Data Access 3

Adding Druid to a cluster

Date of Publish: 2018-07-12

Hortonworks

http://docs.hortonworks.com

Contents

Apache Druid introduction

HDP 3.x includes Apache Druid (incubating). Druid is an open-source, column-oriented data store for online analytical processing (OLAP) queries on event data. Druid is optimized for time-series data analysis and supports the following data analytics features:

- Real-time streaming data ingestion
- Automatic data summarization
- Scalability to trillions of events and petabytes of data
- Sub-second query latency
- Approximate algorithms, such as hyperLogLog and theta

Druid is designed for enterprise-scale business intelligence (BI) applications in environments that require minimal latency and high availability. Applications running interactive queries can "slice and dice" data in motion.

You can use Druid as a data store to return BI about streaming data from user activity on a website or multidevice entertainment platform, from consumer events sent over by a data aggregator, or from a large set of transactions or Internet events.

HDP includes Druid 0.12.1, which is licensed under the Apache License, version 2.0.

Apache Druid architectural overview

Druid Architecture

Apache Druid (incubating) supports streaming ingestion and batch ingestion data analytics modes. A Druid cluster consists of several Druid node types and components. Each Druid node is optimized to serve particular functions. The following list is an overview of Druid node types:

- Realtime nodes

  These nodes ingest and index streaming data that is generated by system events. The nodes construct segments from the data and store the segments until these segments are sent to historical nodes. The realtime nodes do not store segments after the segments are transferred.
- Historical nodes

  These nodes are designed to serve queries over immutable, historical data. Historical nodes download immutable, read-optimized Druid segments from deep storage and use memory-mapped files to load them into available memory. Each historical node tracks the segments it has loaded in ZooKeeper and transmits this information to other nodes of the Druid cluster when needed.
- Broker nodes

  These nodes form a gateway between external clients and various historical and realtime nodes. External clients send queries to broker nodes. The nodes then break each query into smaller queries based on the location of segments for the queried interval and forwards them to the appropriate historical or realtime nodes. Broker nodes merge query results and send them back to the client. These nodes can also be configured to use a local or distributed cache for caching query results for individual segments.
- Coordinator nodes

  These nodes mainly serve to assign segments to historical nodes, handle data replication, and to ensure that segments are distributed evenly across the historical nodes. They also provide a UI to manage different datasources and configure rules to load data and drop data for individual datas sources. The UI can be accessed via Ambari Quick Links.
- Middle manager nodes

These nodes are responsible for running various tasks related to data ingestion, realtime indexing, and segment archives. Each Druid task is run as a separate JVM.

- Overlord nodes

  These nodes handle task management and maintain a task queue that consists of user-submitted tasks. The queue is processed by assigning tasks in order to the middle manager nodes, which actually run the tasks.The overlord nodes also support a UI that provides a view of the current task queue and access to task logs. The UI can be accessed via Ambari Quick Links for Druid.

Apache Druid content roadmap

The content roadmap provides links to the available content resources for Apache Druid (incubating). The hyperlinks in the table and many others throughout this documentation jump to content published on the Druid site. Do not download any Druid code from this site for installation in an HDP cluster. Instead, install Druit by selecting Druid as a Service in an Ambari-asssisted HDP installation.

Table 1: Apache Hive Content roadmap

Type of Information	Resources	Description
Introductions	About Druid	Introduces the feature highlights of Druid, and explains in which environments the data store is best suited. The page also links to comparisons of Druid against other common data stores.
	Druid Concepts	This page is the portal to the druid.io technical documentation. While the body of this page describes some of the main technical concepts and components, the right-side navigation pane outlines and links to the topics in the druid.io documentation.
	Druid: A Real-time Analytical Data Store	This white paper describes the Druid architecture in detail, performance benchmarks, and an overview of Druid issues in a production environment. The extensive References section at the end of the document point to a wide range of information sources.
	Druid Architecture Paper	
Tutorial	Druid Quickstart	A getting started tutorial that walks you through a Druid package, installation, and loading and querying data. The installation of this tutorial is for instructional purposes only and not intended for use in a Hortonworks Hadoop cluster.
Integration with Hive	Druid and Hive Integration	Explains how to index data from Hive into Druid and query Druid datasources from Hive.
Developing on Druid	Developing on Druid	Provides an overview of major Druid components to help developers who want to code applications that use Druid-ingested data. The web page links to another about segments, which is an essential entity to understand when writing applications for Druid.
Data Ingestion	Batch Data Ingestion Loading Streams	These two pages introduce how Druid can ingest data from both static files and real-time streams.
Queries of Druid Data	Querying	Describes the method for constructing queries, supported query types, and query error messages.
Best Practices	Recommendations	A list of tips and FAQs.

Add Apache Druid to the cluster

You use Apache Ambari to add Apache Druid (incubating) to your cluster.

About this task

To use Druid in a real-world environment, the cluster must have access to the following resources to make Druid operational in HDP:

- ZooKeeper:

  A Druid instance requires that you select Apache ZooKeeper as a Service when you add Druid to the cluster; otherwise, Ambari does not add Druid to the cluster. ZooKeeper coordinates Druid nodes and manages elections among coordinator and overlord nodes.
- Deep storage:

  HDFS or Amazon S3 can be used as the deep storage layer for Druid in HDP. In Ambari, you can select HDFS as a Service for this storage layer. Alternatively, you can set up Druid to use Amazon S3 as the deep storage layer by setting the druid.storage.type property to s3. The cluster relies on the distributed file system to store Druid segments for permanent backup of the data.
- Metadata storage:

  The metadata store is used to persist information about Druid segments and tasks. MySQL and Postgres are supported metadata stores. You can select the metadata database when you install and configure Druid with Ambari.
- Batch execution engine:

  Select YARN + MapReduce2 for the execution resource manager and execution engine, respectively. Druid Hadoop index tasks use MapReduce jobs for distributed ingestion of large amounts of data.
- (Optional) Druid metrics reporting:

  If you plan to monitor Druid performance metrics using Grafana dashboards in Ambari, select Ambari Metrics System as a Service.

If you plan to deploy high availability (HA) on a Druid cluster, you need to know which components to install and how to configure the installation so that the Druid instance is primed for an HA environment.

As you add Druid as a service, you generally replicate the following components:

- Druid Historical: Loads data segments.
- Druid MiddleManager: Runs Druid indexing tasks.

Before you begin

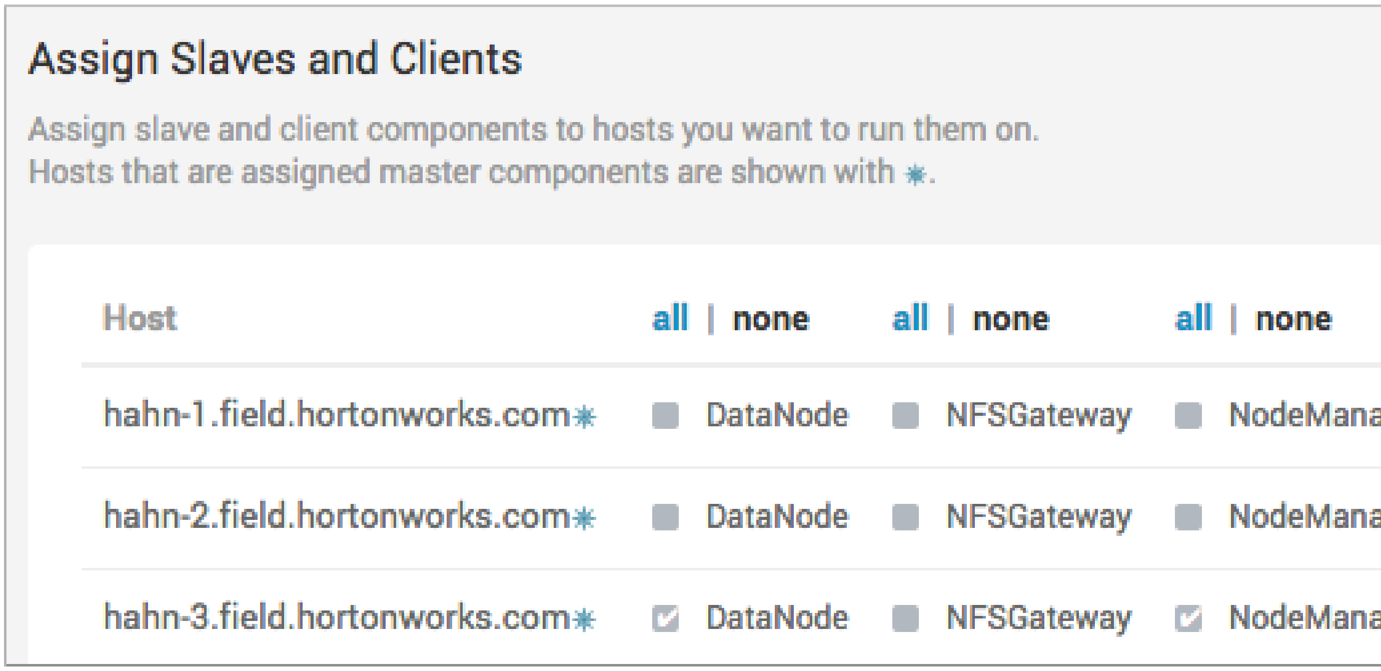
You have installed the following components:

- Ambari 2.7.0 or later
- HDP 3.0 or later using Ambari
- ZooKeeper
- HDFS or Amazon S3
- A MySQL or Postgres database for storing metadata if you need high availability; otherwise, you can use the default Derby database installed and configured by Ambari.
- YARN and MapReduce2

Procedure

1. From the Ambari navigation pane, select Services > Add Service.
2. Select Druid, and click Next.

3. In Assign Masters, select Druid Historical and Druid MiddleManager for multiple nodes.



Assign Slaves and Clients

Assign slave and client components to hosts you want to run them on.
Hosts that are assigned master components are shown with ✱.

Host	all | none	all | none	all | none
hahn-1.field.hortonworks.com✱	☐ DataNode	☐ NFSGateway	☐ NodeMana
hahn-2.field.hortonworks.com✱	☐ DataNode	☐ NFSGateway	☐ NodeMana
hahn-3.field.hortonworks.com✱	✓ DataNode	☐ NFSGateway	✓ NodeMana

4. In Customize Services, select a Druid Metadata storage type or accept the default Derby database.
5. Enter a Metadata storage password.
6. Accept the default configuration values and suggested changes if you use the default Derby database.
7. Click Next, and Deploy.
8. Check Apache Ambari Installation Guide if you encounter any problems to see if other components are needed for your particular cluster.

Configure Apache Druid

Using Ambari facilitates configuring Apache Druid (incubating) by presenting parameters in a graphical user interface.

About this task

This task covers how to configure basic settings. You can access other settings on the Advanced tab.

Ambari populates many configuration settings based on the your previously entered settings and your environment. Although Stack Advisor provides many default settings for Druid nodes and other related entities in the cluster, you might need to manually tune the configuration parameter settings.

Procedure

1. In Configs, change the Druid Metadata storage database name to druid if necessary.
2. From the drop-down, notice the database types that appear, and then select Derby for a quick non-production installation, the MySQL, or the Postgres database.
3. In Metadata storage password, enter a password.

   If you selected MySQL or Postgres, setup your metadata storage database, and other settings on the Advanced tab.
4. Review the druid.io documentation to determine if you need to change any settings.

Set up MySQL for Apache Druid

If you chose MySQL as the metadata store for Apache Druid (incubating) during the installation, you must set up the MySQL connector and follow a few steps to configure the database.

Procedure

1. On the Ambari server host, run the command to set up the connector: ambari-server setup --jdbc-db=mysql --jdbc-driver=/path/to/mysql/mysql-connector-java.jar
2. Create a Druid database by executing the following command, replacing <DRUIDDATABASE> with the Druid database name.

```
# mysql -u root -p
CREATE DATABASE <DRUIDDATABASE> DEFAULT CHARACTER
SET utf8;
```

3. On the mysql command line, create a Druid user with sufficient superuser permissions:

```
CREATE USER '<DRUIDUSER>'@'%' IDENTIFIED BY '<DRUIDPASSWORD>';
GRANT ALL PRIVILEGES ON <DRUIDDATABASE>.* TO '<DRUIDUSER>'@'%’;
FLUSH PRIVILEGES;
```

Configure Apache Druid for high availability

To make Apache Druid (incubating) highly available, you need to configure a sophisticated, multinode structure.

About this task

The architecture of a Druid cluster includes a number of nodes, two of which are designated Overlord and Coordinator. Within each Overlord and Coordinator domain, ZooKeeper determines which node is the Active node. The other nodes supporting each process are in Standby state until an Active node stops running and the Standby nodes receive the failover. Multiple Historical and Realtime nodes also serve to support a failover mechanism. But for Broker and Realtime processes, there are no designated Active and Standby nodes. Muliple instances of Druid Broker processes are required for HA. Recommendations: Use an external, virtual IP address or load balancer to direct user queries to multiple Druid Broker instances. A Druid Router can also serve as a mechanism to route queries to multiple broker nodes.

Before you begin

- You ensured that no local storage is used.
- You installed MySQL or Postgres as the metadata storage layer. You cannot use Derby because it does not support a multinode cluster with HA.
- You configured your MySQL or Postgres metadata storage for HA mode to avoid outages that impact cluster operations. See your database documentation for more information.
- You planned to dedicate at least three ZooKeeper nodes to HA mode.

Procedure

1. In Ambari, enable Namenode High Availability using the Ambari wizard. See Ambari documentation for more information.
2. Install the Druid Overlord, Coordinator, Broker, Realtime, and Historical processes on multiple nodes that are distributed among different hardware servers.

3. Ensure that the replication factor for each data source is greater than 1 in the Coordinator process rules. If you did not change data source rule configurations, no action is required because the default replication factor is 2.

Securing Apache Druid using Kerberos

It is important to place the Apache Druid (incubating) endpoints behind a firewall and configure authentication.

You can configure Druid nodes to integrate a Kerberos-secured Hadoop cluster. Such an integration provides authentication between Druid and other HDP Services. If Kerberos is enabled on your cluster, to query Druid data sources that are imported from Hive, you must set up LLAP.

You can enable the authentication with Ambari 2.5.0 and later to secure the HTTP endpoints by including a SPNEGO-based Druid extension. After enabling authentication in this way, you can connect Druid Web UIs to the core Hadoop cluster while maintaining Kerberos protection. The main Web UIs to use with HDP are the Coordinator Console and the Overlord Console.

To use Hive and Druid together under Kerberos, you must run Hive Interactive Server to configure Hive in low-latency, analytical processing (LLAP) mode.

Related Information

Set up LLAP

Enable Kerberos authentication in Apache Druid

As Administrator, you can set up authentication of users who submit queries through Apache Druid (incubating) to the rest of the Hadoop cluster. If Kerberos is enabled on your cluster, to query Druid data sources that are imported from Hive, you must set up LLAP (low-latency, analytical processing).

Before you begin

- You enabled SPENGO-based Kerberos security on the cluster using the Ambari Server and Services.
- You planned for temporary down-time that is associated with this task.

  The entire HDP cluster must shut down after you configure the Kerberos settings and initialize the Kerberos wizard.
- You set up and enabled LLAP if you want to use Hive and Druid, and Hive Server Interactive is running.

About this task

To enable SPNEGO-based Kerberos authentication between the Druid HTTP endpoints and the rest of the Hadoop cluster, you run the Ambari Kerberos Wizard and manually connect to Druid HTTP endpoints in a command line. In this wizard, you configure the following Advanced Druid Identity Properties.

Property	Default Value Setting	Description
druid.hadoop.security.spnego.excludedPaths	['status'] To set more than one path, enter values in the following format:['/status','/condition']	Specify here HTTP paths that do not need to be secured with authentication. A possible use case for providing paths here are to test scripts outside of a production environment.
druid.hadoop.security.spnego.keytab	keytab_dir/spnego.service.keytab	The SPNEGO service keytab that is used for authentication.
druid.hadoop.security.spnego. principal	HTTP/_HOST@realm	The SPNEGO service principal that is used for authentication.
druid.security.extensions.loadlist	[druid-kerberos]	Indicates the Druid security extension to load for Kerberos.

Initializing the Kerberos Wizard might require a significant amount of time to complete, depending on the cluster size. Refer to the GUI messaging on the screen for progress status.

Procedure

1. In Ambari, launch the Kerberos wizard automated setup.
2. In Configure Identities, adjust advanced Druid configuration settings: Review the principal names, particularly the Ambari Principals on the General tab and either leave the default appended names or adjust them by removing the -cluster-name from the principal name string.
   If your cluster is named druid and your realm is EXAMPLE.COM, the Druid principal that is created is druid@EXAMPLE.COM

   These principal names, by default, append the name of the cluster to each of the Ambari principals.
3. Select the **Advanced** > **Druid drop-down**.
4. Determine for which Advanced Druid Identity properties, if any, you need to change the default settings.

   Generally, you do not need to change the default values.
5. Confirm your configuration, and, ptionally, download a CSV file of the principals and keytabs that Ambari can automatically create.
6. Click Next.
   Kerberos configuration settings are applied to various components, and keytabs and principals are generated. When the Kerberos process finishes, all Services are restarted and checked. After authenticating successfully to Druid, users can submit queries through the endpoints.

Related Information

Set up LLAP

Access Kerberos-protected HTTP endpoints

As a user, before accessing any Druid HTTP endpoints, you need to authenticate yourself using Kerberos and get a valid Kerberos ticket.

Procedure

1. Log in through the Key Distribution Center (KDC).
   kinit -k -t <keytab_file_path> <user@REALM.COM>
2. Verify that you received a valid Kerberos authentication ticket by running the klist command.
   The console prints a Credentials cache message if authentication was successful. An error message indicates that the credentials are not cached.
3. Access Druid with a curl command, including the SPNEGO protocol --negotiate argument.

   ```
   curl --negotiate -u:<anyUser> -b ~/<cookies.txt> -c ~/<cookies.txt> -X
    POST -H'Content-Type: application/json' http://_<endpoint>
   ```

   The negotiate argument is prefaced by two hyphens.
4. Submit a query to Druid:

   ```
   curl --negotiate -u:<anyUser> -b ~/<cookies.txt> -c ~/<cookies.txt> -X
    POST -H'Content-Type: application/json'  http://broker-host:port/druid/
   v2/?pretty -d @query.json
   ```