MarkLogic Server

MarkLogic Server on Amazon EC2 Guide

MarkLogic 8
February, 2015

Last Revised: 8.0-4, November, 2015

Copyright © 2015 MarkLogic Corporation. All rights reserved.

Table of Contents

MarkLogic Server on Amazon EC2 Guide

1.0 Overview of MarkLogic Server on EC2 3
- 1.1 STOP: Before you do Anything! 3
- 1.2 Understanding MarkLogic Server for EC2 3
  - 1.2.1 Amazon EC2 Terminology 4
  - 1.2.2 The Managed Cluster Feature 6
  - 1.2.3 Launching a MarkLogic AMI outside of CloudFormation 8
- 1.3 HealthCheck App Server 8
- 1.4 Typical Architecture 9

2.0 Getting Started with MarkLogic Server on EC2 12
- 2.1 Security 12
- 2.2 Summary of Deployment Procedures 13
- 2.3 Creating an Amazon EC2 Account 13
- 2.4 Enabling a MarkLogic Server for EC2 AMI 14
- 2.5 Initial Setup Procedures 14
  - 2.5.1 Accessing the AWS Management Console 15
  - 2.5.2 Creating an IAM Role 16
  - 2.5.3 Creating a Key Pair 21
  - 2.5.4 Creating a Simple Notification Service (SNS) Topic 21
- 2.6 AWS Configuration Variables 24
- 2.7 EC2 User Data 28
- 2.8 Configuration using the /etc/marklogic.conf File 28
- 2.9 Other Configuration Methods 29
- 2.10 Configuration Security Considerations 29

3.0 Deploying MarkLogic on EC2 Using CloudFormation 31
- 3.1 What CloudFormation Template Version to Use 31
- 3.2 Overview 32
- 3.3 Deployment and Startup 34
- 3.4 Creating a CloudFormation Stack using the AWS Console 38
- 3.5 Creating a CloudFormation Stack using the AWS Command-Line Interface 47
- 3.6 Sample CloudFormation Template 47
  - 3.6.1 Parameters Declaration 48
  - 3.6.2 Mappings Declaration 52
  - 3.6.3 Resources Declaration 55
  - 3.6.4 Outputs Declaration 64
- 3.7 Using CloudFormation with Secure Credentials 64
- 3.8 Deleting a CloudFormation Stack 68

4.0 Managing MarkLogic Server on EC2 69
4.1 Accessing a MarkLogic Server Instance 69
4.1.1 Accessing MarkLogic Server through the ELB 70
4.1.2 Accessing MarkLogic Server through the Instance Public DNS 71
4.2 Accessing an EC2 Instance 72
4.3 Detecting EC2 Errors 74
4.4 Using the mlcmd Script 74
4.4.1 sync-volumes-from-mdb 75
4.4.2 sync-volumes-to-mdb 75
4.4.3 init-volumes-from-system 76
4.4.4 leave-cluster 77
4.5 Configuring MarkLogic for Amazon Simple Storage Service (S3) 77
4.5.1 Set up an S3 Bucket 78
4.5.2 Configure the S3 Endpoint for your Group 78
4.5.3 Configure S3 Credentials 79
4.5.3.1 Configuring S3 Credentials in the Security Database 79
4.5.3.2 Configuring S3 Credentials in Environment Variables 80
4.5.3.3 Configuring an IAM Role with an S3 Access Policy 80
4.5.4 Set an S3 Path in Forest Data Directory 81
4.5.5 Load Content into MarkLogic to Test 82
4.6 Scaling Cluster Resources on EC2 82
4.7 Upgrading the MarkLogic AMI 83
4.8 Monitoring (CloudWatch) 85
4.9 Migrating from Enterprise Data Center to EC2 86
4.10 Creating an EBS Volume and Attaching it to an Instance 86
4.10.1 Creating and EBS Volume 86
4.10.2 Attaching an EBS Volume to an Instance 88
4.11 Pausing or Terminating a MarkLogic Cluster 89

5.0 Technical Support 90

6.0 Copyright 91
6.0 NOTICE 91

1.0 Overview of MarkLogic Server on EC2

This chapter provides an overview of MarkLogic Server on Amazon Elastic Compute Cloud (EC2) using a MarkLogic Amazon Machine Image (AMI), as well as how to create an Amazon EC2 account and order a MarkLogic Server for EC2 AMI. This chapter includes the following sections:

- STOP: Before you do Anything!
- Understanding MarkLogic Server for EC2
- HealthCheck App Server
- Typical Architecture

For more detailed information on Amazon EC2, 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 31. The MarkLogic CloudFormation templates are available from http://developer.marklogic.com/products/aws.

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 6. 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 8.

1.2 Understanding MarkLogic Server for EC2

MarkLogic provides pre-packaged AMIs containing Amazon Linux 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:

- Amazon EC2 Terminology
- The Managed Cluster Feature
- Launching a MarkLogic AMI outside of CloudFormation

1.2.1 Amazon EC2 Terminology

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

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

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

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.

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 AMI and have been pre-installed with MarkLogic and the necessary dependancies.

Type	Pricing
Production	Per-hour EC2 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 one 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.

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

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 77 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 31 and http://aws.amazon.com/cloudformation/. The MarkLogic CloudFormation templates are available from http://developer.marklogic.com/products/aws.

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 6.

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 *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 38.

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.

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.

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.
- Pausing 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 pause 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 21. 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 associate it to the EC2 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`.

You may now go to the admin page on port 8001 and continue configuring your MarkLogic instance.

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 Typical Architecture

This section describes some of the typical configurations of a MarkLogic cluster in an EC2 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.

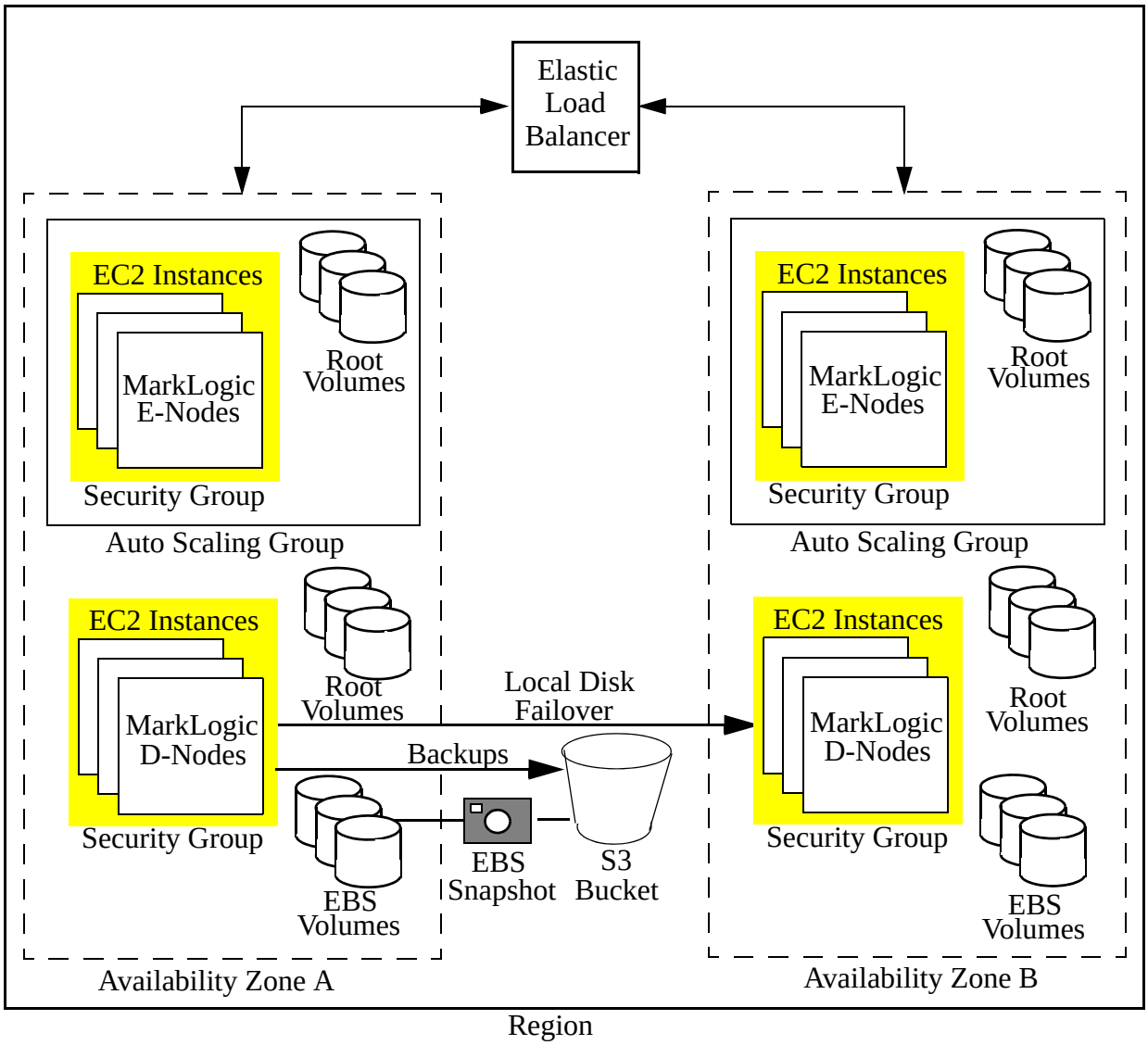
End-user or app-level queries should to be routed to the E-Nodes through a load balancer. You can add E-Nodes to scale up a cluster to handle more queries, more users and more computation.

To ensure high availability, place D-Nodes in different availability zones in the same region and configure them for local-disk failover to ensure that each transaction is written to one or more replicas. In EC2, the latency between zones in same region is low (approximately two milliseconds). For optimum availability, D-Nodes and E-Nodes should be split evenly between two availability zones. For disaster recovery, you can place D-Nodes in different regions and use database replication between the D-Nodes in each region.

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 large sequential IO profile and also aligns to Amazon's pIOPS implementation. Each configured forest on MarkLogic requires a minimum of 20mb/sec. 20mb/sec with 16K blocks is 1280 IOPS. Each instance of MarkLogic Server should be configured with a maximum of 5 Hi-IO volumes/forests. Additional EBS for boot and low-IO forests, such as those used by the Security and Schema databases, can be added. Once the forests are on provisioned IOPS volumes, they can be migrated during maintenance periods or via scripted migration while running using the replicas.

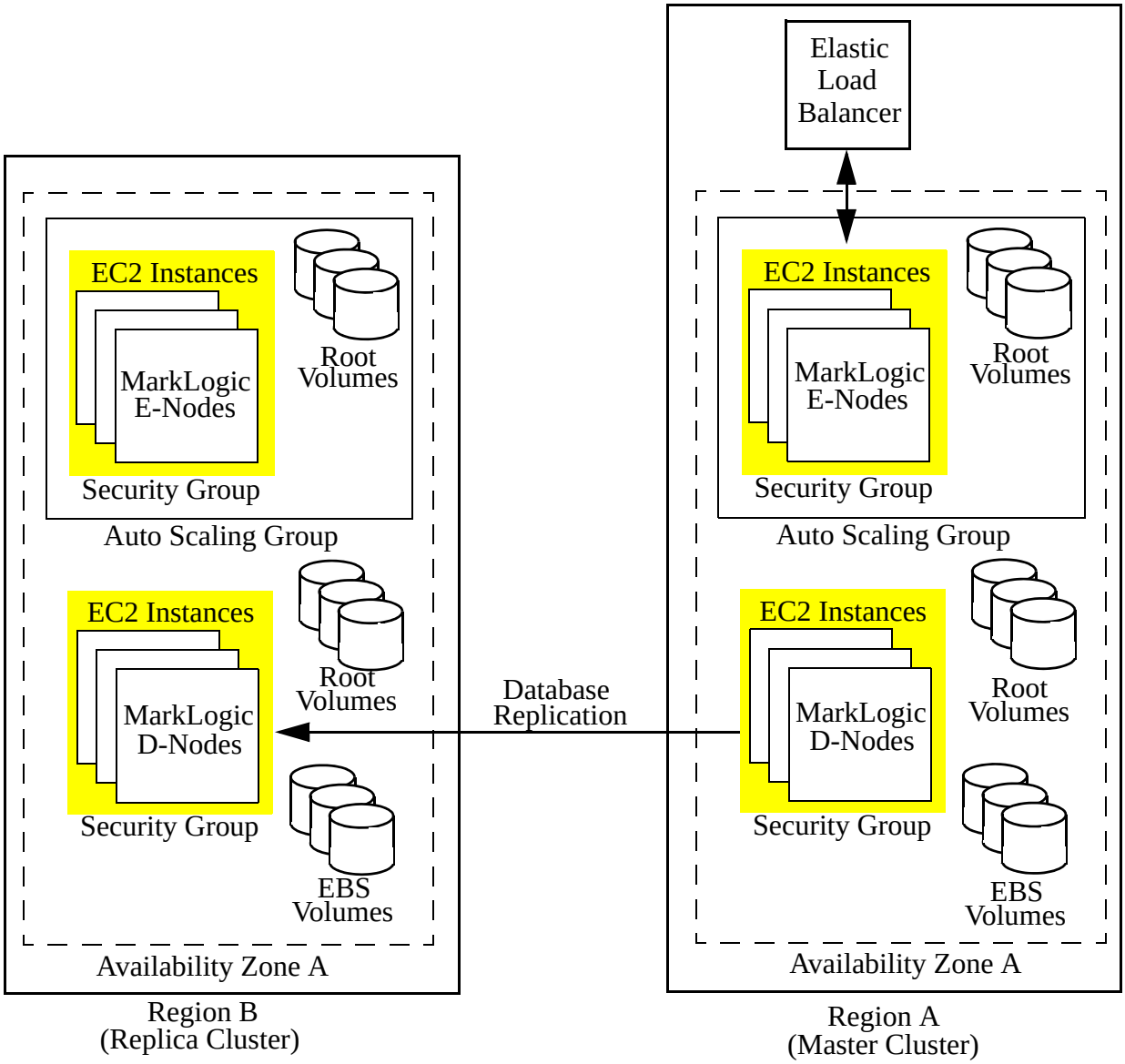
The hi1.4xlarge instance offers the fastest performance. MarkLogic can take advantage of SSD volumes to accelerate all operations. However, because SSD volumes are ephemeral, database replication is required. Alternatively, m2.4xlarge offers a large amount of RAM and relatively high IO along with 1GB/sec IO guaranteed. m1.xlarge also offers that IO but with only 15GB/RAM, much less cache.

The illustration below shows a possible architecture, where there are two clusters on different zones in the same region. The D-nodes in each cluster are configured for local disk failover. Forest data is stored in IOPS EBS volumes and backups and snapshots of EBS volumes are stored in S3.



The illustration below shows a possible architecture, where two clusters are configured with database replication and deployed in different regions. This type of architecture works with Amazon Route 53, which is a scalable Domain Name System (DNS) web service that provides secure and reliable routing of user traffic to your MarkLogic cluster. Should the Master cluster fail, Route 53 can redirect traffic to the Replica cluster in seconds. The state of the Replica will lag somewhat from the Master (5-20 seconds is typical).

For details on Amazon Route 53, see
http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html.



2.0 Getting Started with MarkLogic Server on EC2

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

- Security
- Summary of Deployment Procedures
- Creating an Amazon EC2 Account
- Enabling a MarkLogic Server for EC2 AMI
- Initial Setup Procedures
- AWS Configuration Variables
- EC2 User Data
- Configuration using the /etc/marklogic.conf File
- Other Configuration Methods
- Configuration Security Considerations

2.1 Security

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

- Key Pairs, as described in “Creating a Key Pair” on page 21.
- 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 31.

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.

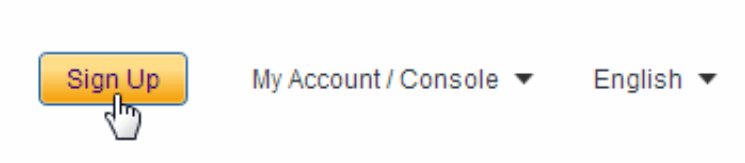
2.2 Summary of Deployment Procedures

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 Amazon EC2 account, create one.	"Creating an Amazon EC2 Account" on page 13
Enable a MarkLogic Server AMI.	"Enabling a MarkLogic Server for EC2 AMI" on page 14
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 21
Create a Simple Notification Service (SNS) Topic.	"Creating a Simple Notification Service (SNS) Topic" on page 21
Create CloudFormation stack from a CloudFormation template.	"Deploying MarkLogic on EC2 Using CloudFormation" on page 31
Open the MarkLogic Server Admin interface.	"Accessing a MarkLogic Server Instance" on page 69

2.3 Creating an Amazon EC2 Account

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



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.

2.4 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.

2.5 Initial Setup Procedures

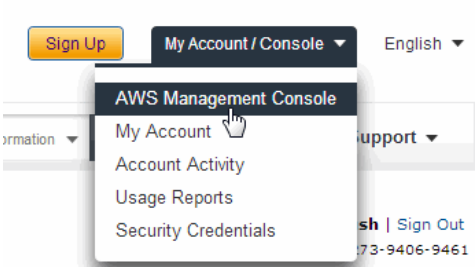
This section describes how to access the AWS management console and create a security group and key pair. Typically, you will create your security groups and key pairs once and reuse them for each instance you create. The topics in this section are:

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

2.5.1 Accessing the AWS Management Console

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

1. Log into your Amazon EC2 account and from the My Account/Console pull-down menu, select AWS Management Console:



2. Click Sign into the AWS Management Console and enter your email address and password for your EC2 account:

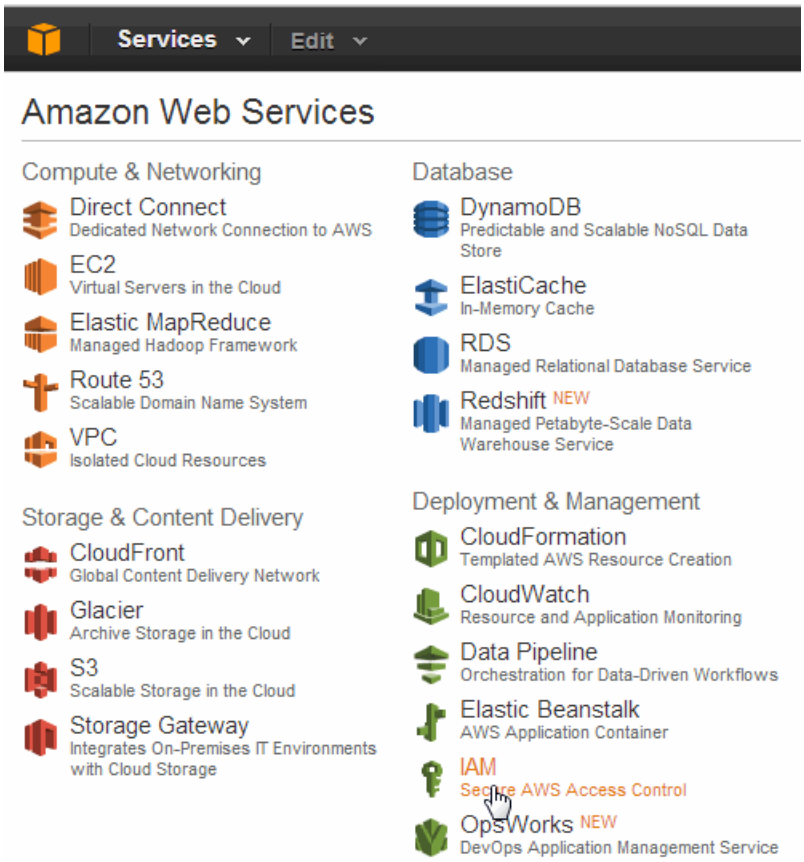


2.5.2 Creating an IAM Role

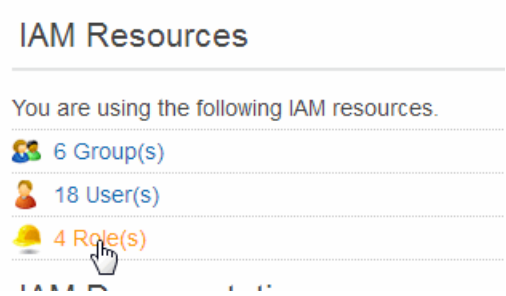
AWS Identity and Access Management (IAM) is a web service that enables you to manage users and user permissions in AWS. The service is targeted at organizations with multiple users or systems that use Amazon EC2, Amazon DynamoDB, and the AWS Management Console. With IAM, you can centrally manage users,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

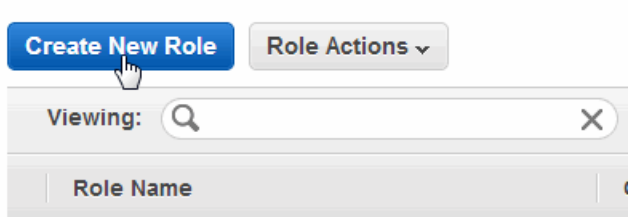
1. In the Amazon Web Services page, click on IAM:



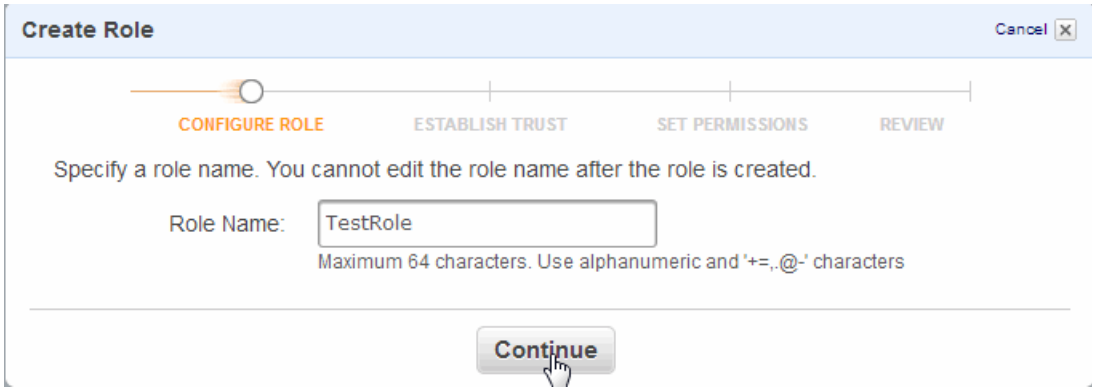
2. In the IAM Resources section of the Getting Started page, click Roles:



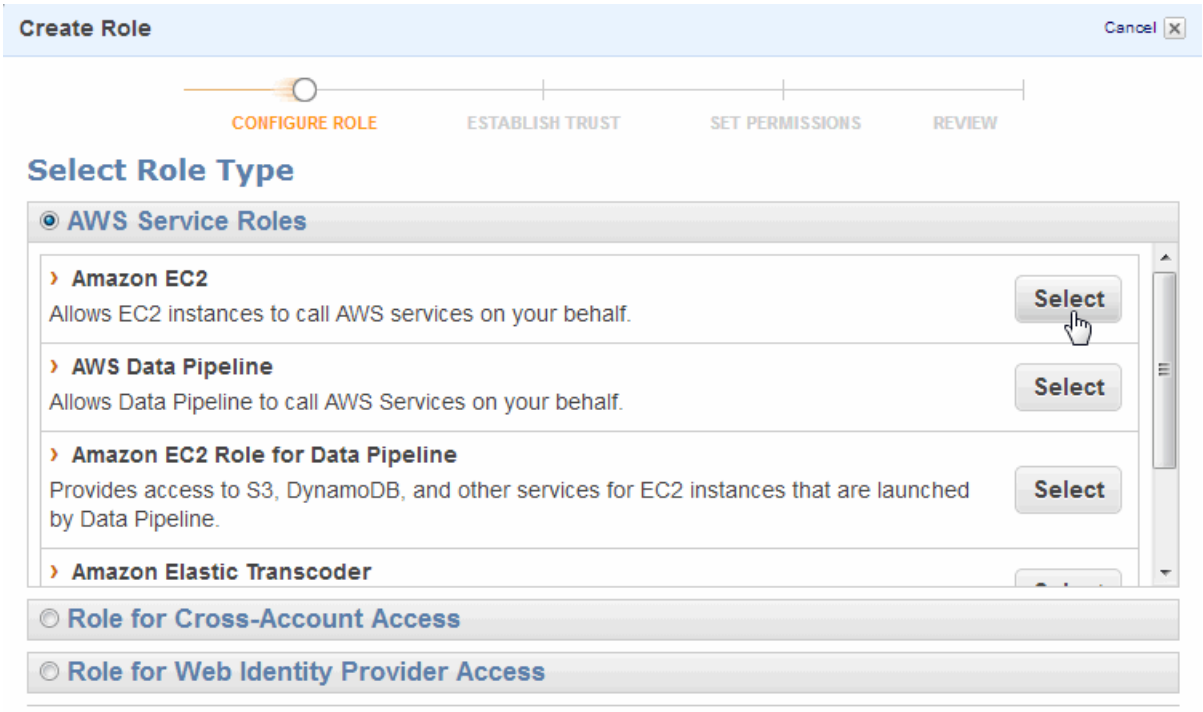
3. In the Roles page, click Create New Role:



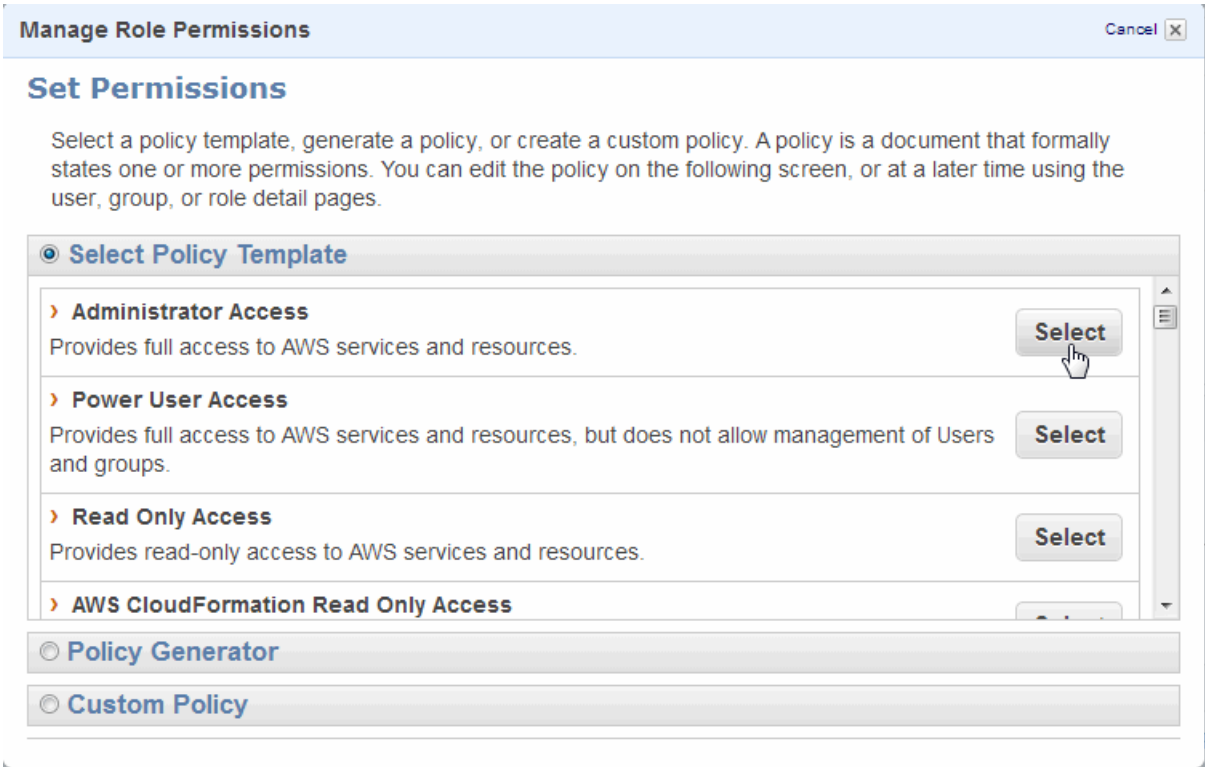
4. In the Create Role window, enter the name of the new role:



5. In the Configure Role window, select Amazon EC2:



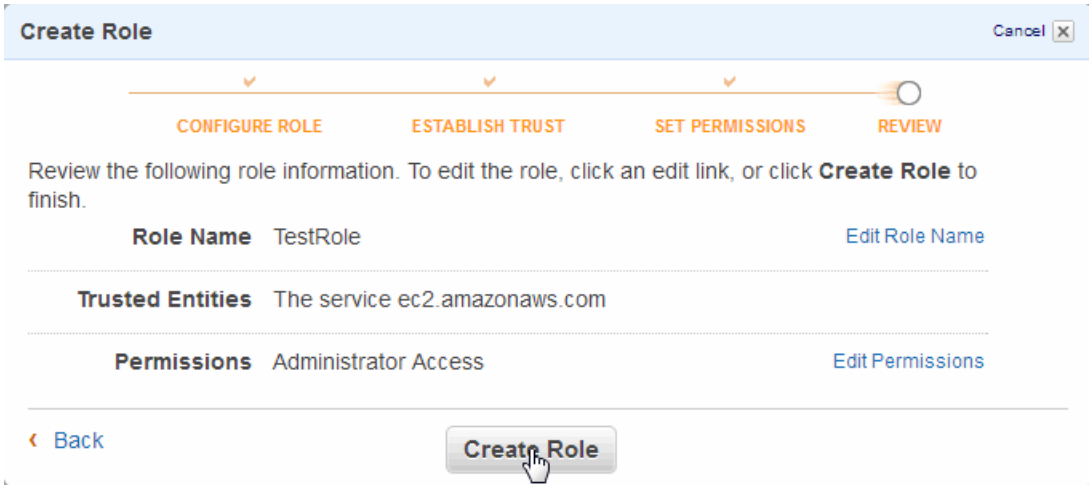
6. In the Set Permissions window, select the access policy for the role. For details on IAM policies, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-policies-for-amazon-ec2.html.



7. Edit the permissions for your selected policy, as described in http://docs.aws.amazon.com/IAM/latest/UserGuide/PoliciesOverview.html. When done, click Continue.



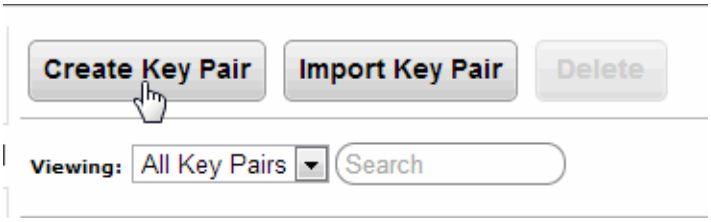
8. In the Review window, review your settings and edit if you want to make changes. When done, click Create Role.



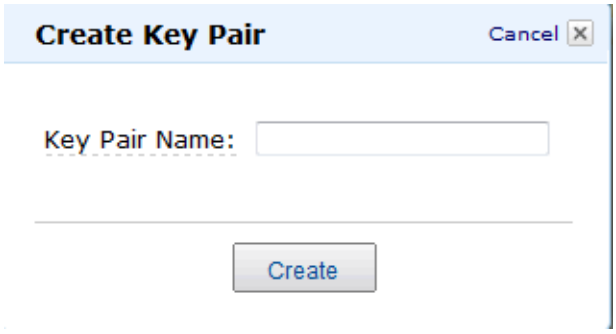
2.5.3 Creating a Key Pair

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 Management Console, select Key Pairs from the left-hand navigation section and click Create Key Pair in the Key Pairs page:



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



3. 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 72.

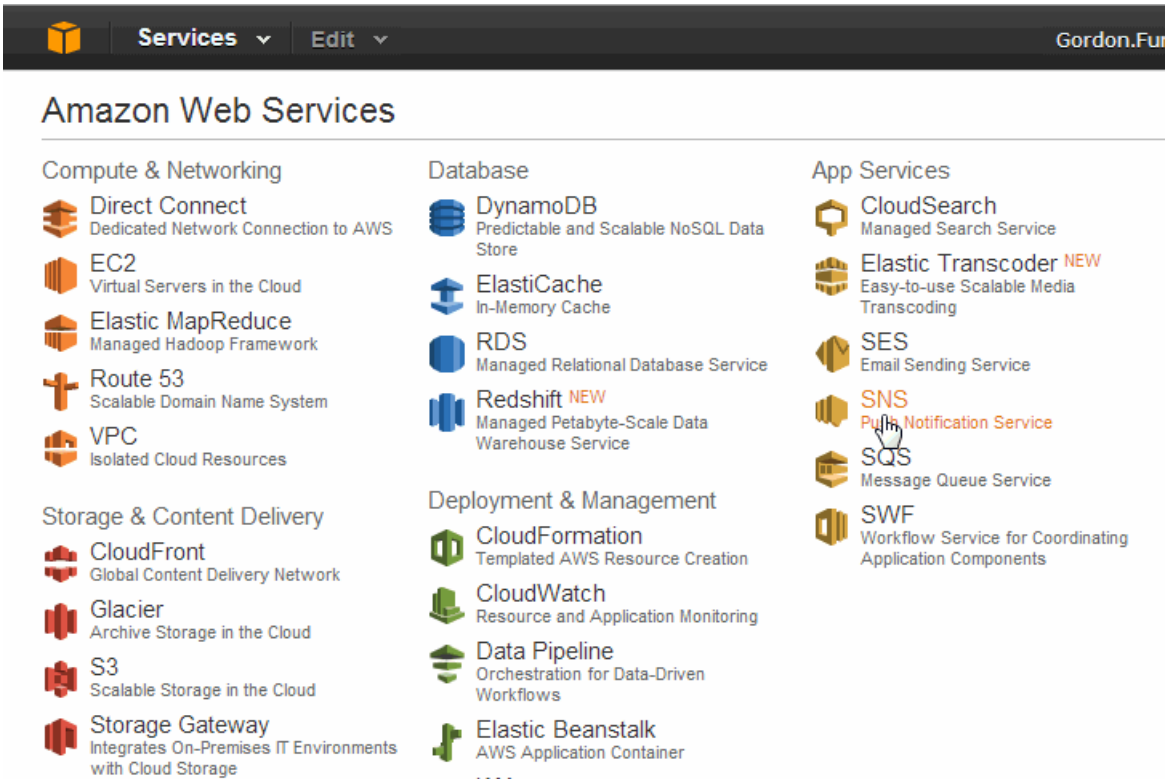
2.5.4 Creating a Simple Notification Service (SNS) Topic

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.

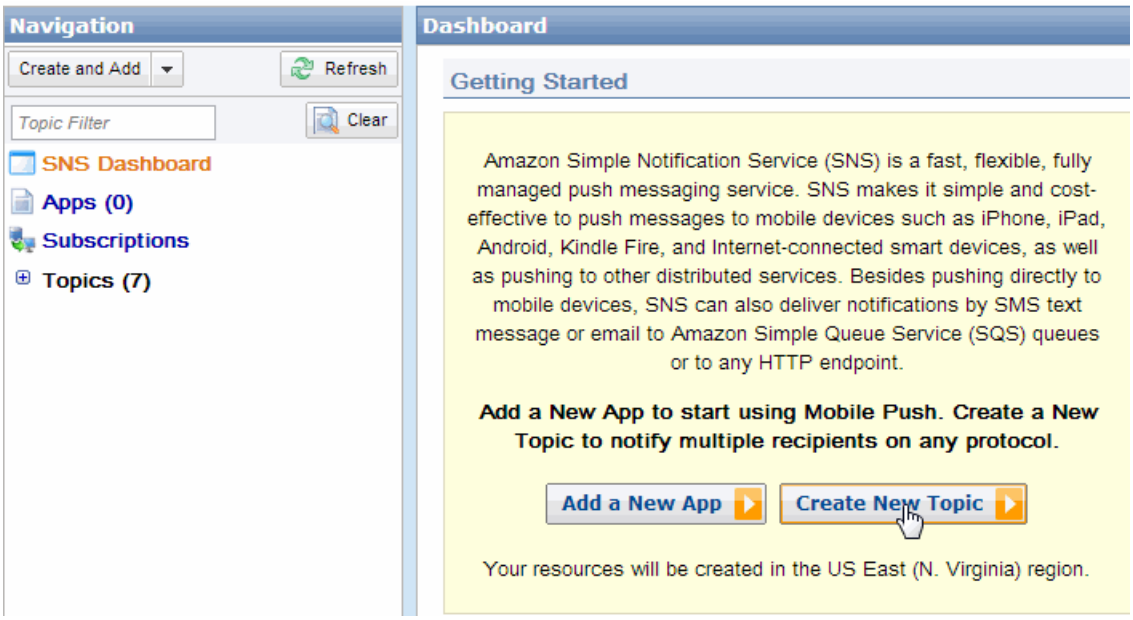
For details on the SQS queue system and creating an SNS topic, see http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqssubscribe.html.

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

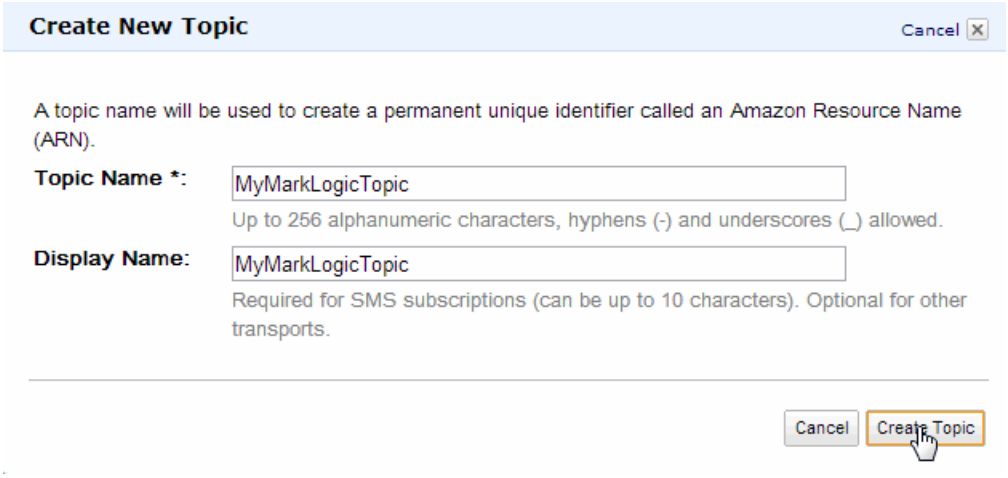
1. Open the SNS Dashboard from the Amazon Web Services menu.



2. Click Create New Topic.

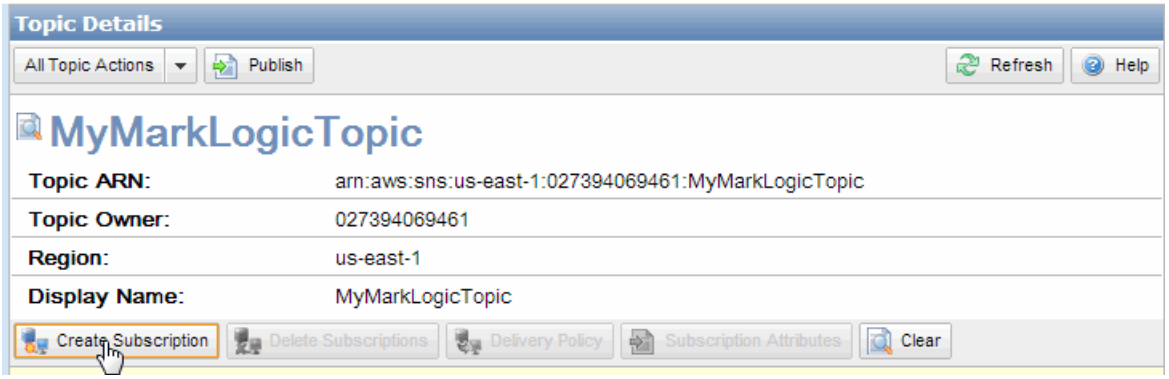


3. Enter a Topic Name and an optional Display Name. Click Create Topic.



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 38.

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.



2.6 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 specific configuration, startup and management features are disabled. (The rest of the variables below are unused.)

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

- 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 use for all AWS services on this instance. Use of this key is discouraged because passing AWS credentials to the cluster is less secure. For better security, use an IAM Role associated with the EC2 instance.
- MARKLOGIC_AWS_SECRET_KEY — An AWS Secret Key to use for all AWS services on this instance. Use of this key is discouraged because passing AWS credentials to the cluster is less secure. For better security, use an IAM Role associated with the EC2 instance.
- 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 `/etc/init.d` in “Deployment and Startup” on page 34.
- 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 29 .

- 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]]:[encrypted]

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"
encrypted	Use EBS encryption at rest

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-1:1234567890123456:mytopic`.

- 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."

2.7 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 (`/etc/init.d/MarkLogic`), 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_.

2.8 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 24, 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 34.

As described in “AWS Configuration Variables” on page 24, 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 29 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.

2.9 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 `/etc/init.d/MarkLogic` directly need to have the same environment as the init environment.

2.10 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 34. 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 retieved 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: http://docs.aws.amazon.com/cli/latest/reference/s3/index.html.

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

```
MARKLOGIC_CLUSTER_NAME=JOE-CFN-JOESecure5x-MarkLogicDDBTable-164OK8LD6ARMY
MARKLOGIC_EBS_VOLUME=vol-1111111
MARKLOGIC_NODE_NAME=NodeA#
MARKLOGIC_ADMIN_USERNAME=admin
##
MARKLOGIC_ADMIN_PASSWORD=\
  $(aws s3 --region us-east-1 cp s3://marklogic.joesbucket/secret-password - )
##
MARKLOGIC_CLUSTER_MASTER=1
MARKLOGIC_LICENSEE=none
MARKLOGIC_LICENSE_KEY=none
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 above 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 should be identical:

```
# ... Same as Zone  except for ...
MARKLOGIC_EBS_VOLUME=vol-2222222
MARKLOGIC_NODE_NAME=NodeB#
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 64.

3.0 Deploying MarkLogic on EC2 Using CloudFormation

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

- What CloudFormation Template Version to Use
- Overview
- Deployment and Startup
- Creating a CloudFormation Stack using the AWS Console
- Creating a CloudFormation Stack using the AWS Command-Line Interface
- Sample CloudFormation Template
- Using CloudFormation with Secure Credentials
- Deleting a CloudFormation Stack

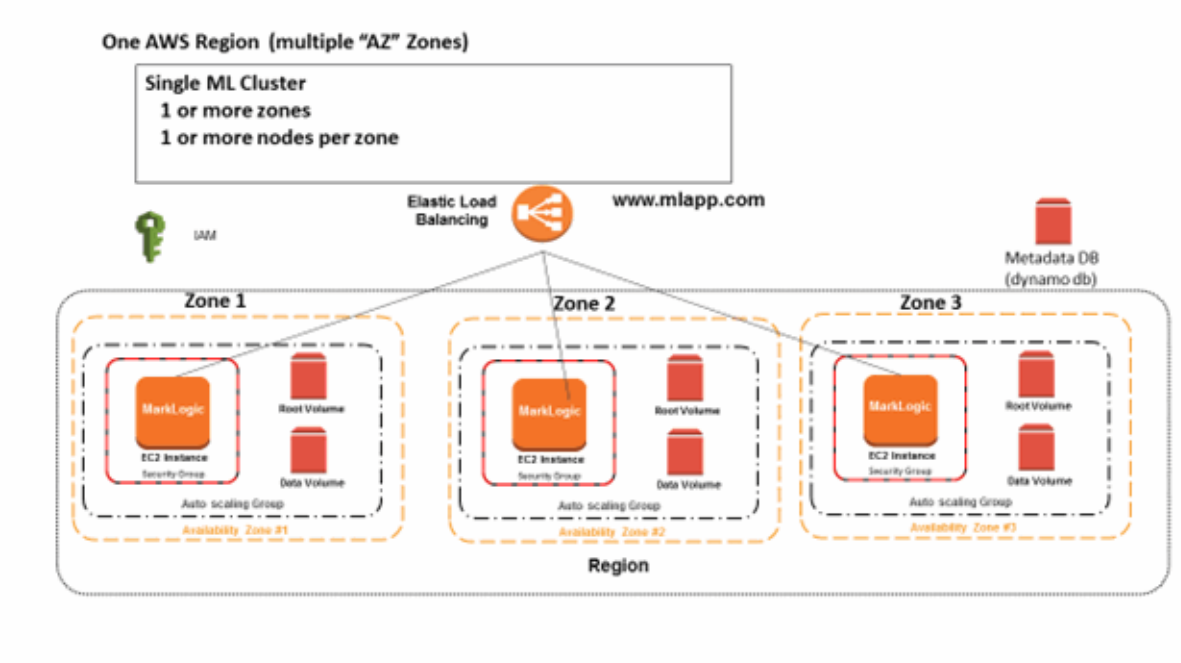
3.1 What CloudFormation Template Version to Use

There are two basic versions of the MarkLogic CloudFormation Template, Version 1 and Version 2. The basic difference between the two template versions is the database used to store the metadata:

- Version 1.0 — This version stores the metadata in SimpleDB. Version 1 is compatible with MarkLogic 7.x to 8.0.1 for new installs. Use this version if updating an existing EC2 cluster, as described in “Upgrading the MarkLogic AMI” on page 83.
- Version 2.0 — This version stores the metadata in DynamoDB. Version 2 is compatible with MarkLogic 8.0.3+ , but not earlier releases.

3.2 Overview

A Managed Cluster is automatically initialized and pre-configured with recommended topology, such as the one illustrated below.



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 pause the cluster by setting the ASG `NodesPerZone` value to 0 for all nodes. You can then restart the node by resetting the `NodesPerZone` to a value of 1 - 20 for each ASG. On restart, either by resuming from pause 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 `/etc/init.d`) 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
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. `/etc/init.d/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. If MARKLOGIC_EC2_USERDATA != 0, then EC2 user data is read and parsed. Any name/value pairs overwrite existing settings.

10. If MARKLOGIC_CLUSTER_NAME, MARKLOGIC_NODENAME and MARKOGIC_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.

11. If Step 10 is performed, all resolved variables are cached by writing to `/usr/local/mlcmd.conf` to avoid the overhead of recalculating the values on a restart.

12. If Step 10 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:

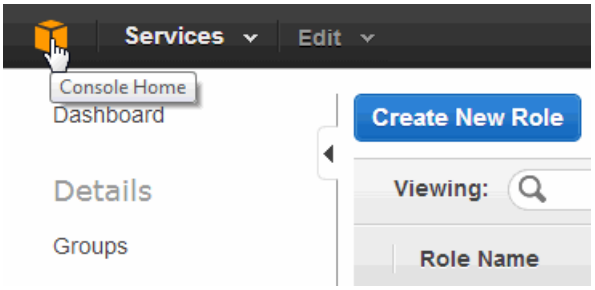
http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-create-stack.html.

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

- Purchase an AMI from Amazon MarketPlace or create your own AMI (ask customer support for assistance). 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 “Sample CloudFormation Template” on page 47. The MarkLogic CloudFormation templates are available from http://developer.marklogic.com/products/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 21
- An SNS Topic, as described in “Creating a Simple Notification Service (SNS) Topic” on page 21
- Optional: An SNS Topic, as described in “Creating a Simple Notification Service (SNS) Topic” on page 21.

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

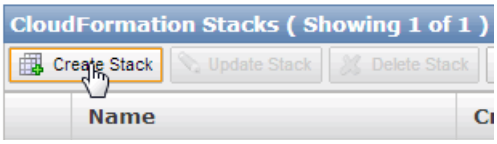
1. Click on the box in the upper left-hand portion of the Services page to access the Amazon Web Services home page:



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

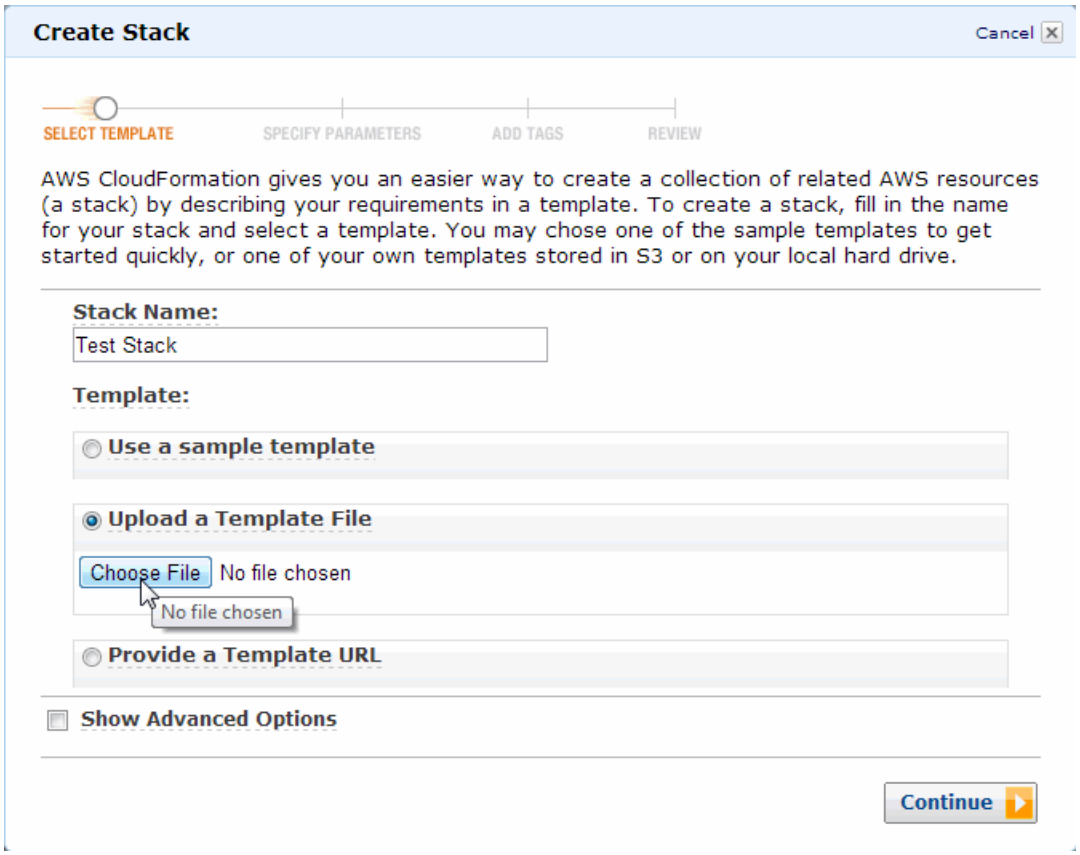


3. In the CloudFormation Stacks page, click Create Stack.

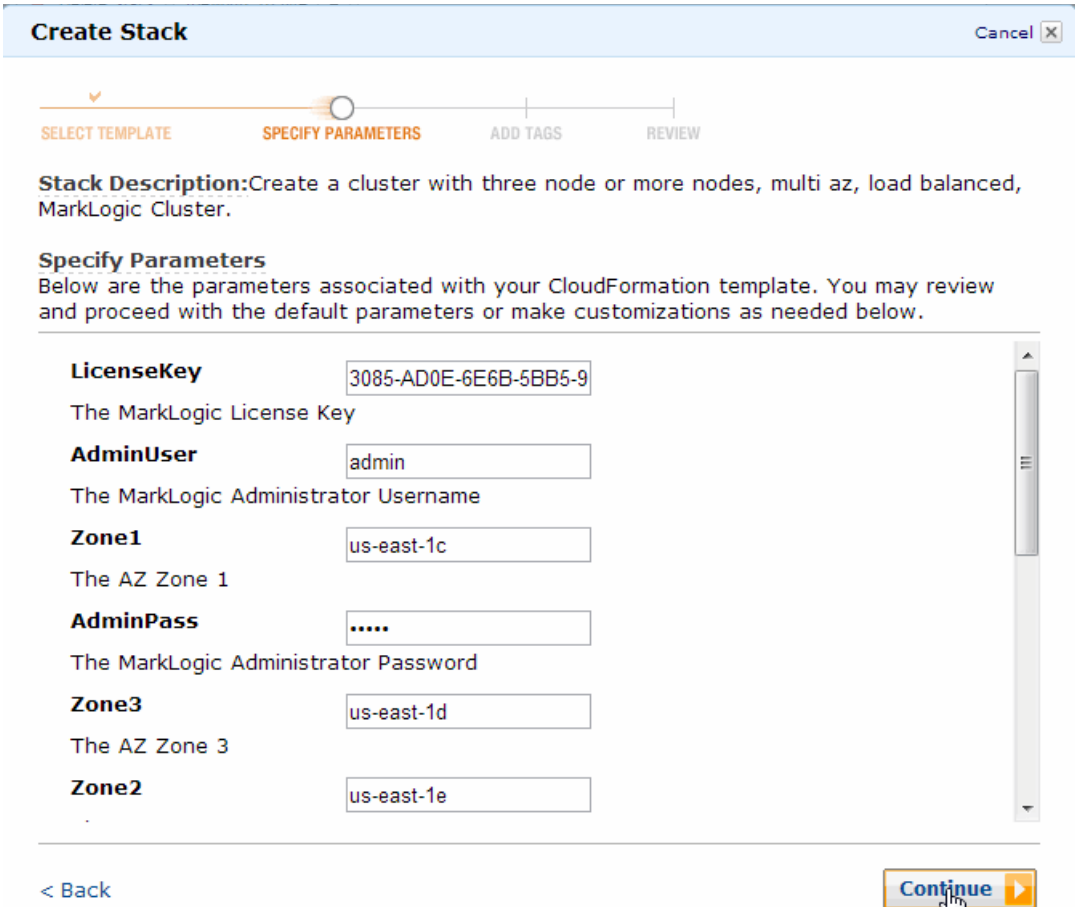


4. In the Select Template window, enter a name for your stack. Click Choose File and select the CloudFormation script file you created in “Sample CloudFormation Template” on page 47. Alternatively, you can click Provide a Template URL and enter one of the URLs listed in http://developer.marklogic.com/products/aws. When done, click Continue.

   Note: 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.



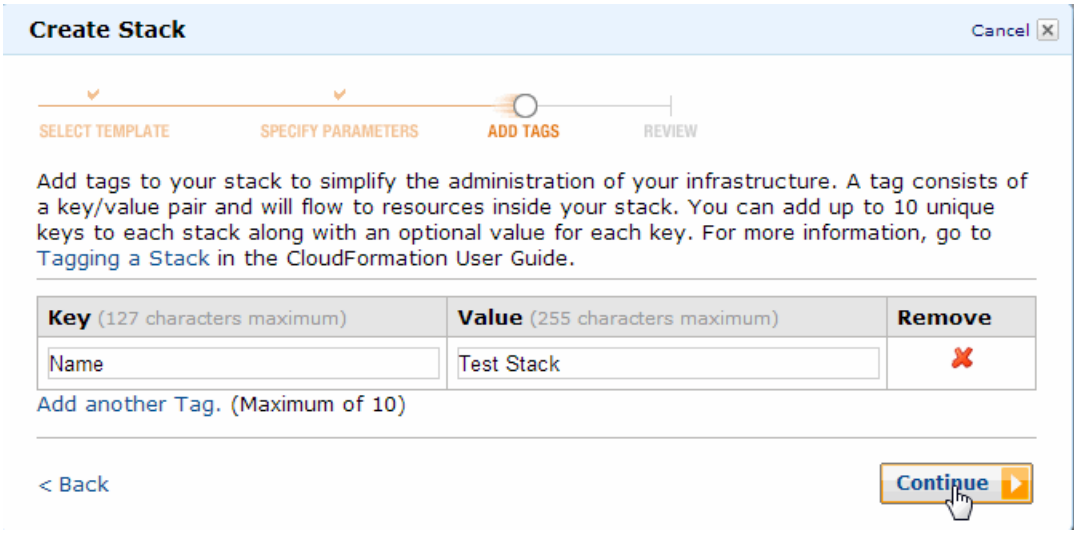
5. In the Specify Parameters window, enter the information shown in the table below. When done, click Continue.



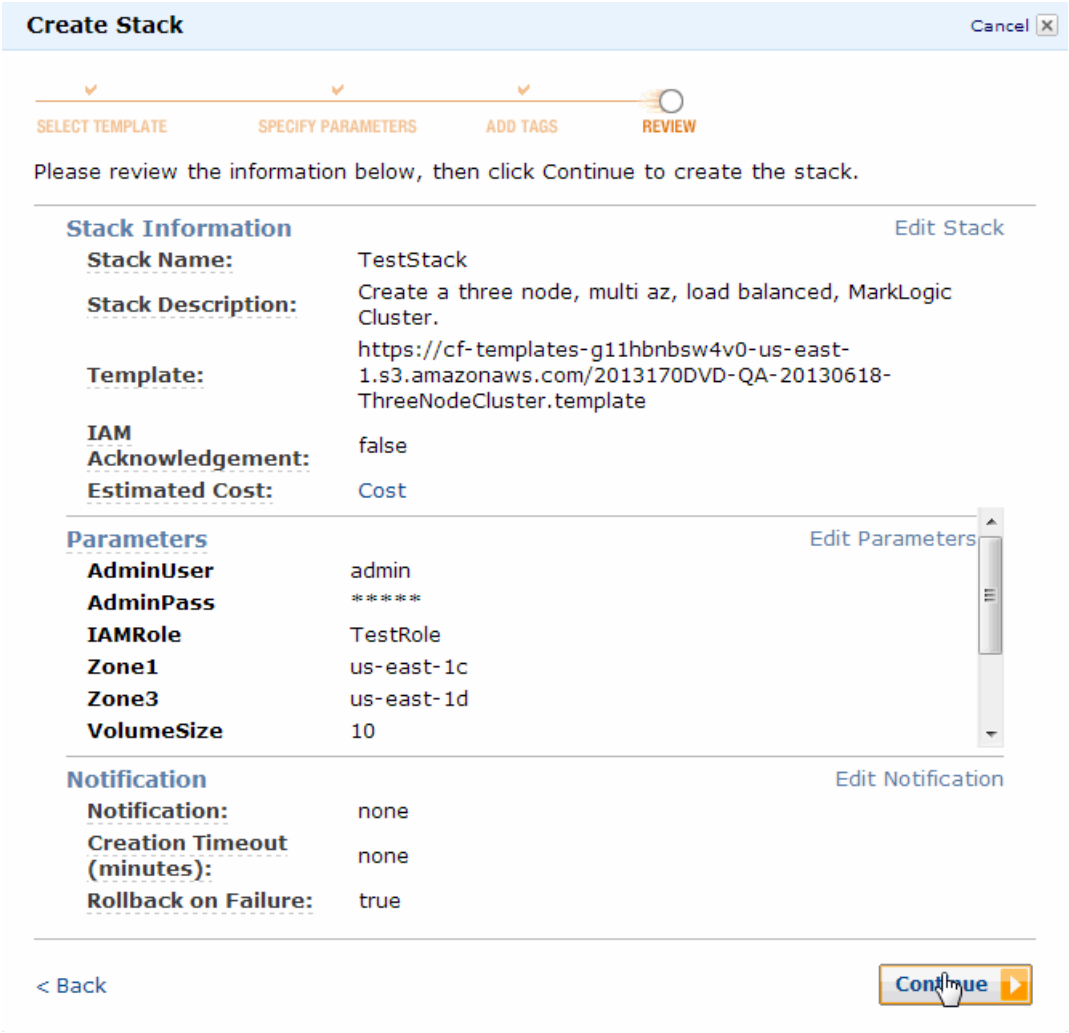
Parameter Name	Description
AdminUser	The username you want to use to log in as the MarkLogic Administrator.
AdminPass	The password you want to use to log in as the MarkLogic Administrator.
IAMRole	The name of the IAM Role you created in “Creating an IAM Role” on page 16.

Parameter Name	Description
InstanceType	The type of EC2 instance to launch. These vary by release, product type, zone, region, and availability. Refer to http://developer.marklogic.com/products/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.
KeyName	The name of the Key Pair you created in “Creating a Key Pair” on page 21.
Licensee	The name of the licensee obtained from your MarkLogic representative. Enter `none` if you plan to enter the license information later.
LicenseKey	The license key obtained from your MarkLogic representative. Enter `none` if you plan to enter the license information later.
LogSNS	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 21.
NodesPerZone	The number of nodes (hosts) to create for each zone. 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 pause all nodes.
SpotPrice	Spot price for instances in USD/Hour. This is 0 by default. If not 0, then the amount given is a spot request for the instances is used instead of on-demand.
VolumeSize	The initial EBS volume size (GB). The range of valid values are 10 - 1000. The default is 10.
Zone1 Zone2 Zone3	The zones on which to create each host in the cluster. Each zone in your cluster should be in the same region, such as `us-east` or `us-west`.

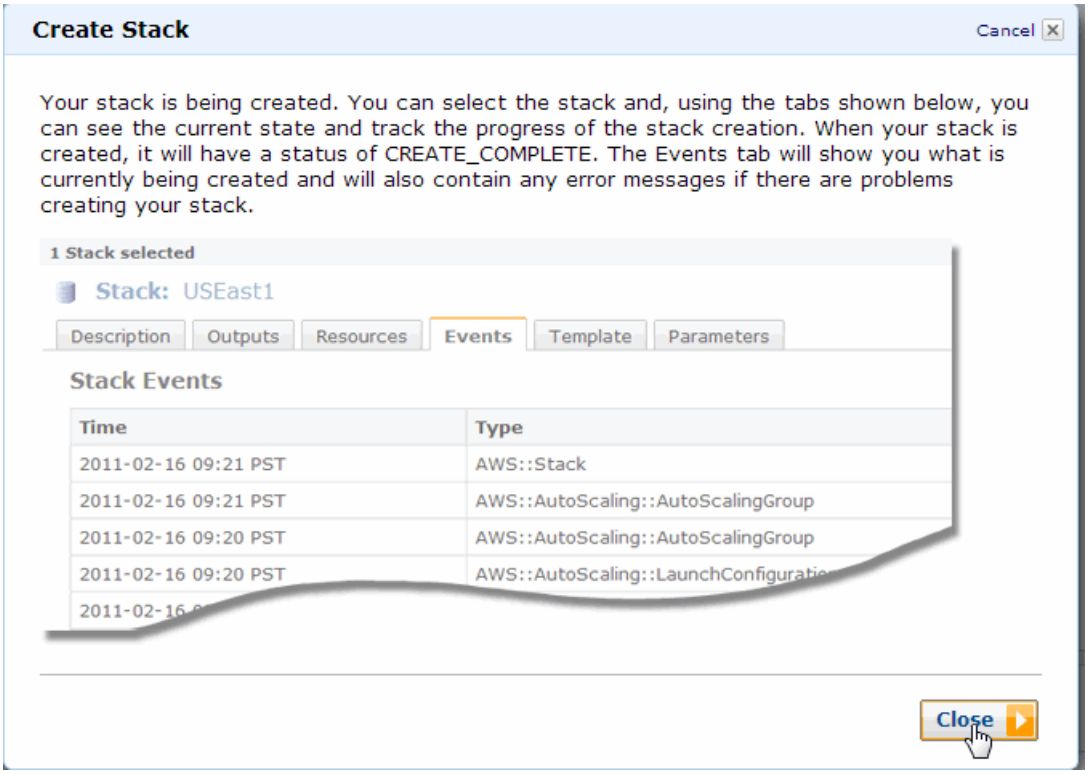
6. In the Add Tags window, enter any tags for your stack. The tag(s) you provide identify your EC2 resources in the EC2 dashboard. For example, if you identify the Key as `Name`, the given Value (`Test Stack`, for example) will appear in the Name column of the Instance list in the EC2 dashboard. For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



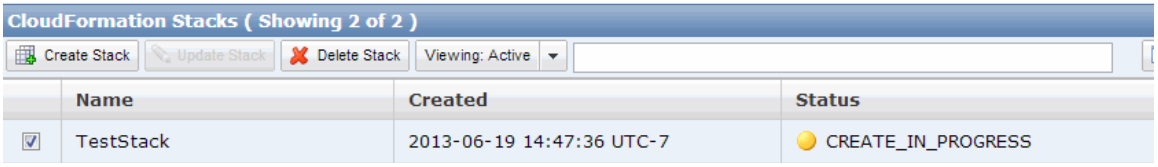
7. In the Review window, review the settings. Click edit to make any changes. When done, click Continue.



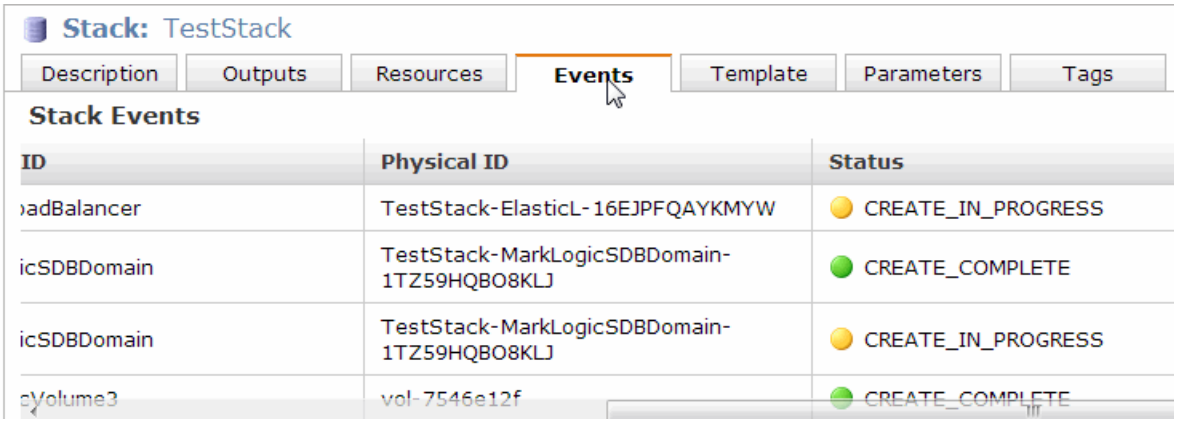
8. You will be notified that the stack is being created. Click Close.



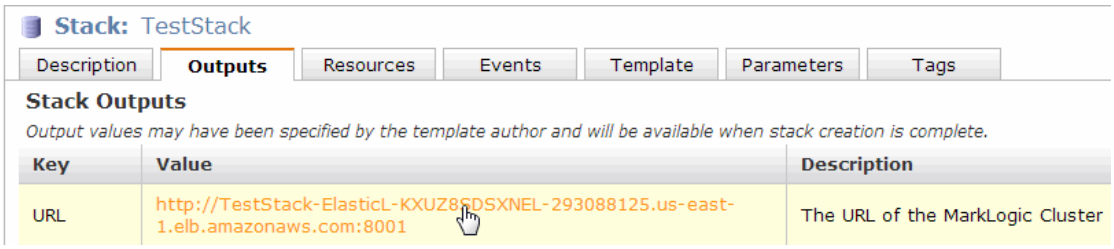
9. The name, create date, and status of your stack will appear at the top of the page.



10. 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.



11. 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 navigating to the Outputs tab 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.

12. 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 21.

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 38.

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 Sample CloudFormation Template

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.

The Sample Templates available from http://developer.marklogic.com/products/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 38, to become familiar with the procedures for building a cloud stack.

The main sections of the CloudFormation Template are as follows:

- Parameters Declaration
- Mappings Declaration
- Resources Declaration
- Outputs Declaration

Note: These sample templates create an ELB, as well as enable a public IP for each MarkLogic Server. The output of the stack lists the URL of the ELB. Applications should generally use the ELB as their endpoint. XCC applications, such as mlcp, need to set the `xcc.httpcompliant=true` mode in order to connect through the ELB reguardless of session affinity issues. For details, see Using a Load Balancer or Proxy Server with an XCC Application in the *XCC Developer's Guide*.

3.6.1 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 38.

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Description":"Create a cluster with three node or more nodes,
multi az, load balanced, MarkLogic Cluster.
MarkLogic-8.0-20150512.x86_64.rpm",
  "Parameters":
   {
    "AdminUser":
     {
      "Description":"The MarkLogic Administrator Username",
      "Type":"String"
     },
```

```
    "AdminPass":
     {
      "Description":"The MarkLogic Administrator Password",
      "Type":"String",
      "NoEcho":"true"
     },
    "InstanceType":
     {
      "Description":"Type of EC2 instance to launch",
      "Type":"String",
      "Default":"r3.8xlarge",
      "AllowedValues":
[ "t2.small", "t2.medium", "m1.medium", "m3.medium", "m3.large",
"m3.xlarge", "m3.2xlarge", "cc1.4xlarge", "cc2.8xlarge", "c3.large",
"c3.xlarge", "c3.2xlarge", "c3.4xlarge", "c3.8xlarge", "cr1.8xlarge",
"r3.large", "r3.xlarge", "r3.2xlarge", "r3.4xlarge", "r3.8xlarge",
"i2.xlarge", "i2.2xlarge", "i2.4xlarge", "i2.8xlarge", "hi1.4xlarge",
"hs1.8xlarge" ]
     },
    "IAMRole": {
      "Description":"IAM Role",
      "Type":"String"
     },
    "Licensee": {
      "Description":"The MarkLogic Licensee or 'none'",
      "Type":"String",
      "Default":"none"
     },
    "LicenseKey": {
      "Description":"The MarkLogic License Key or 'none'",
      "Type":"String",
      "Default":"none"
     },
    "KeyName": {
      "Description":"Name of and existing EC2 KeyPair to enable
SSH access to the instance",
      "Type":"String"
     },
    "LogSNS": {
      "Description":"SNS Topic for logging - optional/advanced",
      "Type":"String",
      "Default":"none"
     },
    "NodesPerZone": {
      "Description":"Total number of nodes per Zone. (3 zones).
Set to 0 to shutdown/hibernate",
      "Type":"Number",
      "MinValue":"0",
      "MaxValue":"20",
      "Default":"1"
     },
```

```
    "SpotPrice": {
      "Description":"Spot price for instances in USD/Hour -
Optional/advanced",
      "Type":"Number",
      "MinValue":"0",
      "MaxValue":"2",
      "Default":"0"
     },
    "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"
    },
    "Zone1": {
      "Description":"The AZ Zone 1 (e.g. us-west-2a)",
      "Type":"String",
      "AllowedValues":
       [ "ap-northeast-1a", "ap-northeast-1b", "ap-northeast-1c",
"ap-southeast-1a", "ap-southeast-1b", "ap-southeast-2a",
"ap-southeast-2b", "eu-west-1a", "eu-west-1b", "eu-west-1c",
"sa-east-1a", "sa-east-1b", "us-east-1a", "us-east-1b", "us-east-1c",
"us-east-1d", "us-east-1e", "us-west-1a", "us-west-1b", "us-west-1c",
"us-west-2a", "us-west-2b", "us-west-2c" ]
     },
    "Zone2": {
      "Description":"The AZ Zone 2",
      "Type":"String",
      "AllowedValues":
       [ "ap-northeast-1a", "ap-northeast-1b", "ap-northeast-1c",
"ap-southeast-1a", "ap-southeast-1b", "ap-southeast-2a",
"ap-southeast-2b", "eu-west-1a", "eu-west-1b", "eu-west-1c",
"sa-east-1a", "sa-east-1b", "us-east-1a", "us-east-1b", "us-east-1c",
"us-east-1d", "us-east-1e", "us-west-1a", "us-west-1b", "us-west-1c",
"us-west-2a", "us-west-2b", "us-west-2c" ]
     },
```

```
    "Zone3": {
      "Description":"The AZ Zone 3",
      "Type":"String",
      "AllowedValues":
       [ "ap-northeast-1a", "ap-northeast-1b", "ap-northeast-1c",
"ap-southeast-1a", "ap-southeast-1b", "ap-southeast-2a",
"ap-southeast-2b", "eu-west-1a", "eu-west-1b", "eu-west-1c",
"sa-east-1a", "sa-east-1b", "us-east-1a", "us-east-1b", "us-east-1c",
"us-east-1d", "us-east-1e", "us-west-1a", "us-west-1b", "us-west-1c",
"us-west-2a", "us-west-2b", "us-west-2c" ]
     }
   },
  "Conditions": {
    "UseLogSNS": {
      "Fn::Not": [ {
         "Fn::Equals": [ {
            "Ref":"LogSNS"
           },"none"]
        }]
     },
    "UseSpot": {
      "Fn::Not": [ {
         "Fn::Equals": [ {
            "Ref":"SpotPrice"
           }, 0 ]
        }]
     }
   },
```

3.6.2 Mappings Declaration

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

The `AWSInstanceType2Arch` map defines the values for all of the possible instance types. The `PVM` value defines the instance type as a Paravirtual Machine, whereas the `HVM` value defines the instance type as a Hardware Virtual Machine. The `AWSRegionArch2AMI` map defines the AMIs for each region. Each region has both a `PVM` and `HVM` AMI.

For details on HVM AMIs, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using_cluster_computing.html.

```
"Mappings":
  {
    "AWSInstanceType2Arch": {
      "c1.medium" : {
        "Arch" : "PVM"
      },
      "c1.xlarge" : {
        "Arch" : "PVM"
      },
      "c3.2xlarge" : {
        "Arch" : "HVM"
      },
      "c3.4xlarge" : {
        "Arch" : "HVM"
      },
      "c3.8xlarge" : {
        "Arch" : "HVM"
      },
      "c3.large" : {
        "Arch" : "HVM"
      },
      "c3.xlarge" : {
        "Arch" : "HVM"
      },
      "cc2.8xlarge" : {
        "Arch" : "HVM"
      },
      "cr1.8xlarge" : {
        "Arch" : "HVM"
      },
      "hi1.4xlarge" : {
        "Arch" : "HVM"
      },
      "hs1.8xlarge" : {
        "Arch" : "HVM"
      },
      "i2.2xlarge" : {
        "Arch" : "HVM"
      },
```

```
"i2.4xlarge" : {
  "Arch" : "HVM"
},
"i2.8xlarge" : {
  "Arch" : "HVM"
},
"i2.xlarge" : {
  "Arch" : "HVM"
},
"m1.large" : {
  "Arch" : "PVM"
},
"m1.medium" : {
  "Arch" : "PVM"
},
"m1.small" : {
  "Arch" : "PVM"
},
"m1.xlarge" : {
  "Arch" : "PVM"
},
"m2.2xlarge" : {
  "Arch" : "PVM"
},
"m2.4xlarge" : {
  "Arch" : "PVM"
},
"m2.xlarge" : {
  "Arch" : "PVM"
},
"m3.2xlarge" : {
  "Arch" : "HVM"
},
"m3.large" : {
  "Arch" : "HVM"
},
"m3.medium" : {
  "Arch" : "HVM"
},
"m3.xlarge" : {
  "Arch" : "HVM"
},
"r3.2xlarge" : {
  "Arch" : "HVM"
},
"r3.4xlarge" : {
  "Arch" : "HVM"
},
"r3.8xlarge" : {
  "Arch" : "HVM"
},
"r3.large" : {
  "Arch" : "HVM"
},
```

```
      "r3.xlarge" : {
        "Arch" : "HVM"
      },
      "t2.medium" : {
        "Arch" : "HVM"
      },
      "t2.small" : {
        "Arch" : "HVM"
      },
      "cc1.4xlarge" : {
        "Arch" : "HVM"
      }
    },
```

`AWSRegionArch2AMI` describes the AMIs used. These will change for each new release of MarkLogic.

```
    "AWSRegionArch2AMI": {

      "us-east-1" : {
        "HVM" : "ami-a5ee08ce"
      },
      "us-west-1" : {
        "HVM" : "ami-d313fb97"
      },
      "us-west-2" : {
        "HVM" : "ami-ed81bddd"
      },
      "eu-west-1" : {
        "HVM" : "ami-e9b9c99e"
      },
      "ap-southeast-1" : {
        "HVM" : "ami-181e264a"
      },
      "ap-southeast-2" : {
        "HVM" : "ami-a993ea93"
      },
      "ap-northeast-1" : {
        "HVM" : "ami-ae61b0ae"
      },
      "sa-east-1" : {
        "HVM" : "ami-35b13028"
      }
    }
  },
```

3.6.3 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:

- 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
  - Elastic Load Balancer (ELB)
- ELB ports
- Health Check values
- Security Group for each EC2 Instance

The first part of the `Resources` portion define 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.

```
"Resources":
 {
  "MarklogicVolume1":
   {
    "Type":"AWS::EC2::Volume",
    "Properties": {
      "AvailabilityZone": {
        "Ref":"Zone1"
       },
      "Size": {
        "Ref":"VolumeSize"
       },
      "Tags": [ {
        "Key":"Name",
        "Value":"MarkLogicData 1"
        }],
```

```
            "VolumeType" : {
              "Ref" : "VolumeType"
            }
          }
        },
```

`MarkLogicDDBTable` creates a DynamoDB database used as the Metadata Database, described in “Amazon EC2 Terminology” on page 4, 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" : [ {
              "AttributeName" : "node"
              "KeyType" : "HASH",
            } ],
            "ProvisionedThroughput" : {
              "WriteCapacityUnits" : "10",
              "ReadCapacityUnits" : "10"
            }
          }
        },
```

`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",
      "Properties": {
        "AvailabilityZones": [ {
           "Ref":"Zone1"
         } ],
        "LaunchConfigurationName": {
          "Ref":"LaunchConfig1"
         },
        "MinSize":"0",
        "MaxSize": {
          "Ref":"NodesPerZone"
         },
        "DesiredCapacity": {
          "Ref":"NodesPerZone"
         },
        "Cooldown": "300",
        "HealthCheckType": "EC2",
        "HealthCheckGracePeriod": "300",
        "LoadBalancerNames": [ {
           "Ref":"ElasticLoadBalancer"
        } ],
        "NotificationConfiguration": {
           "Fn::If" : [ "UseLogSNS", {
             "TopicARN": {
               "Ref":"LogSNS"
             },
```

`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"
          } ]
        }
      }
    },
```

`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",
      "Properties": {
        "KeyName": {
          "Ref":"KeyName"
         },
        "ImageId": {
          "Fn::FindInMap": ["AWSRegionArch2AMI", {
             "Ref":"AWS::Region"
          }, {
          "Fn::FindInMap": ["AWSInstanceType2Arch", {
             "Ref":"InstanceType"
          }, "Arch" ]
        } ]
      },
```

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 24. 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": {
  "Fn::Base64" : {
    "Fn::Join" : [ "", [ "MARKLOGIC_CLUSTER_NAME=", {
      "Ref" : "MarkLogicDDBTable"
    }, "\n", "MARKLOGIC_EBS_VOLUME=", {
      "Ref" : "MarklogicVolume1"
    }, ",:", {
      "Ref" : "VolumeSize"
    }, "::", {
      "Ref" : "VolumeType"
    }, "::,*\n", "MARKLOGIC_NODE_NAME=NodeA#\n",
       "MARKLOGIC_ADMIN_USERNAME=", {
      "Ref" : "AdminUser"
    }, "\n", "MARKLOGIC_ADMIN_PASSWORD=", {
      "Ref" : "AdminPass"
    }, "\n", "MARKLOGIC_CLUSTER_MASTER=1\n",
       "MARKLOGIC_LICENSEE=", {
      "Ref" : "Licensee"
    }, "\n", "MARKLOGIC_LICENSE_KEY=", {
      "Ref" : "LicenseKey"
    }, "\n", "MARKLOGIC_LOG_SNS=", {
      "Ref" : "LogSNS"
    }, "\n" ] ]
  }
},
```

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": {
          "Fn::If": ["UseSpot", {
             "Ref":"SpotPrice"
           }, {
            "Ref":"AWS::NoValue"
           } ]
        }
      }
    },
```

`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",
      "Properties": {
        "AppCookieStickinessPolicy": [ {
           "CookieName":"SessionID",
           "PolicyName":"MLSession"
         } ],
        "AvailabilityZones": [ {
           "Ref":"Zone1"
         }, {
           "Ref":"Zone2"
         }, {
           "Ref":"Zone3"
         }],
        "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"
     }
   }
},
```

`InstanceSecurityGroup` defines the Security Group for each EC2 Instance. For details on the `AWS::EC2::SecurityGroup` type, see http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-security-group.html.

```
"InstanceSecurityGroup": {
 "Type":"AWS::EC2::SecurityGroup",
 "Properties": {
   "GroupDescription":"Enable SSH access and HTTP
    access on the inbound port",
   "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",
      "SourceSecurityGroupOwnerId": {
        "Fn::GetAtt": [
          "ElasticLoadBalancer","SourceSecurityGroup.OwnerAlias"
        ] },
      "SourceSecurityGroupName": {
        "Fn::GetAtt": [
          "ElasticLoadBalancer","SourceSecurityGroup.GroupName"
        ] }
     },
```

```
                {
                 "IpProtocol":"tcp",
                 "FromPort": "8000",
                 "ToPort": "8010",
                 "CidrIp":"0.0.0.0/0"
                }, {
                 "IpProtocol":"tcp",
                 "FromPort": "7997",
                 "ToPort": "7997",
                 "SourceSecurityGroupOwnerId": {
                   "Fn::GetAtt": [
                     "ElasticLoadBalancer","SourceSecurityGroup.OwnerAlias"
                   ] },
                 "SourceSecurityGroupName": {
                   "Fn::GetAtt": [
                     "ElasticLoadBalancer","SourceSecurityGroup.GroupName"
                   ] }
                } ]
            }
          },
```

`InstanceSecurityGroupIngress` defines the ingress rules that control the inbound traffic that is allowed to reach your instances. For details on the `AWS::EC2::SecurityGroupIngress` type, see http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-security-group-ingress.html.

Note: The reason `InstanceSecurityGroupIngress` is separate from `InstanceSecurityGroup` is that it needs to reference the security group itself.

```
  "InstanceSecurityGroupIngress": {
        "Type":"AWS::EC2::SecurityGroupIngress",
        "Properties": {
          "IpProtocol":"tcp",
          "GroupName": {
            "Ref":"InstanceSecurityGroup"
           },
          "FromPort": "7999",
          "ToPort": "7999",
          "SourceSecurityGroupName": {
            "Ref":"InstanceSecurityGroup"
           }
        }
      }
    },
```

3.6.4 Outputs Declaration

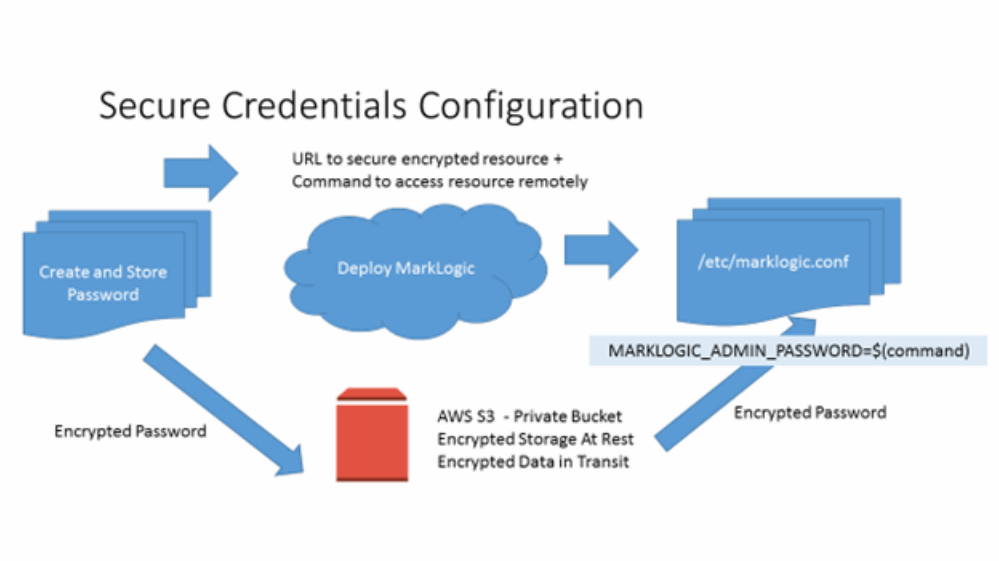
If the `CloudFormation` launch is successful, `Outputs` generates the URL of the ELB pointing to the MarkLogic Admin Interface port (8001).

```
"Outputs": {
  "URL": {
    "Description":"The URL of the MarkLogic Cluster",
    "Value": {
      "Fn::Join": [ "", [ "http://", {
          "Fn::GetAtt": ["ElasticLoadBalancer","DNSName"]
        },":8001" ] ]
    }
  }
 }
}
```

3.7 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 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 28, 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"
        ]  }
  }}
}}
```

3.8 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: When a stack is deleted, the EBS volume that was created for the first node in each zone is also deleted. However the EBS volumes for any additional nodes in each zone are not deleted. This is because they were not created directly in the CloudFormation stack, but instead as a part of the startup process of the additional nodes.

4.0 Managing MarkLogic Server on EC2

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

- Accessing a MarkLogic Server Instance
- Accessing an EC2 Instance
- Detecting EC2 Errors
- Using the mlcmd Script
- Configuring MarkLogic for Amazon Simple Storage Service (S3)
- Scaling Cluster Resources on EC2
- Upgrading the MarkLogic AMI
- Monitoring (CloudWatch)
- Migrating from Enterprise Data Center to EC2
- Creating an EBS Volume and Attaching it to an Instance
- Pausing or Terminating a MarkLogic Cluster

4.1 Accessing a MarkLogic Server Instance

This section describes how to access the an instance of MarkLogic Server in EC2.

There are two ways to access a MarkLogic Server instance:

- Using the Elastic Load Balancer (ELB) URL
- Using the Public DNS for an Instance

The difference is that the ELB URL will direct you to any available instance of MarkLogic Server in your cluster. If you want to access a specific instance, as you would when running the `mlcmd` script described in “Using the mlcmd Script” on page 74, then use the Public DNS for that instance.

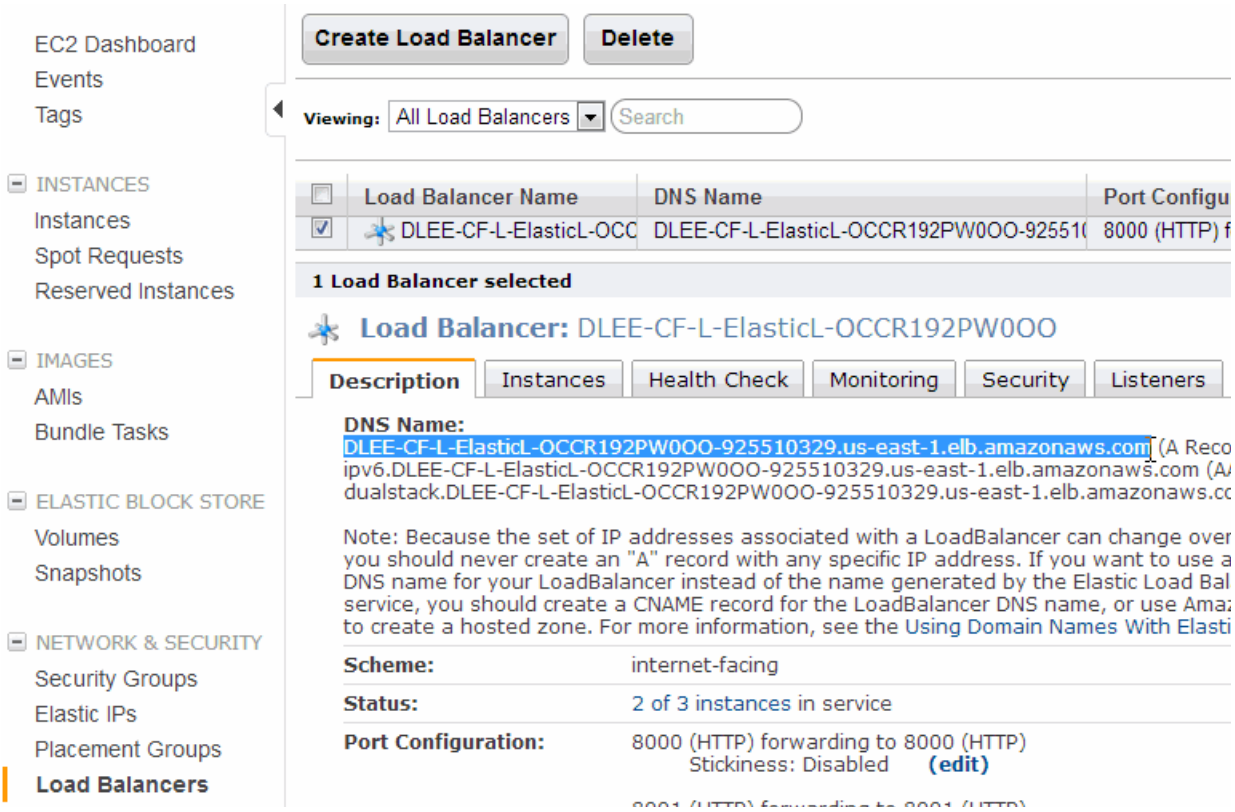
4.1.1 Accessing MarkLogic Server through the ELB

You can access the MarkLogic Admin Interface through the ELB by clicking on the URL in the Outputs portion of the CloudFormation Console, as described in Step 11 in "Creating a CloudFormation Stack using the AWS Console" on page 38.

This section describes how to access MarkLogic Server through the ELB from the EC2 Dashboard.

1. In the EC2 Dashboard, click on Load Balancers in the left-hand navigation menu.

2. Copy URL from the DNS name.



You use the URL to access the MarkLogic Server. You can access any of the ports you have defined as an ELB port. For example, if the URL is `DLEE-CF-L-ElasticL-OCCR192PW0OO-925510329.us-east-1.elb.amazonaws.com`, then, to access MarkLogic port 7800, the URL you enter into the browser would be:

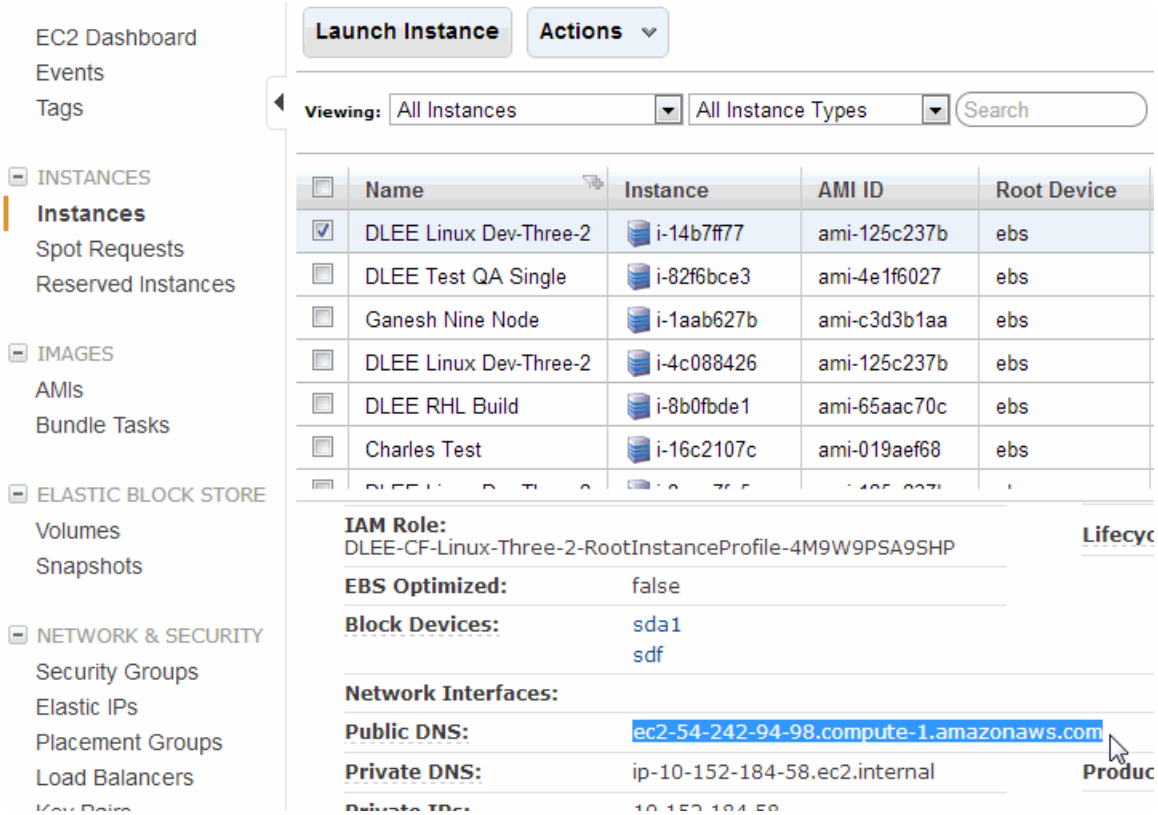
```
http://DLEE-CF-L-ElasticL-OCCR192PW0OO-925510329.us-east-
1.elb.amazonaws.com:7800
```

Note: Do not use a load balancer to access MarkLogic port 8001. The Admin Interface is not designed to be used behind a load balancer.

4.1.2 Accessing MarkLogic Server through the Instance Public DNS

This section describes how to access MarkLogic Server through the Public DNS of an Instance.

1. In the EC2 Dashboard, click on Instances in the left-hand navigation menu. Select your MarkLogic Server Instance and copy the Public DNS value:



You use the Public DNS to formulate part of the URL to access MarkLogic Server. For example, if the Public DNS is `ec2-54-242-94-98.compute-1.amazonaws.com`, then, to access the Admin Interface, the URL you enter into the browser would be:

```
http://ec2-54-242-94-98.compute-1.amazonaws.com:8001
```

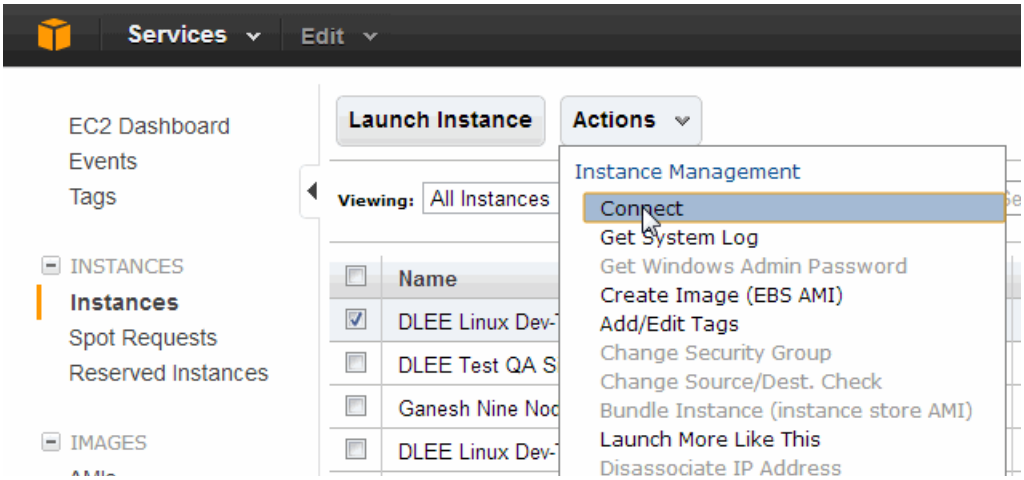
4.2 Accessing an EC2 Instance

You may need to SSH into an EC2 instance for certain task, such as checking the log files for that instance, as described in “Detecting EC2 Errors” on page 74.

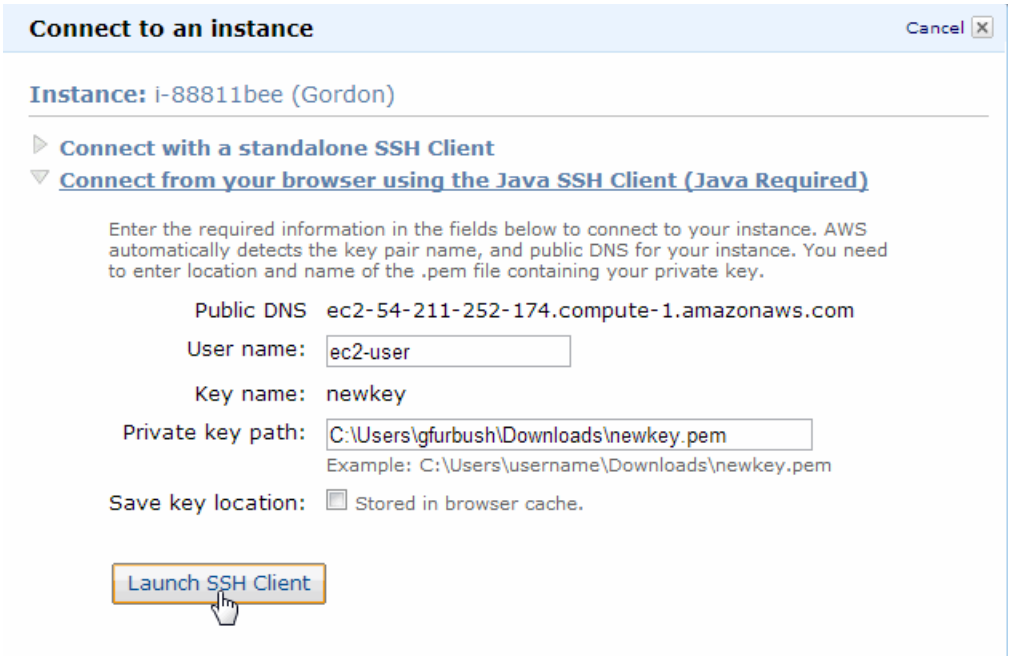
> **Note:** You cannot SSH to the load balancer, you must SSH to a specific EC2 instance. To SSH into an EC2 instance, you must have the key pair used by the instance downloaded to your local host.

To SSH into an instance, do the following:

1. Open the EC2 Dashboard.
2. Select Instances from the left-hand navigation section.
3. Select the instance to which you want to connect.
4. Select Connect from the Actions pull-down menu.



5. Specify a `ec2-user` as the User Name and provide the path to your copy of the key pair you downloaded to your local host. Click Launch SSH Client.



6. This will open up a shell window to the EC2 instance. When you first connect in this manner, you may be prompted to create various directories. Respond by clicking `Yes` for each prompt.

Alternatively you can open a shell window and SSH into an instance using the following command:

```
ssh -i /path/to/keypair.pem ec2-user@<Public DNS>
```

For example, if your keypair, named `newkey.pem`, is stored in your `c:/stuff/` directory, you can access the instance with a public DNS of `ec2-54-242-94-98.compute-1.amazonaws.com` as follows:

```
ssh -i c:/stuff/newkey.pem ec2-user@ec2-54-242-94-98.compute-1.amazonaws.com
```

4.3 Detecting EC2 Errors

Start up errors are stored in the `/var/log/messages` file in each instance. To view the `messages` file, SSH into an instance as described in “Accessing an EC2 Instance” on page 72.

To access the `messages` file, you must be super user. For example, if you want to tail the `messages` file, enter:

```
sudo tail -f /var/log/messages
```

You can also capture errors related to CloudFormation stack by means of the SNS Topic, as described in “Creating a Simple Notification Service (SNS) Topic” on page 21.

4.4 Using the mlcmd Script

The `mlcmd` script supports startup operations and advanced use of the Managed Cluster features. The `mlcmd` script is installed as an executable script in `/opt/MarkLogic/bin/mlcmd`.

In order to run `mlcmd`, you must be logged into the host and running as root or with root privileges. You must also have Java installed and the `java` command in the `PATH` or `JAVA_HOME` set to the `JRE` or `JDK` home directory. The first time you start MarkLogic on your server the `/var/local/mlcmd.conf` file is created, which is required to use the `mlcmd` script. Once the `/var/local/mlcmd.conf` file is created, it is not necessary to start MarkLogic to use the `mlcmd` script.

The syntax of `mlcmd` is as follows:

```
mlcmd command
```

The `mlcmd` commands are listed below:

mlcmd Command	Description
sync-volumes-from-mdb	Attaches EBS volumes not currently attached to this instance.
sync-volumes-to-mdb	Synchronizes the EBS volumes currently attached to the system and stores them in the Metadata Database.
init-volumes-from-system	Initialize volumes identical to the process performed on startup.
leave-cluster	Removes the host on which it is executed from the cluster.

4.4.1 sync-volumes-from-mdb

This command looks in the Metadata Database and does the following:

- Locates any EBS volumes not currently attached to this instance and attaches them.
- If the volume does not contain a filesytem, a filesystem is created (ext4).
- Mounts the device to the mount point indicated in the Metadata Database.
- Applies all tags from the current EC2 instance prefixed by `marklogic:` to the EBS volume.

4.4.2 sync-volumes-to-mdb

This command can be run any time after the initial startup. It synchronizes the EBS volumes currently attached to the system and stores them in the Metadata Database so that on the next restart they will be attached and mounted. The following steps are performed:

- Locates all EBS volumes to the system.
- For all volumes in the managed range enters an entry to the Metadata Database indicating the following:
    - EBS Volume ID
    - EBS Mount device
    - Operating System mount device
    - Operating system mount point (directory)
- For volumes which are attached but not mounted then the mount point is set to the default mount point for that volume (see the Default EBS Mount Points table below).

No changes to existing attachments, filesystem, or mount points are performed.

Default EBS Mount Points

EC2 Device	RedHat Device	Linux Device	Mount Point
/dev/sdf	/dev/xvdj	/dev/xvdf	/var/opt/MarkLogic
/dev/sdg	/dev/xvdk	/dev/xvdg	/var/opt/volume1
/dev/sdh	/dev/xvdl	/dev/xvdh	/var/opt/volume2
/dev/sdi	/dev/xvdm	/dev/xvdi	/var/opt/volume3
/dev/sdj	/dev/xvdn	/dev/xvdj	/var/opt/volume4
/dev/sdk	/dev/xvdo	/dev/xvdk	/var/opt/volume5
/dev/sdl	/dev/xvdp	/dev/xvdl	/var/opt/volume6
/dev/sdm	/dev/xvdq	/dev/xvdm	/var/opt/volume7
/dev/sdn	/dev/xvdr	/dev/xvdn	/var/opt/volume8
/dev/sdo	/dev/xvds	/dev/xvdo	/var/opt/volume9

4.4.3 init-volumes-from-system

This command looks at the current system and attempts to initialize volumes identical to the process performed on startup.

- For each volume listed as a user data variable MARKLOGIC_EBS_VOLUME<N>:
  - Attaches the volume to the system if needed.
  - Creates a filesystem if needed.
- For each EBS volume attached to the system in the managed range:
  - Creates a filesystem if needed.
  - Mounts the device to the default mount point (or the mount point currently in the the Metadata Database).
  - Updates the Metadata Database with the current EBS Volume, OS device and mount point.

4.4.4 leave-cluster

This command can be executed on a host to remove that host from the cluster. This command also removes the host from the cluster configuration information stored in the Metadata Database. The command leaves the host server in pre-initialized state (same as a fresh install). If the server is restarted, then it will re-join the cluster the same manner as an initial start.

Use the optional `-terminate` argument to terminate the instance and decrement the `DesiredCount` attribute of the AutoScaling group by one after leaving the cluster.

4.5 Configuring MarkLogic for Amazon Simple Storage Service (S3)

Amazon S3 support is built into MarkLogic Server as an available file system type. You configure S3 access at the group level. Once you have configured a group for S3, any forest in the group can be placed on S3 by specifying an S3 Path. Additionally, any host in the group can do backups to S3, restore from S3, as well as read and write directories and files on S3.

> **Note:** Transaction journaling does not work on S3 because the S3 file system cannot do the file operations necessary to maintain a journal. Unless your S3 forest is configured with a fast data directory or updates allowed is set to `read-only`, you must set the journaling option on your database to `off` before attaching the forest to the database. This is not a requirement for backup/restore operations on a database, however.

To configure MarkLogic to access Amazon S3, do the following:

- Set up an S3 Bucket
- Configure the S3 Endpoint for your Group
- Configure S3 Credentials
- Set an S3 Path in Forest Data Directory
- Load Content into MarkLogic to Test

4.5.1 Set up an S3 Bucket

Follow the directions in http://docs.aws.amazon.com/gettingstarted/latest/wah/getting-started-create-bucket.html to set up your S3 bucket.

Warning There can be multiple problems if the bucket name contains a period (.). Instead use a dash (-) for maximum compatibility with S3.

Bucket names are global and they are not scoped to your account. You should choose bucket names that have a good chance of being universally unique. For example:

- Bad: `test`
- Good: `zippy-software-org-test`

Note: Do not use the S3 Management Console to upload your content to S3. Instead, follow any of the procedures described in the *Loading Content Into MarkLogic Server Guide* after you have completed the configuration procedures.

4.5.2 Configure the S3 Endpoint for your Group

The S3 Endpoint is configured by specifying the S3 properties for your MarkLogic group.

1. Log into the Admin Interface.
2. Click the name of your group under the Groups icon on the left tree menu.
3. In the Group Configuration page, scroll down to the bottom to locate the S3 fields:



Set the S3 fields as follows:

Setting	Description
s3 domain	The domain used for the S3 endpoint. The default value is set for your region. However, you can change it, if necessary. References to the regional endpoints can be found at http://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region.
s3 protocol	You can choose either `http` or `https` for communication with S3. The default is `http`.
s3 server side encryption	Storage on S3 can participate in server-side encryption. The default is `none` but you can set `aes256` to enable server-side encryption.

4.5.3 Configure S3 Credentials

In order to use S3 you must supply S3 credentials. You can configure S3 credentials in one of three ways:

- Configuring S3 Credentials in the Security Database
- Configuring S3 Credentials in Environment Variables
- Configuring an IAM Role with an S3 Access Policy

The order of precedence for locating S3 credentials is:

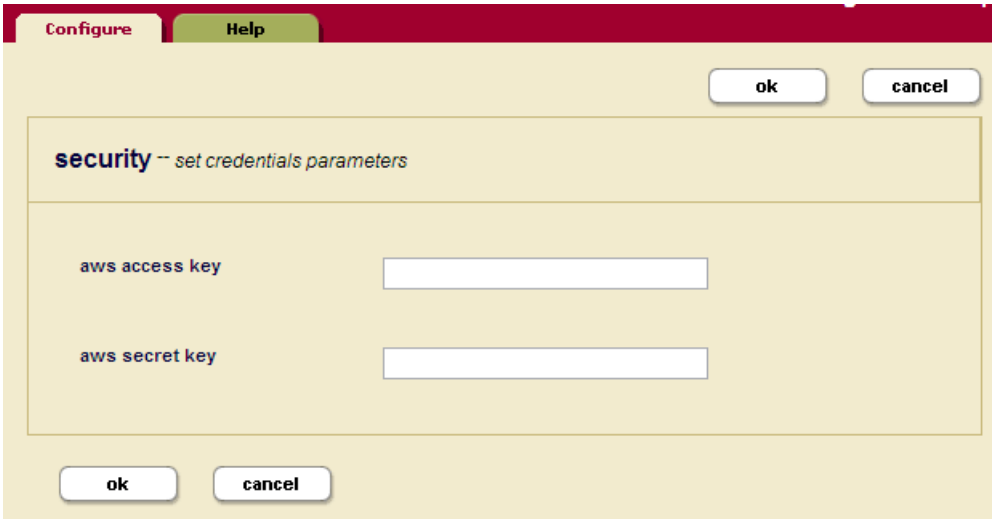
1. Credentials configured in the server Security database
2. Environment variables
3. IAM Role

4.5.3.1 Configuring S3 Credentials in the Security Database

In the Security, Credentials, Configure tab are fields for specifying the S3 credentials.

1. Log into the Admin Interface.
2. Click Security icon on the left tree menu.
3. Click Credentials to open the Credential Configuration page.

4. Enter the aws access key and aws secret key provided for your AWS account.



4.5.3.2 Configuring S3 Credentials in Environment Variables

You can set a pair of environment variables that the server will use as S3 AWS Credentials. These can be passed in as EC2 User Data or set into the environment in which MarkLogic runs.

- MARKLOGIC_AWS_ACCESS_KEY -- Your AWS Access Key
- MARKLOGIC_AWS_SECRET_KEY -- Your AWS Secret Key

4.5.3.3 Configuring an IAM Role with an S3 Access Policy

If you run an EC2 instance with an associated IAM Role, you can select a policy template that provides S3 access, such as “Amazon S3 Full Access” or “Amazon S3 Read Only Access.”

Your IAM Role will be used for your security credentials so you do not need to store any AWS Credentials in MarkLogic or on the EC2 instance in order access S3 resources. This is the most secure way of accessing S3.

IAM roles are only used on the server if the `MARKLOGIC_AWS_ROLE` environment variable is set. This happens automatically for you unless you disable the EC2 configuration (such as setting `MARKLOGIC_EC2_HOST=0`), in which case the server will not use the `MARKLOGIC_AWS_ROLE` variable.

4.5.4 Set an S3 Path in Forest Data Directory

Set the data directory for the forest to a valid S3 path. For details on setting the forest data directory, see Creating a Forest in the *Administrator's Guide*. Multiple forests can be configured for the same bucket.

The form of an S3 path is:

```
s3://bucket/directory/file
```

Where:

Item	Description
bucket	The name of your S3 bucket.
directory	Zero or more directory names, separated by forward slashes (/).
file	The filename, if the path is to a specific file.

For a directory path (such as a Forest data directory), then a bucket by itself is sufficient and files will be placed in the bucket root.

Example paths to S3 directories:

```
s3://my-company-bucket

s3://my-company-bucket/directory

s3://my-company-bucket/dir1/dir2/dir3
```

Example paths to S3 files:

```
s3://my-company-bucket/file.xml

s3://my-company-bucket/directory/file.txt

s3://my-company-bucket/dir1/dir2/dir3/file.txt
```

Warning Unless your S3 forest is configured with a fast data directory or updates allowed is set to `read-only`, you must set journaling on your database to `off` before attaching the forest to the database. Failure to do so will result in a forest error and you will have to restart the forest after you have disabled journaling on the database.

4.5.5 Load Content into MarkLogic to Test

Load content into your S3 database using any of the methods described in *Loading Content Into MarkLogic Server Guide* and run a query to confirm you have successfully configured MarkLogic Server with S3.

> **Note:** Content uploaded directly to your bucket using the S3 Management Console will not be recognized by MarkLogic Server.

4.6 Scaling Cluster Resources on EC2

If you have created your stack using the 3+ CloudFormation template, you can temporarily add nodes and forests to scale up your cluster for periods of heavy use and then remove them later when less resources are needed.

Adding more hosts to a cluster is simple. Simply use the Update Stack feature to reapply the 3+ CloudFormation template and provide a larger number for the `NodesPerZone` setting. Alternatively, you can add hosts by means of your Auto Scaling Groups. The recommended way to scale up the data capacity of your cluster is to add additional volumes, as described in “Creating an EBS Volume and Attaching it to an Instance” on page 86.

Scaling a cluster down involves some manual intervention. The procedure is as follows:

1. Use MarkLogic and AWS tools to identify equal number of hosts in each ASG to delete. Never delete the host with the Security database, or any of the other built-in MarkLogic databases, such as Meters, App-Services, Modules, and so on.

2. Delete or move the data from the hosts to be removed to other hosts. This can be done by using the REST Management API or XQuery `tieredstorage` API to migrate partitions or forests to a volume on another host. For details on migrating data, see Migrating Forests and Partitions in the *Administrator's Guide*.

3. As a super user, run the leave-cluster `-terminate` command on each host to be removed. This will cause the node to leave the cluster, and adjust the AutoScaling Group `DesiredCount` setting. For details, see “leave-cluster” on page 77.

4. Delete any unused volumes.

5. Update the CloudFormation template to represent downsized cluster and use the Update Stack feature to reapply the template to the stack to alert AWS of the updated configuration.

4.7 Upgrading the MarkLogic AMI

AN existing Cloud Formation Template can be updated as long as the software and IT architecture are compatible and if the changes do not require destructive modifications to AWS Resources needed by the Managed Cluster feature or your data. General guidance on the effects of template updates can be found at http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks.html.

The latest sample templates are versioned starting with V8.0.3. Major version number changes represent incompatible implementations of the Managed Cluster feature so an EC2 stack created using an earlier CloudFormation template cannot be updated with a new CloudFormation template.

Within the same major template version, upgrades are supported by updating the AMI ids in your original template with the latest AMI ids from Marketplace corresponding to the same:

- AWS Region
- Paravirtualzation Type (HVM vs PVM)
- Instance Type
- EBS Volume Type

Warning Depending on your customizations, even using the same version and AMIs, some stack updates may be destructive to a running cluster and should be tested before applied to a production workload.

The following procedure describes how to upgrade your stack to use new AMIs for a new release of MarkLogic within the same major template version:

1. Locate the AMI ids in your original template and find the corresponding updated AMI ids from AWS MarketPlace or those listed at http://developer.marklogic.com/products/aws. For example, the `AWSRegionArch2AMI` definition in your original template might look like the following:

```
"AWSRegionArch2AMI":
 {
  "us-east-1":
   {
    "PVM":"ami-41633528",
    "HVM":"ami-4363352a"
   },
  "us-west-2":
   {
    "PVM":"ami-e85bc5d8",
    "HVM":"ami-ea5bc5da"
   },
  "eu-west-1":
   {
```

```
      "PVM":"ami-68fa1c1f",
      "HVM":"ami-56fa1c21"
     }
   }
},
```

If, for example, your instances are located in the `us-east-1`and `us-west-2` regions, open the new template, locate the `AWSRegionArch2AMI` definition, and copy the AMI ids for the `us-east-1`and `us-west-2` regions. For example, the new template contains:

```
"us-east-1" : {
   "HVM" : "ami-96ffe7fe"
},

"us-west-2" : {
   "HVM" : "ami-75d3f245"
},
```

You can then update your `AWSRegionArch2AMI` definition as follows:

```
"AWSRegionArch2AMI":
 {
  "us-east-1":
   {
    "PVM":"ami-41633528",
    "HVM":"ami-96ffe7fe"
   },
  "us-west-2":
   {
    "PVM":"ami-e85bc5d8",
    "HVM":"ami-75d3f245"
   },
  "eu-west-1":
   {
    "PVM":"ami-68fa1c1f",
    "HVM":"ami-56fa1c21"
   }
  }
},
```

2. Backup any important data.

3. Update stack with your updated CloudFormation template. Make sure the stack update is complete.

4. In the EC2 Dashboard, stop one instance at the time and wait for it to be replaced with a new one.

   Note: Some changes made outside of CloudFormation before the upgrade will cause the upgrade to fail.

5. In the EC2 Dashboard, terminate the other nodes. (Ideally one by one. They will come up and reconnect without any UI interaction.)

6. Go to 8001 port on the new instance where an upgrade prompt should be displayed. (`security-upgrade.xqy` screen)

7. Click OK and wait for the upgrade to complete on the instance.

The following procedure describes how to upgrade instances that are brought up directly from an AMI. For each MarkLogic instance in your cluster, do the following:

1. Terminate the instance.

2. Launch a new instance from the upgraded AMI.

3. Attach the EBS data volume associated with the original instance.

   Note: Customizations made before the upgrade to the instance or AMI may cause the upgrade to fail.

4.8 Monitoring (CloudWatch)

AWS provides robust monitoring of EC2 instances, EBS volumes, and other services via the CloudWatch service. You can use CloudWatch to set thresholds on individual AWS services and send notifications via SMS or Email when these thresholds have been exceeded. For example, you can set a threshold on excessive storage throughput. You can also create your own metrics to monitor with CloudWatch. For example, you might write a custom metric to monitor the current free memory on your instances and to alarm or trigger an automatic response should a memory threshold be exceeded.

For details on the use of CloudWatch, see http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/CHAP_UsingCloudWatch.html.

4.9 Migrating from Enterprise Data Center to EC2

This section describes to steps for migrating your data and configuration from a data center to EC2.

There are a number of ways you could migrate from a local data center to EC2. The following is one possible procedure.

1. Backup your databases on S3 storage. To do this, set your S3 security credentials, as described in “Configure S3 Credentials” on page 79, on your local MarkLogic cluster and, for your backup directory, provide the path to your S3 bucket, as described in “Set an S3 Path in Forest Data Directory” on page 81.

2. Use Configuration Manager to export all of the configuration data for your cluster, as described in Exporting a Configuration in the *Administrator's Guide*.

3. Create a CloudFormation template, as described in “Deploying MarkLogic on EC2 Using CloudFormation” on page 31, to recreate the hosts for your cluster on EC2.

4. Import your configuration data into your EC2 cluster, as described in Importing a Configuration in the *Administrator's Guide*.

5. Restore your backed-up data from S3 to your configured EC2 forests.

4.10 Creating an EBS Volume and Attaching it to an Instance

This section describes how to create an EBS volume and attach it to your MarkLogic Server instance.

In general, it is a best practice is to have one volume per node and one forest per volume.

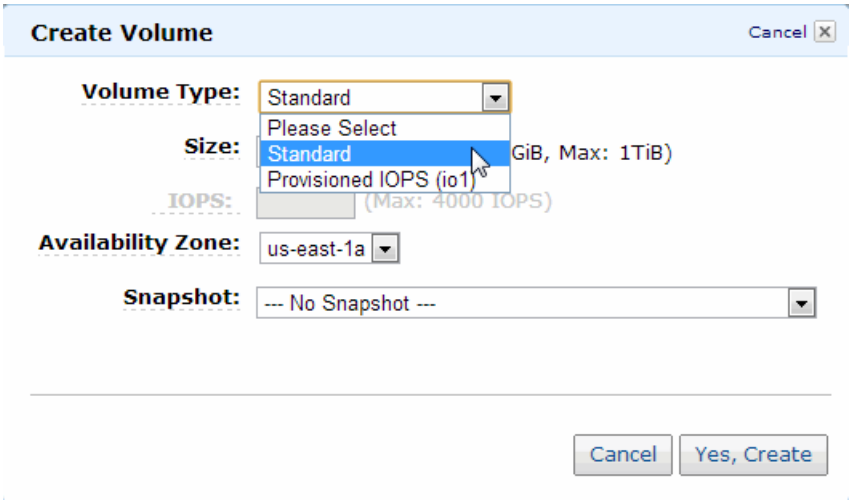
4.10.1 Creating and EBS Volume

Use the following procedure to create an EBS volume.

1. Open the EC2 Dashboard, select Volumes from the left-hand navigation section. In the EBS Volumes page, select Create Volume:



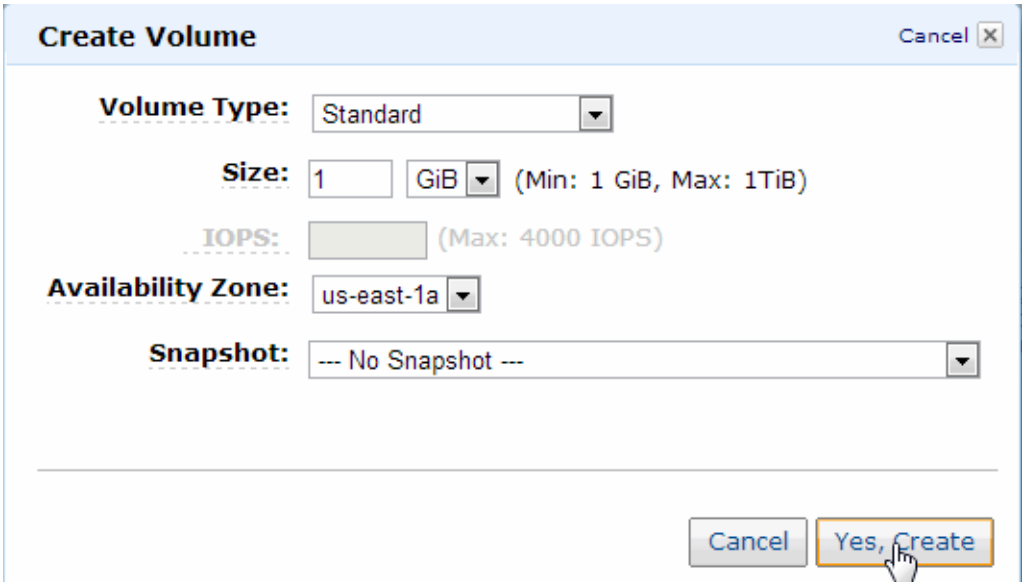
2. In the Create Volume window, specify the Volume Type from the pull-down menu.



3. Specify a volume size large enough for your needs and the same availability zone associated with your instance. Specify the same zone as the instance to which you intend to attach the volume. You can also optionally specify an EBS snapshot. See Help on the EBS snapshot page for details on how to create a snapshot.

Warning The zones for your instance and EBS volume may not be the same by default.

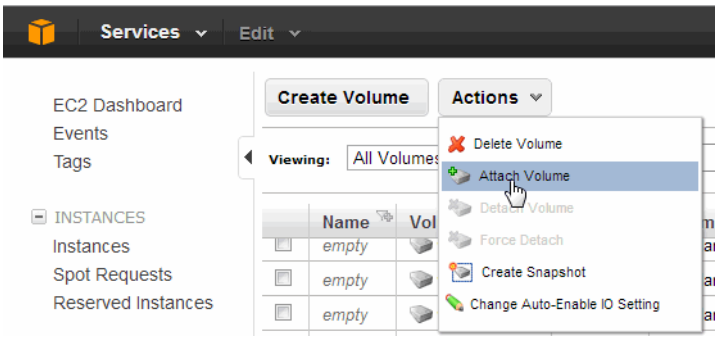
When finished, click Yes, Create. Locate the reference to this new volume in the right-hand section of the management console and verify that the State is `available`.



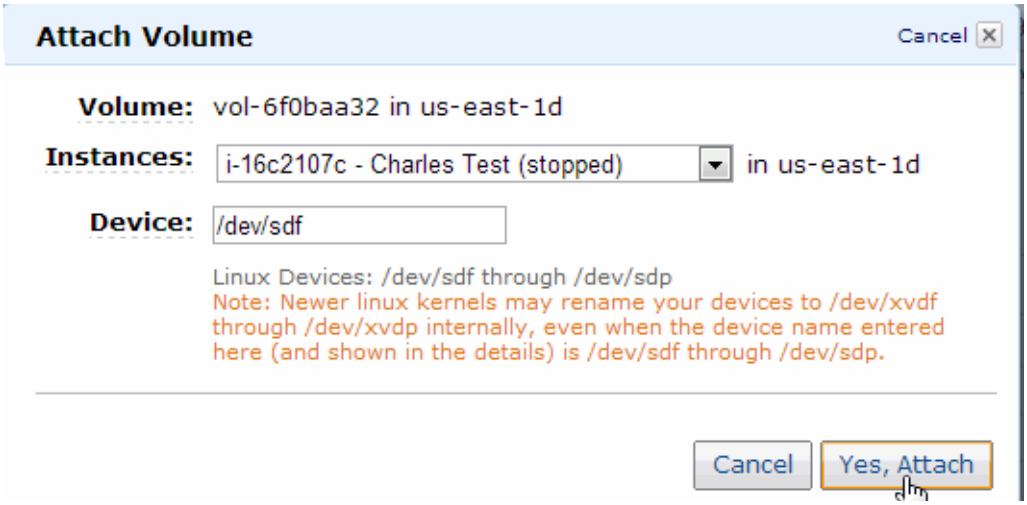
4.10.2 Attaching an EBS Volume to an Instance

This section describes how to use the EC2 Dashboard to attach a volume to an instance.

1. Select Volumes from the left-hand navigation section and then click Attach Volume.



2. In the Attach Volume window, specify the instance you launched from the MarkLogic Server AMI. For the Device selection, use `/dev/sdf`. Click Yes, Attach when you are finished. Locate the reference to this volume in the right-hand section of the management console and verify that the status is "in-use". If the status is not “in-use,” continue to click Refresh until the status changes to “in-use.”



3. SSH into the instance and execute the init-volumes-from-system command to create a filesystem for the volume and update the Metadata Database with the new volume configuration. The init-volumes-from-system command will output a detailed report of what it is doing. Note the mount directory of the volume from this report.

4. Once the volume is attached and mounted to the instance, log into the Administrator Interface on that host and create a forest, specifying host name of the instance and the mount directory of the volume as the forest Data Directory. For details on how to create a forest, see Creating a Forest in the *Administrator’s Guide*.

4.11 Pausing or Terminating a MarkLogic Cluster

At any time you can pause the cluster by using the Update Stack feature to reapply the CloudFormation template to your stack and setting the ASG `NodesPerZone` value to 0 for all nodes. You can later restart the node by resetting the `NodesPerZone` to a value of 1 - 20 for each ASG.

Warning Do not manually stop your MarkLogic instances from the EC2 dashboard, as each AutoScaling Group will detect that they have stopped and will automatically restart them. The same is true if you shutdown MarkLogic from the Admin Interface or by means of a MarkLogic API call.

To terminate your MarkLogic cluster, you can delete the stack, as described in “Deleting a CloudFormation Stack” on page 68.

5.0 Technical Support

For complete product documentation, the latest product release updates, and other useful information for developers, visit our developer site at http://developer.marklogic.com/cloudcomputing.

Support services are not included with MarkLogic Server for EC2 products. You can take advantage of community support by using the developer mailing list available on the developer site. Should you require formal support services, please inquire about the available options by emailing support-cloud@marklogic.com.

6.0 Copyright

MarkLogic Server 8.0 and supporting products.

NOTICE

Copyright © 2018 MarkLogic Corporation.

This technology is protected by one or more U.S. Patents 7,127,469, 7,171,404, 7,756,858, 7,962,474, 8,935,267, 8,892,599 and 9,092,507.

All MarkLogic software products are protected by United States and international copyright, patent and other intellectual property laws, and incorporate certain third party libraries and components which are subject to the attributions, terms, conditions and disclaimers found at http://docs.marklogic.com/guide/copyright/legal.

MarkLogic and the MarkLogic logo are trademarks or registered trademarks of MarkLogic Corporation in the United States and other countries. All other trademarks are property of their respective owners.

For all copyright notices, including third-party copyright notices, see the Combined Product Notices for your version of MarkLogic.