Alfresco Enterprise on AWS: Reference Architecture

October 2013

(Please consult **http://aws.amazon.com/whitepapers/** for the latest version of this paper)

Abstract

Amazon Web Services (AWS) provides a complete set of services and tools for deploying business-critical enterprise workloads on its highly reliable and secure cloud infrastructure. Alfresco is an enterprise content management system (ECM) useful for document and case management, project collaboration, web content publishing and compliant records management. Few classes of business-critical applications touch more enterprise users than enterprise content management (ECM) and collaboration systems.

This whitepaper provides IT infrastructure decision-makers and system administrators with specific technical guidance on how to configure, deploy, and run an Alfresco server cluster on AWS. We outline a reference architecture for an Alfresco deployment (version 4.1) that addresses common scalability, high availability, and security requirements, and we include an implementation guide and an AWS CloudFormation template that you can use to easily and quickly create a working Alfresco cluster in AWS.

Introduction

Enterprises need to grow and manage their global computing infrastructures rapidly and efficiently while simultaneously optimizing and managing capital costs and expenses. The computing and storage services from AWS meet this need by providing a global computing infrastructure as well as services that simplify managing infrastructure, storage, and databases. With the AWS infrastructure, companies can rapidly provision compute capacity or quickly and flexibly extend existing on-premises infrastructure into the cloud.

Alfresco is an enterprise content management (ECM) platform for use by organizations interested in managing business-critical processes that are related to document management, collaboration, secure mobile and desktop access to vital files. The flexible compute, storage, and database services that AWS offers make it an ideal platform on which to run an Alfresco deployment.

Alfresco Enterprise Reference Architecture

While Alfresco supports a wide variety of content management use cases (including documents, records, web publishing, and more), this whitepaper presents a single common core configuration that you can adapt to virtually any scenario. The reference architecture described in this whitepaper maps AWS services to all of the components required by an Alfresco service. This whitepaper also includes some information on using an AWS CloudFormation template to install and configure an Alfresco cluster, which can be performed in approximately 30-40 minutes. For a full detailed walkthrough of the security groups, policies, and configuration file modifications used in the relevant AWS CloudFormation template, see the Implementation Guide that accompanies this whitepaper.

A typical Alfresco cluster requires the following components:

- An HTTP(S) load balancer
- Two or more Alfresco servers
- Shared file storage
- A shared database

You can run each of these components using Amazon Elastic Compute Cloud (Amazon EC2). We recommend that you simplify administration and probably lower your overall costs by using the other AWS services that correspond to Alfresco requirements. Here are the AWS services that correspond to the Alfresco requirements and that we use in this whitepaper.

- The Elastic Load Balancing service provides HTTP and HTTPS load balancing across the Alfresco servers.

  Note: When you use Elastic Load Balancing, you must upload the web server's certificate and private key to the AWS Identity and Access Management (IAM) service before you can enable the HTTPS listener.

- The Amazon EC2 service provides auto scaling, with which your Alfresco cluster can add or reduce servers based on their use, providing additional servers during peak hours and lowering costs by removing servers during off hours. This functionality is tightly integrated with the Elastic Load Balancing service and automatically adds and removes instances from the load balancer.
- Amazon Simple Storage Service (Amazon S3) provides shared file storage for the cluster. Amazon S3 is an ideal storage system for Alfresco for several reasons:
  - It is highly durable object storage designed to provide 11 9’s (99.999999999%) of durability, which means you no longer need to manage backups of your content store.
  - Alfresco stores items as objects. Changes to objects are stored as unique objects rather than as updates to existing objects. This makes Amazon S3 a perfect storage system, because POSIX compatibility is not required.
  - Amazon S3 provides virtually unlimited scalability with support for an unlimited number of objects up to 5 TB in size, and customers only pay for they use. This greatly simplifies sizing your environment, because you don't need to worry about how much space your cluster will need in the future, and your storage costs map directly to the amount of storage that you use.
- Amazon Relational Database Service (Amazon RDS) for MySQL is used for the shared database. Amazon RDS is a managed database service — all the administrative tasks for managing the database are handled by AWS. The database is deployed in multiple Availability Zones for high availability and automatically backed up on a schedule that you define.

Architecture Overview

Before you begin working with the AWS CloudFormation template, it's a good idea to familiarize yourself with regions, Availability Zones, and endpoints, which are components of the AWS secure global infrastructure.

Regions, Availability Zones, and Endpoints

Use AWS regions to manage network latency and regulatory compliance. When you store data in a specific region, it is not replicated outside that region. It is your responsibility to replicate data across regions, if your business needs require that. AWS provides information about the country, and, where applicable, the state where each region resides; you are responsible for selecting the region to store data with your compliance and network latency requirements in mind. Regions are designed with availability in mind and consist of at least two, often more, Availability Zones.

Availability Zones are designed for fault isolation. They are connected to multiple Internet Service Providers (ISPs) and different power grids. They are interconnected using high speed links, so applications can rely on Local Area Network (LAN) connectivity for communication between Availability Zones within the same region. You are responsible for carefully selecting the Availability Zones where your systems will reside. Systems can span multiple Availability Zones, and we recommend that you design your systems to survive temporary or prolonged failure of an Availability Zone in the case of a disaster.

AWS provides web access to services through the AWS Management Console, available at *https://aws.amazon.com/console*, and then through individual consoles for each service. AWS provides programmatic access to services through Application Programming Interfaces (APIs) and command line interfaces (CLIs). Service endpoints, which are managed by AWS, provide management (“backplane”) access.

Alfresco Architecture

To help ensure high availability, this architecture deploys the Alfresco servers across two Availability Zones within a region. The “multi-AZ” feature is enabled for the Amazon RDS database, which is deployed in both Availability Zones in a master/slave configuration.

Amazon Virtual Private Cloud (Amazon VPC) creates a logically isolated networking environment that you can connect to your on-premises datacenters or have as a standalone environment.

Note: In Amazon VPC subnets, the first four IP addresses and the last IP address are reserved for networking purposes.

With Amazon VPC, you can create a deployment in which all of the Alfresco instances and Amazon RDS database instances are in private subnets, exposing only the Elastic Load Balancing listener and a NAT instance to the Internet.

The following diagram illustrates this architecture:

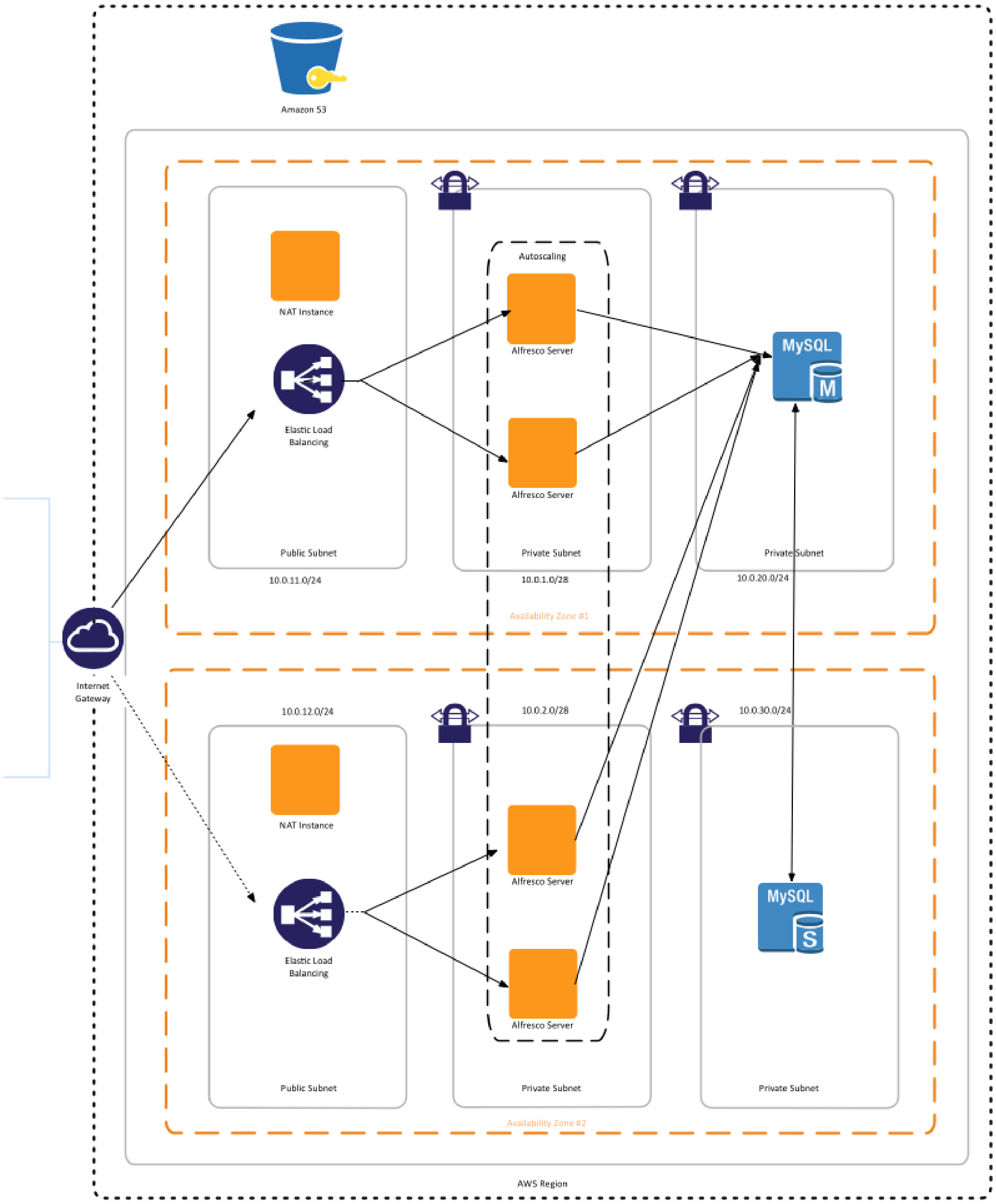


Figure 1: Alfresco Enterprise Reference Architecture

Note that Amazon VPC also gives you control over several networking aspects of a deployment. For example, when you create the VPC you define the overall IP address space of the VPC as well as the IP space that each of the subnets will use. This is important because Alfresco requires that the IP address of all the potential cluster members be defined in its configuration file. Because the subnet that the servers are launched into is defined by the user, you can control which IP space is used. As illustrated in the preceding diagram, the IP space of our subnets used for the Alfresco servers are set to 10.0.1.0/28 and 10.0.2.0/28, creating subnets with a usable IP range of 10.0.1.0-10.0.1.14 and 10.0.2.0-10.0.2.14 respectively. This allows us to pre-populate a reasonable number of potential IP addresses that the Alfresco instances should check for cluster members on in the Alfresco global configuration file.

Use the AWS CloudFormation Template to Deploy an Alfresco Cluster

This section explains the rationale behind the design of the architecture and describes the steps that the AWS CloudFormation template performs when it creates the infrastructure and configures the Alfresco servers.

The AWS CloudFormation template will perform three main tasks:

- Creating the AWS infrastructure
- Installing Alfresco and modifying configuration files
- Configuring the AWS Auto Scaling service

Creating the Infrastructure

First, we create a new Amazon VPC environment for the deployment. When you create a new VPC, you first must choose the IP addresses space the VPC will use.

We have chosen the default (10.0.0.0/16) and created six subnets across two Availability Zones. Each Availability Zone has three subnets.

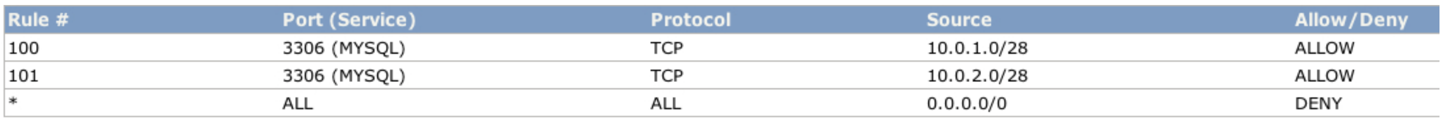
The subnets and their contents are detailed in the following table:

Subnet Type	IP Range	Contents
Public	10.0.11.0/24 10.0.12.0/24	NAT Instances
Private	10.0.1.0/28 10.0.2.0/28	Alfresco Servers
Private	10.0.20.0/24 10.0.30.0/24	Amazon RDS Instances

Table 1: Subnets, IP Ranges, and Contents

The NAT instances allow the Alfresco servers to access the Internet, including the AWS API endpoints, and they also serve as SSH administrative hosts. The administrative hosts are used to allow an administrator to SSH to the Alfresco instances in the private subnets. The “SSH From” parameter in the AWS CloudFormation template allows an administrator to limit the IP addresses that are permitted to SSH to the NAT instances.

Each of the subnets is configured with Network ACLs to permit only the required traffic for that subnet's purpose. For example, the Amazon RDS subnets are configured to allow only traffic from the Alfresco server subnets on the MySQL ports and deny all other traffic. This is illustrated in the following table.



Rule #	Port (Service)	Protocol	Source	Allow/Deny
100	3306 (MYSQL)	TCP	10.0.1.0/28	ALLOW
101	3306 (MYSQL)	TCP	10.0.2.0/28	ALLOW
*	ALL	ALL	0.0.0.0/0	DENY

Table 2: Subnets and Traffic

You can find a detailed description of all the subnet ACLs in the *Security Group and Network ACL Configuration* section in the implementation guide.

Configure the Database

Alfresco supports several different database options, including PostgreSQL, MySQL, Oracle, Microsoft SQL Server and DB2. In this whitepaper, we focus on MySQL.

Rather than requiring you to install, configure, and manage the database server, we use Amazon RDS to provide a managed MySQL database. To help ensure high availability, we enable the Amazon RDS Multi-AZ feature, which will deploy an Amazon RDS instance in both of the Availability Zones and will be referenced using a DNS name to allow for failover to the slave instance in the event the master fails.

Alfresco uses a database to store metadata information about objects while the files themselves are placed in the content store. In this case we will use Amazon S3 for to store the data. The database typically does not need to be very large, nor does it require a very large instance type.

The default values we've provided in the AWS CloudFormation template create a 5 GB database of type db.m1.small. These values are appropriate for a small- to mid-sized deployment. Depending on the size of your deployment, you might need to modify these default values to increase the database size and use a larger instance type, but we recommend that you start with the default values. If you outgrow the default settings, you can easily re-size your Amazon RDS database by following the steps described in this article: http://aws.amazon.com/articles/Amazon-RDS/2936.

Install Alfresco

The Alfresco software is installed on an Amazon EC2 instance through a Linux binary installer. The installation involves only a few user inputs, which the AWS CloudFormation template passes to the installer through an options file to automate the installation.

After the installer has completed, you must update the configuration files with settings for both the shared storage and clustering components.

Configure storage

In order to leverage Amazon S3 for your content store the Alfresco Amazon S3 connector must be installed. This connector is an Alfresco Module Package (AMP) and is installed using the AMP installation process provided by Alfresco as part of the installation steps that the AWS CloudFormation template performs.

The AWS CloudFormation template creates an IAM user and associated API credentials with permissions to call the Amazon S3 API commands necessary for the connector to function. These credentials and the bucket name are added to the alfresco-global.properties file after installation.

Note: You must create the bucket before you start the Alfresco server. The Amazon S3 connector does not support automatically creating a bucket if the bucket listed does not exist.

For the complete IAM policy for this user, along with other IAM policies used throughout this deployment, see the IAM Policy section of the accompanying Implementation Guide.

Set Up the Cluster

Setting up clustering of Alfresco Enterprise in Amazon EC2 involves modifying the Alfresco configuration files and configuring Ehcache and Hazelcast. Ehcache is an open source Java distributed cache that is used to improve performance, and Hazelcast is an open source data distribution and clustering package.

Hazelcast has several methods it can use to identify other nodes in a cluster. In Amazon EC2, Hazelcast must be configured to identify members based on their Amazon EC2 security group membership. To enable Hazelcast to query the AWS APIs to identify an instance's security group, the application requires a set of API keys.

In the AWS CloudFormation template we create an IAM user with permissions to describe instances, allowing it to identify which instances use the specified security group. The IAM API keys, the security group that is created for the Alfresco servers, and the cluster name and password are all added to the Hazelcast configuration file after the installation has completed.

For a complete list of configuration changes required to enable Ehcache and Hazelcast, see the accompanying Implementation Guide.

In addition to configuring Ehcache and Hazelcast, you must define the set of IP addresses that a new instance should check when looking for existing members in the cluster. Because the IP addresses of the Alfresco instances are dynamically assigned, you must include all of the potential IP addresses in the subnet. To limit the number of potential IP addresses that need to be checked, the Alfresco subnet was created with a CIDR block of /28. This leaves sufficient room for the application to scale while keeping the number of IPs that need to be checked to a reasonable number.

One key decision in how an environment in AWS is set up is to determine the amount of configuration that is performed dynamically, often referred to as bootstrapping, and what is pre-configured as part of the AMI. The full set of steps to create a new instance for the cluster, including the installation of the Alfresco binaries, takes approximately 12-15 minutes to complete and have a new node ready to accept requests. While this process can be scripted and performed in an automated fashion after a new instance is created, the amount of time it takes to install and configure the new cluster node is too long to be effective in an autoscaling environment. To allow the deployment to quickly scale up, the final step in configuring the cluster is to create a new AMI from the currently running instance. This AMI will be used to configure the autoscaling launch configuration. After the autoscaling configuration is complete, this setup instance is no longer needed and will be terminated.

Configure Auto scaling

The Auto scaling configuration creates instances in two Availability Zones, which are specified as parameters to the AWS CloudFormation template. The AMI ID from the new AMI created in the last step of the previous section is used when the Auto scaling Launch Configuration is created. Because this AMI ID hasn't been generated before the AWS CloudFormation template is launched, the auto scaling configuration is performed using a Python script sourced from an Amazon S3 instance.

When configuring Auto scaling you must specify the minimum, maximum, and desired number of instances. By default we will use a minimum of two, a maximum of six, and a desired number of instances also at two. With a maximum of six instances deployed across two Availability Zones, a deployment should be able to support approximately 600 concurrent users (although this is highly dependent on intended real-world utilization). We also create scaling policies based on the CPU utilization of the Alfresco instances as well as the latency from the elastic load balancer to the Alfresco instance.

The default scaling policies will add two instances when the average CPU utilization exceeds 60 percent or if the latency from the elastic load balancer to the Alfresco instance exceeds one second over two periods that are 60 seconds apart. A single instance will be removed if the average CPU utilization falls below 30 percent over two 60-second periods and the current number of instances exceed the minimum and desired number of instances.

The complete set of Python commands to configure Auto scaling is detailed in the accompanying Implementation Guide.

Auto scaling integrates with the Elastic Load Balancing service, and instances that are created by the Auto Scaling service are automatically added to the elastic load balancer. The elastic load balancer is created with a health check that will periodically check the Alfresco Share URL. If an instance stops responding to the health checks, it will be removed from the load balancer and replaced by the Auto Scaling service.

Security Group and Network Access Control List (ACL) Configuration

The deployment in this whitepaper uses four different security groups and three Network ACLs.

The security groups are as follows:

- Elastic Load Balancing
- Alfresco
- NAT Instances
- Amazon RDS

The Network ACLs are as follows:

- Amazon RDS
- Alfresco
- NAT Instances

The following tables detail the rules for these groups and lists and describe the traffic that the rule is designed to allow.

Elastic Load Balancing Security Group

Direction	Source or Destination	Protocol/Port	Description
Inbound	0.0.0.0/0	TCP/80	Allow inbound HTTP requests to the elastic load balancer.
Inbound	0.0.0.0/0	TCP/8080	Allow inbound SharePoint traffic on 8080.
Outbound	10.0.1.0/28	TCP/7070	SharePoint listener on Alfresco Instances in Availability Zone 1
Outbound	10.0.2.0/28	TCP/7070	SharePoint listener on Alfresco Instances in Availability Zone 2.

Outbound	10.0.1.0/28	TCP/8080	HTTP listener on Alfresco instances in Availability Zone 1.
Outbound	10.0.2.0/28	TCP/8080	HTTP listener on Alfresco instances in Availability Zone 2.

Alfresco Security Group

Direction	Source or Destination	Protocol/Port	Description
Inbound	Elastic load balancer	TCP/8080	Allow inbound HTTP requests from the elastic load balancer.
Inbound	Elastic load balancer	TCP/7070	Allow inbound SharePoint traffic from the elastic load balancer.
Inbound	10.0.1.0/28 10.0.2.0/28	TCP/5700-5710	Allow Hazelcast traffic.
Inbound	10.0.1.0/28 10.0.2.0/28	TCP/5800-5810	Alfresco RMI.
Inbound	10.0.1.0/28 10.0.2.0/28	TCP/7800	JGroups cluster port.
Inbound	<NAT Instances>	TCP/22	Allow SSH only from either of the two NAT instances.
Outbound	0.0.0.0	TCP/0-65535	All outbound.

NAT Instances Security Group

Direction	Source or Destination	Protocol/Port	Description
Inbound	<SSH From Parameter>	TCP/22	Allow SSH from IP range specified.
Inbound	10.0.0.0/16	TCP/80	Accept HTTP traffic from instances in the Amazon VPC.

Inbound	10.0.0.0/16	TCP/443	Accept HTTPS traffic from instances in the Amazon VPC.
Outbound	0.0.0.0/0	TCP/80	Outbound HTTP traffic.
Outbound	0.0.0.0/0	TCP/443	Outbound HTTPS traffic.
Outbound	10.0.1.0/28 10.0.2.0/28	TCP/22	Outbound SSH to Alfresco instances.

Amazon RDS Security Group

Direction	Source or Destination	Protocol/Port	Description
Inbound	Alfresco Security Group	TCP/3306	Allow MySQL traffic from Alfresco instances
Outbound	0.0.0.0/0	ALL	Allow outbound

Amazon RDS Subnet Network ACL

Direction	Source or Destination	Protocol/Port	Description
Inbound	10.0.1.0/28 10.0.2.0/28	TCP/3306	Allow MySQL traffic from Alfresco subnets.
Inbound	0.0.0.0/0	ALL	Deny all.
Outbound	0.0.0.0/0	TCP	Allow all TCP.

Alfresco and NAT Subnet Network ACL

Direction	Source or Destination	Protocol/Port	Description
Inbound	0.0.0.0/0	TCP	Allow all TCP.
Outbound	0.0.0.0/0	TCP	Allow all TCP.

IAM Policies

Two IAM roles and one IAM user are created by the AWS CloudFormation template that comes with this whitepaper. The IAM user is used by the Amazon S3 connector and Hazelcast. (Neither supports IAM roles). One IAM role is used by the initial instance from which the custom AMI with Alfresco installed and configured is created, and the second IAM role is used by the Alfresco instances that are in production.

IAM User Policy

```
{"Statement":[{
"Resource":"*",
"Action":"cloudformation:DescribeStackResource",
"Effect":"Allow"},
{"Resource":"*",
"Action":"EC2:Describe*",
"Effect":"Allow"},
{"Resource":"*","Action":"cloudwatch:PutMetricData",
"Effect":"Allow"},
{"Resource":"arn:aws:s3:::<Bucket Name>/*",
"Action":["s3:GetObject","s3:PutObject","s3:DeleteObject","s3:ListBucket"
,"s3:Get*","s3:List*"],
"Effect":"Allow"},
{"Resource":"*","Action":["s3:List*"],
"Effect":"Allow"}]}
```

Setup Role

```
{"Statement":[{
"Resource":"*",
"Action":"cloudformation:DescribeStackResource",
"Effect":"Allow"},
{"Resource":"*",
"Action":["EC2:Describe*","EC2:CreateImage","ec2:TerminateInstances"],
"Effect":"Allow"},
{"Resource":"*","Action":"elasticloadbalancing:DescribeLoadBalancers",
"Effect":"Allow"},
{"Resource":"*","Action":["autoscaling:create*","autoscaling:put*","autos
caling:DescribePolicies"],
"Effect":"Allow"},
{"Resource":"arn:aws:iam::<Account Number>:role/<Alfresco Role>",
"Action":"iam:PassRole",
"Effect":"Allow"},
{"Resource":"*",
"Action":["cloudwatch:PutMetricData","cloudwatch:EnableAlarmActions","clo
udwatch:PutMetricAlarm"],
"Effect":"Allow"},
{"Resource":"arn:aws:s3:::<Bucket Name>/*",
"Action":["s3:GetObject","s3:PutObject","s3:DeleteObject","s3:ListBucket"
,"s3:Get*","s3:List*"],
"Effect":"Allow"},
{"Resource":"*",
"Action":["s3:List*"],
"Effect":"Allow"}]}
```

Alfresco Role

```
{"Statement":[{
"Resource":"*",
"Action":"cloudformation:DescribeStackResource",
"Effect":"Allow"},
{"Resource":"*",
"Action":"EC2:Describe*",
"Effect":"Allow"},
{"Resource":"*",
"Action":"cloudwatch:PutMetricData",
"Effect":"Allow"},
{"Resource":"arn:aws:s3:::<Bucket Name>/*",
"Action":["s3:GetObject","s3:PutObject","s3:DeleteObject","s3:ListBucket"
,"s3:Get*","s3:CreateBucket","s3:List*"],
"Effect":"Allow"},
{"Resource":"*",
"Action":["s3:List*"],
"Effect":"Allow"}]}
```

Conclusion

This paper describes a common deployment scenario for Alfresco Enterprise and how it can be deployed in the AWS cloud environment in a manner that is highly available, can scale up and down and provides a storage option that is both highly durable and low cost. By leveraging deployment services such as AWS CloudFormation to create a deployment you also are assured that the results are easily portable to other regions and will have a repeatable and known output every time.

Further Reading

1. AWS Alfresco Partner Page: http://www.aws-partner-directory.com/PartnerDirectory/PartnerDetail?id=7609
2. Alfresco on AWS: http://www.alfresco.com/aws
3. Alfresco Enterprise on AWS : Implementation Guide : http://media.amazonwebservices.com/AWS_Alfresco_Enterprise_Implementation_Guide.pdf