Security 3

Deploying SAM Applications in a Secure Cluster

Date of Publish: 2019-03-15

https://docs.hortonworks.com/

Contents

Deploying SAM Applications in a Secure Cluster

In a secure/kerberized env, SAM will deploy an application to kerberized Storm cluster and that app has to talk to secure services (e.g.: Secure Kafka, HBase, HDFS, Hive, etc..). Deploying secure streaming apps that talks to secure services has been traditionally very difficult to configure. SAM makes it considerably easier to deploy secure streaming apps.

Connecting to a Secure Service that Supports Delegation Tokens

SAM uses *delegation tokens* when possible, when talking to secure streaming services. The concept of delegation token is introduced to avoid frequent authentication checks against Kerberos(AD/MIT). After the initial authentication against Namenode using Kerberos, subsequent authentication are done without a Kerberos service ticket(TGT). Once the client authentication with Kerberos for Namenode is successful, the client receives a delegation token from the Namenode. This token has an expiration and max issue date but can be reviewed.

A delegation token is secret key shared with the Storm/NameNode/HBase which provides a mechanism for Storm/ NameNode/HBase to impersonate a user to perform an operation. Delegation tokens are supported for the following services: Storm, HDFS, Hive, HBase.

You can use Storm's Nimbus service to get delegation tokens on behalf of the topology submitter user. Nimbus can get HDFS, HBase and other delegation tokens associated with the user who submitted the topology and can push it to the users stream application. This decreases operational/deployment complexity because you do not have to distribute keytabs to all possible key tabs.

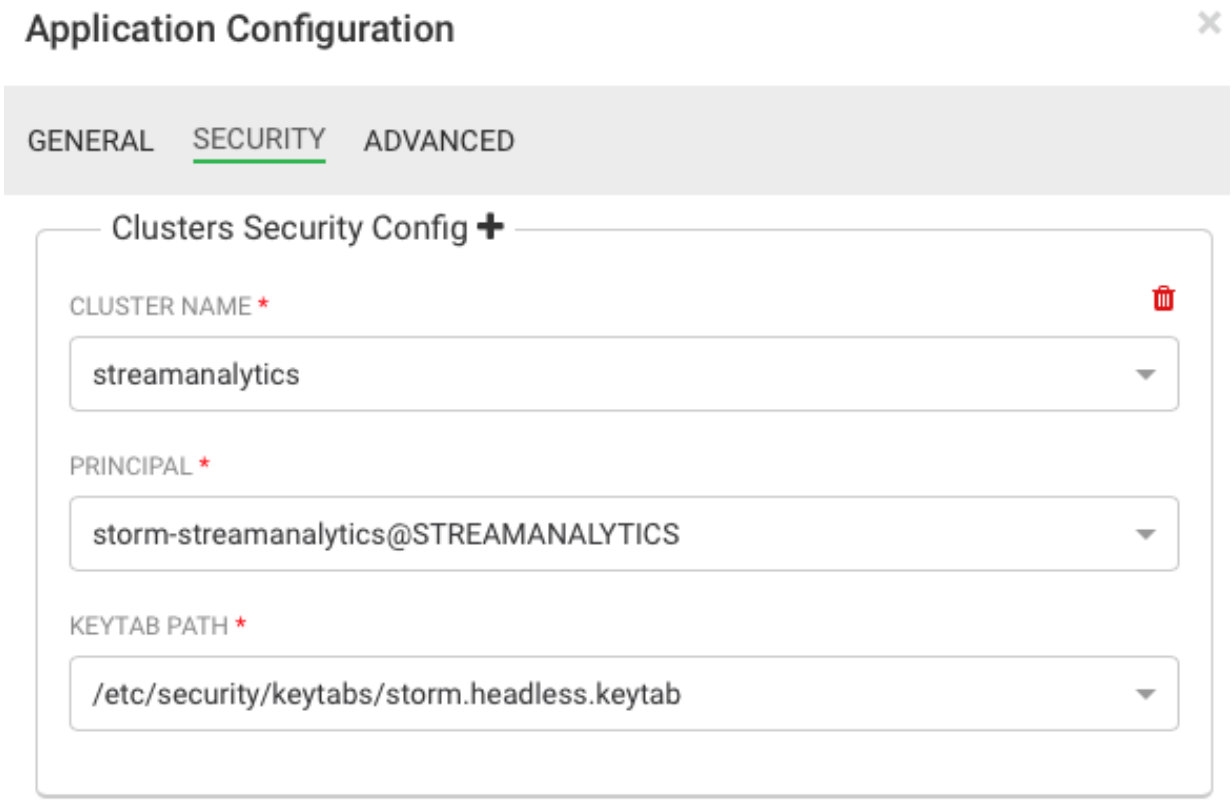
Example

If your application is going to interact with secure HBase, your bolts/states needs to be authenticated by HBase. Typically, you are required to have storm.keytab.file on all the potential worker hosts. If you have multiple topologies on a cluster, each with different user, you will have to create multiple keytabs and distribute it to all workers.

With SAM, you can configure Nimbus to automatically get delegation tokens on behalf of the topology submitter user. To do this in SAM, you can configure a single principal and keytab in SAM for a given application and this principal is used by Nimbus to impersonate the user/app. The only requirement is that the keytab for this principal must reside on the host where Nimbus is located. To configure this single principal that will be used by Nimbus to impersonate the user/app when connecting to secure big data services like HBase, HDFS, Hive, do the following:

Steps

1. Click into your stream application, and then click **Edit**.
2. Click the **Configure** icon
   ()
   located on top right of the stream application.
3. Select the **Security** tab.
4. Select the principal and Keytab path. SAM automatically populates all the principal and key tabs located on the Nimbus Host to make this easier. Then click **Ok**.



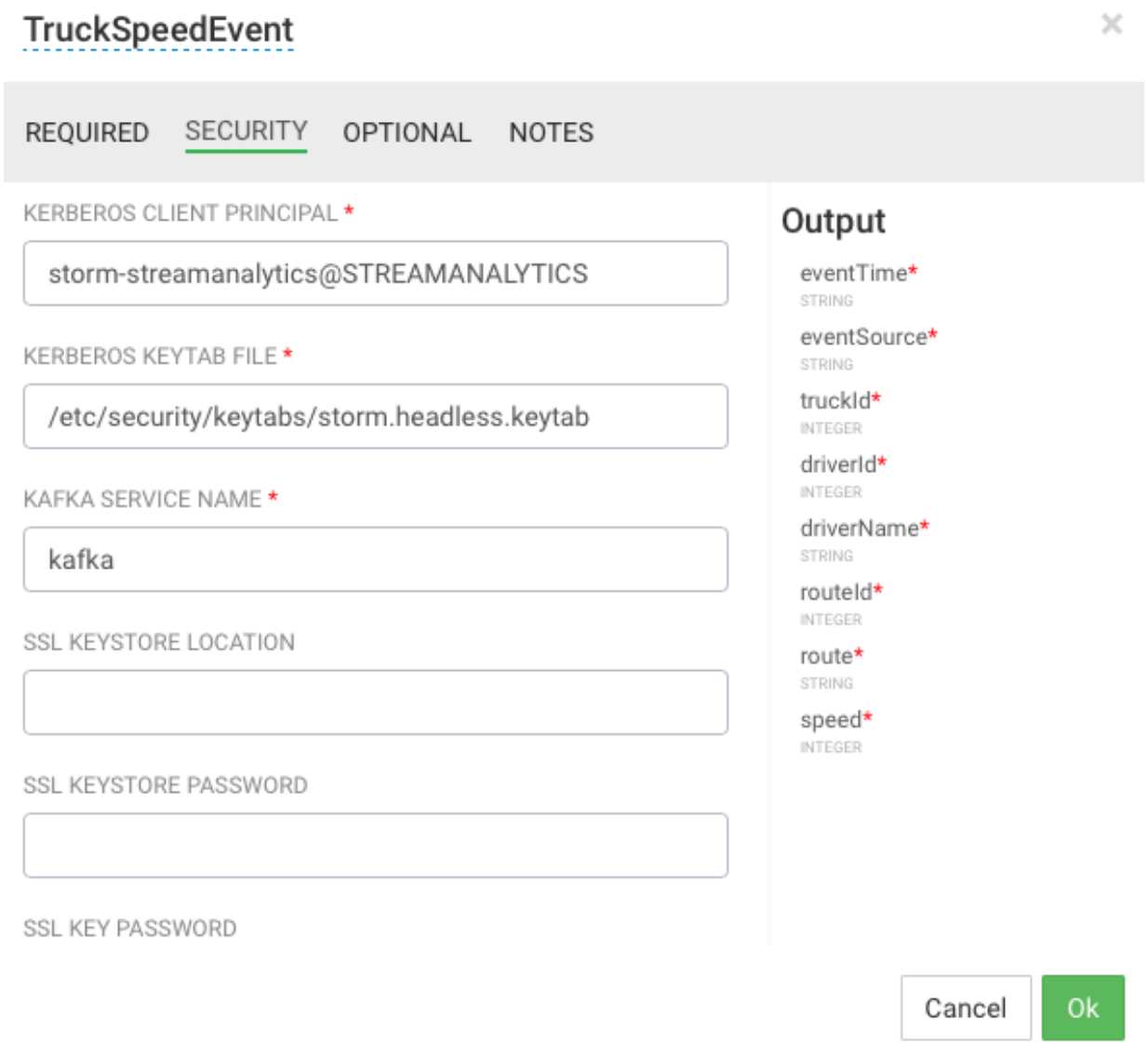
Result

When user X deploys the application, Nimbus uses the principal and the keytab configured above to impersonate user X when interacting with the big data services in the application.

Connecting to Secure Kafka

Kafka does not support delegation tokens. You must configure the Kafka source/sink processor with the principal and keytab used to authenticate the stream applications with Kafka. Use these steps to configure SAM to communication with a secure Kafka service.

1. Double click on the Kafka source/sink component on the canvas.
2. Select the **Security** tab.
3. Configure the Kerberos client principal, Kerberos keytab file, and the Kafka service name. The client principal and keytab selected must exist on all the worker nodes in the cluster. Using the storm-<cluster-name> principal is recommended because Ambari creates that keytab on each worker node when running the Ambari Kerberos Wizard . Set the Kafka service name to “kafka”.



Securing SAM – An End-to-End Workflow

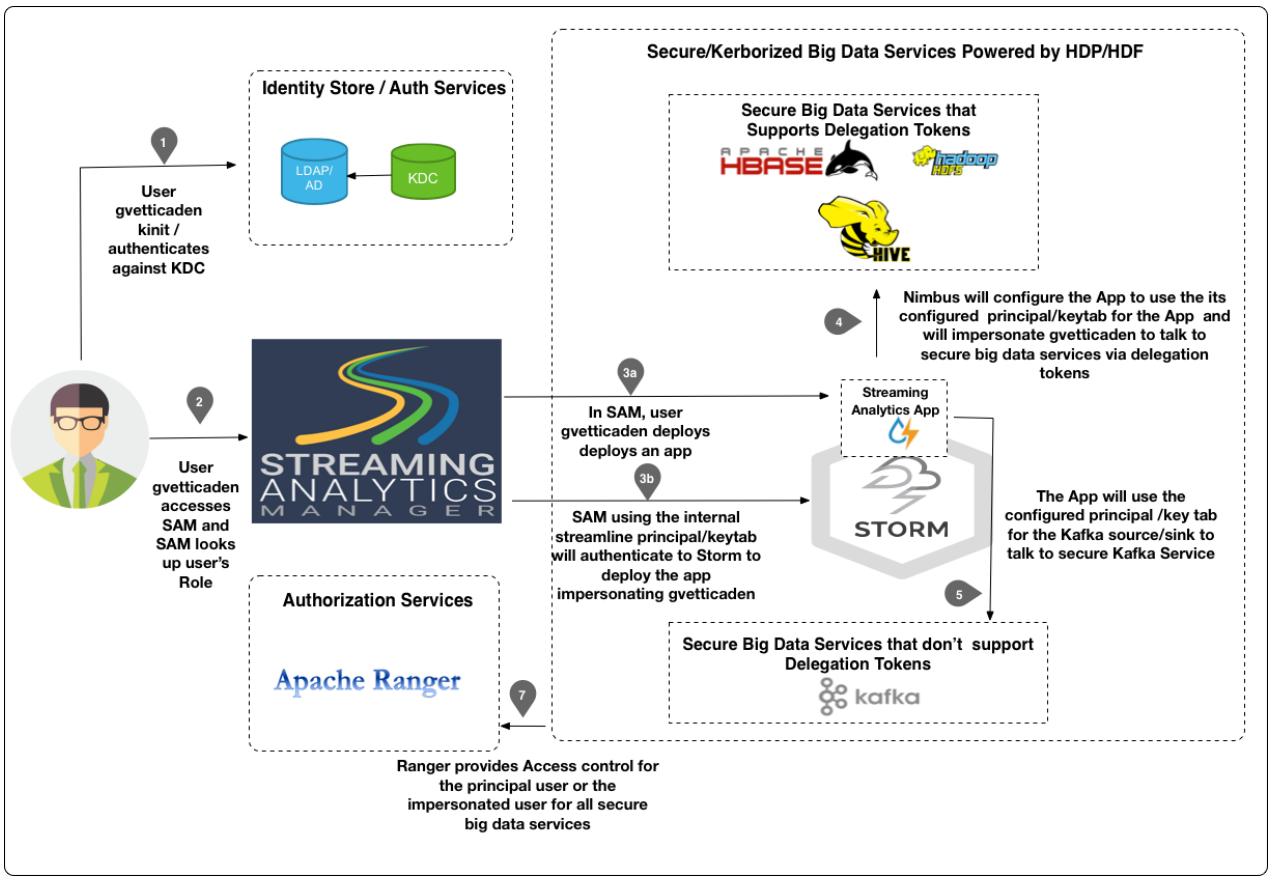
To help understand how all of this fits together, let's walk through a use case to see how SAM to deploys applications in a secure cluster.

The use case details are the following:

- An organization has a secure HDF/HDP cluster and all cluster services have been Kerberize.
- User gvetticaden is a developer and part of the release engineering team, builds a streaming application that includes of the following capabilities:
  - Creating streams from a set of Kafka topics from a secure Kafka Broker.
  - Doing analytics on the stream.
  - Persisting different events to following secure data stores: HDFS, HBase, Hive
- User gvetticaden wants to deploy the streaming application to a secure storm Service.

Understanding the End-to-End Workflow

The below image provides an explanation on how SAM functions for the above use case.



Step 1: Initial Login

User gvetticaden authenticates himself to the organization AD/KDC by doing a kinit. Typically in an organization, the ticket is granted when the user logs into the corporate LAN.

Principal/Keytab Used to Connct: gvetticaden

Step 2: SAM Grants Access Based on Roles and Permissions

SAM looks up the roles for gvetticaden. Based on the permissions associated with the roles, SAM gives gvetticaden access to specific features.

Step 3a: Build and Deploy a Streaming Application

User gvetticaden builds the streaming analytics application and deploys it. The application includes the following capabilities:

- Creating streams from a set of Kafka topics from a secure Kafka Broker.
- Doing analytics on the stream.
- Persisting different events to following secure data stores: HDFS, HBase, Hive

Step 3b: SAM Communicates with Storm

SAM communicates with Storm Streaming Engine to deploy the stream application using the streamline principal/keytab. SAM is functioning as a client submitting a job to Secure Storm. The internal streamline user will impersonate gvetticaden when it talks to Storm. Hence ACLs within Ranger for Storm can be configured for gvetticaden, the person deploying the streaming application.

Principal/Keytab Used to Connect: The streamline principal/keytab is used to connect, and user gvetticaden is impersonated.

Step 4: Communication with Secured Big Data Services

When SAM deploys the application, it passes the application principal and keytab to Nimbus. Nimbus uses this principal to authenticate to big data services that support tokens. The principal impersonates gvetticaden. The result is that all Ranger ACLs for HBase, Hive, and HDFS are configured for gvetticadne, the user deploying the streaming application.

Step 5: Communication with Secured Big Data Services that do not Support Delegation Tokens

If the application uses a Kafka Source or Sink, then the application uses the principal and keytab configured under the Kafka component security settings.

Principal/Keytab Used to Connect:: The principal/keytab configured in Kafka are used to connect.