FHIR Works on AWS

Implementation Guide

FHIR Works on AWS: Implementation Guide

Table of Contents

Implement a software toolkit to give developers the ability to add serverless FHIR healthcare APIs to their own applications and connect their healthcare applications to others

Publication date: *December 2020 (last update (p. 21): November 2021)*

This implementation guide describes architectural considerations and configuration steps for deploying FHIR Works on AWS 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 architects, developers, DevOps, and data analysts who have practical experience architecting in the AWS Cloud.

Overview

The FHIR Works on AWS solution helps software engineers at independent software vendors, system integrators, and healthcare information technology teams to enhance their own products. Using this solution, they can transfer medical records to both mobile devices and web portals in minutes rather than days or hours by integrating with the Fast Healthcare Interoperability Resources (FHIR) standard APIs. The FHIR standard was developed by Health Level Seven International (HL7) to make it easier to exchange health data between software systems: such as practice management, electronic health records, and billing; data standards; and data exchange interfaces. It uses a serverless FHIR API that supports FHIR resource types and operations to help healthcare providers leverage the FHIR standard to manage healthcare records. It is an example implementation, designed to be extensible.

This solution includes the following new features:

- Reading and writing Amazon OpenSearch Service documents from aliases instead of indexes.
- Using Cognito ID token instead of the access token to authorize requests.
- Using pooled infrastructure model for multi-tenancy.

For information on upgrade instructions related to the new features, see the Upgrade instructions section.

Upgrade instructions

- FHIR Works on AWS now reads/writes Amazon OpenSearch Service documents from aliases instead of indexes. This change simplifies performing re-indexing operations without downtime. Aliases are automatically created when resources are written to Amazon OpenSearch Service, but read operations might fail for existing deployments if the aliases do not exist already.

  Mitigation: After upgrade, send one update/create request on each resource type to get the aliases created.
- The Cognito ID token is now used instead of the access token to authorize requests.

  Mitigation: After upgrade, use AWS CLI to request new ID token, then paste the token into Postman directly. Token can no longer be requested through Postman. For more information, see Accessing the FHIR API (p. 15).

Cost

You are responsible for the cost of the AWS services used while running this solution. As of February 2021, the estimated cost for running the FHIR Works on AWS with 10 GB of structured data, 1 TB of unstructured (binary) data, and 500 transactions per day in the US East (N. Virginia) Region is approximately **$140.00 per month**.

Cost per month

This solution uses the following resources that are billed on a monthly basis.

AWS service	Quantity	Cost
Amazon API Gateway	500 requests per day	$0.03

AWS service	Quantity	Cost
Amazon DynamoDB	10GB of storage and 250 read and 250 write requests per day	$4.02
Amazon OpenSearch Service (OpenSearch Service)	10GB of EBS storage, 1 t3.medium.elasticsearch instance	$110.21
Amazon Simple Storage Service (Amazon S3)	1 TB of storage; 5 reads and 5 writes per day	$23.00
AWS Key Management Service (AWS KMS)	4 customer master keys and 500 cryptographic operations per day	$4.05
AWS Lambda	500 Lambda invocations per day	$0.16
Amazon Cognito	1,000 monthly active users	No cost
Total monthly cost:		**$141.47**

Prices are subject to change. For full details, refer to the pricing webpage for each AWS service used in this solution.

Architecture overview

Deploying this solution with the default parameters builds the following environment in the AWS Cloud.

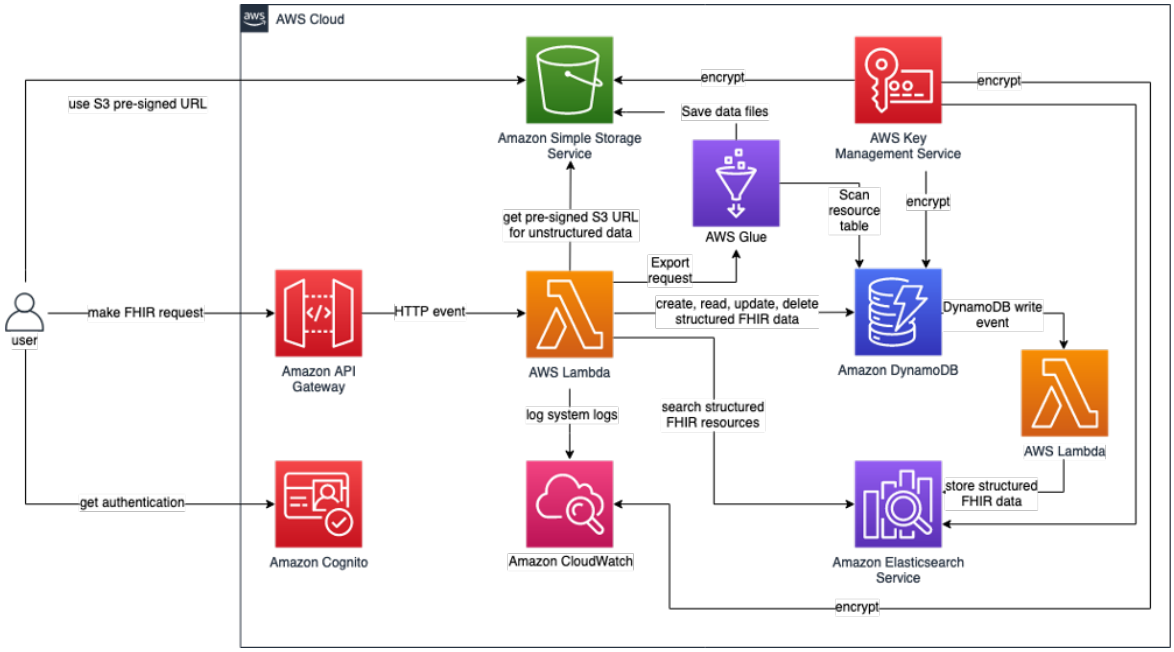

Figure 1: FHIR Works on AWS architecture

The AWS CloudFormation template deploys the serverless infrastructure necessary to serve FHIR HTTP requests. This includes the following:

- One Amazon Cognito user pool, domain, and client to authenticate the requesting user's identity and determine which group the user is in.
- One Amazon API Gateway to route the request to a Lambda function. The API Gateway also has an Amazon Cognito authorizer to confirm the request has a valid ID token created by this stack's Amazon Cognito user pool.
- Two AWS Lambda functions. One to process FHIR requests, routing them to the correct persistence layer, either Amazon Simple Storage Service (Amazon S3) for unstructured FHIR resources, Amazon DynamoDB for create, read, update, delete (CRUD) operations or Amazon OpenSearch Service (OpenSearch Service) for all search operations. Another Lambda function to read updates from the FHIR resource DynamoDB table and stream those changes to OpenSearch Service.
- One DynamoDB table to store all structured FHIR resources, which after a write operation streams the update to the OpenSearch Service domain.
- One OpenSearch Service domain to support FHIR searching requests.
- One Amazon S3 bucket to hold FHIR binary resources, such as unstructured data, X-rays, and raw notes.
- Four AWS Key Management Service (AWS KMS) keys to encrypt DynamoDB, Amazon S3, Amazon CloudWatch Logs, and the OpenSearch Service domain.

Requests to and responses from the APIs are logged in Amazon CloudWatch and can be optionally archived to Amazon S3 in order to optimize cost.

Components

Authentication mechanism

FHIR Works on AWS uses an Amazon Cognito user pool for Amazon API Gateway authentication. Once authenticated, Amazon Cognito provides a JSON Web Token (JWT) to the requestor that is provided with all subsequent API requests. If a valid JWT is not provided, the API request fails and returns a HTTP 403 Forbidden response.

After a request is authenticated in Amazon API Gateway, an AWS Lambda function provides further authorization logic; the *groups* claim in the JWT specifies which group the requestor is from. Based on the group, the requester has the following permissions:

- **Practitioner group member:** permission to perform any operation on any resource.
- **Non-practitioner group member:** permission to read all financial resources, such as *Invoice* or *ExplanationOfBenefit*.
- **Auditor group member:** permission to read the *Patient* resource.

Resource modification and search mechanism

After a create, update or delete request has been authorized and is within the AWS Lambda function, the change is persisted to the `resource-db-dev` table in Amazon DynamoDB. This table is connected to a DynamoDB stream, which pushes changes to the Amazon OpenSearch Service domain. Amazon OpenSearch Service is used to support search functionality in FHIR Works on AWS. Refer to the capability statement for search parameters supported.

FHIR bundle mechanism

Bundles are a container for a collection of resources. This container can contain many requests for the FHIR server (such as a request that writes 10 resources at once instead of calling the FHIR server 10 separate times). These bundles can be handled as either batches or transactions, however FHIR Works on AWS only supports transactions. Transactions require all requests within the bundle to succeed or everything rolls back to the initial state prior to bundle submission.

With this transactional requirement, this solution supports locking on the Amazon DynamoDB table. To support locking in DynamoDB an additional status value is added to each resource to indicate if the bundle has completed the transaction.

FHIR binary resource mechanism

This solution supports unstructured data, which constitutes the binary FHIR resource and represents the data of a single raw artifact as digital content accessible in its native format. A binary resource can contain any content, whether text, image, pdf, zip archive, etc. They are stored in the `fhirbinarybucket` Amazon S3 bucket and are indexed via a binary resource entry in the Amazon DynamoDB table. Binary resources cannot be searched.

This solution handles binary resources by using Amazon S3 `getPresignedUrl` APIs and vending that URL to the requestor, to which they can then upload the file. The following workflow outlines the activities upon receipt of a *CreateBinary* request:

- Amazon API Gateway authorizes the request and sends the request to AWS Lambda.
- Lambda validates whether the user is in an appropriate group.
- Lambda writes the Binary metadata to Amazon DynamoDB.
- Lambda returns an Amazon S3 pre-signed URL allowing the customer to upload directly to Amazon S3.
- The customer uses the pre-signed URL and uploads the file to Amazon S3.

This Amazon S3 pre-signed URL approach is outside of the FHIR specification, but corresponds with the Bulk Data Implementation Guide specification, which remains at the Standard for Trial Use 1 (STU1) level of maturity. Refer to HL7 balloting levels for more information about the STU level.

FHIR bulk data access mechanism

Bulk Export allows you to export your data from DDB to S3. We currently support the system-level export and group export. To test this feature on FHIR Works on AWS, you can make API requests using the `Fhir.postman_collection.json` file by following these steps:

1. In the FHIR examples collection, under the Export folder, use **GET System Export** request to initiate an export request.
2. In the response, check the **Content-Location header** field for the URL. The URL is in the *<base-url>*/$export/*<jobId>* format.
3. To get the status of the export job, in the Export folder, use the **GET System Job** status request. Enter the job ID value from step 2 in that request.
4. Check the response from **GET System Job** status. If the job is in progress, the response header displays the **x-progress: in-progress** field. Keep polling that URL every 10 seconds until the job is complete. Once the job is complete, you'll get a JSON body with the pre-signed *S3* URLs of your exported data.
5. Download the exported data using those URLs.

   Example:

```
{
    "transactionTime": "2021-03-29T16:49:00.819Z",
    "request": "https://xyz.execute-api.us-west-2.amazonaws.com/$export?
_outputFormat=ndjson&_since=1800-01-01T00%3A00%3A00.000Z&_type=Patient",
    "requiresAccessToken": false,
    "output":
    [
        {
            "type": "Patient",
            "url": "https://fhir-service-dev-bulkexportresultsbucket-.com/abc"
        }
    ],
    "error": []
}
```

   Note
   To cancel an export job, use the**Cancel Export Job** request in the Export folder located in the Postman collections.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit AWS Cloud Security.

IAM roles

AWS Identity and Access Management (IAM) roles allow customers to assign granular access policies and permissions to services and users in the AWS Cloud. This solution creates an IAM role, `serverlessApiGatewayCloudWatchRole`, to grant access to the solution's Amazon API Gateway to send logs to Amazon CloudWatch.

Design considerations

Regional deployments

This solution uses Amazon Cognito, which is not currently available in all AWS Regions. You must launch this solution in an AWS Region where Amazon Cognito is available. For the most current availability by Region, refer to the AWS Regional Services List.

Multi-tenancy

Multi-tenancy allows a single FHIR Works on AWS stack to serve as multiple FHIR servers for different tenants. FHIR Works on AWS uses a pooled infrastructure model for multi-tenancy. This means that all tenants share the same infrastructure (Amazon DynamoDB tables, Amazon S3 buckets, Amazon OpenSearch Service cluster, etc.), but the data is logically partitioned to ensure that tenants are prevented from accessing another tenant's resources.

Turn on multi-tenancy

To turn on multi-tenancy, set the `EnableMultiTenancy` parameter to `true` when deploying the solution. When turned on, `/tenant/<tenantId>/` is appended to the FHIR server URL. For example, a client with `<tennantId>` as `tenantA` would use the following URL to search for patients:

```
GET /tenant/<tenantID>/Patient
```

```
GET https://1234567890.execute-api.us-west2.amazonaws.com/dev/tenant/tenantA/Patient
```

Turning on this setting provides each tenant a unique FHIR server-based URL.

> **Note**
> Updating an existing (single-tenant) stack to enable multi-tenancy is a breaking change. Multi-tenant deployments use a different data partitioning strategy that renders the old, single-tenant, data inaccessible. If you wish to switch from single-tenant to a multi-tenant model, it is recommended that you create a new multi-tenant stack and then migrate the data from the old stack. Switching from multi-tenant to a single-tenant model is also a breaking change.

Tenant identifiers

Tenants are identified by a tenant ID in the `auth` token. A tenant ID is a string that can contain alphanumeric characters, dashes, and underscores and have a maximum length of 64 characters.

The deployment creates a custom claim `custom:tenantId` to the Cognito user pool. When creating users, a tenant ID needs to be assigned to the user.

AWS CloudFormation template

This solution uses AWS CloudFormation to automate the deployment of FHIR Works on AWS in the AWS Cloud. It includes the following CloudFormation template, which you can download before deployment:

fhir-works-on-aws.template: Use this template to launch the solution and all associated components. The default configuration deploys Amazon API Gateway, Amazon CloudWatch Logs, Amazon Cognito, Amazon DynamoDB, Amazon OpenSearch Service, AWS Lambda, Amazon Simple Storage Service (Amazon S3), and AWS Identity and Access Management (IAM). You can customize the template to meet your specific needs by visiting the GitHub repository for the solution. Using the GitHub repository, you can also configure functionalities, such as multi-tenancy, implementation guides, log level, and Amazon OpenSearch Service hard delete.

Automated deployment

Before you launch the solution, review the architecture, configuration, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 30 minutes

Prerequisites

To generate a Cognito ID token with FHIR Works requires the AWS Command Line Interface (AWS CLI) to be available in your environment. This ID token is required for all reads or writes on FHIR resources. For installation instructions, refer to What Is the AWS Command Line Interface in the *AWS CLI User Guide*.

Deployment overview

Use the following steps to deploy this solution on AWS. For detailed instructions, follow the links for each step.

Step 1. Launch the stack (p. 10)

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters:
  - **Stage**
  - **CognitoOAuthDefaultRedirectURL**

Step 2. Post-configuration tasks (p. 12)

- Obtain the Amazon API Gateway API key.
- Create an Amazon Cognito user and obtain an authentication domain.
- Generate a Cognito ID token where a password change is required.
- Generate another Cognito ID token that does not require a password change.

Step 1. Launch the stack

This automated AWS CloudFormation template deploys FHIR Works on AWS in the AWS Cloud.

> **Note**
> You are responsible for the cost of the AWS services used while running this solution. For more details, visit to the Cost (p. 2) section in this guide, and refer to the pricing webpage for each AWS service used in this solution.

1. Sign in to the AWS Management Console and select the button to launch the `fhir-works-on-aws.template` AWS CloudFormation template.

Alternatively, you can download the template as a starting point for your own implementation.

2. The template launches 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.

   Note
   This solution uses the AWS Cognito service, which is not currently available in all AWS Regions. You must launch this solution in an AWS Region where AWS Cognito is available. For the most current availability by Region, refer to the AWS Regional Services List.

3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Stage	`dev`	A short name for identifying the stage. `dev` is the only option. **Note:** `dev` results in a lower-capacity, hence lower cost Amazon OpenSearch Service cluster than if you were to choose a non-dev stage. For details, refer to the GitHub code explanation.
CognitoOAuthDefaultRedirectURL	`http://localhost`	The authorization endpoint. For details, refer to Configuring a User Pool App Client. If using Postman, consider using https://oauth.pstmn.io/v1/browser-callback. If not applicable, use the default.
EnableMultiTenancy	`false`	To turn on multi-tenancy, refer to the Multi-tenancy (p. 8) section.
ExportGlueNumberWorkers	`5`	Number of Glue workers to be used during bulk data export.
ExportGlueWorkerType	`G.2X`	Glue worker type to be used for bulk data export. Valid values are G.2X and G.1X.
logLevel	`error`	Log level for CloudWatch logs. Valid values are: `debug`, `info`, `warn`, and `error`.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.

8. On the **Review** page, review and confirm the settings. 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 receive a CREATE_COMPLETE status in approximately 30 minutes.
10. From the CloudFormation Stacks page, choose the **Outputs** tab and record the value for `ServiceEndpoint`. This is the URL for FHIR Works on AWS.

Step 2. Post-configuration tasks

Obtain the API Gateway API key

FHIR Works on AWS requires an API key to authorize access.

1. Sign in to the Amazon API Gateway console.
2. Under **APIs**, select the <*stage*>-fhir-service API created by this solution.
3. In the left navigation pane, choose **API Keys**.
4. Select the API key for the API.
5. In the main panel, choose **Show** and record the API key for making FHIR API calls.

Create an Amazon Cognito user and obtain authentication domain

FHIR Works on AWS implements role-based access control (RBAC) to health data using Amazon Cognito user pools and ID tokens. Create a test user in Amazon Cognito to provision an ID token.

When multi-tenancy is enabled:

1. Create a new user by running the following command:

```
aws cognito-idp admin-create-user --user-pool-id <user-pool-id> --username <user-name>
 --user-attributes Name="custom:tenantId",Value="<tenant-id>" Name="email",Value="<user-
email>" --temporary-password <temp-password> --message-action SUPPRESS
```

2. Sign in to the Amazon Cognito console.
3. Choose the **Groups** tab, then **practitioner** from the list.
4. Choose **Add users**. Choose the **+** to add the new user to the **practitioner** group. This ensures that ID tokens generated for this user can access FHIR resources.
5. Choose **App client settings** in the menu and record the App client ID necessary for future Cognito API calls.
6. Choose **App Integration** in the menu and record the Domain if you plan to use Postman to make calls to the FHIR API.

When multi-tenancy is disabled:

1. Sign in to the Amazon Cognito console.
2. Choose **Manage User Pools**.
3. Choose the user pool created by this solution. It has the same name as the CloudFormation stack that was used to deploy the solution.

4. Choose **Users and groups**, then **Create user**.
5. Create a user with a temporary password.
6. Choose the **Groups** tab, then **practitioner** from the list.
7. Choose **Add users**. Choose the **+** to add the new user to the **practitioner** group. This ensures that ID tokens generated for this user can access FHIR resources.
8. Choose **App client settings** in the menu and record the App client ID necessary for future Cognito API calls.
9. Choose **App Integration** in the menu and record the Domain if you plan to use Postman to make calls to the FHIR API.

Generate a Cognito ID token (password change required)

When a user authenticates to Amazon Cognito, a temporary ID token is provided. The ID token is required for API calls to FHIR Works on AWS.

1. Launch a command line session in an environment with the AWS CLI configured.

   Note
   Ensure that the AWS CLI environment is using the same account and Region where FHIR Works on AWS is deployed.

2. Run the following *initiate-auth* command for the user, providing the username, temporary password, and App client ID:

```
aws cognito-idp initiate-auth --auth-flow USER_PASSWORD_AUTH --client-id <App-client-ID> --auth-parameters USERNAME=<Username>,PASSWORD=<Temporary password> --region <Your-Region>
```

   Replace *<App-client-ID>* with the ID from the Amazon Cognito user pool, under **App client settings**.

3. The new user requires a password change. Amazon Cognito responds with a `NEW_PASSWORD_REQUIRED` challenge. Record the session ID.
4. Run the following *respond-to-auth-challenge* command to select a permanent password:

```
aws cognito-idp respond-to-auth-challenge --client-id <App-client-ID> --challenge-name
 NEW_PASSWORD_REQUIRED --challenge-responses USERNAME=<Username>,NEW_PASSWORD=<Permanent-Password> --session <Session-ID> --region <Your-Region>
```

   Replace *<Session-ID>* with the session ID recorded from the CLI response in the previous step.

5. Amazon Cognito responds with an ID token that can be used to make FHIR API calls to FHIR Works on AWS. For details, refer to Accessing the FHIR API (p. 15).

Note
ID tokens expire after 60 minutes by default.

Generate a Cognito ID token (no password change required)

When the ID token expires, it can no longer be used in requests to FHIR Works on AWS.

To generate a new token, run the following *initiate-auth* command with the permanent password:

```
aws cognito-idp initiate-auth --auth-flow USER_PASSWORD_AUTH --client-id <App-client-ID> --
auth-parameters USERNAME=<Username>,PASSWORD=<Permanent-password>
```

The response includes a new ID token.

FHIR Works on AWS is now ready to receive API calls.

Accessing the FHIR API

FHIR API support

FHIR Works on AWS provides these FHIR capabilities:

- CRUD operations for all R4 base FHIR resources
- Basic search per resource type
- Versioned reads (vread)
- Ability to get and post binary resources
- Ability to post a transaction bundle of 25 entries or less

For details about the FHIR API, refer to the HL7 FHIR page.

There are several ways of accessing the FHIR API. REST syntax and Postman are two common ways.

Use REST syntax to make API Requests

Access the FHIR API using REST syntax as defined by FHIR by running the following command:

```
curl -H "Accept: application/json" -H "Authorization:<Cognito-access-token>" -H "x-api-key:<API-key>" <API URL>
```

Replace *<Cognito-access-token>* with an unexpired ID token from Amazon Cognito. For details, refer to Generate a Cognito ID token (p. 13).

Replace *<API-key>* with the API Key for the FHIR API. For details, refer to Obtain the API Gateway API Key (p. 12).

Replace *<API URL>* with the `ServiceEndpoint` output parameter from CloudFormation.

Additional resources

AWS services	
• AWS CloudFormation • AWS Lambda • Amazon DynamoDB • Amazon OpenSearch Service • Amazon Simple Storage Service (Amazon S3)	• Amazon API Gateway • Amazon Cognito • AWS Key Management Service (AWS KMS) • Amazon CloudWatch • AWS Identity and Access Management (IAM)

Uninstall the solution

You can uninstall the FHIR Works on AWS solution from the AWS Management Console or by using the AWS Command Line Interface. You must manually delete the Amazon Simple Storage Service (Amazon S3) buckets and Amazon DynamoDB table created by this solution. AWS Solutions Implementations do not automatically delete these resources in case you have stored data to retain.

Using the AWS Management Console

To uninstall FHIR Works on AWS solution, do the following.

1. Sign in to the AWS CloudFormation console.
2. On the **Stacks** page, select the solution stack.
3. Choose **Delete**.

Using AWS Command Line Interface

Verify that the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, refer to What Is the AWS Command Line Interface in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command.

```
$ aws cloudformation delete-stack --stack-name <your-stack-name>
```

Deleting the Amazon S3 buckets

The following five types of buckets are created by FHIR Works on AWS:

Bucket	Description
`auditlogsbucket`	Stores audit logs
`bulkexportresultbukcet`	Stores export data
`fhirbinarybucket`	Contains FHIR binary resources
`fhirlogsbucket`	Contains Amazon API Gateway logs.
`gluescriptsbucket`	Stores glue script for bulk export

1. Sign in to the Amazon S3 console.
2. Select the bucket to be deleted.
3. Choose **Empty**.
4. Choose **Delete**.

To delete the S3 bucket using AWS CLI, run the following command:

```
$ aws s3 rb s3://<bucket-name> --force
```

Alternatively, you can configure the AWS CloudFormation template to delete the Amazon S3 bucket automatically. Before deleting the stack, change the deletion behavior in the AWS CloudFormation DeletionPolicy attribute.

Deleting the Amazon DynamoDB table

FHIR Works on AWS creates DynamoDB tables which are not automatically deleted. To delete these tables, use the steps below.

1. Sign in to Amazon DynamoDB console.
2. Select the `resource-db-dev` table, which contains FHIR resource data.
3. Choose **Delete**.
4. Select the `export-request-dev` table, which contains all data export requests.
5. Choose **Delete**.

Source code

Visit the GitHub repository for the solution to download templates and scripts, and to share your customizations with others.

Contributors

The following individuals contributed to this document:

- Karl Camp
- Maggie Krummel
- Neel Sethia
- Robert Smayda
- Simon Woldemichael
- Stephen Younge
- Yanyu Zheng

Revisions

Date	Change
December 2020	Initial release
February 2021	Release v2.1.2: Bug fixes and documentation updates. For more information, refer to the CHANGELOG.md file in the GitHub repository.
May 2021	Release v2.1.3: Code build improvements. For more information, refer to the CHANGELOG.md file in the GitHub repository.
November 2021	Release v4.0.0: Code build improvements and documentation updates. For more information, refer to the CHANGELOG.md file in the GitHub repository.

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 AWS current 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. AWS responsibilities and liabilities 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.

FHIR Works on AWS is licensed under the terms of the of the Apache License Version 2.0 available at https://www.apache.org/licenses/LICENSE-2.0.

FHIR is the registered trademark of HL7 and is used with the permission of HL7. Use of the FHIR trademark does not constitute endorsement of this product by HL7.