GetAtt [VpcStack, Outputs.PrivateSubnet1Id]
        ParentStackName: !Ref 'AWS::StackName'
        ParentStackId: !Ref 'AWS::StackId'
        SecurityGroup: !Ref InstanceSecurityGroup
      TemplateURL: !Join ['/', [!FindInMap
[Variable,"TemplateUrl","base"],!FindInMap
[Variable,"S3Directory","base"],'ml-managedeni.template']]
      TimeoutInMinutes: 5
```

The `NodeMgrLambdaStack` portion of the template calls the `ml-nodemanager.template` sub-template to deploy a Lambda function that is hooked up with Auto Scaling Group's life cycle event and manages each cluster node. The resources created by the stack are described in “Node Manager Stack” on page 63.

```
  NodeMgrLambdaStack:
    Type: AWS::CloudFormation::Stack
    DependsOn: ManagedEniStack
    Properties:
      NotificationARNs:
        - !If
          - UseLogSNS
          - !Ref LogSNS
          - !Ref 'AWS::NoValue'
      Parameters:
        S3Bucket: !Join [ "", [!FindInMap
[Variable,"LambdaPackageBucket","base"], !Ref 'AWS::Region']]
        S3Directory: !FindInMap [Variable,"S3Directory","base"]
      TemplateURL: !Join ['/', [!FindInMap
[Variable,"TemplateUrl","base"],!FindInMap
[Variable,"S3Directory","base"],'ml-nodemanager.template']]
      TimeoutInMinutes: 5  NodeMgrLambdaStack:
```

The EBS volumes used by `/var/opt/MarkLogic` for the first node in Zone1, Zone2 and Zone3. For details on the `AWS::EC2::Volume` type, see http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-ebs-volume.html.

All EBS volume definitions are similar to `MarklogicVolume1` for Zone1, shown below.

```
  MarklogicVolume1:
    Type: 'AWS::EC2::Volume'
    Properties:
      AvailabilityZone: !Select [0, !Ref AZ]
      Size: !Ref VolumeSize
      Tags:
        - Key: Name
          Value: MarkLogicData 1
      VolumeType: !Ref VolumeType
      Encrypted: !If [UseVolumeEncryption, 'true', 'false']
      KmsKeyId: !If [HasCustomEBSKey, !Ref VolumeEncryptionKey, !Ref
'AWS::NoValue']
    Metadata:
      'AWS::CloudFormation::Designer':
        id: c81032f7-b0ec-47ca-a236-e24d57b49ae3
```

`MarkLogicDDBTable` creates a DynamoDB database used as the Metadata Database, described in "AWS Terminology" on page 5, and returns the name of the DynamoDB Table.

> **Note:** The read and write capacity are both set to 10 for a three-node template and 2 for a single-node template. It is critical to make sure you have enough capacity provisioned for peak periods, which occur when the instances in large cluster are restarted simultaneously. If you don't have enough capacity, the cluster may not recouple correctly when nodes are replaced following termination. You can set a CloudWatch alarm on capacity, which can either alert you manually or trigger a script to modify the capacity.

For details on the `AWS::DynamoDB::Table` type, see http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-dynamodb-table.html.

```
  MarkLogicDDBTable:
    Type: 'AWS::DynamoDB::Table'
    Properties:
      AttributeDefinitions:
        - AttributeName: node
          AttributeType: S
      KeySchema:
        - KeyType: HASH
          AttributeName: node
      ProvisionedThroughput:
        WriteCapacityUnits: '10'
        ReadCapacityUnits: '10'
    Metadata:
```

```
        'AWS::CloudFormation::Designer':
          id: e7190602-c2de-47ab-81e7-1315f8c01e2d
```

`MarkLogicServerGroup1`, `MarkLogicServerGroup2` and `MarkLogicServerGroup3` are the AutoScaling Groups (ASGs) for Zone1, Zone2 and Zone3. For details on the `AWS::AutoScaling::AutoScalingGroup` type, see http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-group.html. All of them are similar to `MarkLogicServerGroup1` for Zone1, shown below.

```
    MarkLogicServerGroup1:
        Type: 'AWS::AutoScaling::AutoScalingGroup'
        DependsOn:
          - VpcStack
          - ManagedEniStack
          - NodeMgrLambdaStack
        Properties:
          VPCZoneIdentifier:
            - !GetAtt [VpcStack, Outputs.PrivateSubnet1Id]
          LaunchConfigurationName: !Ref LaunchConfig1
          MinSize: '0'
          MaxSize: !Ref NodesPerZone
          DesiredCapacity: !Ref NodesPerZone
          Cooldown: '300'
          HealthCheckType: EC2
          HealthCheckGracePeriod: '300'
          LoadBalancerNames:
            - !Ref ElasticLoadBalancer
          NotificationConfiguration: !If
            - UseLogSNS
            - TopicARN: !Ref LogSNS
              NotificationTypes:
                - 'autoscaling:EC2_INSTANCE_LAUNCH'
                - 'autoscaling:EC2_INSTANCE_LAUNCH_ERROR'
                - 'autoscaling:EC2_INSTANCE_TERMINATE'
                - 'autoscaling:EC2_INSTANCE_TERMINATE_ERROR'
            - !Ref 'AWS::NoValue'
          Tags:
            - Key: marklogic:stack:name
              Value: !Ref 'AWS::StackName'
              PropagateAtLaunch: 'true'
            - Key: marklogic:stack:id
              Value: !Ref 'AWS::StackId'
              PropagateAtLaunch: 'true'
          LifecycleHookSpecificationList:
            - LifecycleTransition: 'autoscaling:EC2_INSTANCE_LAUNCHING'
              LifecycleHookName: NodeManager
              HeartbeatTimeout: 4800
              NotificationTargetARN: !GetAtt [NodeMgrLambdaStack,
    Outputs.NodeMgrSnsArn]
              RoleARN: !GetAtt [NodeMgrLambdaStack, Outputs.NodeMgrIamArn]
        Metadata:
          'AWS::CloudFormation::Designer':
            id: 31621dd0-4b18-4dcd-b443-db9cef64ebb1
```

`NotificationTypes` describes the notifications to be sent to the SNS Topic supplied to the cloud formation script to allow monitoring of AutoScaling group actions.

```
              NotificationTypes:
                - 'autoscaling:EC2_INSTANCE_LAUNCH'
                - 'autoscaling:EC2_INSTANCE_LAUNCH_ERROR'
                - 'autoscaling:EC2_INSTANCE_TERMINATE'
                - 'autoscaling:EC2_INSTANCE_TERMINATE_ERROR'
            - !Ref 'AWS::NoValue'
```

The `InstanceSecurityGroup` defines the ingress rules for the `SecurityGroup`.

```
        InstanceSecurityGroup:
            Type: 'AWS::EC2::SecurityGroup'
            DependsOn:
              - VpcStack
            Properties:
              GroupDescription: Enable SSH access and HTTP access on the
        inbound port
              VpcId: !GetAtt [VpcStack, Outputs.VpcId]
              SecurityGroupIngress:
                - IpProtocol: tcp
                  FromPort: '22'
                  ToPort: '22'
                  CidrIp: 0.0.0.0/0
                - IpProtocol: tcp
                  FromPort: '7998'
                  ToPort: '7998'
                  CidrIp: 0.0.0.0/0
                - IpProtocol: tcp
                  FromPort: '8000'
                  ToPort: '8010'
                  CidrIp: 0.0.0.0/0
                - IpProtocol: tcp
                  FromPort: '7997'
                  ToPort: '7997'
                  CidrIp: 0.0.0.0/0
                - IpProtocol: tcp
                  FromPort: '7999'
                  ToPort: '7999'
                  CidrIp: 0.0.0.0/0
          InstanceSecurityGroupIngress:
            Type: 'AWS::EC2::SecurityGroupIngress'
            DependsOn:
              - InstanceSecurityGroup
            Properties:
              IpProtocol: tcp
              FromPort: '0'
              ToPort: '65355'
              GroupId: !Ref InstanceSecurityGroup
              SourceSecurityGroupId: !Ref InstanceSecurityGroup
          ElbSecurityGroup:
            Type: 'AWS::EC2::SecurityGroup'
```

```
    DependsOn: VpcStack
    Properties:
      GroupDescription: Enable SSH access and HTTP access on the
inbound port
      VpcId: !GetAtt [VpcStack, Outputs.VpcId]
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '22'
          ToPort: '22'
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: '7998'
          ToPort: '7998'
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: '8000'
          ToPort: '8010'
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: '7997'
          ToPort: '7997'
          CidrIp: 0.0.0.0/0
```

`LaunchConfig1`, `LaunchConfig2 and LaunchConfig3` are the Launch Configurations for ASG 1, ASG 2 and ASG 3. These describe how to look up the AMI id associated with the region, instance type, and architecture (PVM vs. HVM). All are similar to that below for ASG 1. For details on the `AWS::AutoScaling::LaunchConfiguration` type, see http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-launchconfig.html.

```
LaunchConfig1:
    Type: 'AWS::AutoScaling::LaunchConfiguration'
    DependsOn:
      - InstanceSecurityGroup
    Properties:
      BlockDeviceMappings:
        - DeviceName: /dev/sdf
          NoDevice: true
          Ebs: {}
      KeyName: !Ref KeyName
      ImageId: !If [EssentialEnterprise, !FindInMap
[LicenseRegion2AMI,!Ref 'AWS::Region',"Enterprise"], !FindInMap
[LicenseRegion2AMI, !Ref 'AWS::Region', "BYOL"]]
```

Each Launch Configuration has a `UserData` and a `SecurityGroups` property, as shown below.

The `UserData` property that is populated with the data assigned to the variables described in “AWS Configuration Variables” on page 32. Below is the `UserData` property for ASG 1.

Note: In `VolumeSize`, the `,*` defines the volume size for the 2nd and any additional nodes in each ASG. The `#` indicates that the nodes are dynamically named and a numeric suffix is added from `1 - MaxNodesPerZone`.

```
        UserData: !Base64
                'Fn::Join':
                  - ''
                  - - MARKLOGIC_CLUSTER_NAME=
                    - !Ref MarkLogicDDBTable
                    - |+

                    - MARKLOGIC_EBS_VOLUME=
                    - !Ref MarklogicVolume1
                    - ',:'
                    - !Ref VolumeSize
                    - '::'
                    - !Ref VolumeType
                    - |
                      ::,*
                    - |
                      MARKLOGIC_NODE_NAME=NodeA#
                    - MARKLOGIC_ADMIN_USERNAME=
                    - !Ref AdminUser
                    - |+

                    - MARKLOGIC_ADMIN_PASSWORD=
                    - !Ref AdminPass
                    - |+

                    - |
                      MARKLOGIC_CLUSTER_MASTER=1
                    - MARKLOGIC_LICENSEE=
                    - !Ref Licensee
                    - |+

                    - MARKLOGIC_LICENSE_KEY=
                    - !Ref LicenseKey
                    - |+

                    - MARKLOGIC_LOG_SNS=
                    - !Ref LogSNS
                    - |+

                    - !If
                      - UseVolumeEncryption
                      - !Join
                        - ''
                        - - 'MARKLOGIC_EBS_KEY='
                          - !If
                            - HasCustomEBSKey
                            - !Ref VolumeEncryptionKey
                            - 'default'
                      - ''
```

Each Launch Configuration has a `SecurityGroups` property that assigns the security group defined by `InstanceSecurityGroup` to the Amazon EC2 instances in the Auto Scaling group. Each property is like the following.

```
        SecurityGroups:
          - !Ref InstanceSecurityGroup
        InstanceType: !Ref InstanceType
        IamInstanceProfile: !Ref IAMRole
        SpotPrice: !If
          - UseSpot
          - !Ref SpotPrice
          - !Ref 'AWS::NoValue'
      Metadata:
        'AWS::CloudFormation::Designer':
          id: 2efb8cfb-df53-401d-8ff2-34af0dd25993
```

`ElasticLoadBalancer` is the Load Balancer for all of the ASGs. For details on the `AWS::ElasticLoadBalancing::LoadBalancer` type, see http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-elb.html.

```
  ElasticLoadBalancer:
      Type: 'AWS::ElasticLoadBalancing::LoadBalancer'
      DependsOn:
        - VpcStack
        - ElbSecurityGroup
      Properties:
        AppCookieStickinessPolicy:
          - CookieName: SessionID
            PolicyName: MLSession
        SecurityGroups:
          - !Ref ElbSecurityGroup
        Subnets:
          - !GetAtt [VpcStack, Outputs.PublicSubnet1Id]
          - !If [MultiZone, !GetAtt [VpcStack, Outputs.PublicSubnet2Id],
!Ref 'AWS::NoValue']
          - !If [MultiZone, !GetAtt [VpcStack, Outputs.PublicSubnet3Id],
!Ref 'AWS::NoValue']
        ConnectionDrainingPolicy:
          Enabled: 'true'
          Timeout: '60'
        CrossZone: 'true'
```

`Listeners` defines all of the ports the Elastic Load Balancer (ELB) opens to the public.

```
        Listeners:
          - LoadBalancerPort: '8000'
            InstancePort: '8000'
            Protocol: HTTP
            PolicyNames:
              - MLSession
          - LoadBalancerPort: '8001'
            InstancePort: '8001'
            Protocol: HTTP
            PolicyNames:
              - MLSession
          - LoadBalancerPort: '8002'
            InstancePort: '8002'
```

```
      Protocol: HTTP
      PolicyNames:
        - MLSession
    - LoadBalancerPort: '8003'
      InstancePort: '8003'
      Protocol: HTTP
      PolicyNames:
        - MLSession
    - LoadBalancerPort: '8004'
      InstancePort: '8004'
      Protocol: HTTP
      PolicyNames:
        - MLSession
    - LoadBalancerPort: '8005'
      InstancePort: '8005'
      Protocol: HTTP
      PolicyNames:
        - MLSession
    - LoadBalancerPort: '8006'
      InstancePort: '8006'
      Protocol: HTTP
      PolicyNames:
        - MLSession
    - LoadBalancerPort: '8007'
      InstancePort: '8007'
      Protocol: HTTP
      PolicyNames:
        - MLSession
    - LoadBalancerPort: '8008'
      InstancePort: '8008'
      Protocol: HTTP
      PolicyNames:
        - MLSession
```

`HealthCheck` checks the health of each MarkLogic instance by contacting its HealthCheck App Server on port 7997 every number of seconds specified by `Interval`. Any answer other than "200 OK" within the `Timeout` period (in seconds) is considered unhealthy and that instance is removed from the ELB. For details on the HealthCheck parameters, see http://docs.aws.amazon.com/ElasticLoadBalancing/latest/APIReference/API_HealthCheck.html.

```
    HealthCheck:
      Target: 'HTTP:7997/'
      HealthyThreshold: '3'
      UnhealthyThreshold: '5'
      Interval: '10'
      Timeout: '5'
  Metadata:
    'AWS::CloudFormation::Designer':
      id: e188e71e-5f01-4816-896e-9bd30b9a96c1
```

The ALB declaration:

```
Alb:
  Condition: MultiZone
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  DependsOn:
    - VpcStack
    - ElbSecurityGroup
  Properties:
    SecurityGroups:
    - !Ref ElbSecurityGroup
    Subnets:
    - !GetAtt [VpcStack, Outputs.PublicSubnet1Id]
    - !If [MultiZone, !GetAtt [VpcStack, Outputs.PublicSubnet2Id], !Ref
'AWS::NoValue']
    - !If [MultiZone, !GetAtt [VpcStack, Outputs.PublicSubnet3Id], !Ref
'AWS::NoValue']
  Metadata:
    'AWS::CloudFormation::Designer':
    id: e188e71e-5f01-4816-896e-9bd30b9a96c1
```

ALB Target group section after the ALB declaration:

```
Description: #Descriptions of the 9 TargetGroups for MultiZone
deployments (3 zones). TargetGroups route requests to registered
targets.

#Health checks are performed on each TargetGroup.
AlbTargetGroup1:
Condition: MultiZone
Type: "AWS::ElasticLoadBalancingV2::TargetGroup"
DependsOn:
  - VpcStack
Properties:
  HealthCheckIntervalSeconds: 10
  HealthCheckTimeoutSeconds: 5
  HealthyThresholdCount: 3
  HealthCheckPort: 7997
  UnhealthyThresholdCount: 5
  Port: 8000
  Protocol: HTTP
  TargetGroupAttributes:
    - Key: stickiness.enabled
      Value: true
    - Key: stickiness.type
      Value: lb_cookie
    - Key: stickiness.lb_cookie.duration_seconds
      Value: 3600
    - Key: deregistration_delay.timeout_seconds
      Value: 60
VpcId: !GetAtt [VpcStack, Outputs.VpcId]
```

ALB Listener groups section after the ALB target groups:

```
Description:   #Descriptions of the 9 Listeners for MultiZone
deployments (3 zones). Each Listener connects Application Load Balancer
to a TargetGroup with a particular port.

AlbListener1:
  Condition: MultiZone
  Type: "AWS::ElasticLoadBalancingV2::Listener"
  DependsOn:
    - Alb
    - AlbTargetGroup1
  Properties:
    DefaultActions:
      - TargetGroupArn: !Ref AlbTargetGroup1
        Type: forward
    LoadBalancerArn: !Ref Alb
    Port: 8000
    Protocol: HTTP
```

3.7.6 Outputs Declaration

If the `CloudFormation` launch is successful, `Outputs` generates the URL of the ELB pointing to the MarkLogic Admin Interface port (8001).

```
Outputs:
  URL:
    Description: The URL of the MarkLogic Cluster
    Value: !Join
      - ''
      - - 'http://'
      - !If [MultiZone, !GetAtt [Alb, DNSName], !GetAtt
[ElasticLoadBalancer, DNSName]]
        ':8001'
        - !GetAtt
          - ElasticLoadBalancer
          - DNSName
        - ':8001'
  PrivateSubnetRouteTableID:
    Description: Private Subnet Route Table ID
    Value: !GetAtt [VpcStack, Outputs.PrivateSubnetRouteTableID]
  InstanceSecurityGroupID:
    Description: Instance Security Group ID
    Value: !Ref InstanceSecurityGroup
```

Note: The CloudFormation template supports both a Classic Load Balancer (CLB) and an Application Load Balancer (ALB).

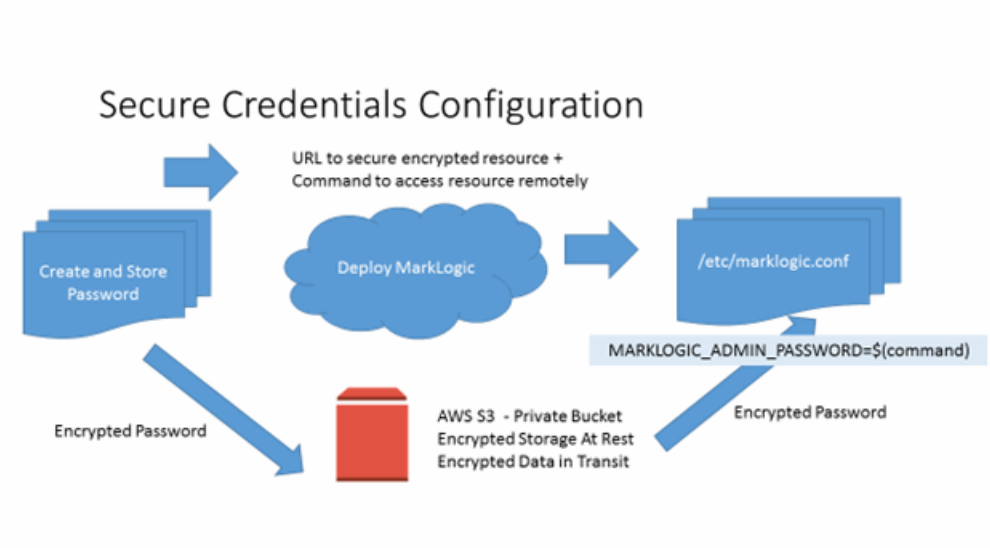
The CloudFormation template creates a different type of load balancer depending on the number of zones to which you deploy. The CloudFormation template will create a CLB if you deploy to *one zone*. The CloudFormation template will create an ALB if you deploy to *three zones*. If you are using your own CloudFormation template, you can deploy to two or more zones to create an ALB.

CLB runs at OSI layers 4 (transport) and 7 (application). ALB runs at OSI layer 7. If the CloudFormation template creates an ALB, you are unable to use an ODBC connection with business intelligence (BI) tools. To use an ODBC connection with BI tools, you can create a separate Network Load Balancer for ODBC connections.

3.8 Using CloudFormation with Secure Credentials

The sample templates are not designed for production environments. Most deployments will have specific infrastructure and integration requirements you will need to address. An important issue is how to manage secure credentials for MarkLogic in a automated "hands off" process. The sample templates pass the Admin Password in plain text as cloud formation parameters which then are converted into simple EC2 User Data name/value pairs. This is not a secure method of handling credentials.

As Mentioned in "Configuration using the /etc/marklogic.conf File" on page 38, an alternative to EC2 UserData is creating `/etc/marklogic.conf` during the deployment. This can be done in CloudFormation fairly easily. For Production deployments using CloudFormation, the `AWS::CloudFormation::Init` Resource (and the helper `cfn-init` commands) are recommended for deployment and configuration. See:
http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-init.html.



If not using CloudFormation the `cloud-init` service, the low-level API which CloudFormation uses, can be used directly. See http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html for details.

With the CloudInit resource, EC2 UserData is only used for a small 'bootstrap' script that accesses the configuration variables from the template metadata resource securely via `cfn-init`. By passing a reference to a secure channel for credentials instead of the credentials themselves, no confidential data is passed directly from the origin to the EC2 instance. This process is recommended by AWS and discussed in this posting:

http://blogs.aws.amazon.com/application-management/post/Tx3LKFZ27CWZBKO/Authenticated-File-Downloads-with-CloudFormation

There are many options for configuring the necessary authentication and providing a protected storage and access. Choosing the appropriate configurations is specific to your requirements and integration strategy and should be part of your overall IT and security planning. Integration MarkLogic deployment with CloudFormation or another orchestration requires only that the file /etc/marklogic.conf be created prior to the first startup of MarkLogic on that instance.

Below are snippets of the Launch Configuration and AutoScalingGroup sections from an example CloudFormation template that makes use of CloudInit and a secure S3 bucket for the admin password. Note that the URL itself for the S3 file does not need to be confidential, so it may be safely passed as a CloudFormation parameter and stored for the lifetime of the instance. In the Launch Configuration, a simple script is used to invoke `cfn-init`, passing a reference to the MetaData resource associated with the AutoScalingGroup for a zone. The MetaData resource is a sibling of the "Properties" tag in the AutoScalingGroup section.

The "files" entry in the AutoScalingGroup section writes `/etc/marklogic.conf` with the root owner and group (read-only by owner).

The "services" entry in the AutoScalingGroup section starts MarkLogic after CloudInit is complete and restarts it if `/etc/marklogic.conf` or `/etc/sysconfig/MarkLogic` is updated by CloudInit in the future.

Example Launch Configuration Snippet:

```
"LaunchConfig1" : {
      "Type" : "AWS::AutoScaling::LaunchConfiguration",
      "Properties" : {
         .... },
"UserData": {"Fn::Base64": {"Fn::Join": [
  "",
   [
    "#!/bin/bash\n",
    "function error_exit\n",
    "{\n",
    "logger -t MarkLogic  \"$1\"",
    "exit 1\n",
    "}\n",
    "yum update -y aws-cfn-bootstrap\n",
    "yum update -y\n",
    "# Install application\n",
    "/opt/aws/bin/cfn-init -v -s ",
```

```
            {"Ref": "AWS::StackId"}, " -r ASG1  --region ",
            {"Ref": "AWS::Region"}, " || error_exit 'Failed to run cfn-
        init'\n",
            "\n",
            "# All is well so signal success\n",
            "\n"
          ]
        ]}}}
```

Example AutoScalingGroup Snippet:

```
"ASG1" : {
       "Type" : "AWS::AutoScaling::AutoScalingGroup",
       "Properties" : {                .....
     },
  "Metadata": {
  "MarkLogic::MetaDataVersion": "2015-07-17-14:49:23",
  "AWS::CloudFormation::Init": {
    "config": {
      "files": {"/etc/marklogic.conf": {
        "content": {"Fn::Join": [
          "",
          [
          "MARKLOGIC_CLUSTER_NAME=",{"Ref": "MarkLogicDDBTable"}, "\n",
          "MARKLOGIC_EBS_VOLUME=", {"Ref": "MarkLogicVolume1"}, "\n",
          "MARKLOGIC_NODE_NAME=NodeA#\n",
          "MARKLOGIC_ADMIN_USERNAME=",{"Ref": "AdminUser"},"\n",
          "# Password obtained via protected S3 file\n",
          "# MARKLOGIC_ADMIN_PASSWORD=\n",
          "# $(s3 cp --region us-west-2 s3://bucket/secret-password - ) \n",
          "MARKLOGIC_ADMIN_PASSWORD=$( aws s3 --region ",
          {"Ref": "AWS::Region"}, " cp ", {"Ref": "AdminPassS3URL"}, " - )\n",
          "MARKLOGIC_CLUSTER_MASTER=0\n"
          ] ]} ,
        "mode": "000400",
        "owner": "root",
        "group": "root"
      }},
      "services": {"sysvinit":
    {"MarkLogic": {
        "enabled": "true",
        "ensureRunning": "true",
        "files": [
          "/etc/marklogic.conf",
          "/etc/sysconfig/MarkLogic"
        ]  }
  }}
}}
```

3.9 Deleting a CloudFormation Stack

To delete a CloudFormation stack, follow the procedure described in http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-delete-stack.html.

Deleting your CloudFormation stack removes most of the EC2 resources (instances, security groups, etc.) created by your CloudFormation template. The exception is that the EBS volumes are not removed. Should you want to remove the EBS volumes after deleting your stack, you must manually remove them by following the procedure described in http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-deleting-volume.html.

Note: