Start Lab 01:30:00

Ingesting FHIR Data with the Healthcare API

Lab 1 hour No cost Introductory

GSP457

Overview

Cloud Healthcare API provides a managed solution for storing and accessing healthcare data in Google Cloud, providing a critical bridge between existing care systems and applications hosted on Google Cloud. Using the API, you can unlock significant new capabilities for data analysis, machine learning and application development, and use these capabilities to build the next generation of healthcare solutions.

In this lab you will discover and use the basic functionality of Cloud Healthcare API using Fast Healthcare Interoperability Resources (FHIR) data model, how to export data to BigQuery, and how to access data in BigQuery via SQL.

What you learn

In this lab, you will:

- Gain a general understanding of Cloud Healthcare API and its role in managing healthcare data.
- Learn how to create Cloud Healthcare API datasets and stores.
- Import and export FHIR data using the Cloud Healthcare API.
- Export data from Cloud healthcare API to BigQuery
- Access data in BigQuery via SQL

Healthcare API introduction

Cloud Healthcare API provides a managed solution for storing and accessing healthcare data in Google Cloud, providing a critical bridge between existing care systems and applications hosted on Google Cloud. Using the API, you can unlock significant new capabilities for data analysis, machine learning and application development, and use these capabilities to build the next generation of healthcare solutions.

The API is comprised of three modality-specific interfaces that implement key industry-wide standards for healthcare data:

- FHIR, an emerging standard for health data interchange
- HL7v2, the most widely adopted method for health systems integration
- Import and export FHIR data using the Cloud Healthcare API
- Export data from Cloud healthcare API to BigQuery
- Access data in BigQuery via SQL

Healthcare API introduction

Cloud Healthcare API provides a managed solution for storing and accessing healthcare data in Google Cloud, providing a critical bridge between existing care systems and applications hosted on Google Cloud. Using the API, you can unlock significant new capabilities for data analysis, machine learning and application development, and use these capabilities to build the next generation of healthcare solutions.

The API is comprised of three modality-specific interfaces that implement key industry-wide standards for healthcare data:

- FHIR, an emerging standard for health data interchange
- HL7v2, the most widely adopted method for health systems integration
- Import and export FHIR data using the Cloud Healthcare API
- Export data from Cloud healthcare API to BigQuery
- Access data in BigQuery via SQL

Healthcare API introduction

Cloud Healthcare API provides a managed solution for storing and accessing healthcare data in Google Cloud, providing a critical bridge between existing care systems and applications hosted on Google Cloud. Using the API, you can unlock significant new capabilities for data analysis, machine learning and application development, and use these capabilities to build the next generation of healthcare solutions.

The API is comprised of three modality-specific interfaces that implement key industry-wide standards for healthcare data:

- FHIR, an emerging standard for health data interchange
- HL7v2, the most widely adopted method for health systems integration
- Import and export FHIR data using the Cloud Healthcare API
- Export data from Cloud healthcare API to BigQuery
- Access data in BigQuery via SQL

Healthcare API introduction

Cloud Healthcare API provides a managed solution for storing and accessing healthcare data in Google Cloud, providing a critical bridge between existing care systems and applications hosted on Google Cloud. Using the API, you can unlock significant new capabilities for data analysis, machine learning and application development, and use these capabilities to build the next generation of healthcare solutions.

The API is comprised of three modality-specific interfaces that implement key industry-wide standards for healthcare data:

- FHIR, an emerging standard for health data interchange
- HL7v2, the most widely adopted method for health systems integration
- Import and export FHIR data using the Cloud Healthcare API
- Export data from Cloud healthcare API to BigQuery

- Access data in BigQuery via SQL

Healthcare API introduction

Cloud Healthcare API provides a managed solution for storing and accessing healthcare data in Google Cloud, providing a critical bridge between existing care systems and applications hosted on Google Cloud. Using the API, you can unlock significant new capabilities for data analysis, machine learning and application development, and use these capabilities to build the next generation of healthcare solutions.

The API is comprised of three modality-specific interfaces that implement key industry-wide standards for healthcare data:

- FHIR, an emerging standard for health data interchange
- HL7v2, the most widely adopted method for health systems integration
- Export data from Cloud healthcare API to BigQuery
- Access data in BigQuery via SQL

Healthcare API introduction

Cloud Healthcare API provides a managed solution for storing and accessing healthcare data in Google Cloud, providing a critical bridge between existing care systems and applications hosted on Google Cloud. Using the API, you can unlock significant new capabilities for data analysis, machine learning and application development, and use these capabilities to build the next generation of healthcare solutions.

The API is comprised of three modality-specific interfaces that implement key industry-wide standards for healthcare data:

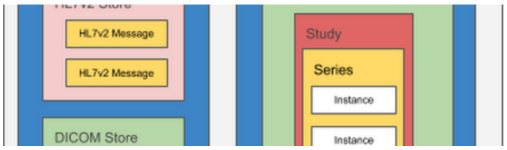
- FHIR, an emerging standard for health data interchange
- HL7v2, the most widely adopted method for health systems integration
- Export data from Cloud healthcare API to BigQuery
- Access data in BigQuery via SQL

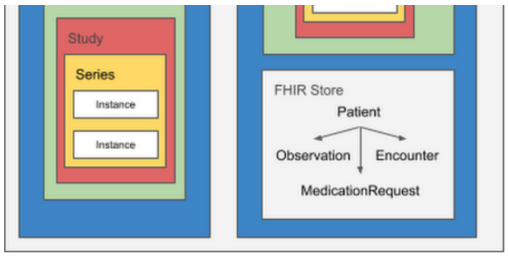
Healthcare API introduction

Cloud Healthcare API provides a managed solution for storing and accessing healthcare data in Google Cloud, providing a critical bridge between existing care systems and applications hosted on Google Cloud. Using the API, you can unlock significant new capabilities for data analysis, machine learning and application development, and use these capabilities to build the next generation of healthcare solutions.

The API is comprised of three modality-specific interfaces that implement key industry-wide standards for healthcare data:

- FHIR, an emerging standard for health data interchange
- HL7v2, the most widely adopted method for health systems integration





There are many ways to structure datasets and stores. As you design systems that use the Cloud Healthcare API, you may want to take the following into consideration:

- **Security and access control:** Rules can be defined at both a dataset and store level, but you may choose to group all data for a particular application into the same dataset, and set access control rules such that only that application can access the dataset.
- **Application requirements:** An application processing different types of data may have all of its data for all modalities in a single dataset.
- **Source systems:** Often, the structure of healthcare data can vary according to the source system and modality. Separating data for different source systems into their own datasets may facilitate processing.
- **Intended use:** Data from different systems can have different intended uses, such as research, analytics, or machine learning predictions. Grouping data by intended use may facilitate ingestion into the target system.
- **Separating ePHI from de-identified data:** Cloud Healthcare API data de-identification functions read from a source dataset and write the output into a new dataset that you specify. If you are preparing data to be used by researchers, for example, this approach to de-identifying data may be a consideration in how you use datasets to segregate data.

MLLP Adapter

The minimal lower layer protocol (MLLP) is the standard used for transmitting HL7v2 messages over TCP/IP connections within a network, such as a hospital.

MLLP does not offer an exact mapping to the Cloud Healthcare API HL7v2 REST API], which uses HTTP. Therefore, an MLLP adapter must be used to convert messages transmitted over MLLP into a format that an HTTP/REST API can accept. To transmit messages over MLLP and then to the Cloud Healthcare API, use the Google Cloud MLLP adapter.

There are many ways to structure datasets and stores. As you design systems that use the Cloud Healthcare API, you may want to take the following into consideration:

- **Security and access control:** Rules can be defined at both a dataset and store level, but you may choose to group all data for a particular application into the same dataset, and set access control rules such that only that application can access the dataset.
- **Application requirements:** An application processing different types of data may have all of its data for all modalities in a single dataset.
- **Source systems:** Often, the structure of healthcare data can vary according to the source system and modality. Separating data for different source systems into their own datasets may facilitate processing.
- **Intended use:** Data from different systems can have different intended uses, such as research, analytics or machine learning predictions. Grouping data by intended use may facilitate ingestion into the target system.
- **Separating ePHI from de-identified data:** Cloud Healthcare API data de-identification functions read from a source dataset and write the output into a new dataset that you specify. If you are preparing data to be used by researchers, for example, this approach to de-identifying data may be a consideration in how you use datasets to segregate data.

API structure

Data in the Cloud Healthcare API datasets and stores can be accessed and managed using a REST API that identifies each store using its project, location, dataset, store type and store name. This API implements modality-specific standards for access that are consistent with industry standards for that modality. For example, the Cloud Healthcare DICOM API natively provides operations for reading DICOM studies and series that are consistent with the DICOMweb standard, and supports the DICOM DIMSE C-STORE protocol via an open-source adapter. Similarly, the FHIR API provides operations for accessing or searching FHIR entity types that is based on the FHIR standard, and the HL7v2 API provides operations for reading and searching HL7v2 messages based on HL7v2 message or segment criteria.

Operations that access a modality-specific store use a request path that is comprised of two pieces: a base path, and a modality-specific request path. Administrative operations—which generally operate only on locations, datasets and stores—may only use the base path, but data modality-specific retrieval operations use both the base path (for identifying the store to be accessed) and request path (for identifying the actual data to be retrieved).

To reference a particular store within a Cloud Healthcare API dataset, you would use a base path structured like this:

`/projects/<PROJECT>/locations/<LOCATION>/datasets/<DATASET>/<STORE-TYPE>/<STORE-NAME>`

A concrete base path example might look like this:

/projects/myProj/locations/`REGION`/datasets/central-ds1/hl7V2Stores/clinical-store1

which references a Cloud Healthcare HL7 v2 store in the Google Cloud project "myProj", in the "`REGION`" region, in a dataset called "central-ds1", and with a name of "clinical-store1". This is an HL7 v2 store because of the "hl7V2Stores" type; if you want to access a FHIR store in the same dataset you can use the "fhirStores" type, and if the store contained DICOM data you can used the "dicomStores" type.

To access a specific piece of data, the base path is used in combination with a request path that is formatted according to the appropriate modality standard. For example, a request to read a specific FHIR "Patient" entity using the entity ID might look like this:

`<basePath>/resources/Patient/{patient_id}`

with `/Patient/{patient_id}` being a path—structured according to the FHIR standard—for the Patient resource whose identifier is specified by `{patient_id}`. Similarly, DICOMweb requests to a DICOM store might look like this:

`<basePath>/dicomWeb/studies/{study_id}/series?PatientName={patient_name}`

where `{study_id}` identifies a particular DICOM study, and the patient's name is specified by `{patient_name}`. In this example, the path specification is consistent with the DICOMweb standard path structure.

Setup and requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab---remember, once you start, you cannot pause a lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

How to start your lab and sign in to the Google Cloud console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:
    - The **Open Google Cloud console** button
    - Time remaining
    - The temporary credentials that you must use for this lab
    - Other information, if needed, to step through this lab
2. Click **Open Google Cloud console** (or right-click and select **Open Link in Incognito Window** if you are running the Chrome browser).

    The lab spins up resources, and then opens another tab that shows the **Sign in** page.

    Tip: Arrange the tabs in separate windows, side-by-side.

    Note: If you see the **Choose an account** dialog, click **Use Another Account**.
3. If necessary, copy the **Username** below and paste it into the **Sign in** dialog.

    ```
    "Username"
    ```

    You can also find the **Username** in the **Lab Details** panel.
4. Click **Next**.
5. Copy the **Password** below and paste it into the **Welcome** dialog.

    ```
    "Password"
    ```

    You can also find the **Password** in the **Lab Details** panel.
6. Click **Next**.

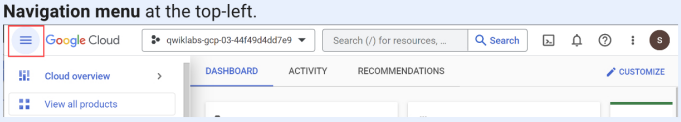
    Important: You must use the credentials the lab provides you. Do not use your Google Cloud account credentials.

    Note: Using your own Google Cloud account for this lab may incur extra charges.
7. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Google Cloud console opens in this tab.

Note: To view a menu with a list of Google Cloud products and services, click the **Navigation menu** at the top-left.



Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell** at the top of the Google Cloud console.

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `PROJECT_ID`. The output contains a line that declares the **Project_ID** for this session:

```
Your Cloud Platform project in this session is set to "PROJECT_ID"
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

2. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

3. Click **Authorize**.

Output:

```
ACTIVE: *
ACCOUNT: "ACCOUNT"

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```

4. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

Output:

```
[core]
project = "PROJECT_ID"
```

Note: For full documentation of `gcloud`, in Google Cloud, refer to the gcloud CLI overview guide.

Task 1. Define the variables needed

- In Cloud Shell run the following to set variables needed for this lab:

```
export PROJECT_ID=$(gcloud config list --format 'value(core.project
export PROJECT_NUMBER=$(gcloud projects list --filter=projectId:$PR
  --format="value(projectNumber)")
export LOCATION=REGION
export DATASET_ID=dataset1
export FHIR_STORE_ID=fhirstore1
export TOPIC=fhir-topic
export HL7_STORE_ID=hl7v2store1
```

Task 2. Enable the Healthcare API

1. In the Cloud Console, go to **Navigation menu** (≡) scroll down to the Analytics section, then choose **Healthcare**.

2. Click **Enable**.

Task 3. Create BigQuery datasets

1. Run the following in Cloud Shell to create a dataset in BigQuery:

```
bq --location=REGION mk --dataset --description HCAPI-dataset $PRO
```

You'll see a success message:

```
Dataset '{project_id}:{dataset_id}' successfully created.
```

2. Create a second dataset in BigQuery:

```
bq --location=REGION mk --dataset --description HCAPI-dataset-de-ic
```

Output:

```
Dataset '{project_id}:de_id' successfully created.
```

3. Set up the appropriate permissions to enable exporting data from the FHIR store to BigQuery:

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member=serviceAccount:service-$PROJECT_NUMBER@gcp-sa-
healthcare.iam.gserviceaccount.com \
--role=roles/bigquery.dataEditor
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member=serviceAccount:service-$PROJECT_NUMBER@gcp-sa-
healthcare.iam.gserviceaccount.com \
```

```
--role=roles/bigquery.jobUser
```

Task 4. Healthcare API setup

- Create a dataset for the healthcare API datastores to be organized under:

```
gcloud healthcare datasets create $DATASET_ID \
--location=$LOCATION
```

Task 5. Create FHIR store

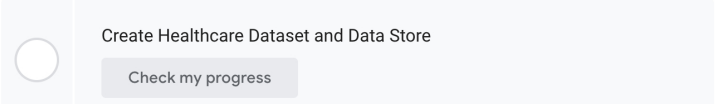
1. From the Datasets Browser screen, click the **Refresh** icon.
2. Then, click on **dataset1**.
3. Next, click **Create Data Store**.
4. Select the type: **FHIR**
5. Click in the ID field and name the data store **fhirstore1**.
6. Click **Next**.
7. Under Configure your FHIR Store, select **R4**.
8. Click **Next**.
9. Under Stream resource changes to BigQuery, make no changes and click **Next**.
10. In the Receive Cloud Pub/Sub notifications section, click **Add a cloud Pub/Sub topic > Select a Cloud Pub/Sub topic > Create a topic**.
11. Name the topic **fhir-topic**, then click **Create**.
12. Click **Create**.

Your first FHIR store is now created.

13. Create a second datastore by clicking **Create Data Store**.
14. Select **FHIR** in the Type dropdown.
15. Name the ID of the data store **de_id**.
16. Click **Next**.
17. Select **R4** for the FHIR Store Configuration option.
18. Click **Create**. Your second FHIR store is now created.

You should now see the two FHIR stores listed on the Data stores view.

Click **Check my progress** to verify the objective.



Task 6. Data creation

Import to FHIR data stores

Now you'll import sample data into the FHIR stores and stream to BigQuery.

1. Load the sample FHIR data into your FHIR store by calling the API below. In this API call, data is taken from an existing Cloud Storage bucket and loaded into a FHIR store:

```
gcloud healthcare fhir-stores import gcs $FHIR_STORE_ID \
--dataset=$DATASET_ID \
--location=$LOCATION \
--gcs-uri=gs://spls/gsp457/fhir_devdays_gcp/fhir1/* \
--content-structure=BUNDLE_PRETTY
```

This may take a couple of minutes to complete.

2. Click on the **Operations** tab to monitor the process.

The `CreateDataset` was a success and the `ImportResources1` may still be running. Wait until the operation has been completed before moving on.

3. Click on the **Data Stores** tab to view the datastores again.

FHIR bulk export

1. Using Cloud Shell, bulk export the FHIR data in **fhirstore1** to the first BigQuery dataset created:

```
gcloud healthcare fhir-stores export bq $FHIR_STORE_ID \
--dataset=$DATASET_ID \
--location=$LOCATION \
--bq-dataset=bq://$PROJECT_ID.$DATASET_ID \
--schema-type=analytics
```

This may take a couple of minutes to complete.

2. You can view progress in the **Operations** tab in the Console.

3. Click on the **Data Stores** tab to view the datastores again once the operation is complete.

FHIR data de-identification

1. Click the **Actions** button for **fhirstore1**.

2. From the dropdown, select **de-identify**.

3. Select **dataset1** as the dataset and **de_id** as the destination data store.

4. Click **Append** for the pop-up.

5. Click **Next**.

6. Click **de-identify**.

7. You can view progress in the **Operations** tab in the Console.

8. Click on the **Data Stores** tab to view the datastores again once the operation is complete.

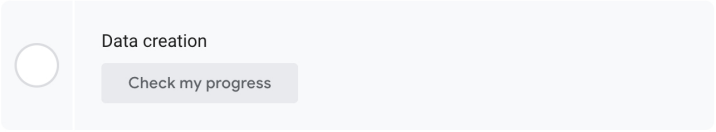
Wait for this operation to complete before moving to the next step.

FHIR bulk export

- Using Cloud Shell, bulk export the FHIR data in **de_id** to the second BigQuery data set created earlier. Before running, ensure that the previous bulk export has completed.

```
gcloud healthcare fhir-stores export bq de_id \
--dataset=$DATASET_ID \
--location=$LOCATION \
--bq-dataset=bq://$PROJECT_ID.de_id \
--schema-type=analytics
```

Click **Check my progress** to verify the objective.



Task 7. Exploring data in BigQuery

1. In the Cloud Console, use the **Navigation menu** to open **BigQuery**.

2. In the left pane, under resources, select your Project ID and expand the drop-down. You should see the two recently created datasets named `dataset1`, and `de_id`.

3. Select **dataset1** and expand the drop-down.

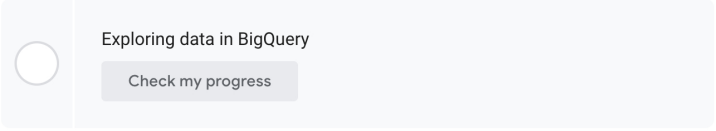
4. Navigate to the **Patient** table and preview the **Schema**.

5. Click the **+** icon to open a new Query Editor tab, then add the following SQL command to view patient data exported from the FHIR stores:

```
SELECT
  id AS patient_id,
  name[safe_offset(0)].given AS given_name,
  name[safe_offset(0)].family AS family,
  birthDate AS birth_date
FROM dataset1.Patient LIMIT 10
```

6. Then, click **Run**.

Click **Check my progress** to verify the objective.



7. In the Query window, execute the following SQL command to view de-identified patient data exported from the de-identified FHIR store:

```
SELECT
id AS patient_id,
  name[safe_offset(0)].given AS given_name,
  name[safe_offset(0)].family AS family,
  birthDate AS birth_date
FROM de_id.Patient LIMIT 10
```

See the difference in the data? In the query on the de-identified data, given_name and family name have been redacted, and the birth_date date shifted, while retaining the non-PHI PatientID.

Task 8. Streaming data export

In this section, you will create a new FHIR Patient resource in the FHIR store and export the newly created FHIR resource to BigQuery using streaming export.

1. Using BigQuery's UI, query for Darcys in the database:

```
SELECT
id AS patient_id,
  name[safe_offset(0)].given AS given_name,
  name[safe_offset(0)].family AS family,
  birthDate AS birth_date
  FROM `dataset1.Patient`
  where name[safe_offset(0)].family="Smith"
```

This will not return any results. You will now stream this patient into the dataset and query again to demonstrate the newly created resource.

2. To enable BigQuery streaming, you must update the FHIR store's streamConfigs field. To update the FHIR store, make a PATCH request with the following information:

- The parent dataset
- The FHIR store
- The BigQuery dataset
- The BigQuery project
- An update mask
- An access token

3. Run this `Patch` command in your Cloud Shell:

```
curl -X PATCH \
    -H "Authorization: Bearer $(gcloud auth application-default
print-access-token)" \
    -H "Content-Type: application/json; charset=utf-8" \
    --data "{ 'streamConfigs': [ { 'bigqueryDestination': {
'datasetUri':
'bq://$PROJECT_ID.$DATASET_ID', 'schemaConfig': { 'schemaType':
'ANALYTICS' } }
} ] }" \
"https://healthcare.googleapis.com/v1/projects/$PROJECT_ID/locatior
updateMask=streamConfigs"
```

4. Run this command to load the sample FHIR data into your FHIR store:

```
curl -X POST \
    -H "Authorization: Bearer $(gcloud auth application-default
print-access-token)" \
    -H "Content-Type: application/fhir+json; charset=utf-8" \
    --data "{
      \"name\": [
        {
```

```
        \"use\": \"official\",
        \"family\": \"Smith\",
        \"given\": [
          \"Darcy\"
        ]
      }
    ],
    \"gender\": \"female\",
    \"birthDate\": \"1970-01-01\",
    \"resourceType\": \"Patient\"
  }" \

"https://healthcare.googleapis.com/v1/projects/$PROJECT_ID/locatior
```

In this API call, you are creating a new FHIR Patient resource in the FHIR store **fhirstore1**.

5. Query for Darcys in the dataset again:

```
SELECT
  id AS patient_id,
  name[safe_offset(0)].given AS given_name,
  name[safe_offset(0)].family AS family,
  birthDate AS birth_date
FROM dataset1.Patient where name[safe_offset(0)].family='Smith'
```

You should see a new patient created, the recently imported Darcy patient! This is a result of the streaming FHIR data export of the newly created FHIR Patient Resource to the BigQuery Dataset.

Lab review

Cloud Healthcare API provides a comprehensive facility for ingesting, storing, managing, and securely exposing healthcare data in FHIR, DICOM, and HL7 v2 formats. Using Cloud Healthcare API, you can ingest and store data from electronic health records systems (EHRs), radiological information systems (RISs), and custom healthcare applications. You can then immediately make that data available to applications for analysis, machine learning prediction and inference, and consumer access.

Cloud Healthcare API enables application access to healthcare data via widely-accepted, standards-based interfaces such as FHIR STU3 and DICOMweb. These APIs allow data ingestion into modality-specific data stores, which support data retrieval, update, search and other functions using familiar standards-based interfaces.

Further, the API integrates with other capabilities in Google Cloud through two primary mechanisms:

- *Cloud Pub/Sub*, which provides near-real-time updates when data is ingested into a Cloud Healthcare API data store, and
- *Import/export APIs*, which allow you to integrate Cloud Healthcare API into both Google Cloud Storage and Google BigQuery.

Using Cloud Pub/Sub with Google Cloud Functions enables you to invoke machine learning models on healthcare data, storing the resulting predictions back in Cloud Healthcare API data store. A similar integration with Cloud Dataflow supports transformation and cleansing of healthcare data prior to use by applications.

To support healthcare research, Cloud Healthcare API offers de-identification capabilities for FHIR and DICOM. This feature allows customers to share data with researchers working on new cutting-edge diagnostics and medicines.

Congratulations!

In this lab you:

- Gained a general understanding of Cloud Healthcare API and its role in managing healthcare data.
- Learned how to create datasets and stores for FHIR data.
- Imported FHIR data from Cloud Storage
- Exported FHIR data to BigQuery in both for bulk data export and streaming
- Reviewed a number of queries against FHIR data in BigQuery

Manual Last Updated March 04, 2024

Lab Last Tested March 04, 2024