FHIR Works on AWS

Migration Tool User Guide

July 2023

Last update: August 2023

Contents

Overview

The FHIR Works on AWS Migration tool uses a collection of scripts to export FHIR resources from the FHIR Works on AWS solution and import it into to Amazon HealthLake. Migration tool exports FHIR resources stored in the Amazon S3 bucket and Amazon DynamoDB tables of FHIR Works on AWS solution into an S3 bucket as NDJSON files. Migration tool imports NDJSON files from the S3 bucket into HealthLake datastore. FHIR resource history, including deleted resources, also migrate into Amazon HealthLake. The migration tool converts binary resources stored in native file formats in FHIR Works on AWS to FHIR Binary Resource type during the import process into Amazon HealthLake. Migration tool outputs logs containing informational events and any errors during the migration process. Additionally, FHIR Works on AWS export and Amazon HealthLake import transaction logs will be available.

For questions regarding the migration process, contact fwoa-migration-support@amazon.com. Alternatively, you may also contact your account manager, Solutions Architect, or AWS Support for Amazon HealthLake.

Features

The migration tool provides the following features:

- Allows export of available resource history using export script from FHIR Works on AWS
- Allows export of deleted resources using export script from FHIR Works on AWS
- Allows import of FHIR resources into Amazon HealthLake using HealthLake import APIs
- Converts Binary resources stored in native file formats within FHIR Works on AWS to the FHIR Binary Resource type and imports it into Amazon HealthLake
- Allows the retention of original FHIR resource logical id stored in FHIR Works on AWS
- HealthLake adds last updated date based on the time the import successfully happens
- Allows batch export from FHIR Works on AWS using the since parameter for incremental export
- Imports data to Amazon HealthLake in separate data stores for multi-tenant configurations of FHIR Works on AWS
- Uses hashes to verify during import content remains unchanged after import (excludes metadata)
- Provides logs for troubleshooting

Use cases

- **Single Tenant Configuration of FHIR Works on AWS** - Migration tool allows data import from single tenant configuration into a single data store in Amazon HealthLake.

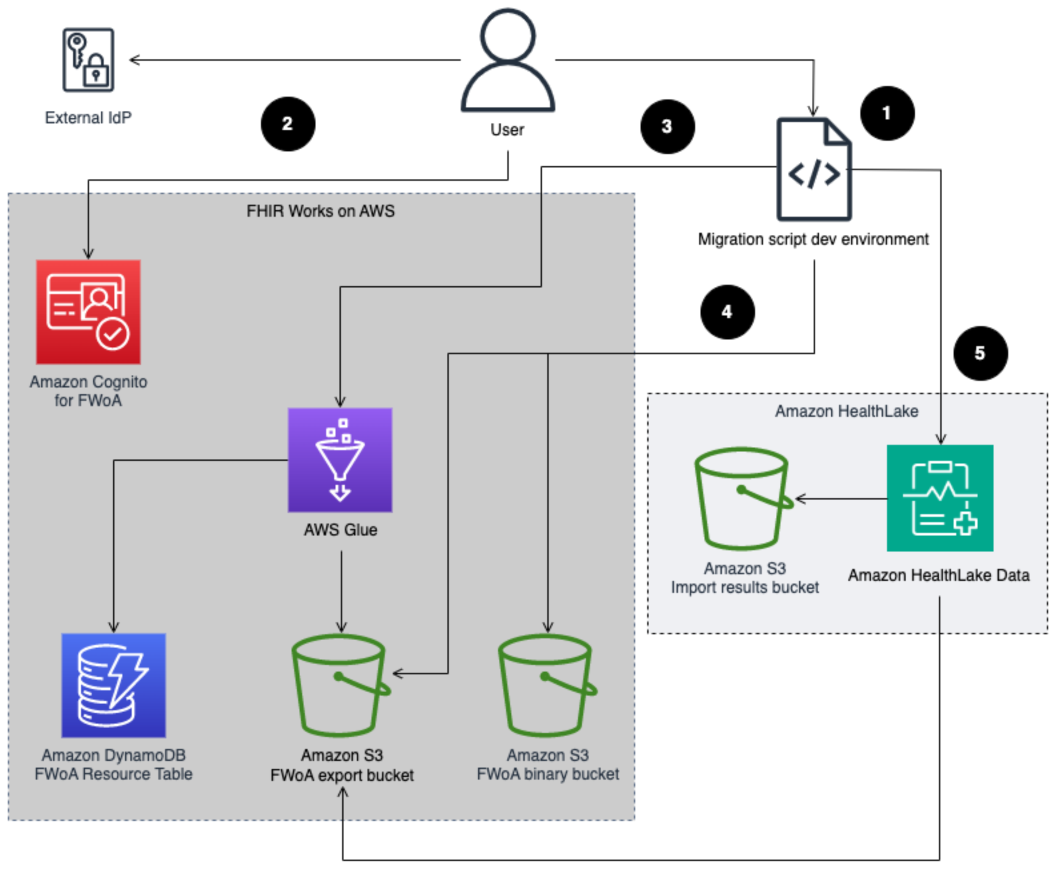
- **Multi-tenant Configuration of FHIR Works on AWS** - Migration tool allows to export data by tenant and import into Amazon HealthLake in separate data stores.
- **Migrate Data in batches** - Migration tool allows to incrementally export from FHIR Works on AWS and import into Amazon HealthLake to maintain high availability of your FHIR Works on AWS solution.
- **Migrate Data from a region where Amazon HealthLake is not available** - If you intend to migrate FHIR Works on AWS deployed from another region to a region where Amazon HealthLake is available, you need to replicate the FHIR Works on AWS export bucket to a supported Amazon HealthLake region before import.

Customer responsibilities

Customers are responsible for the following during and after migration:

- Running migration in a secure environment limiting access to the environment following the principle of least privilege and organizational best practices
- Setup of IAM role/permissions and authentication/credentials for migration scripts to interact with AWS resources
- Setup of Identity Provider configurations for SMART FHIR Works on AWS deployments to authenticate through username/password
- Verification that data migration to Amazon HealthLake is complete
- Update APIs in their applications to Amazon HealthLake
- Proper cleanup of migration environment, including logs and AWS resources
- Get re-certified with ONC Certification after migration to HealthLake if needed
- Decommissioning of FHIR Works on AWS solution and archiving any logs and other data associated with the FHIR Works on AWS solution in the customer's AWS Account.

Architecture diagram


External IdP
User
1
2
3
4
5
Migration script dev environment
FHIR Works on AWS
Amazon Cognito for FWoA
AWS Glue
Amazon DynamoDB FWoA Resource Table
Amazon S3 FWoA export bucket
Amazon S3 FWoA binary bucket
Amazon HealthLake
Amazon S3 Import results bucket
Amazon HealthLake Data

Step 1. Setup a Cloud9 environment in an AWS Account and download the migration scripts.

Step 2. Setup the authentication of your existing FHIR Works on AWS solutions using the authentication script provided.

Step 3. Run the export script to export data from FHIR Works on AWS Solution into the FHIR Works on AWS export bucket.

Step 4. Run the binary conversion script to export the binary resources into the FHIR Works on AWS export bucket.

Step 5. Run the import script to ingest FHIR resources into Amazon HealthLake.

Prerequisites

Before you migrate FHIR resources, complete these prerequisite tasks.

Step 1: Setup AWS accounts

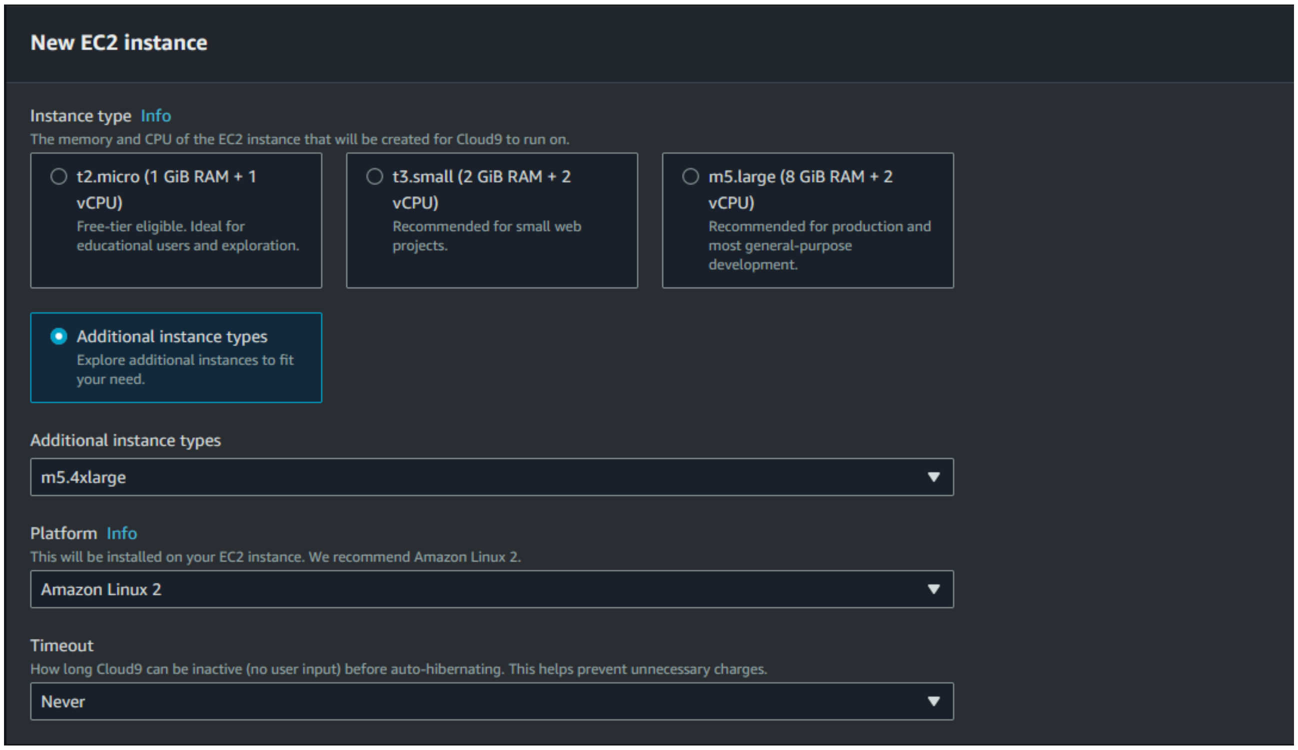
Create an AWS account or use an existing AWS account for setting up a development environment to run the migration tool.

Step 2: Setup Cloud9 environment

Note

The size of your data storage in FHIR Works impacts the necessary memory, compute, and storage size needed for your environment. Please contact your AWS account team to determine the optimal AWS Cloud9 configuration for your needs before beginning the migration.

1. Follow the instructions to Setup a Cloud9 environment in your AWS account.
   - We recommend, at minimum, an xlarge instance type running Amazon Linux 2 with a minimum of 64 GB RAM to run the migration scripts.
   - We recommend turning off the timeout setting to allow for uninterrupted execution of migration scripts.


New EC2 instance

Instance type Info
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.

○ t2.micro (1 GiB RAM + 1 vCPU) Free-tier eligible. Ideal for educational users and exploration.
○ t3.small (2 GiB RAM + 2 vCPU) Recommended for small web projects.
○ m5.large (8 GiB RAM + 2 vCPU) Recommended for production and most general-purpose development.
● Additional instance types Explore additional instances to fit your need.

Additional instance types
m5.4xlarge

Platform Info
This will be installed on your EC2 instance. We recommend Amazon Linux 2.
Amazon Linux 2

Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.
Never

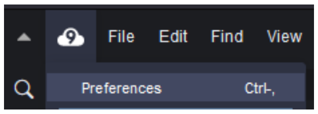
2. Follow the instructions to adjust your hard disk size of the Cloud9 environment to a desired setting. We recommend at least 1.5 times the size of your FHIR Works on AWS Binary S3 Bucket.

3. Run the following command to install ts-node:

```
npm i -g ts-node
```

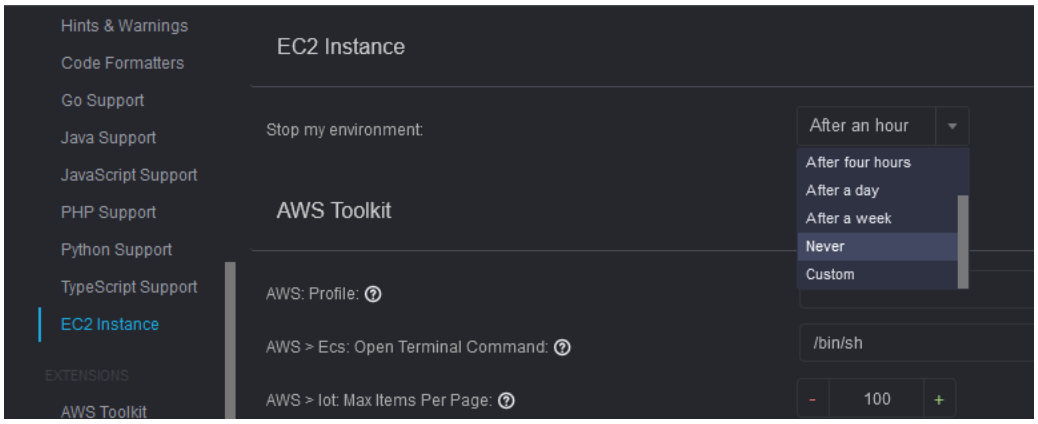
4. If you expect a longer migration, we recommend reducing the default sleep configuration of your Cloud9 instance. You can adjust the EC2 preferences in Cloud9 to ensure the instance does not go into sleep mode during the script execution.

   a. From **Cloud9**, choose **Preferences**.



   b. Choose **EC2 instance**.

   c. From **Stop my environment**, choose **Never**. Though we recommend choosing Never, you may choose the setting appropriate to your migration.



Step 3: Setup Amazon HealthLake data store

Follow the directions to create a HealthLake data store and skip the optional step 6 to preload Synthea data.

If you are using a multi-tenant deployment of FHIR Works on AWS, you will need to create a data store for each tenant.

Step 4: Setup migration tool code and dependencies

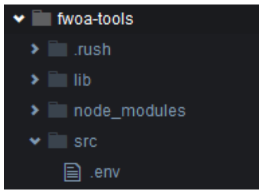
1. Download the Migration scripts into your Cloud 9 environment.
2. Install the node modules needed for the migration script:
   a. If you are running with the entire FHIR Works on AWS repository and intend to use rush:
      i. Run `rush install`.

b. If you are only using the `fwoa-tools` folder from the FHIR Works on AWS repository:

i. Remove line 49 from the `package.json` file.

ii. Run `npm install` in the root `fwoa-tools` directory.

3. Create a file named `.env` in the `<ROOT FOLDER>/fwoa-tools/src` directory. This file holds all environment variables and credentials needed for the migration process.



Step 5: Set up environment variables

If you would prefer to use the CLI to retrieve the environment variables that are from the CloudFormation Stack outputs, you can use the following commands to retrieve all stack resources and outputs:

```
aws cloudformation describe-stacks --stack-name <your
fhir works stack name> --query 'Stacks[0].Outputs' --
region <your region>
```

```
aws cloudformation describe-stack-resources --stack-name
<your fhir works stack name> --region <your region>
```

Variables for access keys

Add the following information to the .env file:

Make sure your AWS credentials don't expire before your migration is complete.

```
#### AWS Credentials
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
```

Variables for authentication setup

Add the following information to the `.env` file:

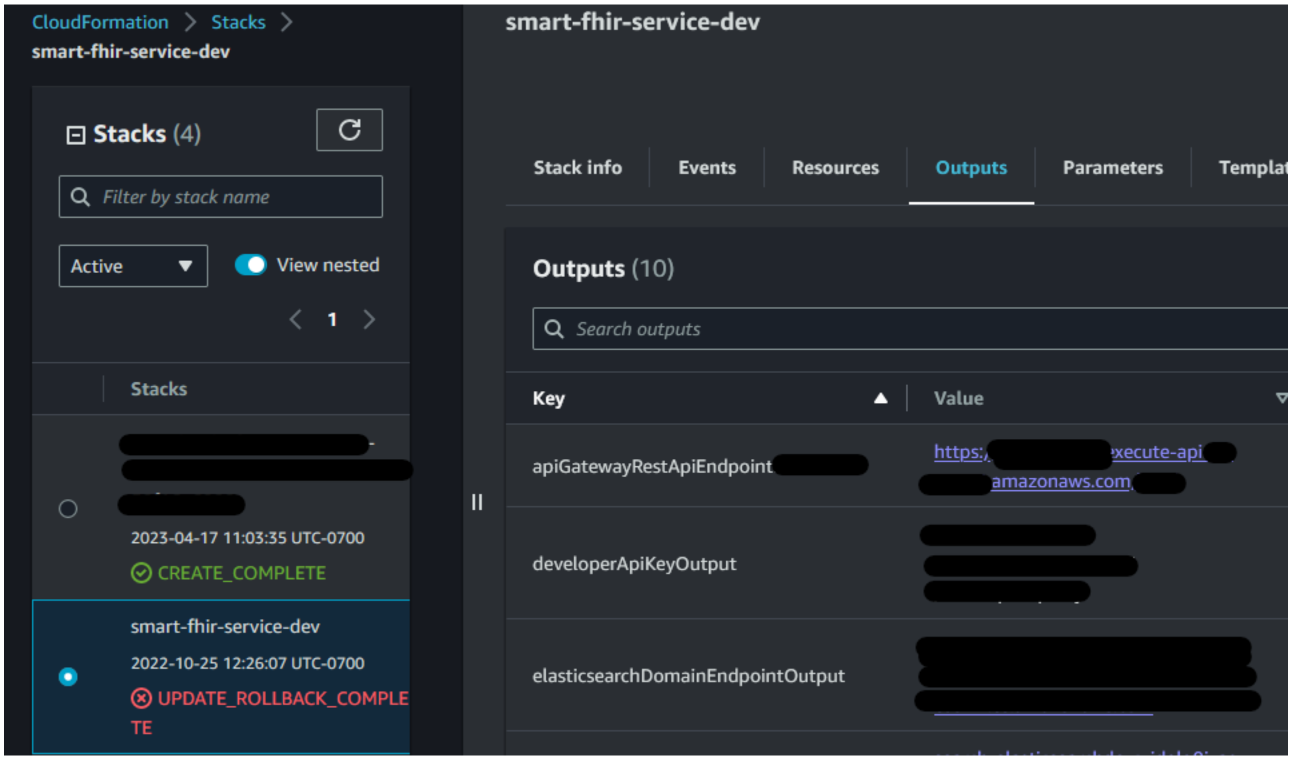
1. For a **SMART FHIR Works on AWS deployment**, you need the following environment variables:

a. Add the following template to the `.env` file:

```
#### SMART - FHIR Works on AWS Credentials
SMART_SERVICE_URL=
SMART_API_KEY=
SMART_AUTH_USERNAME=
SMART_AUTH_PASSWORD=
SMART_CLIENT_ID=
SMART_CLIENT_SECRET=
SMART_OAUTH2_API_ENDPOINT=
```

b. `SMART_SERVICE_URL` = the API Gateway URL of the FHIR Works on AWS deployment. This can be obtained from the CloudFormation stack outputs as `apiGatewayRestApiEndpoint`.

If multi-tenant, append the tenant suffix to the end of the url (e.g. `API_URL/tenant/tenantId`)

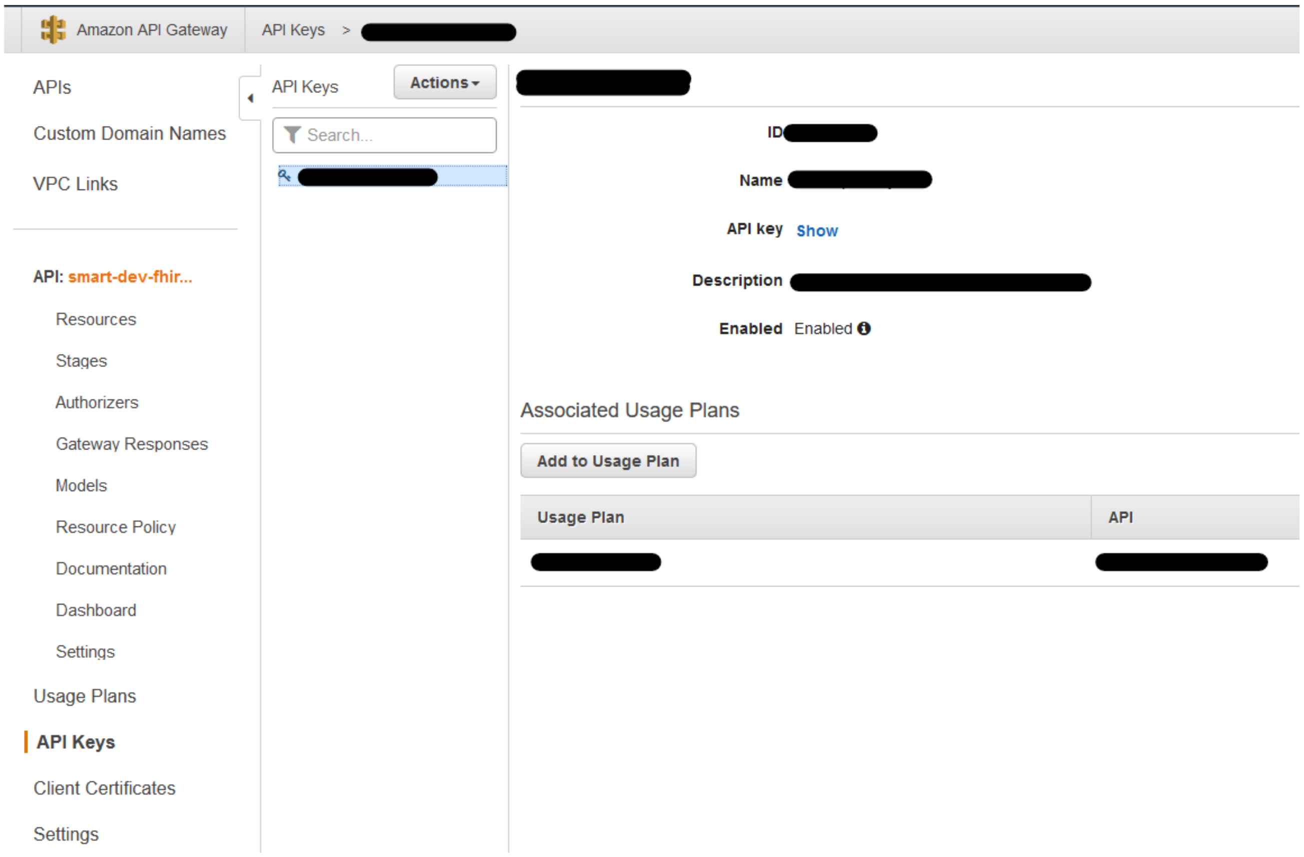


[redacted]
2023-04-17 11:03:35 UTC-0700
CREATE_COMPLETE

smart-fhir-service-dev
2022-10-25 12:26:07 UTC-0700
UPDATE_ROLLBACK_COMPLETE

Key	Value
apiGatewayRestApiEndpoint [redacted]	https:/[redacted]xecute-api[redacted] [redacted]amazonaws.com[redacted]
developerApiKeyOutput	[redacted]
elasticsearchDomainEndpointOutput	[redacted]

c. `SMART_API_KEY` = the API Gateway Key for accessing the API. This can be found in the AWS Console by navigating the API Gateway page, and then choosing the FHIR Works on AWS API. Then, navigate to the **API Keys** tab to find the API Key.



[redacted]

ID [redacted]

Name [redacted]

API key Show

Description [redacted]

Enabled Enabled

Usage Plan	API
[redacted]	[redacted]

d. `SMART_AUTH_USERNAME` = the IdP login username for a client configured to allow resource owner password

e. `SMART_AUTH_PASSWORD` = the IdP login password for a client configured to allow resource owner password

f. `SMART_OAUTH2_API_ENDPOINT` = the authorize endpoint of the IdP

g. `SMART_CLIENT_ID` = the IdP client id

h. `SMART_CLIENT_SECRET` = the IdP client password/secret

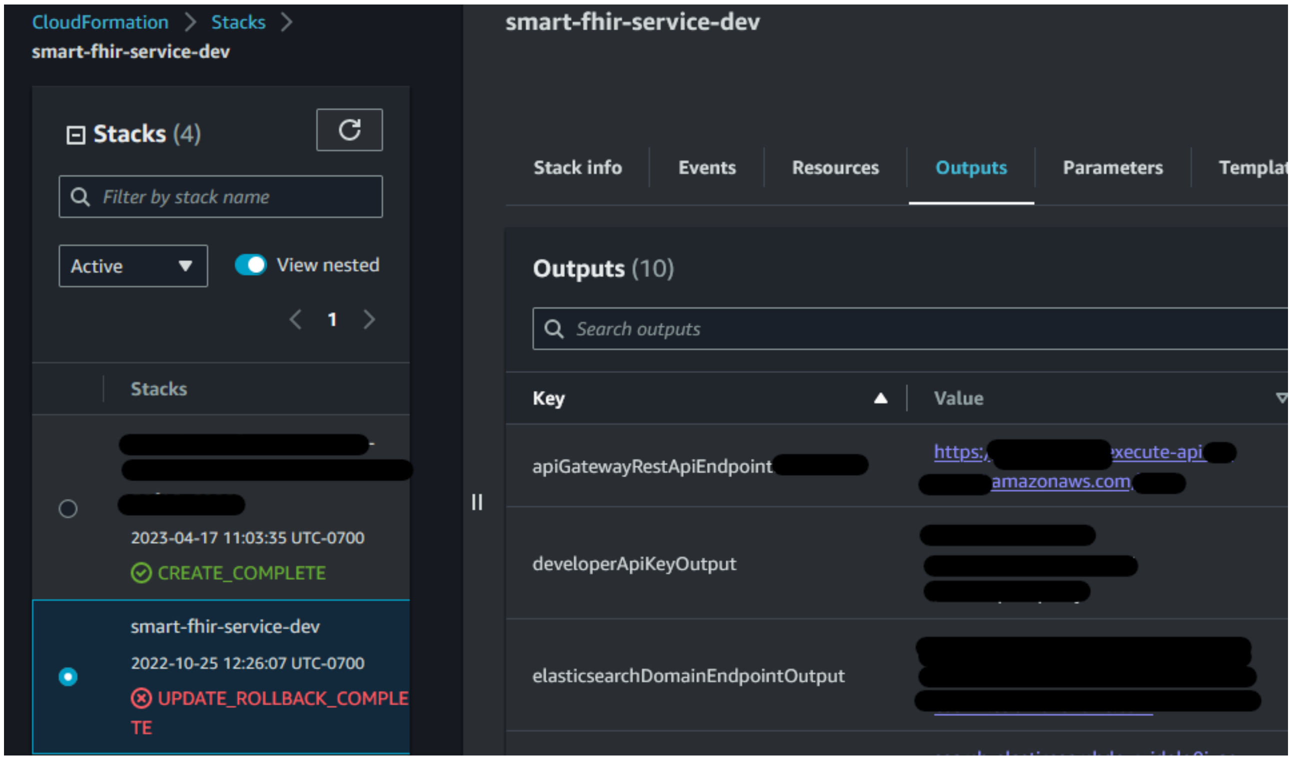
2. For a **Cognito-based FHIR Works on AWS deployment**, you need the following environment variables:

   a. Add the following template to the `.env` file:

```
#### FHIR Works on AWS Credentials
API_URL=
API_KEY=
COGNITO_CLIENT_ID=
COGNITO_USERNAME=
COGNITO_PASSWORD=
```

b. `API_URL` = the API Gateway URL of the FHIR Works deployment. Obtain this from the CloudFormation Stack outputs as `apiGatewayRestApiEndpoint`.

If multi-tenant, append the tenant suffix to the end of the url (e.g. `API_URL/tenant/tenantId`)



CloudFormation > Stacks > smart-fhir-service-dev

smart-fhir-service-dev

Stacks (4)

Filter by stack name

Active ▼ View nested

< 1 >

Stacks

[redacted]
2023-04-17 11:03:35 UTC-0700
CREATE_COMPLETE

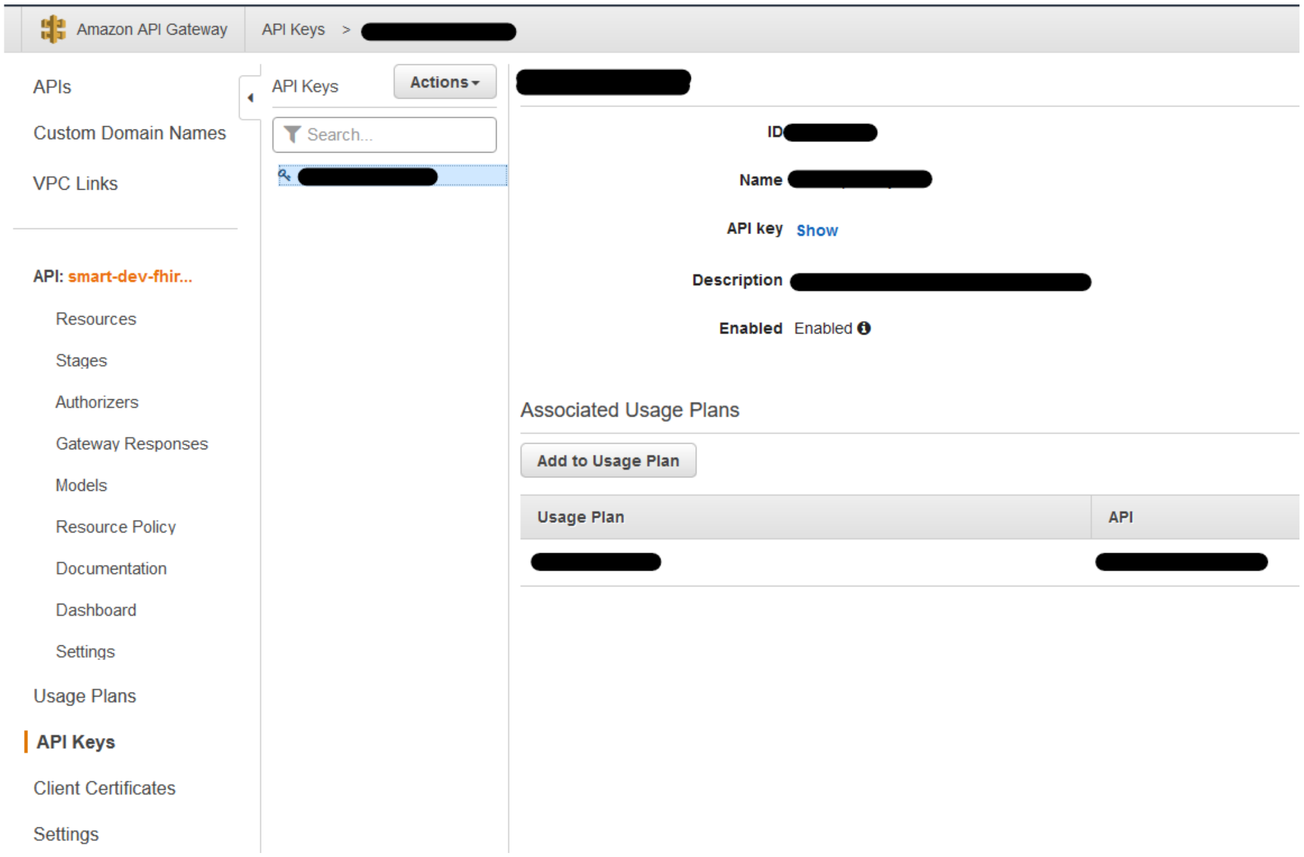
smart-fhir-service-dev
2022-10-25 12:26:07 UTC-0700
UPDATE_ROLLBACK_COMPLETE

Outputs (10)

Search outputs

Key	Value
apiGatewayRestApiEndpoint[redacted]	https:/[redacted]execute-api[redacted] amazonaws.com[redacted]
developerApiKeyOutput	[redacted]
elasticsearchDomainEndpointOutput	[redacted]

c. `API_KEY` = the API Gateway Key for accessing the API. This can be found in the AWS Console by navigating the API Gateway page, and then choosing the FHIR Works on AWS API. Then, navigate to the **API Keys** tab to find the API Key.

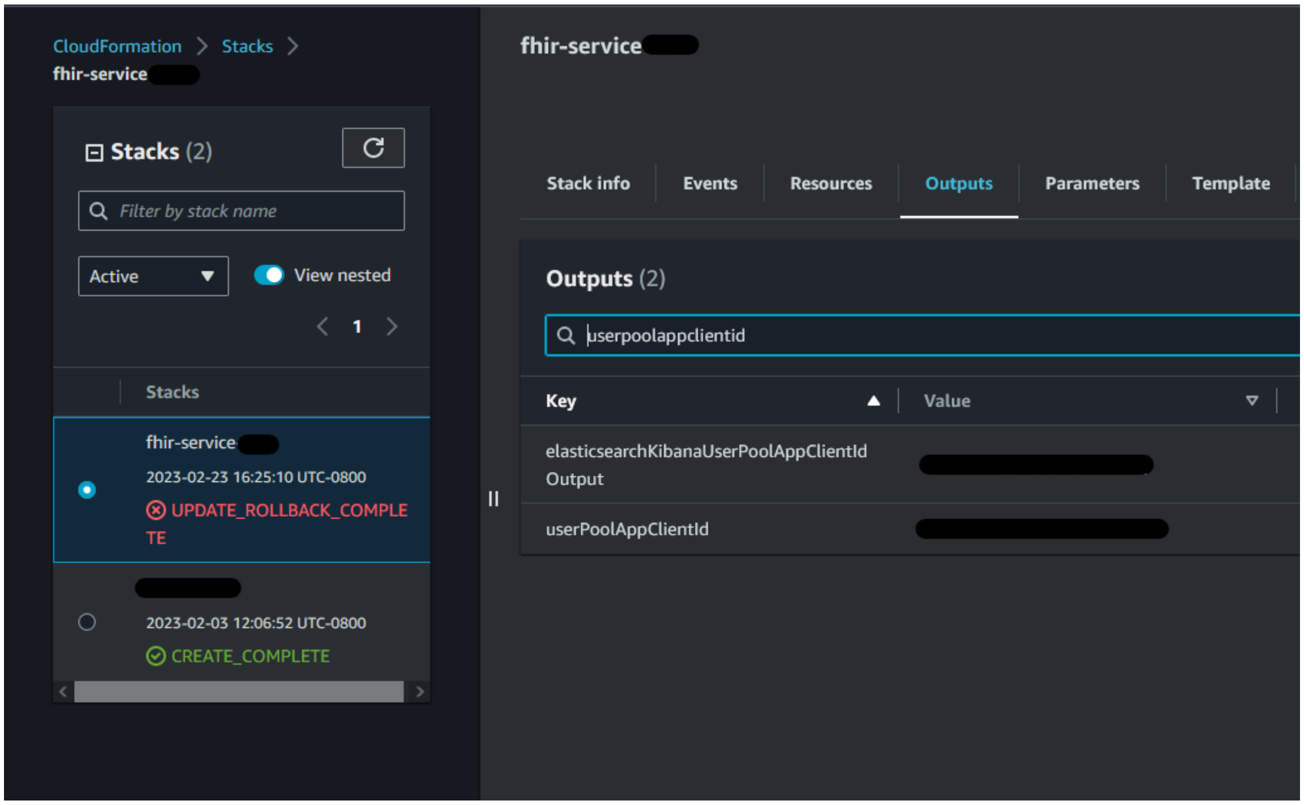


ID [redacted]
Name [redacted]
API key Show
Description [redacted]
Enabled Enabled

Associated Usage Plans

Usage Plan	API
[redacted]	[redacted]

d. `COGNITO_CLIENT_ID` = the client id of the Cognito User Pool. This can be obtained from the CloudFormation Stack outputs as `userPoolAppClientId`.



Key	Value
elasticsearchKibanaUserPoolAppClientId Output	[redacted]
userPoolAppClientId	[redacted]

e. `COGNITO_USERNAME` = the username of a user in the user pool

f. `COGNITO_PASSWORD` = the password of a user in the user pool

If you don't have a Cognito username and password, you can create a new user.

3. For a multi-tenant deployment of FHIR Works on AWS, add `MIGRATION_TENANT_ID` to the `.env` file, equal to the id of the tenant to migrate.

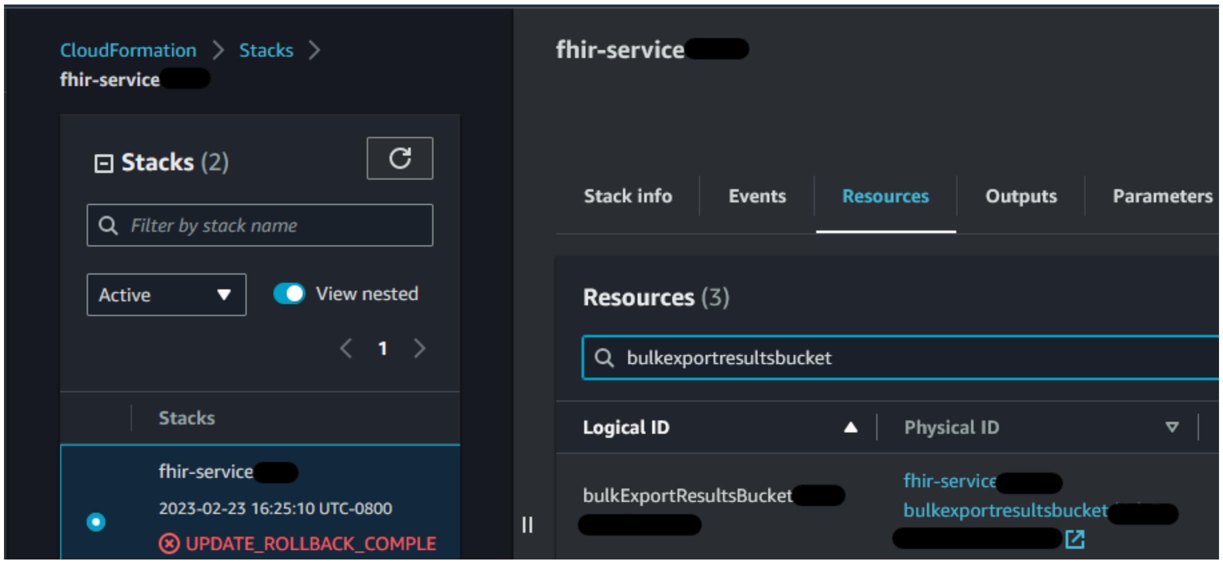
Variables for export bucket and Glue job

Add the following information to the `.env` file for export bucket and glue job name information:

1. Add the following template to the `.env` file:

```
#### FHIR Works on AWS Resources
EXPORT_BUCKET_NAME=
EXPORT_BUCKET_URI=
BINARY_BUCKET_NAME=
API_AWS_REGION=
GLUE_JOB_NAME=
```

2. `EXPORT_BUCKET_NAME` = the name of the `bulkexportresultsbucket` S3 bucket that is deployed alongside FHIR Works on AWS. It can be found in the CloudFormation Stack resources tab.



For a multi-tenant deployment of FHIR Works on AWS, make sure to append the tenant id to the uri. For example:

```
s3://smart-fhir-service-dev-bulkexportresultsbucketabc-
123456789/tenantId
```

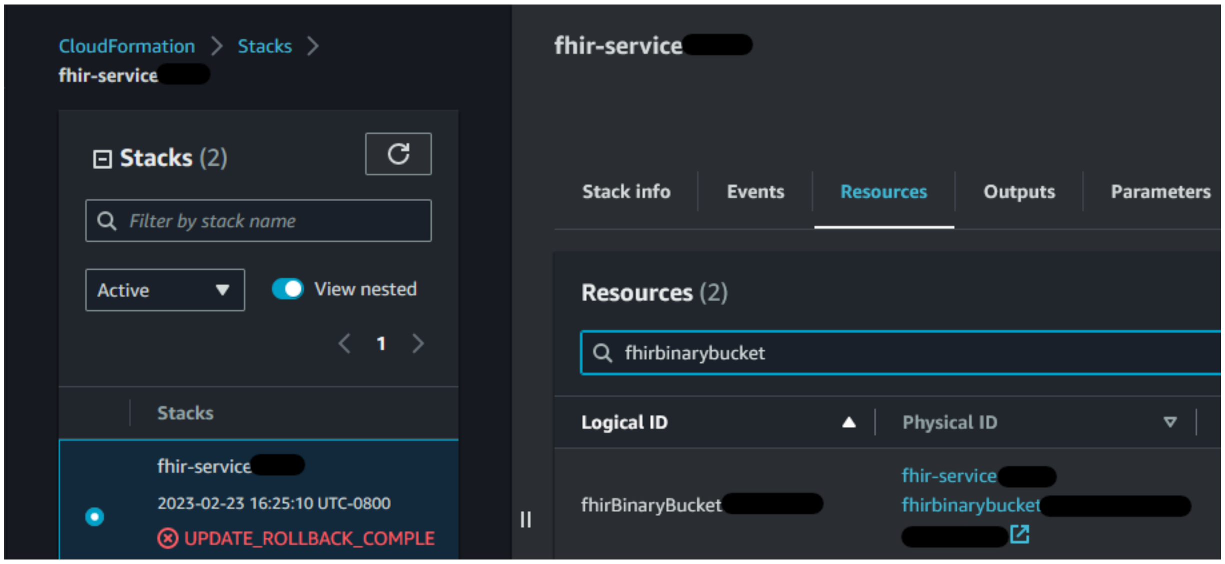
3. Update the `EXPORT_BUCKET_URI` using the name of the `bulkexportresultsbucket` S3 bucket as the `<EXPORT_BUCKET_NAME>.` For example:

   ```
   s3://smart-fhir-service-dev-bulkexportresultsbucketabc-123456789
   ```

   For a multi-tenant deployment of FHIR Works on AWS, make sure to append the tenant id to the uri. For example:

   ```
   s3://smart-fhir-service-dev-bulkexportresultsbucketabc-
   123456789/tenantId
   ```

4. `BINARY_BUCKET_NAME` = the name of the `fhirbinarybucket` S3 bucket that is deployed alongside FHIR Works on AWS. It can be found in the Resources tab of the CloudFormation Stack.



fhir-service[redacted]
2023-02-23 16:25:10 UTC-0800
UPDATE_ROLLBACK_COMPLE

fhirbinarybucket

Logical ID	Physical ID
fhirBinaryBucket[redacted]	fhir-service[redacted] fhirbinarybucket[redacted] [redacted]

5. `GLUE_JOB_NAME` = this value will be found from Step 6 below.

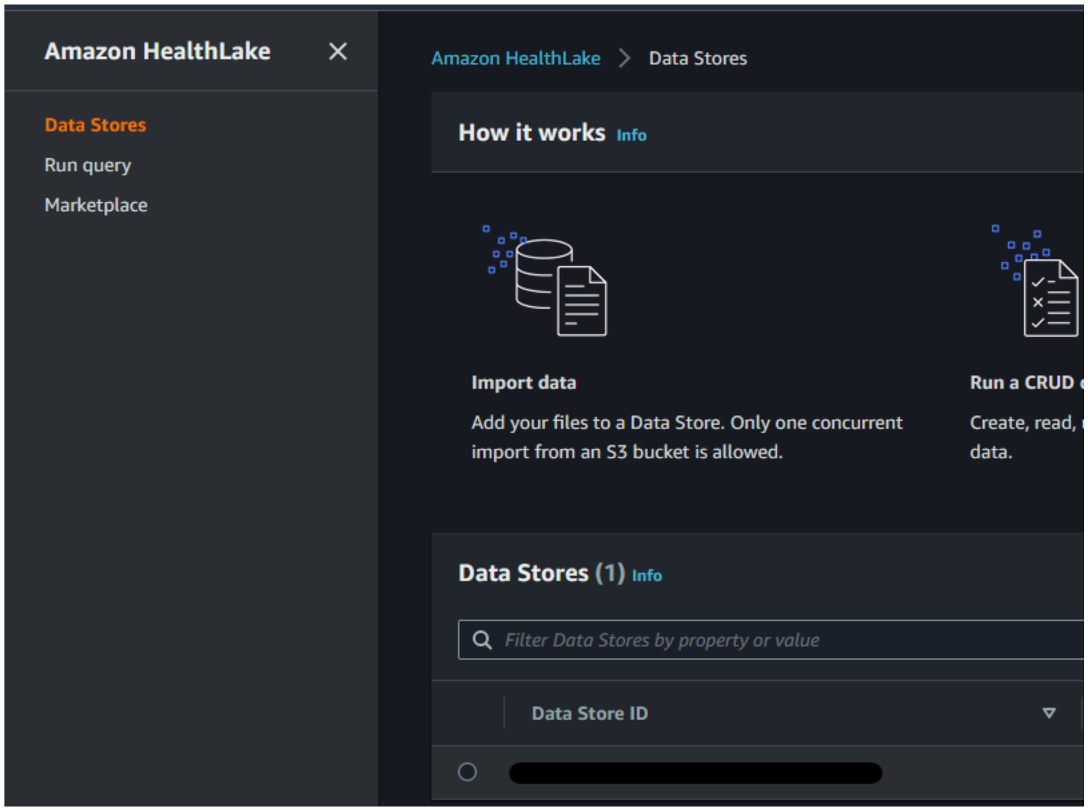
Variables for import

Add the following information to the `.env` file for import:

1. Add the following template to the `.env` file:

```
#### Amazon HealthLake Variables
DATASTORE_ID=
DATASTORE_ENDPOINT=
DATA_ACCESS_ROLE_ARN=
IMPORT_KMS_KEY_ARN=
IMPORT_OUTPUT_S3_BUCKET_NAME=
IMPORT_OUTPUT_S3_URI=
HEALTHLAKE_CLIENT_TOKEN=
```

2. `DATASTORE_ID` = the id of the data store. This can be found in the HealthLake page of the AWS Console.

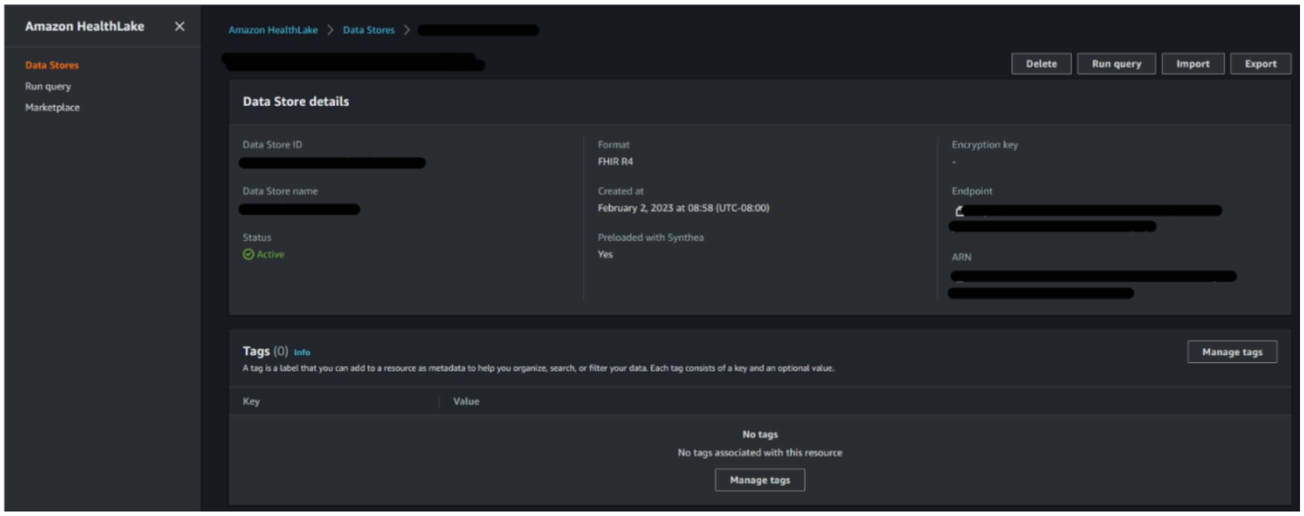


Add your files to a Data Store. Only one concurrent import from an S3 bucket is allowed.

Create, read,
data.

[redacted]

3. `DATASTORE_ENDPOINT` = the endpoint of the Datastore to make requests to. This can be found in the HealthLake page of the AWS Console, on the page displaying information about a specific datastore.



[redacted]

[redacted]

Data Store details

Data Store ID
[redacted]

Data Store name
[redacted]

Status
Active

Format
FHIR R4

Created at
February 2, 2023 at 08:58 (UTC-08:00)

Preloaded with Synthea
Yes

Encryption key
-

Endpoint
[redacted]

ARN
[redacted]

Tags (0) Info

A tag is a label that you can add to a resource as metadata to help you organize, search, or filter your data. Each tag consists of a key and an optional value.

Key	Value

No tags

No tags associated with this resource

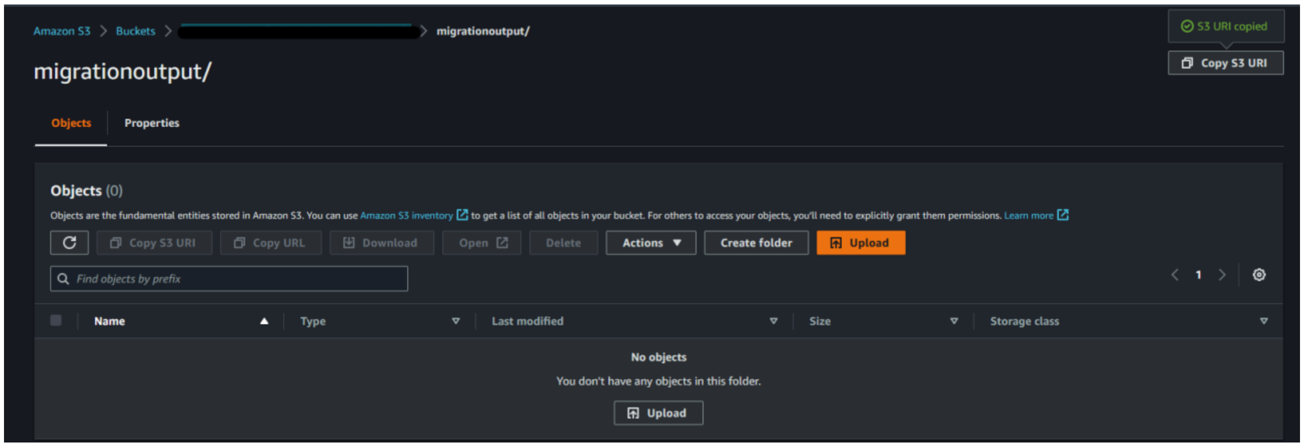
Prepare the environment file

1. Comment out any unused environment variables in the `.env` file.
2. Create an S3 Folder to store the output of the import process, this can be in any location in the same region of your choice, in any S3 bucket.

This folder should be in an Amazon HealthLake supported region.

3. Store the following as environment variables:

   a. `IMPORT_OUTPUT_S3_URI` = the S3 location URI of the output folder location



migrationoutput/

Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class

No objects

You don't have any objects in this folder.

   b. `IMPORT_OUTPUT_S3_BUCKET_NAME` = the name of the bucket containing the output folder location

4. Create a KMS Key that can be used to encrypt/decrypt the data in the import output location in the previous step. Store the Key Amazon Resource Number (ARN) as an environment variable `IMPORT_KMS_KEY_ARN`.

   For the Key Policy, you can use the example policy below, replacing the AWS Account Id with your AWS Account Id.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow administration of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<AWS ACCOUNT
ID>:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

5. Create an IAM Role that has permissions to access the buckets referenced in this migration tool above, and to use the Key to decrypt. Store the ARN of the role as an environment variable `DATA_ACCESS_ROLE_ARN`.

   You can use the Policy Example below to create the IAM Role.

   a. Replace `<EXPORT BUCKET NAME>` with the name of the export bucket name.

   b. Replace `<IMPORT OUTPUT LOCATION NAME>` with the name of the Import Output Bucket.

   c. Replace `<KMS KEY ID>` with the ID of the KMS Key created in the previous step.

   d. Replace `<AWS ACCOUNT ID>` with the Account Id of your AWS Account.

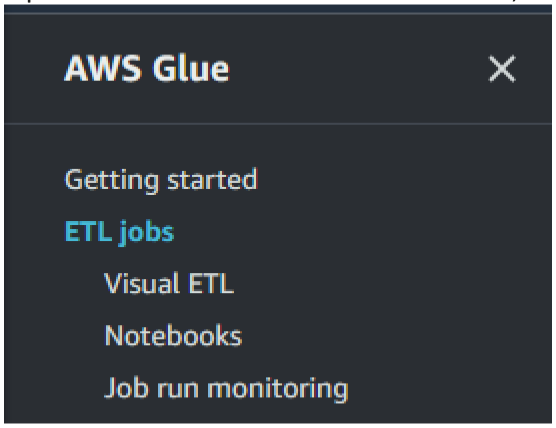
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::<EXPORT BUCKET NAME>/**"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketPublicAccessBlock",
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::<EXPORT BUCKET NAME>",
                "arn:aws:s3:::<IMPORT OUTPUT BUCKET
NAME>/*"
            ],
            "Effect": "Allow"
```

```
          },
          {
              "Action": [
                  "s3:PutObject"
              ],
              "Resource": [
                  "arn:aws:s3:::<IMPORT OUTPUT BUCKET
NAME>/*",
                  "arn:aws:s3:::<IMPORT OUTPUT BUCKET
NAME>"
              ],
              "Effect": "Allow"
          },
          {
              "Action": [
                  "kms:DescribeKey",
                  "kms:GenerateDataKey*",
                  "kms:Encrypt",
                  "kms:ReEncrypt*",
                  "kms:Decrypt"
              ],
              "Resource": [
                  "arn:aws:kms:us-east-2:<AWS ACCOUNT
ID>:key/<KMS KEY ID>"
              ],
              "Effect": "Allow"
          }
      ]
  }
```

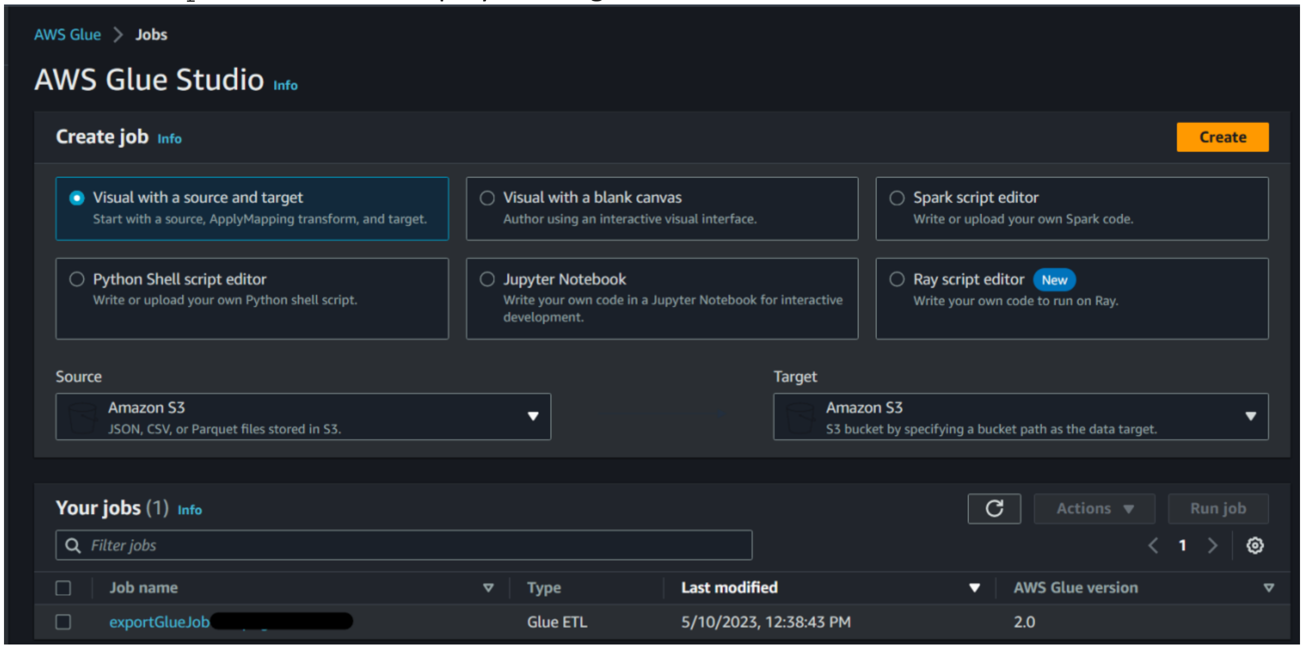
6. Choose a random string value for the `HEALTHLAKE_CLIENT_TOKEN` environment variable.

Step 6: Create Migration Export Job

1. Open AWS Glue from the AWS Console, and choose **ETL Jobs** from the menu on the left.

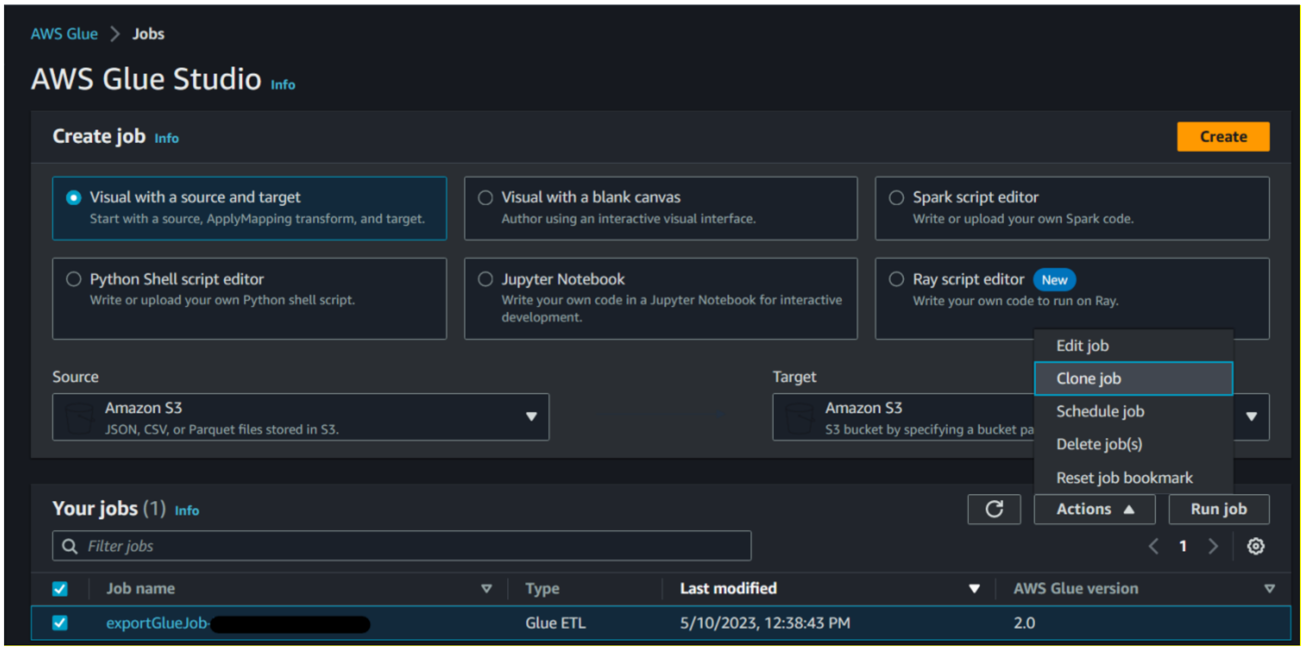


2. Locate the `exportGlueJob` deployed alongside the FHIR Works on AWS Stack.


AWS Glue > Jobs
AWS Glue Studio Info
Create job Info
Create
Visual with a source and target
Start with a source, ApplyMapping transform, and target.
Visual with a blank canvas
Author using an interactive visual interface.
Spark script editor
Write or upload your own Spark code.
Python Shell script editor
Write or upload your own Python shell script.
Jupyter Notebook
Write your own code in a Jupyter Notebook for interactive development.
Ray script editor New
Write your own code to run on Ray.
Source
Amazon S3
JSON, CSV, or Parquet files stored in S3.
Target
Amazon S3
S3 bucket by specifying a bucket path as the data target.
Your jobs (1) Info
Actions
Run job
Filter jobs

Job name	Type	Last modified	AWS Glue version
exportGlueJob[redacted]	Glue ETL	5/10/2023, 12:38:43 PM	2.0

3. Clone the `exportGlueJob` by selecting the **Job name** and then choose **Clone Job** from the Actions menu.


AWS Glue > Jobs
AWS Glue Studio Info
Create job Info
Create
Visual with a source and target
Start with a source, ApplyMapping transform, and target.
Visual with a blank canvas
Author using an interactive visual interface.
Spark script editor
Write or upload your own Spark code.
Python Shell script editor
Write or upload your own Python shell script.
Jupyter Notebook
Write your own code in a Jupyter Notebook for interactive development.
Ray script editor New
Write your own code to run on Ray.
Edit job
Clone job
Schedule job
Delete job(s)
Reset job bookmark
Source
Amazon S3
JSON, CSV, or Parquet files stored in S3.
Target
Amazon S3
S3 bucket by specifying a bucket pa
Your jobs (1) Info
Actions
Run job
Filter jobs

Job name	Type	Last modified	AWS Glue version
✓ exportGlueJob[redacted]	Glue ETL	5/10/2023, 12:38:43 PM	2.0

4. Enter the name of the cloned job as the value for the `GLUE_JOB_NAME` environment variable.

5. From the Script tab of the cloned export job, replace the code with the `export-script.py` script found in the migration tool folder at `<ROOT DIR>/fwoa-tools/src/export-script.py`.

Step 7 (Optional): Authentication Setup

If you prefer to setup the authentication to your FHIR Works on AWS solution using client credential setup for SMART-on-FHIR deployment, you need to replace the code as below for the `getAuthToken` method in `migrationUtils.ts` in the `fwoa-tools/src` folder. You need to fill the correct `audience` parameter in the code below which is your FHIR Resource server.

```
async function getAuthToken(
  username: string,
  password: string,
  clientId: string,
  clientPw: string,
  oauthApiEndpoint: string,
  requestAdditionalScopes?: boolean
): Promise<string> {


  const data = stringify({
    grant_type: 'client_credentials',
    scope: 'system/*.read',
    audience: <your audience>

  });

  const authToken = `Basic
${Buffer.from(`${clientId}:${clientPw}`).toString('base64')}`;

  const config: AxiosRequestConfig = {
    method: 'post',
    url: `${oauthApiEndpoint}/token`,
    headers: {
      Accept: 'application/json',
      Authorization: authToken,
      'Content-Type': 'application/x-www-form-urlencoded'
    },
    data
  };

  const response = await axios(config);
  return response.data.access_token;
}
```

It is expected that the client has the appropriate scope configured in Identity Provider to export the data, in this case `system/*.read`.

Step 8: Change Glue Job Configurations

1. Open your Glue Job in the AWS Console.
2. Set the **Job timeout value** to 43,200 (30 days) to ensure the job does not timeout. By default, the value is set to two days.
3. Set the requested number of workers to 60.
   - Update the `glueJobRole` IAM Poilicy for `ddbAccess` to match the following:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "dynamodb:DescribeTable",
        "dynamodb:Scan",
        "dynamodb:ExportTableToPointInTime",
        "dynamodb:DescribeExport"
      ],
      "Resource": [
        "arn:aws:kms:<region>:<accountId>:key/<keyId> ",

"arn:aws:dynamodb:<region>:<accountId>:table/resource-db-
<stage>",

"arn:aws:dynamodb:<region>:<accountId>:table/resource-db-
<stage>/export/*"
      ]
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:Scan",
        "dynamodb:ExportTableToPointInTime",
        "dynamodb:DescribeExport"
      ],
      "Resource":
"arn:aws:dynamodb:<region>:<accountId>:table/resource-db-
<stage>"
    }
  ]
}
```

Export FHIR resources from DynamoDB

For SMART-on-FHIR deployments

1. From the terminal, enter the `<ROOT FOLDER>/fwoa-tools/src` directory.
2. For checking configurations before starting the export, run the following dry run command:

   ```
   nohup ts-node ./migrationExport.ts -d -s
   ```

3. To export all resources, run the following command:

   ```
   nohup ts-node ./migrationExport.ts -s
   ```

4. If you have more resources since the previous export, run the following command:

   ```
   nohup ts-node ./migrationExport.ts -s -t "2023-05-
   15T05:00:00.00Z"
   ```

   The time stamp must be in ISO 8601 format.

   The start time of any previous export can be found in the `export_output_... log` file. It is recommended to have some overlap to avoid missing any data. For example, if the previous export's start time was `2023-05-15T05:00:00.00Z`, you can start the next export with `–t 2023-05-15T04:00:00.00Z`.

5. For a multi-tenant export, leverage snapshots for quicker exports. Glue Job snapshots the database before transformations and data exports. Use the following command with the `.env` variables updated to the tenant name:

   ```
   nohup ts-node ./migrationExport.ts -s -e -l
   s3://snapshotlocation
   ```

Note

Data added to the FHIR server after the last snapshot will not be available. To get an updated snapshot with the new data, use the `-e -l` parameters described in step 5 to get a new snapshot of the database.

For Cognito deployments

1. For checking configurations before starting the export, run the following dry run command:

```
nohup ts-node ./migrationExport.ts -d
```

2. To export all resources, run the following command:

```
nohup ts-node ./migrationExport.ts
```

3. If you have more resources since the previous export, run the following command

```
nohup ts-node ./migrationExport.ts -t "2023-05-
15T05:00:00.00Z"
```

The time stamp must be in ISO 8601 format

The start time of any previous export can be found in the `export_output_...` log file. It is recommended to have some overlap to avoid missing any data. For example, if the previous export's start time was `2023-05-15T05:00:00.00Z`, you can start the next export with `–t 2023-05-15T04:00:00.00Z`.

Note

The export Job ID can be found from the logs of the export job. You can use this Job ID to check on the status of an export by making a GET request:

```
GET <API_URL>/$export/jobId
```

The response headers for this request will contain a field for the progress of the export job: x-progress: in-progress. Once the job is complete, the response will be a JSON body with pre-signed S3 URLs of the exported data. You can download the exported data using those URLs. For 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": []
}
```

4. For a multi-tenant export, leverage snapshots for quicker exports. Glue Job snapshots the database before transformations and data exports. Use the following command with the `.env` variables updated to the tenant name:

```
nohup ts-node ./migrationExport.ts -s -e -l
s3://snapshotlocation
```

Note

Data added to the FHIR server after the last snapshot will not be available. To get an updated snapshot with the new data, use the `-e -l` parameters described in step 4 to get a new snapshot of the database.

Export binary resource from S3 bucket

1. For checking configurations before starting the binary export, run the following dry run command:

```
nohup ts-node ./binaryConverter.ts -d
```

2. Run the Binary Conversion script with the following command:

```
nohup ts-node ./binaryConverter.ts
```

S3 bucket replication of exported data (optional)

If your FHIR Works on AWS deployment exists in a different region than is supported by Amazon HealthLake, you will need to replicate your S3 Export bucket to a supported region before proceeding to the next step.

> **Note**
>
> You will incur inter-region data transfer costs. Refer to the pricing calculator to estimate charges.

You can run the following command to copy objects from one bucket to another:

```
aws s3 sync s3://<EXPORT BUCKET NAME> s3://<DESTINATION
BUCKET NAME>
```

For `EXPORT BUCKET NAME`, you can retrieve the value from the previous `.env` setup.

For additional information, please see: https://repost.aws/knowledge-center/move-objects-s3-bucket

Please ensure that the environment variables for export bucket are updated after the replication, so that the import step looks for resources in the correct bucket. Namely, update the following environment variables with the destination bucket's information:

- `EXPORT_BUCKET_NAME`
- `EXPORT_BUCKET_URI`

You will also need to update the IAM Role and Policy accordingly.

Import into Amazon HealthLake

1. To check configurations before starting the import, run the following dry run command:

```
nohup ts-node ./migrationImport.ts -d
```

2. Run the Import script:

```
nohup ts-node ./migrationImport.ts
```

3. To run without stopping execution on a partially successful import job, you can specify the `--continueOnError` flag when executing the import script

   ```
   nohup ts-node ./migrationImport.ts -c
   ```

Verification of Import (optional)

If you would like to verify that all resources in Amazon HealthLake match those in FHIR Works on AWS, you can leverage the verification script.

For SMART-on-FHIR deployments

1. To check configurations before starting the verification, run the following dry run command:

   ```
   nohup ts-node ./migrationVerify.ts -s -d
   ```

2. Run the Import script:

   ```
   nohup ts-node ./migrationVerify.ts -s
   ```

3. To run without stopping execution on a partially successful verification job, you can specify the `--continueOnError` flag when executing the import script

   ```
   nohup ts-node ./migrationVerify.ts -c
   ```

For Cognito deployments

1. To check configurations before starting the verification, run the following dry run command:

   ```
   nohup ts-node ./migrationVerify.ts -d
   ```

2. Run the Import script:

   ```
   nohup ts-node ./migrationVerify.ts
   ```

3. To run without stopping execution on a partially successful verification job, you can specify the `--continueOnError` flag when executing the import script

   ```
   nohup ts-node ./migrationVerify.ts -c
   ```

Migration tool logs

Logs for the migration process are stored on the local environment. Each migration step has its own log file name prefix:

- `export_output_...`
- `binary_conversion_output_...`
- `import_output_...`
- `import_verification_...`

These files will be suffixed with the timestamp of the execution, and contain information on all activities performed by the migration tool as well as any errors that occur.

Teardown for migration tool and FHIR Works on AWS

Step 1. Delete Cloud9 environment

Delete `<fwoa-tools>` folder in Cloud9.

Make sure you move the settings of the Cloud9 EC2 instance back to default sleep configuration to reduce costs after migration or shutdown the instance from the AWS Console.

Step 2. Glue Job Script and Configuration

Please follow your organization's best practices for updating production environments. To restore the original FHIR Works on AWS export script, navigate to the `gluescriptsbucket` as described in Step 6: Create Migration Export Job. Use the downloaded original copy of the `export-script.py` to replace the script in the Bucket.

Revert the Glue Job timeout value to the original value of 2880.

Step 3. Shut down FHIR Works on AWS

Follow your organization's best practices to shut down the FHIR Works on AWS solution. Before shutting down FHIR Works on AWS solution, if you need to save CloudWatch Logs from FHIR Works on AWS, you can use the AWS CLI with `create-export-task` to export the logs to an S3 bucket. Please follow the instructions to create an export task.

To terminate FHIR Works on AWS solution, you can delete the FHIR Works on AWS CloudFormation stack from your account through AWS Console.

Step 4. Delete Resources Created for Migration Process

Please ensure that any KMS Keys, migration logs in local environment, IAM Policies, or other AWS resources created during this migration process are deleted per your organization's best practices.

Archiving FHIR Works on AWS CloudWatch logs

Customers who prefer to archive FHIR Works on AWS CloudWatch logs for compliance can use the AWS CLI to create an export task so that you can export data from a log group to an Amazon S3 bucket. Assuming you have created an IAM user and a destination bucket with permissions set up for AWS CLI to put objects into the bucket, an example command will look like:

```
aws logs create-export-task --profile <Profile Name> --
task-name <Task Name> --log-group-name <Cloud Watch Log
Group Name> --from <From Time> --to <To Time> --
destination <Bucket Name> --destination-prefix <Custom
destination prefix> --region <Region> --profile <AWS CLI
Profile>
```

For more information on the command usage, see the create export documentation.

Troubleshooting

1. Where can I find logs of the migration process?

   Logs can be found in the same directory as the migration scripts after running each step. These logs will be named as: `export_output_<timestamp>`, `import_output_<timestamp>`, or `binary_conversion_output_<timestamp>` depending on the script that was run. Additionally, FHIR Works on AWS Export logs can be found in CloudWatch under `the /aws-glue/jobs/fhir-works-export-security-config...` and `BulkExportSM-Logs...` log groups. Amazon HealthLake Import logs can be found in the `<IMPORT_OUTPUT_S3_BUCKET_NAME>` S3 Bucket specified earlier.

2. What if my export did not succeed?

   If the export did not succeed, consult the logs to find the error that has occurred. If it is related to authorization or authentication, double check the environment variable setup, and that your IAM/KMS Policies are able to successfully perform the actions needed. Otherwise, ensure that your timeout configuration in AWS Glue is long enough to sustain the export job. Note that only one export job can be running on a FHIR Works on AWS server by default. You can calculate the total size of the export job folder to verify if the number of resources exported matches that of the DynamoDB resource table.

3. What if my import did not succeed?

   If the import did not succeed, consult the logs to find the error that has occurred. Refer to the HealthLake documentation for additional troubleshooting details. If it is related to authorization or authentication, double check the environment variable setup, and that your IAM/KMS Policies are able to successfully perform the actions needed. You can retry imports as they are idempotent and resources that are imported twice will overwrite the previous version. For other errors, consult your Account SA and the HealthLake guide to ensure that your data is supported by HealthLake.

4. What should I check before starting the import job?

   HealthLake limits a single import job to 500 GB. Check the size of files and folders to ensure they are within limits:

   – Check that individual folders have no more than 500 GB of data.

   – Move .ndjson files to separate folders so that there are no folders more than 500 GB of data.

   – No single file can be more than 5 GB. If your binary file is more than 5 GB, it will not be imported into HealthLake. Please contact support if you have files greater than 5 GB.

5. How do I check the status of the import job?

   You can run the following command to check the status of an import job:

   ```
   aws healthlake describe-fhir-import-job --datastore-id
   <datastore id> --job-id <job id> --region <your region>
   ```

6. How to check if my resources are successfully imported into Amazon HealthLake?

   The import script automatically checks each resource in each import job to make sure that the representation in HealthLake is present and matching the data in FHIR Works on AWS. If there are any mismatches or missing data, the script will log an error and fail the import script.

7. What if binary conversion script did not succeed?

   Double check that each Binary resource in FHIR Works on AWS has a corresponding binary object uploaded into the Binary S3 Bucket. You can also consult the logs for the Binary Conversion to see where any errors may have occurred.

8. What if I need more guidance or support during the migration process?

   Please work with your account team, AWS Support or contact fwoa-migration-support@amazon.com for support.

Frequently asked questions

1. What data is migrated from FHIR Works on AWS to Amazon HealthLake?

   Migration tool exports FHIR resources stored in S3 bucket and DynamoDB table from FHIR Works on AWS into an interim S3 bucket. Migration tool imports the FHIR resources in the interim S3 bucket into Amazon HealthLake.

2. Which versions of FHIR Works on AWS can be migrated?

   Any version of FHIR Works on AWS v4.2.0 or greater and v.2.4.0-smart or greater.

3. Where should I run the migration scripts?

   We recommend to run the migration scripts from inside an AWS account using Cloud9 IDE.

4. How do I know if the data got imported successfully?

   You can check the migration tool log in addition to transaction logs in Amazon HealthLake for the status of the import. Any failures will be reported in migration tool log.

5. How can reduce downtime and maintain high availability during migration?

   If you prefer to maintain high availability it is recommended to use the since parameter in export. Please record this parameter in every export. This parameter should be used as an input in the subsequent batch export. We recommend overlapping the time information to avoid any data loss.

6. How long will the migration process take?

   Migration time is a function of how many FHIR resources you have in FHIR Works on AWS solution. Please work with your Account SA to determine the effort and time needed for migration as it varies by customer and use cases.

7. How can I assess Amazon HealthLake feature compatibility with FHIR Works on AWS?

   Please work with your account team, AWS Support or contact fwoa-migration-support@amazon.com for support.

8. Which regions are supported by Amazon HealthLake?

   Amazon HealthLake is available in US East (Ohio), US East (N. Virginia), US West (Oregon) and Asia Pacific (Mumbai). Visit the AWS Region Table to see all the regions.

9. How can I transform the data before export?

   If you need additional transformations to clean up data being exported from FHIR Works on AWS, you can leverage the `additional_transformations` function provided in the `export-script.py` Glue script.

10. Can I migrate if my FHIR Works on AWS is hosted on a region not supported by Amazon HealthLake?

    Yes. After export, you need to replicate the exported bucket to a supported region of Amazon HealthLake. After replication, you can follow the normal import process of Amazon HealthLake.

11. Which implementation guides are supported by Amazon HealthLake?

    Amazon HealthLake currently supports the following implementation guides: us-core, ABDM and CarinBB. For full feature parity information and IGs in the roadmap, please work with your account team.

12. How can I get the pricing information for Amazon HealthLake?

    Please check Amazon HealthLake pricing.

13. What if I need more guidance or support to migrate from AWS?

    Please work with your account team, AWS Support or contact fwoa-migration-support@amazon.com for support.

References

https://docs.aws.amazon.com/healthlake/

https://github.com/aws-solutions/fhir-works-on-aws

Contributors

- Balaji Raman
- Sukeerth Vegaraju
- Karina Cadette
- Bakha Nurzhanov
- Ananya Gupta
- Brandi Hopkins
- Sridhar Ramachandran

Revisions

Date	Change
July 2023	Initial release

Date	Change
August 2023	Revisions for release v6.1.2. For more information, see the Changelog.md file.

Notices