FHIR Works on AWS

Implementation Guide

December 2020

Last update: March 2023

(see Revisions)

Contents

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 improve the exchange of health data between software systems such as practice management, electronic health records, billing, 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 a reference implementation, designed to be extensible.

This implementation guide provides an overview of the FHIR Works on AWS solution, its reference architecture and components, considerations for planning the deployment, and configuration steps for deploying the FHIR Works on AWS solution to the Amazon Web Services (AWS) Cloud.

Use this navigation table to quickly find answers to these questions:

If you want to . . .	Read . . .
Know the cost for running this solution.	Cost
Understand the security considerations for this solution.	Security
Know how to plan for quotas for this solution.	Quotas
Know which AWS Regions are supported for this solution.	Supported AWS Regions

This guide is intended for solution architects, business decision makers, DevOps engineers, data scientists, and cloud professionals who want to implement FHIR Works on AWS in their environment.

Features and benefits

FHIR Works on AWS provides the following features:

Reads and writes Amazon OpenSearch Service documents from aliases instead of indexes.

This simplifies performing re-indexing operations and reduces downtime.

Uses Cognito ID token instead of access token to authorize requests.

Cognito uses role based access control to limit access and operation on the defined resource types.

Uses pooled infrastructure model for multi-tenacy.

Multi-tenacy allows a single `fhir-works-on-aws` stack to serve as multiple FHIR servers for different tenants. For more information, see Using Multi-tenancy.

Integration with AWS Service Catalog AppRegistry and AWS Systems Manager Application Manager

This solution includes a Service Catalog AppRegistry resource to register the solution's CloudFormation template and its underlying resources as an application in both AWS Service Catalog AppRegistry and AWS Systems Manager Application Manager. With this integration, you can centrally manage the solution's resources.

Use cases

Cognito Based Authentication

For customers interested in a quick setup which does not require ONC Compliance, a Cognito-based authentication setup would be sufficient. Cognito uses role based access control (RBAC) to limit access and operations on the defined resource types. Cognito User Groups determine a user's roles and passes group information through the OAuth access token when a user makes a FHIR API request.

SMART Authentication

Substitutable Medical Applications, Reusable Technologies (SMART on FHIR) specification provides the guidelines for this attribute-based access control (ABAC) authorization flow. FHIR Works on AWS functions as a resource server for SMART on FHIR and is not an authorization server or a SMART front-end app. Customers using this flow should have an OAuth2 SMART compliant authorization server. To learn more, see OAuth2 Flow and Authorization Prerequisites.

Architecture overview

This section provides a reference implementation architecture diagram for the components deployed with this solution.

Architecture diagram

Deploying this solution with the default parameters deploys the following components in your AWS account.

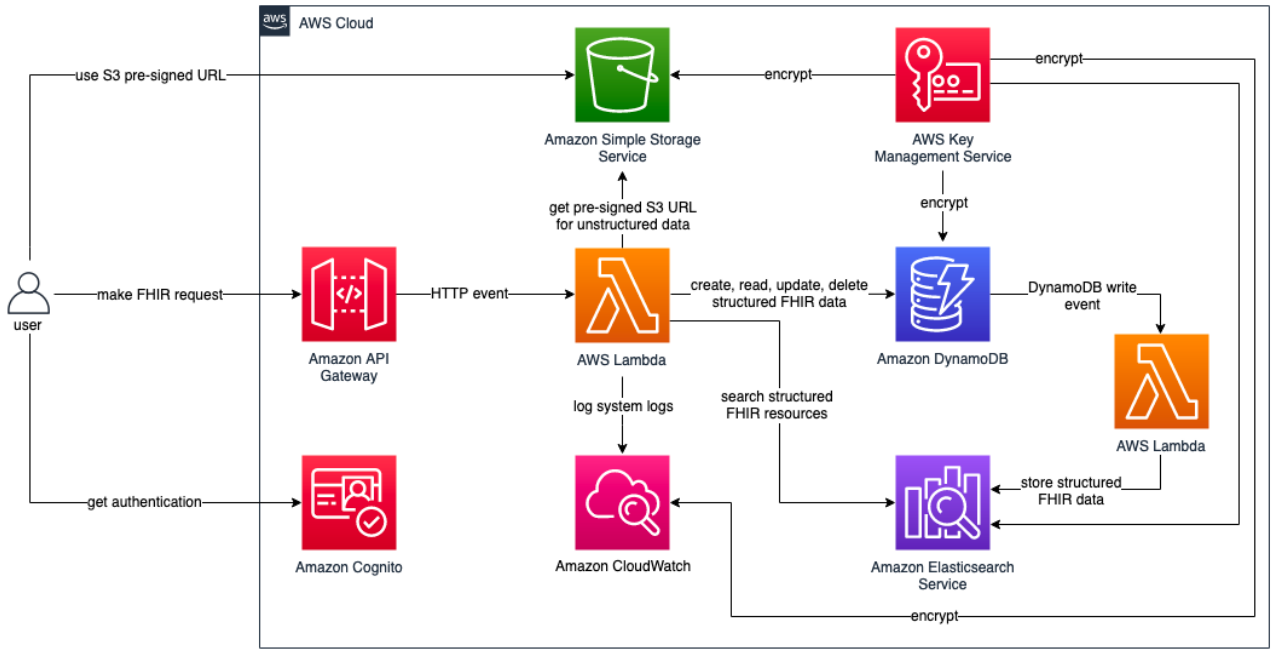


Figure 1: FHIR Works on AWS architecture

> **Note:** AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

The high-level process flow for the solution components deployed with the AWS CloudFormation template is as follows:

1. Amazon Cognito user pool, domain, and client authenticates the requester's identity and determines which group the user is in.
2. Amazon API Gateway routes 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.

3. AWS Lambda functions handle two processes. One processes 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 reads updates from the FHIR resource DynamoDB table and streams those changes to OpenSearch Service.
4. DynamoDB table stores all structured FHIR resources and, after a write operation, streams the update to the OpenSearch Service domain.
5. OpenSearch Service domain supports FHIR search requests.
6. Amazon S3 bucket holds FHIR binary resources, such as unstructured data, X-rays, and raw notes.
7. AWS Key Management Service (AWS KMS) keys encrypt DynamoDB, Amazon S3, Amazon CloudWatch Logs, and the OpenSearch Service domain.
8. Amazon CloudWatch logs requests to and responses from the APIs which can be optionally archived to Amazon S3 in order to optimize cost.

Architecture details

This section describes the components and AWS services that make up this solution and the architecture details on how these components work together.

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 Amazon API Gateway authenticates a request, 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.

FWoA supports SMART on FHIR for attribute based access control. Refer to the README file and SMART_ON_FHIR_FAQ for more information.

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.

FHIR subscriptions mechanism

The FHIR Subscription resource defines a push-based subscription from a server to another system. Once a subscription is registered with the server, the server checks every resource that is created or updated. If the resource matches the given criteria, the server sends a message on the defined channel so that another system can take appropriate action.

Currently, only rest-hook notification channels are supported. We do not support SMS, email, or Websocket channels.

FHIR Implementation Guides mechanism

The base FHIR specification describes a set of base resources, frameworks, and APIs that are used in many different contexts in healthcare. However, jurisdictions and healthcare ecosystems widely vary around practices, requirements, regulations, education and feasible and/or beneficial actions. For this reason, the FHIR specification creates a common platform or foundation on which a variety of different solutions may be implemented. These different solutions are expressed as Implementation Guides.

FWoA supports Implementation Guides through using S3 to host customer-specified implementation guides and compiling the Definitions and ValueSets provided into additional extensions to FHIR capabilities such as Search, Resource Types, and Profiles.

AWS services in this solution

AWS service	Description
Amazon API Gateway	**Core.** Serves as a single point of access for FWoA API
Amazon Cognito	**Optional.** Used for Authentication in FWoA (not used in SMART Implementation)
AWS CloudFormation	**Core.** Used to deploy FHIR server in customer's AWS account
Amazon Cloudwatch	**Supporting.** Used for logging system logs
AWS Data Pipeline	**Core.** Orchestration service that helps export data from DynamoDB to OpenSearch
Amazon DynamoDB	**Core.** Persist FHIR records
AWS Glue	**Core.** Enables export of FHIR records from DynamoDB to S3
Amazon IAM	**Core.** IAM roles needed to deploy solution and manage services
Amazon KMS	**Core.** Manages encryption of CloudWatch logs, S3, backup, SNS, open search, DynamoDB
AWS Lambda	**Core.** Contains all FWoA APIs and functions used to communicate OpenSearch, API Gateway requests, S3, Cloudwatch, and DynamoDB
Amazon Open Search	**Core.** Paired with Dynamo DB for system-wide and type searching
Amazon S3	**Core.** Used for storing FHIR binary resource, bulk export, and deployment
Amazon SNS	**Optional.** Used for push-based subscription from one server to another
Amazon SQS	**Core.** Used as a dead letter queue for a stream between DynamoDB and OpenSearch
AWS Step Functions	**Core.** Used for Bulk Export workflow

Plan your deployment

Cost

> You are responsible for the cost of the AWS services used while running this solution. As of March 2023, the cost for running this solution with the default settings in the US East (N. Virginia) is approximately **$140.00 a month**.
>
> See the pricing webpage for each AWS service used in this solution.

We recommend creating a budget through AWS Cost Explorer to help manage costs. Prices are subject to change. For full details, see the pricing webpage for each AWS service used in this solution.

Sample cost table

The following table provides a sample cost breakdown for deploying this solution with the default parameters in the US East (N. Virginia) Region for one month.

AWS service	Dimensions	Cost [USD]
Amazon API Gateway	500 requests per day	$ 0.03
Amazon DynamoDB	10GB of storage and 250 read and 250 write requests per day	$ 4.02
Amazon OpenSearch Service	10GB of EBS storage, 1 t3.medium.elasticsearch instance	$ 110.21
Amazon Simple Storage Service	1TB of storage, 5 reads and 5 writes per day	$ 23.00
AWS Key Management Service	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

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared responsibility 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 on the AWS Cloud. This solution creates IAM roles that grant the solution's AWS Lambda functions access to create Regional resources. This solution creates an IAM role, `serverlessApiGatewayCloudWatchRole`, to grant Amazon API Gateway access which sends logs to Amazon CloudWatch.

Amazon CloudFront

This solution deploys a web console hosted in an Amazon Simple Storage Service (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 CloudFront user that provides public access to the solution's website bucket contents. For more information, see Restricting Access to Amazon S3 Content by Using an Origin Access Identity in the *Amazon CloudFront Developer Guide*.

Security groups

The security groups created in this solution are designed to control and isolate network traffic between the Lambda functions, CSR instances, and remote VPN endpoints. We recommend that you review the security groups and further restrict access as needed once the deployment is up and running.

Amazon Cognito

This solution deploys Cognito as an authorization mechanism using custom claims and user groups to divide permission levels for access to data. Users can be created in the Cognito User Pool and placed into groups to receive authentication tokens to make requests. For more information, see the Cognito User Guide.

Security Logging

This solution allows the option to enable security logging during deployment. This allows CloudWatch logs to report encrypted data on each request, such as who made the request, how the request was made, what the request was made for, and when and where the request originated. For more information on best practices, see Best Practices in the FHIR Works GitHub repository.

Supported AWS Regions

This solution uses the Amazon Cognito service, 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 of AWS services by Region, see the AWS Regional Services List.

FHIR Works on AWS is supported in the following AWS Regions:

Region name	
US East (Ohio)	Asia Pacific (Tokyo)
US East (N. Virginia)	Canada (Central)
US West (Northern California)	Europe (Frankfurt)
US West (Oregon)	Europe (Ireland)
Asia Pacific (Mumbai)	Europe (London)
Asia Pacific (Seoul)	Europe (Paris)
Asia Pacific (Singapore)	Europe (Stockholm)
Asia Pacific (Sydney)	South America (São Paulo)

Deploy the solution

Before you launch the solution, review the cost, architecture, network security, and other considerations discussed earlier 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 30-45 minutes

> **Important:** 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. AWS owns the data gathered though this survey. Data collection is subject to the AWS Privacy Policy.
>
> To opt out of this feature, download the codes, remove the CUSTOM_USER_AGENT string, and then use the updated codes to deploy the solution. For more information, see the Anonymous data collection section of this guide.

Follow the instructions located in the GitHub repository to deploy the solution.

Monitoring the solution with AWS Service Catalog AppRegistry

The FHIR Works on AWS solution includes a Service Catalog AppRegistry resource to register the CloudFormation template and underlying resources as an application in both AWS Service Catalog AppRegistry and AWS Systems Manager Application Manager.

AWS Systems Manager Application Manager gives you an application-level view into this solution and its resources so that you can:

- Monitor its resources, costs for the deployed resources across stacks and AWS accounts, and logs associated with this solution from a central location.
- View operations data for the resources of this solution in the context of an application. For example, deployment status, CloudWatch alarms, resource configurations, and operational issues.

The following figure depicts an example of the application view for the FHIR Works on AWS stack in Application Manager.

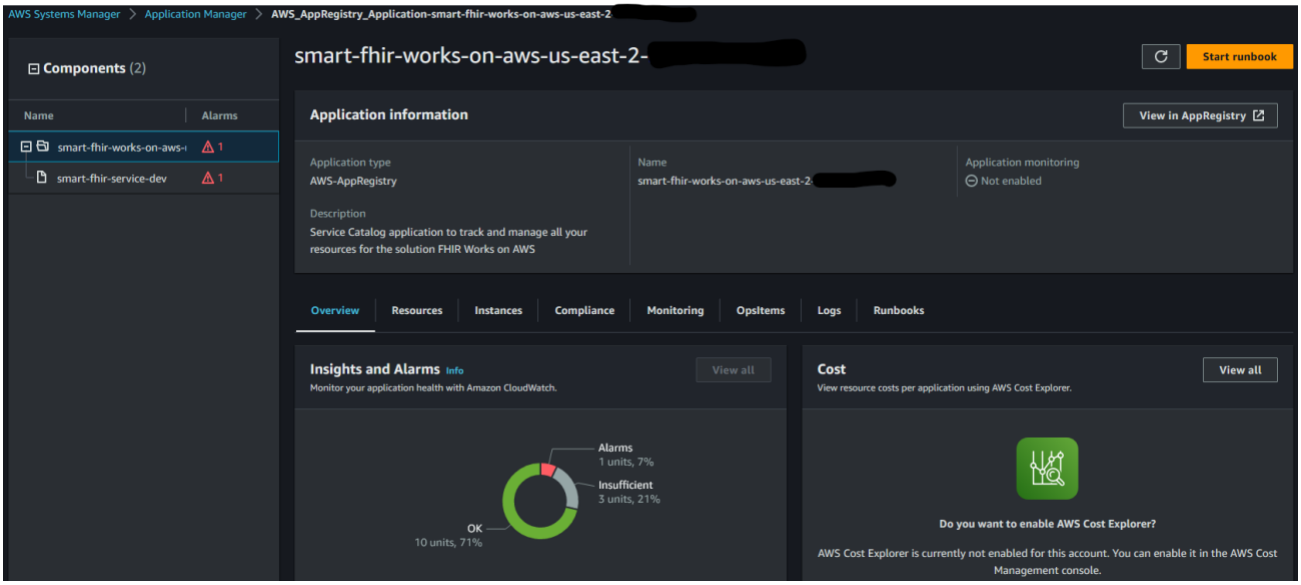


Figure 2: FHIR Works on AWS stack in Application Manager

> **Note:** You must activate CloudWatch Application Insights, AWS Cost Explorer, and cost allocation tags associated with this solution. They are not activated by default.

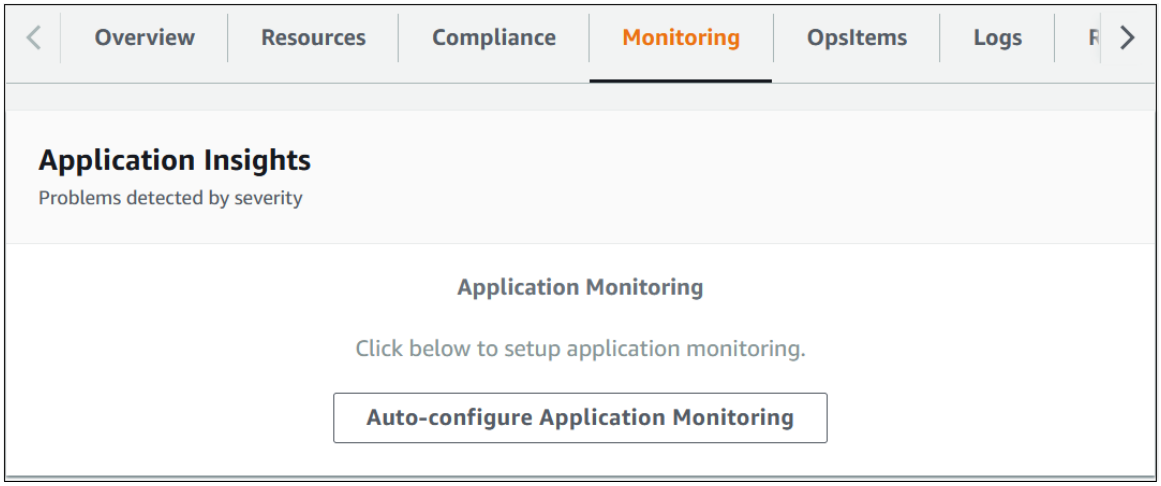
Activate CloudWatch Application Insights

1. Sign in to the Systems Manager console.
2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, choose **AppRegistry applications**.

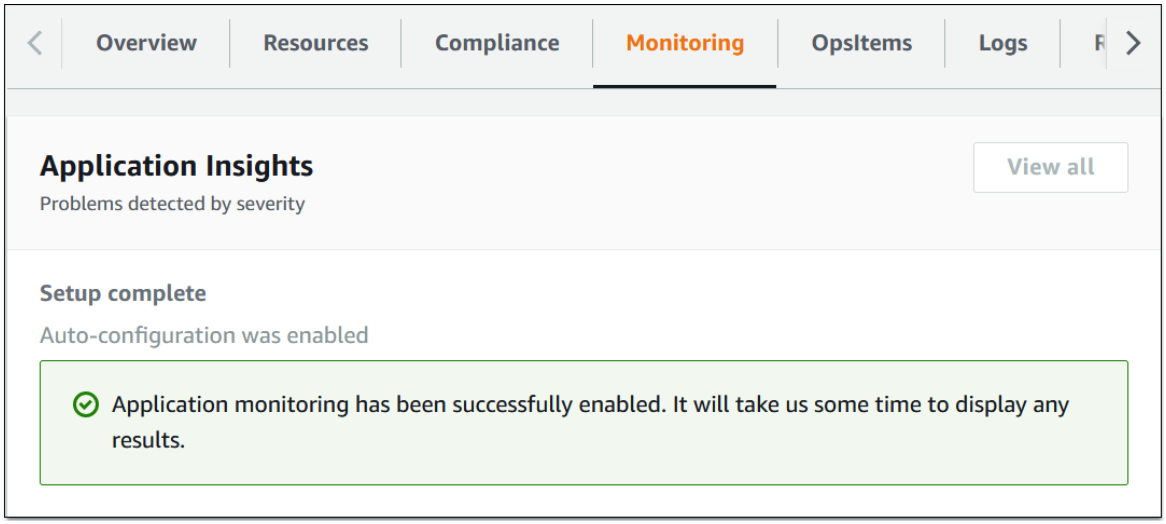
4. In **AppRegistry applications**, search for the application name for this solution and select it.

The next time you open Application Manager, you can find the new application for your solution in the **AppRegistry application** category.

5. In the **Components** tree, choose the application stack you want to activate.
6. In the **Monitoring** tab, in **Application Insights**, select **Auto-configure Application Monitoring**.



Monitoring for your applications is now activated and the following status box appears:



Activate AWS Cost Explorer

You can see the overview of the costs associated with the application and application components within the Application Manager console through integration with AWS Cost Explorer which must be first activated. Cost Explorer helps you manage costs by providing a view of your AWS resource costs and usage over time. To activate Cost Explorer for the solution:

1. Sign in to the AWS Cost Management console.

2. In the navigation pane, select **Cost Explorer**.
3. On the **Welcome to Cost Explorer** page, choose **Launch Cost Explorer**.

The activation process can take up to 24 hours to complete. Once activated, you can open the Cost Explorer user interface to further analyze cost data for the solution.

Activate cost allocation tags associated with the solution

After you activate Cost Explorer, you must activate the cost allocation tags associated with this solution to see the costs for this solution. The cost allocation tags can only be activated from the management account for the organization. To activate cost allocation tags:

1. Sign in to the AWS Billing and Cost Management console.
2. In the navigation pane, select **Cost Allocation Tags**.
3. On the **Cost allocation tags** page, filter for the `AppManagerCFNStackKey` tag, then select the tag from the results shown.
4. Choose **Activate**.

The activation process can take up to 24 hours to complete and the tag data to appear.

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, Amazon DynamoDB table, CloudWatch logs, and KMS key aliases 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

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

Using AWS Command Line Interface

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, see 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
<installation-stack-name>
```

Deleting the Amazon S3 buckets

This solution is configured to retain the solution-created Amazon S3 bucket (for deploying in an opt-in Region) if you decide to delete the AWS CloudFormation stack to prevent accidental data loss. After uninstalling the solution, you can manually delete this S3 bucket if you do not need to retain the data. Follow these steps to delete the Amazon S3 bucket.

1. Sign in to the Amazon S3 console.
2. Choose **Buckets** from the left navigation pane.
3. Locate the *<stack-name>* S3 buckets.
4. Empty the S3 buckets.
5. Select the S3 bucket and choose **Delete**.

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

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

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**.

Developer guide

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others.

The FHIR Works on AWS templates are generated using the AWS Cloud Development Kit (AWS CDK). See the README.md file for additional information.

API reference

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.

Replace `<API-key>` with the API Key for the FHIR API.

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

Postman Collection

To access APIs, you can use Postman as well. To use the Postman collection, see Using Postman to make API for FHIR and Using Postman to make API requests for SMART on FWoA.

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-tenacy

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-tenacy

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.

Troubleshooting

For troubleshooting FWoA, see Troubleshooting FHIR Works on AWS.

For troubleshooting SMART on FWoA, see Troubleshooting SMART on FWoA.

Reference

This section includes information about an optional feature for collecting unique metrics for this solution, pointers to related resources, and a list of builders who contributed to this solution.

Anonymous data collection

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 invoked, the following information is collected and sent to AWS:

- **Solution ID** - The AWS solution identifier
- **Unique ID (UUID)** - Randomly generated, unique identifier for each <solution-name> deployment

- **Timestamp** - Data-collection timestamp
- **Example: Instance Data** - Count of the state and type of instances that are managed by the EC2 Scheduler in each AWS Region

  Example data:

  Running: {t2.micro: 2}, {m3.large:2}

  Stopped: {t2.large: 1}, {m3.xlarge:3}

Related resources

- 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)

Contributors

- Robert Smayda
- Stephen Younge
- Yanyu Zheng
- Brandi Hopkins
- Balaji Raman
- Jay Luo
- Karina Cadette
- Nikhil Sankepalli
- Sergey Vun
- Sukeerth Vegaraju

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.

Revisions

For more information, see the CHANGELOG.md file in the GitHub repository.

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.
March 2023	Release v6.0.0: Code build improvements and documentation updates. For more information, refer to the CHANGELOG.md file in the GitHub repository.