Federal Aviation Administration

Enterprise Information Management (EIM) Platform

Developers Guide

Version 15

April 30, 2021

Prepared for:

U.S. Federal Aviation Administration

Prepared by:

GENERAL DYNAMICS
Information Technology

General Dynamics Information Technology (GDIT)

3170 Fairview Park Dr., Falls Church, VA 22042

Version Control

Version Number	Created/Change By	Effective Date	Description of Change
1.0	CSRA	12/15/2017	Compiled Original Document
1.0	CSRA	1/16/2018	Added documentation, edited, modified and formatted
1.1	CSRA	2/7/2018	Edited and modified sections per client request.
1.2	CSRA	2/27/2018	Added, corrected and removed documentation
2.0	CSRA	3/16/2018	Added Zeppelin Interpreter documentation
2.0	CSRA	3/27/2018	Added RBAC and MyAccess documentation
2.0	CSRA	3/27/2018	Updated GitLab information with FAA links
3.0	CSRA	4/17/2018	Added Apache Livy information
4.0	GDIT	6/11/2018	Updated Security, EIM Platform and added Hortonworks documentation information.
4.0	GDIT	7/13/2018	Added MyAccess Knox Ranger Integration
5.0	GDIT	11/6/2018	Added updated technical documentation information.
5.1	GDIT	12/18/2018	Edited, modified and formatted document.
6.0	GDIT	3/8/2019	Added, edited and modified information.
6.1	GDIT	4/2/2019	Added the SFTP documentation
7.0	GDIT	4/18/2019	Revamp – Updated section 2.0 to improve flow of information for EIM Platform components. Remove section 3 – User Access/Data Monitoring (user access will move to RBAC section). Add Section 3.0 to discuss data on the platform. Revamp – Section 4.0 update appropriate dev cluster deployment steps and IOC Dev cluster access procedures.
7.1	GDIT	5/31/2019	Added Fortify on Demand documentation. Add, edit and formatted additional information.
7.2	GDIT	6/13/2019	Added FAMM Jump Host Configuration documentation.
8.0	GDIT	8/9/2019	Removed Collibra Data Governance Center Section. Updated MuleSoft documentation.
9.0	GDIT	11/20/2019	Update SFTP documentation
10.0	GDIT	2/10/2020	EIM Rel 1.3 updates. Feeds, Software versions,

			Mule RT provisioning, AWS CLI connectivity.
11.0	GDIT	4/02/2020	AWS Glue, Monitoring Subsystems, FAA DEVOPS
12.0	GDIT	7/27/2020	Added Cylance, VOLPE, EMR, and Privacera, updated feeds, edited and modified documentation.
12.1	GDIT	8/18/2020	Added and updated information for the new technology added
12.2	GDIT	11/03/2020	Updated documentation, edited and formatted
13.0	GDIT	11/05/2020	Added and updated SSM Parameter, Athena JDBC, Flink, EMR and API documentation
14.0	GDIT	1/29/2021	EIM version 2.2 updates
15.0	GDIT	4/30/2021	EIM version 2.3 updates

Table of Contents

1 Introduction to the EIM Platform

This Developer's Guide is designed to support the Federal Aviation Administration (FAA) developers who maintain the Enterprise Information Management (EIM) Platform system. The purpose of the Platform is to provide FAA users, departments, and programs with an effective and efficient environment to perform post-operational data analysis and provide information management support functions using FAA National Airspace System (NAS), mission support, administrative and other data. The information described in this document is related to the provisioning and development of the EIM Platform.

1.1 Background

The FAA EIM Platform is a cloud-based system that provides a platform of reusable data and information management services and big data processing capabilities for broad cross-agency use. FAA Cloud Services (FCS)[1] provides core Infrastructure-as-a-Service (IaaS) capabilities hosted on the Government Community Cloud (GCC)[2] enabling the Platform to integrate and leverage shared capabilities and common, reusable data and information management capabilities on the FAA Mission Support Network.

The EIM Platform is based on the NIST Cloud Computing Reference Architecture[3] to provide common enterprise information management functions to support modern big data environments.

1.2 Purpose

The EIM Platform supports the development, integration, and deployment of FAA IT capabilities. The EIM Platform-as-a-Service (PaaS) contains a common "unified data layer" and hosts shared, common enterprise information management capabilities, processes, products, and tools. The Platform provides robust processing and computational capabilities through the scalable FCS infrastructure. Enterprise-scale tools and services enable powerful analytic capabilities that can exploit the growing volume and variety of data that is required to support FAA systems.

Programs seeking to develop new systems or modernize existing systems can identify their required data sources, technical components, and services to determine if these artifacts are available on the platform for reuse. The platform was created with the intent of producing a collaborative forum which meet the needs for both system development as well as contribute new capabilities back to the EIM Platform.

1.3 Scope

This document articulates how developers use the EIM Platform to further enhance and develop capabilities to meet business and the FAA needs on the EIM Platform.

2 Enterprise Information Management Platform Description

The design of the EIM Platform provides a preview of the capabilities developed and planned for the EIM Platform at Initial Operational Capability. The inherent scalability and flexibility of the EIM Platform enables tools and capabilities available to users, developers and programs to advance the goals of the FAA and stakeholders.

[1] https://my.faa.gov/content/myfaa/en/tools_resources/it_services_support/work_smarter/cloud.html
[2] https://aws.amazon.com/govcloud-us/
[3] https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=909505

2.1 Enterprise Information Management

EIM is a set of core processes and principles that enable data and information management. The EIM legacy work and documents including the vision and strategy, technical, governance and communication frameworks will be leveraged to guide the platform's development and deployment efforts.

2.2 EIM Platform

The EIM Platform provides common functions to collect, access data, enable and support the processing of large amounts of data. The EIM Platform enables users with system access to the enterprise capabilities for data analysis using several different tools.

2.2.1 EIM Platform High-Level Architecture

The EIM Platform architecture is detailed below in the EIM Platform Authorization Boundary.

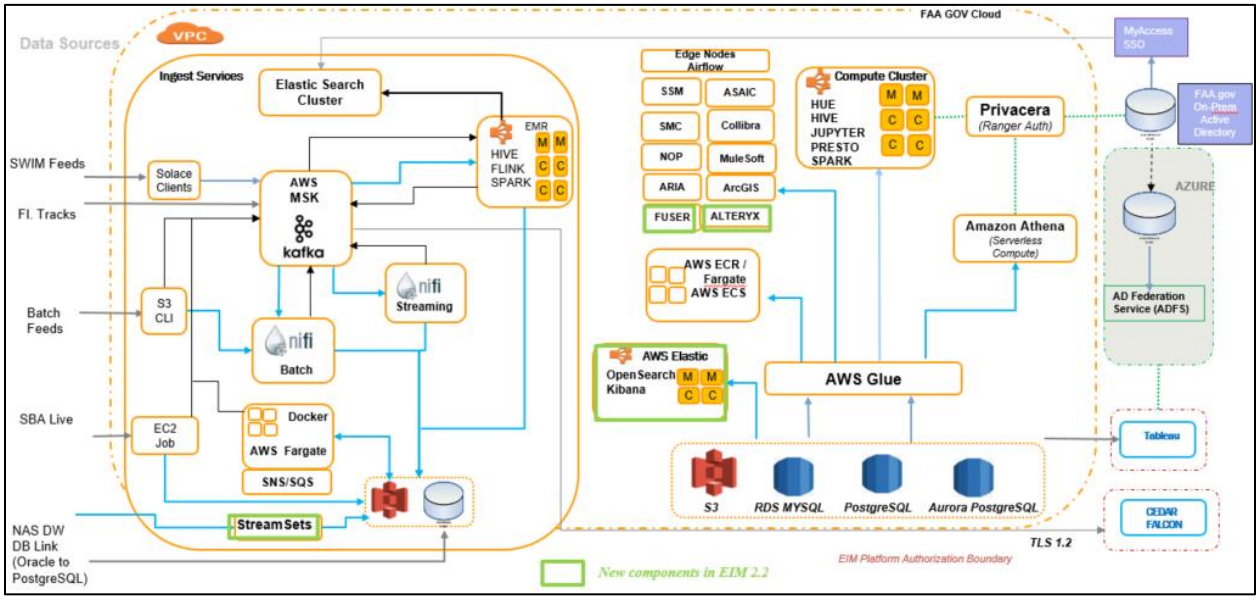


Figure 1: EIM Data Platform – Component Architecture

2.3 Components of the EIM Platform

The current EIM Release includes functional enhancements and optimizations in data ingestion, process and computation and security frameworks across the entire platform.

- Data – Additional feeds on-boarded as part of the enterprise data lake. Data feeds are also transformed to queryable serialized data formats. See the list in the data feeds section
- Process – Ingest processing has been enhanced with additional EMR clusters to use Spark based in-memory compute processes to transform data from raw to AVRO format
- Compute – Auto scalable EMR cluster has been added to compute stack with Hive, Presto, Hue, Jupyter, Spark and Presto query capability
- Security – Azure single sign-on implemented for all EIM applications. Ranger based unified security model has been adopted for AWS frameworks. More details in the following sections.

2.3.1 Data Collection

The data collection component provides reusable standard enterprise capabilities to support the collection (ingestion) of data sources required to support FAA systems, applications, and users. The data collection component supports one-time, periodic, and streaming data ingestion. Data onboarding occurs through transfer of files into Amazon Simple Storage Service (S3) buckets and via Kafka message queues and

Apache Niagara Files (NiFi) process orchestration. Additional means of transferring data into the platform from relational databases to cloud native databases and one time lift-and-shift migrations are also supported on the EIM Platform.

2.3.2 Data Storage

The storage component provides capabilities to support big data and relational structures for structured, unstructured and semi-structured data. Data storage provides access to multiple levels of performance storage ("Hot"/" Warm"/" "Cold[4]") and enables the ability to store and manage access to controlled and sensitive data. The Platform stores original (raw), transformed, and derived data sources and data and information products. Each storage tier serves specific processing needs. The platform utilizes HDFS on Elastic Block Storage (EBS) and S3. As the platform matures, data lifecycle capabilities will continue to be introduced to reflect the data will be securely stored in the most cost-effective AWS storage services which will meet performance and security requirements.

2.3.3 Data Processing

The data processing component enables the curation (cleansing and enrichment) of ingested data. The data processing capabilities are available (or can be added) to support data curation requirements that are shared as well as those that are unique to an individual system. Data processing supports both shared and program-specific algorithms and tools to use in exploiting the data. This component consists of underlying EIM Platform technologies which support and enable user facing services and tools.

EIM supports server-less computing capabilities using Lambda, Athena, distributed compute using EMR and a centralized data catalog AWS Glue.

2.3.4 Platform Services

The Platform Services component provides access to shared and program-specific tools, services, and applications that can be used to orchestrate work on the Platform and to extend the baseline collection, storage and processing capabilities. Examples of Platform Services include; elastic search, visualization tools, and data catalog. The requirements of the enterprise users are defined as individual FAA programs are continuously developed with the capabilities hosted on the EIM Platform. The EIM Platform services will be reused to support other FAA programs and users. The Platform provides shared and program-specific tools, services, and applications used to orchestrate the delivery of operational technical data within the FAA systems. The EIM Platform provides a presentation layer for users to search data sets in the platform hosted within the App mall and to launch applications, analytics, or visualization capabilities in the platform hosted app mall.

2.3.5 Security

The security component provides the technologies and capabilities to authorize appropriate levels of role based access (RBAC) to individuals and systems allowing for controlled access to different categories of controlled and sensitive data. FAA systems and applications that are hosted on the EIM Platform will derive relative security benefit, in terms of inherited security controls, for program capabilities that are fully or partially hosted on the Platform.

2.4 EIM BETA Environment

For application programs in the initial stages of on-boarding to the EIM platform, EIM provides a 'Development-like' environment with similar capabilities with continuous availability of data and

[4] Anticipate post IOC

compute resources that are always ‘on’ within the platform. This environment is called ‘EIM-BETA’ Programs can connect to the EIM BETA environment once on-boarded in their own VPC using a peering connection.

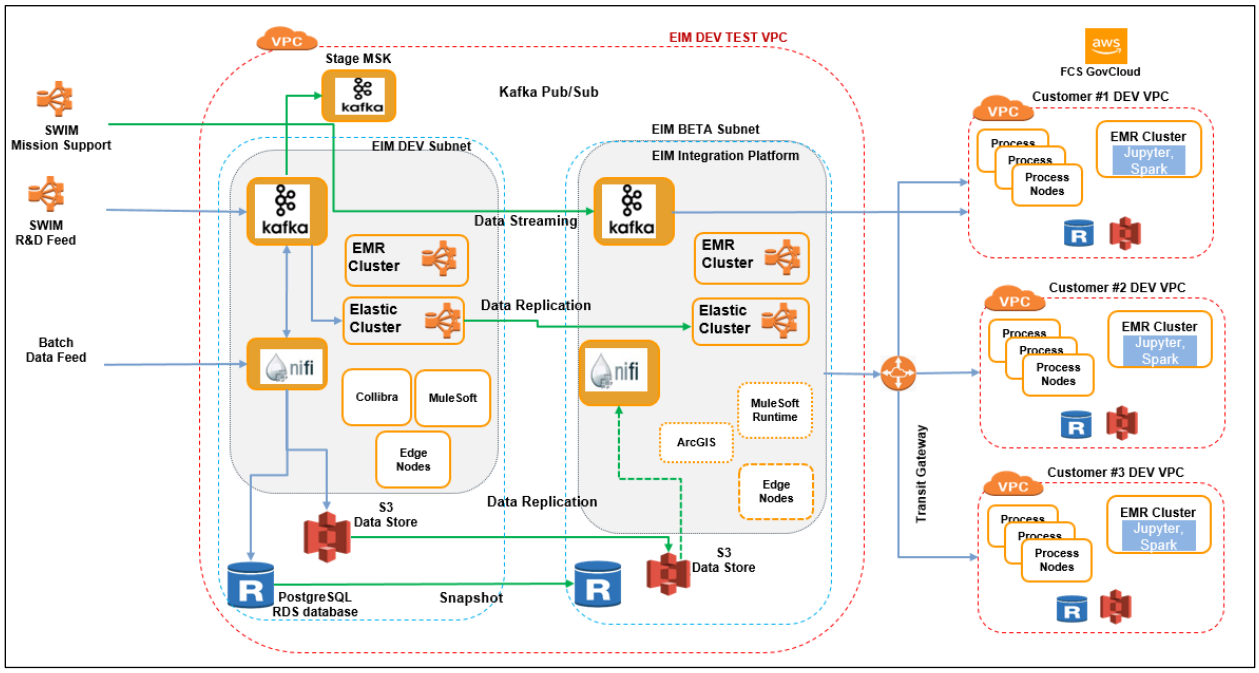


Figure 2: EIM BETA Environment

2.5 EIM Platform: Core Components

The major components of the platform include cloud native frameworks for data ingest, storage, compute, analytics, geospatial visualization, data governance and data access API framework.

AWS Managed Kafka Service for Streaming data

Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a fully managed service that enables developers to build and run applications that use open-source Apache Kafka to process streaming data.

Data Flow Orchestration using Apache NiFi

Apache NiFi supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic. NiFi provides web user interface to design, control and monitor data flows with secure and scalable configurations.

Elastic MapReduce (EMR) for Ingest processing

EIM ingest operations also includes EMR clusters for transforming raw data feeds to serialized queryable data formats. EMR clusters also used for persisting metadata for all data feeds in to Elastic search indices.

EIM platform uses two Elasticsearch clusters (1) EIM hosted Elasticsearch for metadata indices (2) AWS managed Elastic OpenSearch for log ingestion.

The metadata Elasticsearch stack is the **Elasticsearch** based distributed, RESTful search and analytics engine. **Kibana** is front end tool used to visualize Elastic Search data. Security is supported by the X-Pack extension for Elastic Stack. EIM uses hosted Elastic search to manage metadata of the data ingest and monitor the data feeds. EIM does not index content in Elastic search as of now.

For Enterprise log ingestion, EIM uses AWS managed Elasticsearch which can manage the log ingestion

from cloud based and on-prem systems.

MuleSoft provides integration platforms which connect API's, data, applications across on –premises and cloud computing environments within the EIM Platform.

Collibra delivers active data governance on the platform where data custodians define data standards, policies and usage patterns. Collibra integrates with various big data repositories such as Hive, HDFS, S3 and also RDS databases and visualization tools like Tableau. Collibra manages the metadata for every data assets ingested to the platform and provides a single pane of glass view to search and find the data availability, definitions, ownership, relationship and other information. Collibra integrates with the enterprise active directory for authorizing users on the platform. (Data Governance Center)

ArcGIS is part of the FAA EIM Data Platform tool stack and is hosted in the AWS GovCloud VPC. ArcGIS enterprise allows users to create maps, share maps, scenes, apps, and other geographic information with other people within the organization. The portal administrator will customize the portal to fit the users' organization's view and access.

The ArcGIS Enterprise portal brings together all the geographic information in the user ArcGIS platform and shares it throughout organization. ArcGIS allows:

- Create, save, and share web maps and scenes.
- Create and host web mapping apps.
- Search for GIS content within your organization.
- Create groups to share GIS information with colleagues.
- Share links to GIS apps.
- Share map and layer packages to use in ArcGIS desktop.

The FAA ArcGIS enterprise portal allows GIS accessible for users of all experience levels. Geographic viewers are designed for those who are just beginning with GIS, while experienced GIS users can connect to the portal from ArcGIS Desktop, developer APIs, and other applications.

AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy for developers to prepare and load their data for analytics. You can create and run an ETL job with a few clicks in the AWS Management Console. You simply point AWS Glue to your data stored on AWS, and AWS Glue discovers your data and stores the associated metadata (e.g. table definition and schema) in the AWS Glue Data Catalog. Once cataloged, your data is immediately searchable, query-able, and available for ETL.

The AWS Glue Data Catalog is a persistent metadata store. It is a managed service that lets you store, annotate, and share metadata in the AWS Cloud in the same way you would in an Apache Hive metastore. AWS Glue also lets you set up crawlers that can scan data in all kinds of repositories, classify it, extract schema information from it, and store the metadata automatically in the AWS Glue Data Catalog. From there it can be used to guide ETL operations.

AWS Glue ETL Operations use the metadata in the Data Catalog, AWS Glue can auto generate Scala or Spark (the Python API for Apache Spark) scripts with AWS Glue extensions that you can use and modify to perform various ETL operations. The **AWS Glue Jobs system** provides managed infrastructure to orchestrate your ETL workflow. You can create jobs in AWS Glue that automate the scripts you use to extract, transform, and transfer data to different locations. Jobs can be scheduled and chained, or they can be triggered by events such as the arrival of new data.

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is server less, so there is no infrastructure to manage, and you pay only for the queries that you run. This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets. Athena is out-of-the-box integrated with AWS Glue Data Catalog, allowing users to create a unified metadata repository across various services, crawl data sources to discover schemas and populate your Catalog with new and modified table and partition definitions, and maintain schema versioning.

EMR Compute/Presto Amazon EMR (Elastic Map Reduce) is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data. By using these frameworks and related open-source projects, such as Apache Hive and Presto, one can process data for analytics purposes and business intelligence workloads. Additionally, Amazon EMR can be used to transform and move large amounts data from and into storage in Amazon S3 and other databases.

Jupyter Hub is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and narrative text. JupyterHub provides hosting of multiple instances of a single-user Jupyter notebook server. While provisioning a cluster with JupyterHub, Amazon EMR creates a Docker container on the cluster's master node. JupyterHub, notebooks and all the components required for Jupyter notebooks run within the container.

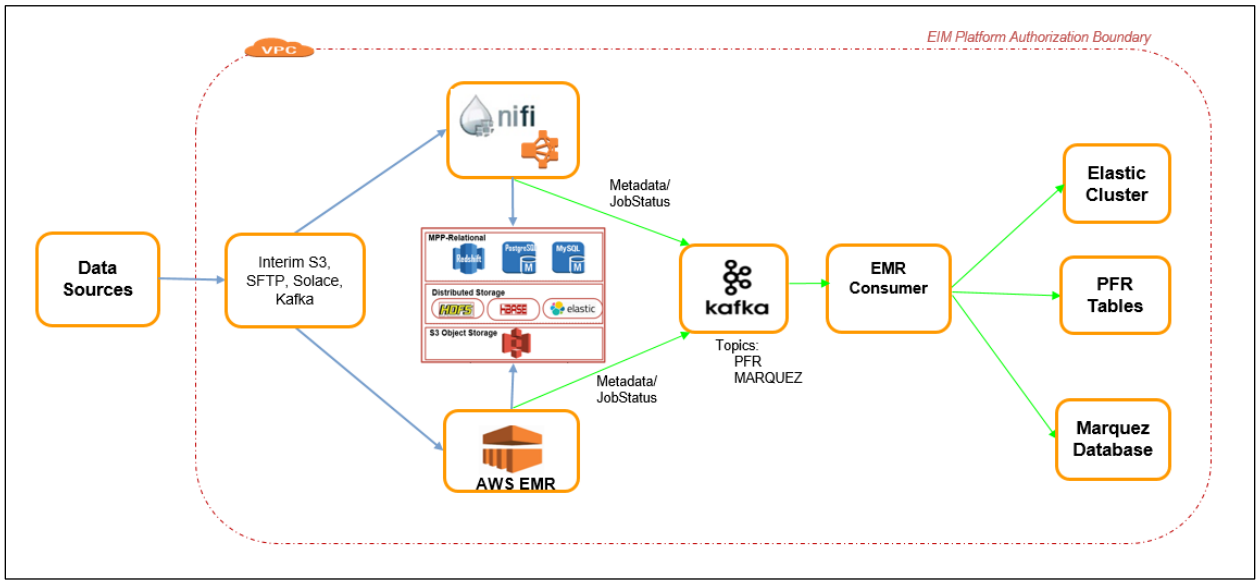


Figure 3: EIM Platform Data Ingestion Patterns

2.6 App Mall

The App Mall is the entry point for EIM applications and is integrated with the FAA Azure single sign-on using SAML. The EIM Platform AppMall is launched using:

https://apps.eim.faa.gov/appmall/

3 EIM Platform Data

EIM Platform along with its tools and the capabilities is primarily a data lake. The major activities occurring are data ingestion, storage and presentation. Refer to Appendix A for the data feeds available on the EIM platform.

3.1 EIM Data Ingest

The System Wide Information Management (SWIM) EIM Platform ensures data is treated as an enterprise asset and helps ensure all sources of data are reconciled and consistent. SWIM EIM Platform includes the management of metadata to establish and maintain links between data and information assets.

SWIM enables the EIM Platform:

- Reusable, loosely coupled interfaces versus many point-to-point interfaces
- Reduced time and complexity for building new applications and interfacing to existing applications
- Common shared services for information management replacing costly redundancies
- Enhanced enterprise-level security controls that protect the information and systems deployed to the NAS

EIM Platform has a subscribed service with the NAS/SWIM systems and ingests 8 SWIM feeds. The SWIM feeds allow the exchange of weather and flight information for the FAA. SWIM provides information for three (3) types of feeds:

- R&D (Non-restricted)
- FTI National Test Bed (FNTB)
- Operational (Restricted and non-restricted)

The SWIM Feeds provide an information-sharing platform for aviation data using operational data ingestion. Currently there are ten (10) operational SWIM feeds listed below.

Table 1: SWIM Feeds

SWIM Feed Acronym	SWIM Feed Title
STDDS TAIS	STARS Terminal Radar
STDDS SMES	ASDE-X Surface Radar
STDDS APDS	Runway Visual Range
STDDS ITWS	Terminal Weather
STDDS ISMC	STDDS Monitor and Control
TFMS Flow	Flow Traffic Management
TFMS Flight	Flight Traffic Management
SFDPS	EnRoute Radar
TBFM	Time Based Flow Management
TFDM	Terminal Flight Data Manager
TDES	Tower Departure Events

EIM Platform subscribes to the NAS/SWIM Solace endpoints and receives the ten (10) SWIM data feeds via Solace clients that are published in the EIM Platform as Kafka topics. The Kafka Topics are read by our NiFi workflows and the data is sent to S3 data stores.

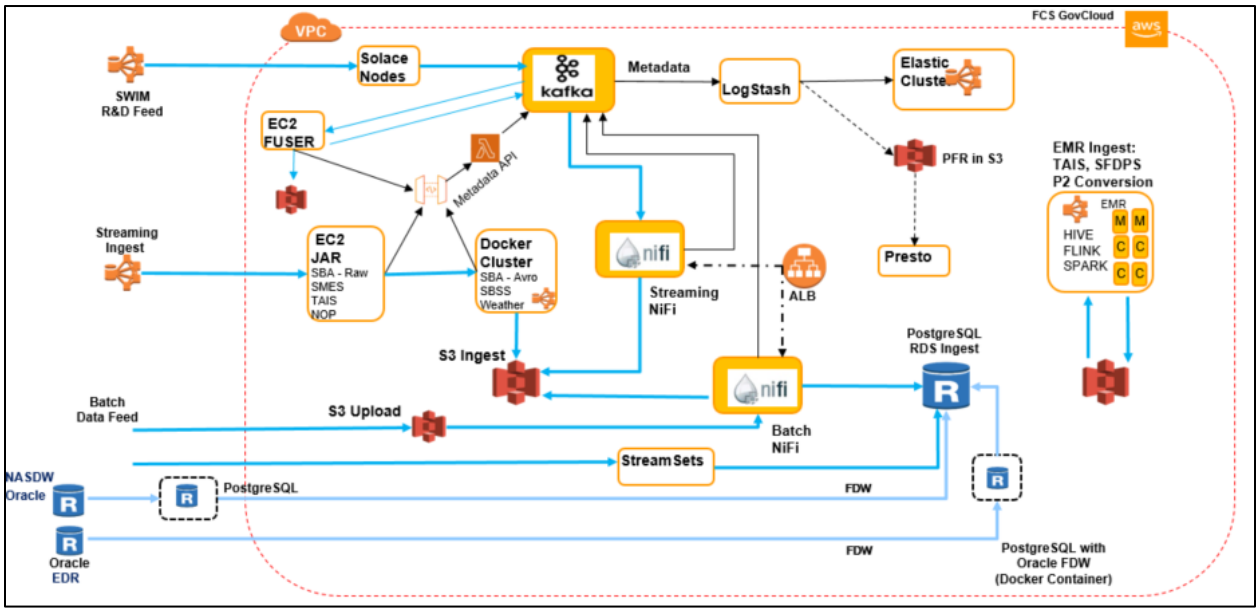


Figure 4: Data Ingestion

As shown in the above diagram, EIM data ingestion pattern uses AWS MSK Kafka for streaming data ingest. There are two separate NiFi clusters for batch and streaming data. Metadata ingestion to Elastic is streamed using Kafka and processed by EMR/Flink jobs. The serialization of raw SWIM data to avro/parquet format is performed using EMR/Spark clusters. Sensitivity of the information is tagged to the data appropriately before being stored.

There are three sensitivity levels or classifications for data:

- **Restricted Data**

Data is classified as Restricted when the unauthorized disclosure, alteration or destruction of that data could cause a significant level of risk. Examples of Restricted data include data protected by state or federal privacy regulations and data protected by confidentiality agreements. The highest level of security controls should be applied to Restricted data.

- **Unrestricted Data (FAA wide)**

Data is classified as Private when the unauthorized disclosure, alteration or destruction of that data could result in a moderate level of risk. A reasonable level of security controls is applied to Private data.

- **Public Data**

Data is classified as Public when the unauthorized disclosure, alteration or destruction of that data could result in a low level of risk. Examples of Public data include press releases, course information and research publications. While little or no controls are required to protect the confidentiality of Public data, some level of control is required to prevent unauthorized modification or destruction of Public data.

Unrestricted

- FAA Wide (No AD Group; Common Policy in Privacera/Elastic Search)

Restricted

- Restricted Filtered (RF)
- Restricted Unfiltered
  - PII Sensitive (RUPII)

  - Flight Sensitive (RUFS)
- Sensitive Unclassified Information/For Official Use Only (SUI)
- Acquisition Sensitive Restriction (ACQ)
- Licensing Restriction (LIC)
- Restricted Distribution (DIS)
- Restricted Other (OTH)

The data governance team manages the data sensitivity and appropriate access policies to the data sets thru data stewards (Data Governance Center).

3.2 Elastic Metadata Integration

The EIM Platform provides functions and services to enable processing and transformation of data into derivative forms in accordance with policies or business rules. These services include but are not limited to: data validation and quality control checks, data cleansing, standardization, and enrichment processes such as entity extraction and application of ontologies. The data preparation function is a multi-threaded activity that will allow each data source to be prepared in a variety of ways to support diverse systems and business requirements. The unique data and information output from each preparation "thread" is indexed and stored with the original data to create the virtual "unified data layer" of the EIM Platform.

3.2.1 Elasticsearch Index

The EIM Platform index is represented as data stored in Elasticsearch. Entries are created in the index via a NiFi ingestion pipeline that stores raw documents in S3, transforms the raw data to JSON, and normalizes it for querying. The Elasticsearch index allows the use of Boolean search terms, for applications, and exposes a RESTful data repository for others to use (see above). Elasticsearch provides instant response time, making it a great data store for the indexing use case.

Elasticsearch maintains a daily index that includes metadata corresponding to each data file stored in S3 buckets as part of the ingest process. The metadata also contains links (path) of the raw file stored in S3. This allows developers to rely on the data files directly using APIs. A daily index is added to record the metadata of the data stored in the EIM Platform.

3.3 Elasticsearch File Registry

Elasticsearch metadata is used as an index for storing the metadata for all the data that ingested into the EIM Platform. Whenever a feed from external source is received (SWIM, HR etc.) during the NIFI workflow for this Source the developer store the following details in the Metadata registry.

SWIM Feed => Solace Client => Kafka => NiFi => Message with lot of Sub Message Types ===> An entry is created for this in ES Metadata Index

For each Sub Message creates an entry in ES Metadata Index.

Example – TFMS flight has 18 different sub-message types, the message may contain several instances of each sub-message and each sub-message will be stored as a separate xml file. Therefore, if there are 100 sub-messages, then there will be 101 entries in the Registry for this message (1 for the parent and rest for the sub-messages)

For TFMS Flight feed, only track information sub-message type has been converted to AVRO. Therefore, that registry entry will have all three S3 location, HDFS location and Elastic index populated.

All feeds have an entry in File Registry in Elastic.

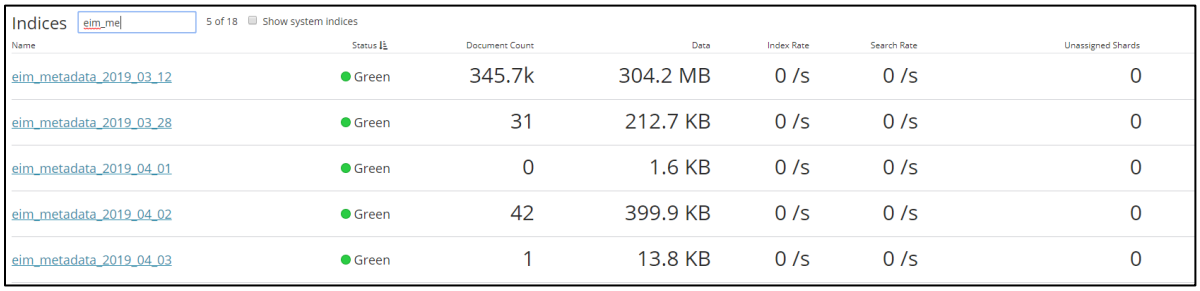
Feed content is NOT indexed in Elastic. Some data is transformed to Avro or parquet serialization formats and stored INS that can be directly queried in Athena or Presto.

Field Name	Field Type	Mandatory	Description	Example/Format
Unique Identifier	String	Y	Unique identifier of the instance of the feed	
Parent Id	String	N	Id of the Source Message. If the developer split the incoming message into multiple parts the developer need a way to link back to the Source/Original Message	
File Name	String	N	Actual File Name from the external source	
Feed Source	String	N	Actual Source Name (Ex: SWIM, SWIM, HR, FAMM)	
Feed Type	String	N	Feed Type (TFMS_FLIGHT, TFMS_FLOW, HR_PAY)	
Feed Sub Type	String	N	Feed Sub Type. Ex: TrackInformation etc.	
Feed Size	long	N	Size of the Feed in bytes	
Schema	String	N	URL of the AVRO Schema this data conforms. The URL should contain the version info as well	
S3 Location	String	N	The S3 Location where this File/Message is stored	
HDFS Location	String	N	The HDFS Location where this File/Message is stored. This location might not be valid based on the Feed Disposition rules	
Sensitive Data	Boolean	N	Flag to indicate whether this instance of the Data is sensitive or not	
Created Date	Timestamp	N	The Date when this instance of the Feed is ingested into EIM	
Expiration Date	Timestamp	N	The Date by which this data no longer exists in Hot Storage (HDFS, Elastic Search)	
Message Date	Timestamp	N	If the developer want to store the Date from the Message itself use this Column	Not being opulated
Elastic Search Index Name	String	N	Name of the ES Index where this data is Stored.	
Site	String	N	The Site Name from which the Feed came from	
Timestamp	Date Time	N	The timestamp of the Feed. Is this a Duplicate of Message Date ??	
Tim0eframe	String	N	If the developer are combing the feed for every hour or every day this field will contain the time frame.	
Optional Props	String	N	If there are any additional properties that are not mentioned	
JMS Header	String	N	Only for SWIM	
Hash Code	String	N		

The Metadata index from NiFi (eim_metadata_YYYY_mm_dd) is created daily, containing a JSON document with Metadata for every message.

Below is the mapping used for metadata index document "Metadata."

Index Patterns of the metadata indices that will be created in Elasticsearch.



Indices eim_me 5 of 18 Show system indices

Name	Status	Document Count	Data	Index Rate	Search Rate	Unassigned Shards
eim_metadata_2019_03_12	Green	345.7k	304.2 MB	0 /s	0 /s	0
eim_metadata_2019_03_28	Green	31	212.7 KB	0 /s	0 /s	0
eim_metadata_2019_04_01	Green	0	1.6 KB	0 /s	0 /s	0
eim_metadata_2019_04_02	Green	42	399.9 KB	0 /s	0 /s	0
eim_metadata_2019_04_03	Green	1	13.8 KB	0 /s	0 /s	0

Figure 5: Metadata Index

3.4 AWS Elastic OpenSearch for Log Ingestion

Amazon Elasticsearch (OpenSearch) Service is a fully managed service that makes it easy for you to deploy, secure, and run Elasticsearch cost effectively at scale. The service provides support for open source Elasticsearch APIs, managed Kibana, integration with Logstash and other AWS services, and built-in alerting and SQL querying. Amazon Elasticsearch Service provides fully managed cluster with EBS volume backed data nodes for active 'hot' data indices as well as S3 backed 'ultra warm' storage for data that is in-frequently accessed. AWS OpenSearch provides Kibana as the front end query interface and is fully integrated with FAA enterprise authentication using SAML federation.

EIM provides enterprise log ingestion capability using AWS OpenSearch platform. This function provides secure ingestion, storage and access pattern using role based access control using FAA Active Directory groups. The log ingestion can be setup for cloud based systems as well as on-prem data center systems and securely analyzed to gain insights.

For details related to onboarding and managing audit or operational logs to EIM, refer to the wiki link → EIM Log Ingestion

3.5 Storage

The EIM Platform storage architecture supports and enables storage capacity beyond what any one organization can provide. The Platform is hosted in the AWS cloud and utilizes AWS elastic block storage (EBS) volumes and S3 for its storage. Data onboarding currently occurs through a transfer of files into S3 buckets using NiFi and Kafka message queues.

Ingested sets of data are initially stored in AWS S3, where costs are low to moderate and performance is moderate to high. Ingested sets of data that will not be accessed in the near or intermediate future can be stored in S3 Infrequent Access (IA) storage. As the EIM Platform users execute analytics jobs to process the data, the users will retrieve it from S3 for processing. EIM compute framework EMR uses HDFS on EBS storage volumes for working storage, caching, buffers and all other needs for processing the current working set of data being analyzed and processed by a data processing job. The results of those jobs are transferred back to different S3 buckets or to different components of the EIM Platform (for example Presto or Athena.) making the service retrieval user-friendly. The aforementioned components use the EBS storage volumes and as a result, are only suitable for small sets of output. Larger output data sets are stored in S3.

EIM leverages tiered S3 storage modality for managing vast volumes of data in a cost efficient manger. In addition to the S3 standard storage, AWS provides S3 Infrequent access, Glacier archive as well as

Glacier deep archive which can be used for long term archival storage of the data.

3.6 Relational Database Service (RDS): MySQL, PostgreSQL and Aurora PostgreSQL

EIM offers AWS managed RDS services - MySQL and PostgreSQL - as part of the EIM baseline relational database platform. The primary database for relational data is PostgreSQL, MySQL is used only as an exception. EIM is also adopting the Aurora PostgreSQL as part of the baseline.

Developers can use the Amazon RDS management tools to create the Amazon RDS MySQL DB instance. The MySQL server provides EIM Platform mission-critical access into the broader scope of the platform's data stores.

PostgreSQL provides the EIM Platform with an open source object-relational database (ORD) system that uses and extends the SQL language and is combined with many features which safely store and scale data workloads. PostgreSQL provides features for developers to build applications, ensure data integrity, build fault-tolerant environments and assist with management of data within a large dataset.

EIM supports Aurora PostgreSQL 11 Relational database service in AWS GovCloud. These database cluster are also configured to support feeds and application instances.

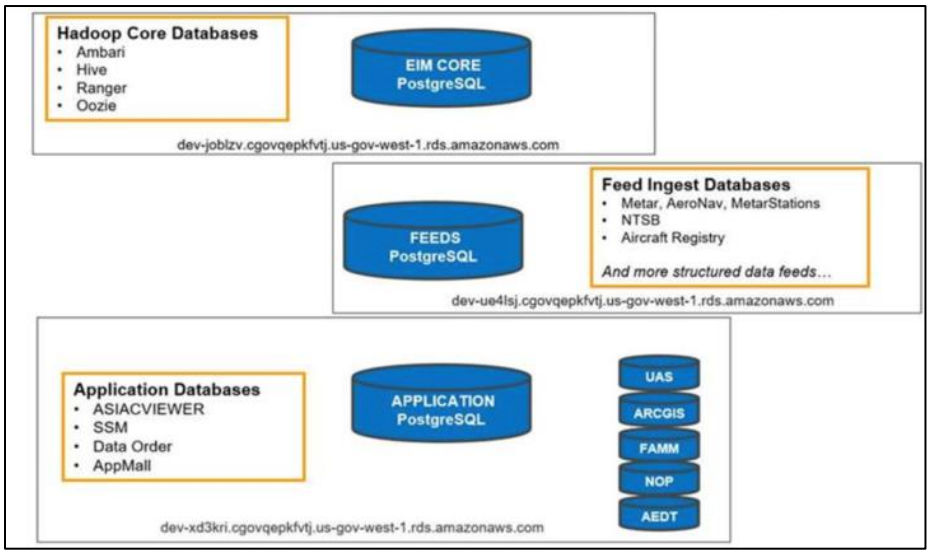


Figure 6: EIM AWS RDS Instances

3.6.1 Access to RDS databases using Presto Federation

RDS PostgreSQL and Aurora PostgreSQL databases are exposed using EMR based Presto endpoints using the RBAC access model explained in the security architecture in the next chapter.

RDS data feeds access will only allow access to either non-restricted data (*_nonrestrict) or restricted (*_restrict). Direct table access will not be given. The 'restrict' schema access will be designed based on data masking of some sort, defined by the data stewards for that feed. Access to these restrict schema's will only be granted by data stewards of that feed.

The default access level, defined as 'FAA wide' will be given to all FAA employees with ability to access Presto server. To access the RDS database feeds in EIM development environment use connection string:

jdbc:presto://ip-10-22-27-NNN.fcs.faa.gov:8446/postgresql?SSL=true&SSLKeyStorePath=<Path to jks ssl file given to you>&SSLKeyStorePassword=<Password given to you from EIM>

(**NOTE**: You will be required to supply your FAA AD user name and password to this connection.)

All RDS feeds are visible thru Hue based on the users level of access. Below figure shows all RDS feed databases - restricted and non-restricted - as an example.

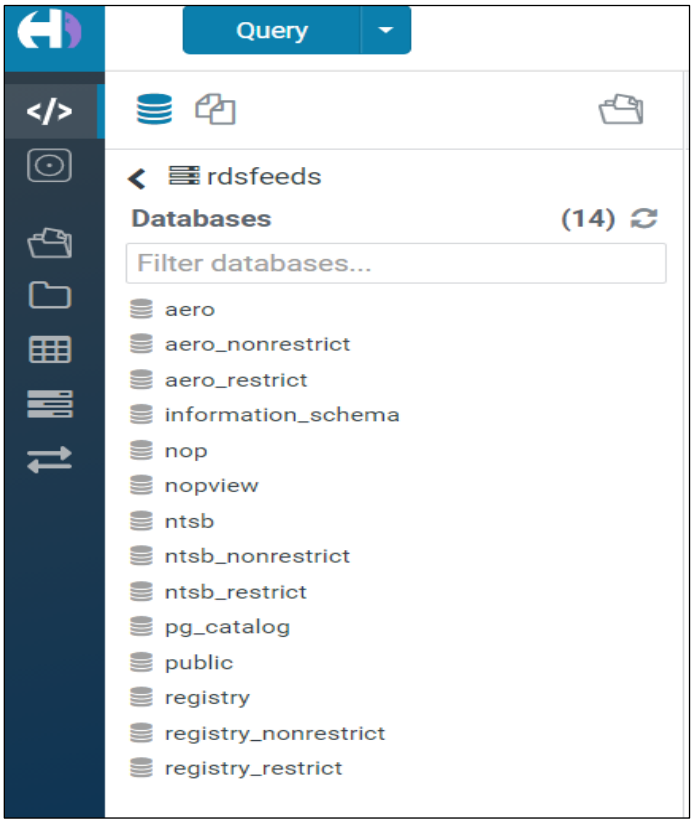


Figure 7: RDS Feeds Query

Current RDS Feeds: Aero_nonrestrict: Includes Metar data and Stations data which includes Metar stations

Registry_nonrestrict: Holds current data from: https://www.faa.gov/licenses_certificates/aircraft_certification/aircraft_registry/releasable_aircraft_download/

Ntsb_nonrestrict: Public version of database at: https://app.ntsb.gov/avdata/Access/

3.6.2 NAS Data Warehouse Data sources (NASDW)

EIM provides query based access to the NASDW data assets using 'Foreign Data Wrapper' DB link connectivity. The list of the data tables that are exposed in EIM as of the current release are shown below.

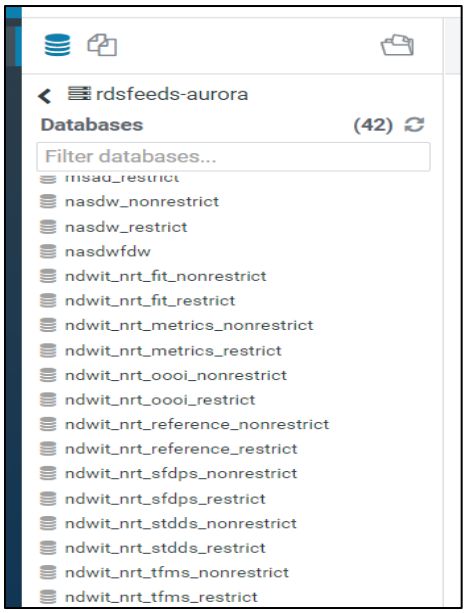


Figure 8: RDSfeeds-Aurora Data Tables

For the FDW data flow to EIM is in the Figure 4 in the 'EIM Data Ingest' section above.

NOTE: All EIM database table schema details are found in the wiki here → EIM Data Objects Schema Definitions

4 EIM Security Architecture

4.1 User Authentication to EIM Applications

EIM Data Platform is a distributed data lake platform that integrates several enterprise applications in the FAA GovCloud. These applications use the SAML based single sign-on to validate their identity and securely login to the system. The FAA EIM 2.2, SSO authentication has been integrated with the FAA Azure Active directory federation service. (The legacy SiteMinder based authentication is no longer supported).

All EIM Platform applications access is controlled by the SSO authentication and each user will have specific Role Based Access Control (RBAC) security restrictions established by the EIM Platform team during the user's on boarding process. Personal Identification Verification (PIV) credentials are required prerequisites for account establishment. These credentials are managed by FAA and users are required to have them prior to requesting access to the system for a specific role. By default, every user has a FAA issued PIV card, which provides a default access to the EIM platform.

To access to specific applications and datasets that are restricted, each user will have specific RBAC restrictions established by the EIM Platform team during the users' on boarding and access to each required application.

4.2 Developer access to EIM Servers

Role based access control (RBAC) provides security and restriction access to EIM's computer systems and network resources based on the roles of authorized individual users within the enterprise. RBAC is built into the Ansible Tower and allows administrators to delegate access to server inventories, organizations and systems. Administrators centralize the management of man credentials, allowing end

users to leverage needed information without exposing unsolicited secure information to the end user. RBAC controls allow Ansible Tower to increase security and streamline management.

Users can be members of a group, which gives them certain access to any resources associated with that role, or any resources associated with "descendant" roles. A role is essentially a collection of capabilities. Users are granted access to these capabilities and Ansible Tower's resources through the roles to which they are assigned or through roles inherited through the role hierarchy.

4.3 User Authorization

User Access to EIM data enforced by the role based access control (RBAC) mechanism defined around seurity roles and privileges within the EIM Platform. RBAC provides a fine grained system access based on Roles associated with the user/groups. Roles are created for various job functions. Permissions and policies are attached to specific roles. This section will assist the developer understand the rationale behind EIM user roles.

A users RBAC role is initiated at the data governance level. An enterprise dataset on boarded to EIM is assigned a Data Steward in the Data Governance Center (Collibra). User of a business unit must be added the specific FAA Active Directory (AD) group(s) that are added to the policy in the Ranger Authorization framework (Privacera). Ultimately, the Role is assigned to users as part of user provisioning or user on-boarding process. Any FAA user has default access to data that is not restricted on the platform. Access to **restricted** data that is FAA controlled (varying degrees of sensitivity) is provided by the data stewards (Business group/data owners) by managing the users in AD groups.

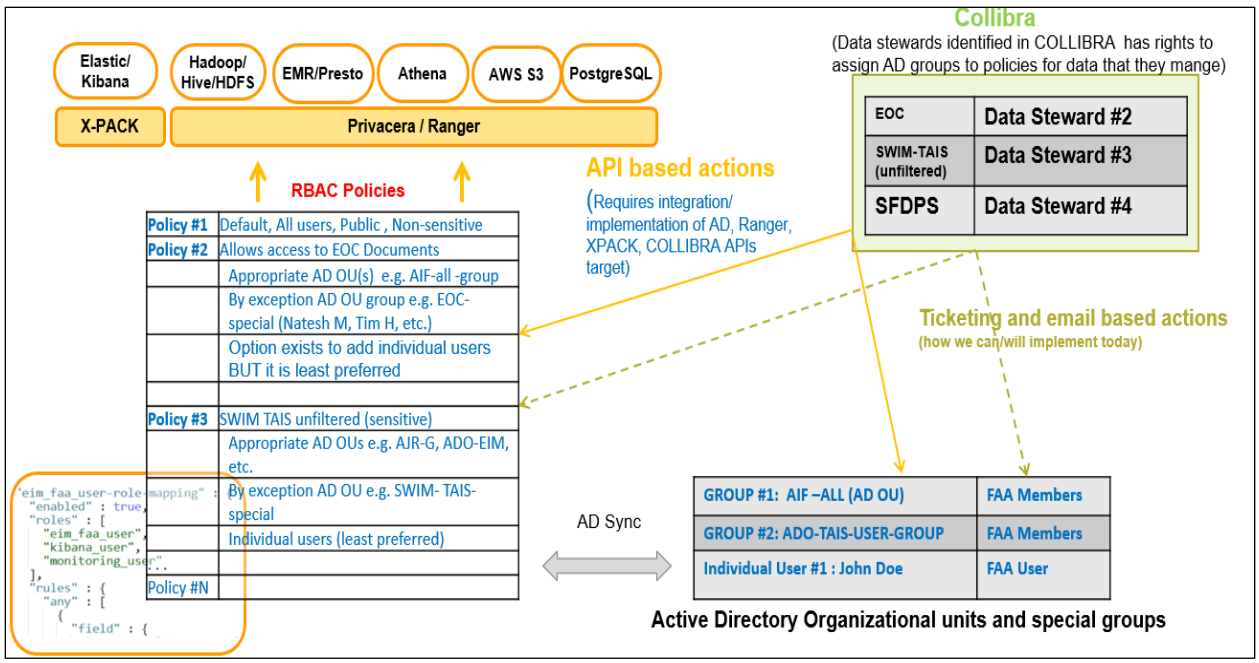


Figure 9: Access to EIM Data (End State)

EIM supports the RBAC policy management using the Ranger framework (by Privacera) for majority of the data frameworks in EIM – i.e. Hadoop/Hive, S3, Glue, Athena, EMR/Presto, and PostgreSQL RDS.

For Elastic search, EIM continues to support X-Pack role mapping framework to define the access policies. Both the RBAC policy management frameworks are fully integrated with FAA Active Directory and apply the rules to all AD groups consistently.

4.4 Ranger Policy Framework (Privacera)

- Privacera provides the Ranger policy framework to authorize users to access data persisted in various AWS repositories.
- Privacera Data Server is fully integration to the EIM platform and authenticated using Azure ADFS SAML authentication. Use identities are synchronized using FAA.GOV Active directory in Azure. EIM Administrators can configure and define the authorization policies using the Azure AD users and groups by using the Ranger policy manager.
- Privacera Data server acts as a proxy for the clients who query EIM data from S3 using Athena queries.
- Data queries sent to Athena are validated by the Privacera data server using Ranger policies defined in advance by administrators. Only the data sets that are persisted in tables/S3 buckets that have the read/write privileges to the users/groups will the returned.
- Access to data using Presto queries will be intercepted by the Ranger plug-in process running in the EMR cluster nodes

Ranger policies apply to the data while accessing:

- AWS S3
- Athena
- EMR
- Presto
- HDP
- PostgreSQL

EIM Platform uses X-Pack for Elasticsearch and Kibana RBAC access.

4.5 Privacera Components

Privacera Data Server is fully integrated to the EIM platform and authenticated using Azure ADFS SAML authentication. User identities are synchronized using FAA.GOV Active directory in Azure. EIM Administrators can configure and define the authorization policies using the Azure AD users and groups by using the Ranger policy manager.

Privacera Data server acts as a proxy for the clients who query EIM data from S3 using Athena queries. Typical use case is the client sets up Command Line interface (CLI) on an endpoint (either an EC2 instance or an on-prem server) using Privacera provided scripts and CLI tokens. After the server is configured, the use needs to enable the Privacera proxy before running any CLI commands

Data queries sent to Athena are validated by the Privacera data server using Ranger policies defined in advance by administrators. Only the data sets that are persisted in tables/S3 buckets that have the read/write privileges to the users/groups will the returned.

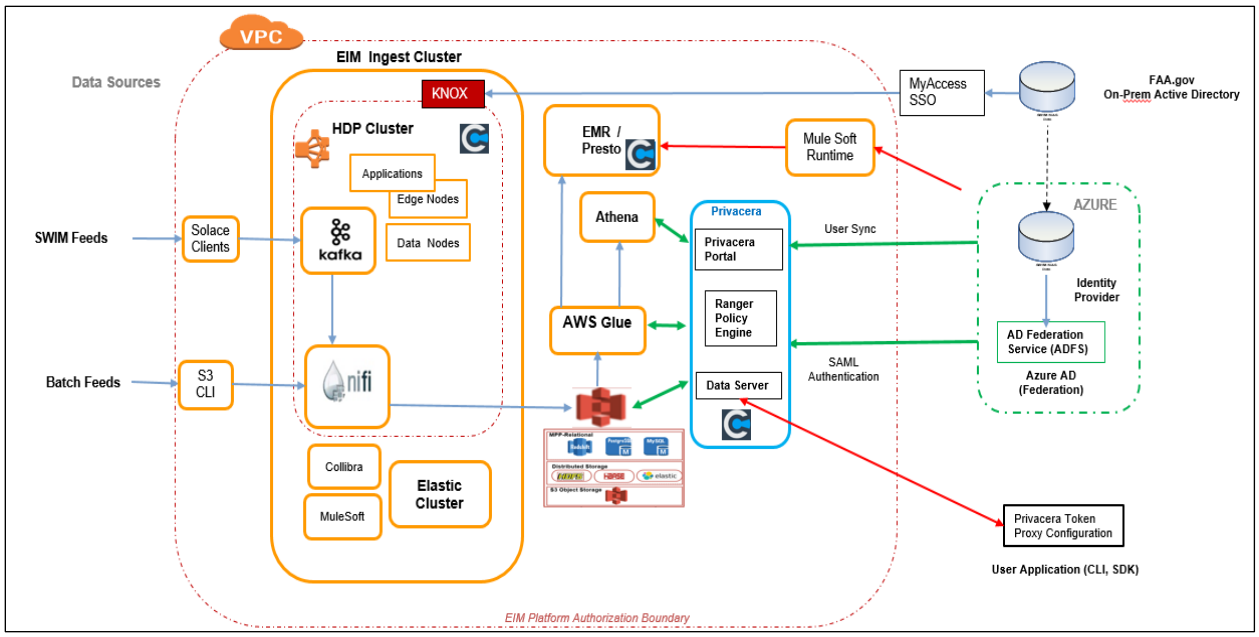


Figure 10: Privacera Integration Architecture

The COTS deployment will be comprised of the following components, provisioned and deployed in the AWS GovCloud EIM Platform Production subnet:

- Privacera Data Server, Portal and Ranger Policy Engine.
- AWS RDS PostgreSQL database (service provided by AWS, configured to support storage of rules/policies for Privacera Ranger).

4.5.1 EIM and Privacera Implementation

Privacera is a container-based solution deployed as the authorization framework on the EIM platform. It integrates with enterprise authentication standards and provides consistency across services and hybrid cloud. Privacera integrates seamlessly with Amazon S3, Amazon EMR, Amazon Athena and other AWS services and provides continuous governance, security, and privacy across the stack. Privacera access management consists of a centralized policy manager and ephemeral enforcement points that are initiated as needed. It leverages Apache Ranger to enable column, row and file-level access management and enforce centralized access policies across Amazon S3, Amazon EMR, Amazon Redshift, and other AWS services.

Privacera has a centralized access management, based on Apache Ranger, allows administrators to define data access policies from a single console and enables distributed enforcement across heterogeneous cloud and on-premises data services.

EIM Data Platform leverages Privacera Ranger framework to provide seamless authorization to the below AWS components for users who authenticate to the FAA Active Directory.

AWS S3 buckets

- Athena Query Engine
- AWS Glue
- Presto Query Engine

4.5.2 Privacera Security Architecture

Privacera Data Server is fully integrated to the EIM platform and authenticated using Azure ADFS SAML authentication. Use identities are synchronized using FAA.GOV Active directory in Azure. EIM Administrators can configure and define the authorization policies using the Azure AD users and groups by using the Ranger policy manager.

Privacera Data server acts as a proxy for the clients who query EIM data from S3 using Athena queries. Typical use case is the client sets up Command Line interface (CLI) on an endpoint (either an EC2 instance or an on-prem server) using Privacera provided scripts and CLI tokens. After the server is configured, the use needs to enable the Privacera proxy before running any CLI commands.

E.g.

```
. ~/privacera_aws.sh --config-token
. ~/privacera_aws.sh --enable-proxy
. ~/privacera_aws.sh --enable-endpoint
```

Data queries sent to Athena are validated by the Privacera data server using Ranger policies defined in advance by administrators. Only the data sets that are persisted in tables/S3 buckets that have the read/write privileges to the users/groups will the returned.

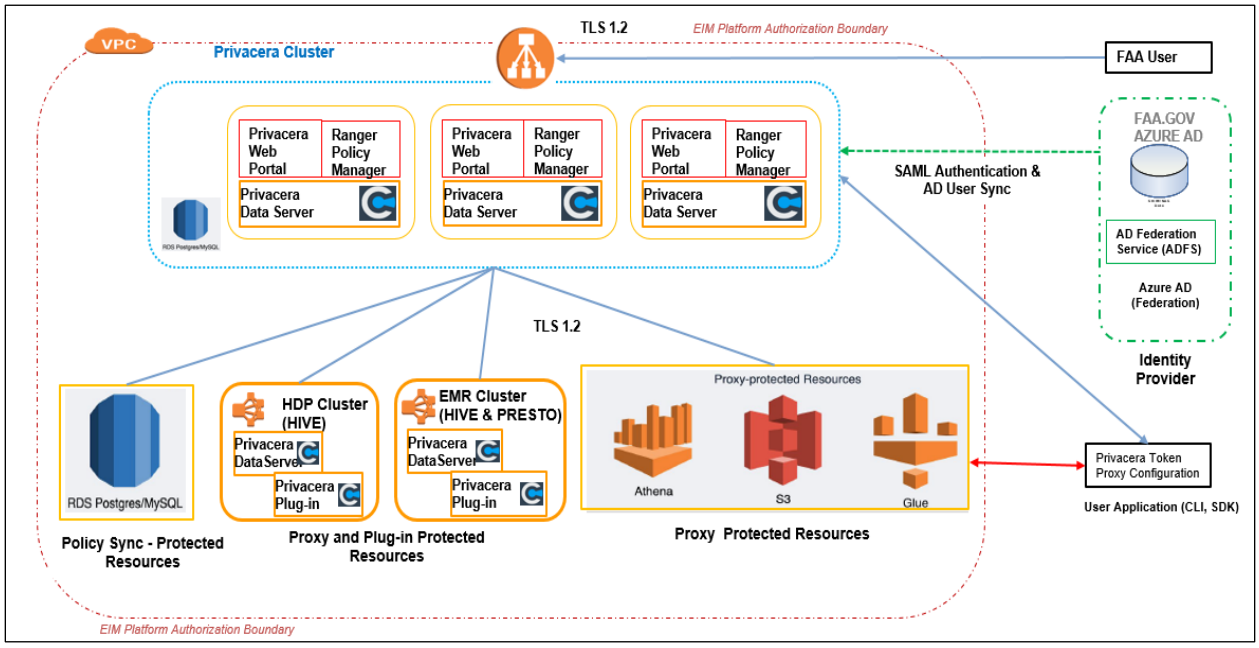


Figure 11: Privacera/Ranger Framework Architecture

The COTS deployment will be comprised of the following components, provisioned and deployed in the AWS GovCloud EIM Platform Production subnet:

- Privacera Data Server, Portal and Ranger policy engine
- AWS RDS PostgreSQL database (service provided by AWS, configured to support storage of rules/policies for Privacera Ranger)

User based request– Generate Privacera token thru API using requesting users' credentials.
API documentation for Privacera are linked below:

- Installation Guides, User Guides, Tools, Release Notes, and Community https://docs.privacera.com/privacera_cloud/audits_kafka.html?highlight=audits#dataserver-audits-to-kafka
- Installation Guides, User Guides, Tools, Release Notes, and Community https://docs.privacera.com/privacera_cloud/audits_kafka.html?highlight=audits#ranger-admin-logs-to-kafka
- Data server API --> https://docs.privacera.com/privacera_cloud/dataserver_api.html
- Privacera Token API --> https://docs.privacera.com/privacera_cloud/api.html

4.5.3 Privacera Portal Access

Privacera provides a platform to manage security and privacy challenges in their on-premise and cloud data platforms.

1. A single system for scalable data policy management across multiple cloud services.
2. Maximize visibility and assess risk of sensitive data distributed across multiple cloud services.
3. Instantly enable existing Apache Ranger compliance policies from on-prem and securely move data to the cloud.

Using File explorer from Privacera portal

1. Click on environment specific Privacera App on the Azure Access Panel:

   https://account.activedirectory.windowsazure.com/r#/applications

2. This will take you directly to Privacera portal https://privacera.eimstage.faa.gov:6868/public/index.html#/launch_pad?_k=b0l66s
3. Click on Data Inventory

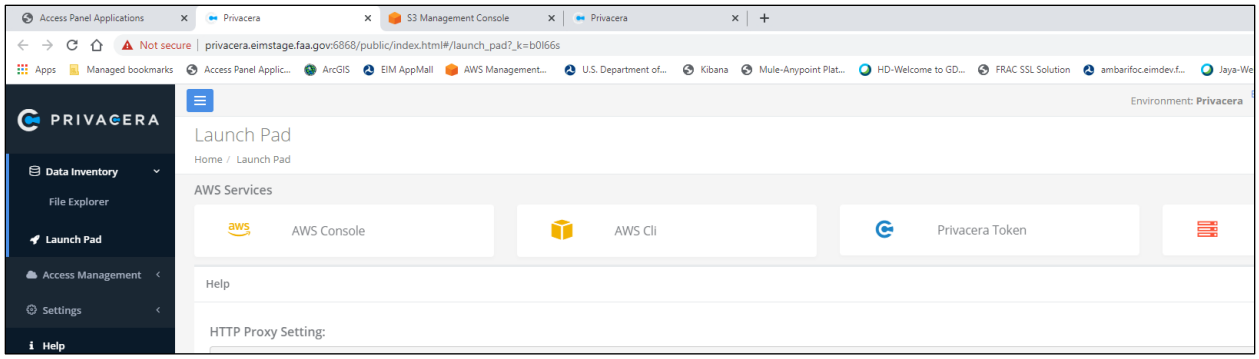


Figure 12: Privacera Data Inventory

4. Click on File Explorer
5. Navigate to the desired bucket and folders.

4.6 Ranger authorization for Querying Presto and Hive Tables

Ranger provides authorization policies to be defined at the database, table and column levels for both Presto as well as Hive query engines for all data tables/views that are organized as Hive partitions S3 bucket/folders. For Hive, Ranger provides row level filers. Access to tables/views can be provided at the metadata level, read or write. For all EIM enterprise data, users have read-only access to the data thru specific views. If the data is of restricted category, then the users have to be added to specific AD groups by the data owners to get access.

4.7 EIM Platform User Access

To access the EIM Platform user needs to be part of the FAA Active directory and would have the default access to the platform.

Table 2: User Role with Description

User Role	Description	AD Group name
Administrator	Overall Platform Administrator, mainly for EIM Ops team – This gives access to the HDP administration and Elastic Search X-Pack Management	I-ADO-APP-EIM-Admin
Business users	Access to Data assets as delegated by Data Stewards	I-ADO-APP-<source>
Default	This role gives basic access to EIM Platform – components and non-restricted data	Group Not Required

4.8 Endpoint Security using Cylance Protect

Cylance Protect is the endpoint agent for all EIM Platform hosts. It provides antivirus and malware protection. Cylance deployment - to Linux & Windows on DEV.

1. Develop Terraform Config to Deploy Infrastructure for Cylance Server (Dev)
2. Create Ansible Role and Playbook to Configure Cylance Server
3. Create Role and Playbook to Deploy Cylance Agent on RHEL7
4. Create Ansible Role and Playbook to Deploy Cylance Agent on Windows
5. Create AMI from Cylance Server OVA

5 EIM Platform Deployment and Maintenance

5.1 EIM Data Platform: Environments

EIM Platform will have four types of environments as outlined in Table 3 below. Based on the environment the developer's role and access will vary. See below for role and access details.

Table 3: EIM Platform Environments

Environment	Purpose	Lifecycle	Access
EIM DEV	Static cluster for exclusive us of EIM development. Includes Hadoop, Kafka, NiFi, Privacera, Collibra, ArcGIS – any components may be taken down during a working day without notice	Intermittently on or off based on development	Requires access to the individual server on the cluster so will be added via the steps outlined in the Runbook (add user Ansible playbook), this will be conducted by the Cluster Admin. Detailed steps outlined further in this document. For testing any EIM user access use Add to FAA AD steps outlined in the previous section
EIM BETA	Static cluster for a multi-tenant development and integration environment that is	Always on	Developers will have access using cross-account permissions to connect from their

Environment	Purpose	Lifecycle	Access
	identical to EIM DEV Used for integration of programs to consume EIM data for program development and testing purpose.		respective VPCs to EIM VPC for data sharing from S3 or Kafka streaming
EIM STAGE	For use with System Testing – functional and security testing.	For acceptance testing before go-live	Requirement to access the individual cluster server so will be added via the steps outlined in the run Book (Add User Ansible playbook), this will be conducted by Cluster Admin. Detailed steps outlined further in this document. For testing any EIM user access use Add to FAA AD steps outlined in the previous section
EIM PROD	Static Production environment for continuous ingestion and hosting production applications.	Always on	Temporary access for maintenance during a release or to debug production bugs can be requested with EIM Cluster Administrator (Add User Ansible playbook). For testing any EIM user access use Add to FAA AD steps outlined in the previous section.

For program integration, the EIM BETA environment provides stable flow of non-restricted data feeds through various data output interfaces such as Kafka, Data services layer and JDBC query interfaces. Next section shows the current architecture of EIM BETA environment and existing connections.

5.2 EIM BETA Environment

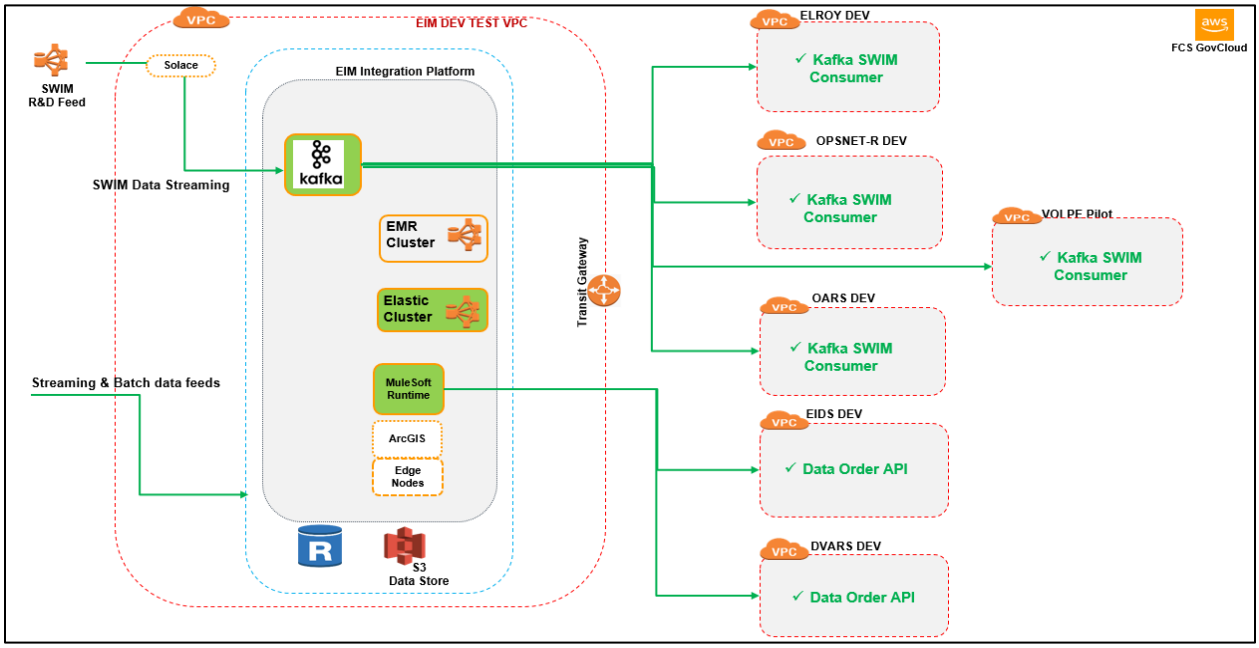


Figure 13: EIM BETA Environment

EIM BETA has a dedicated connection to SWIM R&D feeds which can be consumed using Kafka topics.

For details of connectivity to Kafka in BETA, use this reference in Wiki → EIM Beta Kafka Connectivity

All data feeds that are ingested in DEV environment are also available for query and API interfaces.

For details of access to APIs in BETA, use this reference in Wiki → EIM Data APIs

EIM BETA environment also provides data query capability for those feeds that are either stored in Relational databases or have been transformed to serializationf romatios like avro or parquet. These data sets can be queried using Prest query engine hosted in the EIM BETA EMR cluster. (**https://emr.eimbeta.faa.gov)**

To access restricted feeds, a program requires one or more 'Application Accounts' specific to the application and be authorized for system to system communication. These accounts are provisioned in FAA Active Directory and are configured in EIM security policies for specific data assets.

For details of provisioning application accounts, use this reference in Wiki → Service Accounts and Application Accounts.

5.3 CI/CD Toolchain

EIM Platform is deployed on the FCS AWS Cloud. Below is a list of tools available to FAA development operations:

Table 4: List of Tools used for the CI/CD Pipeline

Tool Name	Purpose	Description
SonarQube	For Quality Analysis	Ability to configure the Test coverage and will only proceed to deployment if it meets the required threshold
Maven	For Creating deployable artifacts	As most of the services/components the developer use Java based and will be using maven for building these components and the artifacts (JARS, NARS and etc.) produced will be stored either in S3 or Artifactory.
Bitucket	Source Code Management	Currently using FAA Bitbucket for the source code management and version control.
Artifactory / Nexus	Repository for artifact binaries	Maven will be used a good idea, NPM repo artifacts and the developer will be using either Artifactory or Nexus for this purpose.
Jenkins	For orchestration of the CI/CD Pipeline	The ability to create Jobs which will orchestrate how various components are deployed.
Ansible	For creating infrastructure	Ansible is used for provisioning the Instances in AWS and will use heavily where ever the developer can automate the tasks. Most of the time these Ansible playbooks will be invoked via Jenkins Jobs.

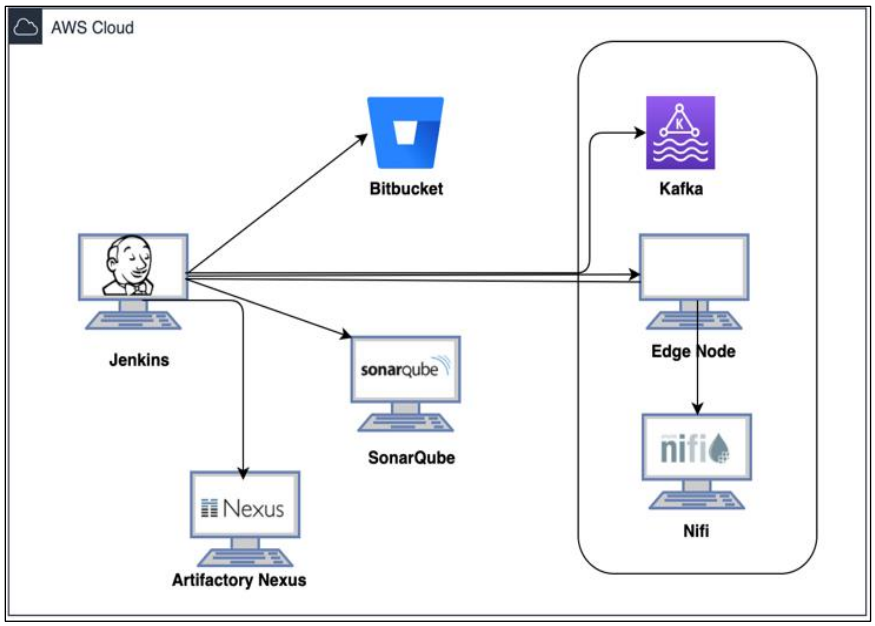


Figure 14: EIM Platform CI/CD Tools

The EIM Platform Runbook (https://git.faa.gov/projects/EIM-EC/repos/eim-developer-guide/browse) steps are used to deploy the components of the EIM Platform. The EIM Platform Runbook outlines the following steps to deploy the EIM Platform:

Foundational AWS/FCS Components (Infrastructure Provisioning)

a) STEP 1-1 (MA) – Define Infrastructure for Project
b) STEP 1-2 (AD) – Establish FCS Account
c) STEP 1-3 (AD) – Admin Right Approval
d) STEP 1-4 (MA) – Clone DEV EIM DEV Platform Repo
e) STEP 1-5 (MA) – Provision FCS/EIM Platform Resources
f) STEP 1-5 (AD) – View Inventories
g) STEP 1-6 – Obtain FCS DNS Entries
h) STEP 1-7 – Provision EIM Platform Disks
i) STEP 1-8 – Set Up User Account
j) STEP 1-9 – Port Configuration
k) STEP 1-10 – AD Security for EDG1

Foundational EIM Components (EIM Specific Technologies)

a) STEP 2-1 – Base Ansible Script Deployment
b) STEP 2-2 – Update group_vars
c) STEP 2-3 – Execute Ansible Scripts
d) STEP 2-4 – Elastic Search Active Directory Integration
e) STEP 2-5 – Install and configure MuleSoft
f) STEP 2-6 – Install and Configure Collibra
g) STEP 2-7 – Install and Configure SFTP
h) STEP 2-8 – Install and Configure APP Mall

Supplemental Hadoop Related Tasks

a) STEP 3-1 – NameNode Metadata Backup

b) STEP 3-2 – Backup Elastic Search Indices
c) STEP 3-3 – HDFS Heap Modifications
d) STEP 3-5 – Upgrading Kibana

Deploying and Configuring Data Ingest

a) STEP 4-1 – Add NiFi Directories
b) STEP 4-2 – Configure DBS
c) STEP 4-3 – Load sample data
d) STEP 4-4 – Perform sample queries

All the code required for the deployment of EIM Platform is maintained in the FAA Bitbucket repos https://git.faa.gov/projects/EIM-EC.

The types of code contained in these repo are:

1. Ansible
2. NAR files
3. JAR files
4. Property files
5. Templates

5.4 Links to FAA Bitbucket Source Codes

Below are the links to the FAA Bitbucket source codes created.

Table 5: FAA Bitbucket Source Information

Component Name	FAA Bitbucket URL	Comments
Java Service for Storing/Updating Metadata in RDS	https://git.faa.gov/EIM-EC/eim-apps/tree/master/eim-metadata-registry	Current Properties files are using MySQL connection string in local environment. If there is a need to support any RDBS, the developer need to add the corresponding Driver Jar dependency in pom.xml
NiFi Processor for Creating/Updating Metadata	https://git.faa.gov/EIM-EC/faa-eim/blob/feature/EIM-928-HIVE/eim-nifi-ingest/eim-nifi-ingest-nar/src/main/java/gov/faa/eim/nifi/metadata/UpdateMetadata.java	

Instructions and access links to use the applications in Production environment can be found in the EIM Platform User Guide.

An important step to access the EIM Platform is to be added to the FAA AD.

User Added to the FAA Active Directory (AD) Process

1. The user **MUST** have a FAA PIV or email credentials.
2. A request must be sent to the FAA help desk to add EIM Platform user groups from the FAA manger.

a. The email must include the list of the user group, a normal user may at the time have only one user group.

3. Track the FAA email for access provided.
4. Can be verified:
   a. By accessing the EIM Platform (but access will be limited based on group assigned to)
   b. or verifying in the Command prompt with the below command

Template: net user /domain "<AD Account>"

Example: net user /domain "username"

##

Template: net group /domain "<AD Group>"

Example: net group /domain "I-ADO-APP-EIM-FAA-User"

Pre-requisites for working on the EIM Platform as a developer:

1. Requires a GFE with FAA PIV.
2. Requires FAA VPN access on a GFE if working remotely. To process this create a request in https://frac.ftiharris.com/ while in FAA network.
3. GFE requires elevated privileges. Steps to process this request is included in Appendix x
4. Requires to be added to the FAA AD. Read above section for process details.
5. Requires access to the FAA BitBucket (https://git.faa.gov/projects/EIM-EC). To process this access send an email to the FAA Helpdesk.
6. Will require to access to FAA Artifactory and FAA Ansible Tower. To process this create a ticket in FAA JIRA (https://jira.faa.gov/servicedesk/customer/portal/1/create/62).
7. List of software/tools to be installed on the GFE for working with EIM Platform:
   a. Tool to SSH – example GITBASH or PUTTY (required)
   b. IDE for code development – example Eclipse (required)
   c. Code merge to repos – example Source Tree (as needed)
   d. My SQL workbench – Connect to RDS (as needed)
   e. Any other supporting libraries or plugins
8. Requires access to Dev EIM Platform cluster nodes primarily to the following:
   a. Ambari Management Node - required
   b. Edge Node - required
   c. NiFi Node(s) – required
   d. Solace Node(s) – request as needed
   e. Kafka Node(s) – request as needed
   f. Elasticsearch/Kibana – request as needed
   g. MuleSoft – request as needed
   h. Collibra – request as needed

5.5 Access the EIM Platform Cluster Nodes in Secure Shell (SSH) Mode

SSH access to the nodes will be required to perform some or all of the tasks such as code deployments, reviewing logs, updating component configurations, updating Linux configurations or permissions.

The developer will required to use Gitbash or Putty to access the nodes for CLI access. Use Gitbash or similar and enter the below command.

```
ssh <username>@<node ip address>
or
ssh -A -q <username>@<node ip address>
```

The developer will be challenged with password. Enter the AD password to access the node on CLI
SWIM Ingest Pattern

EIM Platform has a subscribed service with the NAS/SWIM systems and ingests eight (8) SWIM feeds.

Solace Clients connect to the NAS/SWIM endpoints in FAA and then obtains the data from Solace End Point and stores the data in a Kafka Topic. SWIM feeds are ingested from ATC Solace endpoints. EIM Platform Runbook contains the detailed deployment steps.

5.6 Amazon Simple Storage Service (Amazon S3)

The SWIM data is received in XML format from the NAS/SWIM source. This file is then stored in a S3 bucket in raw format.

5.7 Serialization of Raw Data

EIM uses the Avro format, to serialize and store data in big data format. Apache Avro is the primary serialization and eventually some data sets will be available in columnar format (Parquet) in S3.

Avro provides:

- Rich data structures.
- A compact, fast, binary data format.
- A container file, to store persistent data.
- Remote procedure call (RPC).
- Simple integration with dynamic languages. Code generation is not required to read or write data files nor to use or implement RPC protocols. Code generation as an optional optimization, only worth implementing for statically typed languages.

See below a sample of the AVRO schema for SWIM message type.

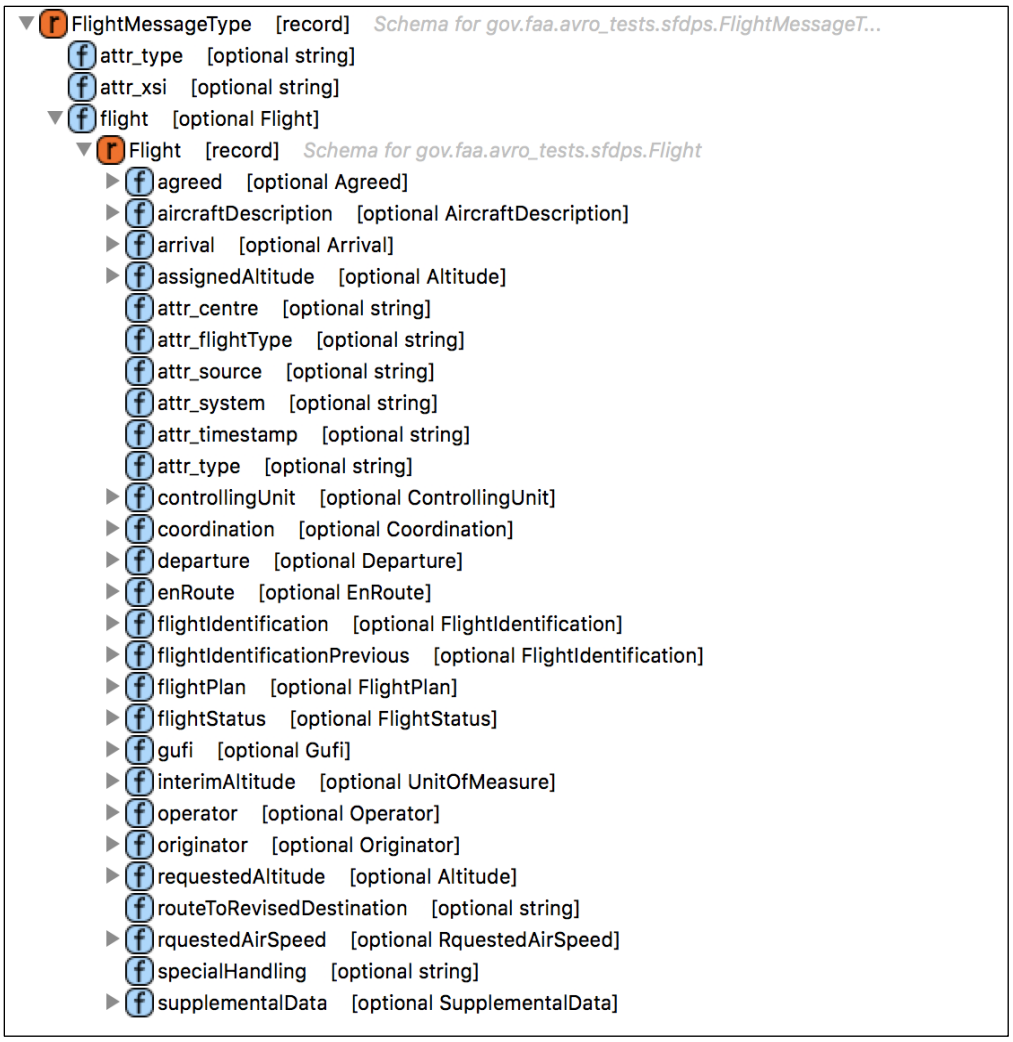


Figure 15: AVRO Schema for SWIM Message Type

By storing this SWIM (TFMS Flight) message in AVRO the data can be retrieved through the HDP component gives a SQL-like interface to query data stored in HDFS.

5.8 Application Deployments using Containers

EIM 2.2 release has introduced container based application deployment. Initial container applications use Docker based containers using AWS managed orchestration infrastructure and server less hosting environment. In the future releases, EIM plans to support Kubernetes or Open shift based container management environments. This section explains the foundational details of application deployment using containers on the EIM stack.

EIM leverages the fully managed container orchestration service called **Amazon Elastic Container Service (Amazon ECS)** to run the Docker container clusters. ECS can natively integrate with other AWS services such as Secrets Manager, Identity and Access Management (IAM), and Amazon CloudWatch providing a familiar experience to deploy and scale containers. ECS is also able to quickly integrate with other AWS services and can deploy containers on EC2 instances.

Using ECS, you can choose to run ECS clusters on **AWS Fargate**, which is a server less compute engine for containers. Fargate removes the need to provision and manage servers, lets you specify and pay for resources per application, and improves security through application isolation by design. ECS can also

deploy the containers to EC2 instances that are managed by customers.

Docker container images are managed in a fully managed container registry called **Elastic Container Registry**(ECR) which enables developers to easily store, maintain and deploy the Docker images to ECS clusters hosted using Fargate.

Below architecture shows the typical container hosting topology on AWS using ECS and Fargate.

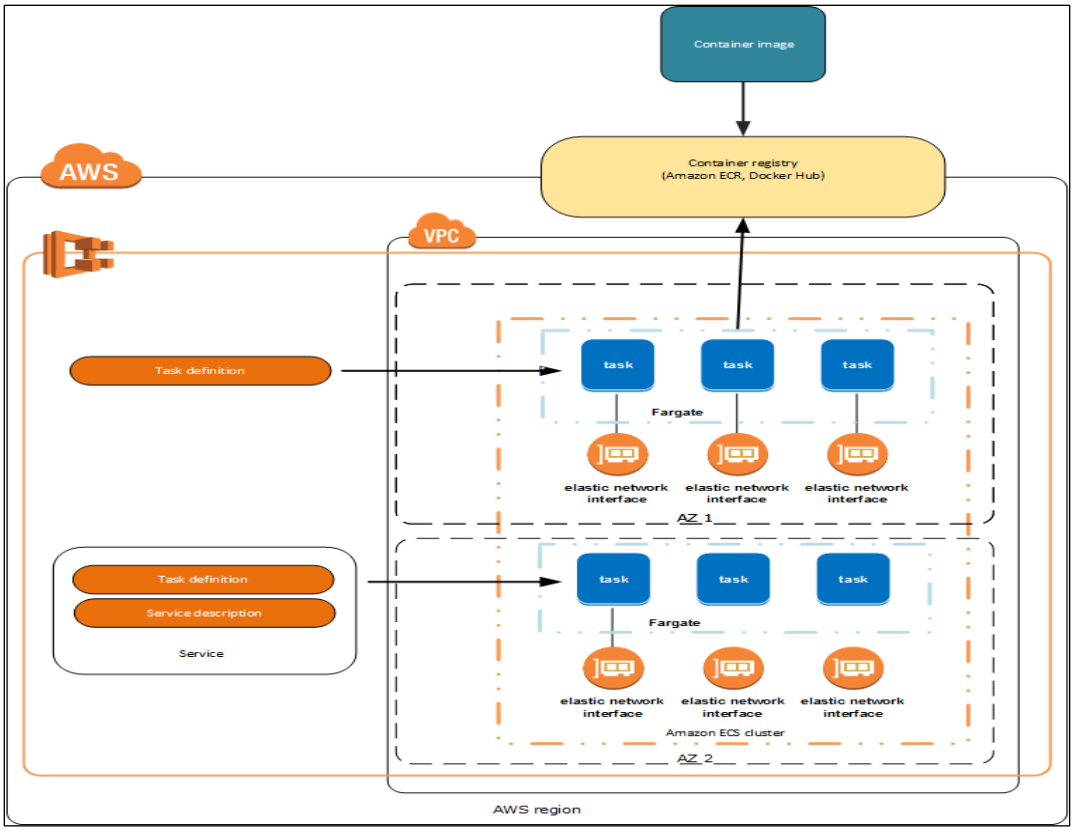


Figure 16: Container Hosting Topology on AWS using ECS and Fargate

Task Definition - Describes one or more containers, up to a maximum of ten, that form the application
Task -Instantiation of a task definition within a cluster
Service - A specified number of tasks simultaneously in an Amazon ECS cluster
Cluster - Logical grouping of tasks or services

5.8.1 DevOps Process for Deploying Containers in EIM (Applications PCPS and Marquez)

The DevOps pipeline for deploying containers is shown in the flow diagram below. The sections below illustrate the process of building the Docker container application image using Jenkins build process and Terraform based ECS cluster infrastructure provisioning process.

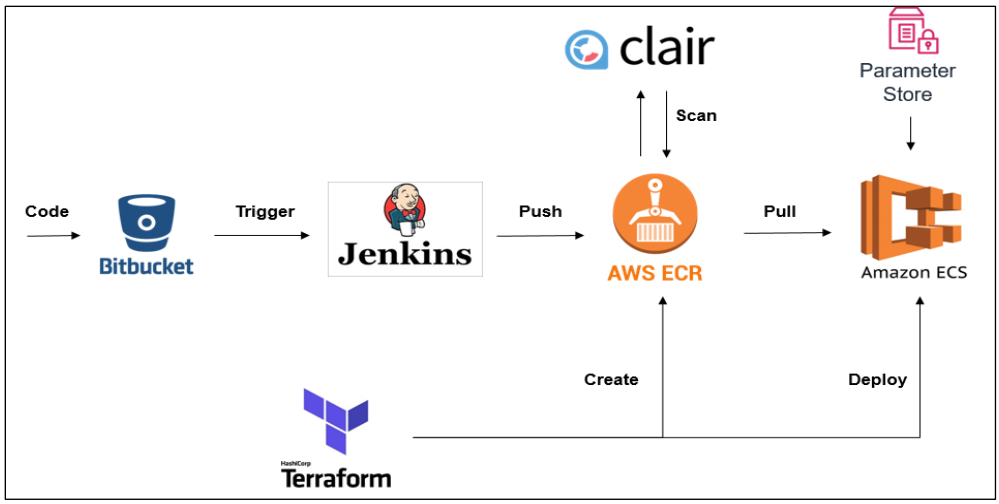


Figure 17: Jenkins Build Process and Terraform

5.8.2 Provisioning the ECR Repository

Automation for the creation of ECR is work in progress. This is currently being done manually by the EIM DevOps team using the convention eim/{env}-{application}

e.g. eim/dev-marquez-app

The terraform scripts can be viewed at this location:

https://git.faa.gov/projects/EIM-EC/repos/eim-ecr-repos-provisioning/browse

The repo contains examples to provision ECR (Marquez and PCPS). Jenkins ECR User which is used for deploying the container to the ECR repository. This includes basic automation and will be re-architected in future EIM release.

5.8.3 Provisioning an Application ECS Cluster

For purposes of this documentation the developer will use the Marquez application as an example

The source code is stored here – https://git.faa.gov/projects/EIM-EC/repos/eim-ecs-marquez-provisioning/browse?at=refs%2Fheads%2Fdev

Our terraform repos are structured to use dev, test and prod branches. (Use the dev branch for reference).

Master is generally used as the baseline initialization. Setup Terraform workspaces for each environment. In the case of marquez the developer have setup three (3) workspaces. marquez-dev, marquez-test and marquez-prod

marquez-dev	EIM-EC/eim-ecs-marquez-provisioning	a
marquez-prod	EIM-EC/eim-ecs-marquez-provisioning	a
marquez-test	EIM-EC/eim-ecs-marquez-provisioning	a

The terraform workspaces are created using Ansible (ansible.faa.gov), using the following template:

Create Terraform Workspace – **EIM**

https://ansible.faa.gov/#/templates/job_template/1753?template_search=page_size:20;order_by:name;type:workflow_job_template,job_template;search:terraform

When you launch the template you will be prompted to enter the workspace name, git repo, branch and working directory (if applicable). Ensure the right machine credentials and AWS Credentials are being used based on you AWS account.

Now that the ECR and Terraform Workspaces have been setup, let's focus on the content and structure of the terraform config files:

5.8.4 Templates

This directory will contain any application configurations generally in a JSON format or jinja (.j2) template where variables will be substituted based on the environment

Here is an extract from the template –

https://git.faa.gov/projects/EIM-EC/repos/eim-ecs-marquez-provisioning/browse/templates/eim_ecs_app.json.tpl?at=refs%2Fheads%2Fdev

```
"portMappings": [
      {
        "containerPort": ${app_port},
        "hostPort": ${app_port}
      }
```

The variables are being used – These variables need to be defined in you variables file – in this case these are defined in ecs_varaibles.tf so that provisioning time the correct value is picked. In this example values happen to be the same but will not be the case with all applications

```
variable "app_port" {
  description = "Port exposed by the docker image to redirect traffic
to"
  default = {
    dev = "5000"
    test = "5000"
    prod = "5000"
  }
}
Variables are passed to the template file in the following manner –
(from ecs.tf)

data "template_file" "eim_marquez_app" {
  template = file("./templates/eim_ecs_app.json.tpl")

  vars = {
    app_name       = var.app_name[var.faa_environment]
    app_image      = var.app_image[var.faa_environment]
    app_version    = var.app_version[var.faa_environment]
    app_port       = var.app_port[var.faa_environment]
    fargate_cpu    = var.fargate_cpu[var.faa_environment]
    fargate_memory = var.fargate_memory[var.faa_environment]
    aws_region     = var.aws_region
  }
}
```

This will ensure that the correct variables are passed based on the environment. For provisioning the infrastructure in the Terraform EIM organization these are mandatory:

000_aws_env.tf – Contains information on the provider, VPC and subnets

001_admin_security_groups.tf – This provisions the security groups and provides access to FCS security teams to access EIM infrastructure

002_ec2_ami.tf – Contains information on the AMI's used to provision EIM EC2 instances. While not used in the current example there will be other applications using a combination of EC2 instances and containers.

sg_innercomm.tf – This allows for inter communication of all infrastructure provisioned within the terraform workspace.

variables.tf – Contains all the variables associated for the workspace, project, tagging, certificates, Ansible by environment. It is preferred that you add your application specific beginning at the bottom of the file or in a separate file.

In the case of marquez the application specific variables are maintained in ecs_varaibles.tf

verisons.tf – Contains the version of terraform

Application Specific Files

To keep this simple each **tf** file provisions a piece of infrastructure required for the application.

alb.tf – Provisions the ALB

Application Load Balancer (ALB) is used for routing user requests and load balancing the container hosted applications. For smaller or low utilization applications it is recommended to use the EIM Apps ALB – app.eim.faa.gov.

Your application URL will be https://apps.eim.faa.gov/{app}/

auto_scaling_api_service.tf – Sets up the auto scaling group for the Marquez API service

ecs.tf – Provisions the ECS cluster

roles.tf – Defines the roles and IAM policies that will be used by the application

logs.tf – Sets up the AWS cloud watch monitoring

outputs.tf – Any outputs you would like Terraform to print in the log file. E.g. – ALB Name, IP addresses

sg_marquez.tf (sg_{app_name}.tf) – Security group needed for the application.

5.8.5 Naming Conventions Used in the Cluster

Cluster

The following naming convention should be used

{env}-eim-ecs-{app_name}-cluster

Variables should be passed to the template file by environment (if needed) see example above

Service

The following naming convention should be used

{env}-eim-ecs-{app_name}-service

6 EIM Compute Platform using Amazon EMR

Amazon EMR (Elastic Map Reduce) is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, in AWS FCS GovCloud to process and analyze vast amounts of data. By using these frameworks and related open-source projects, such as Presto, one can process data for analytics purposes and business intelligence workloads. Additionally, Amazon EMR can be used to transform and move large amounts data from and into storage in Amazon S3 and other databases.

EIM uses EMR clusters for two distinct functions

- Ingest EMR
- Compute EMR

Both the EMR clusters below common characteristics:

- EMR version 5.30.1
- Long-running Persistent clusters (non-ephemeral)
- Master nodes are configured for High Availability
- Auto-scalable for additional core nodes
- Clusters are enabled for data encryption in-transit between the nodes and at rest.

6.1 Services Deployed in EMR Cluster

Ingest Cluster

- Spark 2.4.5, Flink 1.10.0

Compute EMR

- Tez 0.9.2, Spark 2.4.5, Ganglia 3.7.2,
- JupyterHub 1.1.0, Zookeeper 3.4.14,
- Livy 0.7.0, Hue 4.6.0, Presto 0.232, TensorFlow 1.14.0

The EMR deployment topology for EIM platform is shown in the below in the EIM EMR architecture diagram.

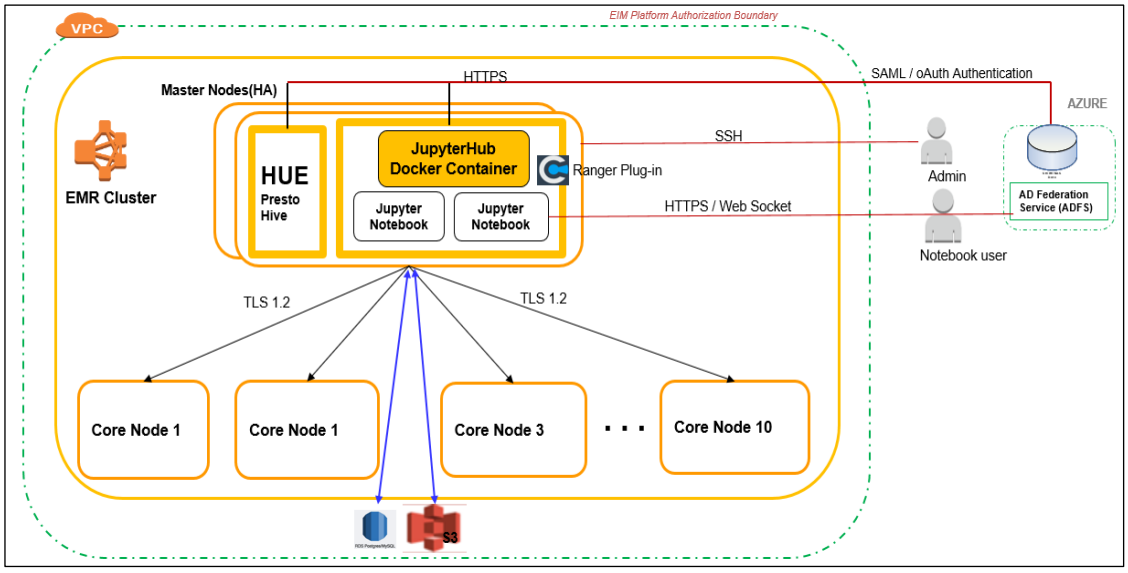


Figure 18: EMR Architecture diagram

6.2 Accessing Data using Jupyter Notebook

Each Jupyter node book runs as an independent process inside the JupyterHub container.

Jupyter deployment in the EIM Platform provides the core Python and R libraries. Additional drivers can be installed additional drivers using the Jupyter Terminal from the browser interface.

EMR URLs :

https://emr.eimbeta.faa.gov (BETA)

https://emr.eim.faa.gov (PROD – Primary EMR)

EIM Production environment has a blue green EMR topology for high availability

https://emr-bravo.eim.faa.gov (Prod – alternate EMR)

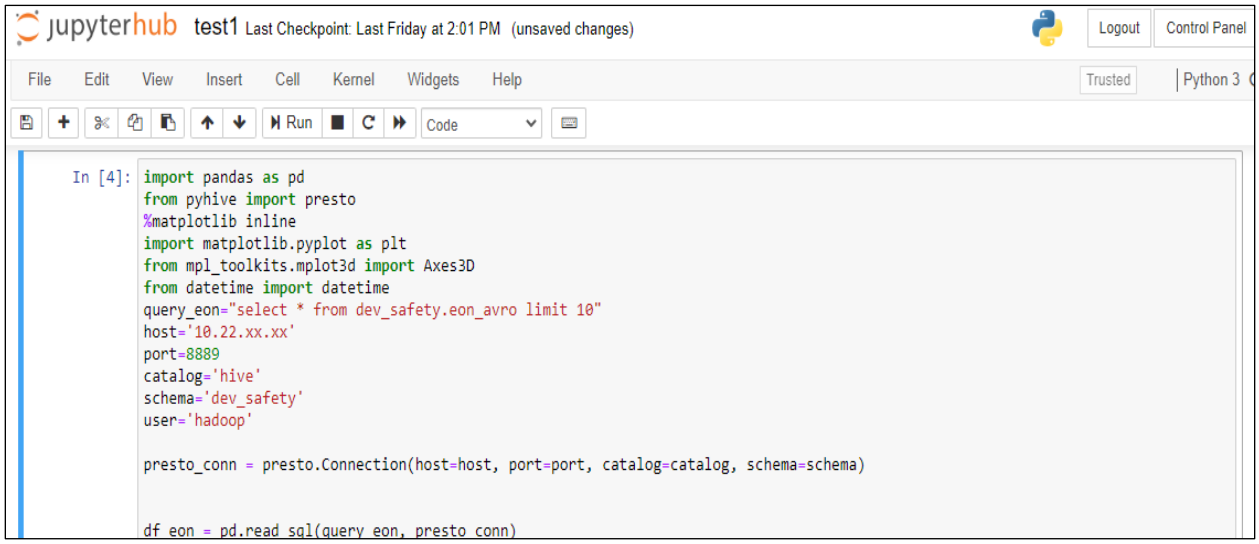


Figure 19: JupyterHub Test Sample

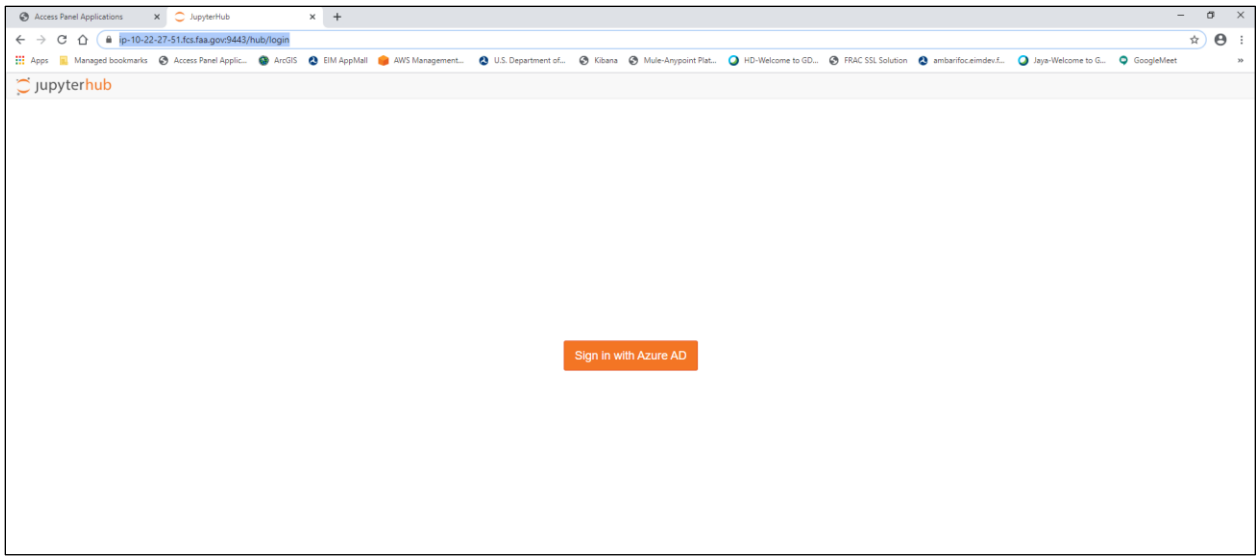
6.2.1 Jupyter Notebooks – Creating, Executing and Saving Notebooks

JupyterHub is the best way to serve Jupyter notebook for multiple users. It is a multi-user Hub that spawns, manages, and proxies multiple instances of the single-user Jupyter notebook server.

The **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Login to JupyterHub (Access through the AppMall)

1. **Click** on Sign in with Azure AD



2. **Sign in with Azure AD** will take the developer to their notebook server.

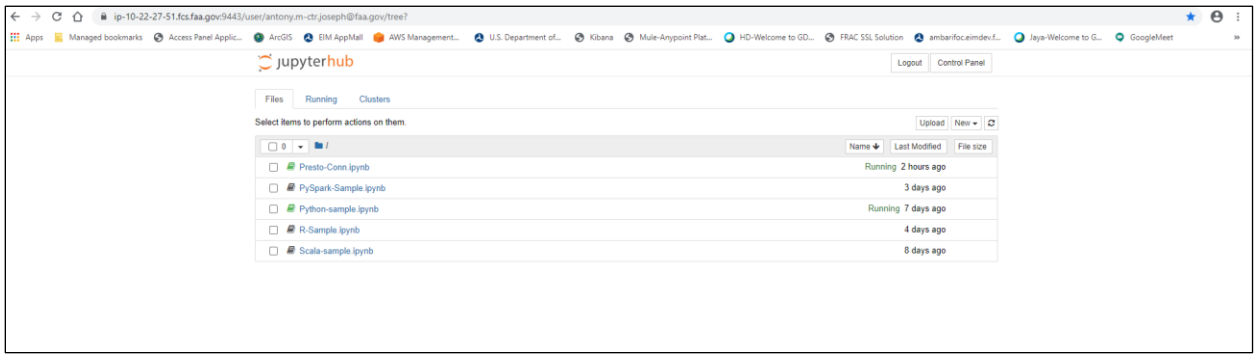


Figure 20: Jupyter Notebook

3. The files are AutoSaved at eimspare2-emr-storage Bucket.
4. The developer can choose the kernel to create a new notebook.

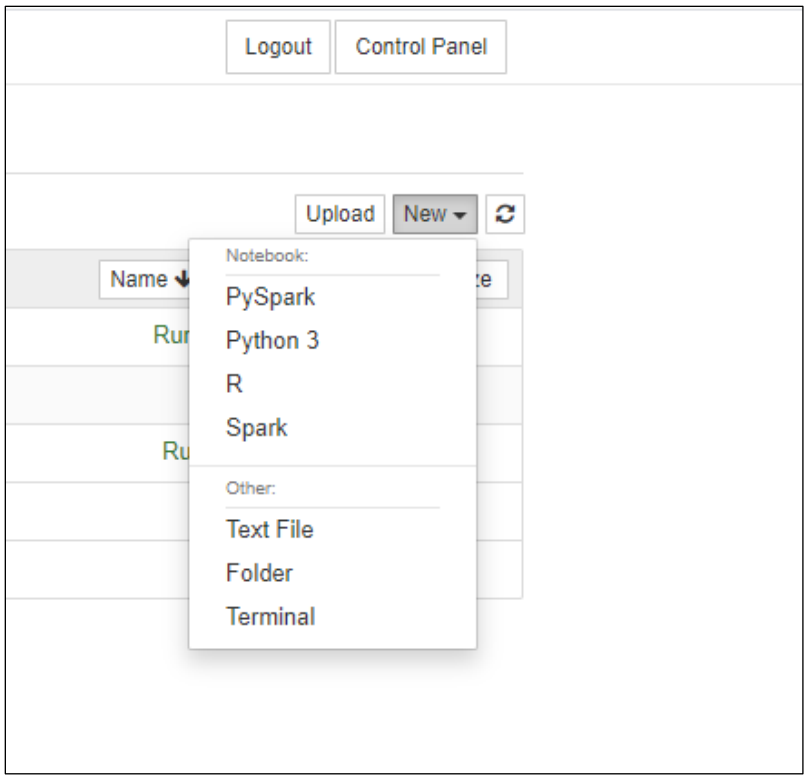


Figure 21: Notebook Selection Options

5. Sample Presto Connector using Python.

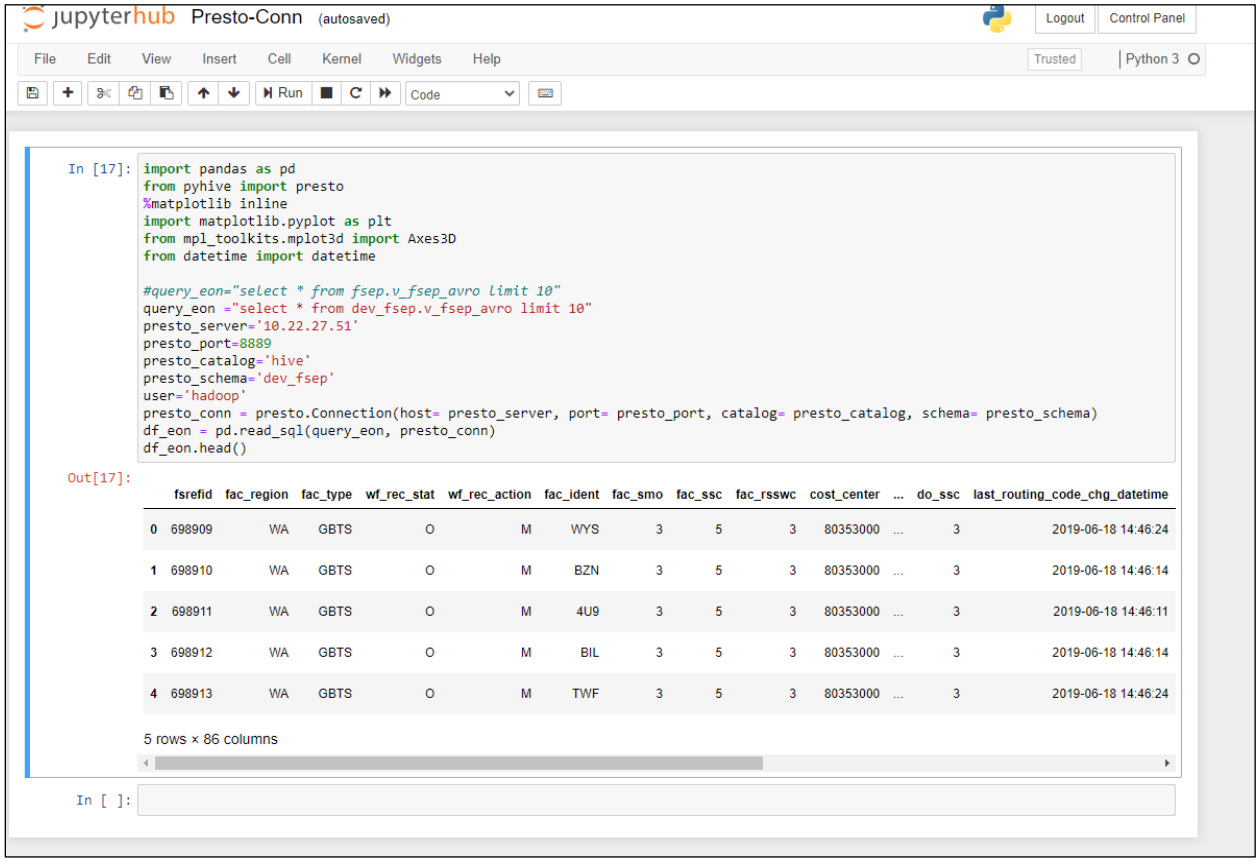


Figure 22: Jupyter Presto Conn

6. Sample PySpark notebook

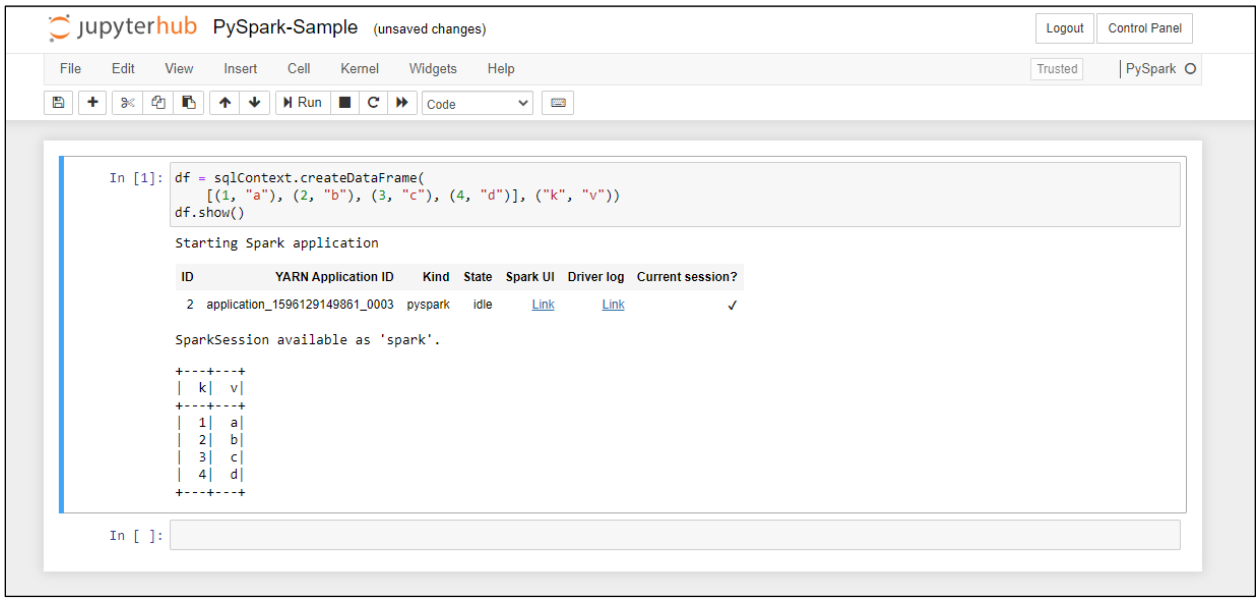


Figure 23: JupyterHub PySpark-Sample

7. Sample R notebook

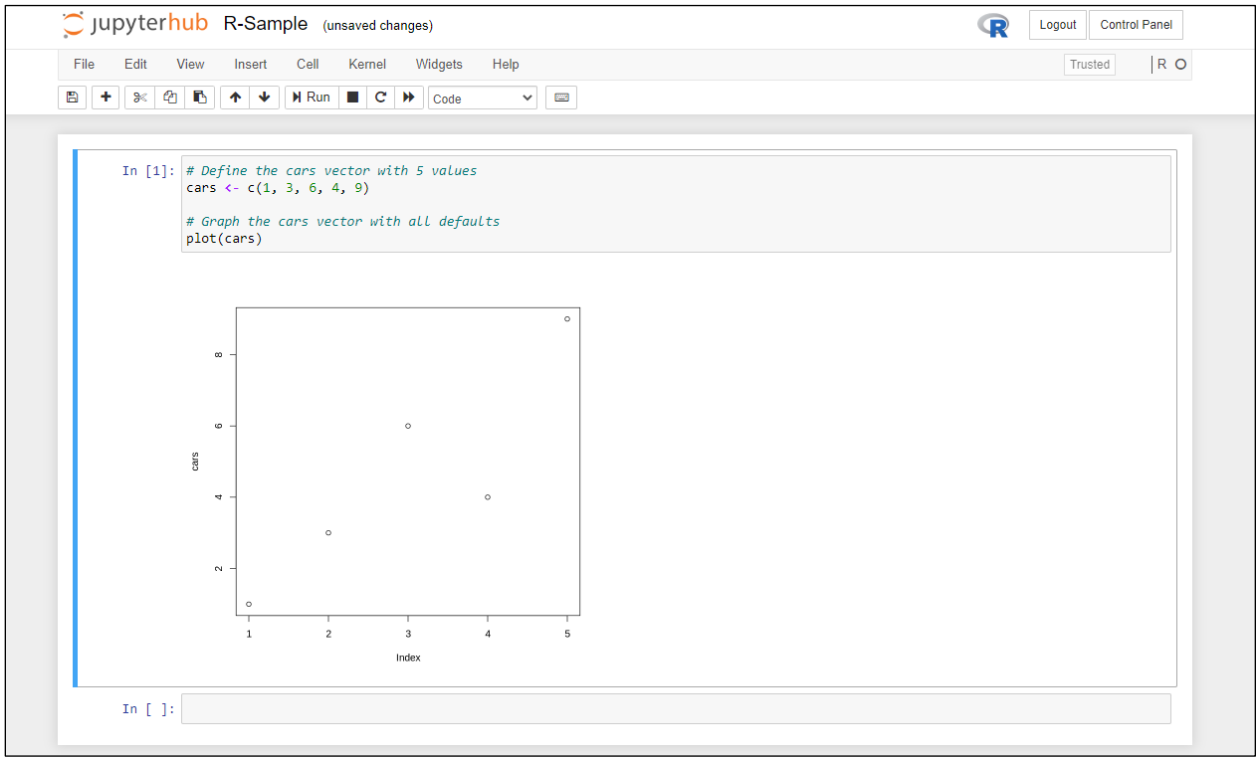


Figure 24: JupyterHub R-Sample

8. Sample Scala notebook

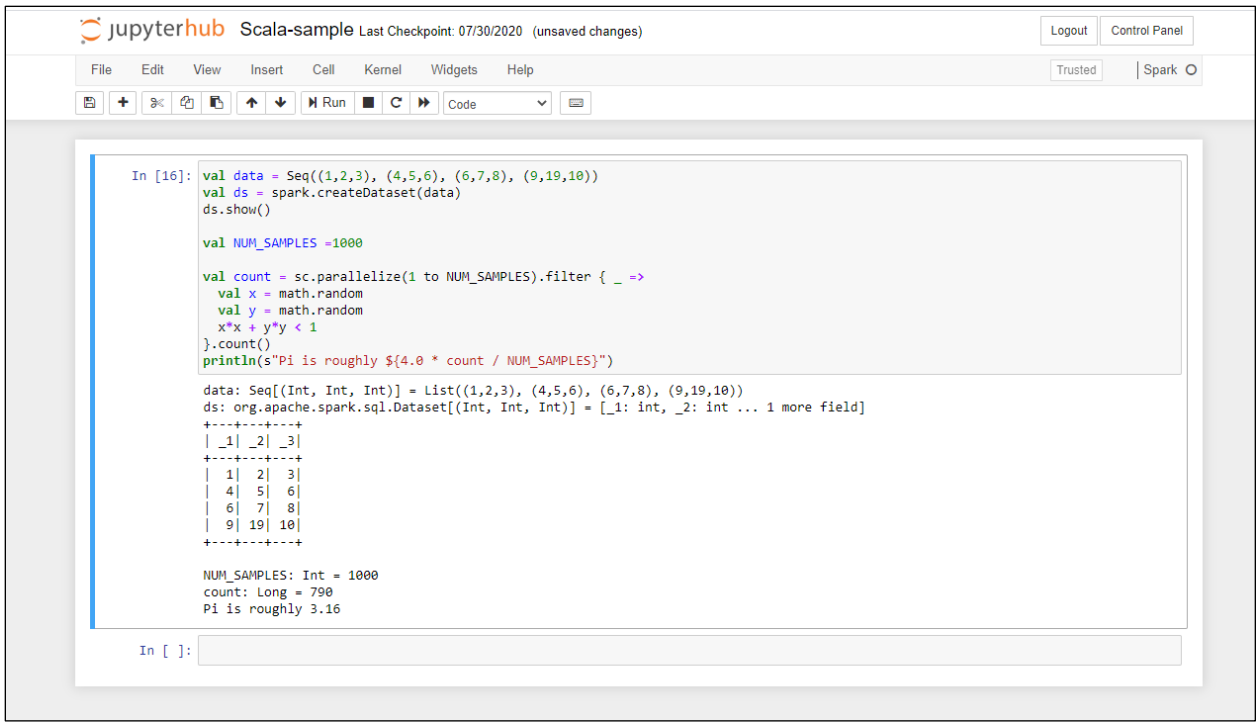
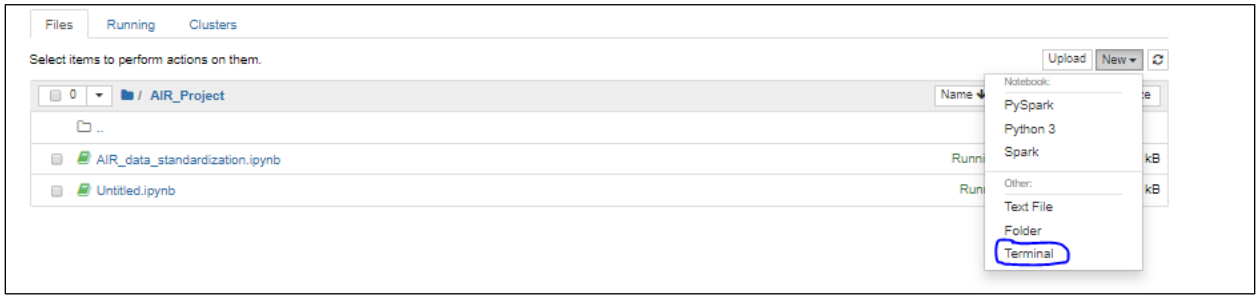
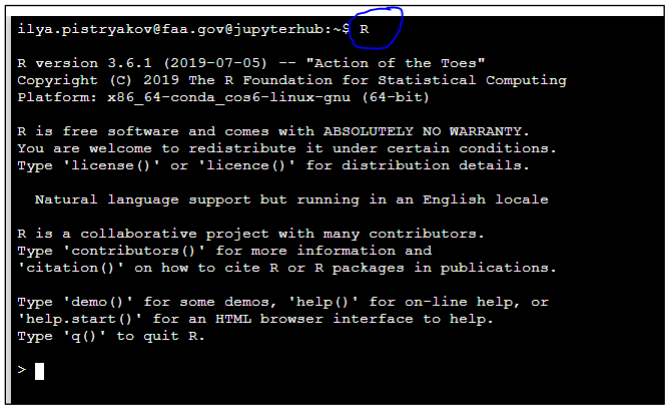


Figure 25: JupyterHub Scala-Sample

6.2.2 Using a Terminal to Access Backend



- To install a package for **R:**
- In the above terminal, enter R prompt by typing “R”.



A number of libraries have been deployed on EMR. For a comprehensive list use this link:

Jupyter Libraries

6.2.3 Hue for Running SQL queries

HUE (Hadoop User Experience) is a Web interface for analyzing data with **Apache** Hadoop and other query engines that are deployed on the EMR cluster.

Users can query datasets using Presto query engines using Hue interface. Hue interface also provides query capability for the PostgreSQL RDS databases.

The EMR cluster is integrated with Privacera Ranger policies and enforces RBAC control of the data assets in Presto. When the users use single sign-on and login to Hue using their FAA provided user identification credentials, Ranger policies intercept the user queries and ensure the user has the access to the requested data source.

1. Login to Hue (From the AppMall) (see below how Presto and RDS query can be executed in Hue)

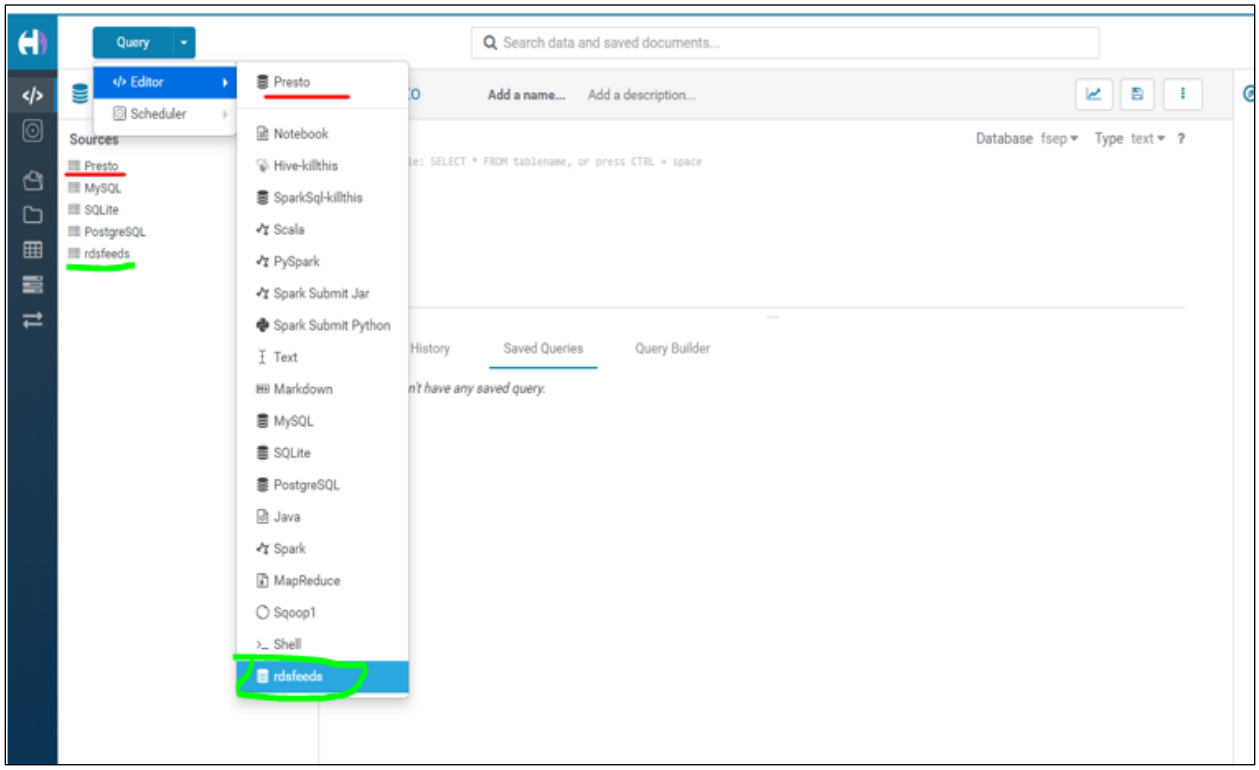
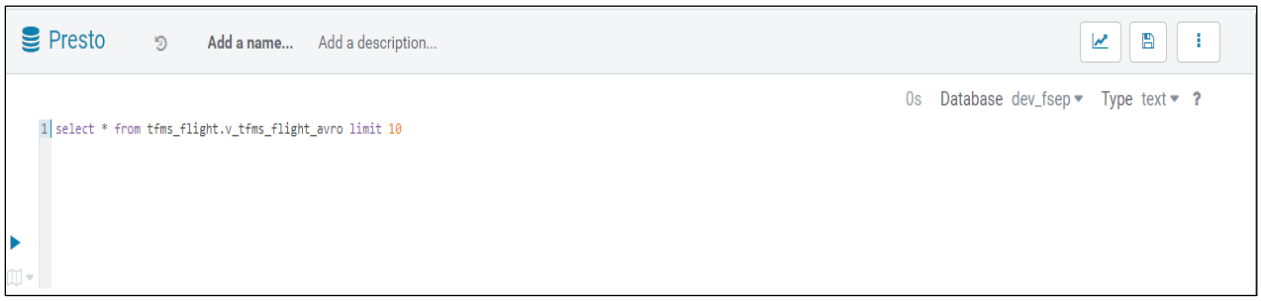


Figure 26: Hue Query

2. Click **on Query/Editor** and select your SQL engine
3. Run queries in the SQL editor.



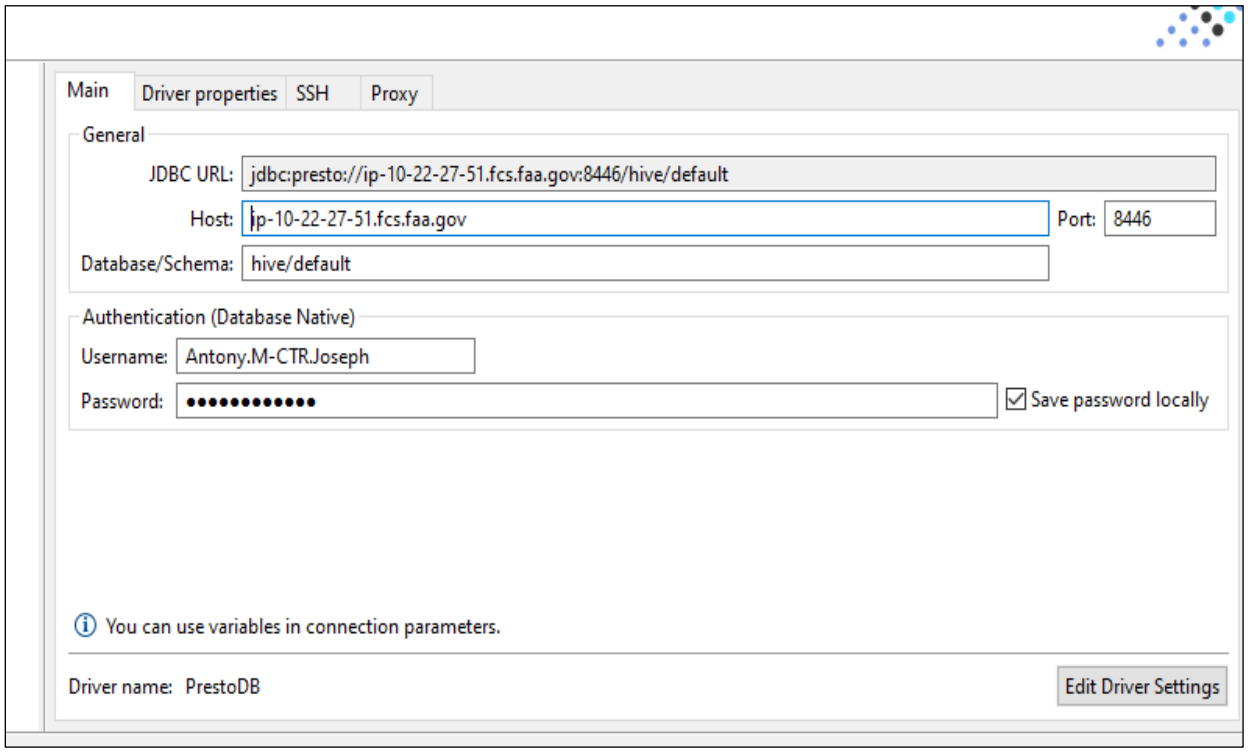
Access to the Glue catalog tables using Presto queries is controlled using the Privacera/Ranger policies. Users can retrieve data only if they are part of the AD group that has been defined in the policy for specific data asset. For all non-restricted data, there is no need to be part of a specific AD group.

This concept has been explained in detail in the chapter titled EIM Security Architecture.

6.3 EIM Data Query using Presto Query Engine

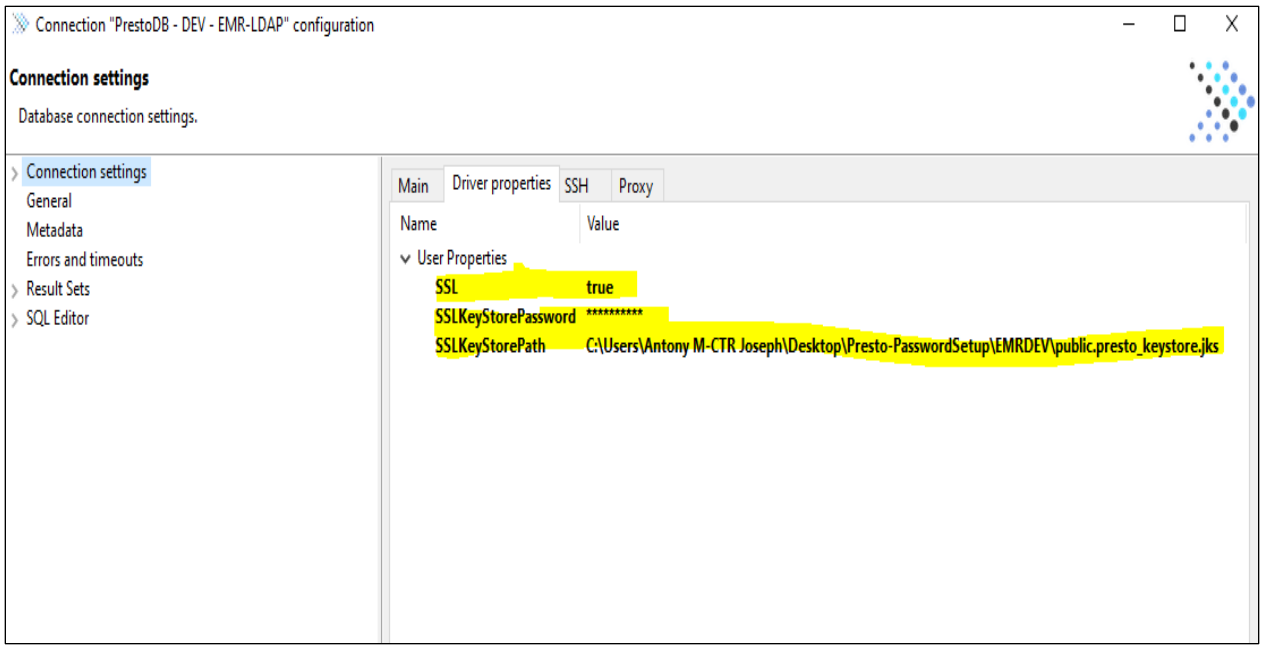
Presto is a distributed database query engine deployed on the AWS EMR cluster and configured to access the EIM databases using AWS Glue catalog. Presto federates connections to the RDS PostgreSQL and Aurora PostgreSQL databases. Presto databases can be queries using Hue user interface or using JDBC connections. Presto is fully integrated to authenticate with FAA Active Directory using LDAP protocol and data authorization using Ranger framework. Below is an illustration of Presto connection using desktop SQL tool DBeaver.

Step 1 -Click on Connection Settings and provide Hostname, port, user and password.



Step 2 -Obtain Presto EMR SSL keystore JKS file (Must be public key file) and keystore password from EIM Admin.

Step 3 -Add the driver properties to enable SSL connectivity using public key, below.



For examples, using public JKS certificate in JDBC connections:

https://wiki.faa.gov/display/EIMSD/EMR+Presto+authentication+using+certificates

Presto Performance Optimization - For more detailed information, refer to the **appendix.**

6.4 EIM and Tableau Integration

EIM data sources that are configured for SQL queries are exposed using the Presto query engine. All databases/tables/schemas from the Glue catalog as well as RDS databases are available to be configured as data sources in Tableau. Tableau integration with EIM can be setup using Tableau Desktop and published to the FAA enterprise Tableau servers (test and production).

Here is the link for the Presto site on Tableau Test sever.

https://awstest-tableau.faa.gov/#/site/_Presto/home

Tableau access is configured to use the single sign-on authentication. Presto has been integrated with FAA Active Directory for authentication and data queries are authorized using Ranger policies. Hence, the data retrieved in Tableau is security trimmed based on the user privileges setup in Ranger. FAA users with privilege to create and publish data sources to a Tableau server will use the below configurations in Tableau desktop software to connect to EIM data source(s).

Tableau Roles:

a. Creator – Create and publish new data sources
   Create and publish new workbooks, dashboards and visualizations
b. Explorer – Author and edit existing dashboards and visualizations
c. Viewer – Collaborate/View/download visualizations

To publish a data source connection to the Tableau server, at least one user needs to have the ‘Creator’ privilege. A data source can be configured as connection for ‘Live’ data or as an ‘extract’ to be synchronized and stored on the Tableau server. Additionally, the creator has the option to embed user credentials in the data connection (i.e. Service Account) or prompt the viewers to key-in their credentials

while viewing the dashboards.

Tableau uses JDBC drivers to connect to backend database engines such as Presto. For more information, refer the below links:

https://help.tableau.com/current/pro/desktop/en-us/examples_presto.htm

https://prestodb.io/download.html (JDBC driver)

Server; IP-10-22-27-51.fcs.faa.gov
Port: 8446
Catalog: Hive
Authentication; LDAP
User Name: FAA Windows user name
Password: FAA Windows Password
(Requires SSL)

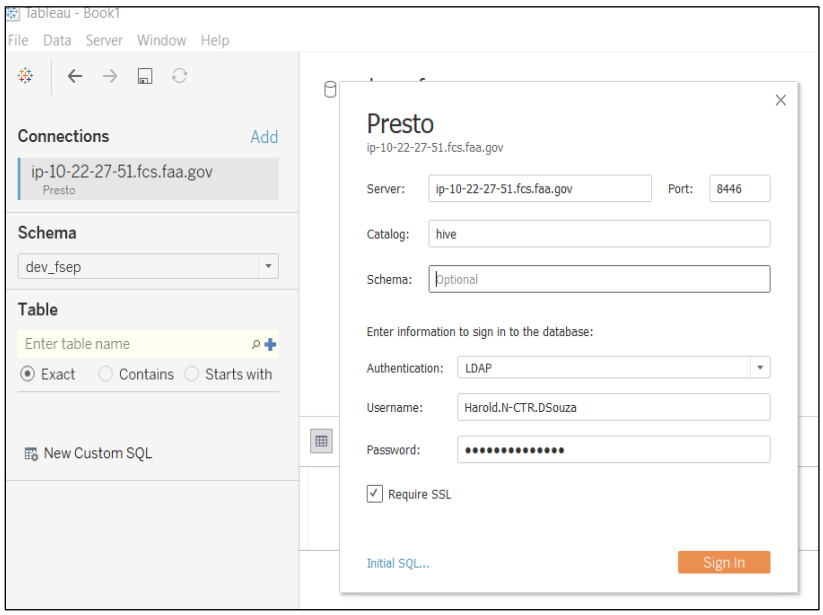


Figure 27: Presto Integration

6.5 AWS Services Access

How to create services using either AWS Console, AWS cli, or Ansible. Create an SNS (Simple Notification Service) topic to allow the end user to obtain email or other types of output returned.

6.5.1 Using AWS Console:

https://docs.aws.amazon.com/sns/latest/dg/sns-getting-started.html

1. Go to SNS Dashboard
2. Click Create Topic
    a. Fill in
        i. Topic Name
        ii. – Internally used name
        iii. Display Name – From name used in emails
    b. Click ‘Create Topic’ button

c. Click ‘Other topic actions’ button
   i. Choose ‘Edit topic policy’
      1. Change at minimum ‘Allow these users to subscribe to this topic’ section
         a. Choose radio button ‘Only users with endpoints that match’ then type in ‘@faa.gov’
   ii. Change ‘Use these delivery protocols’ to what is being used in this topic
d. In Subscriptions section
   a. Click ‘Create subscription’ button
      i. Change ‘Protocol’ to whatever is being used like ‘Email’
      ii. Fill in ‘Endpoint’ with value like email.ctr.address@faa.gov for email or appropriate value for protocol used.

6.5.2 Using AWS CLI

https://docs.aws.amazon.com/cli/latest/reference/sns/index.html#cli-aws-sns

1. Go to a server that has or can install AWS cli
   a. sudo pip install AWSCLI
2. Create topic:

   aws sns create-topic --name < Internally used name > --attributes '{ "DisplayName": "
   From name used in emails }'

   Returns:
   {
   "TopicArn": "arn:aws-us-gov:sns:us-gov-west-1:<account number>:<topic name>"
   }
3. Set topic’s policy (Note the json attribute value must have internal double quotes delimited):

   aws sns set-topic-attributes --topic-arn "< TopicArn returned from previous step >" --attribute-name "Policy" --attribute-value "{
   \"Version\": \"2008-10-17\",
   \"Id\": \"__default_policy_ID\",
   \"Statement\": [
   {
   \"Sid\": \"__default_statement_ID\",
   \"Effect\": \"Allow\",
   \"Principal\": {
   \"AWS\": \"*\"
   }, … }”
4. Add subscribers to topic:

   aws sns subscribe \
   --topic-arn < Topic arn returned from step 2 > \
   --protocol email \
   --notification-endpoint < email address of the subscriber >

6.6 EMR Cluster Security

The EMR cluster is secured using TLS encryption for all intra-cluster communication as well as for data at rest. Security configuration is defined in advance of provisioning an EMR cluster and can be consistently used for all EMR clusters to secure all components of a cluster. Below listed options are pre-defined in the security configurations as a template.

6.7 S3 Encryption

Enabling at-rest encryption for EMRFS data in Amazon S3: The EMR File System (EMRFS) is an implementation of HDFS that all Amazon EMR clusters use for reading and writing regular files from Amazon EMR directly to Amazon S3. EMRFS provides the convenience of storing persistent data in Amazon S3 for use with Hadoop while also providing features like consistent view and data encryption. Amazon S3 encryption works with EMR File System (EMRFS) objects read from and written to Amazon S3. Specify server-side encryption (SSE-S3)**.**

6.7.1 Enable At-rest Encryption for Local Disks

Amazon EC2 instance store volumes and the attached Amazon Elastic Block Store (EBS) storage volumes are encrypted using AWS KMS Keys. This includes EBS encryption of EBS root device and storage volumes.

6.7.2 Data in Transit Encryption

Enable in-transit encryption Transport Layer Security (TLS) provides encrypted channel of communication between the hosted applications user interfaces as well as the inter-node communication between the nodes of the cluster for distributed compute. TLS encryption is enabled by providing the privatekey.pem and certificateChain.pem files as zip files as part of the security configuration.

6.7.3 Securing Hadoop Components

Several encryption mechanisms are enabled with in-transit encryption and at-rest encryption in the security configuration:

Hadoop – Hadoop MapReduce Encrypted communication uses TLS

- Secure Hadoop RPC is set to "Privacy" and uses SASL
- Data encryption on HDFS block data transfer uses AES 256

Presto – Internal communication between Presto nodes uses SSL/TLS

Spark – Internal RPC communication between Spark components, such as the block transfer service and the external shuffle service, is encrypted using the AES-256 cipher in Amazon EMR.

6.8 Admin Access to EMR Clusters

Administrators access the EMR cluster nodes using secure shell communication (SSH). This requires the use of a key pair Pem file that is associated with the cluster at the time of provisioning. The key pair file is for use by the administrator/operational team and not accessible by any common users of the EIM platform.

6.8.1 End User Access to Compute EMR Cluster

The normal user access to the EMR cluster is controlled by the SAML federation with FAA Azure AD. EMR Compute cluster user interfaces are integrated with the Azure ADFS. FAA users with valid PIV card can authenticate using the Azure federation SAML provider, which provides the necessary user specific tokens to launch one to the two applications on EMR cluster (Hue or Jupyter). The data available for the user is controlled by user's level of privilege as defined in the EIM Role Based Access control policies.

6.8.2 Ingest EMR Cluster (Spark and Flink Jobs)

EIM uses EMR clusters as part of ingest processing i.e. the P2 conversion of raw datasets (i.e. TFMS, NOP). The EMR clusters are also used for monitoring the SWIM data using analytics on the logs generated from Solace client. The core processing of these clusters uses Spark and Flink frameworks.

The ingest are manually provisioned from the console as described in the sections below. To provision an EMR Cluster for Running the Flink and Spark jobs follow the steps below:

1. Go to AWS Console and Select EMR Service and click on Create Cluster

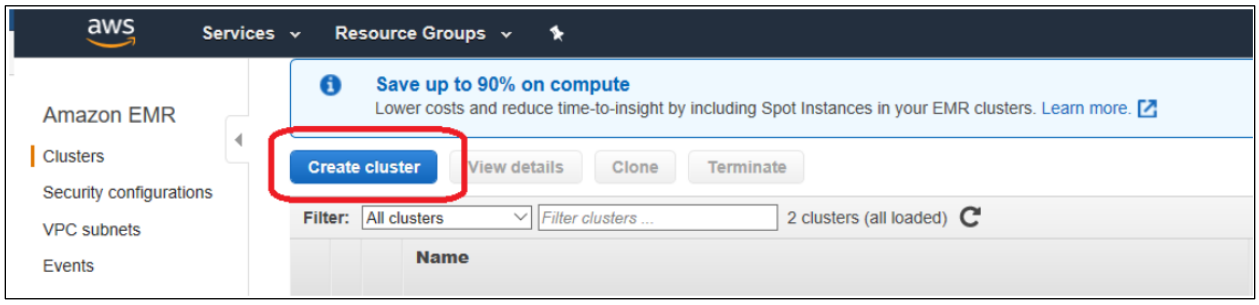
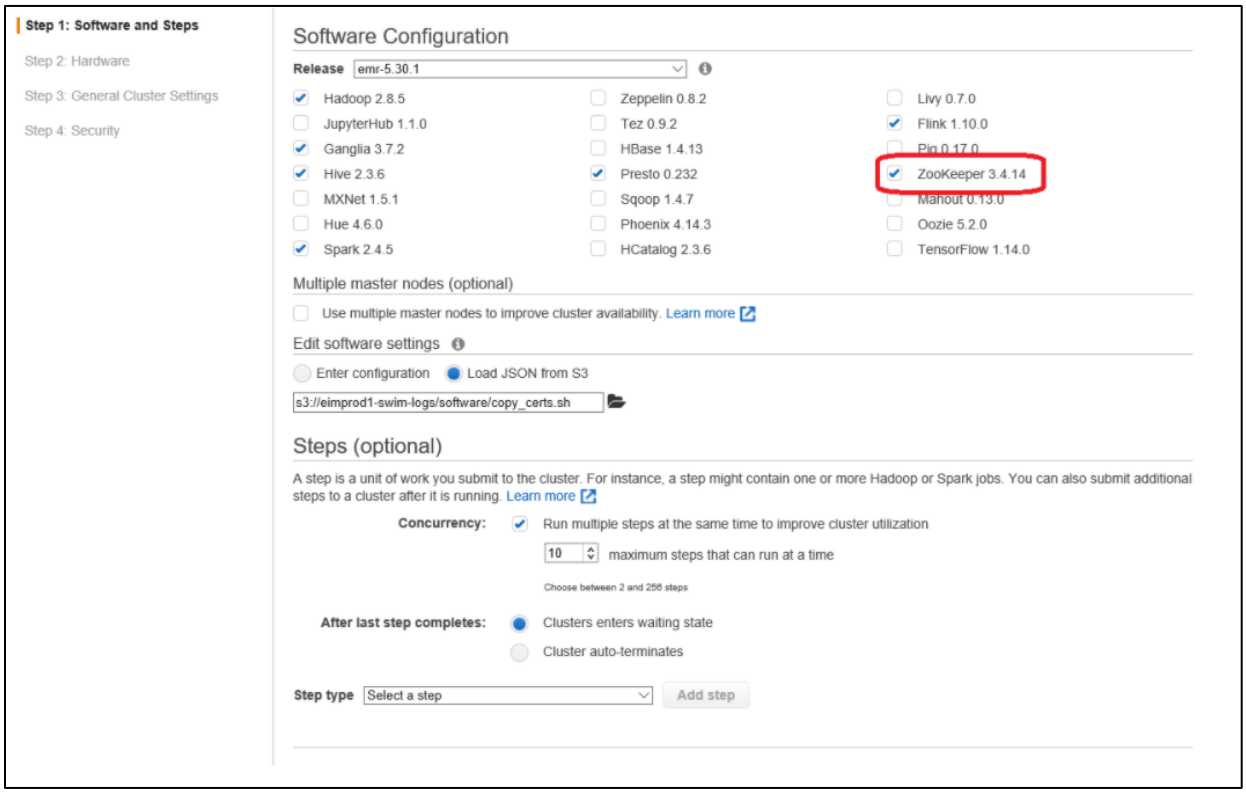
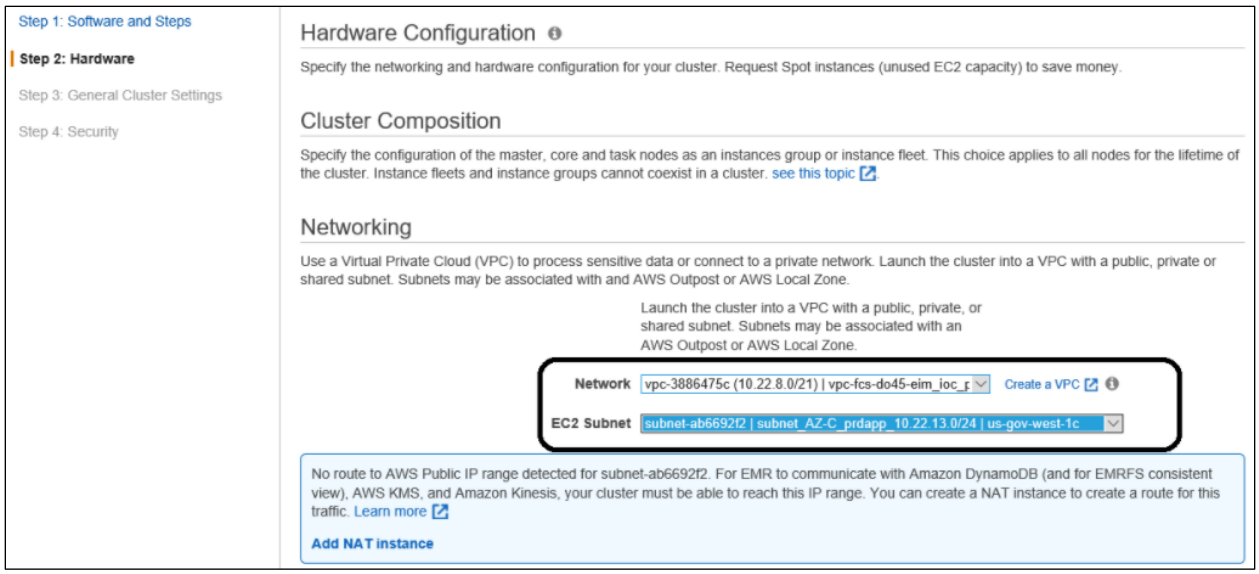


Figure 28: AWS Console

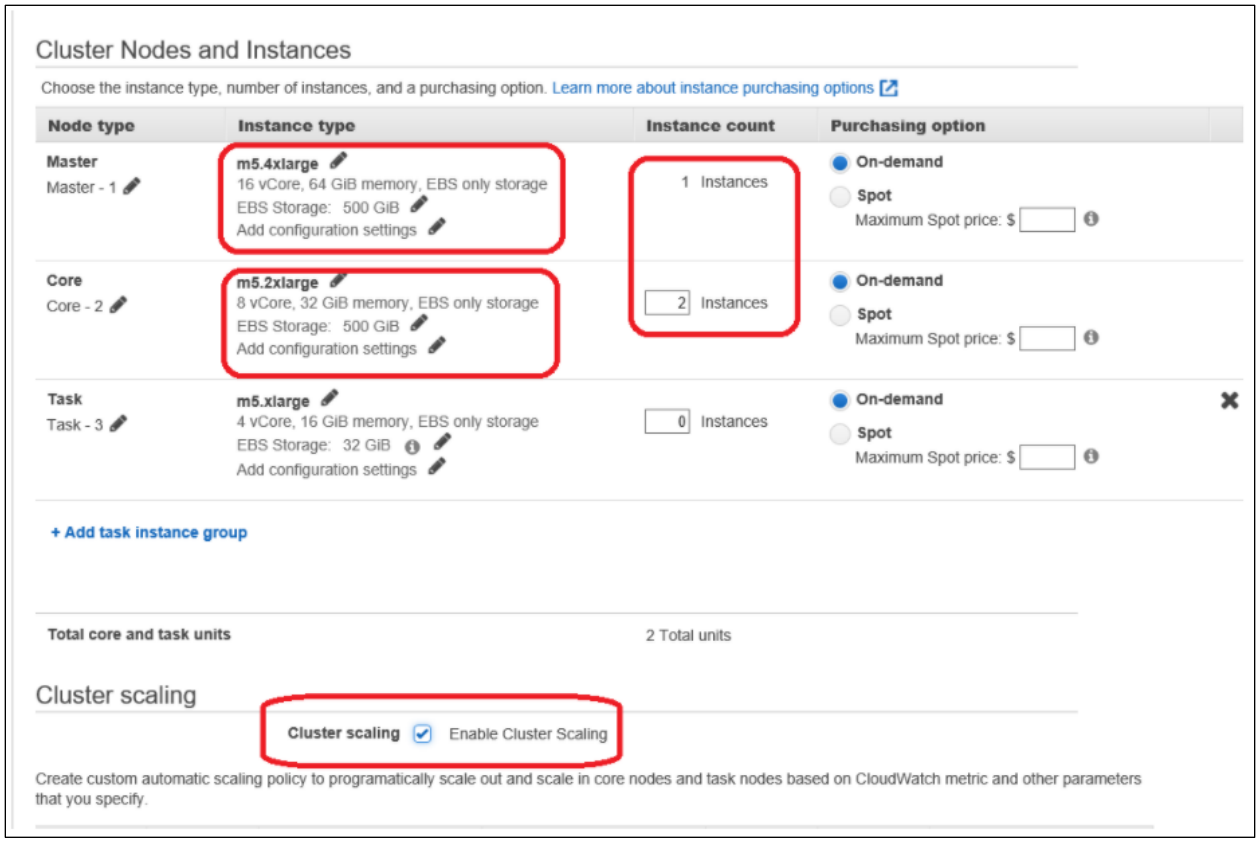
2. Select the option listed below (Zookeeper is only required if the developer want HA Option)



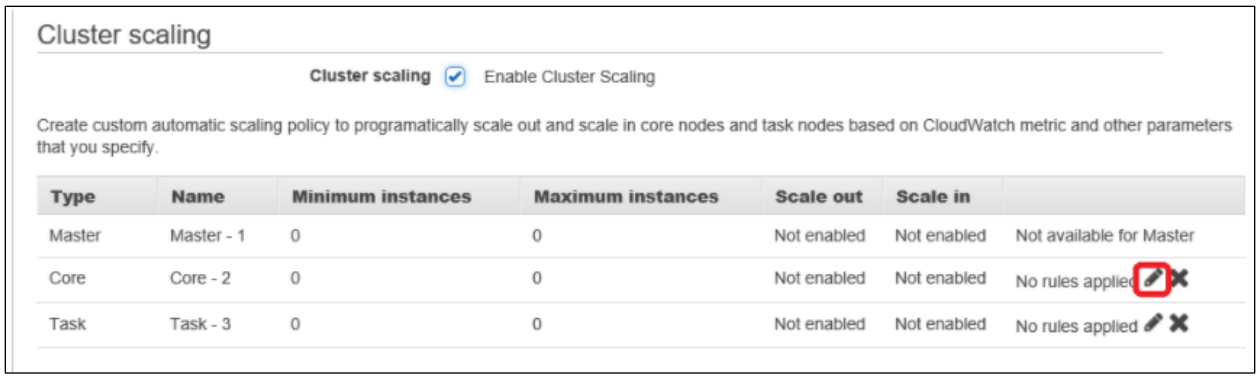
3. Confirm the right VPC and Subnet are selected.



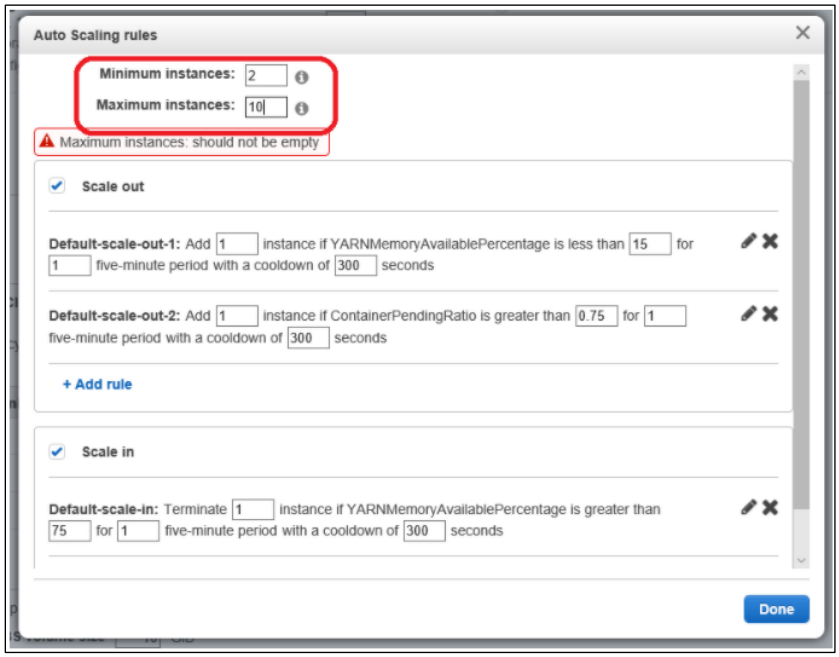
4. **Select** proper instance type for Master and Core Nodes. Also, ensure the EBS Volume Storage has changed. Enable Cluster Scaling.



5. In Cluster Scaling, **click** on Edit option for Core Nodes.

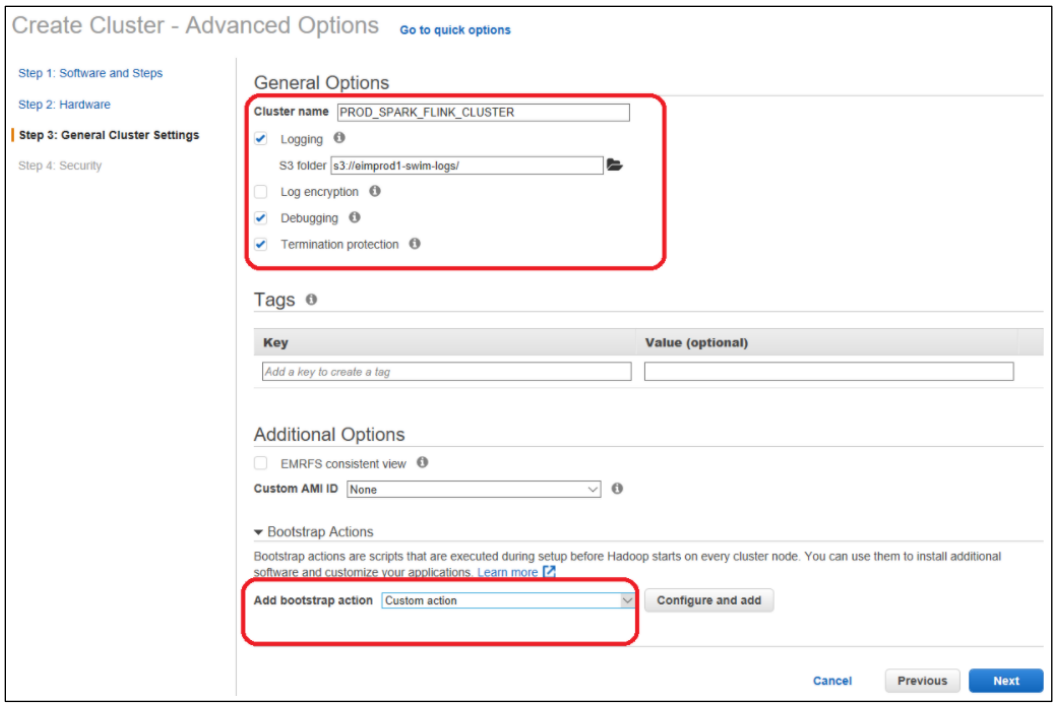


6. Enter the min and max instances.

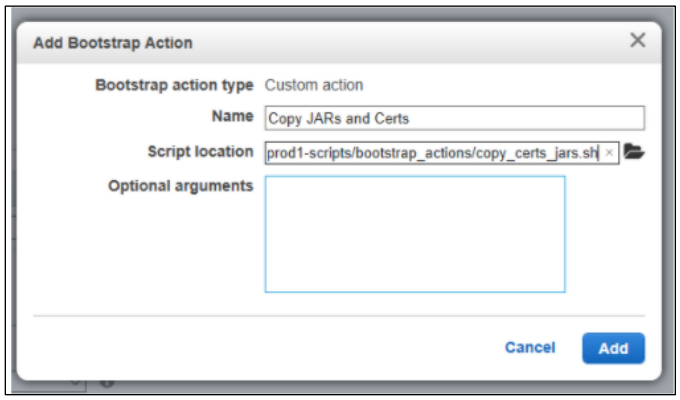


7. Change the Root Volume Size to 100GB.

8. Enter the Cluster Name and Logging Location and **select** "Custom action" for Bootstrap actions.



9. Update the Script Location and Name.

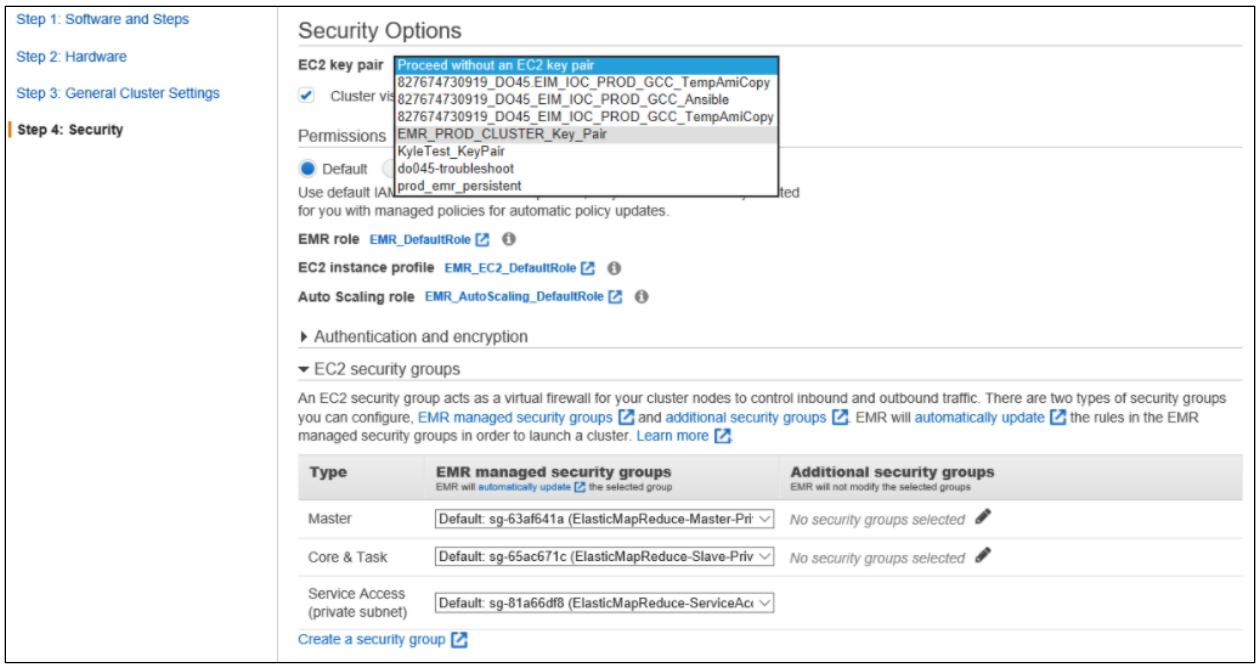


10. This Bootstrap Action does the following.

- Copying the Certs required by Elastic into Local EBS Volume
- Copying the Spark Application JAR from S3 to local EBS Volume
- Importing the Certs into Java trust Store

```
s3://eimprod1-scripts/bootstrap_actions/copy_certs_jars.sh
```

11. **Select** the Key pair and make sure the Security groups are proper.



12. Finally click **Create Cluster** and it should create the Cluster.

6.9 Spark and Flink Jobs in EMR

The developers are using EMR to run Flink and Spark Streaming jobs to address the following use cases

- Converting Data Types of incoming feeds (Ex from XML to AVRO)
- To Add the metadata (JMS Header info) to the Actual Message
- To Organize the Data based on certain metadata attributes

6.9.1 Flink Jobs

Job Name	Description	Service / Component	Comment
TFMS Flow Flink JMS Headers	Extracts the Metadata from Kafka Headers and Updates the Message Payload	SWIM	
TFMS Flow Flink JMS Headers	Extracts the Metadata from Kafka Headers and Updates the Message Payload	SWIM	
TFMS Flow Flink JMS Headers	Extracts the Metadata from Kafka Headers and Updates the Message Payload	SWIM	
NOP Filebeat Flink Site A	Extracts the Site Info from Kafka Headers and Writes the Message to Site Specific Partition in Target Kafka Topic	NOP	NOP Data from Site A
NOP Filebeat Flink	Extracts the Site Info from Kafka Headers and Writes the Message to Site Specific Partition in Target Kafka Topic	NOP	NOP Data from Site B

6.9.2 Spark Jobs

Job Name	Description	Service / Component	Comment
TFMS_FLIGHT_XML_TO_AVRO_PROD	Converts the TFMS Flight XML Data into AVRO Format	SWIM	
TFMS_FLOW_XML_TO_AVRO_PROD	Converts the TFMS Flow XML Data into AVRO Format	SWIM	
SFDPS_XML_TO_AVRO_PROD	Converts the SFDPS XML Data into AVRO Format	SWIM	

All the above Listed Spark Jobs are developed using the Spark Streaming and the Source code of the Project is in Git

https://git.faa.gov/projects/EIM-EC/repos/eim.swim.avro.converter.spark/browse

6.9.3 How to Run Spark Jobs

Initially they are launched from AWS Console.

Here is how the EIM developer launch the TFMS Flight Spark Job

```
spark-submit --executor-memory 4g
--conf spark.executor.extraJavaOptions=-Dspring.profiles.active=prod
--packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5,org.apache.
spark:spark-avro_2.11:2.4.5,com.databricks:spark-xml_2.11:0.9.0
--driver-java-options=-Dspring.profiles.active=prod
--jars s3://eimprod1-scripts/jars/spark-xml-to-avro-converter-0.0.2-SNAPSHOT.jar
--class gov.faa.eim.service.swim.avro.converter.ConsoleApplication
/home/hadoop/spark-xml-to-avro-converter-0.0.2-SNAPSHOT.jar
ip-10-22-8-160.fcs.faa.gov:9092,
ip-10-22-8-161.fcs.faa.gov:9092,
ip-10-22-8-162.fcs.faa.gov:9092,ip-10-22-8-163.fcs.faa.gov:9092,
ip-10-22-8-164.fcs.faa.gov:9092,
ip-10-22-12-160.fcs.faa.gov:9092,
ip-10-22-12-161.fcs.faa.gov:9092,
ip-10-22-12-162.fcs.faa.gov:9092,
ip-10-22-12-163.fcs.faa.gov:9092
SWIM_TFMS_FLIGHT_WITH_JMS_HEADERS_HA
tfmDataService tfmDataService tfmDataService
s3://eimprod1-swim-tfmsflight/schema/tfms-flight_xml.json
s3://eimprod1-swim-tfmsflight/tfms-flight/flight-track/convert/avro/ true
TFMS_FLIGHT
s3://eimprod1-swim-logs/swim/tfms-flight/checkpoint/
```

6.9.4 How to Run Flink Jobs

We run a Long Running Flink Session using the following Command as a Step Function

```
flink-yarn-session -s 16 -tm 8192 -d
```

This Returns an Application Id

Then the developer launch/submit each of the Flink Jobs to the Above Flink Application Id

Example:

TFMS Flight Flink JMS Headers

```
 flink run -m yarn-cluster -p 2 -yid application_1597342444892_0001
-yD env.java.opts=-Dspring.profiles.active=prod-tfms-flight
-yD --spring.profiles.active=prod-tfms-flight
/home/hadoop/swim-jms-headers-flink-0.0.1-SNAPSHOT.jar
```

NOP Filebeat Flink Site A

```
flink run -m yarn-cluster -p 2 -yid application_1597342444892_0030
-yD env.java.opts=-Dspring.profiles.active=prod-a
-yD --spring.profiles.active=prod-a
/home/hadoop/nop-filebeat-flink-0.0.1-SNAPSHOT.jar
```

6.9.5 How are Monitoring the Spark/Flink Jobs Completed

- Currently a Spring Boot Service to monitor the Spark and Flink Jobs on Edge Node is running.
- This Job monitors the EMR Cluster (configurable using the launch script) and if the Spark Job has Stopped it can launch the Job
- Currently it is scheduled to check the health of Spark Jobs **every Minute (at 45th second)**
- This Monitoring Job is configured to send SNS Notifications to a Topic (**arn:aws-us-gov:sns:us-gov-west-1:827674730919:EIMdevops**).
- However the required privileges are not granted to the Role with which the developer does the Monitoring Job.

7 AWS Managed Kafka Service

Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a fully managed service that enables developers to build and run applications that use open-source Apache Kafka to process streaming data. Amazon MSK provides the control-plane operations, such as those for creating, updating, and deleting clusters. Apache Kafka client operations, such as those for producing and consuming data tooling, and plug-ins are fully compatible to use in MSK Kafka.

The diagram demonstrates the interaction between the MSK Components and EIM ingest processing clusters.

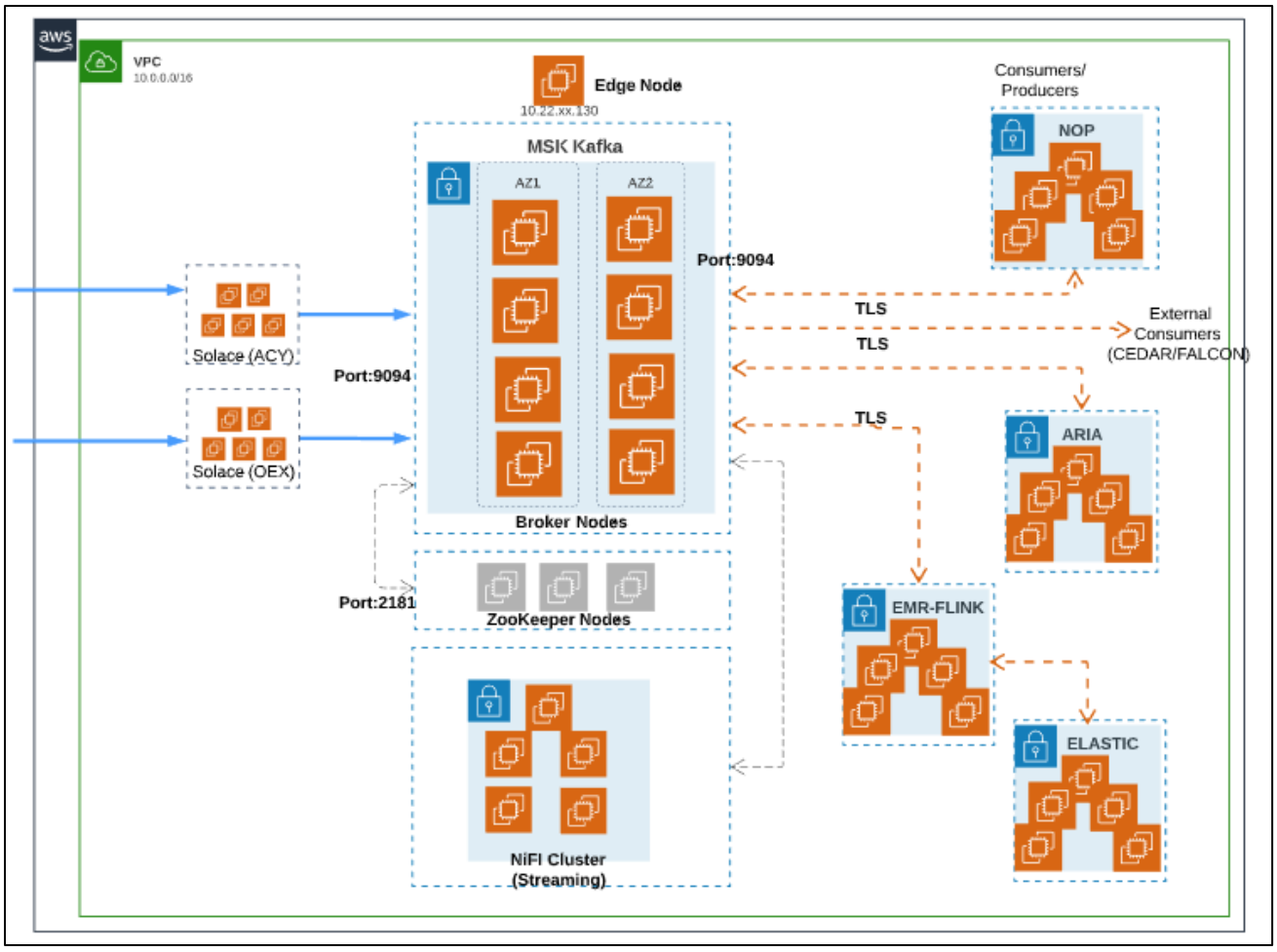


Figure 29: MSK Components and EIM Ingest Processing Clusters

- **Broker nodes** — Amazon MSK cluster is provisioned using broker nodes based on your specific use case. Each broker can be configured in a different subnet corresponding to an Availability Zone in your VPC.
- **ZooKeeper nodes** — Amazon MSK also creates the Apache ZooKeeper nodes (three nodes for quorum). Apache ZooKeeper is an open-source server that enables highly reliable distributed coordination and these nodes are internally managed by AWS
- **Producers, consumers, and topic creators** — Amazon MSK allow the developer use Apache Kafka data-plane operations to create topics and to produce and consume data.
- **AWS CLI** — The developer can use the AWS Command Line Interface (AWS CLI) or the APIs in the SDK to perform control-plane operations. For example, you can use the AWS CLI or the SDK to create or delete an Amazon MSK cluster, list all the clusters in an account, or view the properties of a cluster.

Amazon MSK detects and automatically recovers from the most common failure scenarios for clusters so that your producer and consumer applications can continue their write and read operations with minimal impact. When Amazon MSK detects a broker failure, it mitigates the failure or replaces the unhealthy or unreachable broker with a new one. In addition, where possible, it reuses the storage from the older

broker to reduce the data that Apache Kafka needs to replicate. Your availability impact is limited to the time required for Amazon MSK to complete the detection and recovery. After a recovery, your producer and consumer apps can continue to communicate with the same broker IP addresses that they used before the failure.

7.1 Producer and Consumer authentication

Broker Port Information

The following list provides the numbers of the ports that Amazon MSK uses to communicate with client machines.

- To communicate with producers and consumers in plaintext, brokers use port 9092.
- To communicate with producers and consumers in TLS, brokers use port 9094.
- Apache ZooKeeper nodes use port 2181.
- Identity and Access Management
- Security group – access to the server
- ACLs to access the topics

7.2 EIM Dev – MSK Kafka Brokers (Bootstrap servers)

b-2.dev-eim-msk.6sal25.c4.kafka.us-gov-west-1.amazonaws.com:9094
b-3.dev-eim-msk.6sal25.c4.kafka.us-gov-west-1.amazonaws.com:9094
b-4.dev-eim-msk.6sal25.c4.kafka.us-gov-west-1.amazonaws.com:9094

7.3 ZooKeeper connections

z-1.dev-eim-msk.6sal25.c4.kafka.us-gov-west-1.amazonaws.com:2181
z-3.dev-eim-msk.6sal25.c4.kafka.us-gov-west-1.amazonaws.com:2181
z-2.dev-eim-msk.6sal25.c4.kafka.us-gov-west-1.amazonaws.com:2181

7.4 Configuring Consumers and Producer Clients

MSK Kafka is an AWS managed cluster that does not provide SSH access to individual broker nodes.

To communicate with MSK brokers, you need to setup a clientEC2 instance (or Edge node in EIM) and install the Kafka client components. For more details, refer to the documentation from AWS (Kafka Client setup). The access to Kafka Brokers is managed using security group configurations. All Kafka connections from clients (producers and consumers) must be added to the inbound rules for MSK cluster.

Access to Topics is managed thru Kafka access control lists (ACL).

7.5 Configuring ACL to access MSK Topics

```
kafka-acls.sh --authorizer-properties zookeeper.connect=<zookeeper
nodes endpoint:2181 >
--add --allow-principal User:* --operation All
--topic
<list of topics >
--allow-host
<list of client hosts, NiFi, or other producers or consumers>
```

7.6 Encryption of data at rest

MSK cluster configuration provides the data at rest encryption settings using customer managed Customer Master Keys (CMK) in AWS Key Management Service (KMS).

Example:

```
{
  "EncryptionAtRest": {
    "DataVolumeKMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/abcdabcd-1234-abcd-1234-abcd123e8e8e"
  },
  "EncryptionInTransit": {
    "InCluster": true,
    "ClientBroker": "TLS"
  }
}
```

7.7 Encryption of Data in Transit

All communication between the MSK brokers and clients is encrypted using the TLS protocol using the TLS cert of the Kafka broker node.

This is done using the SSL certificate from any of the Kafka brokers:

Example:

openssl s_client -connect b-2.dev-eim-msk.6sal25.c4.kafka.us-gov-west-1.amazonaws.com:9094 > eim_dev_msk.pem

Import the certificate into the trust store of the client application:

keytool -import -keystore /home/nifi/cacerts.jks -alias eim_dev_msk -file /home/nifi/eim_dev_msk.pem

keytool -import -keystore /opt/nifi/conf/truststore.jks -alias b-3.dev-eim-msk.6sal25.c4.kafka.us-gov-west-1.amazonaws.com -file /home/nifi/eim_dev_msk.pem

keytool -import -keystore /opt/nifi/conf/eim-truststore.jks -alias b-3.dev-eim-msk.6sal25.c4.kafka.us-gov-west-1.amazonaws.com -file /home/nifi/eim_dev_msk.pem

7.8 IAM Policies

By default, IAM users and roles do not have permission to execute Amazon MSK API actions. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions. For example, if a Lambda job needs access to MSK topics, the IAM user or role that is attached to the Lambda function must have the MSK policies attached to the role.

7.9 Create Kafka Topics

1. List of Kafka Topics Required for SWIM XML to AVRO Conversion

```
# SWIM_SFDPS_WITH_JMS_HEADERS

./kafka-topics.sh --create --zookeeper 10.22.12.100:2181 --replication-factor 3 --partitions 25 --topic SWIM_SFDPS_WITH_JMS_HEADERS

# SWIM_TFMS_FLIGHT_WITH_JMS_HEADERS
```

```
./kafka-topics.sh --create --zookeeper 10.22.12.100:2181 --replication-factor 3 --
partitions 4 --topic SWIM_TFMS_FLIGHT_WITH_JMS_HEADERS

# SWIM_TFMS_FLOW_WITH_JMS_HEADERS

./kafka-topics.sh --create --zookeeper 10.22.12.100:2181 --replication-factor 3 --
partitions 4 --topic SWIM_TFMS_FLOW_WITH_JMS_HEADERS
```

2. Kafka Topics Required for AIRA

```
# ARIA_AIRBORNE_EVENTS

./kafka-topics.sh --create --zookeeper 10.22.12.100:2181 --replication-factor 3 --
partitions 200 --topic ARIA_AIRBORNE_EVENTS
```

3. Kafka Topics Required for TBFM

```
# SWIM_TBFM_OEX

./kafka-topics.sh --create --zookeeper 10.22.12.100:2181 --replication-factor 3 --
partitions 10 --topic SWIM_TBFM_OEX

# SWIM_TBFM_ACY

./kafka-topics.sh --create --zookeeper 10.22.12.100:2181 --replication-factor 3 --
partitions 10 --topic SWIM_TBFM_ACY

# SWIM_TBFM_HA

./kafka-topics.sh --create --zookeeper 10.22.12.100:2181 --replication-factor 3 --
partitions 10 --topic SWIM_TBFM_HA
```

4. Kafka Topic for NOP20 (nopPoints)

```
# nop Points from Source A

./kafka-topics.sh --topic nopPoints20_A --zookeeper 10.22.12.100:2181 --create --
replication-factor 3 --partitions 195

# nop Points from Source B

./kafka-topics.sh --topic nopPoints20_B --zookeeper 10.22.12.100:2181 --create --
replication-factor 3 --partitions 195

# nop Points for Deduped Topic

./kafka-topics.sh --topic nopPoints20_HA --zookeeper 10.22.12.100:2181 --create --
replication-factor 3 --partitions 195
```

8 API Management

EIM leverages MuleSoft API management platform to develop and publish APIs for consumers to access data securely from the EIM data lake. In addition to the MuleSoft APIs hosted in Mule runtime instances on the EIM platform, there are other APIs developed using Java/Springboot service. An important note here is the APIs are used primarily for the consumer side of the data lake and not for ingesting data to the data lake.

8.1 Platform Access

8.1.1 API Portal

The API public portal is accessible to anyone in the FAA network, enabling developers to access an organization's REST, SOAP, and HTTP APIs. This does not provide access to *develop* APIs. To develop APIs continue to 'Anypoint Platform Access' below.

The Public API Portals can be accessed here:

- Nonprod (PCE) https://dev-anypoint.faa.gov/exchange/portals/faa
- Production (PCE) https://anypoint.faa.gov/exchange/portals/faa-gov/
- CloudHub (API Community Manager) https://api.faa.gov

8.1.2 Anypoint Platform Access

If there is an existing Anypoint Platform business group that you want to join, ask the organization administrator to add you to an appropriate AD group. Once you are added to the AD group you will be able to log in to the platform through SSO.

Instructions for organization administrators:

- Add User to AD Group
- Verify AD Group Membership

If a new business group need to be provisioned, see instructions for Onboarding a New Business Group. Business groups are self-contained resource groups that contain Anypoint Platform resources such as APIs and applications. Business groups provide a way to separate and control access to Anypoint Platform resources, as users have access only to the business groups in which they have a role.

Business groups reside within the master organization and are organized in a hierarchical tree where the top-level business group is the root. Each business group you create has one direct parent and can have multiple children.

8.1.3 Local Users

If a basic authentication is required to log in to the platform (i.e. via Maven or Anypoint Studio) you can create local users on the platform using the following instructions:

Private Cloud Edition: Create a Local User

CloudHub:

- How to add an Anypoint Platform user after enabling External Identity
- (Preferred) Creating a Connected App for Developers

8.2 Users, Roles, and Permissions

Every business group is provisioned with a standard set of AD groups which are mapped to roles on the Anypoint Platform.

This describes the standard users, their roles, and the permissions they are granted. These roles are create for each business group. Details are included below:

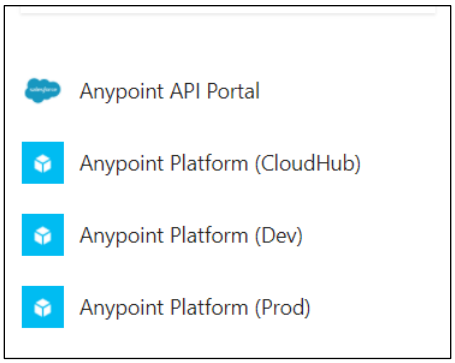
- Standard Roles and Permissions
- Custom Roles and Permissions
- Example: Sample Permissions by User
- Users mapped to AD Groups
- Example: AD Groups for EIM Team
- AD Groups mapped to Anypoint Platform Roles

8.3 New Business Group Onboarding

Follow the below steps to set up a new business group and add users to that group.
New Business Group Owner/Sponsor:
Initiate new business group request by submitting a ticket to EIM
EIM team:

1. Create a new business group
2. Use the following template to submit a help desk ticket requesting the following AD groups
3. Once AD groups created, map the AD groups to the Anypoint Roles.
    - Additional reference: Map Users to Roles in an LDAP Group
4. Add Users to AD groups using instructions above
5. Notify new business group they can log in using Azure SSO.
    - https://myapplications.microsoft.com/#optIn

Users that belong to any 'I-ADE001-APP-Mule-*' AD group will see the following tiles:



Important: This only involves provisioning platform access and does not include provisioning AWS resources like EC2 instances where applications can be deployed

Proceed to AWS Infrastructure Provisioning

8.4 AWS Infrastructure Provisioning

The process of provisioning the Mule runtime instances has been automated to a significant extent using Terraform templates as well as Ansible playbooks. MuleSoft runtime and MuleSoft runtime software has been packaged as an AMI that is deployed by Terraform template.

Additional details of the specifications like instance count, environment specific variables and routing rule etc. must be provided for each use case. As part of the AMI, the latest Mule runtime distribution along with the runtime license has been bundled in to the AMI. In addition, all API related events generated can be streamed to Elastic cluster and viewed using pre-built visualizations. This is packaged into the Ansible playbook.

Mule runtime AMI's have been created for each environment – The AMI has the MuleSoft software pre-installed and configured for each environment – Dev, Stage and Prod.

Here is the AMI for "dev" = "ami-75341514" and is deployed using the Terraform template:

https://git.faa.gov/projects/EIM-EC/repos/eim-mule-terraform-provisioning/browse

Steps to provision new Mule runtime:

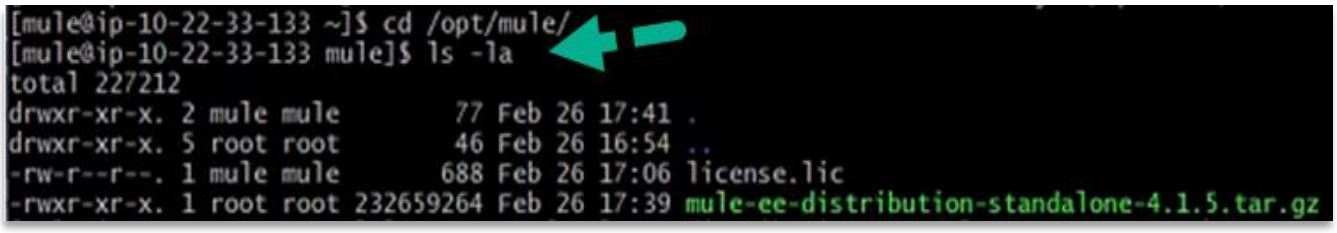
- To provision a mule runtime for an organization
- Clone one of the existing runtime templates
- Update variables.tf for the organization and other template related vats
- Update the paths (if applicable in the ALB-{env.tf)
- Create the security groups for the runtime based on requirements
- Checking the code and submit a Pull Request for approval
- On approval the terraform workspace will execute the plan
- Review the terraform plan and apply the plan.

Terraform logs can be view at the workspace. Ensure there are no errors during the apply, if so, troubleshoot and revise the template. There is a log for each run that was executed which can be download if needed:

Location of the logs: https://terraform.faa.gov/app/eim/workspaces/eim-foc-mulesoft-dev/runs

Runtime License is already incorporated into the Mule runtime AMI. For rotation of the new license upon expiry.copy "license.lic" into the /opt/mule folder.

```
$ cp license.lic  /opt/mule
```



8.5 API Access

APIs and access to APIs should be self-describing. All required information should be available on the API in the API Portal. See instructions in this document for accessing the API Portal

High-level workflow included here (for reference):

API Client:

1. Create AD Service Account

API Developer

1. Create API Permissions (Azure)
   - 'runas-user.[Service Account]' scope (used for impersonation)

API Client:

1. Create Azure Registered App
2. Request API Permissions in Azure
   - 'runas-user.[Service Account]'.scopes (impersonation)
   - scopes.[resource].[method]
3. Create help desk ticket to approve API permissions
4. Request AD group membership for Service Account
   - Map AD group to XPack Policies
   - Map AD group to Privacera Policies

Additional information

- OAuth: Create a client application and request access to an API (Azure)
- OAuth: Create an application that exposes an API (Azure)
- OAuth: Secure an API using the JWT Validation Policy (Anypoint Platform)
- Deploy the JWT Validation Policy to Exchange

Important! If there is insufficient information in the API documentation, the API documentation should be updated using the **API-PORTAL-README.md** which is downloaded via the parent pom assemblies when the Developer builds their application locally. This documentation should also be checked in and versioned with the application in BitBucket.

See API Documentation for more information.

8.6 API Documentation

The developer configure the parent pom inheritance, additionally the developers are responsible for completing the following documentation:

Reference the eim-parent-pom project for how to use and inherit the eim parent pom.

https://git.faa.gov/projects/MULE/repos/eim-parent-pom/browse

- **API-PORTAL-README.md** – this document provides a template which should be completed with full information for how to access and use the API, including any other relevant information. This should be version-controlled along with the application. Copy/Paste the content from here into the APIs documentation in Exchange.
- **RELEASES.md –** this document provides a template which should be completed each release cycle with information about the version, release date, new features, and known issues. Add this as a 'Release Notes' page in Exchange.
- **README.md** – this document provides a template which should be completed for BitBucket. This includes information for developers that may develop or test this mule application.

8.7 API Authorization Pattern

Each API layer is protected using the Microsoft (Azure) Identity Platform and the OpenID Connect Protocol.

- The Experience API Layer is protected using either Access Tokens **or** Identity Tokens.
- The Process API Layer is protected using **only** Access Tokens. This is because only Experience

APIs, which are system-based clients, will only ever access the Process API layer.

- The System API Layer is protected using **only** Access Tokens. This is because only Process APIs, which are system-based clients, will only ever access the Process API layer.

The following are the two types of identities which can be passed (per Azure):

- **Identity Token** – represents a user which is requesting the data, usually through a web application.
- **reference:** https://docs.microsoft.com/en-us/azure/active-directory/develop/id-tokens
- **Access Token** – represents a system-based client, like a daemon or another API (where a specific user is not involved in the API request).
- **reference:** https://docs.microsoft.com/en-us/azure/active-directory/develop/access-tokens

8.8 API Impersonation Pattern

The EIM Data Platform requires the identity of the client to be known at the time of the API request. The identity of the API client is extracted from either the ID Token or the Access Token that is sent by the client in the API request.

The **impersonation trait** is applied to the Process API layer and System API layer which requires that Experience API layer passes the identity of the client application that was extracted from the JWT Token. This ensures that the identity of the client is passed to the EIM Data Platform, which will be used to validate authorization to the data requested.

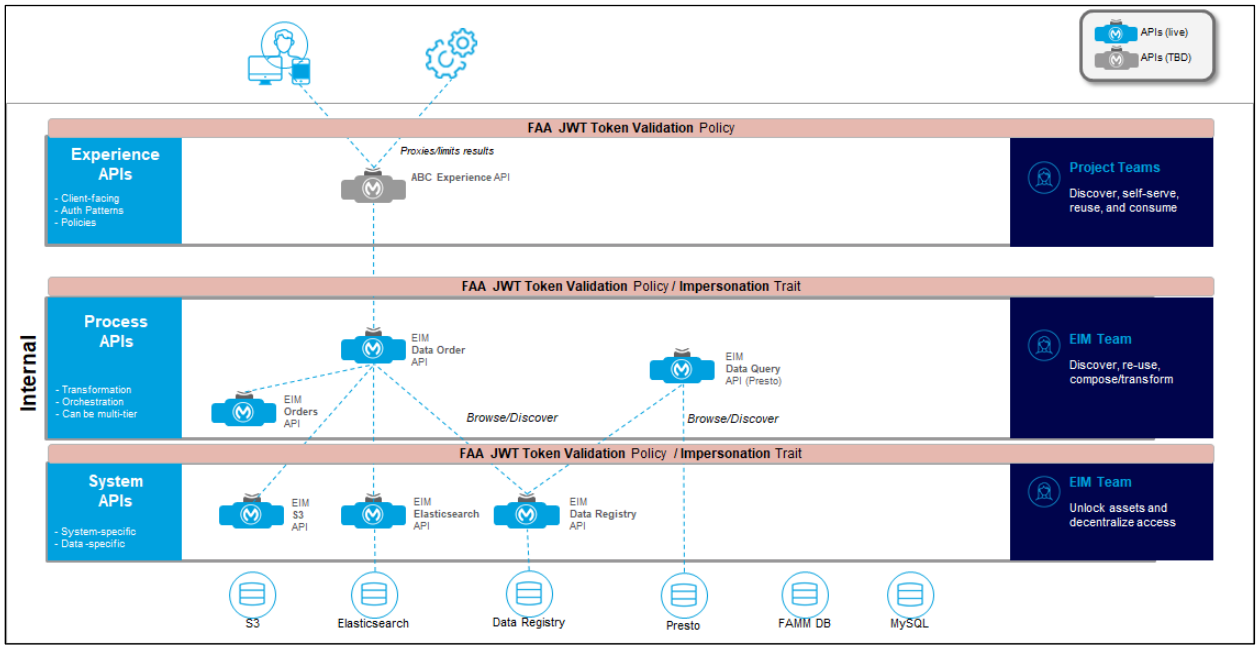


Figure 30: API layers

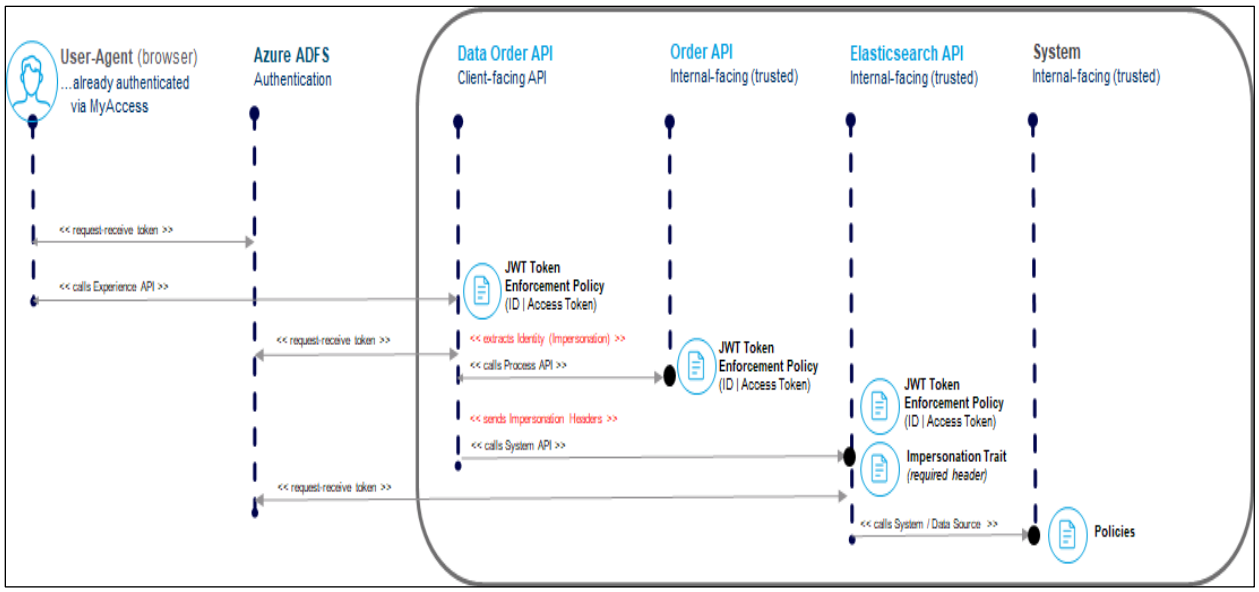


Figure 31: Azure ADFS

8.9 Migration Activities

EIM team is in the process of deploying API platform infrastructure to MuleSoft hosted Cloud service called CloudHub. Also, EIM hosted Mule runtime instances are being migrated to container based runtimes. Follow the below instructions for migration related information and activities:

Migration of Mule applications from EC2 instances to Containers

Migration of Mule applications from PCE to CloudHub

8.10 Flight Sensitivity Service

Flight Sensitivity Service is a Springboot service used to check whether a Flight is sensitive or not based on Aircraft IT (acid), Aircraft type (acType) and Beacon Code. This Service is deployed as a REST Service and users /services can invoke the REST End Point.

Latest Sensitive and Non Sensitive Flight Codes , Sensitive acTypes and Sensitive Beacon Codes are copied to S3 Periodically and the developer has a Lambda Job that is monitoring the S3 Bucket and as soon as new Files are copied to S3 it reads those Files and copies them to the EC2 Instance where the Flight Sensitive Service is Running.

Flight Sensitive Service is implemented as a Spring Boot Service and it has a Scheduler Job which monitors the local storage and loads the files (The files copied from S3 to EBS Volume by the Lambda Job) into Memory and when a Request is made for checking whether a Flight is sensitive or not it used this information in the Logic to determine and send the response to the requestor.

Environment	REST End Point	IP Address	Active
DEV	https://apps.eimdev.faa.gov/flightfilter/status	10.22.28.130	YES
PROD	https://apps.eim.faa.gov/flightfilter/status	10.22.12.130	YES
STAGE	https://apps.eimstage.faa.gov/flightfilter/status	10.22.29.130	YES

8.10.1 REST End Points Exposed

Here are the list of REST End Points exposed.

End Point	Description	Comment
/status	Main End Point for finding the Flight Status	
/acids	To get the List of Sensitive Aircraft Ids	
/acTypes	To get the List of Sensitive Aircraft Types	
/beaconCodes	To get the List of Sensitive Beacon Codes	

8.10.2 Sample Request and Response

Http GET request

```
https://apps.eimdev.faa.gov/flightfilter/status?acid=ABCD&acType=B787&beaconCode=7777
```

Response

```
{
    "acid": "ABCD",
    "acType": "B787",
    "beaconCode": "7777",
    "errorsMap": {},
    "sensitive": true
}
```

9 Fortify on Demand Static Application Security Scanning

Fortify on Demand (FoD) provides an integrated secure development, security testing and continuous monitoring application for developers. Fortify on Demand is independent, third-party system of record, conducting a consistent, unbiased analysis of an application and providing a detailed tamper-proof report to the security team. The developer must create their own personal account on FoD site to perform the assessment and create the security report.

9.1 Vulnerability Remediation Process

To complete a static assessment, an application must meet the Fortify SCA minimum requirements for currently supported languages and must successfully compile prior to submission of the application.

Assessments must be run according to these standards. In situations where there are multiple Units of Code in one application solution, each of these units would be assessed individually.

A Unit of Code can vary by language. Below are a few examples:

- **Java** – A single EAR, or multiple associated WAR/JAR files that would run on a single server instance of Apache Tomcat (for example).
- **.NET** – A single Visual Studio solution containing stand-alone code for a web application.
- **Python** – A collection of Python code files, including any required imports necessary to execute the application.

- **ABAP** – A program and its dependent components which are compiled/activated together such as a report, a module pool program or a BSP application with dependent units included.
- **Objective-C** – An X-code project file, full source code, and all frameworks required to build the iOS application.

Service Oriented Architecture (SOA) – For applications that are built using a service oriented architecture, each component is considered a unit of code, even though all components could run on a single platform, as they are loosely coupled.

To ensure complete and accurate scan results, developers need to provide both compiled code and source code files (when applicable) along with any third party libraries. Any code submitted must be fully deployable (fully functional). The FoD link is below:

https://fed.fortify.com/Applications?rpp=25&sortdir=Asc

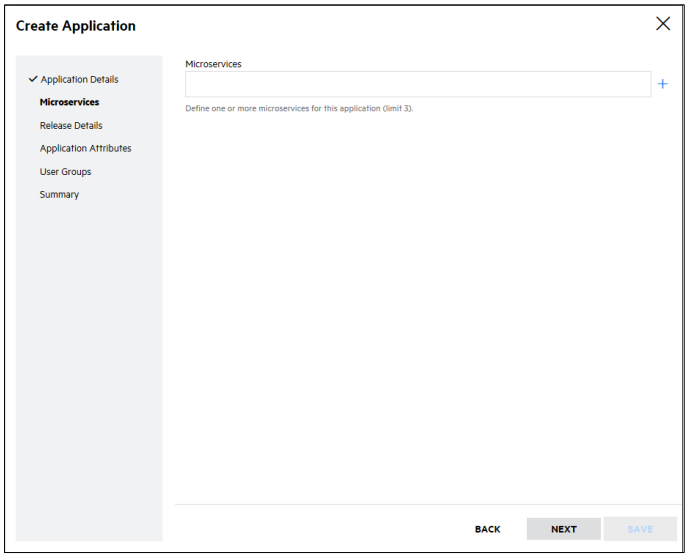
Open an internet browser to create an account (with personal information) to log in to the FoD portal. There is no software to install from the developer's site. Enter the credentials created from the initial login, include the unique tenant id.

9.2 Creating an Application

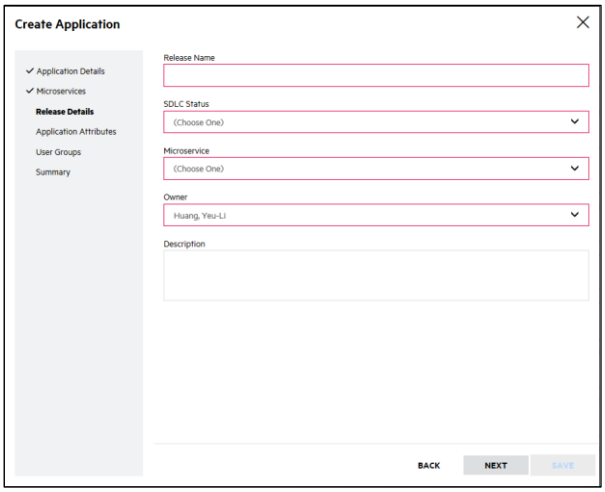
- Select the **Applications view**. Your Applications page appears.
- Click **+New Application**. The Create Application wizard appears.
- In the **Application Details** page, define the application. Fields are required, unless otherwise noted.

- Click **Next**.
- If you designated the application as a micro service application above, in the Micro services page, type the name of a microservice in the text box and click +.
- The microservice is added below. You can add up to 10 micro services.

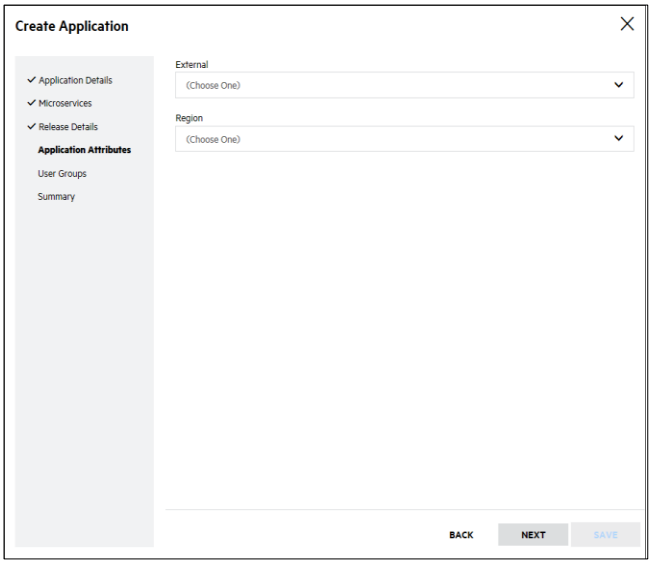
NOTE: You can also add micro services to an application after it has been created.



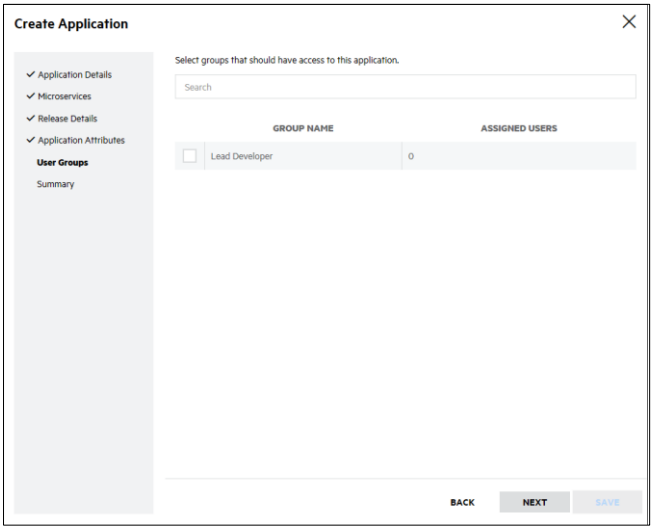
- Click **Next**.
- In the **Release Details** page, define a release of the application. The release represents an iteration of your application that will be tested. Fields are required, unless otherwise noted.



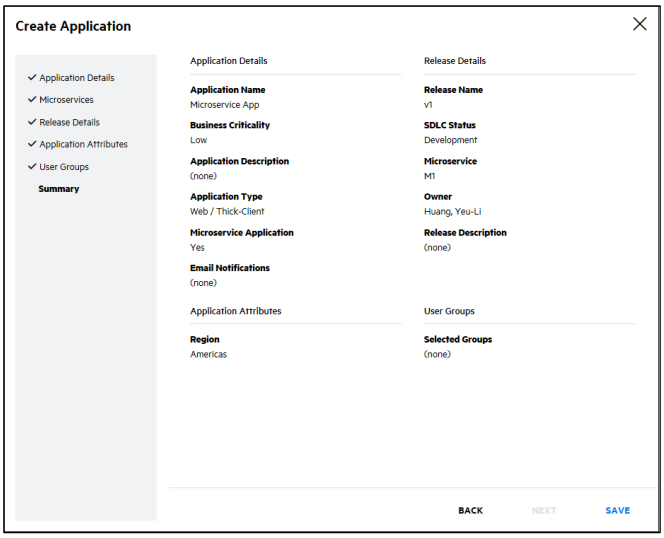
- Click **Next**.
- If custom attributes have been configured for your tenant, in the **Application Attributes** page, specify the application attributes.



- Click **Next**.
- If user groups have been configured for your tenant, in the **User Groups** page, select the groups that have access to the application. You can use the search box to search group names.



- Click **Next**.
- In the **Summary** page, review the application settings.



- Click **Create Application**.

The user is redirected to the Overview page of their new application's release.

9.3 Fortify Security Assistant Plugin for Eclipse

Conduct a test using the plugin in Eclipse and then run the manual scans on HP Fortify. The Fortify Security Assistant for Eclipse integrates with the Eclipse Java development environment. Fortify Security Assistant for Eclipse works with a portion of the Fortify security content to provide alerts to potential security issues as the Java code is written. Fortify Security Assistant for Eclipse provides detailed information about security risks and recommendations for how to secure the potential issue. Fortify Security Assistant for Eclipse can detect:

- Potentially dangerous uses of functions and APIs
- Issues caused by tainted data reaching vulnerable functions and APIs at the intra-class level Fortify Security Assistant for Eclipse requires;
- A valid Fortify license to scan for issues (Up-to-date Fortify software security content).

There will be a prompt to provide a license file and Fortify software security content if necessary.

9.4 Installing Fortify Security Assistant for Eclipse

Install the Fortify Security Assistant Plugin for Eclipse on Windows, Linux, and macOS operating systems. To update from an earlier version of Fortify Security Assistant Plugin for Eclipse, first remove the existing version. For information about how to uninstall the plugin, see Uninstalling Fortify Security Assistant for Eclipse.

NOTE: These instructions describe a third-party product and might not match the specific, supported version being used. See the product documentation for the instructions for the version.

To install Fortify Security Assistant for Eclipse:

1. Start Eclipse.
   - Select **Help > Install New Software**.
   - The Install wizard starts and displays the **Available Software** step.
2. Click **Add**.

3. Click **Archive**, and then locate and select Fortify_SecurityAssistant_Eclipse_Plugin_ <*version*>.zip.
4. Click **OK**.
5. Select the **Fortify Security Assistant Plugin** check box, and then click **Next**.

The **Install Details** step lists **Fortify Security Assistant Plugin or Eclipse**.

To view version and copyright information about the plugin in the **Details** area, click the plugin name.

6. Click **Next**.
7. On the **Review Licenses** step, review and accept the license agreement.
8. Click **Finish**.
9. To complete the installation and restart Eclipse, click **Restart Now** when prompted. The menu bar now includes the **Fortify** menu.

There may be a prompt to specify a Fortify license. Click **Browse** in the Locate Fortify License File dialog box, navigate to the license file, and then click **OK**. Fortify Security Assistant for Eclipse verifies the license file and then attempts to download the Fortify Software Security Content from the Fortify Customer Portal.

9.5 Uninstalling Fortify Security Assistant for Eclipse

NOTE: These instructions describe a third-party product and might not match the specific, supported version currently being used. See the product documentation for the instructions for the current version.

Uninstall Fortify Security Assistant Plugin for Eclipse:

1. Start Eclipse.
2. Select **Help > About Eclipse**, and then click **Installation Details**.
3. On the **Installed Software** tab, select **Fortify Security Assistant Plugin for Eclipse**.
4. Click **Uninstall**.
5. In the Uninstall window, click **Finish**.
6. To implement the change and restart Eclipse, click **Yes** when prompted.

9.6 Configuring Fortify Security Assistant for Eclipse

Fortify Security Assistant for Eclipse requires Fortify Software Security Content to detect issues. There is an option to specify the categories of issues for Fortify Security Assistant for Eclipse to detect. Please specify the settings for the workspace or for a particular project.

To configure settings for the workspace or for a project:

1. Do one of the following: To configure settings for the workspace, select **Fortify > Configure Security Assistant**.
    - To configure settings for a project:
        i. Right-click a project, and then select **Properties**.
        ii. In the left panel, select Fortify Security Assistant.
        iii. Select Enable project specific settings.

NOTE: Specify the category of issues from a Fortify Security Assistant for Eclipse tooltip in the Code editor. Click **Configure Security Assistant**, and then click **Configure Workspace** or **Configure Project**.

2. If Fortify Software Security Content has not been loaded, click **Load Security Content**.

Fortify Security Assistant for Eclipse first attempts to load the Fortify Software Security content from a default installation of Micro Focus Fortify Static Code Analyzer. Otherwise, Fortify Security Assistant for Eclipse attempts to download the Fortify Software Security content from the Fortify Customer Portal.

If there is not a network connection to the Fortify Customer portal, there is an option to import local rules by clicking **Import Custom Security Content**.

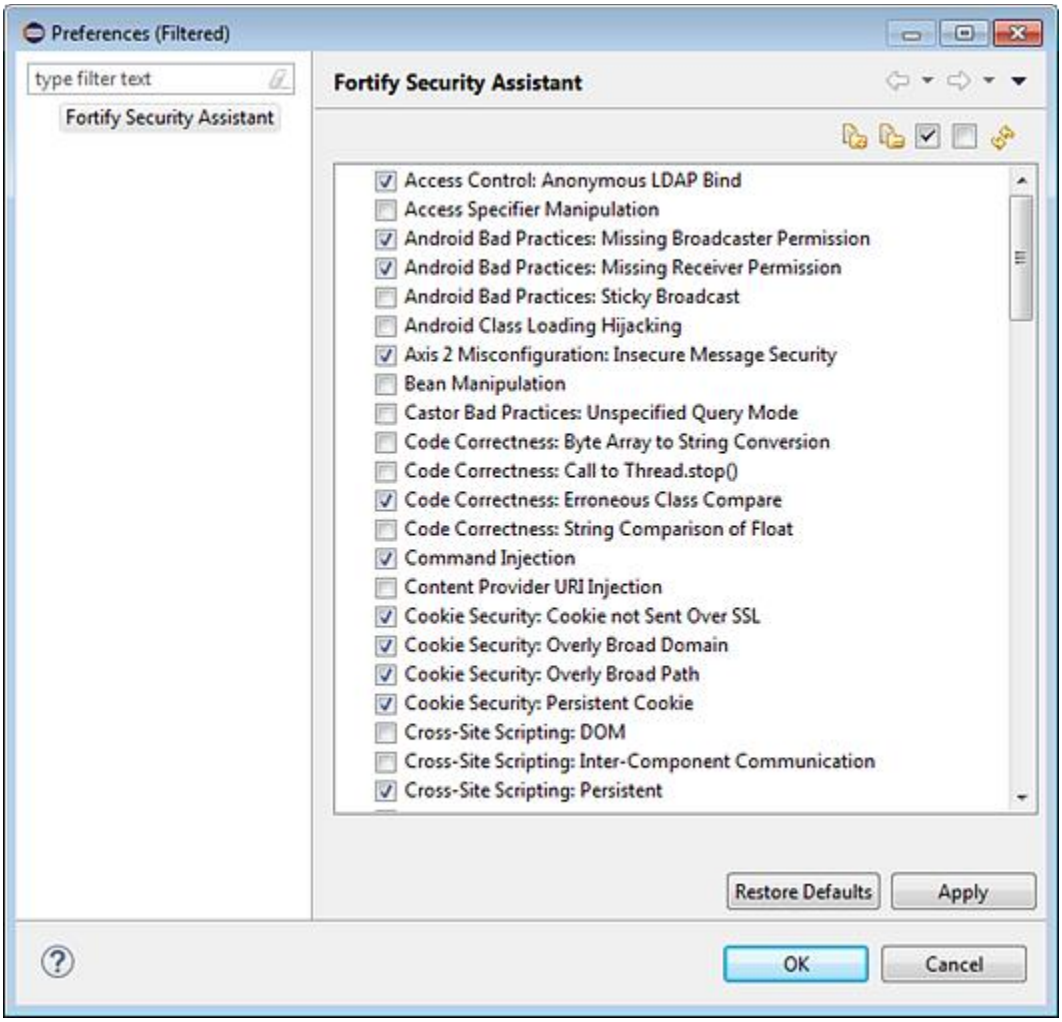


Figure 32: Fortify Security Assistant

3. Select the categories of issues for detection.
4. Right-click in the list of categories, and then select **Select All** or select **Clear All (but one)**.
5. To import custom rules:
   a. Click **Import Custom Security Content**.
   b. Navigate to where the custom rules file is located, select the XML file, and then click **Open**.

NOTE: To remove any custom rules previously imported, click **Clear All Custom Security Content**. Once completed, the action cannot be undone.

6. Click **Apply**.
7. Click **OK**.

Fortify Security Assistant Plugin for Eclipse re-inspects the project to refresh any issues reported so that it matches the specified configuration settings.

9.7 Updating Security Content

To optimize Fortify Security Assistant for Eclipse functionality, complete and up-to-date Fortify Software Security Content. To update security content from the Fortify Customer Portal, connect to the Internet and have the Eclipse network connections configured to access the Fortify Customer Portal (https://update.fortify.com).

NOTE: To update Fortify Software Security Content from a local file, import it as custom security content.

To obtain the latest security content from the Fortify Customer Portal: · Select **Fortify > Update Security Content**.

For full instructions and guidance with the using all of the tools within FoD please review the Micro Focus Fortify on Demand User Guide document and Find Fix Fortify YouTube channel below.

Fortify_on_Demand_Guide_19_5_EN.pdf

Fortify on Demand Guidance

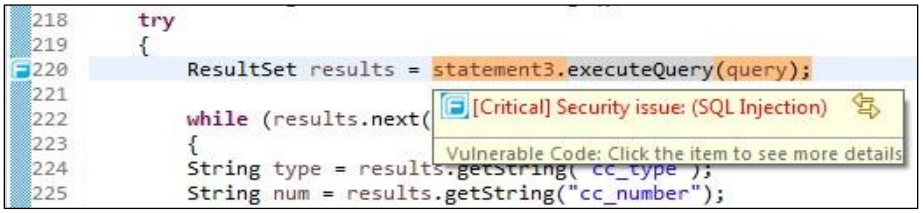
Find Fix Fortify YouTube: https://www.youtube.com/channel/UCUDKcm1wIfE6EWk_SyK0D4w/videos

9.8 Finding Security Issues with the Written Java Code

Review the information about the security issues and update the code as appropriate.

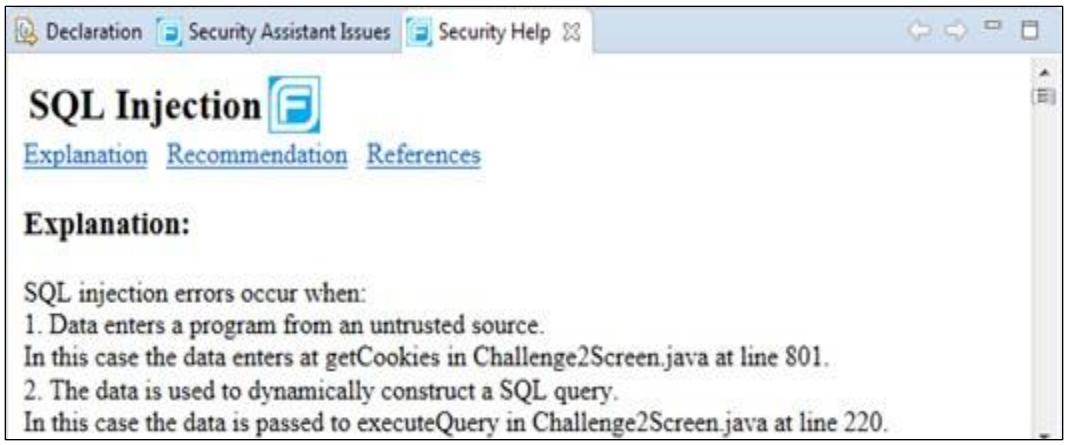
To review the security issues:

- Fortify Security Assistant for Eclipse highlights possible security issues in the code. It also tags the issue with an icon in the left border of the editor area. Pause the cursor over the highlighted code to open a tooltip that briefly describes the issue as shown in the following example:



Fortify Security Assistant for Eclipse sorts issues based on Fortify Priority Order (Critical, High, Medium, and Low).

- Click the issue to see a detailed description of it in the **Security Help** view.

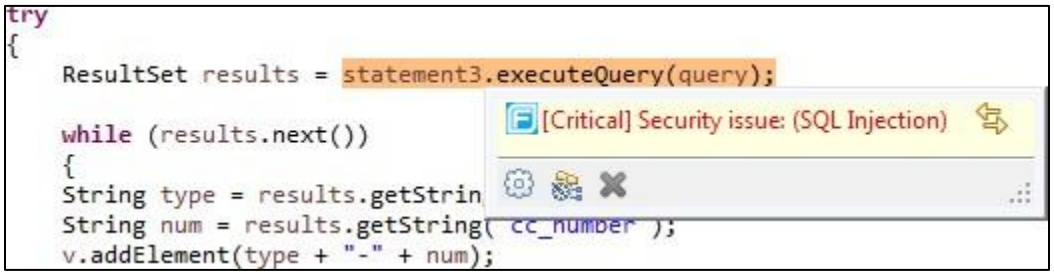


NOTE: Page through the visited descriptions in the **Security Help** view with the **Go Back** and **Go Forward** buttons.

- Select **Fortify > Open Security Issue List** to open the **Security Assistant Issues** view which lists possible issues in the file.

9.9 Working with Issues in the Code

Pause the cursor over the highlighted code to open a tooltip that displays one or more issues. Move the cursor into the Fortify Security Assistant for Eclipse tooltip or press **F2** to access additional options.



10 EIM DEVOPS

10.1 EIM DEVOPS

DevOps is a methodology and approach that pulls together developers, operations personnel and other IT professionals to automate and optimize the delivery of applications and infrastructure.

Core components

Source Code Repository – Bitbucket server (formerly Stash) from Atlassian. Bitbucket server is Atlassian's commercial Git repository. Git itself is a distributed version control system which gives developers much more flexibility when developing code via local branching and merging. Atlassian's Bitbucket server adds role based access control tied to our Active Directory as well as a streamlined web interface that enables code reviews and pull requests.

- Issue tracking System – Jira from Atlassian. An extremely flexible issue tracking system with extensive plugins and excellent support for agile project management. Jira will be used to track all issues including requirements (in the form of Agile Stories), feature requests and defects.

- Continuous Integration Server – Jenkins – Open source. The most well-known and mature product used for continuous integration. Virtually everyone that has done continuous integration has used Jenkins at some point. Version 2.0 introduced pipeline as code which encourages users to define jobs as code. The responsibility of the CI server is to check out source code when changed, run tests, orchestrate other steps such as security scans and optionally deploy the code via Ansible Tower (described below).
- Artifact Repository – Artifactory from JFrog. It is a best practice of DevOps to build source code into a deployable package once and deploy it to all target environments (i.e. Development/Test/Production). Artifactory is a universal artifact repository that will store artifacts from all software products. It provides support for virtually every programming language and also acts as a fully functional Docker Registry to support our future needs for containerization.
- Configuration Management (Infrastructure as Code) – Ansible and Ansible Tower from RedHat. Ansible's motto is "radically simple IT automation". It is an automation framework that allows configuration of Windows and Linux systems via an easy to learn YAML syntax. Ansible is agentless and therefore only requires SSH access to a Linux server and Remote PowerShell access to a Windows server. Ansible allows developers and operations engineers to define the desired configuration for a system and when run ensures that system is in the desired state. Ansible core is open source. Ansible Tower is a commercial offering from RedHat that offers role based access tied to our Active Directory, a user interface, an API for integration with other tools and integration with cloud providers for inventory management (i.e. what servers are managed via Ansible). **Figure 2** below figure shows the Continuous Integration workflow with the tools selected by AIT.

    1. Developer commits source code to Bitbucket repository
    2. Jenkins detects a change in the repository and begins building the jobs configured for this project. Builds are generally done on slave nodes to allow scaling of the Jenkins system
    3. After tests have passed, an artifact is built, versioned and pushed to Artifactory for later deployment.

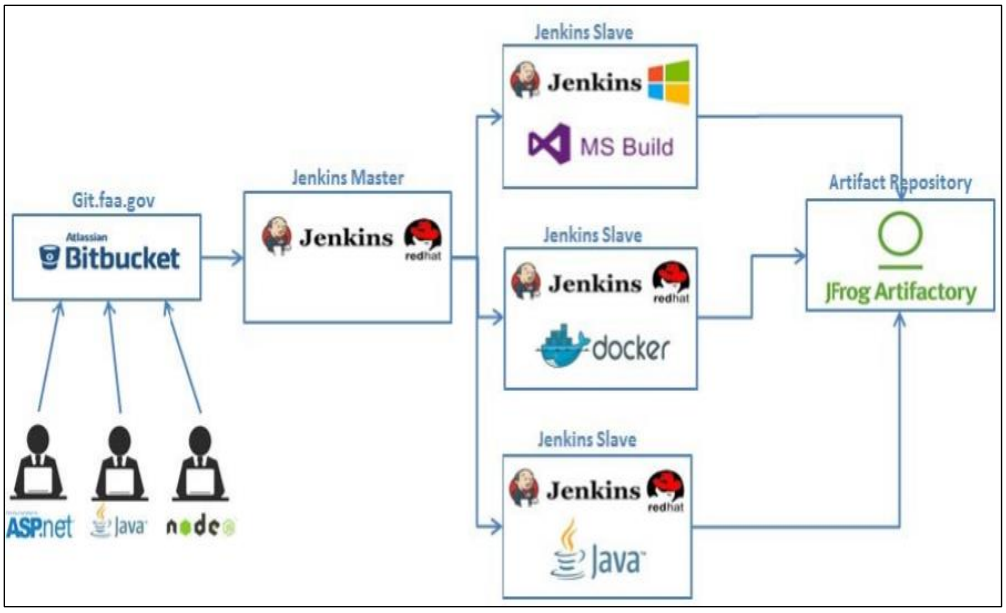


Figure 33: Continuous Integration

10.2 Deployment Flow

The following diagram depicts a typical deployment using Ansible tower and installing an artifact stored in Artifactory.

1. Playbooks are maintained in Bitbucket Repository. A project will have a playbook that will be comprised of several common “roles” (i.e. install tomcat and java and monitoring tools) and some application specific configuration. Many of these roles will install packages stored in Artifactory including the application package built in the Continuous Integration phase.
2. Ansible Tower will have a deployment job for each environment that will run the playbooks against the servers for this project.
3. Tower connects to the target servers via SSH for Linux or Remote PowerShell for Windows and runs the steps in the playbook. The credentials for accessing these servers is stored encrypted in Tower and can’t be retrieved.

NOTE: When a server is provisioned by I&O, the Ansible user is added to the server and given Administrative privileges for the server. This is a pre-requisite for using Ansible.

4. All steps of the playbook run are stored in Tower and can be reviewed.

OPTIONAL: If automatic deployments are desired, a Jenkins job can be configured to start a build via Ansible Tower.

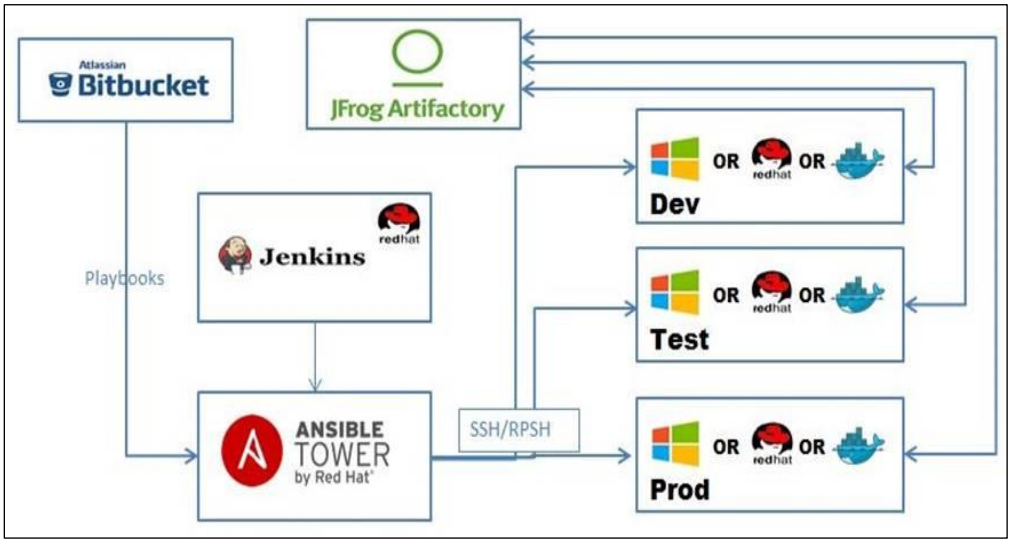


Figure 34: Deployment Applications

For more detailed please check FAA DevOps Wiki.

https://wiki.faa.gov/display/DEVOPS/FAA+DevOps

10.3 EIM Core RDS Database Recovery Procedure

1 Navigate to the RDS Snapshot management pane AWS Console.

1. Create a copy of the latest automated, making note of the name you create.

NOTE: Creating a copy of the automated snapshot ensures it does not get deleted during AWS automated snapshot cleanup. This snapshot copy becomes your “base image” for your RDS instance. If you have a manually created snapshot you prefer to use, make not of that name.
Update the snapshot_identifier variable for the appropriate instance and environment in the aim-

terraform-provisioning.

2. Commit and push the updated code to FAA Bitbucket.
3. Validate the Plan in FAA Terraform for the appropriate environment workspace.

NOTE: The Terraform plan should indicate that it will destroy the existing database and replace it with a database created from the snapshot.

4. Accept and apply the Terraform plan.
5. Verify applications are able to reconnect to the database.
6. Perform any additional activities to recover data lost between the snapshot date and the database failure date.

11 EIM Platform Tools

11.1 Fuser Flight data Aggregation

The Fuser aggregates flight data from multiple FAA SWIM sources, Airline data, and 3rd party data into a single source. Flight information is organized by individual flights (one take-off and one landing) using the Globally Unique Flight Identifier (GUFI). As each new message is received by the Fuser, the flight state is updated. Clean and accurate data is assured through the use of transformation and mediation processes which enforce business rules on the received data. The Fused dataset in EIM's Hue/Presto provides a single point for access to data from many different source data feeds. The fused dataset means users no longer need to concern themselves with implementing their own logic to correlate data from different sources, instead users can focus on their core analysis of the flight data.

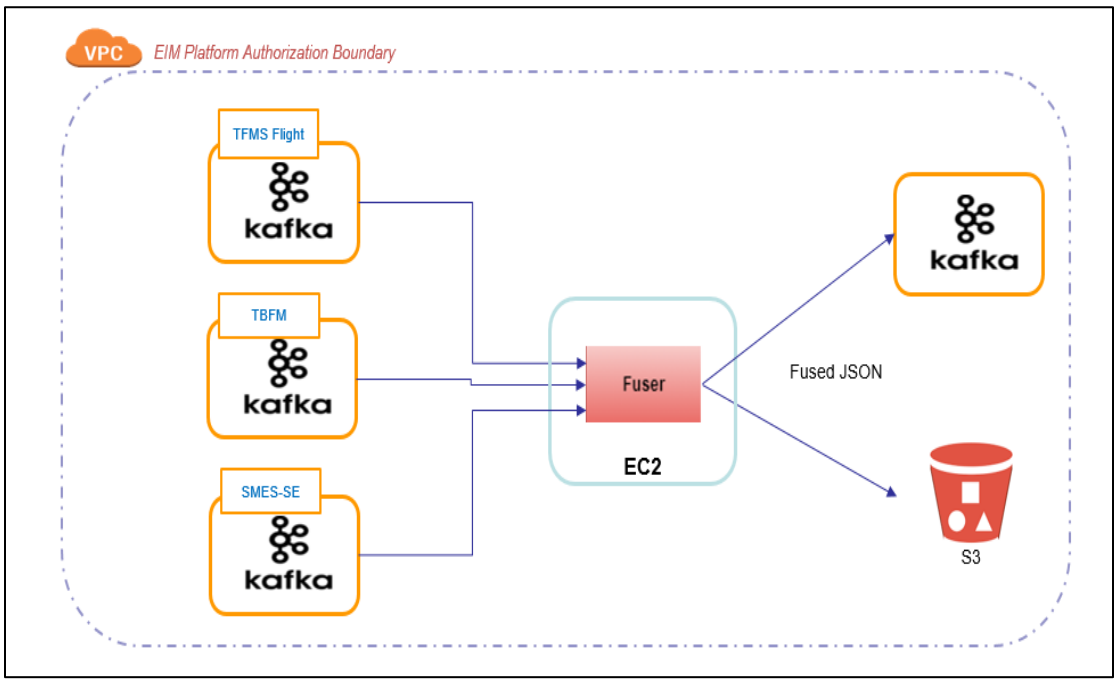


Figure 35: Fuser Aggregate From FAA SWIM Source

Fuser software components and Versions:

Fuser Application– 5.0.61
GufiService – 2.3.51

JmsRepeater – 2.1.9
Fuser-Data-Capture – 5.0.61
TmaLite – 4.3.32
TfmFlightXmlParser – 2.1.32
SmesParser – 1.0.15
LogMonitor – 1.44.22

External software required by the Fuser:

ActiveMQ - 5.15.11
Redis – 5.0.5
AwsCli – 2.1.29
Java – 11.0.10 2021-01-19 LTS

11.2 Alteryx Data Analytics Platform

Alteryx is a workflow based advanced data analytics platform to analyze and wrangle data for data scientists. Alteryx has a server-side component as well as designer component that is installed on the data workers' desktop. Alteryx is hosted in EIM FCS GovCloud environment.

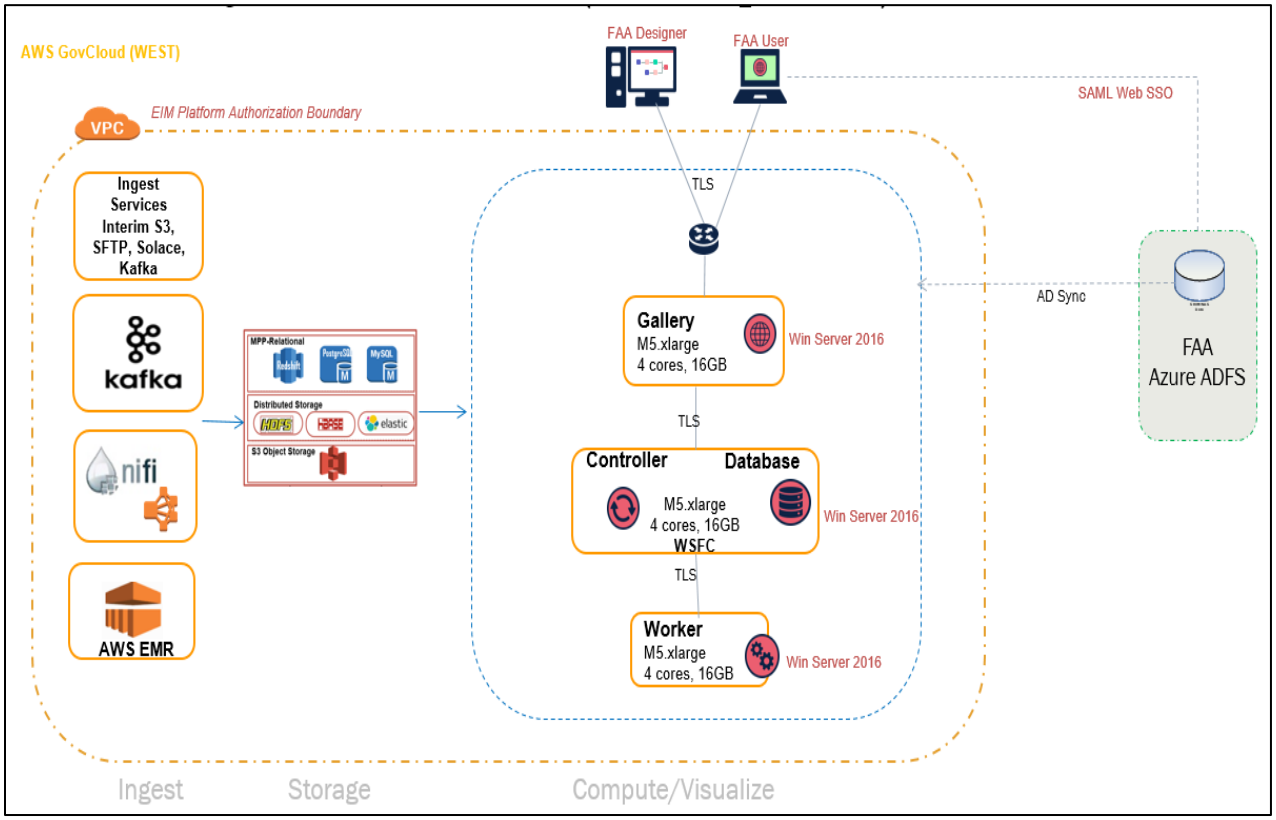


Figure 36: Alteryx Data Analytics

For accessing Alteryx in EIM Production environment, refer to the FAA Wiki (Alteryx for Data Prep)

11.3 StreamSets for Data Integration

StreamSets is a data engineering platform dedicated to building the smart data pipelines needed to power DataOps across hybrid and multi-cloud architectures. Streamsets provides a single software-as-a-service platform to design, deploy, monitor, and manage smart data pipelines at scale on any cloud and on-

premises. The StreamSets DataOps Platform is an end-to-end data engineering platform to deliver continuous data to the business and architected to solve the data engineer's below problems:

1. Minimize the ramp up time needed on new technologies and easily extend data engineering for more complex operations
2. Develop data pipelines that are resilient to the most common forms of data drift with as little intervention as possible
3. Provide operational visibility and management control over all pipelines, across hybrid and multi-cloud deployments

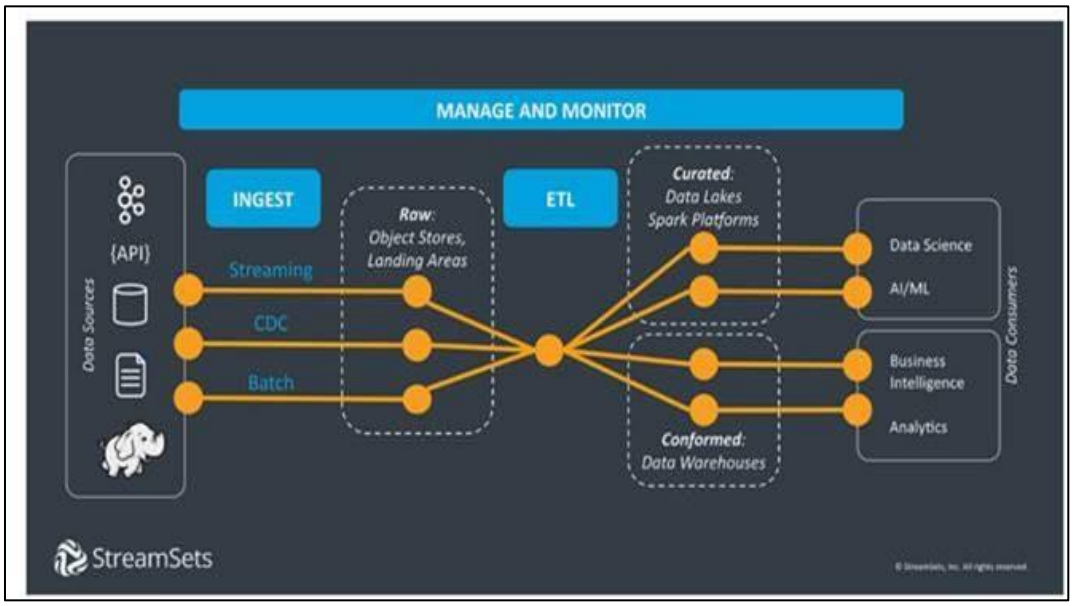


Figure 37: Streamset Pipelines

Streamset pipelines are optimized for continuous ingestion with no data latency. The user can build batch and streaming dataflows with minimal coding. Use cases of streamsets are:

1. Data migration across different data sources
2. Change data capture
3. Data ingestion services for AWS/Azure/Google cloud
4. ETL capabilities

Streamsets Data collector is an open-source data ingestion tool that can be used to design, deploy, monitor, and manage smart data pipelines at scale on any cloud and on-premises. In the EIM platform, Streamsets Data collector helps in data migration from MSSQL servers to AWS Aurora server. While provisioning a cluster with Streamsets, ECS creates a Docker container on the cluster's master node which is integrated with LDAP authentication. LDAP has an AD group for authorized users. All pipelines for data migration run within the ECS streamsets container. Streamsets data pipelines are resilient to the most common forms of data drift with as little intervention as possible and provide operational visibility and management control over data migration.

For technical details of data ingestion using StreamSets in the EIM data platform, refer to the Wiki (StreamSets)

11.4 Marquez Data Monitoring

Marquez provides a service for the collection, aggregation and visualization of a data ecosystem's metadata and maintains the provenance of how datasets are produced and consumed. Marquez also provides visibility into runtime jobs and the centralization of dataset lifecycle management. Core features of Marquez include the centralized metadata management powering dataset lineage, precise and hily dimensional data model for jobs, datasets, runs and logs. It is designed to provide a healthy data ecosystem where the data flows are tracked and available for access through a user interface. Below is the logical deployment topology of Marquez in EIM.

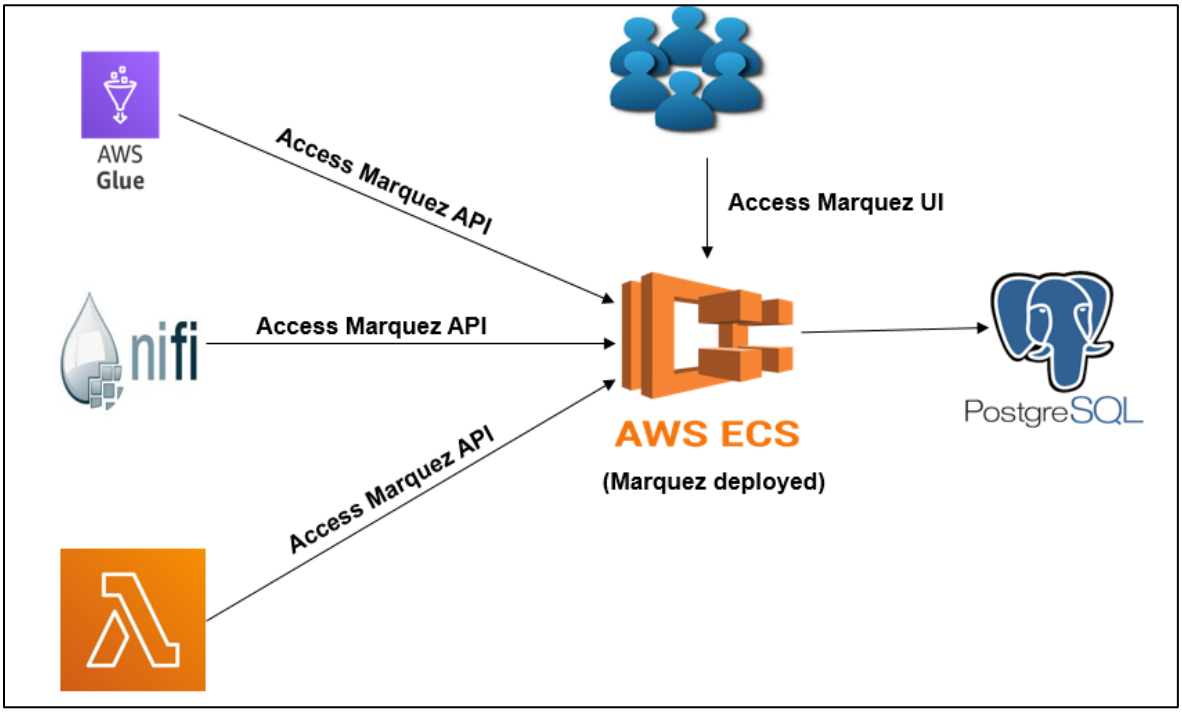


Figure 38:Deployment Topology of Marquez

Marquez enables highly flexible lineage queries across all datasets, while reliably and efficiently associating (upstream, downstream) dependencies between jobs and the datasets they produce and consume. It consists of the following system components:

- **Metadata Repository:** Stores all job and dataset metadata, including a complete history of job runs and job-level statistics (i.e., total runs, average runtimes, Success/Failures etc.,)
- **Metadata API:** RESTful API enabling a diverse set of clients to begin collecting metadata around dataset production and consumption.
- **Metadata UI:** Used for dataflow discovery, connecting multiple datasets and exploring their dependency graph.

For more technical details of this tool, refer to the Wiki guide (Marquez API)

Appendix A: Data Assets on EIM Platform

The EIM Platform provide ingest data feeds for FAA users with current information from the following sources.

ac3code	proj-asaic-map
ac4code	proj-asaic-weather
acdesnfleet	proj-eoc
ads-b-coverage	proj-famm
aeronav-supplements	proj-opsnetrv-etap
aircraft	proj-ospc-aspm
aircraft-make-model	proj-ospc-metrics
airmen-cert	proj-ssm-airops
asde-x	proj-ssm-mor-eor
atc-operational-contingency-plans	proj-ssm-output
atc-pubs	proj-ssm-re
cifp	proj-uasra-io
eim-metadata	radar-sap
eon	radar-site-param
eon-meta	radar-site-survey
eram	rap-ruc
eram-dau	sba33
eram-docs-ti	sbss
eram-full	sdr
eram-nasd	sdr-meta
eram-systemarchive	sfd
fsep	sfdps
helipadlz	srtm
icao-training-module	stdds-apds
ifos	stdds-ismc
ifr-charts	stdds-itws
metar	stdds-smes
mil-train-routes	stdds-tais
msad-burl	stdds-tdes
msad-seat	terminal-charts
nfdc	tfbm
nfdc-airspace	tfmm-adapt
nfdc-saa	tfms-config
nop20a-center	tfms-flight
nop20a-mearts	tfms-flow
nop20a-terminal	tfms-weather
nop2-b-tais	ttfs
ntsb	uas-cov-maps
obstacles	vfr-charts
pdars-airborne-holdings	voice
pdars-go-arounds	wx-overlay
pireps	wx-stations
proj-aria-airborne	

For all data feeds in EIM with relative formats can be seen here Data Feeds in EIM.

For more information on the EIM Platform data sets with metadata and catalog information is available for download here: EIM Metadata Catalog.

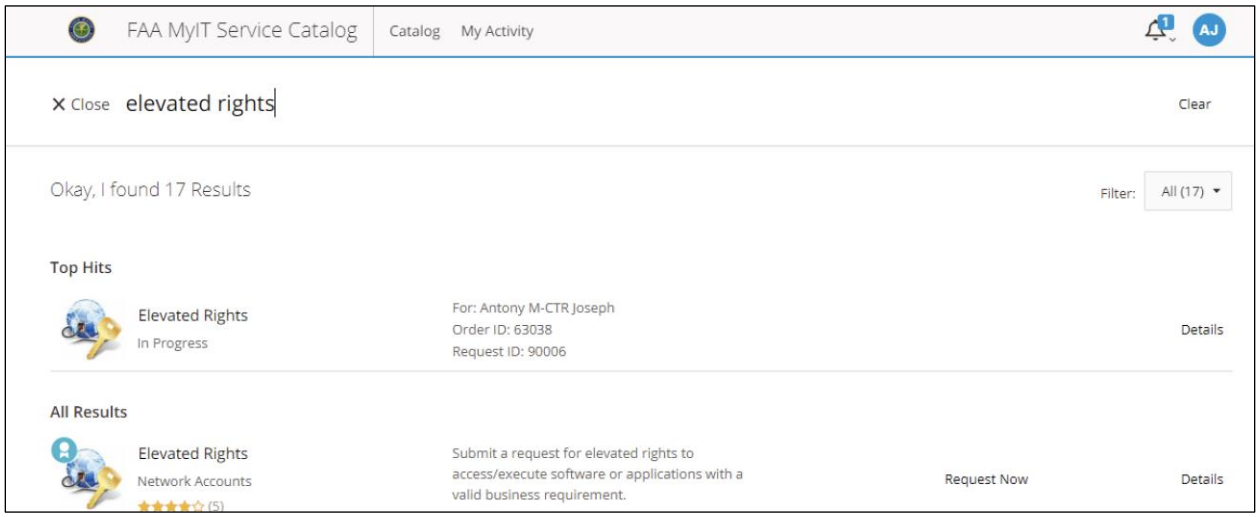
Link of the Data Ingest project documentation are located below: Data Ingest Process.

Appendix B: Abbreviations

Acronym	Definition
AD	Active Directory
AMI	Amazon Machine Image
API	Application Program Interface
CO	computer object
EIM	Enterprise Information Management
FCS	FAA Cloud Service
GCC	Government Community Cloud
HDFS	Hadoop Distributed File System
IaaS	Infrastructure-as-a-Service
IOC	Initial Operating Capability
KMS	Key Management System
NAS	National Airspace System
NER	NAS Enterprise Repository
NiFi	Niagara Files
NIST	National Institute for Standards and Technology
OARS	Operational Analysis and Reporting System
PaaS	Platform-as-a-Service
SFTP	Secure File Transfer Protocol
SPAS	Safety Performance Analysis System
SSH	Secure Shell
SWIM	System Wide Information Management
TLS	Transport Level Security

Appendix C: Elevated Access Request

An elevated access to the government furnished equipment (GFE) hardware, a developer must be logged into the internal FAA network with a FAA PIV smartcard at https://myit.faa.gov/ . Search for elevated rights under the FAA MyIT Service Catalog.



- Once Elevated Rights link is opened, click **Request Now**.
- A window will open, enter the details in the **Provide the details of your request** box with the reasoning why administrative right are needed with the FAA position title.
- Click **Submit**

Appendix D: How to work with EIM Platform NiFi

This guide provides the following information:

- Steps for getting started with NiFi
- Creating a NiFi workflow
- Deploying NiFi through Ansible

Getting Started with NiFi

Start the NiFi service

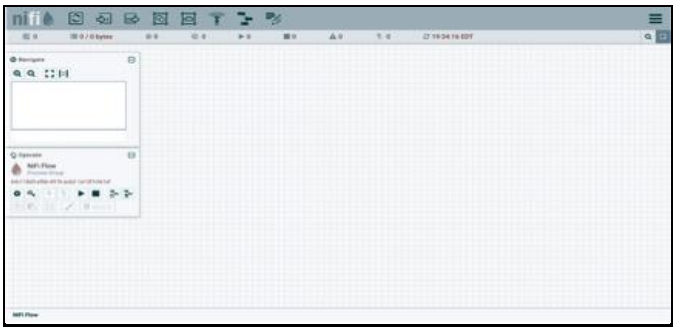
```
Sudo service NiFi start
```

To access NiFi (UI), open the web browser to the address:

- http://<cluster_url>:8081/nifi

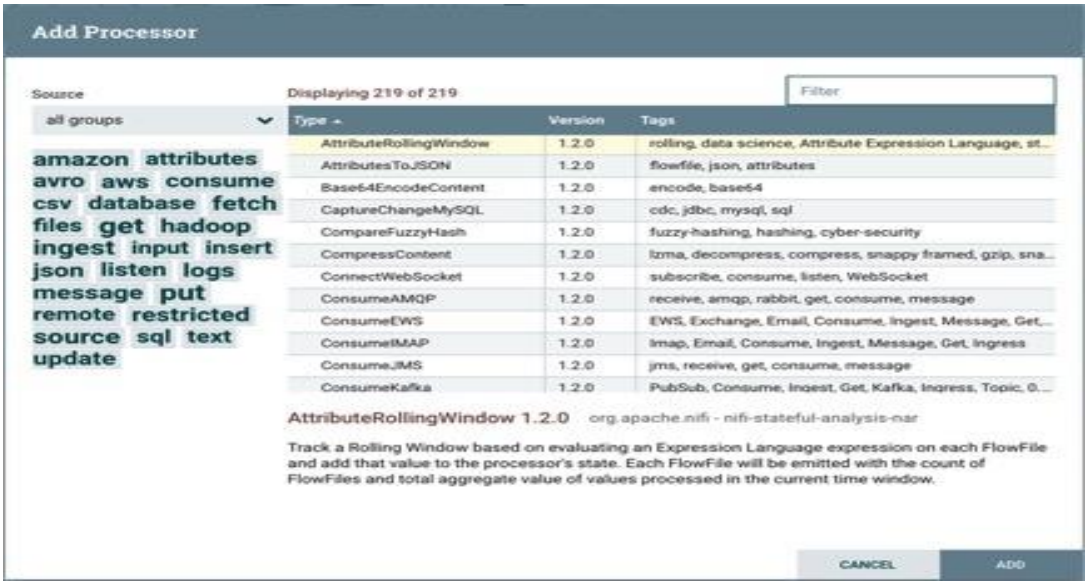
Create a NiFi Workflow

NiFi Canvas:



Click and Drag the Processor icon onto the canvas 

Add Processor dialog:



Select the **GetFile** processor and click **Add**.

Right click the **GetFile** processor:

- Choose **Configure** from the menu.
- Select the **Properties** tab.
- Set the **Directory** property to the location of the batch file(s).

Connections:

- Add a **PutHDFS** processor to archive the batch file in HDFS.
- Hover over the **GetFile** processor until connection icon appear.
- Drag the icon from the **GetFile** processor to the **PutHDFS** processor.

Right click the **PutHDFS** processor:

- Choose **Configure** from the menu.
- Select the **Properties** tab.
- Set the **Directory** property to the HDFS archive location.

Deploying NiFi through Ansible

The EIM Platform takes advantage of Apache Niagara Files (NiFi) as a simple event-processing platform that is designed to automate the flow of data between disparate data sources and systems.

This repository contains the resources for the EIM NiFi Tutorial. Specifically, below will find:

- Ansible playbooks for configuration, backup, and deployment.
- flow.xml.gz (an example containing a serialized NiFi Dataflow).
- NiFi templates (a Dataflow sub-component, see below for description).
- Java source code, demonstrating how to extend and customize NiFi.

Ansible playbooks in this repository will demonstrate the following operations:

1. Backing up the NiFi Dataflow
2. Initial cluster configuration (e.g. creating HDFS directories)
3. Deploying the NiFi Dataflow

For more generic information about how a NiFi Dataflow works from the NiFi Admin, it is recommended in the terminology section of the Apache NiFi User Guide located https://nifi.apache.org/docs/nifi-docs/html/user-guide.html.

Custom NiFi Processors

Custom NiFi processors are written in the Java programming language. To create custom ingest processors follow the example of this class:

```
eim-nifi/src/main/java/gov/faa/eim/nifi/RandomSamplingProcessor.java
```

1. When the Java classes are written, update the following Meta file:

```
eim-nifi/src/main/resources/META-INF/services/org.apache.nifi.processor.Processor
```

2. A single processor will exist in this repository. The fully qualified class name is:

```
gov.faa.eim.nifi.RandomSamplingProcessor
```

3. Notice the entry in the previously mentioned file. If new classes are for New1Processor and New2Processor, update the file to include these in the list:

```
gov.faa.eim.nifi.RandomSamplingProcessor
```

```
gov.faa.eim.nifi.New1Processor
gov.faa.eim.nifi.New2Processor
```

4. After writing the Java code and the Meta file is updated, the project is prepared to be compiled. The Maven build tool is required for this step. Compile the project by:

```
mvn clean package
```

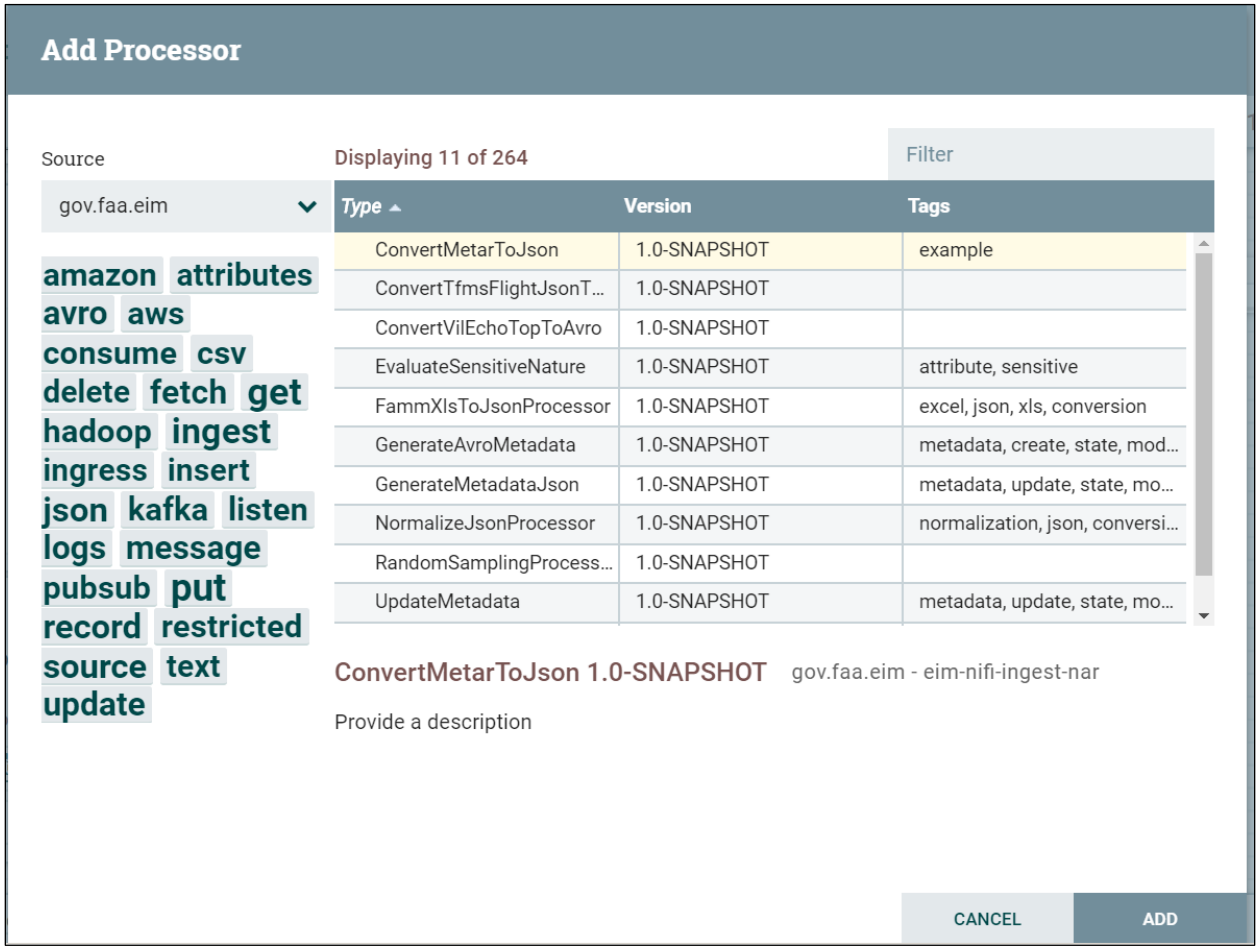
1. Install and configure Java 1.8 or later
2. Configure Gradle 4+ or Maven 3.2+
3. Configure Spring Tool Suite (STS) or IntelliJ IDEA
4. Create a simple web application - in non FCS environment
5. Create a web controller
6. Create an application class

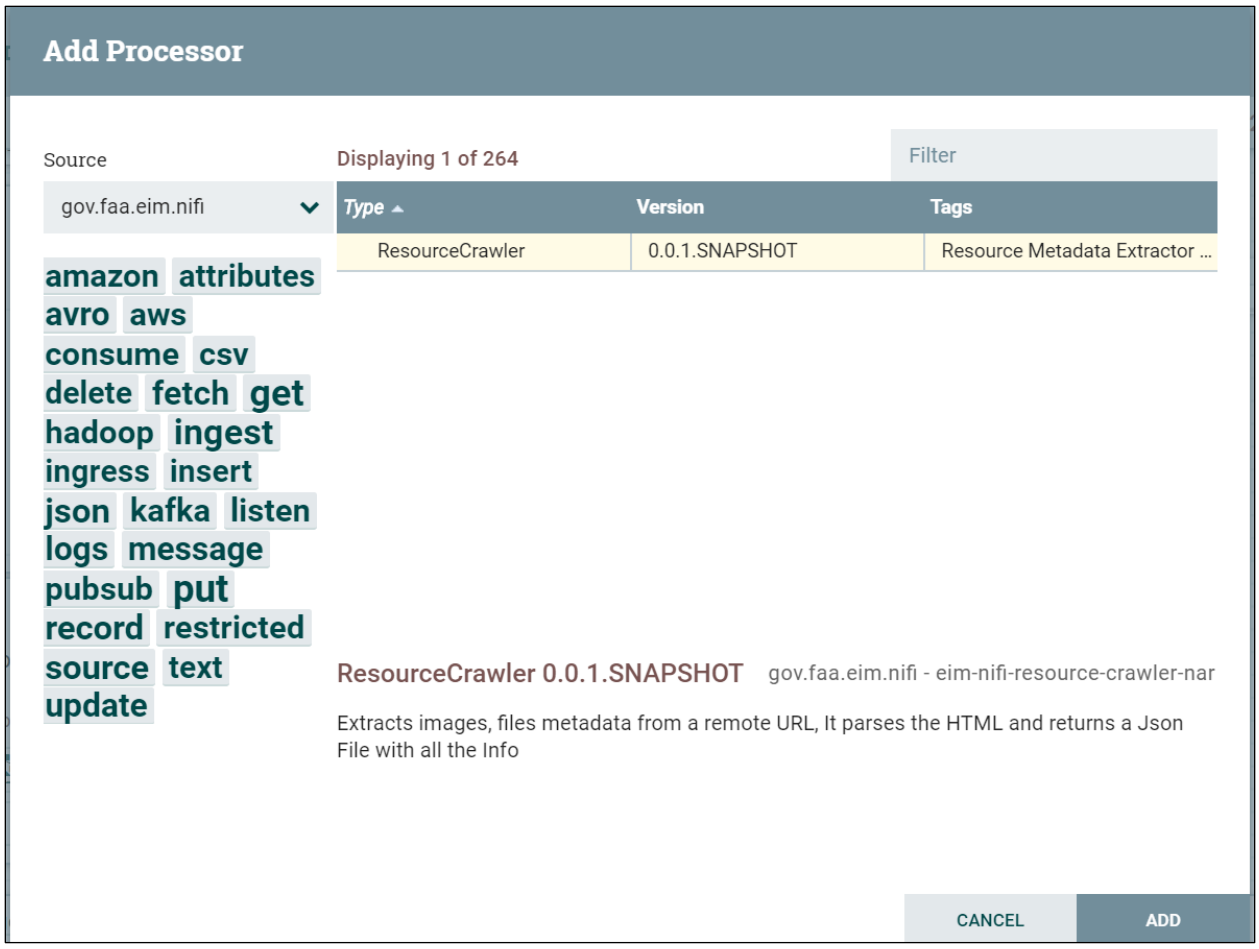
This will yield a NiFi NAR file, located at eim-nifi/target/eim-nifi-1.0-SNAPSHOT.nar. NAR files with custom processors can be used to construct data flows and templates. To deploy a NiFi NAR file, invoke the following playbook:

```
ansible-playbook -i _eim-ingest-inventory -u ec2-user 03_deploy_nifi_nar.yml -vvv
```

Custom NiFi processors is available on EIM Platform.

The Apache NiFi Developers Guide is located on the Apache site for additional assistance: https://nifi.apache.org/docs/nifi-docs/html/developer-guide.html





All of the properties support the expression language syntax. Only the following properties are mandatory

Id ==> UUID of the Metadata entry

Environment ==> In which Environment is the processor used. Supported environments are "Non-FCS", "dev", "stage" and "prod."

Feed Source ==> Name of the Feed Source.

There is also a NiFi processor (GenerateAvroMetadata) for generating AVRO metadata which will be used by the Hive metadata table.

Steps for deploying new NiFi workflows to EIM Platform (DEV)

1. The development work need to be completed on local or personal dev environments and required configurations need to be setup as properties.
2. Nar files and NiFi Templates will be stored in Bitbucket repository. This may be committed to the EIM Platform repo or program/project specific repo will be provided. (More details to be added)
3. There should be no hardcoded values in the NiFi processors
4. Use the NiFi.property descriptor to create properties values to be used for configuring into the processors vs hard coding values. The property values need to be set as per environments.
5. The environment values needs to be provided as property files and uploaded into Bitbucket.

6. NiFi Workflow changes to add or update will be done via using Artifactory, Jenkins and Ansible tower.
7. Special note needs to be taken when creating custom processors, this portion of code needs to be uploaded via Artifactory. Import the code into the dev environment.

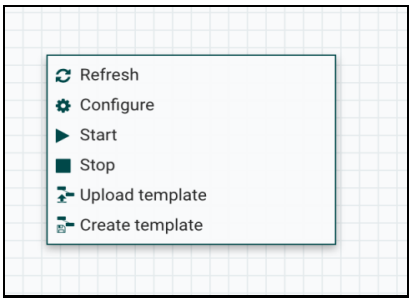
Deploying Templates

A NiFi template is a collection of NiFi building blocks (processors, funnels, input/output ports, process groups, and remote process groups). A template may rely on custom Java code (as described above, deployed via the NAR file).

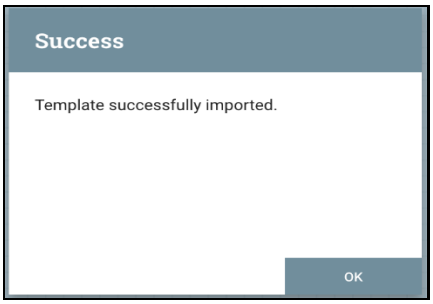
When the same logic is needed several times in a Dataflow repeat the configuration. A template allows building blocks to be grouped together and save the configuration for re-use.

NOTE: The template can then be dragged onto the canvas or can be exported as an XML file and shared with others.

1. To deploy a NiFi template, right click on the NiFi canvas. Right clicking on a blank area within the NiFi GUI will yield the following menu:



2. From this menu, choose "Upload template". From the "Upload Template" screen, select the magnifying glass "Select Template" and browse for the XML file on the local machine. Next, click **Upload**. This should yield the following dialog:



3. Next, click and drag the **Add Template** button (highlighted below) onto the canvas:

4. Ensure the "Pull from Twitter Garden Hose" template has been selected in the drop-down.
5. Click **Add**.

This should yield three (3) Processors and one (1) Output Port on the canvas. Delete the Output Port.

6. Configure the Processor named "Grab Garden Hose." Right-click on it and select **Configure**. Navigate to the "Properties" tab, and edit the Consumer Key, Consumer Secret, Access Token, and Access Token Secret.
7. Add a **PutHDFS** Processor to the canvas. Right click on the processor and then select the **Properties** tab. There, enter the following in the Hadoop Configuration Resources field: /etc/hadoop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml
8. Finally, start the data flow by clicking on a blank area in the canvas and selecting "Start" from the menu. To stop the data flow, click on a blank area on the canvas and choose "Stop".

Deploying a Complete Dataflow

An entire NiFi dataflow can be deployed in a single step; with the condition that all dependent NAR files have been deployed.

Finally, perform the NiFi ingest deployment using the below playbook:

```
ansible-playbook -i _eim-ingest-inventory -u ec2-user 04_deploy_nifi_flow.yml -vvv
```

Build and Deploy of NiFi NAR Files:

1. Checkout FAA-EIM Repository from BitBucket
2. Build the branch of FAA-EIM repo which a developer want to deploy
3. As FAA-EIM repo has dependency on EIM-apps and EIM-SWIM the developer need to build EIM-apps and EIM-SWIM first
4. Check out EIM-SWIM repo and build and install
5. Checkout EIM-apps repo and build and install
6. Finally build FAA-EIM
7. EIM-NiFi-ingest-NAR module of FAA-EIM should generate the NAR file
8. Deploy the NAR file to NiFi server
9. Restart NiFi Server and application tiers.

On EIM Platform NiFi is used to orchestrate workflows for moving data from source to destination.

Apache NiFi supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic. It has a simple web-based user interface that is highly configurable and is secure. For more information on NiFi, see: https://nifi.apache.org/.

NiFi workflows will be deployed on the EIM Platform using Ansible playbooks, property files and NAR files.

The Ansible playbooks configure the NiFi Nodes, deploy the dataflow templates and configure the data flow based on the environment using the property files.

The NAR files contains any custom code developed specifically for the EIM Platform such as custom processors.

SWIM NiFi Flows: Below is an example of a NiFi workflow for SWIM data feed:

```
output/${now():format('yyyy_MM_dd')}/${eim_type}/${eim_stype}/${eim_id}-${now():format('yyyy-MM-dd_HH:mm:ss')}/
```

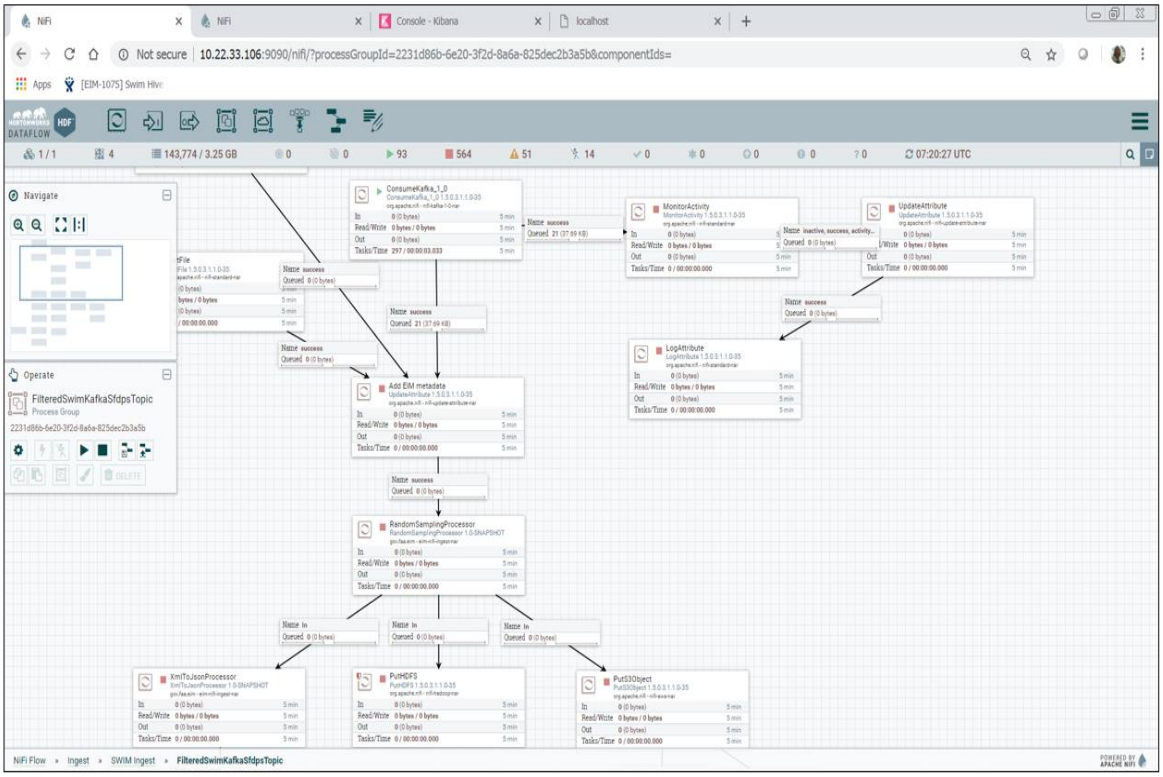


Figure 39: NiFi Workflow for a SWIM Data Feed

Based on the workflow orchestration for the SWIM data, it is moved from the Kafka topic to S3.

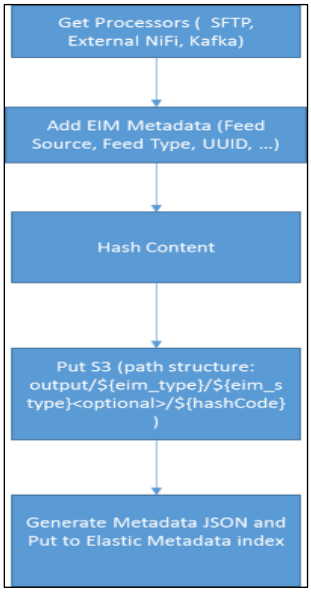


Figure 40: EIM Platform Ingest Flow Pattern Workflow

Alternate Ingest Workflow Pattern – The workflow pattern provided demonstrates the movement of the data content into S3.

Swim uses the alternate ingest workflow pattern. See below an example with details of the workflow components.

Appendix E: Data Ingestion using Secure File Transfers

EIM uses four types of data ingestion design patterns to onboard data from external sources to EIM Data Lake in GovCloud. Each design includes retrieving and transferring data from specific source – internal to FAA on-prem or from external sources. The streaming ingestion using Kafka and NiFi has been explained in the previous sections.

This section explains the two options for batch file transfers.

1. AWS CLI based file transfer (*preferred option*)
2. Secure file transfer using Edge Node

File transfers to GovCloud using AWS CLI

Batch file transfers using AWS Command Line Interface (CLI) is the most preferred approach to transfer batch files to EIM platform. As shown in the diagram above, CLI is installed on an on-prem server or Windows desktop from where the source data is planned to be ingested. Uploaded files are temporarily stored on an 'interim' S3 upload bucket folder before they are finally 'ingested to their final resting place in feed specific S3 bucket with appropriate security controls and indexes. The development teams will need to coordinate with the EIM data ingest team to setup the below as preparation for setting up the ingest pipeline.

1. Request Access Key and Secret Key for IAM user
2. Setup 'interim' S3 upload bucket folder
3. Set up 'ingest' bucket

Development teams are also responsible to provide the NiFi flows to transfer and ingest data from interim bucket to the ingest bucket and any DevOps scripts and configurations to deploy the code to the platform.

The instructions to download CLI and install have been provided in the section below.

Appendix F: Installing and Connecting to AWS S3 CLI

Install AWS CLI

1. Download and install the AWS CLI installer for Windows 64 bit from https://awscli.amazonaws.com/AWSCLIV2.msi. You can validate the installation using the command prompt to execute "aws –version", response should look like below:

C:\> **aws --version**

aws-cli/2.0.0 Python/3.7.4 Windows/10 botocore/2.0.0

Configure AWS CLI

1. In the command prompt, execute "aws configure" to configure the AWS CLI, use the access and secret keys assigned to you. These keys must be requested from the administrator of your EIM account.

$ **aws configure**

2. AWS Access Key ID [None]: *xxxxxxxxxxxxxx*
3. AWS Secret Access Key [None]: *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*
4. Default region name [None]: *us-gov-west-1*
5. Default output format [None]: *json*
6. You can configure AWS CLI for more than one account (EIM and ANG) by using the profile identifier. Example below:

aws configure --profile angs3user

AWS Access Key ID [None]: ***

AWS Secret Access Key [None]: ***

Default region name [None]: ***us-gov-west-1***

Default output format [None]: ***json***

Example List files in bucket
aws s3 ls s3://angdev1-ng-labs-ingest- --profile angs3user

Secure File transfers thru Edge node

The Secure File Transfer Protocol (SFTP) server resides on the FCS AWS cloud inside the EIM Platform boundary and will require credentials to access remotely from the FAA n-premises (on-prem) database servers or GFEs. SFTP will provide FAA user(s)/team(s) the ability to move large quantities of data from an on-prem data store(s) to AWS S3 in the GovCloud for archival and retrieval.

Data is placed on the SFTP server, automatically gets transferred to an S3 'upload bucket' using cron job scripts. EIM NiFi workflow is needed to be configured to ingest the data from 'S3 Upload Bucket' to the appropriate S3 bucket in EIM. The workflow will be maintained by EIM Platform. Development teams are responsible for using an SFTP client of their choice (e.g. FileZilla) on the source of transfer. They also need to develop the NiFi flows as per EIM standard once the ingest bucket is configured for the incoming data feed and request access to EIM platform.

Appendix G: Managing Secrets using SSM Parameter Store

EIM secures database credentials such as passwords, DB strings etc. using the Parameter Store provided by AWS Systems Manager. AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, Amazon Machine Image (AMI) IDs, and license codes as parameter values. You can store values as plain text or encrypted data. You can reference Systems Manager parameters in your scripts, commands, SSM documents, and configuration and automation workflows by using the unique name that you specified when you created the parameter.

The section illustrates how to create the Parameters in the SSM using CLI and Ansible as well as using the parameters to retrieve the database passwords to access the PostgreSQL RDS instance in Python code.

Accessing SSM Parameter Store:

- Roles required to run below: AmazonSSMReadOnlyAccess
- All parameter store parameters should follow this format:
  - /<environment>/<service>/<application>/[<descriptive>-]<user>

Using AWS Console:

https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html

1. Login to System manager
2. Click Application Management → Parameter Store
3. Button 'Create parameter'
4. Fill in 'Parameter details'
   a. Name - <eim-environment>/<service>/<application>/<distinct account name describing what this is for>/
   b. Description – (Optional) Describe this parameter
   c. Tier – Standard
   d. Type – SecureString (For encryption)
   e. Data type – text
   f. Value - <the unencrypted password>
   g. Tags – (Optional)
5. Button 'Create parameter'

Using AWS CLI:

https://docs.aws.amazon.com/cli/latest/reference/ssm/put-parameter.html

1. Go to a server that has or can install aws cli
   a. sudo pip install AWSCLI
2. Encrypt password:

```
aws ssm put-parameter \
--name <eim-environment>/<service>/<application>/<distinct account name describing what this is for>/ \
--description "(Optional) Describe this parameter " \
--value "<the unencrypted password>" \
 --type SecureString
```

From Ansible:

```
- name: Task a - Get aws ssm encrypted passwords to be used with
project database connections
    command: >
      aws ssm get-parameters --name {{ user_item.parameterstore
}} --with-decryption --query "Parameters[*].{Value:Value}"
        --region {{ aws.arn.region }}
    register: output
    no_log: true

  - name: Task b - Create Role for all users
    # Sql equivalent: create role {{ item.name }} password '{{
item.pass }}' LOGIN INHERIT;
    # Sql equivalent: grant connect on database {{
rds.postgres.cloud.dbname }} to {{ item.name }};
    postgresql_user:
      db: "{{ rds.postgres.cloud.dbname }}"
      login_host: "{{ rds.postgres.cloud.dbserver }}"
      login_password: "{{ rds.postgres.cloud.dbadminpass }}"
      login_user: "{{ rds.postgres.cloud.dbadminuser }}"
      port: "{{ rds.postgres.cloud.dbserverport }}"
      ssl_mode: "verify-ca"
      encrypted: "yes"
      name: "{{ user_item.user }}"
      password: "{{ output.stdout | from_json |
json_query('[0].Value') }}"
      role_attr_flags: "LOGIN,INHERIT"
      priv: "CONNECT"
    no_log: true
```

Access Postgres via Python:

Roles required to run below: **AmazonSSMReadOnlyAccess**

../config/database.ini

```
[PostgreSQL]
user = <app user>
host = dev-uuuu.xxxxxxx.us-gov-west-1.rds.amazonaws.com
database = feeds
port = 5432
```

==========

config.py:

```
#!/usr/bin/python
from configparser import ConfigParser

def config(filename='database.ini', section='postgresql'):
    # create a parser
    parser = ConfigParser()
    # read config file
```

```
    parser.read(filename)

    # get section, default to postgresql
    db = {}
  if parser.has_section(section):
        params = parser.items(section)
        for param in params:
            db[param[0]] = param[1]
  else:
    raise Exception('Section {0} not found in the {1}
file'.format(section, filename))
  return db
```

python.py:

```
import psycopg2
import boto3
from config import config

iniFile='./config/connect.ini'

def getConnPass(ssm_parameter_path):
    """
    Get the stored password for the connection account to the
database.
    """
    try:
        passwd = ssmClient.get_parameter(Name=ssm_parameter_path,
WithDecryption=True)['Parameter']['Value']
    except:
        print("Encountered an error loading config from SSM.")
        traceback.print_exc()
        passwd = None
    finally:
        return passwd

def connect(params):
    """ Connect to the database server """
    conn = None
    try:
        # Add encrypted password
        params['password'] = getConnPass(full_config_path)

        # connect to the PostgreSQL server
        conn = psycopg2.connect(**params)

    except (Exception, psycopg2.DatabaseError) as error:
        print('Error during database open: ', error)
        traceback.print_exc()
    finally:
        return conn

def disconnect(conn):
    try:
```

```
        # Check if open connection
        if conn.closed == 0:
            # close the communication with the PostgreSQL
            conn.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print('Error during database close: ', error)
    finally:
        if conn.closed == 0:
            print('Database connection did not close! Trying again.')
            conn.close()
        return None

def get_column_list_string(conn, db, table):
    """ Connect to database to retrieve column list """

    query = "select column_name from information_schema.columns where
table_catalog = '" + db + "' and table_name = '" + \
            table + "' order by ordinal_position"

    # create a cursor
    cur = conn.cursor()

    # execute query
    cur.execute(query)

    # Get the output
    columns = cur.fetchall()

    # close the database cursor
    cur.close()

    col_list = None
    # Build comma separated list of column names in order
    for i in columns:
        if col_list is None:
            col_list = ''.join(i)
        else:
            col_list = col_list + ',' + ''.join(i)

    return col_list


# Initialize boto3 client for ssm at global scope for connection reuse
ssmClient = boto3.client('ssm', region_name=region)

def main():

if __name__ == "__main__":
    main()

    pgres_parms = {}
    pgresconn = None
```

```
# read connection parameters
      params = config(iniFile)
      params['host'] = db_endpoint
      # Add encrypted password
      params['password'] = getConnPass(full_config_path)

    # Connect to db
    pgresconn = connect(pgres_parms)

    # Do some queries against Postgres like this:
    # get the list of column names from the to database therefor the
developer only insert them. 1 to 1 match in names

    disconnect(conn)
```

Appendix H: Athena JDBC Connection – Python

STEPS:

conda install – c conda-forge jpype1

pip install PyAthenaJDBC

Sample Connections

```
from pyathenajdbc import connect
import pandas as pd
#Obtain the keys from Privacera portal for your FAA user identity
conn = connect(access_key=**********,
               secret_key='**********',
               s3_staging_dir='s3://aws-target1/Unsaved/2020/03/19/',
               region_name='us-east-1')  # optional, as used by JPype
df = pd.read_sql("SELECT * FROM test_db.test_tb LIMIT 10", conn)
print(df)
```

As Pandas DataFrame:

```
import contextlib
from pyathenajdbc import connect
from pyathenajdbc.util import as_pandas
with contextlib.closing(
      connect (s3_staging_dir='s3://YOUR_S3_BUCKET/path/to/'
            region_name='us-west-2'))) as conn:
   with conn.cursor() as cursor:
      cursor.execute("""
      SELECT * FROM many_rows
      """)
      df = as_pandas(cursor)
print(df.describe())
```

Appendix I: How to Provision FAA CA Certificates

Create San.cnf

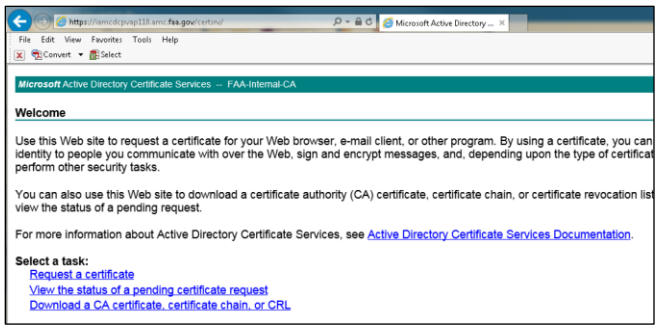
```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext
[ req_distinguished_name ]
countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name (full name)
localityName = Locality Name (eg, city)
organizationName = Organization Name (eg, company)
commonName = Common Name (e.g. server FQDN or YOUR name)
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1 = eams.faa.gov
```

Run OpenSSL to create sslcert.csr

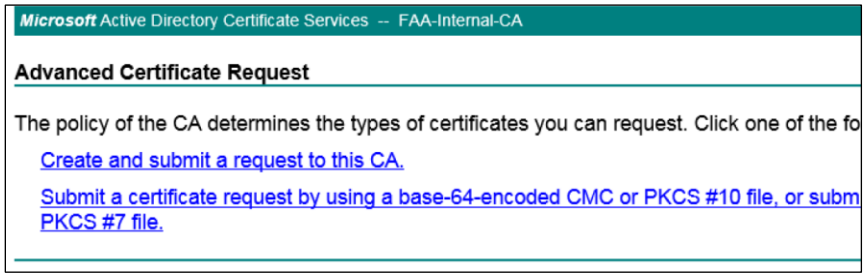
```
openssl req -out sslcert.csr -newkey rsa:2048 -nodes -keyout private.key -config san.cnf
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'private.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:US
State or Province Name (full name) []:NJ
Locality Name (eg, city) []:Atlantic City
Organization Name (eg, company) []:FAA
Common Name (e.g. server FQDN or YOUR name) []:eams.faa.gov
```

Request Certificate

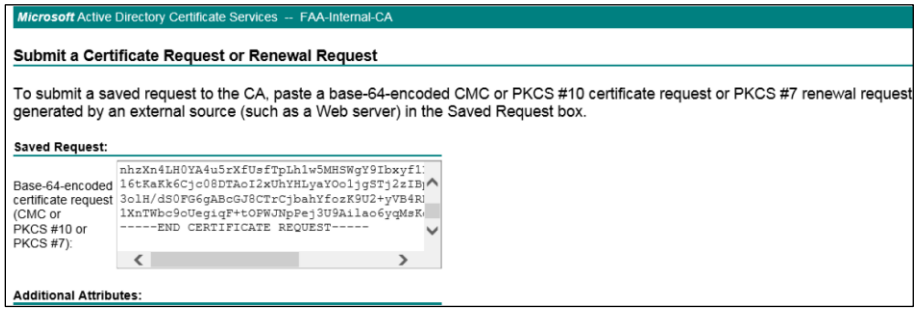
1. Navigate to: https://iamcdcpvap110.amc.faa.gov/certsrv/



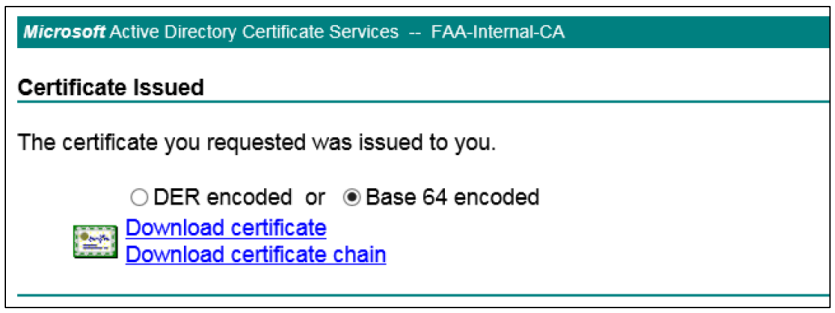
2. Advanced certificate request and submit certificate request



3. Paste contents of sslsert.csr



4. Download Base 64



5. If copying to server via playbook, be sure to encrypt the cert and key:

ansible-vault encrypt <cert> <key>

Automated scripts for the CA request

https://git.faa.gov/projects/EIM-EC/repos/eim-certs/browse/eim-core/eim-core-dev/cert/dev

1. Update ssl.conf with your desired settings
2. Run https://git.faa.gov/projects/EIM-EC/repos/eim-certs/browse/eim-core/eim-core dev/cert/dev/keyandcsr.sh
3. Post the resulting file to the FAA cert server https://iamcdcpvap110.amc.faa.gov/certsrv/
4. Download the DER results
5. Use this script to convert the p7b key to pem format

https://git.faa.gov/projects/EIM-EC/repos/eim-certs/browse/eim-core/eim-core-dev/cert/dev/p7b2pem.sh

6. Add the TLS bundle to a SSL Credential in Ansible Tower
7. Use this playbook to redeploy to instances: https://ansible.faa.gov/#/templates/job_template/1815

Appendix J: Connecting to backend database (Jump Host Configuration)

1.) Add your FAA domain user to a local user, to a Linux server that has access to your RDS instance (helpdesk@faa.gov can facilitate the "Add User" ansbile script).
2.) Open the PuTTY Configuration
3.) Under the Session menu, enter the IP address of your server, the SSH Port (port 22), type in the name of the session to save, and click the **Save** button.

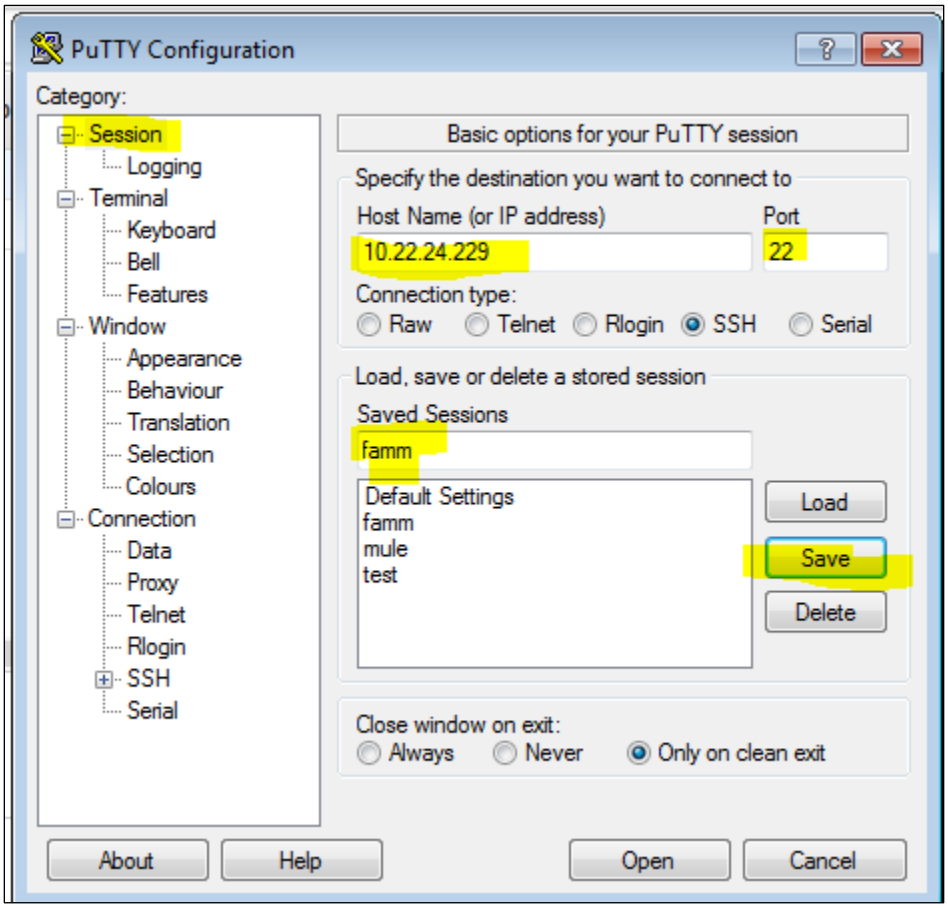


Figure 41: PuTTY Configuration Session

4.) Open the Connection>SSH>Auth Menu, and enable the following:
   a. Attempt "keyboard-interactive" auth (SSH-2)
   b. Allow agent forwarding.
   c. Allow attempted changes of username in SSH-2.

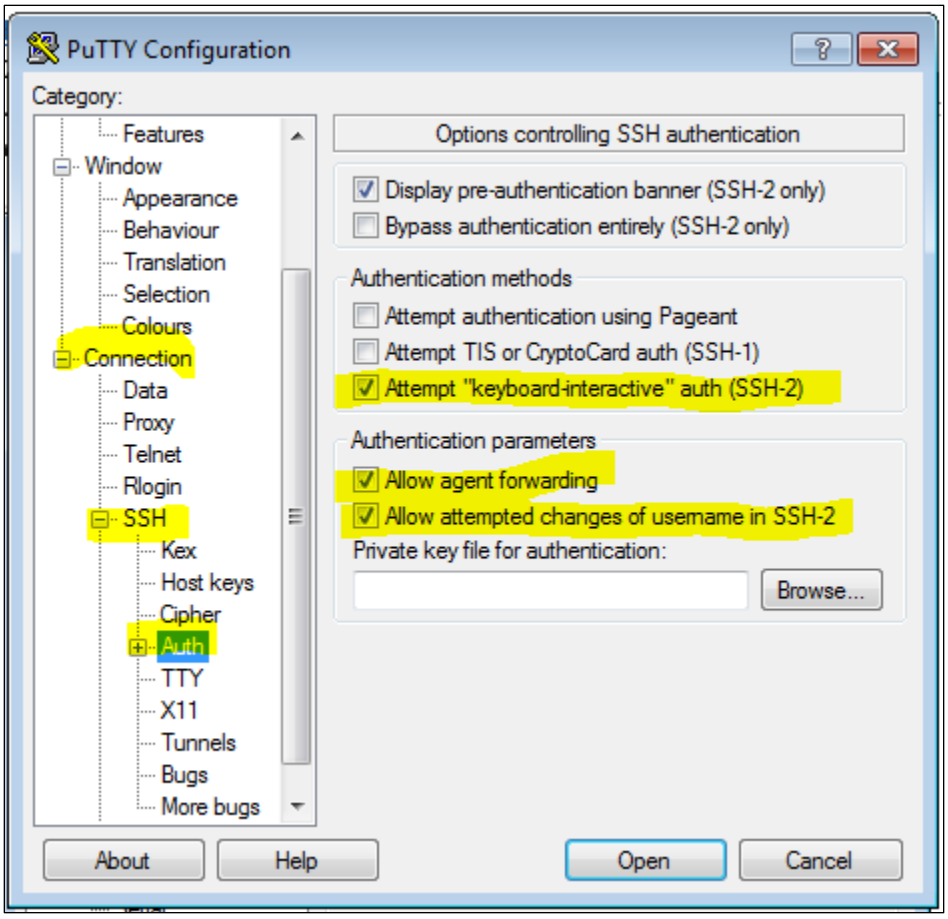


Figure 42: Putty Configuration SSH-2

5.) Navigate to Connection>SSH>Auth and enter under Add new forward port the following information:

a. Under Source Port, enter the port your database engine listens on (i.e. 5432 for Postgres, 3306 for MySQL).

b. In the Destination fields, enter your RDS endpoint name followed by ":port" (i.e. dev-3xbmz.cgovqepkfvtj.us-gov-west-1.rds.amazonaws.com:5432).

c. Under the Destination field: Leave "Local" and "Auto" selected.

d. Click the "Add" button to populate the Forward Ports list with your entry.

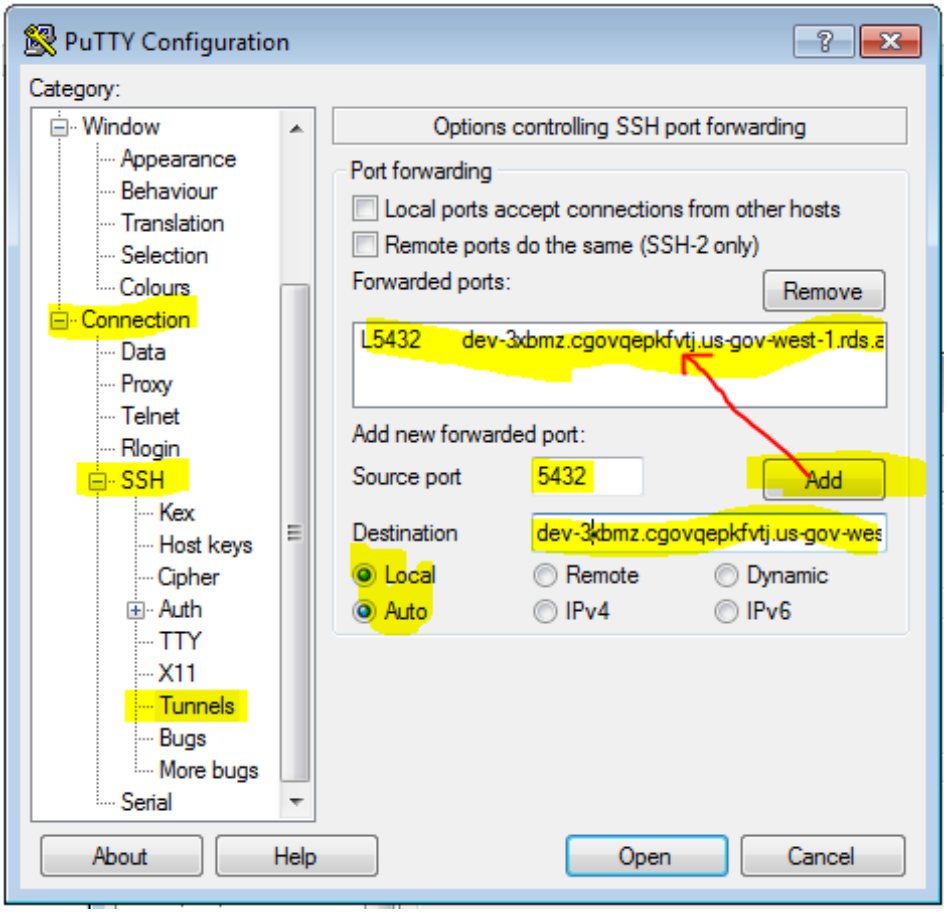


Figure 43: Putty Configuration Forwarded Ports

6.) Navigate back to the Session page, verify everything is still correct, click Save, then Open to start the SSH client.

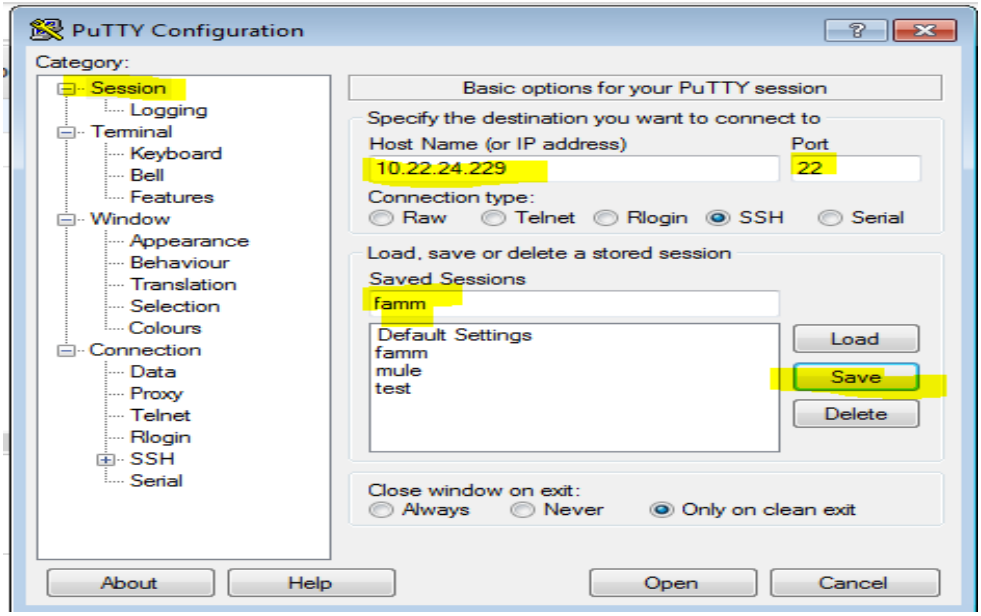
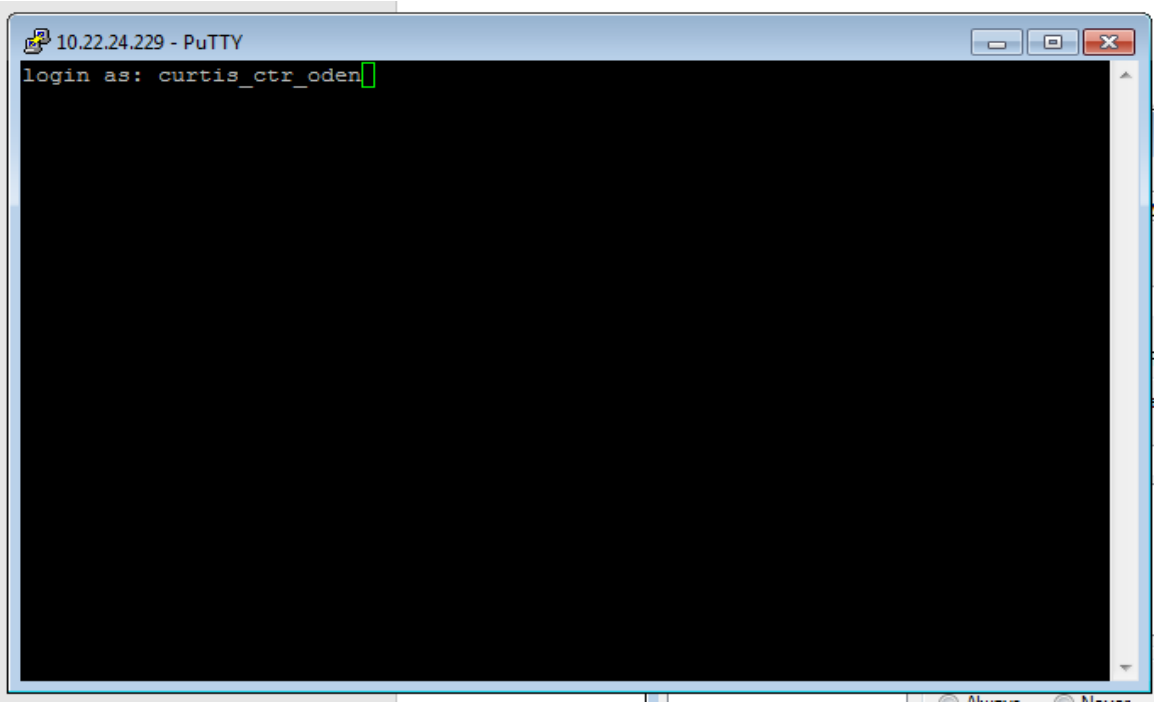
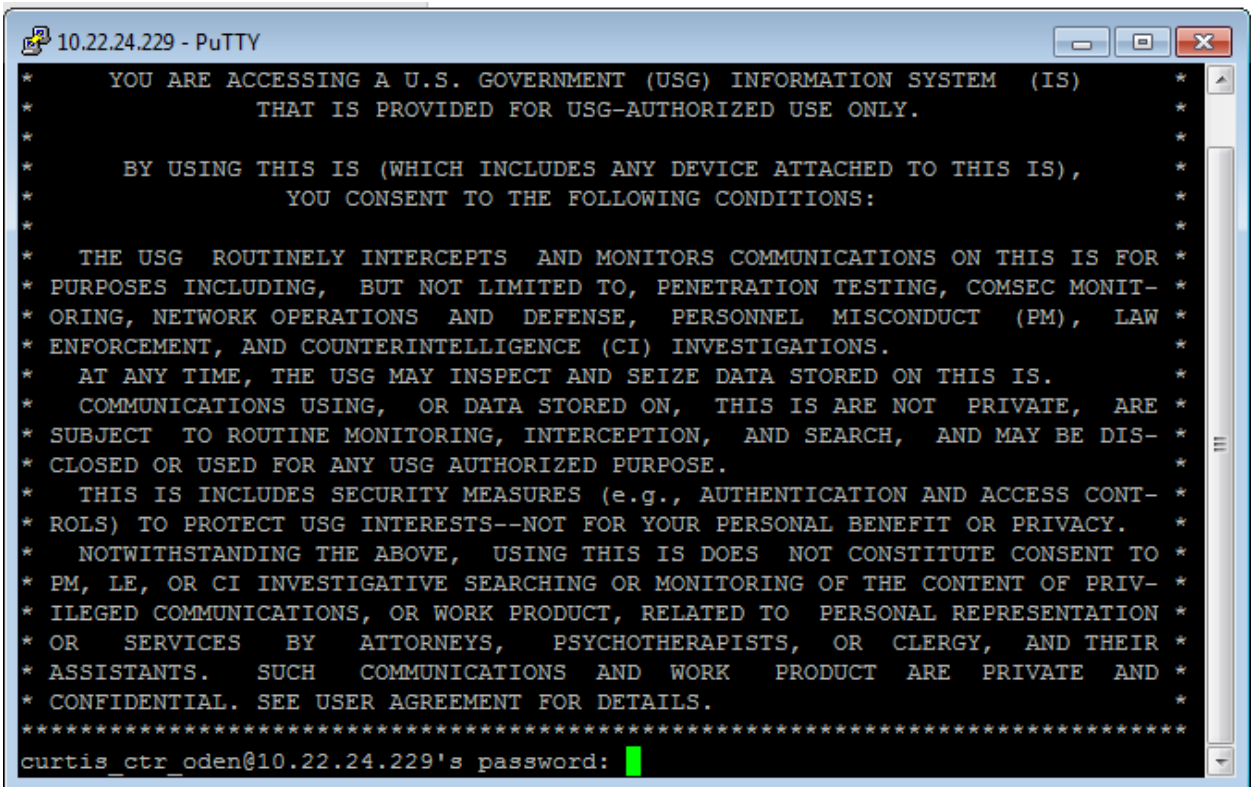


Figure 44: Putty Configuration Saved Sessions

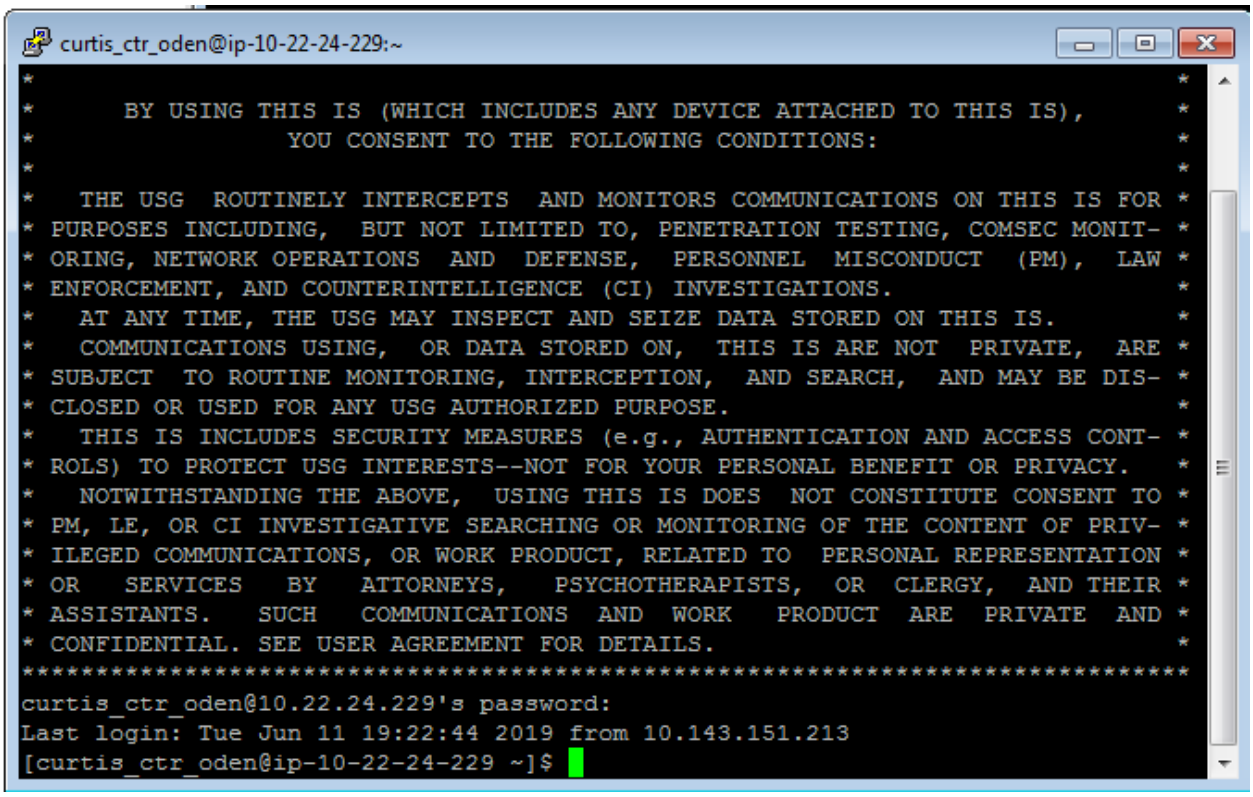
7.) Enter your username in the console:



8.) Enter your FAA Domain password (not your PIV Card PIN):



9.) The connection bridge is now established. Leave the window open and switch to your preferred database client.



10.) Configure your client like you normally would, but instead of the endpoint, use your localhost address (i.e. 127.0.0.1 instead of dev-3xbmz.cgovqepkfvtj.us-gov-west-1.rds.amazonaws.com).

11.) If there are issues connecting, check the following:

a. Connection via the FAA VPN or inside the FAA network.
b. The correct username, password and database credentials are used.
c. There is not a local database engine already running on the listening ports
d. The AWS Security Groups settings allow the instance being used to connect to the RDS database
e. There are not any firewalls or antivirus blocking the port on the local workstation
f. On the Linux box, firewalld or IPsec is not blocking outgoing connections on the database port.

Appendix K: Installing Privacera CLI on Windows and Linux Desktops

Privacera provides command line interface (CLI) to run queries from the command line. It is similar to the AWS CLI. It has two different installation paths (1) Windows (2) RHEL.

Setup Privacera CLI and proxy on Windows GFE

To successfully install the CLI on Windows, you have to enable support for 'Windows Subsystem for Linux' and then download the WSL distribution. Here are the steps:

1. Go to programs and features on the control panel

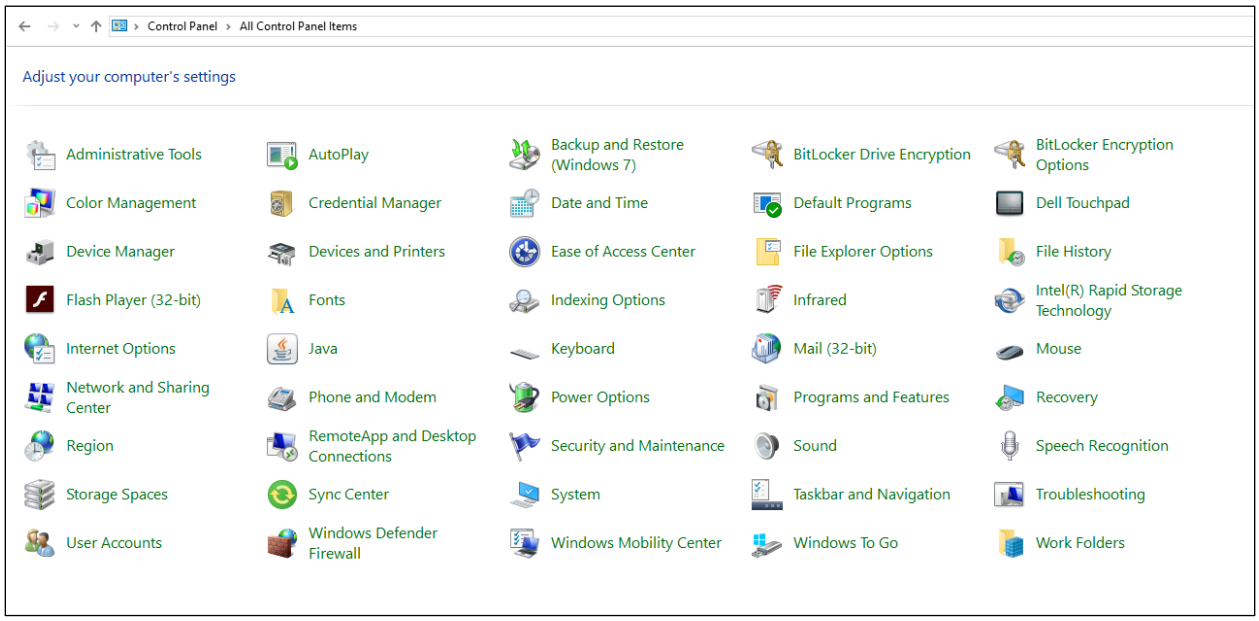


Figure 45: Computer Panel Settings

2. **Click** on Programs and features
3. **Click** on "Turn windows features on or off"

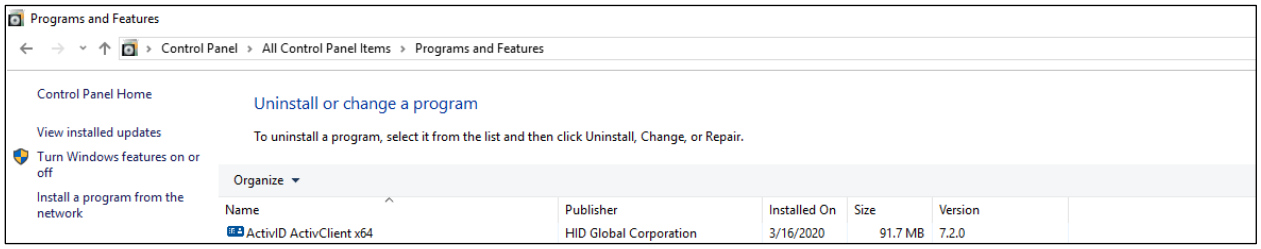
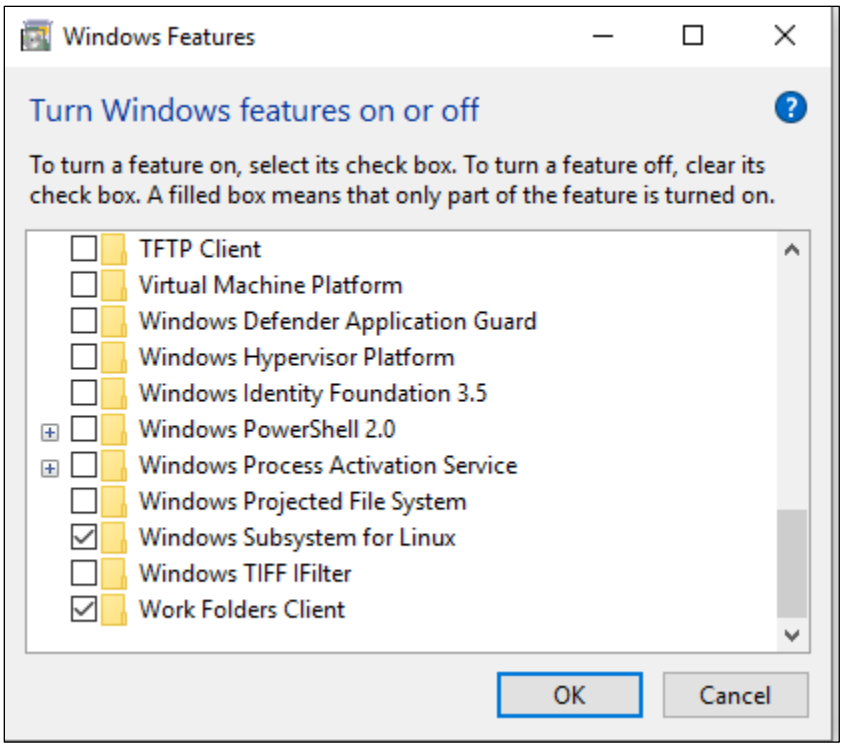


Figure 46: Programs and Features

4. Check Windows subsystem for Linux.



5. Once WSL is enabled, install the Linux distro for Windows (E.g. Ubuntu 20.4)

   You can use below link for download or use Windows app store to download

   https://docs.microsoft.com/en-us/windows/wsl/install-manual (download Ubunto 20.4)

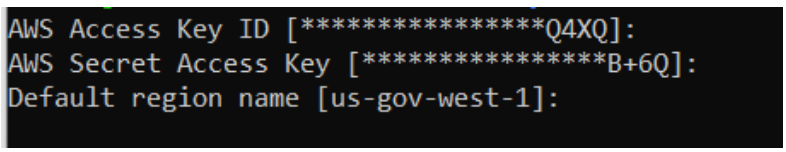
6. Once Linux distro is installed, launch 'wsl' from Windows commprompt
7. Setup your username and password (use lowercase for username per ubuntu regex)
8. Once the username and password is set, download the AWS CLI using the below steps:

```
sudo apt-get update
sudo apt-get install awscli
```

9. Run the aws configure to set the region to

   `AWS configure` (set only the region, leave other prompts as NULL)

```
AWS Access Key ID: < Leave blank, press enter key>
AWS Secret Access Key: < Leave blank, press enter key>
Default Region Name: us-gov-west-1
```



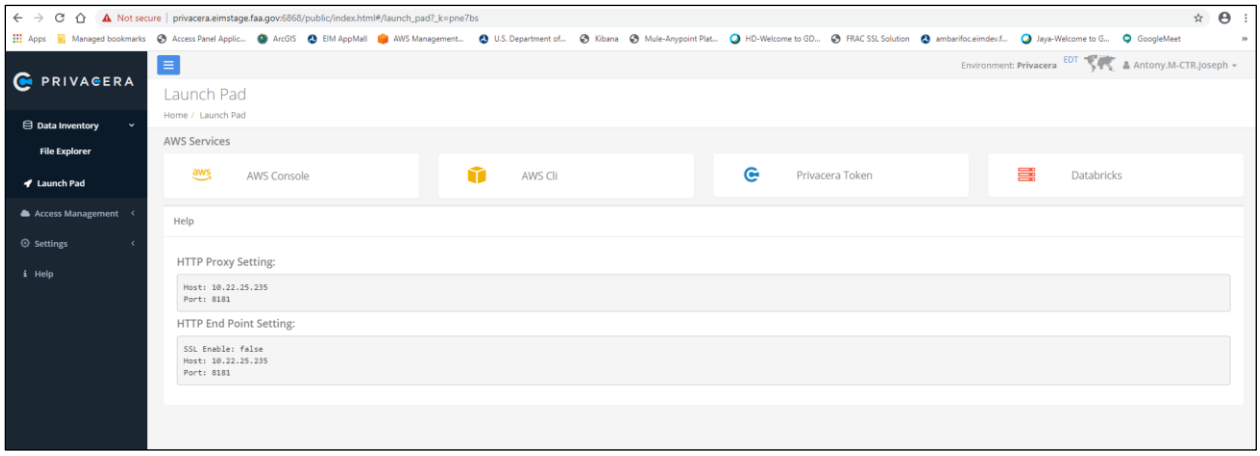
10. Once above steps are done, download privacera_aws.sh and configure token and enable Privacera proxy. Use the steps listed in the 'CLI' tab on the portal. These steps are listed in the Privacera proxy Setup for Linux Machine section below.

Setup Privacera proxy for Linux Machine

1. Click on environment specific Privacera App:

   https://account.activedirectory.windowsazure.com/r#/applications

2. This will take you directly to Privacera portal:
https://privacera.eimstage.faa.gov:6868/public/index.html#/launch_pad?_k=b0l66s



3. Click on AWS CLI
4. Follow instructions given by Privacera under AWS CLI (sample below)
5. Download privacera_aws.sh script

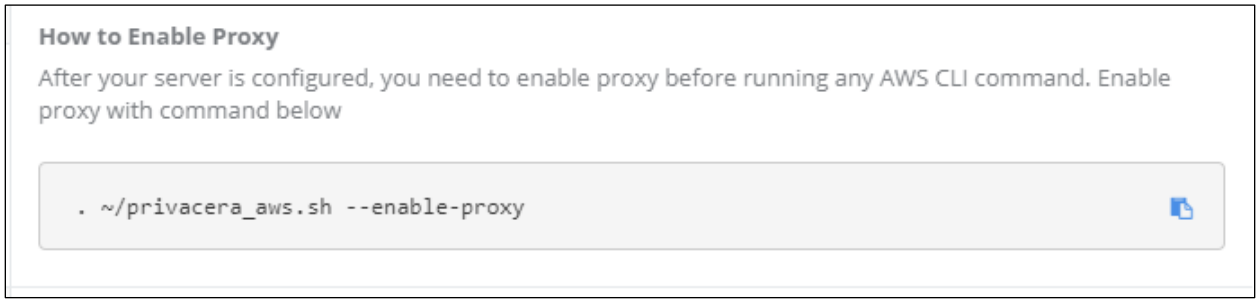
6. Copy the above script in the Linux machine in the home folder.
7. Run `chmod a+x ~/privacera_aws.sh`
8. Run `. ~/privacera_aws.sh --config-token`

9. Generate Config Token

10. Input this token information in the command line (step 8 prompts you to input both access and secret key tokens)

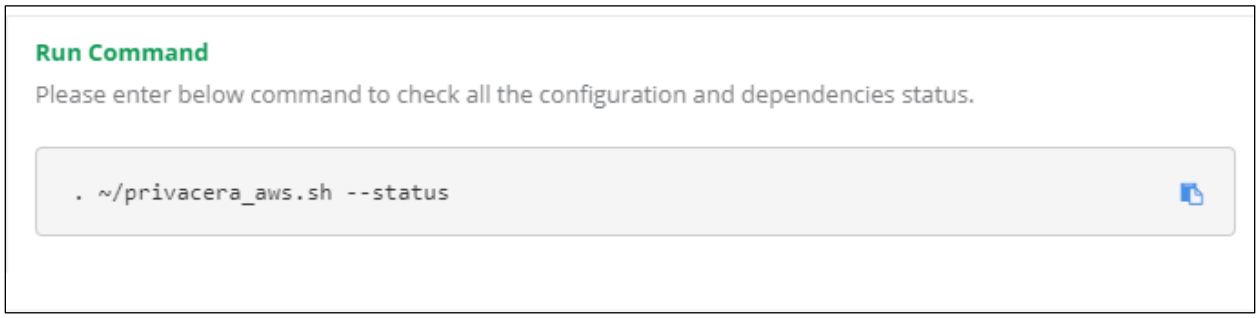
11. Enable proxy by running below command

```
. ~/privacera_aws.sh --enable-proxy
```

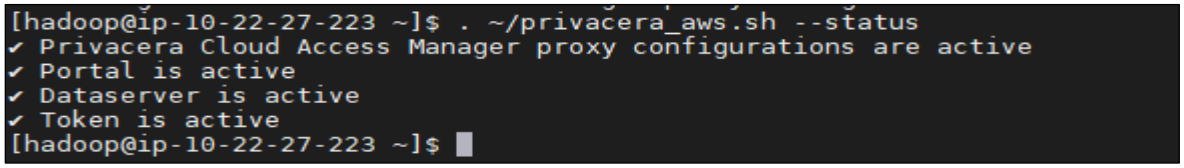


12. Check Status by running below command

```
. ~/privacera_aws.sh –status
```



13. Now you will see below status if all steps followed correctly.



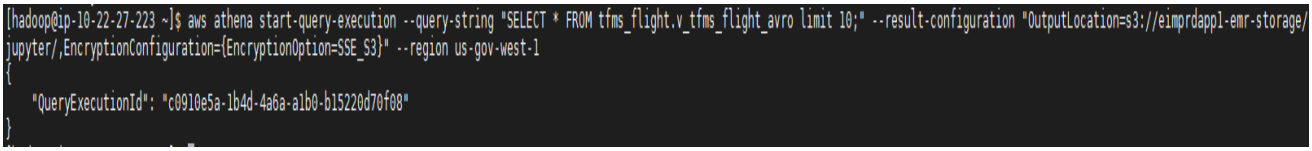
14. Run your AWS CLI S3 and Athena commands

Privacera CLI for Athena Query

1. Once Privacera proxy is enabled, the developer can run Athena queries:

Sample Query

aws Athena start-query-execution --query-string "SELECT * FROM tfms_flight.v_tfms_flight_avro limit 10;" --result-configuration "OutputLocation=s3://eimprdapp1-emr-storage/jupyter/,EncryptionConfiguration={EncryptionOption=SSE_S3}" --region us-gov-west-1



2. The query will return “QueryExecutionId.” The developer can check status of the query using below:

Sample query-execution-id ->334699e8-3d03-42e7-8102-fdb8d56d6a88

AWS Athena get-query-execution --query-execution-id "334699e8-3d03-42e7-8102-fdb8d56d6a88"

```
[hadoop@ip-10-22-27-223 ~]$
[hadoop@ip-10-22-27-223 ~]$ aws athena get-query-execution --query-execution-id "334699e8-3d03-42e7-8102-fdb8d56d6a88"
{
    "QueryExecution": {
        "Status": {
            "SubmissionDateTime": 1596552224.105,
            "State": "SUCCEEDED",
            "CompletionDateTime": 1596552230.001
        },
        "Statistics": {
            "ServiceProcessingTimeInMillis": 10,
            "EngineExecutionTimeInMillis": 5586,
            "QueryPlanningTimeInMillis": 2675,
            "TotalExecutionTimeInMillis": 5896,
            "QueryQueueTimeInMillis": 300,
            "DataScannedInBytes": 10887789
        },
        "ResultConfiguration": {
            "EncryptionConfiguration": {
                "EncryptionOption": "SSE_S3"
            },
            "OutputLocation": "s3://eimprdapp1-emr-storage/jupyter/334699e8-3d03-42e7-8102-fdb8d56d6a88.csv"
        },
        "QueryExecutionId": "334699e8-3d03-42e7-8102-fdb8d56d6a88",
        "QueryExecutionContext": {},
        "Query": "SELECT * FROM tfms_flight.v_tfms_flight_avro limit 10",
        "StatementType": "DML",
        "WorkGroup": "primary"
    }
}
[hadoop@ip-10-22-27-223 ~]$
```

3. To obtain execution results use:

AWS Athena get-query-results --query-execution-id "334699e8-3d03-42e7-8102-fdb8d56d6a88"

```
[hadoop@ip-10-22-27-223 ~]$ aws athena get-query-results --query-execution-id "334699e8-3d03-42e7-8102-fdb8d56d6a88" | head -10
{
    "UpdateCount": 0,
    "ResultSet": {
        "Rows": [
            {
                "Data": [
                    {
                        "VarCharValue": "attr_ds"
                    },
```

Privacera CLI for S3

1. The developer can use AWS CLI commands in the Privacera CLI as shown below. The results of the commands are trimmed based on users' level access in the Ranger S3 policies. Some users may be permitted to see the content metadata only, i.e. List of files in a bucket folder. To download specific files, the policy should have appropriate 'read' permissions.

Sample commands.

aws s3 ls
aws s3 ls eimprdapp1-emr-storage/
aws s3api list-objects-v2 --bucket eimprdapp1-emr-storage

```
[hadoop@ip-10-22-27-223 ~]$ aws s3 ls | head -10
2019-04-02 03:50:10 config-bucket-827674730919
2020-07-30 19:25:40 eimprdapp1-bastion-cdn
2020-07-31 01:10:33 eimprdapp1-emr-boostrap
2020-07-30 19:25:38 eimprdapp1-emr-logs
2020-07-30 19:25:38 eimprdapp1-emr-storage
2019-09-09 19:12:20 eimprod1-acregistry
2019-06-25 16:26:14 eimprod1-adapt
2019-06-25 16:26:14 eimprod1-aeronaut
2020-02-25 16:26:14 eimprod1-arcgis
2019-06-25 16:26:20 eimprod1-asdex

[Errno 32] Broken pipe
[hadoop@ip-10-22-27-223 ~]$
```

```
[hadoop@ip-10-22-27-223 ~]$ aws s3api list-objects-v2 --bucket eimprdapp1-emr-storage  | head -30
{
    "Contents": [
        {
            "LastModified": "2020-08-04T14:43:50.000Z",
            "ETag": "\"b7248cef9ad78c9b382d28523bbac0f5\"",
            "StorageClass": "STANDARD",
            "Key": "jupyter/334699e8-3d03-42e7-8102-fdb8d56d6a88.csv",
            "Size": 1098524
        },
        {
            "LastModified": "2020-08-04T14:43:50.000Z",
            "ETag": "\"94105c77bff2cd40cf6f7ff399828384\"",
            "StorageClass": "STANDARD",
            "Key": "jupyter/334699e8-3d03-42e7-8102-fdb8d56d6a88.csv.metadata",
            "Size": 2735
        },
        {
            "LastModified": "2020-08-03T20:15:10.000Z",
            "ETag": "\"3a1e65ae21e3c78d06fd4031f8fcd1c3\"",
            "StorageClass": "STANDARD",
            "Key": "jupyter/372b8f02-283b-43f9-aad2-d9eb7e1af6ae.csv",
            "Size": 6512
        },
        {
            "LastModified": "2020-08-03T20:15:10.000Z",
            "ETag": "\"e7f3faecbb040b7c4c8b5807a4eef29e\"",
            "StorageClass": "STANDARD",
            "Key": "jupyter/372b8f02-283b-43f9-aad2-d9eb7e1af6ae.csv.metadata",
            "Size": 4728
        },
[hadoop@ip-10-22-27-223 ~]$
```

Appendix L: Runbook for registering and processing new feeds

For the technical design of the schema registration process, please refer to the FAA Wiki link here.

This processes automates the configuration of data feeds that are serialzed to avro format, registered as tables in Glue catalog and are queryable using Athena and Presto. The process includes registration of data feeds with a well-defined schema or using sample files (i.e. csv or json) to derive the schema.

Overall concept and the underlying logic of this process can be found here.

Below section explains the automation of this process as applicable to new feeds.

NOTE: This ansible project is executed first and only one time before any future data feeds can be processed below

Create Ansible Project (eim-data-management)

1. Name: eim-data-management
2. Organization: EIM Core
3. SCM Type: Git
4. SCM Source: ssh://git@git.faa.gov:7999/eim-ec/eim-data-management.git
5. SCM Credential: EIM Core SCM Credential
6. SCM BRANCH/TAG/COMMIT: ansible

Create Ansible Template

1. Name: eim-data-management-<environment>
2. Inventory: <Environment created for>
3. Project: eim-data-management
4. Playbook: aws.yml
5. Credentials:
    a. AWS Credential – For environment
        i. Credential type: Amazon Web Services
6. Limit: Extra Variables:

    eim_environment: “dev”,”test” or “prod”
    lambdaRole: "role for lambda functions to run everything"
    vpcSecurityGroups: Id's of security groups that have access to the vpc that lambda function runs in.
        i. Prod: sg-e099ce87
        ii. Non-prod: sg-045e3a63
    vpcSubnets: Id's of subnets within vpc to be used for lambda"
        i. Prod: “subnet-54d14630,subnet-fb6e9aa2”
        ii. Stage: “subnet-860df9df,subnet-e530a481”
        iii. Dev: "subnet-e630a482,subnet-c302f69a"
    eim_account_id: An AWS account ID is a 12-digit number, that you use to construct Amazon Resource Names (ARNs)
        i. Prod: 827674730919
        ii. Non-prod: 821313603615
    eim_region: "us-gov-west-1"
    sourceCodeBucket: "bucket that holds all lambda zip files"

NOTE: To create an individual feed in the system run this once for every feed type

Create Ansible Project (eim-data-management-feeds)

1. Name: eim-data-management-feeds
2. Organization: EIM Core
3. SCM Type: Git
4. SCM Source: ssh://git@git.faa.gov:7999/eim-ec/eim-data-management.git
5. SCM Credential: EIM Core SCM Credential
6. SCM BRANCH/TAG/COMMIT: master

Create Ansible Template

1. Name: eim-data-management-<environment>
2. Inventory: <Environment created for>
3. Project: eim-data-management-feeds
4. Playbook: awsfeeds.yml
5. Credentials:
    a. AWS Credential - eim_foc_<environment>_gcc
        i. Credential type: Amazon Web Services
    b. Medical Credential – <environment>
        i. Machine
6. Limit: Extra Variables:

```
eim_environment: “dev”,”test” or “prod”
eim_account_id:
    i. Dev/Stage: "821313603615"
    ii. Prod: “827674730919”
eim_region: "us-gov-west-1"
lambdaRole:
    i. Dev/stage: “spare-eim-foc-data_science-lambda-role"
    ii. Prod: “prod-eim-ioc-data_science-lambda-role”
vpcSecurityGroups:
    i. Dev/Stage: "sg-045e3a63"
    ii. Prod: “sg-e099ce87”
vpcSubnets:
    i. Prod: “subnet-54d14630,subnet-fb6e9aa2”
    ii. Stage: “subnet-860df9df,subnet-e530a481”
    iii. Dev: "subnet-e630a482,subnet-c302f69a"
feedName: "<Name of feed>"
feedBucket: "<s3 bucket for feed>"
sourceCodeBucket: "eim<environment>1-scripts"
queueDeliveryDelay: 0
queueMaxMessageSize: 262144
queueMsgRetentionPeriod: 345600
queueVisibilityTimeout: 30
eim_source: "<db eim_ui.dfr.datafeedregistry.eim_source value>"
eim_type: "<db eim_ui.dfr.datafeedregistry.eim_type value>"
eim_stype: "<db eim_ui.dfr.datafeedregistry.eim_stype value>”
eim_format: "<db eim_ui.dfr.datafeedregistry.eim_format value>"
queryOutputBucket: "eim<environment>1-query-results"
sqs_trigger_event_type: “sns” or “s3”
pre_bucket_partitioning:
```

- name: “<Name of subfolder above partitioning start>| ‘blank’”

Manual steps:

1. Create Glue database.
2. Clone each feeds repo
3. From the subfolder: <environment>/glue_config run:
    a. aws s3 cp . s3://<repo s3 bucket>/configuration/table/ --include "*.config_json" --recursive –sse
4. From the subfolder: <environment>/avsc run for each schema:
    a. aws s3 cp <schema file> s3://<repo s3 bucket>/<schema file path>/<schema file> --sse
5. From Athena: Create View … on top of table(s)

Appendix M: Considerations for Presto Performance Optimizations

Observations

1. Parquet file format based tables are much faster compared to Avro format.
2. Compressing large dataset makes queries run faster.
3. No performance improvement noticed when smaller tables are compressed.
4. Cluster Size play major role in achieving good query performance
5. Enough disc space/storage is required
6. More worker nodes add more processing power and queries gets better performance.
7. Memory tuning is key and there is two main configurations (jvm.config and presto.conf). This memory setting depends on the size of the overall cluster including worker nodes.

Iterations

1. Initial baseline metrics established using Avro tables for performance comparison.
2. Parquet compressed tables created from Avro (Used parquet-tools library for schema inferring)
3. Created a large table for performance assessment (TTFS data /1.3 TB size)
4. Compression of this large table using SNAPPY couldn't complete due to issues with EMR(faced challenges with EMR crashing when huge data with this size is processed for compression)
5. Created Medium Cluster with configuration- m5.16xlarge 64 vCore, 256 GiB memory, EBS Storage:500 GiB and 4 worker node of m5.4xlarge ,16 vCore, 64 GiB memory, EBS only storage EBS Storage:500 GiB
6. Created Small Cluster with configuration - m5.xlarge 4 vCore, 16 GiB memory EBS Storage:64 GiB and 2 worker nodes m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage:64 GiB
7. Changed jvm.config file to increase memory and garbage collection options.
8. Changed query.max-memory and query.max-memory-per-node in presto.config

Metrics

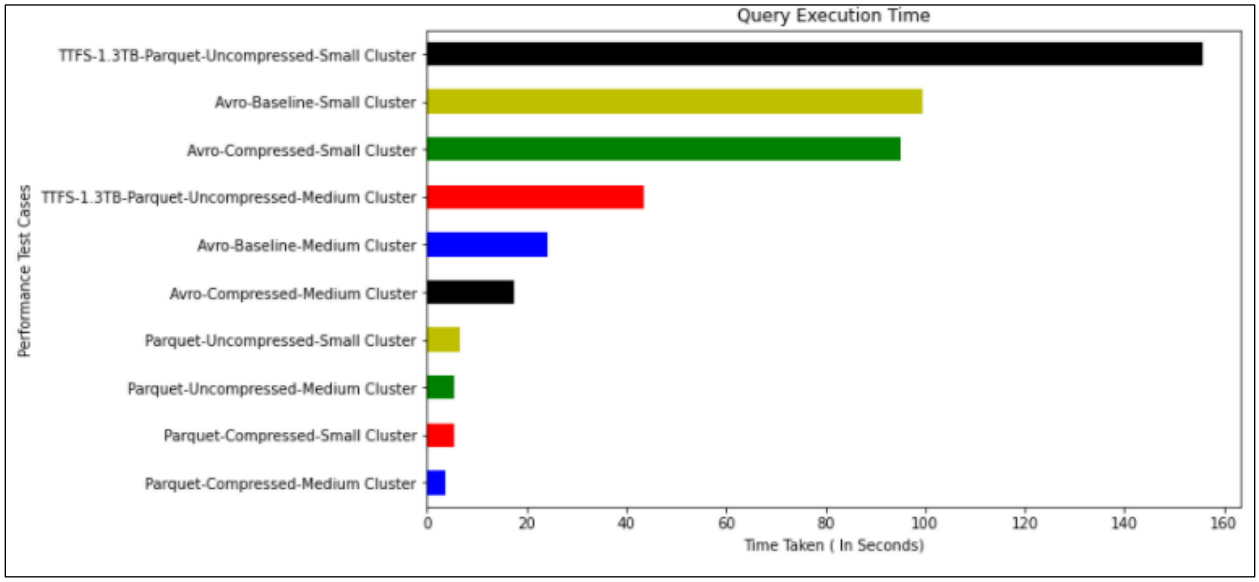


Figure 47: Query Execution Time

Query.max-memory: This parameter, contained in the presto.config properties file, is a cluster-level memory limit. It specifies the maximum memory a query can take, aggregated across all nodes. Setting a higher value of query.max-memory avoids a query hitting an upper limit of memory.

query.max-memory-per-node: This parameter determines the maximum memory a query can use on one node. Ideally, the value of this parameter should be equal to 40 percent of worker instance memory.

Java Virtual Machine (JVM) memory: The jvm.properties file contains details related to the JVM. Since the memory of the system is being shared between the system, Java, and Presto configuration, it is important the developer allocate the right memory to the JVM and spare enough for the OS. The JVM memory should ideally be 80 percent of the total memory of the instance.

Columnar data serialization formats like Parquet can be further optimized to improve query performance. Here are the best practices to improve query performance in Presto and AWS Athena:

1. Convert data to PARQUET file format
2. Apply Compression using SNAPPY
3. Apply Compaction/Creation of large files instead of many small files
4. New Partitioning Strategy - Reduce Over partitioning

Example: Site is removed from the top level partition. eim_day removed from sub partition.

5. Memory tuning for JVM memory, query.max-memory-per-node and query.max-memory

Memory Calculator for performance.

Memory type	**File name**	**Config**	**Formula**
JVM memory - Coordinator	/etc/presto/conf/jvm.config	-Xmx	80% of the total memory
query.max-memory-per-node - Coordinator	/etc/presto/conf/config.properties	query.max-memory-per-node=	JVM * 0.25
query.max-memory - Coordinator	/etc/presto/conf/config.properties	query.max-memory=	query.max-memory-per-node * worker * 0.8

6. New configuration in hive catalog for Presto (/etc/presto/conf/catalog/hive.properties).This allows presto to use column names while reading parquet format based files.

hive.parquet.use-column-names = true

Example:

Aria Airborne Data – JSON to Parquet conversion

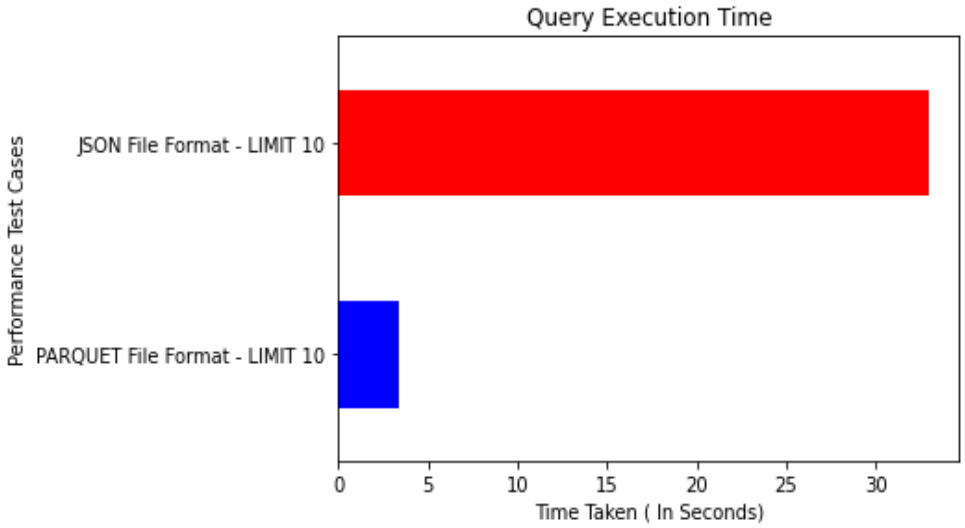
We have applied above conversion and configuration steps to improve the query performance.

Results

- ✓ Query performance improved significantly.
- ✓ Storage consumption reduced.

```
proj_aria_airborne_events_json - JSON format LIMIT 10
Time Taken: 32.988993406295776
proj_aria_airborne_events_parquet3 - PARQUET format LIMIT 10
Time Taken: 3.3275134563446045

Out[6]: Text(0, 0.5, 'Performance Test Cases')
```



References

1. https://aws.amazon.com/blogs/apn/optimizing-presto-sql-on-amazon-emr-to-deliver-faster-query-processing/
2. https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-presto-considerations.html
3. https://prestosql.io/docs/current/admin/properties.html
4. https://teradata.github.io/presto/docs/141t/admin/tuning.html