IT Admin Guide: Review Document

Contents

About the IT Administrator

An individual responsible for maintaining the PHEMI Central system should have experience in the following:

- System administration
- Network security
- Database administration
- Administration of the Apache Hadoop ecosystem

Platform Prerequisites

In describing the tasks of the IT Administrator, this guide assumes that PHEMI Professional Services has installed and set up your base system of cluster nodes and your management node, and has installed and configured all PHEMI Central components on the cluster.

PHEMI Professional Services will also have installed and configured Docker as part of your PHEMI Central deployment. Docker is an open-source engine that automates deploying applications as portable, self-sufficient images. The PHEMI Central Docker containers you will be interacting with include the following:

- The Ambari server. Apache Ambari is an open-source interface for provisioning, managing, and monitoring Hadoop clusters.
- MongoDB. MongoDB is a high-performance, high availability, scalable document database.
- Tornado. Tornado is a Python-based web server and web application framework.
- Ningx. Nginx is an open-source reverse proxy server for the HTTP, HTTPS, SMTP, POP3, and IMAP protocols.

Starting and Stopping the System

Start the System

From a cold state, the components of PHEMI Central must be started in order.

First, start the Hadoop cluster.

1. Start the Ambari server.

   Log on to the node running the Ambari server as a user with sudo privileges. At the command line, enter the following command:

   ```
   sudo service ambari-server start
   ```

2. Access the Ambari console.

   Use a web browser to log on to the node running the Ambari server. The console runs on port 8080.

3. Start Hadoop services managed by Ambari. These include the Hadoop Distributed File System, YARN, MapReduce, Zookeeper, Nagios, Ganglia, Hive and Kafka.

   In the Ambari console, select the cluster. On the left side of the Ambari dashboard, click the **Actions** button and select **Start All**. Ambari begins sending "service start" requests to each node in the cluster. After a few minutes, all services will have been started and the Ambari dashboard will show green.

4. Start the Accumulo, Thrift, Accumulo-Proxy, and Raindrop services.

   Using SSH, log on to the Accumulo master node as a user with sudo privileges. At the command line, issue the following commands:

   ```
   sudo su accumulo
   /usr/lib/accumulo/bin/start-cluster.sh
   ```

After the Hadoop cluster is started, start PHEMI Central Docker containers.

5. Start Docker.

   Log on to the server running PHEMI Central as a user with sudo privileges. At the command line, issue the following command:

   ```
   sudo service docker.io. start
   ```

6. Start MongoDB.

   From the PHEMI Central server, start the Docker container for MongoDB, using the following command:

   ```
   sudo docker start phemi_mongo
   ```

7. Start the Tornado web application framework.

   The Tornado framework runs the PHEMI Central Management and Governance Console. From the PHEMI Central server, start the Docker container for the PHEMI Central Management and Governance Console, using the following command:

   ```
   sudo docker start phemi_central
   ```

8. Start Nginx.

From the PHEMI Central server command line, start the Docker container for Nginx, using the following command:

```
sudo docker start phemi_nginx
```

Stop the System

Stop the PHEMI Central platform using the reverse sequence to when you start the system.

First, stop the PHEMI Central Docker containers.

1. Log on to the server running PHEMI Central as a user with sudo privileges. Stop Nginx.

   From the PHEMI Central server command line, stop the Docker container for Nginx, using the following command:

   ```
   sudo docker stop phemi_nginx
   ```

2. Stop the Tornado web application framework.

   The Tornado web application framework runs the PHEMI Central Management and Governance Console. From the PHEMI Central server command line, start the Docker container for the PHEMI Central Management and Governance Console, using the following command:

   ```
   sudo docker stop phemi_central
   ```

3. Stop MongoDB.

   From the PHEMI Central server command line, stop the Docker container for MongoDB, using the following command:

   ```
   sudo docker stop phemi_mongo
   ```

4. Stop Docker.

   From the PHEMI Central server, stop the Docker, using the following command:

   ```
   sudo service docker.io. stop
   ```

After the PHEMI Central Docker containers have been stopped, stop the Hadoop cluster.

5. Stop the Accumulo, Thrift, Accumulo-Proxy, and Raindrop services.

   Using SSH, log on to the Accumulo master node as a user with sudo privileges. At the command line, issue the following commands:

   ```
   sudo su accumulo
   /usr/lib/accumulo/bin/stop-cluster.sh
   ```

6. Stop the Hadoop services managed by Ambari. These include the Hadoop Distributed File System, YARN, MapReduce, Zookeeper, Nagios, Ganglia, Hive and Kafka.

   Use a web browser to access the Ambari console. The console runs on port 8080. In the Ambari console, select the cluster. On the left side of the Ambari dashboard, click the **Actions** button and select **Stop All**. Ambari begins sending "service stop" requests to each node in the cluster. After a few minutes, all services will have been stopped and the Ambari dashboard will show red.

7. Stop the Ambari server.

Log on to the node running the Ambari server as a user with sudo privileges. At the command line, enter the following command:

```
sudo service ambari-server stop
```

Glossary of Terms and Concepts

access policy

An access policy is a set of rules that specifies how users can consume data stored in PHEMI Central. The access policy lists what user authorizations are required to interact with data tagged with specified visibility. Access policies can be applied to data collections and datasets.

Ambari

Apache Ambari is an open-source interface for provisioning, managing, and monitoring Hadoop clusters.

authorizations

User authorizations are configurable attributes you can assign to PHEMI Central users. Authorizations are defined in PHEMI Central by the PHEMI Administrator, who sets them in accordance with the organization's governance policies.

field

A field is the smallest unit of data storage in PHEMI Central. A field is a single data item, which can range from a single byte up to gigabytes, plus the metadata associated with the data item. Any piece of raw data, regardless of size, is stored in a single field. Elements of derived data (transformed from the raw data) are also each stored individually in fields. Any field can be protected by applying data visibilities. For derived data, each derived item can be individually assigned a visibility (which may be different than that configured for the data collection) by the DPF performing the processing.

code library

A code library is a package of executable code that is included in a DPF archive. Whether the code is source or compiled depends on the coding language. Code libraries must be portable and self-contained; that is, all dependencies required for the DPF to function must be bundled inside the library, in the appropriate way, for whatever language is being used.

data category

Data categories are a way to classify data into broader groupings. Examples of data categories are "Research Reports," "X-Rays," and "Prescriptions."

data collection

In PHEMI Central, a data collection is the set of management and governance rules and policies, as well as the DPF processing, that will be applied to some set of data. A data collection configuration should be defined for each set of data that is to be stored and managed according to the same retention, legal, and governance rules.

Data Processing Function, DPF

A Data Processing Function, or DPF, is an executable piece of code that supplies the instructions for processing raw data to extract meaningful, context-specific information (such as a temperature reading or blood glucose measurement) that can be queried or exported for analysis. The code is executed by the PHEMI Central DPF Engine, which uses it to direct curation of the data. The input to a DPF is the raw binary data ingested into the system. The output of a DPF is a set of structured elements, each of which includes a type property (for example, INT or STRING) and can selectively specify data visibilities (for example, SECRET or IDENTIFIABLE) on a per-field basis. The data elements output by a DPF are called derived data. The collection of derived data produced by a DPF is automatically indexed in PHEMI Central.

data visibilities

See visibilities.

dataset

A dataset is a view, or map, of an underlying set of data. Data items in a dataset can be selected from across multiple data collections. The dataset is a view, or map, to the underlying data. The actual content of the dataset (that is, the dataset's data) is generated when the dataset is executed or when it is queried against.

derived data

Derived data is data that has been parsed, extracted, or otherwise enriched or processed by running a DPF on stored raw data. The set of derived data items can be searched, queried, further processed, or exported from the system.

digital asset

A digital asset is any piece of data stored with metadata in the system. This may be raw data that has had metadata applied during ingestion, or it may be derived data that has been parsed, indexed, catalogued, and/or enriched with additional metadata.

DPF archive

The set of code that makes up a DPF is called a DPF archive. A DPF archive is delivered as a ZIP file archive. It consists of two parts: a manifest file and a code library. To associate a DPF with a data collection, the DPF archive is ``registered`` with the data collection by uploading the DPF archive. Registering the DPF is part of data collection configuration.

ETL

Extract, transform, and load. In databases, a set of tools or processes that extracts data from sources, transforms the format or structure for storage, query, and analysis, and loads it into the receiving or consuming system.

HDFS

The Hadoop Distributed File System, a distributed file system designed for scalability on commodity hardware.

Hadoop

Hadoop is an open-source platform for distributed computing.

ingestion

Ingestion is the process by which data is brought into in PHEMI Central. Often, the sending system submits the data to PHEMI Central, which listens for the data, often using a web service. Data can also be ingested manually, by using the PHEMI Central Management and Governance Console.

JSON

JSON stands for JavaScript Object Notation. JSON is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate. JSON is used in the body of several REST requests in the PHEMI RESTful API. PHEMI Central also includes a system DPF that can create derived data from JSON objects, providing the objects conform to PHEMI’s JSON specification.

key-value pairs

A key-value pair is a set of two linked data items: a key which uniquely identifies some item of data, and the value, which is the data itself. PHEMI Central uses key-value store to efficiently store, process, and retrieve data.

M2M

M2M is a way of referring to machine-to-machine interfaces, used in machine-to-machine communication.

manifest file

A manifest file is a JSON file that specifies the output of a DPF. With the code library, the manifest file makes up the DPF archive that is uploaded to register the DPF with a data collection. The manifest file should include the properties of the DPF along with the details of each derived data item to be generated.

metadata

Metadata is information about a piece of data. In PHEMI Central, metadata is information about how a given piece of data is to be managed. When a piece of raw data is ingested into PHEMI Central, information from the

connection (for example, the timestamp) together with policy information configured for the data collection (for example, the data visibility) and some derived information (for example, a "time to live," as derived from the timestamp and the data retention policy) is used to create metadata properties that are stored with the data. Further, PHEMI Central also automatically indexes and catalogues all stored data, whether raw or derived; the indexes and catalogues can also be considered a kind of metadata.

MongoDB

MongoDB is a high-performance, high availability, scalable document database.

Nginx

(Pronounced "engine-X.") Nginx is an open-source reverse proxy server for the HTTP, HTTPS, SMTP, POP3, and IMAP protocols. It can also act as a load balancer, HTTP cache, and a web origin server .

PII

Personally Identifiable Information, or PII, is a legal concept used in US privacy law and information security to mean information that can be used on its own or with other information to identify, contact, or locate a single person or to identify an individual in context. When thinking about PII, it is important to distinguish legal requirements to remove attributes uniquely identify an individual from a general technical ability to identify individuals. Because of the versatility and power of modern re-identification algorithms, together with the amount of information freely available from all sources, the absence of PII data does not guarantee that de-identified data cannot be used, perhaps in combination with other data, to identify individuals.

raw data

In PHEMI Central, raw data items are files, objects, records, images, and so on that are submitted for ingestion into the system. Raw data is stored exactly as received, along with the metadata generated for it on ingestion.

REST, RESTful API

Representational Statement Transfer (REST) is an architectural style that uses HTTP requests and associated methods (POST, PUT, GET, and DELETE) to create, update, read, and delete data. A RESTful API is an application programming interface (API) based on REST.

Tornado

Tornado is a Python-based web server and web application framework.

VCF

A Variant Call Format (VCF) file is a text file containing tab-separated marker and genotype data. VCF data is used in bioinformatics to store gene sequence variations. As such, VCF files are very large, and a a VCF file can document hundreds, thousands, and even millions of gene sites in a single file.

visibilities

All raw data and derived data stored in PHEMI Central can be tagged with labels that provide information about the data's sensitivity. This sensitivity is described in terms of the visibility the data should have to different system users. The visibility tags you define for your data should reflect the visibility the data is intended to have according to your organization's governance policy.

Two words. (Not "whitepaper.")