Multi-Region Application Architecture

AWS Implementation Guide

George Bearden

Eric Quinones

June 2020

Table of Contents

About This Guide

This implementation guide discusses architectural considerations and configuration steps for deploying the Multi-Region Application Architecture solution in the Amazon Web Services (AWS) Cloud. It includes links to an AWS CloudFormation template that launches and configures the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, and DevOps professionals who have practical experience architecting in the AWS Cloud.

Overview

Many Amazon Web Services (AWS) customers have business requirements that require expedient recovery from regional failure with little-to-no application data loss. To help demonstrate a fault-tolerant application with easy failover to a backup region, AWS offers the Multi-Region Application Architecture solution.

This solution leverages Amazon Simple Storage Service (Amazon S3) Cross-Region replication and Amazon DynamoDB Global Tables to asynchronously replicate application data between the primary and secondary AWS Regions. A sample photo-sharing web application is also deployed in each Region to serve as a visual demonstration of the solution's back-end layers and to verify that regional failover is working.

An application's Recovery Time Objective (RTO) and Recovery Point Objective (RPO) are important metrics when considering failover and disaster recovery scenarios. This solution allows for an RTO of a few seconds and an RPO of 15 minutes for application data stored in Amazon S3 and Amazon DynamoDB.

Cost

You are responsible for the cost of the AWS services used while running this solution. The total cost for running this solution depends on the number of users you create, the number of images you store and how many times they are accessed from the application.

The cost for running this solution with default settings in the primary US East (N. Virginia) Region and Asia Pacific (Tokyo) as the secondary Region is approximately **$10.00 per month**. This estimate includes considerations for photos stored in Amazon S3 (totaling 1GB in size) and comments stored in Amazon DynamoDB replicated to the secondary Region.

Prices are subject to change. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Architecture Overview

Deploying this solution builds the following environment in the AWS Cloud.

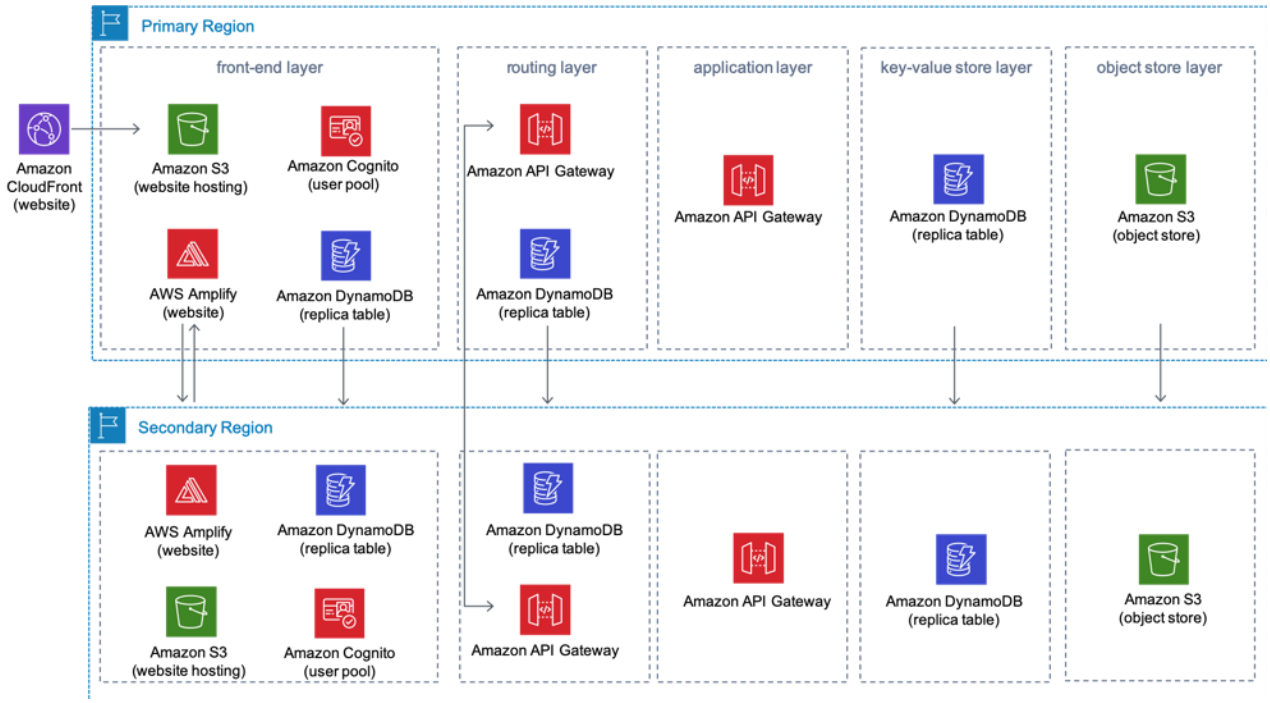

Figure 1: Multi Region Application Architecture on AWS

The AWS CloudFormation template deploys an AWS Amplify sample web application hosted in an Amazon Simple Storage Service (Amazon S3) bucket in both the primary and secondary AWS Regions; an Amazon CloudFront distribution to deliver the web application to users; Amazon Cognito user pools are deployed in each Region to enable users to sign in to the web application; and Amazon DynamoDB Global Tables to replicate data from the primary to the secondary Region.

> **Note:** In the AWS console, you can manually configure Origin Failover on the solution's CloudFront to serve the application from the secondary Region. For more information, see CloudFront Origin Failover in the *Amazon CloudFront Developer Guide*.

When the web application is loaded, it queries the solution's routing layer for the current state of the application (`active`, `fenced`, `failover`), and configures AWS Amplify to target the solution's resources in the correct Region. The state of the application is also retrieved when the user uploads a new photo or adds a comment. If the application is in a

fenced or failover state, these actions are disabled by the application and a message is displayed to the user. For more information, see Application States.

Solution Components

Application States

The Multi-Region Application Architecture solution application is automatically deployed with three states. For information on changing the state, see Appendix A.

Active State: This state indicates the application is operating as normal. When the web application is loaded, it will target back-end resources in the primary Region. Photo uploads and comments are enabled.

Failover State: This state indicates the primary Region is unavailable. When the web application is loaded, it will target back-end resources in the secondary Region. Photo uploads and comments are disabled but will still be viewable.

Fenced State: This state indicates that the application is in a read-only mode. When the web application is loaded, it will target back-end resources in the primary Region. Photo uploads and comments are disabled but will still be viewable.

Front-End Layer

The front-end layer creates a React with AWS Amplify sample photo-sharing web application that can be used as a visual demonstration of the back-end layers of this solution and to verify that regional failover is working. The web application is deployed as a static website hosted in an Amazon Simple Storage Service (Amazon S3) bucket in the primary and secondary Regions. To help reduce latency and improve security, this solution includes an Amazon CloudFront distribution with an origin access identity. This is a special CloudFront user that helps restrict access to the solution's website bucket contents.

The application supports user sign-in, image uploads and sharing, and commenting on images. Amazon Cognito is used to authenticate users on sign-in and to authenticate the application layer. Images are uploaded to the Amazon S3 bucket in the primary Region and replicated to the secondary Region using Amazon S3 Cross-Region replication. Photo comments are stored in an Amazon DynamoDB global table.

> **Note:** Currently, this solution doesn't restrict access to photos using Amazon Cognito. All uploaded photos will be visible to all users of the application.

This layer also deploys Amazon Cognito user pools in the primary and secondary Regions. New users may only be added to the user pool using the Amazon Cognito console in the primary Region. New user sign-ups from the web application are disabled.

An AWS Lambda function is invoked using the Cognito Post Authentication Lambda Trigger. When a new user is confirmed in the primary Region's user pool, the Lambda function writes the user profile information such as the username and attributes to the Amazon DynamoDB global table in the primary Region, where it will subsequently be replicated to the secondary Region's replica table. In the secondary Region, a Lambda function is invoked by an Amazon CloudWatch Events rule every five minutes. This function then queries the front-end layer DynamoDB global table for users that haven't been imported into the secondary Region's user pool and imports them.

Once the user has been initially backed up to the secondary Region's user pool, further edits to the user's profile (attribute value changes, group memberships, etc.) are not replicated. Additionally, user profiles are only replicated from the primary Region to the secondary Region. Any changes to the user pool in the secondary Region are not replicated to the primary Region. In the secondary Region, new user sign-ups from the web console are disabled and a Cognito Pre Authentication Lambda Trigger is configured to prevent users from being created using the Cognito console.

> **Note:** User passwords are not imported when users are backed up to the secondary Region's user pool. Users will need to change their password the first time they sign into the application when it is in failover mode. When the application is switched back to either active or fenced mode, users will need to use their original password or reset it.

Routing Layer

The routing layer deploys an Amazon API Gateway with proxy integration to an Amazon DynamoDB global table the primary and secondary Regions. The DynamoDB global table will store the application's state. When the front-end layer's web application is loaded, it requests the application's state from the API Gateway in the primary Region. If the primary Region is unavailable, the web application will query the API Gateway in the secondary Region. When the current state is received, the application will load the necessary configuration settings in AWS Amplify, and point to resources in the primary or secondary Region. An Amazon CloudFront domain is configured as the allowed origin for the API Gateway.

Application Layer

The application layer is the back-end code for the application. This layer deploys an Amazon API Gateway with proxy integration to a DynamoDB global table that stores user comments

on photos in the primary and secondary Regions. The deployed Amazon Cognito user pools are used as authentication for the API Gateway in each Region.

When a photo is selected in the sample application, a GET request is created to retrieve the comments for the selected photo. Users can also post their own comments on uploaded photos.

Key-Value Store Layer

The key-value store layer deploys Amazon DynamoDB global tables in the primary and secondary Regions to store user comments on photos. The application layer's API Gateway accesses the comments directly using a proxy integration to DynamoDB.

Object Store Layer

The object store layer deploys an Amazon S3 bucket in the primary and secondary Regions to store images uploaded by users. When the web application is in an **active** state and configured to use resources in the primary Region, uploaded photos are stored in the Amazon S3 `object store` bucket in the primary Region. Then, Amazon S3 Cross-Region replication asynchronously replicates the object in the Amazon S3 `object store` bucket in the secondary Region.

> **Note:** Objects are replicated from the primary Region to the secondary Region only. The application does not allow for photos to be uploaded when the application state is in **failover**. Any objects manually added to the Amazon S3 `object store` bucket in the secondary Region will not be replicated to the primary Region.

Design Considerations

AWS CloudFormation Execution Role

The Multi-Region Application Architecture solution leverages AWS CloudFormation StackSets to deploy the solution's resources in both the primary and secondary Regions and automatically assumes a role with the necessary permissions. Each of the solution's layers will create an AWS Identity and Access Management (IAM) policy that enable the necessary resources to be created. The policies are bundled into the `CloudFormation Execution Role`, which is assumed when the StackSet is created.

Cross-Region Replication

This solution leverages Amazon Simple Storage Service (Amazon S3) Cross-Region replication to asynchronously replicate photos uploaded in the Amazon S3 `object store` bucket in the primary Region to the Amazon S3 `object store` bucket in the secondary Region. Currently, photo replication is only one-way so objects uploaded in the secondary Region's bucket will not be replicated to the primary Region's bucket.

> **Note:** The application will prevent image uploads when the application is in a **failover state** or if the application is pointing to the secondary Region's resources after the state is moved to active.

Amazon DynamoDB Global Tables

This solution leverages Amazon DynamoDB global tables to asynchronously replicate the state of the application used in the routing layer and the photo comments in the key-value store layer. The `CreateGlobalTable` API is called to create a replication relationship between the DynamoDB tables in each Region.

RTO/RPO

This solution allows for a Recovery Time Objective (RTO) of a few seconds and a Recovery Point Objective (RPO) of 15 minutes. Profiles for new confirmed users in the primary Region's user pool are replicated to the secondary region every five minutes. Amazon S3 Cross-Region replication is leveraged to replicate the majority of objects uploaded in 15 minutes. Added comments are stored in a DynamoDB global table. In the global table, a newly written item is propagated to all replica tables within seconds.

Failover from the primary Region to the secondary Region requires updating the application's state, which is stored in an item in a DynamoDB global table. Once the application's state has been updated, the web application will target resources in the other Region as soon as the page is refreshed.

AWS CloudFormation Template

This solution uses AWS CloudFormation to automate the deployment of the Multi-Region Application Architecture solution. It includes the following AWS CloudFormation template, which you can download before deployment:

View template

multi-region-application-architecture.template: Use this template to launch the solution and all associated components. The default configuration deploys Amazon Simple Storage Service (Amazon S3) buckets, Amazon DynamoDB tables, AWS Lambda functions, Amazon API Gateways, AWS Identity and Access Management (IAM) roles and policies, an AWS CloudFront distribution, Amazon Cognito User Pools, and AWS CloudFormation StackSets, but you can also customize the template based on your specific needs.

Automated Deployment

Before you launch the automated deployment, please review the architecture, configuration, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy Multi-Region Application Architecture into your account.

Time to deploy: Approximately 20 minutes

Launch the Stack

This automated AWS CloudFormation template deploys Multi-Region Application Architecture.

> **Note**: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button to the right to launch the `multi-region-application-architecture` AWS CloudFormation template.
   You can also download the template as a starting point for your own implementation.

   Launch Solution

2. The template is launched in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the region selector in the console navigation bar.
3. On the **Create stack** page, verify that the correct template URL shows in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Admin Name	*<Requires input>*	Enter the user name for the admin Amazon Cognito user account that will be created.
Admin Email	*<Requires input>*	Enter the email address for the admin Amazon Cognito user account that will be created.
Admin Phone Number	*<Requires input>*	Enter the phone number for the admin Amazon Cognito user account that will be created. **Note:** The number must be in the following format +12345678999.

Parameter	Default	Description
Secondary Region	*<Requires input>*	The secondary AWS Region that will be used in the event of an application failover. **Note:** The primary Region will default to the Region where the stack is deployed.

6. Choose **Next.**
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately 20 minutes.

> **Note:** This solution includes multiple AWS CloudFormation `custom-resource` AWS Lambda functions, which run only during initial configuration or when resources are updated or deleted.
>
> When running this solution, these functions are inactive. However, do not delete these functions as they are necessary to manage associated resources.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

IAM Roles

AWS Identity and Access Management (IAM) roles enable customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles that grant the AWS Lambda functions access to the other AWS services used in this solution.

Amazon CloudFront

This solution deploys a static website hosted in an Amazon S3 bucket. To help reduce latency and improve security, this solution includes an Amazon CloudFront distribution with an

origin access identity, which is a special CloudFront user that helps restrict access to the solution's website bucket contents. For more information, see Restricting Access to Amazon S3 Content by Using an Origin Access Identity.

Amazon API Gateway

Amazon API Gateways deployed in the routing layer are called by the sample web application to retrieve the current state (`active`, `fenced`, `failover`) and subsequently configure AWS Amplify to target the solution's resources in the correct Region. Since this API must be called prior to a user signing into the application, the Cognito user pools are not configured as authorizers.

The application layer Amazon API Gateways are configured to use the deployed Amazon Cognito user pools in each Region as authorizers to verify that only users signed in to the application can post comments.

All API Gateways in the solution are configured to use the Amazon CloudFront domain deployed by the solution as the value for the `Access-Control-Allow-Origin` header.

Photo-Sharing Demo Application

When a user has signed into the photo-sharing application, they will have access to view all photos that have been added by all users. The application does not allow for private photo uploads or for only sharing photos with specific users. Currently, fine-grained access to Amazon S3 objects using Amazon Cognito is not configured for this demo application.

Additional Resources

- AWS Lambda
- Amazon Simple Storage Service
- Amazon CloudWatch
- Amazon DynamoDB
- Amazon API Gateway
- AWS CloudFormation
- AWS Identity and Access Management
- Amazon CloudFront
- AWS Amplify

Appendix A: Changing the Application State

The Multi-Region Application Architecture solution's application state is stored in the Amazon DynamoDB `AppConfigTable` table, created by the routing layer. When the solution is deployed, an `AppId` for the sample photo-sharing application is generated by an AWS Lambda-backed Custom Resource and is used as the key for the corresponding item in the `AppConfigTable`. Updating the `state` attribute of this item in the `AppConfigTable` will change the application's state. Accepted values are `active`, `fenced`, and `failover`. Attributes are case sensitive.

As an alternative to manually editing the item in DynamoDB, the routing layer will also deploy a Lambda `AppStateUpdaterLambda` function. To change the state of the application using this function, invoke it with an event payload similar to the below. The values for `SampleApplicationAppId` and `AppConfigGlobalTableName` can be found in the **Outputs** tab of the solution's main CloudFormation stack.

```
{
  "AppId": "SampleApplicationAppId",
  "TableName": "AppConfigGlobalTableName",
  "NewState": "active/fenced/failover"
}
```

Appendix B: Collection of Operational Metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Version:** The version of the deployed solution
- **Timestamp:** The UTC formatted timestamp of when the event occurred
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Event Name:** A value of `STATE_UPDATED` is used to indicate the state of the application has changed
- **Event Value:** A value representing the new state (`active`, `fenced`,`failover`) for the application

Note that AWS will own the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, modify the AWS CloudFormation template mapping section as follows:

```
Solution:
    Metrics:
      SendAnonymousData: Yes
```

to

```
Solution:
    Metrics:
      SendAnonymousData: No
```

Source Code

You can visit our solution GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change
June 2020	Initial release

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided "as is" without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.