Hortonworks DataFlow

Security

Hortonworks DataFlow: Security

Copyright © 2012-2018 Hortonworks, Inc. Some rights reserved.

Table of Contents

List of Tables

1. Enabling Kerberos

To enable Kerberos on Ambari, complete the following steps:

1. Installing and Configuring the KDC [1]

2. Installing the JCE [6]

3. Enabling Kerberos on Ambari [7]

4. Cluster Component Configuration Updates [8]

1.1. Installing and Configuring the KDC

Ambari is able to configure Kerberos in the cluster to work with an existing MIT KDC, or existing Active Directory installation. This section describes the steps necessary to prepare for this integration.

Note

If you do not have an existing KDC (MIT or Active Directory), Install a new MIT KDC. Installing a KDC on a cluster host *after* installing the Kerberos client may overwrite the krb5.conf file generated by Ambari.

You can choose to have Ambari connect to the KDC and automatically create the necessary Service and Ambari principals, generate and distribute the keytabs ("Automated Kerberos Setup"). Ambari also provides an advanced option to manually configure Kerberos. If you choose this option, you must create the principals, generate and distribute the keytabs. Ambari will not do this automatically ("Manual Kerberos Setup").

• Use an Existing MIT KDC [1]

• Use an Existing Active Directory [2]

• Use Manual Kerberos Setup [2]

For convenience, use the instructions to (Optional) Install a new MIT KDC if you do not have an existing KDC available.

1.1.1. Use an Existing MIT KDC

To use an existing MIT KDC for the cluster, you must prepare the following:

• Ambari Server and cluster hosts have network access to both the KDC and KDC admin hosts.

• KDC administrative credentials are on-hand.

Note

You will be prompted to enter the KDC Admin Account credentials during the Kerberos setup so that Ambari can contact the KDC and perform the necessary

principal and keytab generation. By default, Ambari will not retain the KDC credentials unless you have configured Ambari for encrypted passwords.

1.1.2. Use an Existing Active Directory

To use an existing Active Directory domain for the cluster with Automated Kerberos Setup, you must prepare the following:

• Ambari Server and cluster hosts have network access to, and be able to resolve the DNS names of, the Domain Controllers.

• Active Directory secure LDAP (LDAPS) connectivity has been configured.

• Active Directory User container for principals has been created and is on-hand. For example, "OU=Hadoop,OU=People,dc=apache,dc=org"

• Active Directory administrative credentials with delegated control of "Create, delete, and manage user accounts" on the previously mentioned User container are on-hand.

Note

You will be prompted to enter the KDC Admin Account credentials during the Kerberos setup so that Ambari can contact the KDC and perform the necessary principal and keytab generation. By default, Ambari will not retain the KDC credentials unless you have configured Ambari for encrypted passwords.

Note

If Centrify is installed and and being used on any of the servers in the cluster, it is critical that you refer to Centrify's integration guide before attempting to enable Kerberos Security on your cluster. The documentation can be found in the Centrify Server Suite documentation library, with a direct link to the Hortonworks specific PDF here.

1.1.3. Use Manual Kerberos Setup

To perform Manual Kerberos Setup, you must prepare the following:

• Cluster hosts have network access to the KDC.

• Kerberos client utilities (such as kinit) have been installed on every cluster host.

• The Java Cryptography Extensions (JCE) have been setup on the Ambari Server host and all hosts in the cluster.

• The Service and Ambari Principals will be manually created in the KDC before completing this wizard.

• The keytabs for the Service and Ambari Principals will be manually created and distributed to cluster hosts before completing this wizard.

1.1.4. (Optional) Install a new MIT KDC

The following gives a very high level description of the KDC installation process. To get more information see specific Operating Systems documentation, such as RHEL documentation, CentOS documentation, or SLES documentation.

Note

> Because Kerberos is a time-sensitive protocol, all hosts in the realm must be time-synchronized, for example, by using the Network Time Protocol (NTP). If the local system time of a client differs from that of the KDC by as little as 5 minutes (the default), the client will not be able to authenticate.

Install the KDC Server

1. Install a new version of the KDC server:

   RHEL/CentOS/Oracle Linux

   ```
   yum install krb5-server krb5-libs krb5-workstation
   ```

   SLES

   ```
   zypper install krb5 krb5-server krb5-client
   ```

   Ubuntu/Debian

   ```
   apt-get install krb5-kdc krb5-admin-server
   ```

2. Using a text editor, open the KDC server configuration file, located by default here:

   ```
   vi /etc/krb5.conf
   ```

3. Change the [realms] section of this file by replacing the default "kerberos.example.com" setting for the kdc and admin_server properties with the Fully Qualified Domain Name of the KDC server host. In the following example, "kerberos.example.com" has been replaced with "my.kdc.server".

   ```
   [realms]
    EXAMPLE.COM = {
      kdc = my.kdc.server
      admin_server = my.kdc.server
   }
   ```

4. Some components such as HUE require renewable tickets. To configure MIT KDC to support them, ensure the following settings are specified in the `libdefaults` section of the `/etc/krb5.conf` file.

   ```
   renew_lifetime = 7d
   ```

   Note

   > For Ubuntu/Debian, the setup of the default realm for the KDC and KDC Admin hostnames is performed during the KDC server install. You can re-run

setup using dpkg-reconfigure krb5-kdc. Therefore, Steps 2 and 3 above are not needed for Ubuntu/Debian.

Create the Kerberos Database

• Use the utility kdb5_util to create the Kerberos database.

RHEL/CentOS/Oracle Linux

```
kdb5_util create -s
```

SLES

```
kdb5_util create -s
```

Ubuntu/Debian

```
krb5_newrealm
```

Start the KDC

• Start the KDC server and the KDC admin server.

RHEL/CentOS/Oracle Linux 6

```
/etc/rc.d/init.d/krb5kdc start
```

```
/etc/rc.d/init.d/kadmin start
```

RHEL/CentOS/Oracle Linux 7

```
systemctl start krb5kdc
```

```
systemctl start kadmin
```

SLES

```
rckrb5kdc start
```

```
rckadmind start
```

Ubuntu/Debian

```
service krb5-kdc restart
```

```
service krb5-admin-server restart
```

> ⚠️ **Important**
>
> When installing and managing your own MIT KDC, it is **very important** to **set up the KDC server to auto-start on boot**. For example:
>
> **RHEL/CentOS/Oracle Linux 6**
>
> ```
> chkconfig krb5kdc on
> ```

```
chkconfig kadmin on
```

RHEL/CentOS/Oracle Linux 7

```
systemctl enable krb5kdc
```

```
systemctl enable kadmin
```

SLES

```
chkconfig rckrb5kdc on
```

```
chkconfig rckadmind on
```

Create a Kerberos Admin

Kerberos principals can be created either on the KDC machine itself or through the network, using an "admin" principal. The following instructions assume you are using the KDC machine and using the `kadmin.local` command line administration utility. Using `kadmin.local` on the KDC machine allows you to create principals without needing to create a separate "admin" principal before you start.

> **Note**
>
> You will need to provide these admin account credentials to Ambari when enabling Kerberos. This allows Ambari to connect to the KDC, create the cluster principals and generate the keytabs.

1. Create a KDC admin by creating an admin principal.

   ```
   kadmin.local -q "addprinc admin/admin"
   ```

2. Confirm that this admin principal has permissions in the KDC ACL. Using a text editor, open the KDC ACL file:

   RHEL/CentOS/Oracle Linux

   ```
   vi /var/kerberos/krb5kdc/kadm5.acl
   ```

   SLES

   ```
   vi /var/lib/kerberos/krb5kdc/kadm5.acl
   ```

   Ubuntu/Debian

   ```
   vi /etc/krb5kdc/kadm5.acl
   ```

3. Ensure that the KDC ACL file includes an entry so to allow the admin principal to administer the KDC for your specific realm. When using a realm that is different than EXAMPLE.COM, **be sure there is an entry for the realm you are using**. If not present, principal creation will fail. For example, for an admin/admin@HADOOP.COM principal, you should have an entry:

   ```
   */admin@HADOOP.COM *
   ```

4. After editing and saving the kadm5.acl file, you must restart the kadmin process.

RHEL/CentOS/Oracle Linux 6

```
/etc/rc.d/init.d/kadmin restart
```

RHEL/CentOS/Oracle Linux 7

```
systemctl restart kadmin
```

SLES

```
rckadmind restart
```

Ubuntu/Debian

```
service krb5-admin-server restart
```

1.2. Installing the JCE

Before enabling Kerberos in the cluster, you must deploy the Java Cryptography Extension (JCE) security policy files on the Ambari Server and on all hosts in the cluster.

> **Important**
>
> If you are using Oracle JDK, **you must** distribute and install the JCE **on all hosts** in the cluster, including the Ambari Server. **Be sure to restart Ambari Server after installng the JCE**. If you are using OpenJDK, some distributions of the OpenJDK come with unlimited strength JCE automatically and therefore, installation of JCE is not required.

1.2.1. Install the JCE

1. On the Ambari Server, obtain the JCE policy file appropriate for the JDK version in your cluster.

   • For Oracle JDK 1.8:

     http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

   • For Oracle JDK 1.7:

     http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

2. Save the policy file archive in a temporary location.

3. On Ambari Server and on each host in the cluster, add the unlimited security policy JCE jars to `$JAVA_HOME/jre/lib/security/`.

   For example, run the following to extract the policy jars into the JDK installed on your host:

```
unzip -o -j -q jce_policy-8.zip -d /usr/jdk64/jdk1.8.0_60/jre/lib/security/
```

4. Restart Ambari Server.

1.3. Enabling Kerberos on Ambari

Once you have completed the prerequisites, you are ready to enable Kerberos for Ambari.

1. From the Ambari UI, click **Admin**, and select **Kerberos**.
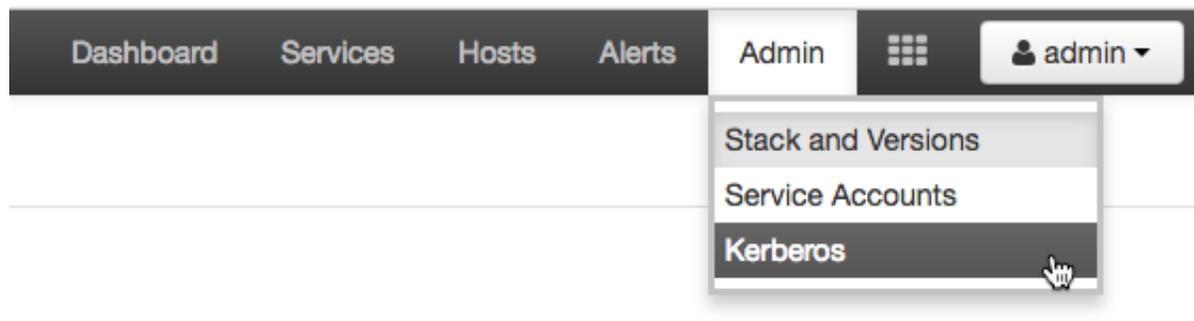
2. Click **Enable Kerberos** to launch the **Enable Kerberos Wizard**.

3. From the **Get Started** screen, select the type of KDC you want to use.

4. Provide information about the KDC and admin account.

    a. In the **KDC** section, enter the following information:

    - In the **KDC Host** field, the IP address or FQDN for the KDC host. Optionally a port number may be included.

    - In the **Realm name** field, the default realm to use when creating service principals.

    - (Optional) In the **Domains** field, provide a list of patterns to use to map hosts in the cluster to the appropriate realm. For example, if your hosts have a common domain in their FQDN such as host1.hortonworks.local and host2.hortonworks.local, you would set this to:

      ```
      .hortonworks.local,hortonworks.local
      ```

    b. In the **Kadmin** section, enter the following information:

    - In the **Kadmin Host** field, the IP address or FQDN for the KDC administrative host. Optionally a port number may be included.

    - The **Admin principal** and **password** that will be used to create principals and keytabs.

    - (Optional) If you have configured Ambari for encrypted passwords, the **Save Admin Credentials** option will be enabled. With this option, you can have Ambari store the KDC Admin credentials to use when making cluster changes.

5. From the **Install and Test Kerberos Client** page, proceed with the install. Click **Next** when complete.

6. From the **Configure Identities** page, you can customize the Kerberos identities as needed, and proceed to kerberize the cluster.

   Be sure to review the principal names, particularly the **Ambari Principals** on the **General** tab. These principal names, by default, append the name of the cluster to each of the Ambari principals. You can leave this as default or adjust these by removing the "-${**cluster-name**}" from principal name string.

   Click the **Advanced** tab to review the principals and keytabs for each service.

7. Confirm your configurations, and click next to proceed kerberizing your cluster.

1.4. Cluster Component Configuration Updates

After you have enabled Kerberos, some cluster components require additional configuration updates.

SAM Configuration Changes

Steps

1. From the Ambari UI, select **Admin | Kerberos**.

2. From **Ambari Principals**, set the SAM principal name:

```
streamline_principal_name : memo the principal without @${realm} — (1)
```

3. From **Ambari Infra | Configs | Advanced | Advanced infra-solr-security-json**, add the following to the bottom text box. Replace <<streamline>> (bolded) to streamline principal (1) before pasting.

```
{
  "authentication": {
    "class": "org.apache.solr.security.KerberosPlugin"
  },
  "authorization": {
    "class": "org.apache.ambari.infra.security.
InfraRuleBasedAuthorizationPlugin",
    "user-role": {
      "{{infra_solr_kerberos_service_user}}@{{kerberos_realm}}": "admin",
      "{{logsearch_kerberos_service_user}}@{{kerberos_realm}}":
 ["{{infra_solr_role_logsearch}}", "{{infra_solr_role_ranger_admin}}",
 "{{infra_solr_role_dev}}"],
      "<<streamline>>@{{kerberos_realm}}": ["{{infra_solr_role_logsearch}}",
 "{{infra_solr_role_ranger_admin}}", "{{infra_solr_role_dev}}"],
      "{{logfeeder_kerberos_service_user}}@{{kerberos_realm}}":
 ["{{infra_solr_role_logfeeder}}", "{{infra_solr_role_dev}}"],
      "{{atlas_kerberos_service_user}}@{{kerberos_realm}}":
 ["{{infra_solr_role_atlas}}", "{{infra_solr_role_ranger_audit}}",
 "{{infra_solr_role_dev}}"],
{% if infra_solr_ranger_audit_service_users %}
{%   for ranger_audit_service_user in infra_solr_ranger_audit_service_users
 %}
      "{{ranger_audit_service_user}}@{{kerberos_realm}}":
 ["{{infra_solr_role_ranger_audit}}", "{{infra_solr_role_dev}}"],
{%   endfor %}
{% endif %}
      "{{ranger_admin_kerberos_service_user}}@{{kerberos_realm}}":
 ["{{infra_solr_role_ranger_admin}}", "{{infra_solr_role_ranger_audit}}",
 "{{infra_solr_role_dev}}"]
    },
    "permissions": [
      {
        "name" : "collection-admin-read",
        "role" :null
      },
      {
        "name" : "collection-admin-edit",
        "role" : ["admin", "{{infra_solr_role_logsearch}}",
 "{{infra_solr_role_logfeeder}}", "{{infra_solr_role_atlas}}",
 "{{infra_solr_role_ranger_admin}}"]
      },
      {
        "name":"read",
        "role": "{{infra_solr_role_dev}}"
      },
      {
        "collection": ["{{logsearch_service_logs_collection}}",
 "{{logsearch_audit_logs_collection}}", "history"],
        "role": ["admin", "{{infra_solr_role_logsearch}}",
 "{{infra_solr_role_logfeeder}}"],
        "name": "logsearch-manager",
        "path": "/*"
      },
      {
        "collection": ["vertex_index", "edge_index", "fulltext_index"],
        "role": ["admin", "{{infra_solr_role_atlas}}"],
        "name": "atlas-manager",
        "path": "/*"
      },
      {
```

```
        "collection": "{{ranger_solr_collection_name}}",
        "role": ["admin", "{{infra_solr_role_ranger_admin}}",
  "{{infra_solr_role_ranger_audit}}"],
        "name": "ranger-manager",
        "path": "/*"
      }
    ]
  }
}
```

4. Restart Ambari Infra.

Druid Configuration Changes

Update the Druid property druid.hadoop.security.spnego.excludedPaths to the following value:

```
["/status", "/druid/worker/v1", "/druid/indexer/v1"]
```

HDFS Configuration Changes

If you are going to use the HDFS processor in your application in secure mode, add the following properties in the HDFS service under `custom core-site.xml`.

Property Name	Value
hadoop.proxyuser. $principal_you_configured_in_sam_app_settings.groups	*
hadoop.proxyuser. $principal_you_configured_in_sam_app_settings.hosts	*

Example

In SAM, you have configured the following principal and keytab under Application Settings:

The configuration for the 2 HDF properties is:

```
hadoop.proxyuser.storm-streamanalytics.hosts=*
hadoop.proxyuser.storm-streamanalytics.groups=*
```

HBase Configuration

In the HBase service, under custom `hbase-site.xml` add the following properties

- hbase.thrift.support.proxyuser=true

- hbase.regionserver.thrift.http=true

In HDFS service, add the following under custom core-site.xm

- hadoop.proxyuser.streamline-streamanalytics.hosts=*

- hadoop.proxyuser.streamline-streamanalytics.groups=*

2. NiFi Authentication

After you have installed Ambari and the HDF stack, you have a 2 options for enabling SSL for your NiFi services.

- Enabling SSL with a NiFi Certificate Authority [13]

- Enabling SSL with Existing Certificates [14]

You can use the NiFi service Configs tab Advanced nifi-ambari-ssl-config dialog to configure security for these options.

To access the NiFi SSL configuration dialog:

1. From the Ambari services column, click NiFi.

2. Click the Configs tabs.

3. Click Advanced nifi-ambari-ssl-config.

2.1. Enabling SSL with a NiFi Certificate Authority

When you enable SSL with the NiFi Certificate Authority (CA) installed, the NiFi CA generates new client certificates for you through Ambari. If you want to enable SSL with a NiFi CA installed, and are planning to use Ranger to manage authorization:

1. Select the **Enable SSL?** box.

2. Specify the **NiFi CA** token.

If you want to enable SSL with a NiFi CA installed, and are not yet using Ranger to manage authorization:

1. Check the **Enable SSL?** box.

2. Specify the **NiFi CA Token**.

3. Verify that the `authorizations.xml` file on each node does not contain policies. The `authorizations.xml` is located in `{nifi_internal_dir}/conf`. By default, this location is `/var/lib/nifi/conf/`, and the value of `{nifi_internal_dir}` is specified in the **NiFi internal dir** field under **Advanced nifi-ambari-config**.

> **Note**
>
> If `authorizations.xml` does contain policies, you must delete it from each node. If you do not, your **Initial Admin Identity** and **Node Identities** changes do not take effect.

4. Specify the **Initial Admin Identity**. The **Initial Admin Identity** is the identity of an initial administrator and is granted access to the UI and has the ability to create additional users, groups, and policies. **This is a required value** when you are not using the Ranger plugin for NiFi for authorization.

   The **Initial Admin Identity** format is CN=admin, OU=NIFI.

   After you have added the **Initial Admin Identity**, you must immediately generate certificate for this user.

5. Specify the **Node Identities**. This indicates the identity of each node in a NiFi cluster and allows clustered nodes to communicate. **This is a required value** when you are not using the Ranger plugin for NiFi for authorization.

   ```
   <property name="Node Identity 1">CN=node1.fqdn, OU=NIFI</property>
   <property name="Node Identity 2">CN=node2.fqdn, OU=NIFI</property>
   <property name="Node Identity 3">CN=node3.fqdn, OU=NIFI</property>
   ```

   Replace node1.fqdn, node2.fqdn, and node3.fqdn with their respective fully qualified domain names.

2.2. Enabling SSL with Existing Certificates

If you want to enable SSL with existing certificates, and plan to use Ranger for authorization:

1. Check the **Enable SSL?** box.

2. Set **Keystore path**, **Keystore password**, and **Keystore type** values.

   The keystore path is similar to: /etc/security/nifi-certs/keystore.jks

3. Set the **Truststore path**, **Truststore password**, and **Truststore type** values.

   The truststore path is similar to: /etc/security/nifi-certs/truststore.jks

4. Check **Clients need to authenticate?** if you want to ensure that nodes in the cluster are authenticated and are required to have certificates that are trusted by the truststores.

If you want to enable SSL with existing certificates, and are not yet using Ranger for authorization:

1. Check the **Enable SSL?** box.

2. Set **Keystore path**, **Keystore password**, and **Keystore type** values.

   The keystore path is similar to: /etc/security/nifi-certs/keystore.jks

3. Set the **Truststore path**, **Truststore password**, and **Truststore type** values.

   The truststore path is similar to: /etc/security/nifi-certs/truststore.jks

4. Check **Clients need to authenticate?** to ensure that nodes in the cluster are authenticated and are required to have certificates that are trusted by the Truststores.

5. Specify the **Initial Admin Identity**. The **Initial Admin Identity** is the identity of an initial administrator and is granted access to the UI and has the ability to create additional users, groups, and policies. **This is a required value** when you are not using the Ranger plugin for NiFi for authorization.

   The **Initial Admin Identity** format is CN=admin, OU=NIFI.

   After you have added the **Initial Admin Identity**, you must immediately generate certificate for this user.

6. Specify the **Node Identities**. This indicates the identity of each node in a NiFi cluster and allows clustered nodes to communicate. **This is a required value** when you are not using the Ranger plugin for NiFi for authorization.

   ```
   <property name="Node Identity 1">CN=node1.fqdn, OU=NIFI</property>
   <property name="Node Identity 2">CN=node2.fqdn, OU=NIFI</property>
   <property name="Node Identity 3">CN=node3.fqdn, OU=NIFI</property>
   ```

   Replace node1.fqdn, node2.fqdn, and node3.fqdn with their respective fully qualified domain names.

2.3. (Optional) Setting Up Identity Mapping

About This Task

You can use identity mapping properties to normalize user identities. Once you set up identity mapping, NiFi treats identities authenticated by different identity providers (certificates, LDAP, Kerberos) the same. This allows you to avoid creating duplicate users. Additionally, you only need to set up user-specific configurations such as authorizations once per user.

Steps

1. From the NiFi service **Configs** tab, click **Advanced nifi-properties**.

2. Use the Filter box to search for nifi.security.identity.mapping.pattern.

3. Enter the following values:

Table 2.1. Identity mapping values

Field	Sample value
nifi.security.identity.mapping.pattern.dn	^CN=(.*?), OU=(.*?)$
nifi.security.identity.mapping.value.dn	$1@$2
nifi.security.identity.mapping.pattern.kerb	^(.*?)/instance@(.*?)$
nifi.security.identity.mapping.value.kerb	$1@$2

4. Click **Save**.

5. Restart NiFi using the **Restart all Required** option from the **Action** menu.

Example

The following examples demonstrate normalizing DNs from certificates and principals from Kerberos:

```
nifi.security.identity.mapping.pattern.dn=^CN=(.*?), OU=(.*?), O=(.*?), L=(.*?
), ST=(.*?), C=(.*?)$
nifi.security.identity.mapping.value.dn=$1@$2
nifi.security.identity.mapping.pattern.kerb=^(.*?)/instance@(.*?)$
nifi.security.identity.mapping.value.kerb=$1@$2
```

2.4. Generating Client Certificates

If you are using a CA, you can use the TLS Toolkit provided in the HDF management pack to generate the required client certificates so that you can log into NiFi after enabling SSL.

1. Navigate the TLS Toolkit directory, which will be similar to:

```
cd /var/lib/ambari-agent/cache/common-services/NIFI/1.0.0/package/files/
nifi-toolkit-$version
```

For example:

```
cd /var/lib/ambari-agent/cache/common-services/NIFI/1.0.0/package/files/
nifi-toolkit-1.1.0.2.1.3.0-6
```

2. From the command line, run the following:

```
bin/tls-toolkit.sh client
-c <CA host name>
-D "<distinguished name>"
-p <CA host port>
-t <NiFi CA token>
-T <keystore type>
```

Your command should look similar to:

```
bin/tls-toolkit.sh client
-c nifi.cert.authority.example.com
-D "CN=admin, OU=NIFI"
-t nifi
-p 10443
-T pkcs12
```

3. To get your keystore password, enter:

```
cat config.json
```

4. Verify that the installation directory contains the following two files:

- `keystore.pkcs12`

- `nifi-cert.pem`

5. To double-click your keystore file to launch your OS certificate management application, change `keystore.pkcs12` to `keystore.p12`.

6. Import the `nifi-cert.pem` file as your trusted CA.

7. Import `keystore.pkcs12` as the client certificate.

Re-running the TLS Toolkit generates a new set of keystore and configuration files. To avoid having your files overwritten, save the keystore and configuration files to an alternate location before re-running the TLS Toolkit.

For more information about the TLS Toolkit, see TLS Generation Toolkit in the *Administration Guide*.

2.5. Logging into NiFi After Enabling SSL

Now that you have set up SSL, you need to enable logging into NiFi with a certificate:

1. Launch NiFi from the Ambari **Quick Links** menu.

2. Select the certificate you just imported from the browser prompt.

3. Log in with the user name and password you created during installation.

> **Note**
>
> When you are running NiFi on a host with Ambari and with SSL enabled, the default URL becomes secured **https**://<local-host>:**9091**/nifi.

3. Proxying NiFi with Apache Knox

You can use the Apache Knox gateway to provide authentication access security for your NiFi services. The Apache Knox Gateway ("Knox") is a system to extend the reach of Apache™ Hadoop® services to users outside of a Hadoop cluster without reducing Hadoop Security. Knox also simplifies Hadoop security for users who access the cluster data and execute jobs. Knox integrates with Identity Management and SSO systems used in enterprises and allows identity from these systems to be used for access to Hadoop clusters. For more information about Knox, see Apache Knox Gateway Overview.

> **Note**
>
> HA is not currently supported for proxying NiFi in Knox in HDF 3.1.

Installing NiFi authentication with Knox requires the following steps:

- Prerequisites [18]

- Configuring Knox for NiFi [20]

- Accessing NiFi Via Knox [23]

3.1. Prerequisites

To use NiFi with Apache Knox, you must meet the following prerequisites:

Install HDP	Apache Knox is part of the HDP stack. For information on installing HDP, see Getting Ready.
Install Knox on an HDP cluster	For information on installing services using Ambari, see Choose Services.
Install NiFi on the HDP cluster	**⚠ Important** We recommend that NiFi is installed on a different server than Knox. The procedure for installing NiFi on a HDP cluster depends on whether the HDP installation is a new one or an upgrade. See the following section that is appropriate for your setup: • Installing NiFi on a New HDP Cluster [19] • Installing NiFi on an Upgraded HDP Cluster [19]
Configure Knox on the HDP cluster	Prior to enabling access to NiFi, you must configure Knox on the HDP cluster. See the following for information on Configuring the Knox Gateway.
Configure Knox authentication	You need to define policies for the Knox node user and any users accessing NiFi through Knox to allow NiFi to authorize the requests from Knox. For information

on configuring Knox authentication, see Configuring Authentication.

3.1.1. Installing NiFi on a New HDP Cluster

If you are installing NiFi on a new HDP installation, as opposed to an upgraded HDP installation, follow the instructions at Installing NiFi on a New HDP Cluster.

In addition to the steps found in Installing NiFi on a New HDP Cluster, make the following changes:

1. In addition to other settings for the **Advanced nifi-ambari-ssl-config**, add a node identity for the Knox node.

   For example:

   • <property name="Node Identity 1">CN=$NIFI_HOSTNAME, OU=NIFI</property>

   • <property name="Node Identity 2">CN=$NIFI_HOSTNAME, OU=NIFI</property>

   • <property name="Node Identity 3">CN=$NIFI_HOSTNAME, OU=NIFI</property>

   • <property name="Node Identity 4">CN=$KNOX_HOSTNAME, OU=KNOX</property>

2. Add a new property to the Advanced NiFi Properties in Ambari:

   ```
   nifi.web.proxy.context.path=$GATEWAY_CONTEXT/default/nifi-app
   ```

   Refer to the Knox configuration in Ambari, `$GATEWAY_CONTEXT` to determine what the path is for NiFi.

3. Set the host header to match the host port being used by Knox.

   In the **Advanced topology** section in Ambari, add the following information to the **nifi.web.proxy.host** property:

   ```
   $HOSTNAME:$PORTNUMBER
   ```

3.1.2. Installing NiFi on an Upgraded HDP Cluster

If your HDP installation is an upgrade rather than a new installation, follow the instructions at Installing HDF Services on an Existing HDP Cluster.

You will also need to manually edit policies for the users who are going to log into Knox and for the Knox node itself to be authorized as a proxy. At a minimum, the Knox node should be added to the policy for proxying user requests. To do this, complete the steps in one of the following sections:

• Adding a Policy Using NiFi [19]

• Adding a Policy Using Ranger [20]

3.1.2.1. Adding a Policy Using NiFi

If you are using native NiFi, complete the following steps to add a policy for the Knox node user. For more information, see Configuring Users and Access Policies.

1. Add a user for the Knox node.

2. For the component level policies, on the root group, add permissions to "view the data policy" and to "modify the data".

3.1.2.2. Adding a Policy Using Ranger

If you are using Ranger, complete the following steps to add a policy for the Knox node user. For more information, see Creating Policies to View NiFi.

1. Add a user for the Knox node.

2. For the component level policies, on the root group, add permissions to "view the data policy" and to "modify the data".

   To add these permissions:

   • Set WRITE to `/proxy`

   • Set READ and WRITE to `/data/process-group/<root-group-id>`

3.2. Configuring Knox for NiFi

1. Create a `config.json` file in a location accessible to Knox.

   For example, create the file on the Knox server at `/home/knox`.

2. Populate the `config.json` file with the following information:

   Look up the NiFi CA port value before populating the `config.json` file which is in the **Advanced nifi-ambari-ssl-config**.

   ```
   {
     "dn" : "CN=$KNOX_HOST_NAME, OU=KNOX",
     "keyStore" : "/home/knox/knox-nifi-keystore.jks",
     "keyStoreType" : "jks",
     "keyStorePassword" : "$KEY_STORE_PASSWORD",
     "keyPassword" : "$KEY_PASSWORD",
     "token" : "$NIFI_CA_TOKEN_VALUE",
     "caHostname" : "$NIFI_CA_HOSTNAME",
     "port" : $NIFI_CA_PORT,
     "trustStore" : "/home/knox/knox-nifi-truststore.jks",
     "trustStorePassword" : "$TRUSTSTORE_PASSWORD",
     "trustStoreType" : "jks"
   }
   ```

   The keyStorePassword, keyPassword, and trustStorePassword can be set to the Knox Master Password to make it easier to import the keyStore and trustStore created by the NiFi Certificate Authority into Knox's keystore.

3. Confirm that the variables in the `json.config` file from step 2 are set to the values from the **Advanced nifi-ambari-ssl-config**.

4. To create the keystore and truststore used by Knox when proxying NiFI, start the NiFi TLS Toolkit. For the location of the TLS Toolkit, see Release Notes for OS-specific information.

For example:

```
/var/lib/ambari-agent/cache/common-services/NIFI/1.0.0/package/
files/nifi-toolkit-1.2.0.3.0.2.0-71/bin/tls-toolkit.sh client --
subjectAlternativeNames "CN=$HOST_DN_NAME, OU=KNOX" -F -f /home/
knox/config.json
```

The toolkit requests a new certificate and creates two new files containing the keystore and truststore:

```
/home/knox/knox-nifi-keystore.jks
/home/knox/knox-nifi-truststore.jks
```

5. Import the Knox certificate for NiFi into Knox's `gateway.jks` file:

```
keytool -importkeystore -srckeystore /home/knox/knox-nifi-
keystore.jks -destkeystore /usr/hdp/current/knox-server/data/
security/keystores/gateway.jks -deststoretype JKS -srcstorepass
$KEYSTORE_PASSWORD -deststorepass $KNOX_MASTER_PASSWORD
```

The `gateway.jks` file should now contain a PrivateKeyEntry for NiFi.

6. Import NiFi CA's truststore into Knox's `gateway.jks` file:

```
keytool -importkeystore -srckeystore /home/knox/knox-nifi-
truststore.jks -destkeystore /usr/hdp/current/knox-server/data/
security/keystores/gateway.jks -deststoretype JKS -srcstorepass
$KEYSTORE_PASSWORD -deststorepass $KNOX_MASTER_PASSWORD
```

The `gateway.jks` file should now contain a trustedCertEntry for NiFi.

Knox uses the `gateway.jks` to look up certificates in the truststore that it can trust.

7. If you have not already done so, verify the proper keys are in the `gateway.jks` file:

```
keytool -keystore /usr/hdp/current/knox server/data/security/
keystores/gateway.jks -storepass $KEYSTORE_PASSWORD -list -v
```

8. In the `/usr/hdp/current/$KNOX_SERVER/data/services/nifi/1.4.0/`
`service.xml` file, make sure the dispatch element contains the following:

```
<dispatch
classname="org.apache.hadoop.gateway.dispatch.NiFiDispatch" use-
two-way-ssl="true"/>
```

9. In the **Advanced topology** section in Ambari, add the following service definition. Add it to the list of services.

```
<service>
    <role>NIFI</role>
    <url>https://$NIFI_HOSTNAME:$NIFI_HTTPS_PORT/</url>
    <param name="useTwoWaySsl" value="true" />
</service>
```

Where:

- `<url>` Points to the host and port that NiFi is listening on from the `nifi-properties` configuration site.

10.If you want to use Knox SSO authentication, perform the following steps:

a. In Ambari, replace the ShiroProvider with the KnoxSSO provider in the **nifi.security.knox.url** property.

The following information should be in the **nifi.security.knox.url** property:

```
<provider>
        <role>federation</role>
        <name>SSOCookieProvider</name>
        <enabled>true</enabled>
        <param>
            <name>sso.authentication.provider.url</name>
            <value>https://host:port/gateway/idp/api/v1/websso</value>
        </param>
    </provider>
```

where the values of the parameters are specific to your environment:

- `<name>sso.authentication.provider.url</name></value>https://host:port:/gateway/idp/api/v1/websso</value>`

(Required) Indicates the location of the KnoxSSO endpoint and where to redirect the useragent when no SSO cookie is found in the incoming request.

This will indicate the location of the KnoxSSO endpoint and where to redirect the useragent when no SSO cookie is found in the incoming request.

b. If you want to access NiFi directly rather than through Knox, complete the following steps:

This step is not necessary if you plan to access NiFi through Knox and not use Knox's SSO.

i. Export the Knox SSO certificate using the following command:

```
$KNOX_INSTALL_DIR/bin/knoxcli.sh export-cert
```

ii. Set the following properties in Advanced NiFi section in Ambari:

```
nifi.security.user.knox.url=https://localhost:8443/gateway/knoxsso/api/
v1/websso
nifi.security.user.knox.publicKey=<path-to>/gateway-identity.pem
nifi.security.user.knox.cookieName=hadoop-jwt
nifi.security.user.knox.audiences=
```

These properties assume that Knox is running locally on port 8443 and NiFi is secured and running on another port.

The cookieName property must align with what is configured in Knox. The audiences property is used to only accept tokens from a particular audience. The audiences value is configured as part of Knox SSO [1].

11 Save the configuration and restart Knox.

3.3. Accessing NiFi Via Knox

To launch NiFi via Knox, complete the following steps:

1. Enter the following url in a browser:

   ```
   https://$KNOX_HOST:$KNOX_PORT/$GATEWAY_CONTEXT/default/nifi-app/
   nifi
   ```

   Where:

   • `$KNOX_HOST:KNOX_PORT` is the Knox host:gateway port

   • `$GATEWAY_CONTEXT` is the gateway context, defined in the gateway-site
     configuration in Knox

2. When prompted, enter your NiFi username and password to be authenticated by Knox.

   Knox proxies the NiFi UI.

4. SAM Authentication

After you Kerborize the cluster and try to access the SAM UI for the first time, an Authentication Required message displays. To log into SAM for the first time and to create other users, you must log in with the `streamline` principal and keytab. Ambari creates the `streamline` principal and keytab during the Kerberos process.

Details of the principal and the keyab:

• The ketab is called `streamline.service.keytab` and islocated on the node where SAM is installed. It is located under `/etc/security`.

• The principal name has the following convention: `streamline-<cluster_name_lower_case>`

After you have logged into SAM as the `streamline` user, you can create other users

4.1. Logging into SAM for the First Time

About This Task

To log into SAM for the first time and to create other users, you must log in with the `streamline` principal and keytab.

Steps

1. Use Ambari to download the Kerboros client your local machine:

    a. In Ambari, select **Hosts**, and select a host.

    b. Click **Host Actions**, then **Download Client Configs**, then **Kerberos Client**.

    c. Run the following command:

    ```
    tar -zxvf KERBEROS_CLIENT-configs.tar
    ```

    d. Copy the `krb5.conf` to `/etc`.

2. Copy the Keytab for the streamline user.

    Copy the key tab from `$FQDN_SAM_NODE:/etc/security/keytabs/ streamline.service.keytab` to you local machine.

3. Log in as the streamline user:

    ```
    kinit -kt streamline.service.keytab streamline-
    <your_cluster_name>@<REALM_NAME>
    ```

Result

You are now able to access the SAM UI securely, and can proceed with creating additional users.

More Information

See Managing Users and Assigning them to Roles

4.2. Logging In as a Different User

If you want to log in as one of the users you have created, enter:

```
kinit <user-name>@<REALM_NAME>
```

Example

After the user solson is created, to log in as that user, perform a `kinit` as solson. After successful kinit, when the user solson goes to the SAM UI, they see the following:

5. Installing Ranger

Apache Ranger can be installed either manually or using the Ambari UI. Unlike the manual installation process, which requires you to perform a number of installation steps, installing Ranger using the Ambari UI is simpler and easier. Once Ambari has been installed and configured, you can use the **Add Service** wizard to install the following components:

- Ranger Admin

- Ranger UserSync

- Ranger Key Management Service

5.1. Ranger Installation Prerequisites

Before you install Ranger, make sure your cluster meets the following requirements:

- You have installed Log Search or have an external Solr running.

- A MySQL, Oracle, or PostgreSQL database instance must be running and available to be used by Ranger. Configuration of the database instance for Ranger is described in the following sections for some of the databases supported by Ranger.

  - Configuring MySQL for Ranger [26]

  - Configuring PostgreSQL for Ranger [28]

  - Configuring Oracle for Ranger [29]

- If you choose not to provide system Database Administrator (DBA) account details to the Ambari Ranger installer, you can use the `dba_script.py` Python script to create Ranger DB database users without exposing DBA account information to the Ambari Ranger installer. You can then run the normal Ambari Ranger installation without specifying a DBA user name and password. For more information see Setting up Database Users Without Sharing DBA Credentials.

- The Ranger installation creates two new users (default names: rangeradmin and rangerlogger) and two new databases (default names: ranger and ranger_audit).

5.2. Setting up Databases for Ranger

- Configuring MySQL for Ranger [26]

- Configuring PostgreSQL for Ranger [28]

- Configuring Oracle for Ranger [29]

5.2.1. Configuring MySQL for Ranger

1. The MySQL database administrator should be used to create the Ranger databases.

The following series of commands could be used to create the `rangerdba` user with password `rangerdba`.

a. Log in as the root user, then use the following commands to create the `rangerdba` user and grant it adequate privileges.

```
CREATE USER 'rangerdba'@'localhost' IDENTIFIED BY 'rangerdba';

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'localhost';

CREATE USER 'rangerdba'@'%' IDENTIFIED BY 'rangerdba';

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'%';

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'localhost' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'%' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

b. Use the `exit` command to exit MySQL.

c. You should now be able to reconnect to the database as `rangerdba` using the following command:

```
mysql -u rangerdba -prangerdba
```

After testing the `rangerdba` login, use the `exit` command to exit MySQL.

2. Use the following command to confirm that the `mysql-connector-java.jar` file is in the Java share directory. This command must be run on the server where Ambari server is installed.

```
ls /usr/share/java/mysql-connector-java.jar
```

If the file is not in the Java share directory, use the following command to install the MySQL connector .jar file.

RHEL/CentOS/Oracle Linux

```
yum install mysql-connector-java*
```

SLES

```
zypper install mysql-connector-java*
```

3. Use the following command format to set the `jdbc/driver/path` based on the location of the MySQL JDBC driver .jar file.This command must be run on the server where Ambari server is installed.

```
ambari-server setup --jdbc-db={database-type} --jdbc-driver={/jdbc/driver/
path}
```

For example:

```
ambari-server setup --jdbc-db=mysql --jdbc-driver=/usr/share/java/mysql-
connector-java.jar
```

5.2.2. Configuring PostgreSQL for Ranger

1. On the PostgreSQL host, install the applicable PostgreSQL connector.

   RHEL/CentOS/Oracle Linux

   ```
   yum install postgresql-jdbc*
   ```

   SLES

   ```
   zypper install -y postgresql-jdbc
   ```

2. Confirm that the .jar file is in the Java share directory.

   ```
   ls /usr/share/java/postgresql-jdbc.jar
   ```

3. Change the access mode of the .jar file to 644.

   ```
   chmod 644 /usr/share/java/postgresql-jdbc.jar
   ```

4. The PostgreSQL database administrator should be used to create the Ranger databases.

   The following series of commands could be used to create the `rangerdba` user and grant it adequate privileges.

   ```
   echo "CREATE DATABASE $dbname;" | sudo -u $postgres psql -U $postgres
   echo "CREATE USER $rangerdba WITH PASSWORD '$passwd';" | sudo -u $postgres
    psql -U $postgres
   echo "GRANT ALL PRIVILEGES ON DATABASE $dbname TO $rangerdba;" | sudo -u
    $postgres psql -U $postgres
   ```

   Where:

   - `$postgres` is the Postgres user.

   - `$dbname` is the name of your PostgreSQL database

   - `$passwd` is the password for your Ranger Admin user

   - `$rangerdba` is the username for your Ranger Admin user

5. Use the following command format to set the `jdbc/driver/path` based on the location of the PostgreSQL JDBC driver .jar file. This command must be run on the server where Ambari server is installed.

   ```
   ambari-server setup --jdbc-db={database-type} --jdbc-driver={/jdbc/driver/
   path}
   ```

   For example:

   ```
   ambari-server setup --jdbc-db=postgres --jdbc-driver=/usr/share/java/
   postgresql-jdbc.jar
   ```

6. Run the following command:

   ```
   export HADOOP_CLASSPATH=${HADOOP_CLASSPATH}:${JAVA_JDBC_LIBS}:/connector jar
    path
   ```

7. Add Allow Access details for Ranger users:

- (Optional) Use the following command to find the location of `postgresql.conf` and `pg_hba.conf`:

```
echo "SHOW config_file;" | sudo -u $postgres psql -U $postgres
```

- change `listen_addresses='localhost'` to `listen_addresses='*'` (`'*'` = `any`) to listen from all IPs in `postgresql.conf`.

- Add the Ranger Admin user to the appropriate line in the `pg_hba.conf` file, the following provides an example:

```
# TYPE   DATABASE    USER             CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local    all       postgres,rangeradmin,rangerlogger        trust
# IPv4 local connections:
host     all       postgres,rangeradmin,rangerlogger        0.0.0.0/0             trust
# IPv6 local connections:
host     all       postgres,rangeradmin,rangerlogger        ::/0                  trust
"/var/lib/pgsql/data/pg_hba.conf" 74L, 3445C
```

8. After editing the `pg_hba.conf` file, run the following command to refresh the PostgreSQL database configuration:

```
sudo -u postgres /usr/bin/pg_ctl -D $PGDATA reload
```

For example, if the `pg_hba.conf` file is located in the `/var/lib/pgsql/data` directory, the value of `$PGDATA` is `/var/lib/pgsql/data`.

5.2.3. Configuring Oracle for Ranger

1. On the Oracle host, install the appropriate JDBC .jar file.

- Download the Oracle JDBC (OJDBC) driver from http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html.

- For **Oracle Database 11g**: select Oracle Database 11g Release 2 drivers > ojdbc6.jar.

- For **Oracle Database 12c**: select Oracle Database 12c Release 1 driver > ojdbc7.jar.

- Copy the .jar file to the Java share directory. For example:

```
cp ojdbc7.jar /usr/share/java
```

> **Note**
>
> Make sure the .jar file has the appropriate permissions. For example:
>
> ```
> chmod 644 /usr/share/java/ojdbc7.jar
> ```

2. The Oracle database administrator should be used to create the Ranger databases.

The following series of commands could be used to create the `RANGERDBA` user and grant it permissions using SQL*Plus, the Oracle database administration utility:

```
# sqlplus sys/root as sysdba
CREATE USER $RANGERDBA IDENTIFIED BY $RANGERDBAPASSWORD;
GRANT SELECT_CATALOG_ROLE TO $RANGERDBA;
GRANT CONNECT, RESOURCE TO $RANGERDBA;
QUIT;
```

3. Use the following command format to set the `jdbc/driver/path` based on the location of the Oracle JDBC driver .jar file. This command must be run on the server where Ambari server is installed.

```
ambari-server setup --jdbc-db={database-type} --jdbc-driver={/jdbc/driver/
path}
```

For example:

```
ambari-server setup --jdbc-db=oracle --jdbc-driver=/usr/share/java/ojdbc6.
jar
```

5.3. Installing Ranger

To install Ranger using Ambari:

1. Start the Installation [   ]

2. Customize Services [33]

3. Complete the Ranger Installation [57]

Related Topics

• Setting up Database Users Without Sharing DBA Credentials [61]

• Updating Ranger Admin Passwords [62]

5.3.1. Start the Installation

1. Log into your Ambari cluster with your designated user credentials. The main Ambari Dashboard page will be displayed.

2. From the main Ambari Dashboard page, click **Actions**, then select **Add Service**.
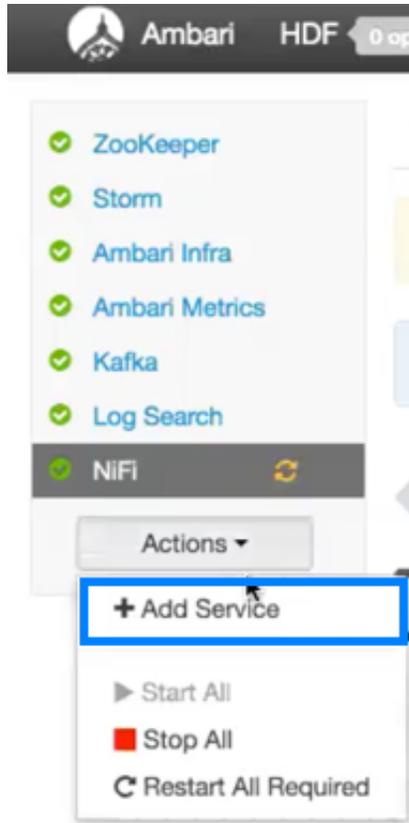
3. On the Choose Services page, select **Ranger**, then click **Next**.

4. The Ranger Requirements page appears. Ensure that you have met all of the installation requirements, then select the "I have met all the requirements above" check box and click **Proceed**.

5. From the **Assign Masters** page, you are then prompted to select the host where you want to install Ranger Admin. This host must have DB admin access to the Ranger DB host and User Sync.

Make a note of the Ranger Admin host for use in subsequent installation steps. Click **Next** when finished to continue with the installation.

Note

The Ranger Admin and Ranger User Sync services must be installed on the same cluster node.

6. From the **Assign Slaves and Clients** page, click **Next**.

7. The Customize Services page appears. These settings are described in the next section.

5.3.2. Customize Services

The next step in the installation process is to specify Ranger settings on the Customize Services page.

• Ranger Admin Settings [33]

• Configure Ranger User Sync [42]

• Specify Plugins to Enable [48]

• Ranger Audit Settings [49]

• Configure Ranger Authentication [50]

5.3.2.1. Ranger Admin Settings

1. On the Customize Services page, select the Ranger Admin tab, then use the **DB Flavor** drop-down to select the database type that you are using with Ranger.
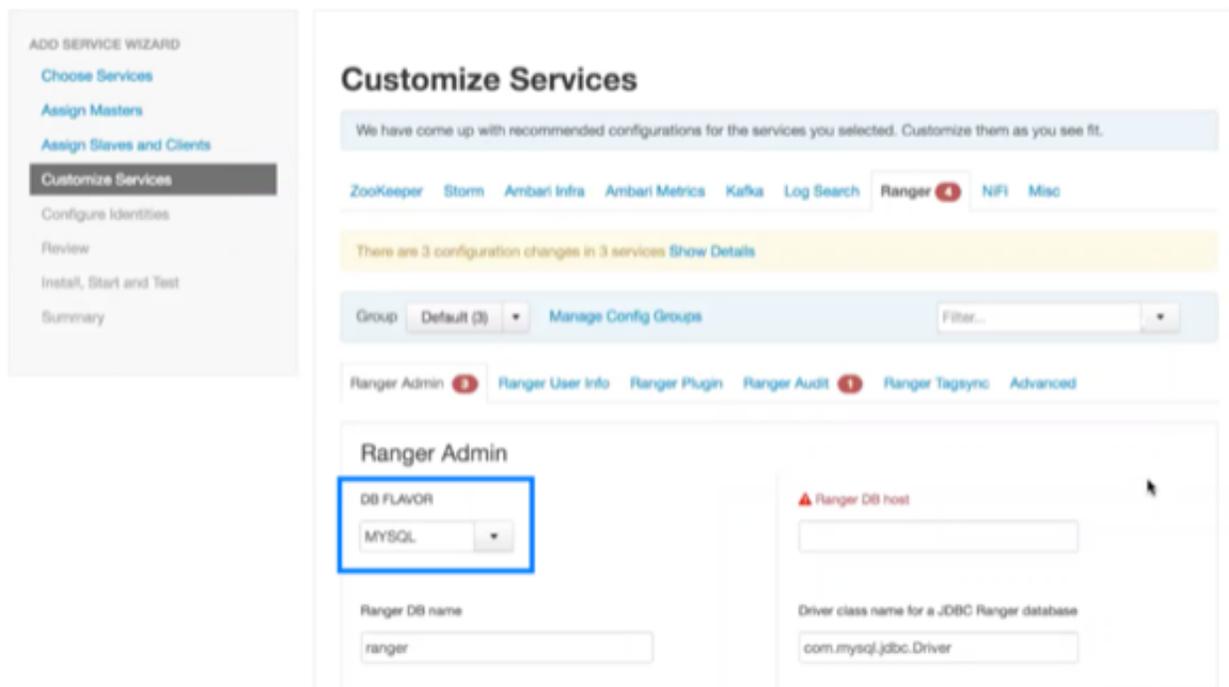
2. Enter the database server address in the **Ranger DB Host** box.

Table 5.1. Ranger DB Host

DB Flavor	Host	Example
MySQL	<HOST[:PORT]>	c6401.ambari.apache.org or c6401.ambari.apache.org:3306
Oracle	<HOST:PORT:SID>	c6401.ambari.apache.org:1521:ORCL
	<HOST:PORT/Service>	c6401.ambari.apache.org:1521/XE
PostgreSQL	<HOST[:PORT]>	c6401.ambari.apache.org or c6401.ambari.apache.org:5432
MS SQL	<HOST[:PORT]>	c6401.ambari.apache.org or c6401.ambari.apache.org:1433
SQLA	<HOST[:PORT]>	c6401.ambari.apache.org or c6401.ambari.apache.org:2638

3. **Ranger DB name** – The name of the Ranger Policy database, i.e. ranger_db. Please not that if you are using Oracle, you must specify the Oracle tablespace name here.

4. Driver class name for a JDBC Ranger database – the driver class name is automatically generated based on the selected DB Flavor. The table below lists the default driver class settings. Currently Ranger does not support any third party JDBC driver.

Table 5.2. Driver Class Name

DB Flavor	Driver class name for a JDBC Ranger database
MySQL	com.mysql.jdbc.Driver
Oracle	oracle.jdbc.driver.OracleDriver
PostgreSQL	org.postgresql.Driver
MS SQL	com.microsoft.sqlserver.jdbc.SQLServerDriver
SQLA	sap.jdbc4.sqlanywhere.IDriver

5. **Ranger DB username** and **Ranger DB Password** – Enter the user name and passwords for your Ranger database server. The following table describes these settings in more detail. You can use the MySQL database that was installed with Ambari, or an external MySQL, Oracle, PostgreSQL, MS SQL or SQL Anywhere database.

Table 5.3. Ranger DB Username Settings

Property	Description	Default Value	Example Value	Required?
Ranger DB username	The username for the Policy database.	rangeradmin	rangeradmin	Yes
Ranger DB password	The password for the Ranger Policy database user.		PassWORd	Yes

6. **JDBC connect string**

> ⚠️ **Important**
>
> Currently the Ambari installer generates the JDBC connect string using the `jdbc:oracle:thin:@//host:port/db_name` format. You must replace the connection string as described in the following table:

Table 5.4. JDBC Connect String

DB Flavor	Syntax	Example Value
MySQL	jdbc:mysql://DB_HOST:PORT/db_name	jdbc:mysql://c6401.ambari.apache.org:3306/ranger_db
Oracle	**For Oracle SID:** jdbc:oracle:thin:@DB_HOST:PORT:SID	jdbc:oracle:thin:@c6401.ambari.apache.org:1521:**ORCL**
	For Oracle Service Name: jdbc:oracle:thin:@//DB_HOST[:PORT][/ServiceName]	jdbc:oracle:thin:@//c6401.ambari.apache.org:1521/**XE**
PostgreSQL	jdbc:postgresql://DB_HOST/db_name	jdbc:postgresql://c6401.ambari.apache.org:5432/ranger_db
MS SQL	jdbc:sqlserver://DB_HOST;databaseName=db_name	jdbc:sqlserver://c6401.ambari.apache.org:1433;databaseName=ranger_db
SQLA	jdbc:sqlanywhere:host=DB_HOST;databaseName=db_name	jdbc:sqlanywhere:host=c6401.ambari.apache.org:2638;data

7. **Setup Database and Database User**

- If set to **Yes** – The Database Administrator (DBA) user name and password will need to be provided as described in the next step.

  > **Note**
  >
  > Ranger does not store the DBA user name and password after setup. Therefore, you can clear these values in the Ambari UI after the Ranger setup is complete.

- If set to **No** – A **No** indicates that you do not wish to provide Database Administrator (DBA) account details to the Ambari Ranger installer. Setting this to No continues the Ranger installation process without providing DBA account details. In this case, you must perform the system database user setup as described in Setting up Database Users Without Sharing DBA Credentials, and then proceed with the installation.

  > **Note**
  >
  > If **No** is selected and the UI still requires you to enter a user name and password in order to proceed, you can enter any value – the values do not need to be the actual DBA user name and password.

8. **Database Administrator (DBA) username** and **Database Administrator (DBA) password** – The DBA username and password are set when the database server is installed. If you do not have this information, contact the database administrator who installed the database server.

Table 5.5. DBA Credential Settings

Property	Description	Default Value	Example Value	Required?
Database Administrator (DBA) username	The Ranger database user that has administrative privileges to create database schemas and users.	root	root	Yes
Database Administrator (DBA) password	The root password for the Ranger database user.		root	Yes

If the Oracle DB root user Role is SYSDBA, you must also specify that in the **Database Administrator (DBA) username** parameter. For example, if the DBA user name is `orcl_root` you must specify `orcl_root AS SYSDBA`.

> **Note**
>
> As mentioned in the note in the previous step, if **Setup Database and Database User** is set to **No**, a placeholder DBA username and password may still be required in order to continue with the Ranger installation.

The following images show examples of the DB settings for each Ranger database type.

> **Note**
>
> To test the DB settings, click **Test Connection**. If a Ranger database has not been pre-installed, **Test Connection** will fail even for a valid configuration.

MySQL

Oracle – if the Oracle instance is running with a Service name.

Oracle – if the Oracle instance is running with a SID.

PostgreSQL

MS SQL

SQL Anywhere

5.3.2.2. Configure Ranger User Sync

This section describes how to configure Ranger User Sync for either UNIX or LDAP/AD.

- Configuring Ranger User Sync for UNIX [42]

- Configuring Ranger User Sync for LDAP/AD [43]

5.3.2.2.1. Configuring Ranger User Sync for UNIX

Use the following steps to configure Ranger User Sync for UNIX.

1. On the Customize Services page, select the **Ranger User Info** tab.

2. Click **Yes** under Enable User Sync.

3. Use the **Sync Source** drop-down to select UNIX, then set the following properties.

Table 5.6. UNIX User Sync Properties

| Property	Description	Default Value
Sync Source	Only sync users above this user ID.	500
Password File	The location of the password file on the Linux server.	/etc/passwd
Group File	The location of the groups file on the Linux server.	/etc/group

5.3.2.2.2. Configuring Ranger User Sync for LDAP/AD

Use the following steps to configure Ranger User Sync for LDAP/AD.

1. On the Customize Services page, select the Ranger User Info tab.

2. Click **Yes** under Enable User Sync.

3. Use the Sync Source drop-down to select LDAP/AD.

4. Set the following properties on the Common Configs tab.

Table 5.7. LDAP/AD Common Configs

Property	Description	Default Value	Sample Values
LDAP/AD URL	Add URL depending upon LDAP/AD sync source	ldap://{host}:{port}	ldap:// ldap.example.com:389 or ldaps:// ldap.example.com:636
Bind Anonymous	If Yes is selected, the Bind User and Bind User Password are not required.	NO	
Bind User	The location of the groups file on the Linux server.	The full distinguished name (DN), including common name (CN), of an LDAP/AD user account that has privileges to search for users. The LDAP bind DN is used to connect to LDAP and query for users and groups.	cn=admin,dc=example,dc=com or admin@example.com
Bind User Password	The password of the Bind User.		
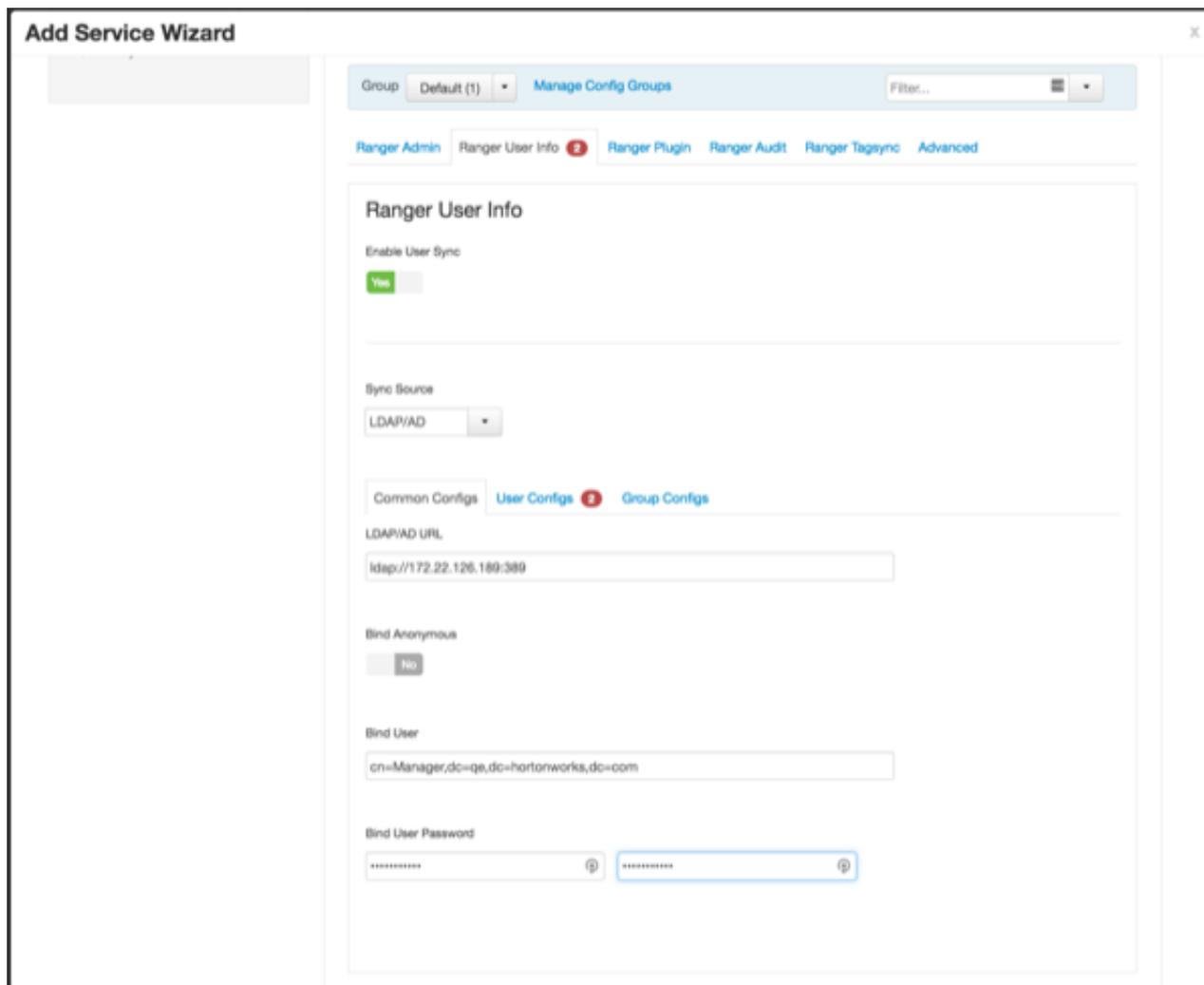
5. Set the following properties on the User Configs tab.

Table 5.8. LDAP/AD User Configs

Property	Description	Default Value	Sample Values
Group User Map Sync	Sync specific groups for users.	Yes	Yes
Username Attribute	The LDAP user name attribute.		sAMAccountName for AD, uid or cn for OpenLDAP
User Object Class	Object class to identify user entries.	person	top, person, organizationalPerson, user, or posixAccount
User Search Base	Search base for users. Ranger can search multiple OUs in AD. Ranger UserSync module performs a user search on each configured OU and adds all the users into single list. Once all the OUs are processed, a user's group membership is computed based on the group search.		cn=users,dc=example,dc=com;ou=example1,ou=e
User Search Filter	Optional additional filter constraining the users selected for syncing.		Sample filter to retrieve all the users: cn=* Sample filter to retrieve all the users who are members of groupA or groupB: (| (memberof=CN=GroupA,OU=groups,DC=example (memberof=CN=GroupB,OU=groups,DC=example
User Search Scope	This value is used to limit user search to the depth from search base.	sub	base, one, or sub
User Group Name Attribute	Attribute from user entry whose values would be treated as group values to be pushed into the Access Manager database. You can provide multiple attribute names separated by commas.	memberof,ismemberof	memberof, ismemberof, or gidNumber
Enable User Search	This option is available only when the "Enable Group Search First" option is selected.	No	Yes

Add Service Wizard

Ranger User Info

Enable User Sync

`Yes`

Sync Source

LDAP/AD   ▾

Common Configs | User Configs | Group Configs

Username Attribute

uid

User Object Class

person

User Search Base

dc=qe;dc=hortonworks;dc=com

User Search Filter

User Search Scope

sub

User Group Name Attribute

memberof, ismemberof

Group User Map Sync

`Yes`

Enable User Search

`Yes`

6. Set the following properties on the Group Configs tab.

Table 5.9. LDAP/AD Group Configs

Property	Description	Default Value	Sample Values
Enable Group Sync	If Enable Group Sync is set to No, the group names the users belong to are derived from "User Group Name Attribute". In this case no additional group filters are applied.	No	Yes

Property	Description	Default Value	Sample Values
	If Enable Group Sync is set to Yes, the groups the users belong to are retrieved from LDAP/AD using the following group-related attributes.		
Group Member Attribute	The LDAP group member attribute name.		member
Group Name Attribute	The LDAP group name attribute.		distinguishedName for AD, cn for OpenLdap
Group Object Class	LDAP Group object class.		group, groupofnames, or posixGroup
Group Search Base	Search base for groups. Ranger can search multiple OUs in AD. Ranger UserSync module performs a user search on each configured OU and adds all the users into single list. Once all the OUs are processed, a user's group membership is computed based on the group search configuration. Each OU segment needs to be separated by a ; (semi-colon).		ou=groups,DC=example,DC=com;ou=group1;ou=
Group Search Filter	Optional additional filter constraining the groups selected for syncing.		Sample filter to retrieve all groups: cn=* Sample filter to retrieve only the groups whose cn is Engineering or Sales: (|(cn=Engineering)(cn=Sales))
Enable Group Search First	When **Enable Group Search First** is selected, there are two possible ways of retrieving users: • If **Enable User Search** is not selected: users are retrieved from the "member" attribute of the group. • If **Enable User Search** is selected: user membership is computed by performing an LDAP search based on the user configuration.	No	Yes

5.3.2.3. Specify Plugins to Enable

From the **Ranger Plugin** tab, use the **ON**/**OFF** slider to indicate which plugins you want to enable. You can also enable plugins at a later time.

If you select the Storm or Kafka plugins here, they are not enabled until you also enable Kerberos.

5.3.2.4. Ranger Audit Settings

Apache Ranger uses Apache Solr to store audit logs and provides UI searching through the audit logs. Solr must be installed and configured before installing Ranger Admin or any of the Ranger component plugins. The default configuration for Ranger Audits to Solr uses the shared Solr instance provided under the Ambari Infra service. Solr is both memory and CPU intensive. If your production system has high volume of access requests, make sure that the Solr host has adequate memory, CPU, and disk space.

SolrCloud is the preferred setup for production usage of Ranger. SolrCloud, which is deployed with the Ambari Infra service, is a scalable architecture that can run as a single node or multi-node cluster. It has additional features such as replication and sharding, which is useful for high availability (HA) and scalability. You should plan your deployment based on your cluster size. Because audit records can grow dramatically, plan to have at least 1 TB of free space in the volume on which Solr will store the index data. Solr works well with a minimum of 32 GB of RAM. You should provide as much memory as possible to the Solr process. It is highly recommended to use SolrCloud with at least two Solr nodes running on different servers with replication enabled. SolrCloud also requires Apache ZooKeeper.

1. On the Customize Services page, select the **Ranger Audit** tab.

2. Under Audit to Solr, click **OFF** under SolrCloud to enable SolrCloud. The button label will change to ON, and the SolrCloud configuration settings will be loaded automatically.

5.3.2.5. Configure Ranger Authentication

This section describes how to configure Ranger authentication for UNIX, LDAP, and AD.

- Configuring Ranger UNIX Authentication [50]

- Configuring Ranger LDAP Authentication [51]

- Configuring Ranger Active Directory Authentication [54]

5.3.2.5.1. Configuring Ranger UNIX Authentication

Use the following steps to configure Ranger authentication for UNIX.

1. Select the Advanced tab on the Customize Services page.

2. Under Ranger Settings, specify the Ranger Access Manager/Service Manager host address in the **External URL** box in the format `http://<your_ranger_host>:6080`.

3. Under Ranger Settings, select **UNIX**.

   HTTP is enabled by default – if you disable HTTP, only HTTPS is allowed.

4. Under UNIX Authentication Settings, set the following properties.

Table 5.10. UNIX Authentication Settings

Property	Description	Default Value	Example Value
Allow remote Login	Flag to enable/ disable remote login. Only applies to UNIX authentication.	true	true
ranger.unixauth.service.hostname	The address of the host where the UNIX authentication service is running.	{{ugsync_host}}	{{ugsync_host}}
ranger.unixauth.service.port	The port number on which the UNIX authentication service is running.	5151	5151

Note

Properties with value {{xyz}} are macro variables that are derived from other specified values in order to streamline the configuration process. Macro variables can be edited if required – if you need to restore the original value, click the Set Recommended symbol at the right of the property box.

5.3.2.5.2. Configuring Ranger LDAP Authentication

Use the following steps to configure Ranger authentication for LDAP.

1. Select the Advanced tab on the Customize Services page.

2. Under Ranger Settings, specify the Ranger Access Manager/Service Manager host address in the **External URL** box in the format `http://<your_ranger_host>:6080`.

3. Under Ranger Settings, select **LDAP**.

4. Under LDAP Settings, set the following properties.

Table 5.11. LDAP Authentication Settings

Property	Description	Default Value	Example Value
ranger.ldap.base.dn	The Distinguished Name (DN) of the starting point for directory server searches.	dc=example,dc=com	dc=example,dc=com
Bind User	The full Distinguished Name (DN), including Common Name (CN) of an LDAP user account that has privileges to search for users. This is a macro variable value that is derived from the **Bind User** value from **Ranger User Info > Common Configs**.	{{ranger_ug_ldap_bind_dn}}	{{ranger_ug_ldap_bind_dn}}
Bind User Password	Password for the Bind User. This is a macro variable value that is derived from the **Bind User Password** value from **Ranger User Info > Common Configs**.		
ranger.ldap.group.roleattribute	The LDAP group role attribute.	cn	cn
ranger.ldap.referral	See description below.	ignore	follow | ignore | throw
LDAP URL	The LDAP server URL. This is a macro variable value that is derived from the **LDAP/AD URL** value from **Ranger**	{{ranger_ug_ldap_url}}	{{ranger_ug_ldap_url}}

Property	Description	Default Value	Example Value
	User Info > Common Configs.		
ranger.ldap.user. dnpattern	The user DN pattern is expanded when a user is being logged in. For example, if the user "ldapadmin" attempted to log in, the LDAP Server would attempt to bind against the DN "uid=ldapadmin,ou=users,dc=example,dc=com" using the password the user provided>	uid={0},ou=users, dc=xasecure,dc=net	cn=ldapadmin,ou=Users, dc=example,dc=com
User Search Filter	The search filter used for Bind Authentication. This is a macro variable value that is derived from the **User Search Filter** value from **Ranger User Info > User Configs**.	{{ranger_ug_ldap_user_searchfilter}}	{{ranger_ug_ldap_user_searchfilter}}

> **Note**
>
> Properties with value `{{xyz}}` are macro variables that are derived from other specified values in order to streamline the configuration process. Macro variables can be edited if required – if you need to restore the original value, click the Set Recommended symbol at the right of the property box.

There are three possible values for `ranger.ldap.referral`: `follow`, `throw`, and `ignore`. The recommended setting is `follow`.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to `follow`, the LDAP service provider processes all of the normal entries first, and then follows the continuation references.

- When this property is set to `throw`, all of the normal entries are returned in the enumeration first, before the `ReferralException` is thrown. By contrast, a

"referral" error response is processed immediately when this property is set to `follow` or `throw`.

- When this property is set to `ignore`, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search.

5.3.2.5.3. Configuring Ranger Active Directory Authentication

Use the following steps to configure Ranger authentication for Active Directory.

1. Select the Advanced tab on the Customize Services page.

2. Under Ranger Settings, specify the Ranger Access Manager/Service Manager host address in the **External URL** box in the format `http://<your_ranger_host>:6080`.

3. Under Ranger Settings, select **ACTIVE_DIRECTORY**.

4. Under AD Settings, set the following properties.

Table 5.12. AD Settings

Property	Description	Default Value	Example Value
ranger.ldap.ad.base.dn	The Distinguished Name (DN) of the starting point for directory server searches.	dc=example,dc=com	dc=example,dc=com
ranger.ldap.ad.bind.dn	The full Distinguished Name (DN), including Common Name (CN) of an LDAP user account that has privileges to search for users. This is a macro variable value that is derived from the **Bind User** value from **Ranger User Info > Common Configs**.	{{ranger_ug_ldap_bind_dn}}	{{ranger_ug_ldap_bind_dn}}
ranger.ldap.ad.bind.password	Password for the bind.dn. This is a macro variable value that is derived from the **Bind User Password** value from **Ranger User Info > Common Configs**.		
Domain Name (Only for AD)	The domain name of the AD Authentication service.		dc=example,dc=com
ranger.ldap.ad.referral	See description below.	ignore	follow | ignore | throw
ranger.ldap.ad.url	The AD server URL. This is a macro variable value that is derived from the **LDAP/AD URL** value from **Ranger User Info > Common Configs**.	{{ranger_ug_ldap_url}}	{{ranger_ug_ldap_url}}
ranger.ldap.ad.user.searchfilter	The search filter used for Bind Authentication. This is a macro variable value that is derived from the **User Search Filter** value from **Ranger User Info > User Configs**.	{{ranger_ug_ldap_user_searchfilter}}	{{ranger_ug_ldap_user_searchfilter}}

Note

Properties with value `{{xyz}}` are macro variables that are derived from other specified values in order to streamline the configuration process. Macro variables can be edited if required – if you need to restore the original value, click the Set Recommended symbol at the right of the property box.

There are three possible values for `ranger.ldap.ad.referral`: `follow`, `throw`, and `ignore`. The recommended setting is `follow`.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to `follow`, the AD service provider processes all of the normal entries first, and then follows the continuation references.

- When this property is set to `throw`, all of the normal entries are returned in the enumeration first, before the `ReferralException` is thrown. By contrast, a "referral" error response is processed immediately when this property is set to `follow` or `throw`.

- When this property is set to `ignore`, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search. In the case of AD, a `PartialResultException` is returned when referrals are encountered while search results are processed.
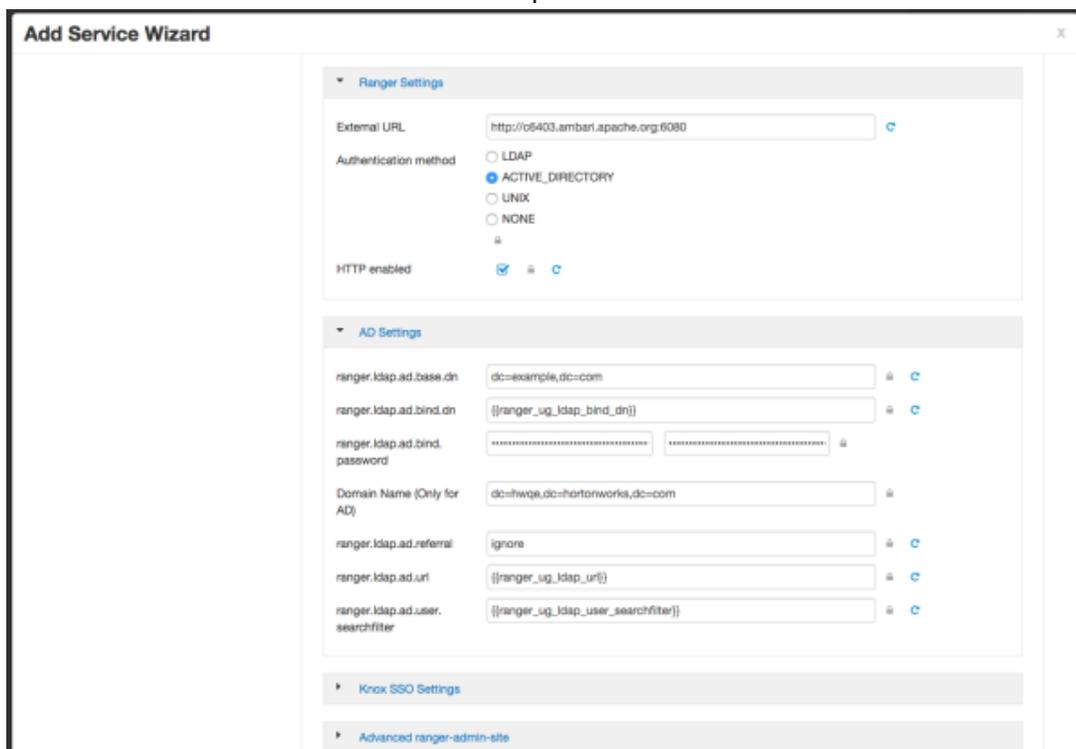
When you have finished configuring all of the Customize Services Settings, click **Next** at the bottom of the page to continue with the installation.

5. When you save the authentication method as Active Directory, a Dependent Configurations pop-up may appear recommending that you set the authentication method as LDAP. This recommended configuration should not be applied for AD, so you should clear (un-check) the **ranger.authentication.method** check box, then click **OK**.
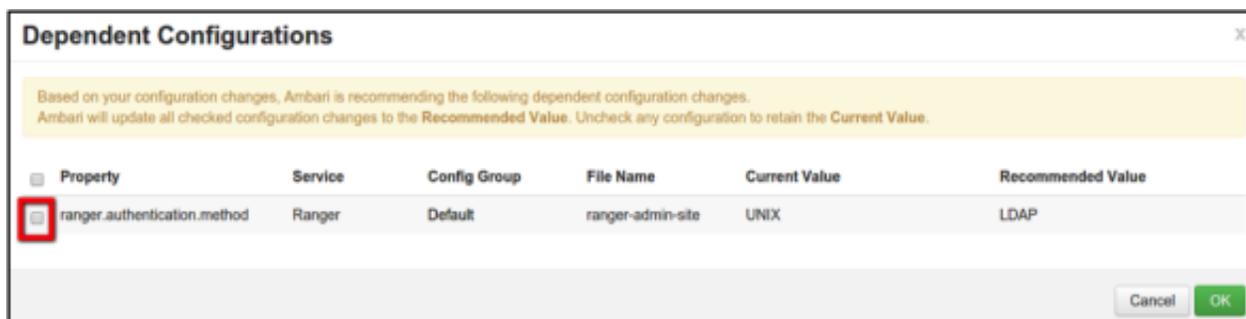
5.3.3. Complete the Ranger Installation

1. On the Review page, carefully review all of your settings and configurations. If everything looks good, click **Deploy** to install Ranger on the Ambari server.

2. When you click **Deploy,** Ranger is installed on the specified host on your Ambari server. A progress bar displays the installation progress.

3. When the installation is complete, a Summary page displays the installation details. You may need to restart services for cluster components after installing Ranger.

> **Note**
>
> If the installation fails, you should complete the installation process, then reconfigure and reinstall Ranger.

5.3.4. Advanced Usersync Settings

To access Usersync settings, select the **Advanced** tab on the Customize Service page. Usersync pulls in users from UNIX, LDAP, or AD and populates Ranger's local user tables with these users.

5.3.4.1. UNIX Usersync Settings

If you are using UNIX authentication, the default values for the Advanced ranger-ugsync-site properties are the settings for UNIX authentication.

5.3.4.2. Required LDAP and AD Usersync Settings

If you are using LDAP authentication, you must update the following Advanced ranger-ugsync-site properties.

Table 5.13. LDAP Advanced ranger-ugsync-site Settings

Property Name	LDAP Value
ranger.usersync.ldap.bindkeystore	Set this to the same value as the `ranger.usersync.credstore.filename` property, i.e, the default value is `/usr/hdf/current/ranger-usersync/conf/ugsync.jceks`
ranger.usersync.ldap.bindalias	ranger.usersync.ldap.bindalias

Property Name	LDAP Value
ranger.usersync.source.impl.class	ldap

Table 5.14. AD Advanced ranger-ugsync-site Settings

Property Name	LDAP Value
ranger.usersync.source.impl.class	ldap

5.3.4.3. Additional LDAP and AD Usersync Settings

If you are using LDAP or Active Directory authentication, you may need to update the following properties, depending upon your specific deployment characteristics.

Table 5.15. Advanced ranger-ugsync-site Settings for LDAP and AD

Property Name	LDAP ranger-ugsync-site Value	AD ranger-ugsync-site Value
ranger.usersync.ldap.url	ldap://127.0.0.1:389	ldap://ad-conrowoller-hostname:389
ranger.usersync.ldap.binddn	cn=ldapadmin,ou=users, dc=example,dc=com	cn=adadmin,cn=Users, dc=example,dc=com
ranger.usersync.ldap.ldapbindpassword	secret	secret
ranger.usersync.ldap.searchBase	dc=example,dc=com	dc=example,dc=com
ranger.usersync.source.impl.class	org.apache.ranger. ladpusersync. process.LdapUserGroupBuilder	
ranger.usersync.ldap.user.searchbase	ou=users, dc=example, dc=com	dc=example,dc=com
ranger.usersync.ldap.user.searchscope	sub	sub
ranger.usersync.ldap.user.objectclass	person	person
ranger.usersync.ldap.user.searchfilter	Set to single empty space if no value. Do not leave it as "empty"	(objectcategory=person)
ranger.usersync.ldap.user.nameattribute	uid or cn	sAMAccountName
ranger.usersync.ldap.user.groupnameattribute	memberof,ismemberof	memberof,ismemberof
ranger.usersync.ldap.username.caseconversion	none	none
ranger.usersync.ldap.groupname.caseconversion	none	none
ranger.usersync.group.searchenabled *	false	false
ranger.usersync.group.usermapsyncenabled *	false	false
ranger.usersync.group.searchbase *	ou=groups, dc=example, dc=com	dc=example,dc=com
ranger.usersync.group.searchscope *	sub	sub
ranger.usersync.group.objectclass *	groupofnames	groupofnames

Property Name	LDAP ranger-ugsync-site Value	AD ranger-ugsync-site Value
ranger.usersync.group.searchfilter *	needed for AD authentication	(member=CN={0}, OU=MyUsers, DC=AD-HDF, DC=COM)
ranger.usersync.group.nameattribute *	cn	cn
ranger.usersync.group.memberattributename *	member	member
ranger.usersync.pagedresultsenabled *	true	true
ranger.usersync.pagedresultssize *	500	500
ranger.usersync.user.searchenabled *	false	false
ranger.usersync.group.search.first.enabled *	false	false

* Only applies when you want to filter out groups.

After you have finished specifying all of the settings on the Customize Services page, click **Next** at the bottom of the page to continue with the installation.

5.3.5. Configuring Ranger for LDAP SSL

You can use the following steps to configure LDAP SSL using self-signed certs in the default Ranger User Sync TrustStore.

1. The default location is `/usr/hdf/current/ranger-usersync/conf/mytruststore.jks` for the `ranger.usersync.truststore.file` property.

2. Alternatively, copy and edit the self-signed ca certs.

3. Set the `ranger.usersync.truststore.file` property to that new cacert file.

```
cd /usr/hdp/<version>/ranger-usersync
service ranger-usersync stop
service ranger-usersync start
```

Where `cert.pem` has the LDAPS cert.

5.3.6. Setting up Database Users Without Sharing DBA Credentials

If do not wish to provide system Database Administrator (DBA) account details to the Ambari Ranger installer, you can use the `dba_script.py` Python script to create Ranger DB database users without exposing DBA account information to the Ambari Ranger installer. You can then run the normal Ambari Ranger installation without specify a DBA user name and password.

To create Ranger DB users using the `dba_script.py` script:

1. Download the Ranger rpm using the yum install command.

```
yum install ranger-admin
```

2. You should see one file named `dba_script.py` in the `/usr/hdf/current/ranger-admin` directory.

3. Get the script reviewed internally and verify that your DBA is authorized to run the script.

4. Execute the script by running the following command:

```
python dba_script.py
```

5. Pass all values required in the argument. These should include `db flavor`, `JDBC jar`, `db host`, `db name`, `db user`, and other parameters.

   - If you would prefer not to pass runtime arguments via the command prompt, you can update the `/usr/hdf/current/ranger-admin/install.properties` file and then run:

   - ```
python dba_script.py -q
```

     When you specify the `-q` option, the script will read all required information from the `install.properties` file

   - You can use the `-d` option to run the script in "dry" mode. Running the script in dry mode causes the script to generate a database script.

     ```
python dba_script.py -d /tmp/generated-script.sql
```

     Anyone can run the script, but it is recommended that the system DBA run the script in dry mode. In either case, the system DBA should review the generated script, but should only make minor adjustments to the script, for example, change the location of a particular database file. No major changes should be made that substantially alter the script – otherwise the Ranger install may fail.

     The system DBA must then run the generated script.

6. Run the Ranger Ambari install procedure, but set **Setup Database and Database User** to **No** in the Ranger Admin section of the Customize Services page.

5.3.7. Updating Ranger Admin Passwords

For the following users, if you update the passwords on the Ranger Configs page, you must also update the passwords on the Configs page of each Ambari component that has the Ranger plugin enabled. Individual Ambari component configurations are not automatically updated – the service restart will fail if you do not update these passwords on each component.

- Ranger Admin user – The credentials for this user are set in **Configs > Advanced ranger-env** in the fields labeled **admin_username** (default value: `admin`) and **admin_password** (default value: `admin`).

- Admin user used by Ambari to create repo/policies – The user name for this user is set in **Configs > Admin Settings** in the field labeled **Ranger Admin username for Ambari** (default value: `amb_ranger_admin`). The password for this user is set in the field labeled **Ranger Admin user's password for Ambari**. This password is specified during the Ranger installation.

The following image shows the location of these settings on the Ranger Configs page:
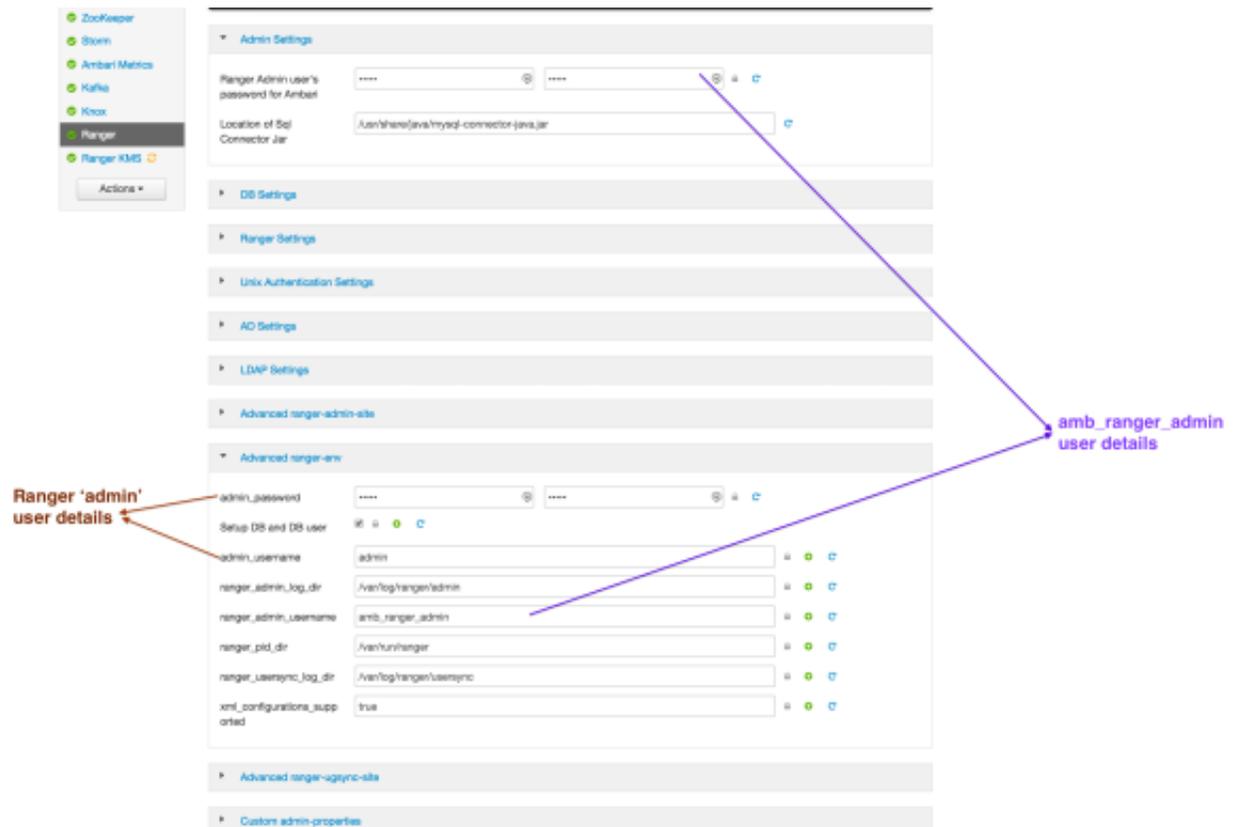
5.3.8. Enabling Ranger Plugins

If you did not enable Ranger plugins during the initial Ranger installation, you can enable them later. This section describes how to enable each of these plugins. For performance reasons, it is recommended that you store audits in Solr, and not in a database.

If you are using a Kerberos-enabled cluster, there are a number of additional steps you must follow to ensure that you can use the Ranger plugins on a Kerberos cluster.

The following Ranger plugins are available:

• Kafka [63]

• Storm [67]

• NiFi [70]

5.3.8.1. Kafka

Use the following steps to enable the Ranger Kafka plugin.

1. On the Ranger Configs page, select the **Ranger Plugin** tab.
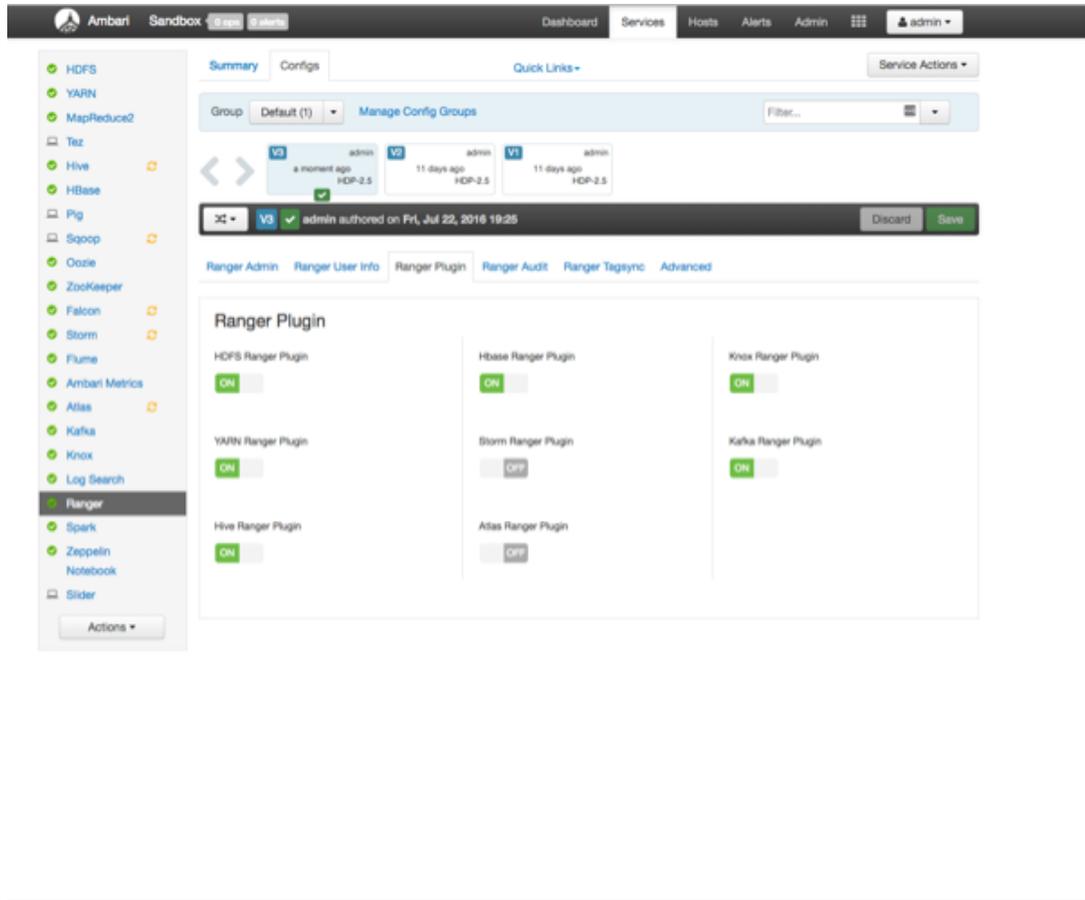
2. Under Kafka Ranger Plugin, select **On**, then click **Save** in the black menu bar.
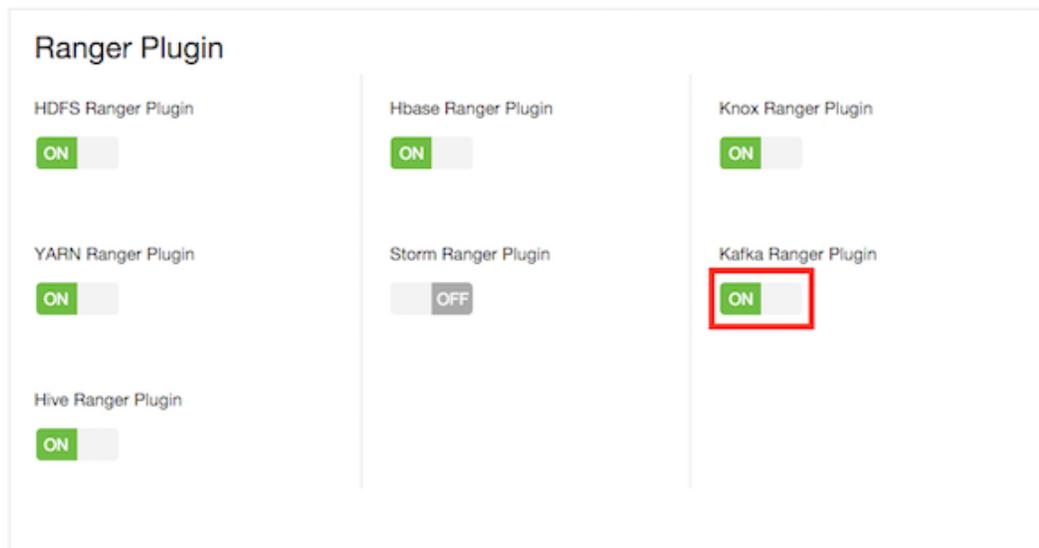
3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.
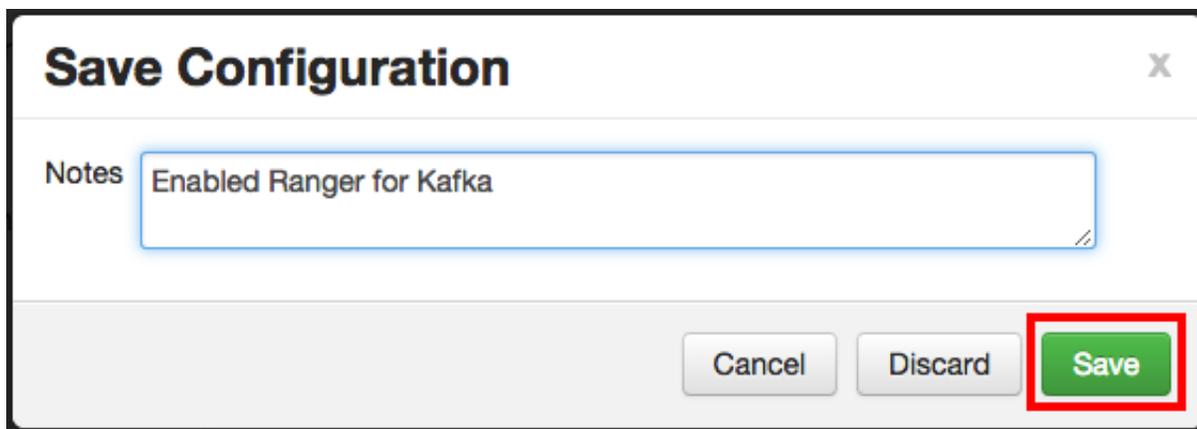
4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.
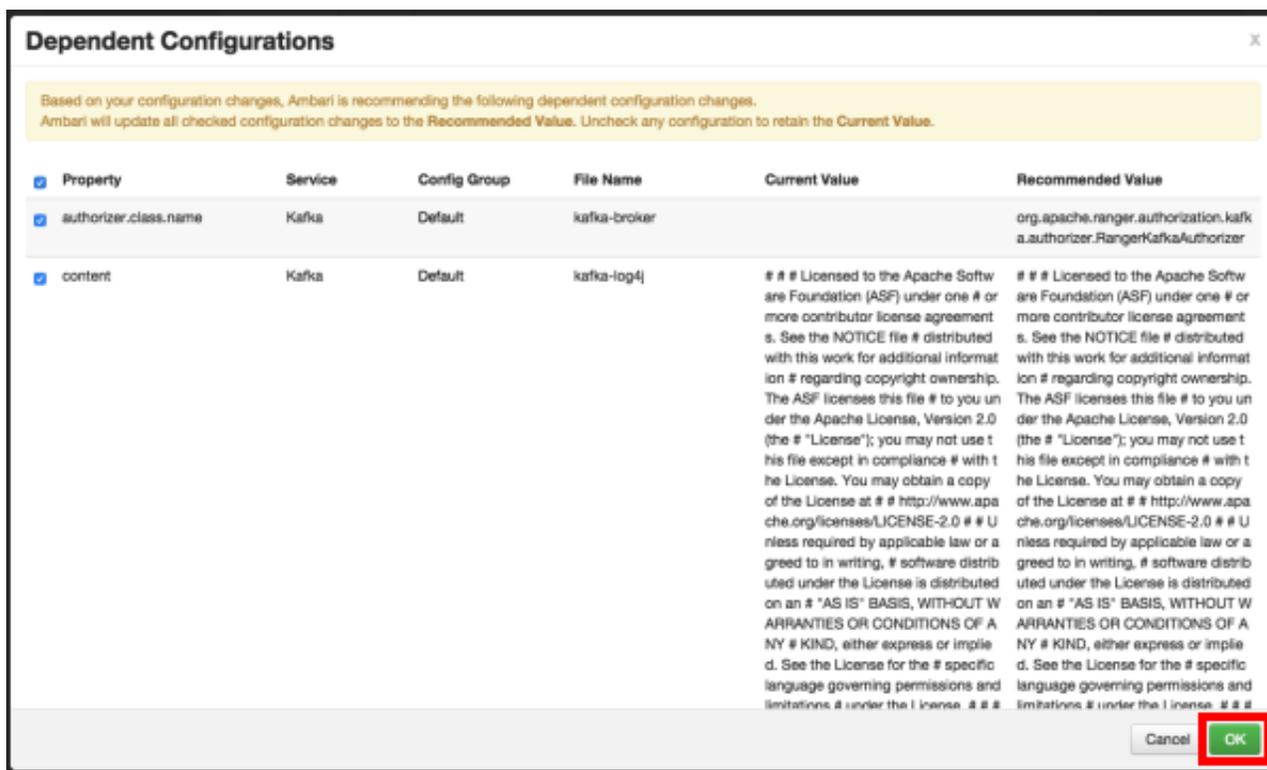
5. Click **OK** on the Save Configuration Changes pop-up.
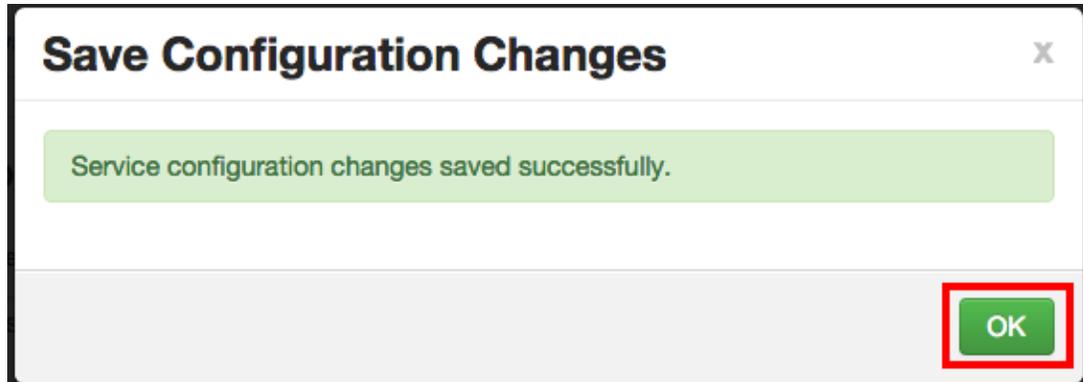
6. Select **Kafka** in the navigation menu, then select **Restart > Restart All Affected** to restart the Kafka service and load the new configuration.

7. Click **Confirm Restart All** on the confirmation pop-up to confirm the Kafka restart.
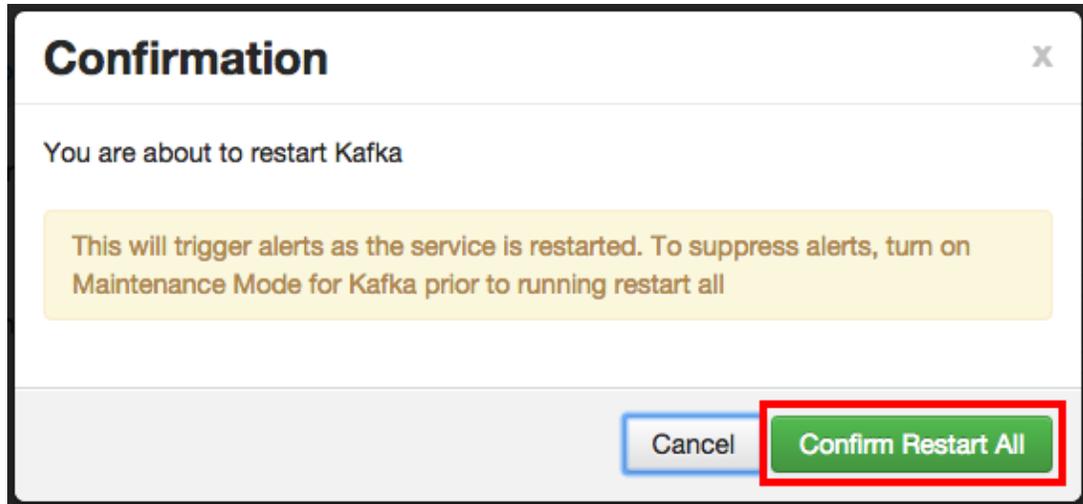
8. After Kafka restarts, the Ranger plugin for Kafka will be enabled.

5.3.8.2. Storm

Use the following steps to enable the Ranger Storm plugin.

1. On the Ranger Configs page, select the **Ranger Plugin** tab.
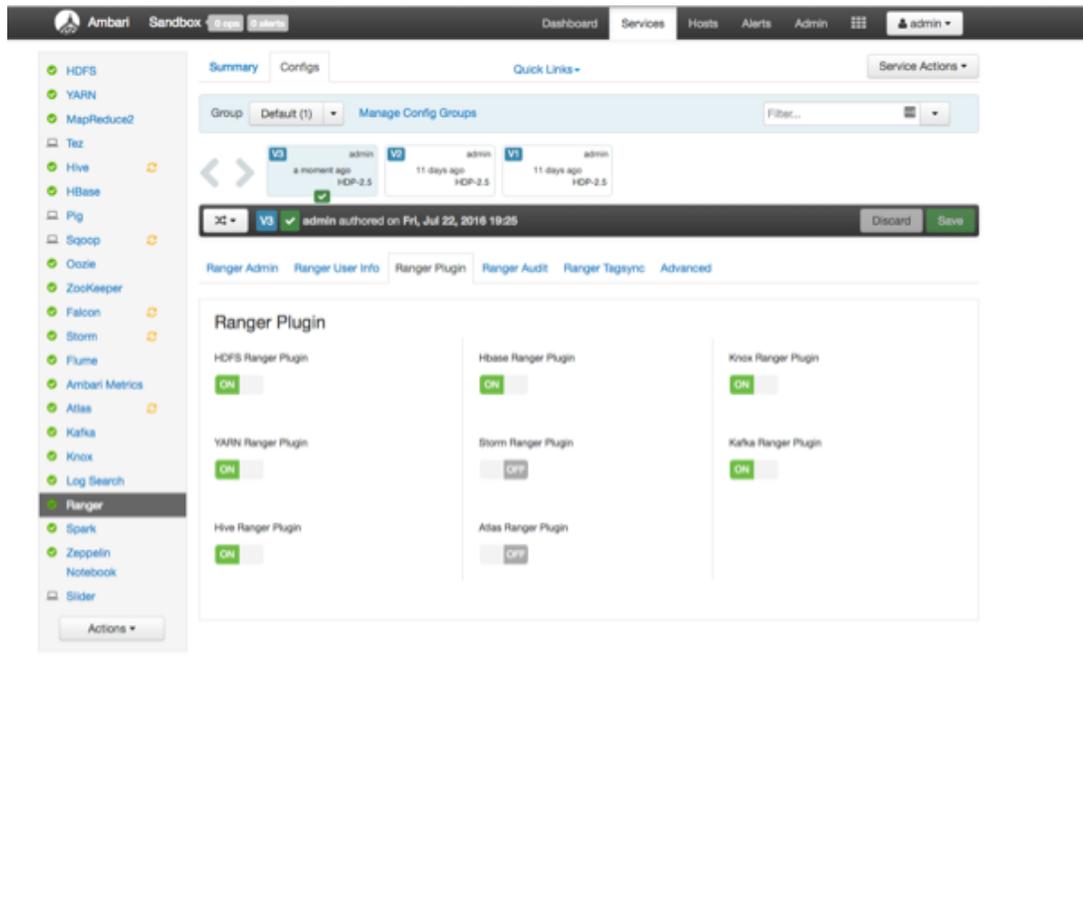
2. Under Storm Ranger Plugin, select **On**, then click **Save** in the black menu bar.
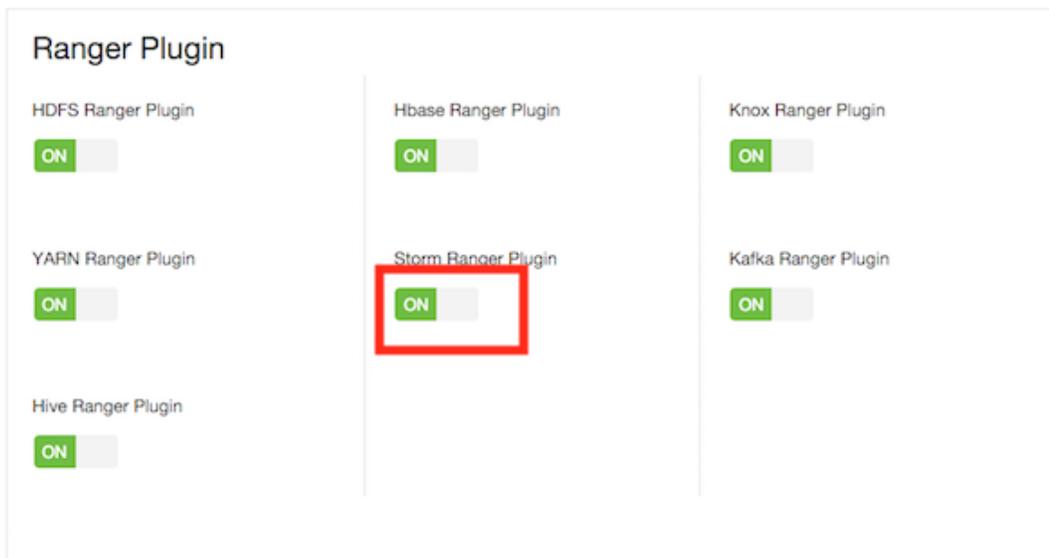
3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.
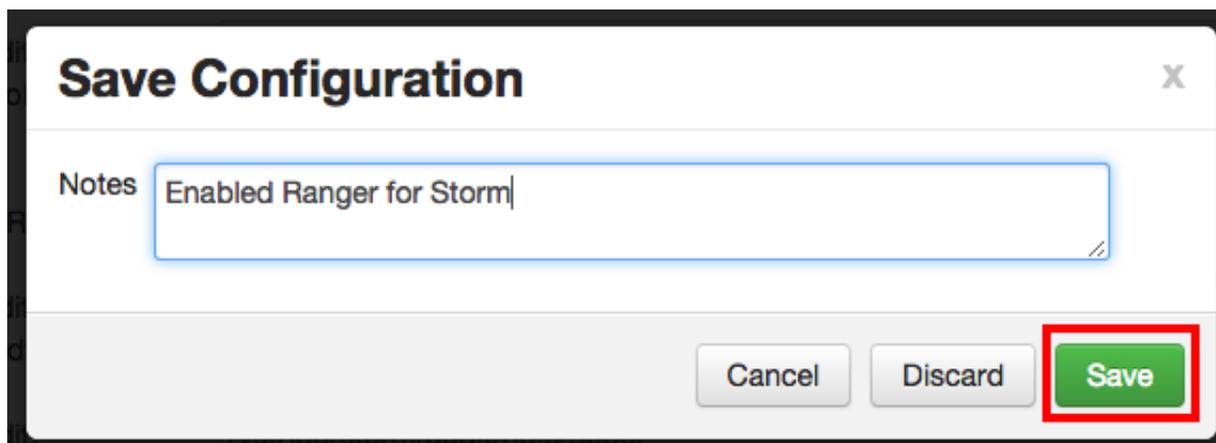
4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.
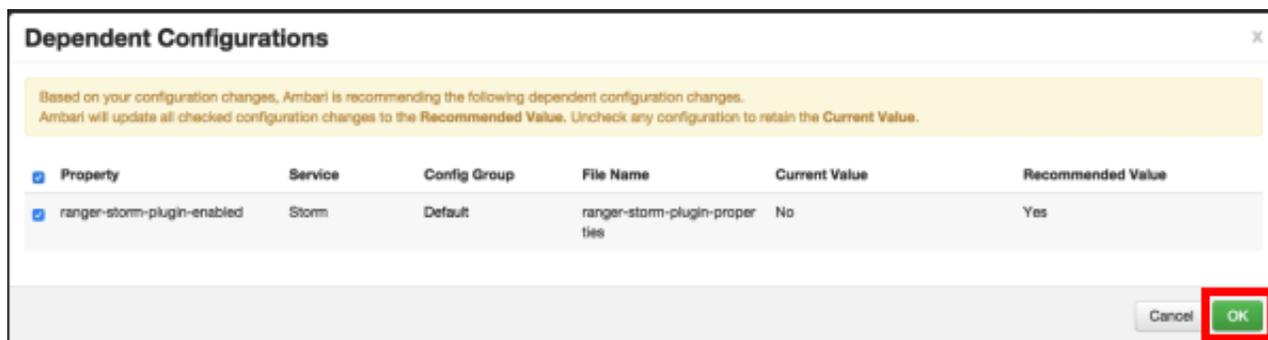
5. Click **OK** on the Save Configuration Changes pop-up.
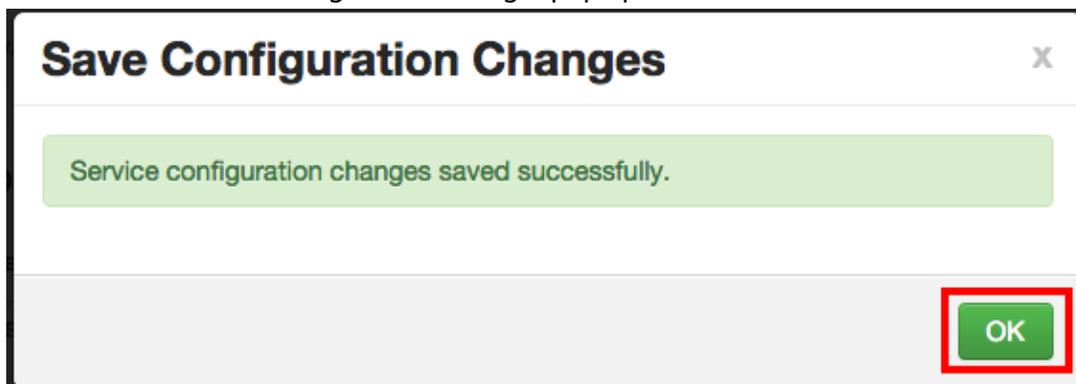
6. Select **Storm** in the navigation menu, then select **Restart > Restart All Affected** to restart the Storm service and load the new configuration.
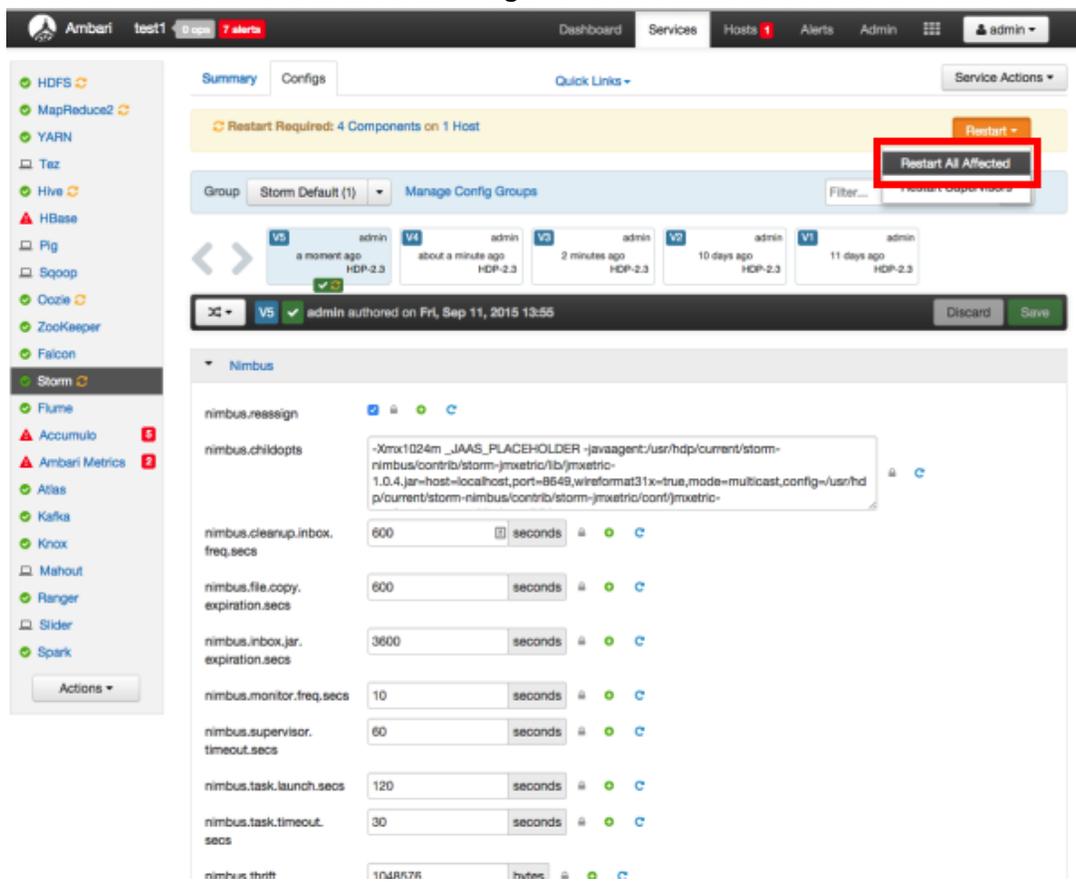
7. Click **Confirm Restart All** on the confirmation pop-up to confirm the Storm restart.
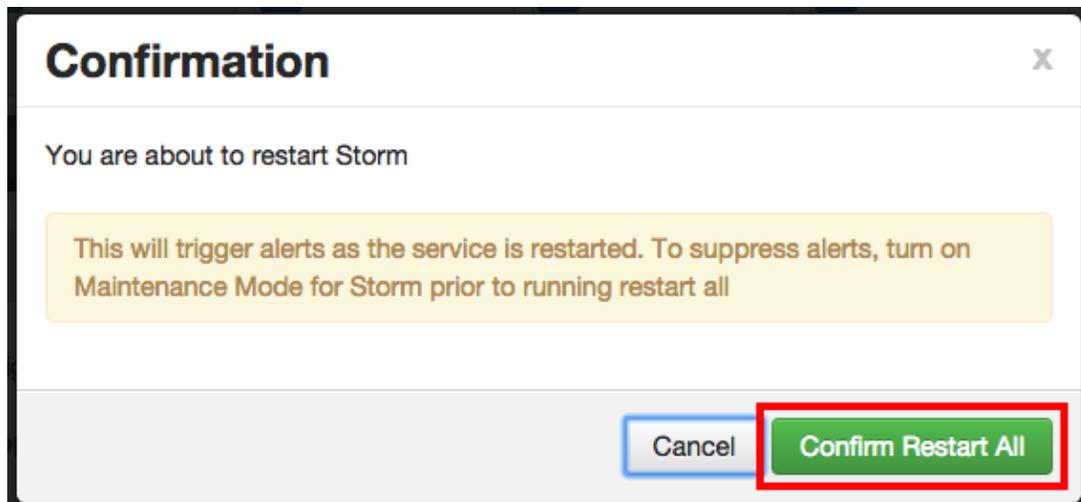
8. After Storm restarts, the Ranger plugin for Storm will be enabled.

5.3.8.3. NiFi

Use the following steps to enable the Ranger NiFi plugin.

1. On the Ranger Configs page, select the **Ranger Plugin** tab.

2. Under NiFi Ranger Plugin, select **ON**, then click **Save** in the black menu bar.
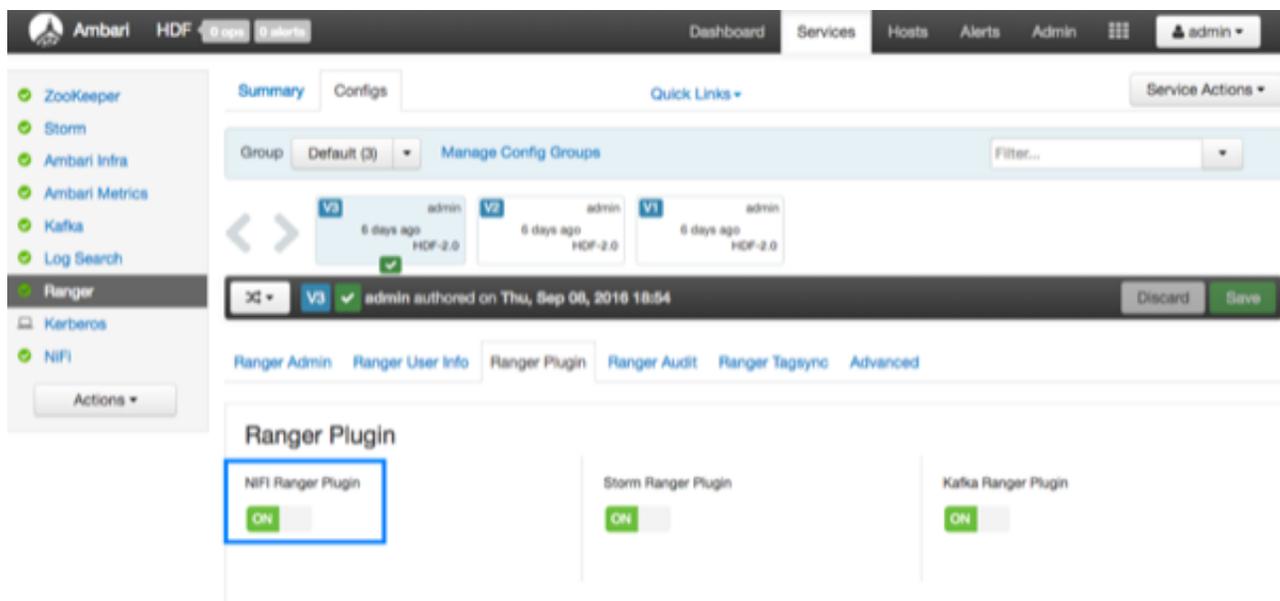
3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.

4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.

5. Click **OK** on the **Save Configuration Changes** pop-up.
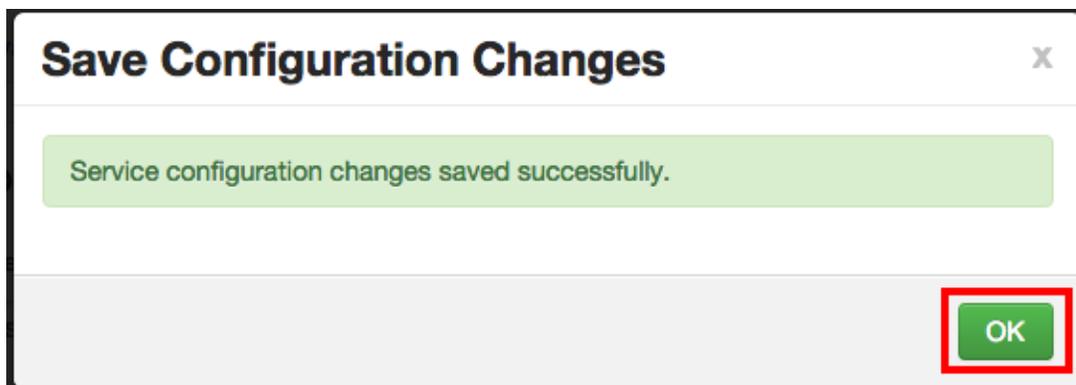
6. From the left navigation menu click **Actions**, then **Restart All Required** to restart NiFi and any additional plugins you have enabled.
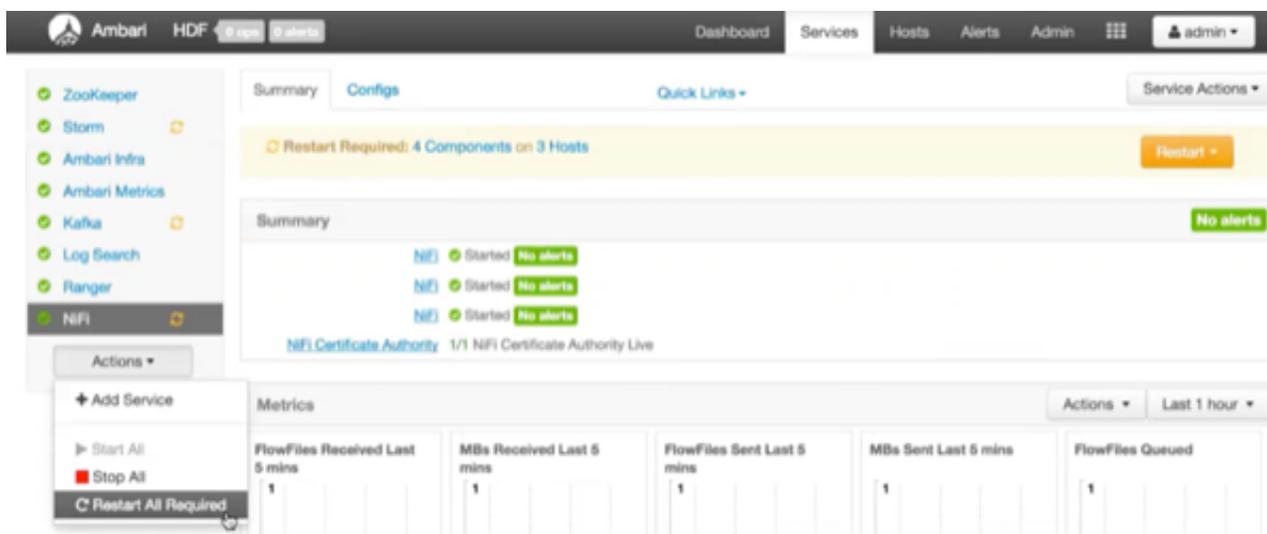
7. Click **Confirm Restart All** on the confirmation pop-up to confirm the NiFi restart along with any other services requiring a restart.
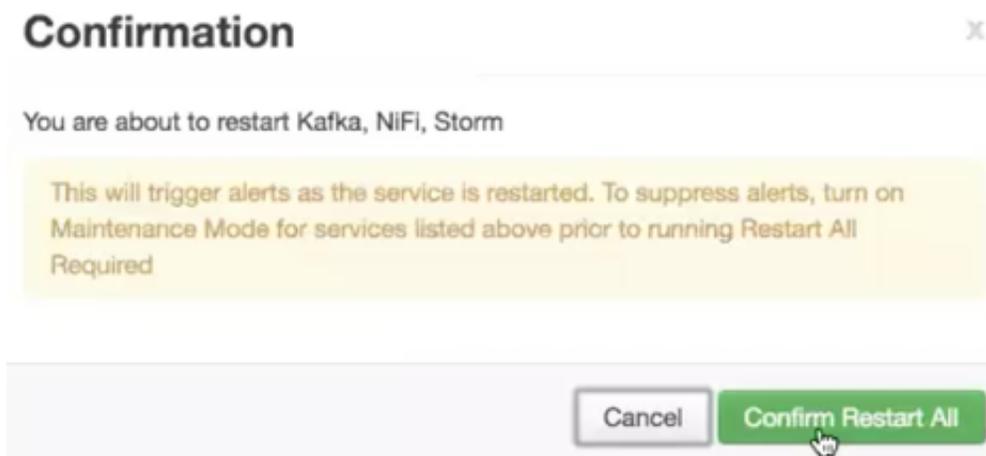
8. After your services restart, the Ranger plugin is enabled.

5.4. Configuring NiFi to Use Ranger for Managing Group Based Access Policies

About This Task

You can configure NiFi to use Ranger to manage one or more external group based access policies. You can perform the confirmation either in the NiFi `authorizers.xml` file or in Ambari Configs.

Before You Begin

• You are running LDAP

• You are using Ranger's user sync to pull LDAP users and groups into Ranger and you have defined your group based access policies in Ranger.

• You are already using Ranger based authorization in NiFi/HDF, and you want to take advantage of group based access policies.

Steps for Configuration using the NiFi authorizers.xml file

1. Open the NiFi `authorizers.xml` file, located in the NiFi `conf` directory

2. Define a User Group Provider (`userGroupProvider`) to bind to the same LDAP instance with which Ranger is configured.

   For example:

```
<userGroupProvider>
        <identifier>ldap-user-group-provider</identifier>
        <class>org.apache.nifi.ldap.tenants.LdapUserGroupProvider</class>
        <property name="Authentication Strategy">ANONYMOUS</property>

        <property name="Manager DN"></property>
        <property name="Manager Password"></property>

        <property name="TLS - Keystore"></property>
        <property name="TLS - Keystore Password"></property>
        <property name="TLS - Keystore Type"></property>
        <property name="TLS - Truststore"></property>
        <property name="TLS - Truststore Password"></property>
        <property name="TLS - Truststore Type"></property>
        <property name="TLS - Client Auth"></property>
        <property name="TLS - Protocol"></property>
        <property name="TLS - Shutdown Gracefully"></property>

        <property name="Referral Strategy">FOLLOW</property>
        <property name="Connect Timeout">10 secs</property>
        <property name="Read Timeout">10 secs</property>

        <property name="Url">ldap://localhost:10389</property>
        <property name="Page Size"></property>
        <property name="Sync Interval">30 mins</property>

        <property name="User Search Base">ou=Users,dc=local</property>
```

```
        <property name="User Object Class">posixAccount</property>
        <property name="User Search Scope">ONE_LEVEL</property>
        <property name="User Search Filter"></property>
        <property name="User Identity Attribute">cn</property>
        <property name="User Group Name Attribute"></property>
        <property name="User Group Name Attribute - Referenced Group
Attribute"></property>

        <property name="Group Search Base">ou=Groups,dc=local</property>
        <property name="Group Object Class">posixGroup</property>
        <property name="Group Search Scope">ONE_LEVEL</property>
        <property name="Group Search Filter"></property>
        <property name="Group Name Attribute">cn</property>
        <property name="Group Member Attribute">memberUid</property>
        <property name="Group Member Attribute - Referenced User
Attribute">uid</property>
    </userGroupProvider>
```

> **Note**
>
> Ensure that the LDAP configuration you have set for Ranger is also set up for NiFi.

3. Update the `ranger-provider` information:

   • Update the Ranger provider class name.

   • Add a user group provider property, referencing the user group you defined in Step 1.

   For example:

```
<authorizer>
          <identifier>ranger-provider</identifier>
          <class>org.apache.nifi.ranger.authorization.
ManagedRangerAuthorizer</class>       <!-- UPDATE CLASS NAME -->
          <property name="User Group Provider">ldap-user-group-provider</
property>        <!-- REFERENCE USER GROUP PROVIDER From Step 1 -->
          <property name="Ranger Audit Config Path">…</property>
          <property name="Ranger Security Config Path">…</property>
          <property name="Ranger Service Type">…</property>
          <property name="Ranger Application Id">…</property>
          <property name="Ranger Admin Identity">…</property>
    </authorizer>
```

Steps for Configuration using Ambari Configs

1. From the Ambari UI, go to the NiFi Configs tab.

2. Add the `authorizers.xml` information you created above to the **Advanced nifi-authorizers-env** field.

3. Click **Save**.

5.5. Adding Users to Ranger

After installing Ranger and enabling the Ranger plugins, add users to Ranger.

1. From the Ranger UI, click **Settings**, then **Users/Groups**.

2. Click **Add New User**.

3. In the **User Detail** screen, provide:

   • User Name in the `CN=<host> OU=<realm>` format. If you have set up identity
     mapping, use the `<host>@<realm>` format.

   • The password the user will use to access Ranger.

   • The Role you want the user to have.

   • The Group you want the user to be part of.

6. Authorization with Ranger

6.1. Creating Policies for NiFi Access

Once you have set up Ranger to manage NiFi authorization, you must create policies so that users can access and operate on the NiFi canvas.

- Creating Policies to View NiFi [76]

- Allowing Users Read and Write Access [78]

6.1.1. Creating Policies to View NiFi

To allow users to view the NiFi UI, create the following policies for each host:

- /flow – read

- /proxy – read/write

To create policies:

1. From the Ranger console, click the NiFi Ranger plugin.

2. From the **List of Policies** page, click **Add New Policy**.

3. In the **Policy Details** dialog, create the /flow and /proxy policies.

4. To create the /flow policy:

    a. Provide the following information:

        • **Policy Name** – /flow

        • **NiFi Resource Identifier**- /flow

        • Select Users and Groups you want to immediately add.

        • Add **Read** permission

    b. Click **Add.**

5. To create the /proxy policy:

    a. Provide the following information:

        • **Policy Name** – /proxy

        • **NiFi Resource Identifier**- /proxy

        • Select Users and Groups you want to immediately add.

        • Add **Read** and **Write** permissions.

    b. Click **Add.**

6.1.2. Allowing Users Read and Write Access

To allow users complete read and write access to NiFi:

1. From the **Policy Details** page, select the global NiFi policy.

   • **Policy Name** – all - nifi-resource

   • **NiFi Resource Identifier** – x

2. Add users.

3. Add **Read** and **Write** permissions.

6.2. Create a Kafka Policy

To add a new policy to an existing Kafka service:

1. On the Service Manager page, select an existing service under Kafka.
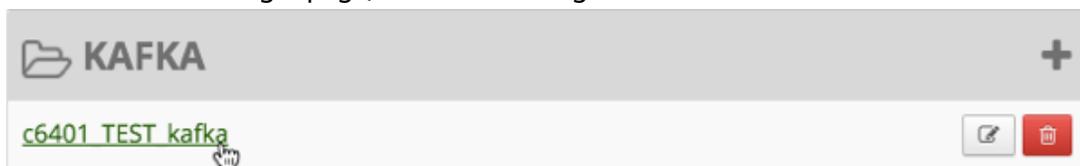
The List of Policies page appears.
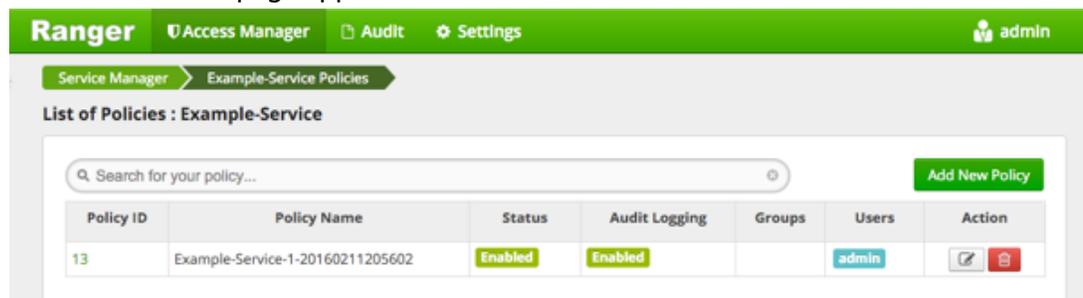
2. Click **Add New Policy**.

The Create Policy page appears.
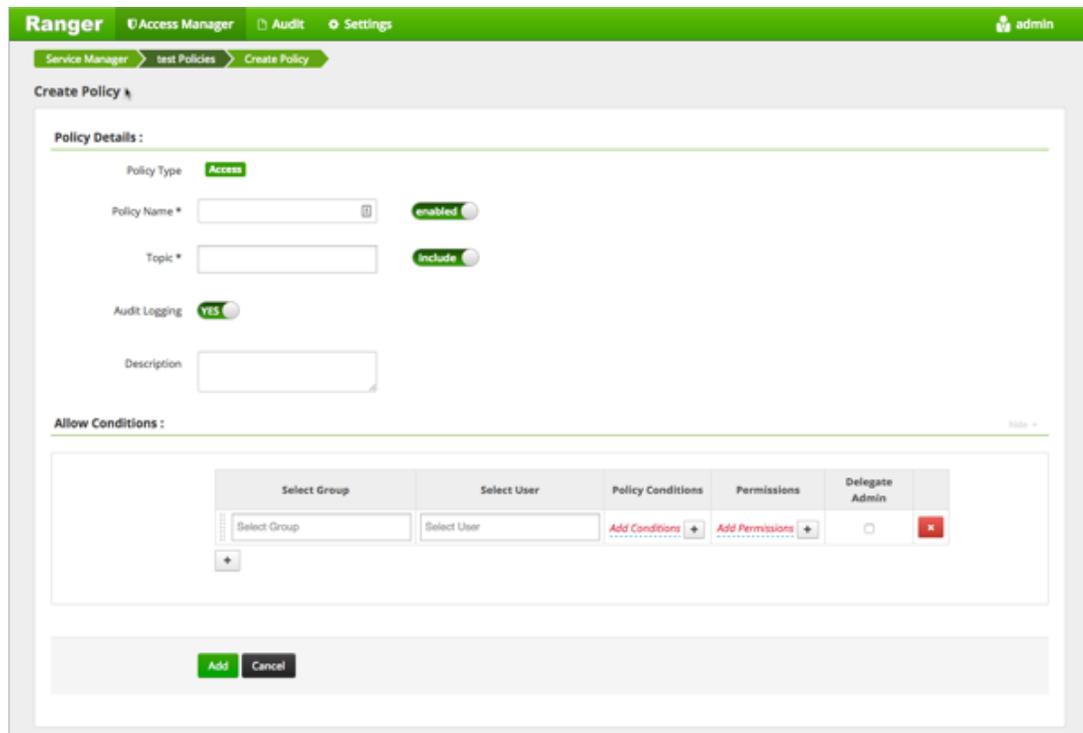
3. Complete the Create Policy page as follows:

Table 6.1. Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
Topic	A topic is a category or feed name to which messages are published.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 6.2. Allow Conditions

Label	Description
Select Group	Specify the group to which this policy applies. To designate the group as an Administrator for the chosen resource, specify Admin permissions. (Administrators can create child policies based on existing policies). The **public** group contains all users, so granting access to the public group grants access to all users.
Select User	Specify a particular user to which this policy applies (outside of an already-specified group) OR designate a particular user as Admin for this policy. (Administrators can create child policies based on existing policies).
Policy Conditions	Specify IP address range.
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.

Label	Description
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

4. You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.

5. Click **Add**.

6.3. Create a Storm Policy

To add a new policy to an existing Storm service:

1. On the Service Manager page, select an existing service under Storm.
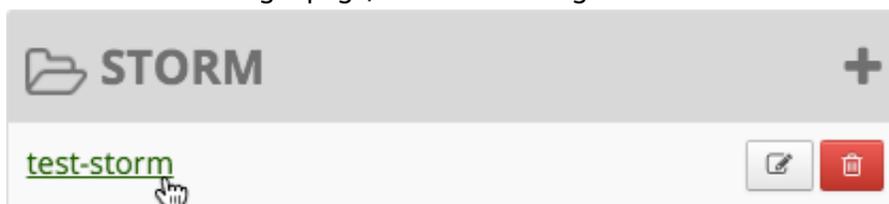
The List of Policies page appears.
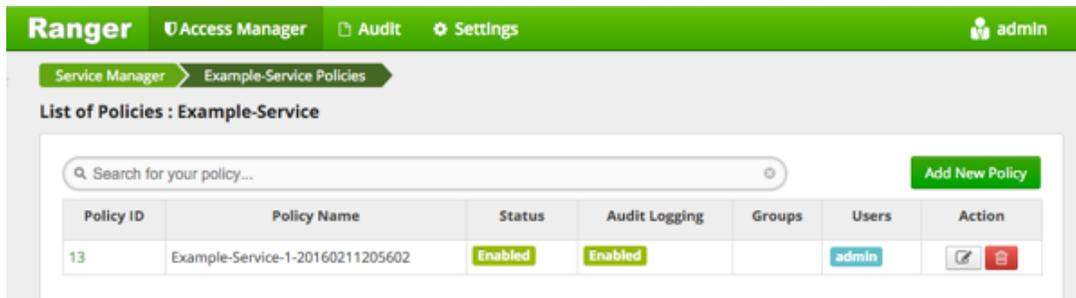
2. Click **Add New Policy**.

The Create Policy page appears.
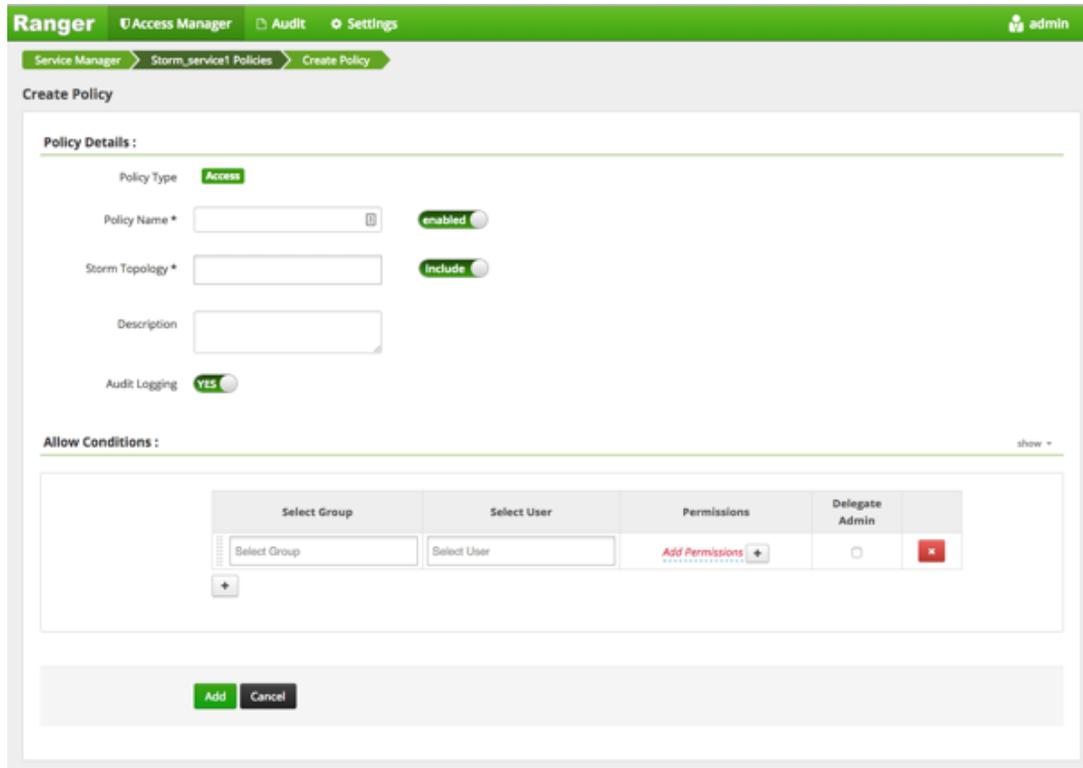
3. Complete the Create Policy page as follows:

Table 6.3. Policy Details

Label	Description
Policy Name	Enter an appropriate policy name. This name is cannot be duplicated across the system. This field is mandatory.
Storm Topology	Enter an appropriate Topology Name.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 6.4. Allow Conditions

Label	Description
Select Group	Specify the group to which this policy applies. To designate the group as an Administrator for the chosen resource, specify Admin permissions. (Administrators can create child policies based on existing policies). The **public** group contains all users, so granting access to the public group grants access to all users.
Select User	Specify a particular user to which this policy applies (outside of an already-specified group) OR designate a particular user as Admin for this policy. (Administrators can create child policies based on existing policies).
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated

Label	Description
	admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

Since Storm does not provide a command line methodology for assigning privileges or roles to users, the User and Group Permissions portion of the Storm Create Policy form is especially important.

Table 6.5. Storm User and Group Permissions

Actions	Description
File upload	Allows a user to upload files.
Get Nimbus Conf	Allows a user to access Nimbus configurations.
Get Cluster Info	Allows a user to get cluster information.
File Download	Allows a user to download files.
Kill Topology	Allows a user to kill the topology.
Rebalance	Allows a user to rebalance topologies.
Activate	Allows a user to activate a topology.
Deactivate	Allows a user to deactivate a topology.
Get Topology Conf	Allows a user to access a topology configuration.
Get Topology	Allows a user to access a topology.
Get User Topology	Allows a user to access a user topology.
Get Topology Info	Allows a user to access topology information.
Upload New Credential	Allows a user to upload a new credential.
Admin	Provides a user with delegated admin access.

4. You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.

5. Click **Add**.

7. NiFi Authorization

After you have configured NiFi to run securely and with an authentication mechanism, you must configure who has access to the system, and the level of their access. You can do this using *multi-tenant authorization*. Multi-tenant authorization enables multiple groups of users (tenants) to command, control, and observe different parts of the dataflow, with varying levels of authorization. When an authenticated user attempts to view or modify a NiFi resource, the system checks whether the user has privileges to perform that action. These privileges are defined by policies that you can apply system-wide or to individual components.

7.1. Authorizer Configuration

An *authorizer* grants users the privileges to manage users and policies by creating preliminary authorizations at startup.

Authorizers are configured using two properties in the *nifi.properties* file:

- The `nifi.authorizer.configuration.file` property specifies the configuration file where authorizers are defined. By default, the *authorizers.xml* file located in the root installation conf directory is selected.

- The `nifi.security.user.authorizer` property indicates which of the configured authorizers in the *authorizers.xml* file to use.

7.2. Authorizers.xml Setup

The *authorizers.xml* file is used to define and configure available authorizers. The default authorizer is the FileAuthorizer, however, you can develop additional authorizers as extensions. The FileAuthorizer has the following properties:

- Authorizations File - The file where the FileAuthorizer stores policies. By default, the *authorizations.xml* in the *conf* directory is chosen.

- Users File - The file where the FileAuthorizer stores users and groups. By default, the *users.xml* in the *conf* directory is chosen.

- Initial Admin Identity - The identity of an initial admin user that is granted access to the UI and given the ability to create additional users, groups, and policies. This property is only used when there are no other users, groups, and policies defined.

- Legacy Authorized Users File - The full path to an existing authorized-users.xml that is automatically converted to the multi-tenant authorization model. This property is only used when there are no other users, groups, and policies defined.

- Node Identity - The identity of a NiFi cluster node. When clustered, a property for each node should be defined, so that every node knows about every other node. If not clustered, these properties can be ignored.

7.2.1. Initial Admin Identity (New NiFi Instance)

If you are setting up a secured NiFi instance for the first time, you must manually designate an "Initial Admin Identity" in the *authorizers.xml* file. This initial admin user is granted access to the UI and given the ability to create additional users, groups, and policies. The value of this property could be a DN (when using certificates or LDAP) or a Kerberos principal. If you are the NiFi administrator, add yourself as the "Initial Admin Identity".

Here is an example LDAP entry using the name John Smith:

```
<authorizer>
        <identifier>file-provider</identifier>
        <class>org.apache.nifi.authorization.FileAuthorizer</class>
        <property name="Authorizations File">./conf/authorizations.xml</
property>
        <property name="Users File">./conf/users.xml</property>
        <property name="Initial Admin Identity">cn=John Smith,ou=people,dc=
example,dc=com</property>
        <property name="Legacy Authorized Users File"></property>
        <!--
        <property name="Node Identity 1"></property>
        <property name="Node Identity 2"></property>
        -->
    </authorizer>
</authorizers>
```

Here is an example Kerberos entry using the name John Smith and realm `NIFI.APACHE.ORG`:

```
<authorizer>
        <identifier>file-provider</identifier>
        <class>org.apache.nifi.authorization.FileAuthorizer</class>
        <property name="Authorizations File">./conf/authorizations.xml</
property>
        <property name="Users File">./conf/users.xml</property>
        <property name="Initial Admin Identity">johnsmith@NIFI.APACHE.ORG</
property>
        <property name="Legacy Authorized Users File"></property>
        <!--
        <property name="Node Identity 1"></property>
        <property name="Node Identity 2"></property>
        -->
    </authorizer>
</authorizers>
```

After you have edited and saved the *authorizers.xml* file, restart NiFi. The "Initial Admin Identity" user and administrative policies are added to the *users.xml* and *authorizations.xml* files during restart. Once NiFi starts, the "Initial Admin Identity" user is able to access the UI and begin managing users, groups, and policies.

Note

For a brand new secure flow, providing the "Initial Admin Identity" gives that user access to get into the UI and to manage users, groups and policies. But if that user wants to start modifying the flow, they need to grant

themselves policies for the root process group. The system is unable to do this automatically because in a new flow the UUID of the root process group is not permanent until the flow.xml.gz is generated. If the NiFi instance is an upgrade from an existing flow.xml.gz or a 1.x instance going from unsecure to secure, then the "Initial Admin Identity" user is automatically given the privileges to modify the flow.

7.2.2. Legacy Authorized Users (NiFi Instance Upgrade)

If you are upgrading from a 0.x NiFi instance, you can convert your previously configured users and roles to the multi-tenant authorization model. In the *authorizers.xml* file, specify the location of your existing *authorized-users.xml* file in the "Legacy Authorized Users File" property.

Here is an example entry:

```
<authorizers>
    <authorizer>
        <identifier>file-provider</identifier>
        <class>org.apache.nifi.authorization.FileAuthorizer</class>
        <property name="Authorizations File">./conf/authorizations.xml</
property>
        <property name="Users File">./conf/users.xml</property>
        <property name="Initial Admin Identity"></property>
        <property name="Legacy Authorized Users File">/Users/johnsmith/
config_files/authorized-users.xml</property>
    </authorizer>
</authorizers>
```

After you have edited and saved the *authorizers.xml* file, restart NiFi. Users and roles from the *authorized-users.xml* file are converted and added as identities and policies in the *users.xml* and *authorizations.xml* files. Once the application starts, users who previously had a legacy Administrator role can access the UI and begin managing users, groups, and policies.

Here is a summary of policies assigned to each legacy role if the NiFi instance has an existing flow.xml.gz:

	Admin	DFM	Monitor	Provenance	NiFi	Proxy
view the UI	*	*	*			
view the controller	*	*	*		*	
modify the controller		*				
view system diagnostics		*	*			
view the dataflow	*	*	*			
modify the dataflow		*				
view the users/groups	*					
modify the users/groups	*					

	Admin	DFM	Monitor	Provenance	NiFi	Proxy
view policies	*					
modify policies	*					
query provenance				*		
access restricted components		*				
view the data		*		*		*
modify the data		*				*
retrieve site-to-site details					*	
send proxy user requests						*

For details on the policies in the table, see Access Policies.

NiFi fails to restart if values exist for both the "Initial Admin Identity" and "Legacy Authorized Users File" properties. You can specify only one of these values to initialize authorizations.

Do not manually edit the *authorizations.xml* file. Create authorizations only during initial setup and afterwards using the NiFi UI.

7.2.3. Cluster Node Identities

If you are running NiFi in a clustered environment, you must specify the identities for each node. The authorization policies required for the nodes to communicate are created during startup.

For example, if you are setting up a 2 node cluster with the following DNs for each node:

```
cn=nifi-1,ou=people,dc=example,dc=com
cn=nifi-2,ou=people,dc=example,dc=com
```

```
<authorizer>
        <identifier>file-provider</identifier>
        <class>org.apache.nifi.authorization.FileAuthorizer</class>
        <property name="Authorizations File">./conf/authorizations.xml</
property>
        <property name="Users File">./conf/users.xml</property>
        <property name="Initial Admin Identity">johnsmith@NIFI.APACHE.ORG</
property>
        <property name="Legacy Authorized Users File"></property>
        <property name="Node Identity 1">cn=nifi-1,ou=people,dc=example,dc=
com</property>
        <property name="Node Identity 2">cn=nifi-2,ou=people,dc=example,dc=
com</property>
    </authorizer>
</authorizers>
```

In a cluster, all nodes must have the same *authorizations.xml*. If a node has a different *authorizations.xml*, it cannot join the cluster. The only exception is if a node has an empty

authorizations.xml. In this scenario, the node inherits the *authorizations.xml* from the cluster.

Now that initial authorizations have been created, additional users, groups and authorizations can be created and managed in the NiFi UI.

7.3. Configuring Users & Access Policies

This section describes:

• How to create users and groups

• How access policies are used to define authorizations

• How to configure access policies by walking through specific examples

Instructions requiring interaction with the UI assume the application is being accessed by User1, a user with administrator privileges, such as the "Initial Admin Identity" user or a converted legacy admin user (see Authorizers.xml Setup).

7.3.1. Creating Users and Groups

From the UI, select "Users" from the Global Menu. This opens a dialog to create and manage users and groups.
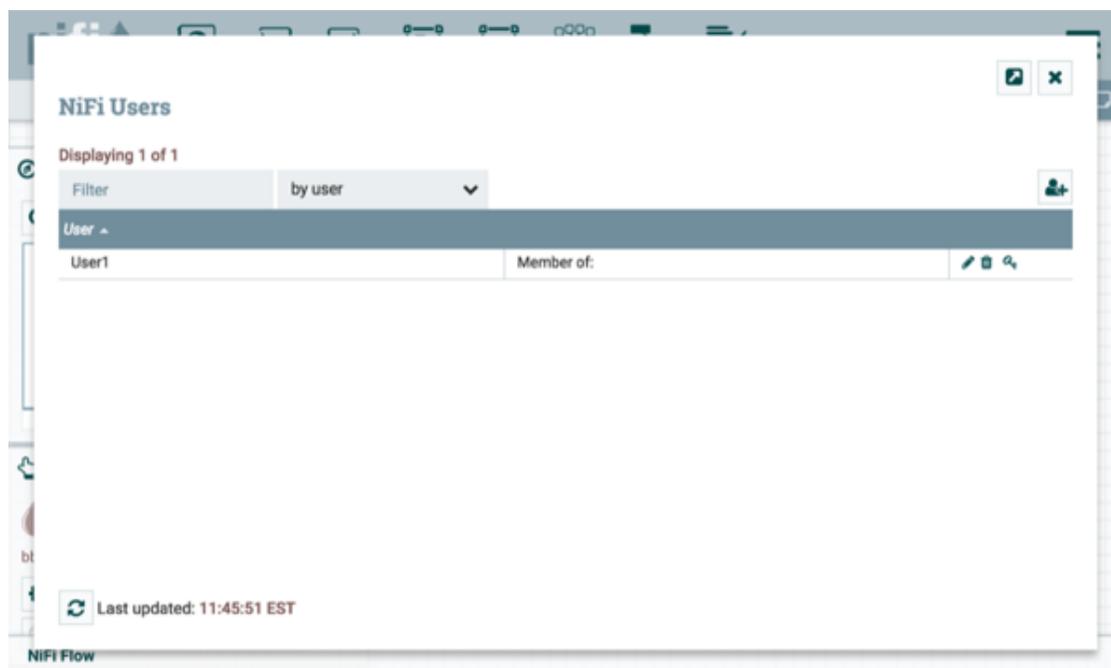
Click the Add icon

( ).

To create a user, enter the *Identity* information relevant to the authentication method chosen to secure your NiFi instance. Click OK.
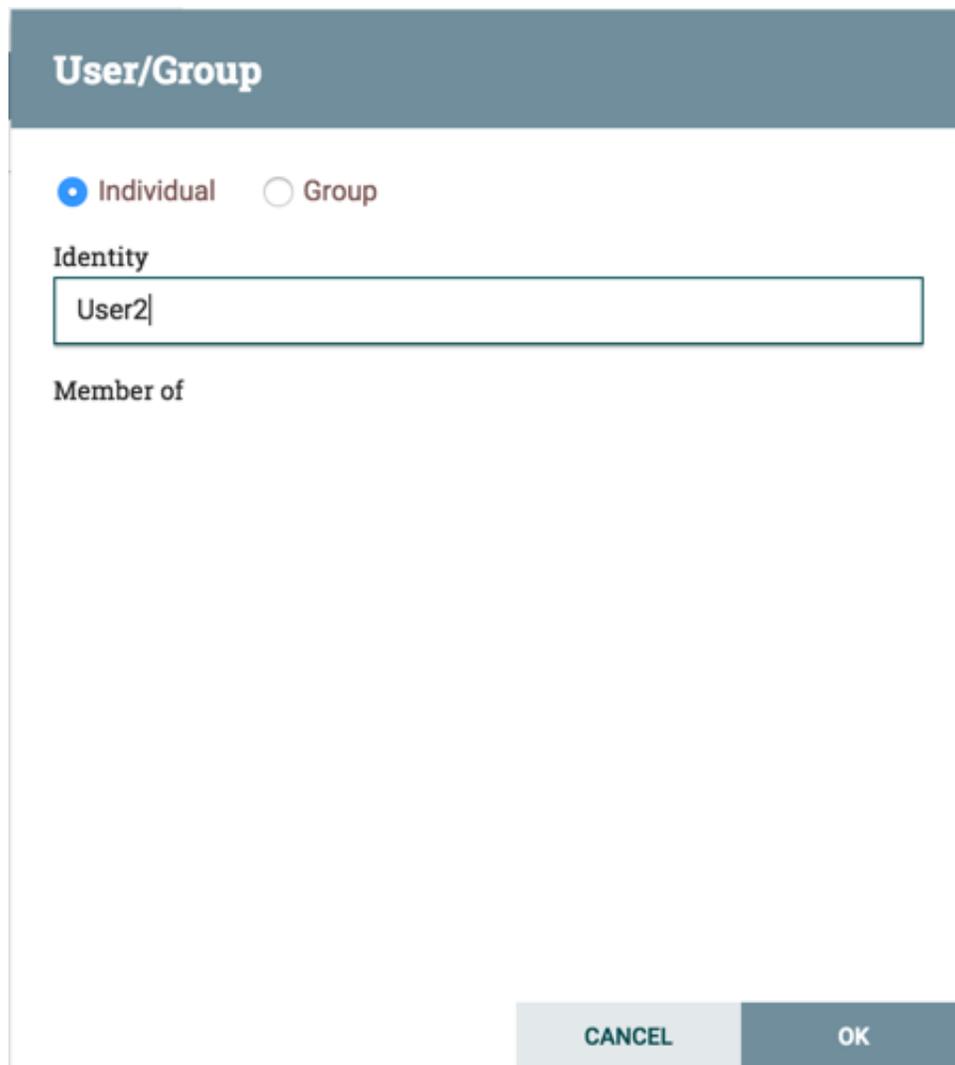
To create a group, select the "Group" radio button, enter the name of the group and select the users to be included in the group. Click OK.

7.3.2. Access Policies

You can manage the ability for users and groups to view or modify NiFi resources using *access policies*. There are two types of access policies that can be applied to a resource:

- View - If a view policy is created for a resource, only the users or groups that are added to that policy are able to see the details of that resource.

- Modify - If a resource has a modify policy, only the users or groups that are added to that policy can change the configuration of that resource.

You can create and apply access policies on both global and component levels.

7.3.2.1. Global Access Policies

Global access policies govern the following system level authorizations:

Policy	Privilege	Global Menu Selection
view the UI	Allow users to view the UI	N/A

Policy	Privilege	Global Menu Selection
access the controller	Allows users to view/modify the controller including Reporting Tasks, Controller Services, and Nodes in the Cluster	Controller Settings
query provenance	Allows users to submit a Provenance Search and request Event Lineage	Data Provenance
access restricted components	Allows users to create/modify restricted components assuming otherwise sufficient permissions	N/A
access all policies	Allows users to view/modify the policies for all components	Policies
access users/user groups	Allows users to view/modify the users and user groups	Users
retrieve site-to-site details	Allows other NiFi instances to retrieve Site-To-Site details	N/A
view system diagnostics	Allows users to view System Diagnostics	Summary
proxy user requests	Allows proxy machines to send requests on the behalf of others	N/A
access counters	Allows users to view/modify Counters	Counters

7.3.2.2. Component Level Access Policies

Component level access policies govern the following component level authorizations:

Policy	Privilege
view the component	Allows users to view component configuration details
modify the component	Allows users to modify component configuration details
view the data	Allows user to view metadata and content for this component through provenance data and flowfile queues in outbound connections
modify the data	Allows user to empty flowfile queues in outbound connections and submit replays
view the policies	Allows users to view the list of users who can view/modify a component
modify the policies	Allows users to modify the list of users who can view/modify a component
retrieve data via site-to-site	Allows a port to receive data from NiFi instances
send data via site-to-site	Allows a port to send data from NiFi instances

You can apply access policies to all component types except connections. Connection authorizations are inferred by the individual access policies on the source and destination components of the connection, as well as the access policy of the process group containing the components. This is discussed in more detail in the Creating a Connection and Editing a Connection examples below.

7.3.2.3. Access Policy Inheritance

An administrator does not need to manually create policies for every component in the dataflow. To reduce the amount of time admins spend on authorization management, policies are inherited from parent resource to child resource. For example, if a user is given access to view and modify a process group, that user can also view and modify the

components in the process group. Policy inheritance enables an administrator to assign policies at one time and have the policies apply throughout the entire dataflow.

You can override an inherited policy (as described in the Moving a Processor example below). Overriding a policy removes the inherited policy, breaking the chain of inheritance from parent to child, and creates a replacement policy to add users as desired. Inherited policies and their users can be restored by deleting the replacement policy.

"View the policies" and "modify the policies" component-level access policies are an exception to this inherited behavior. When a user is added to either policy, they are added to the current list of administrators. They do not override higher level administrators. For this reason, only component specific administrators are displayed for the "view the policies" and "modify the policies" access policies.

You cannot modify the users/groups on an inherited policy. Users and groups can only be added or removed from a parent policy or an override policy.

7.3.3. Access Policy Configuration Examples

The most effective way to understand how to create and apply access policies is to walk through some common examples. The following scenarios assume User1 is an administrator and User2 is a newly added user that has only been given access to the UI.

Let's begin with two processors on the canvas as our starting point: GenerateFlowFile and LogAttribute.
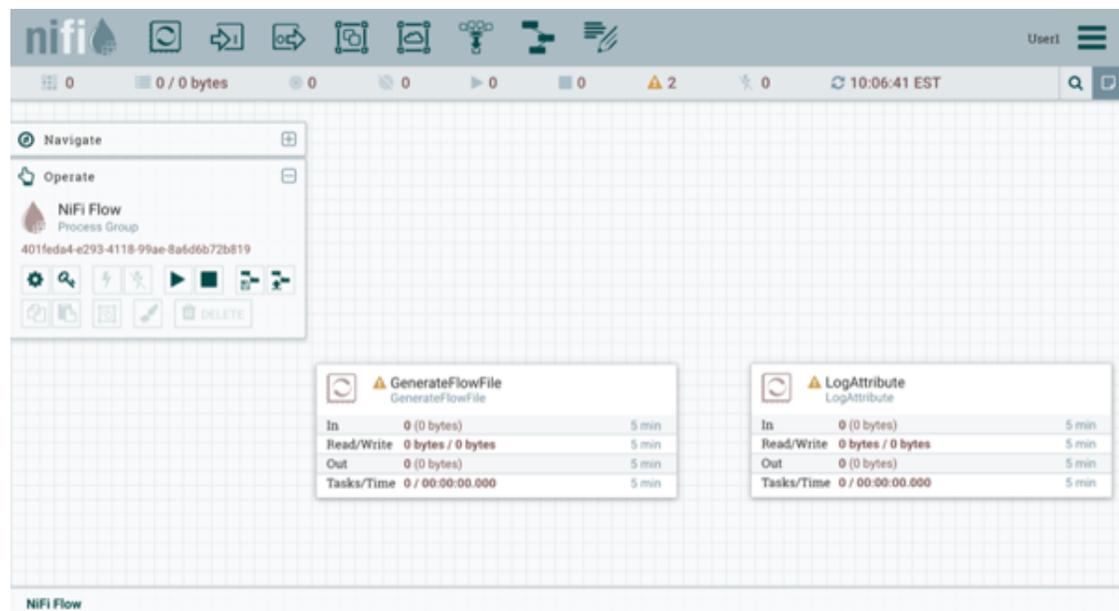
User1 can add components to the dataflow and is able to move, edit and connect all processors. The details and properties of the root process group and processors are visible to User1.
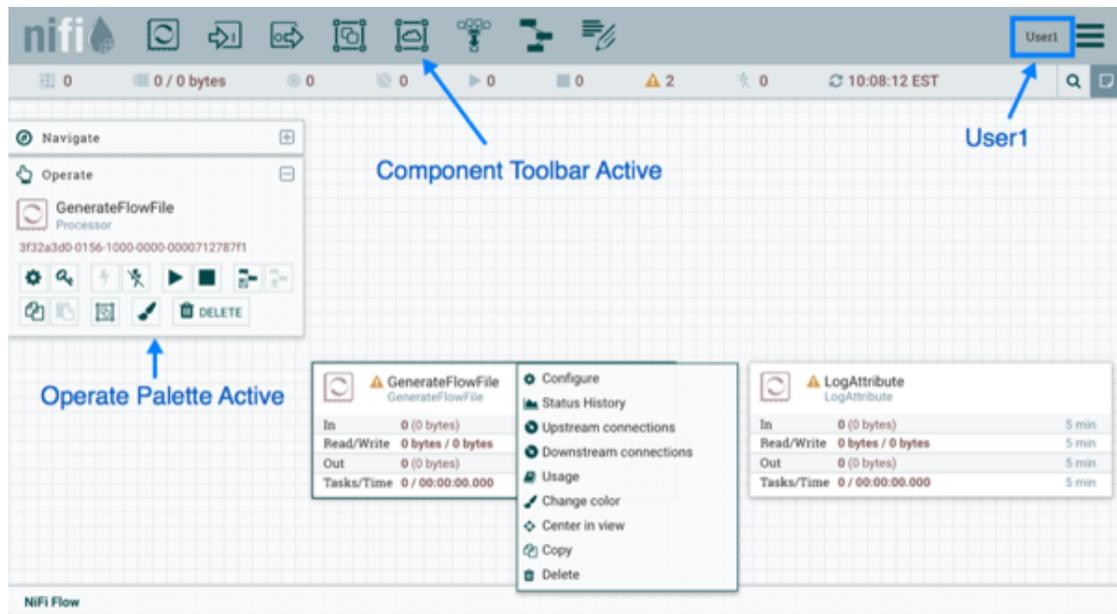
User1 wants to maintain their current privileges to the dataflow and its components.

User2 is unable to add components to the dataflow or move, edit, or connect components. The details and properties of the root process group and processors are hidden from User2.
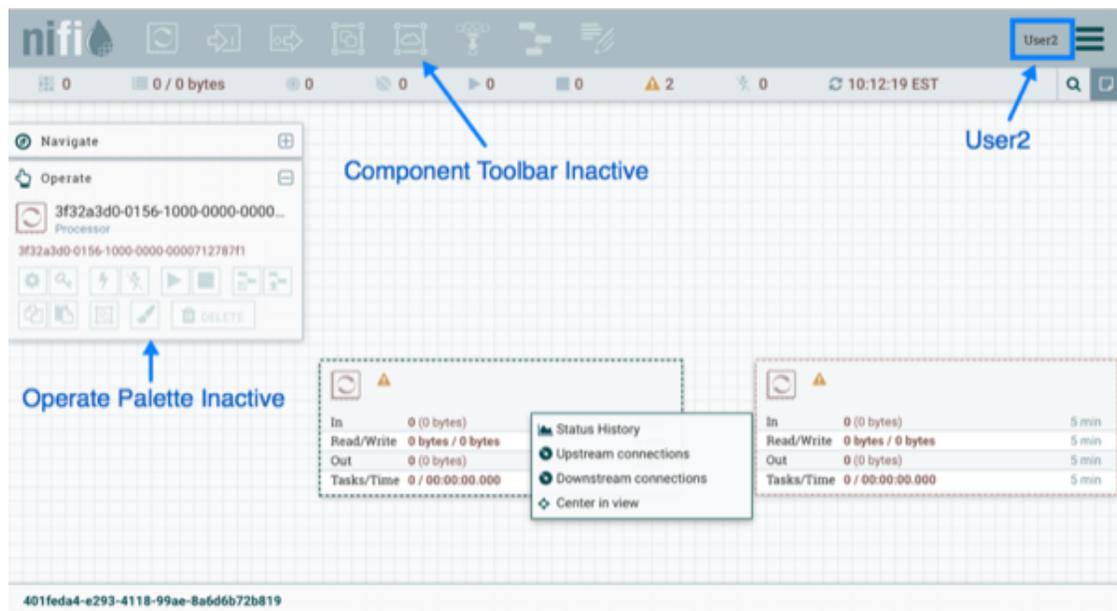
7.3.3.1. Moving a Processor

To allow User2 to move the GenerateFlowFile processor in the dataflow and only that processor, User1 performs the following steps:

1. Select the GenerateFlowFile processor so that it is highlighted.

2. Select the Access Policies icon

(   )
from the Operate palette and the Access Policies dialog opens.

3. Select "modify the component" from the policy drop-down.
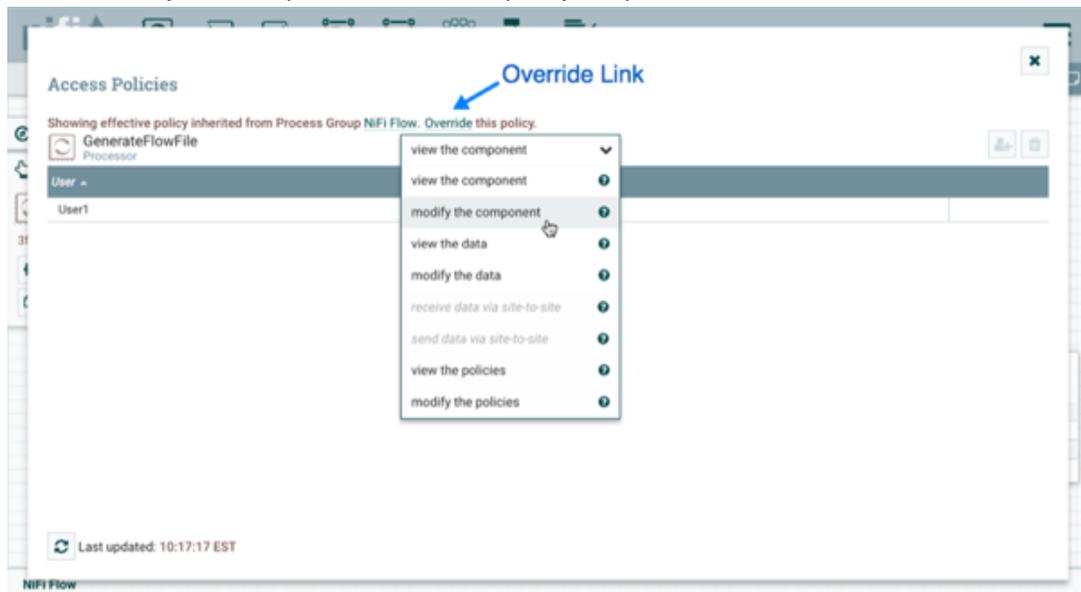
The "modify the component" policy that currently exists on the processor (child) is the "modify the component" policy inherited from the root process group (parent) on which User1 has privileges.

4. Select the Override link in the policy inheritance message. When creating the replacement policy, you are given a choice to override with a copy of the inherited policy or an empty policy.
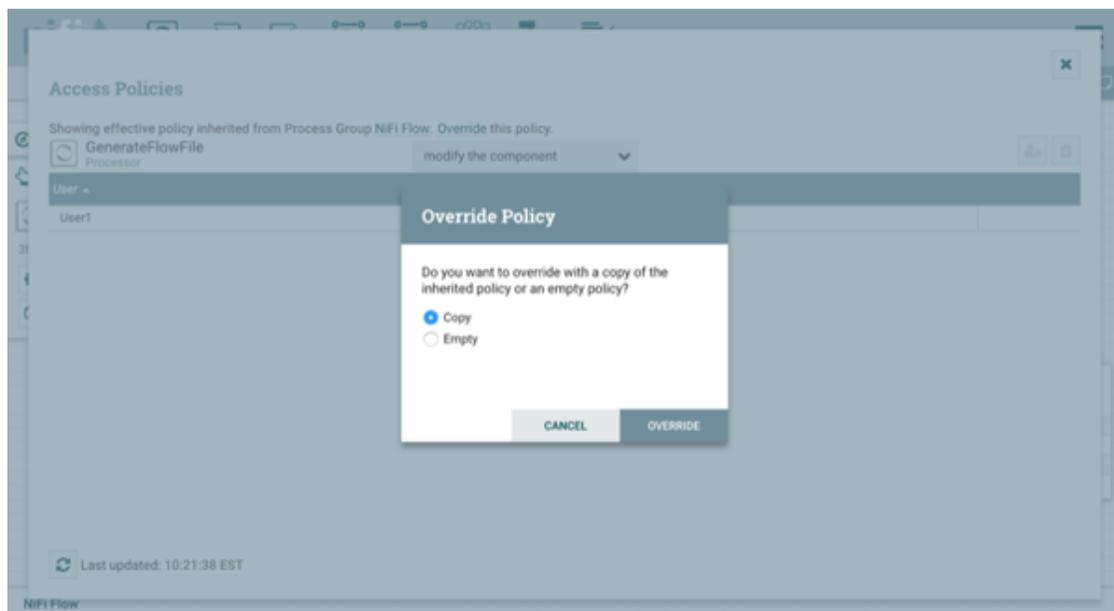
Select the Override button to create a copy.

1. On the replacement policy that is created, select the Add User icon
   (

).
   Find or enter User2 in the User Identity field and select OK.
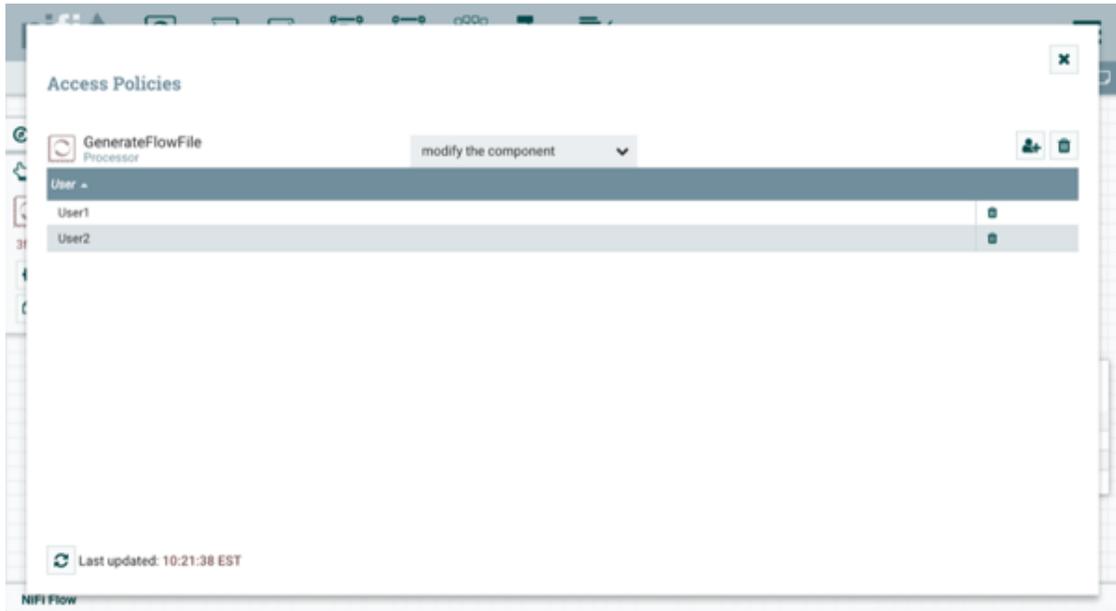
With these changes, User1 maintains the ability to move both processors on the canvas. User2 can now move the GenerateFlowFile processor but cannot move the LogAttribute processor.
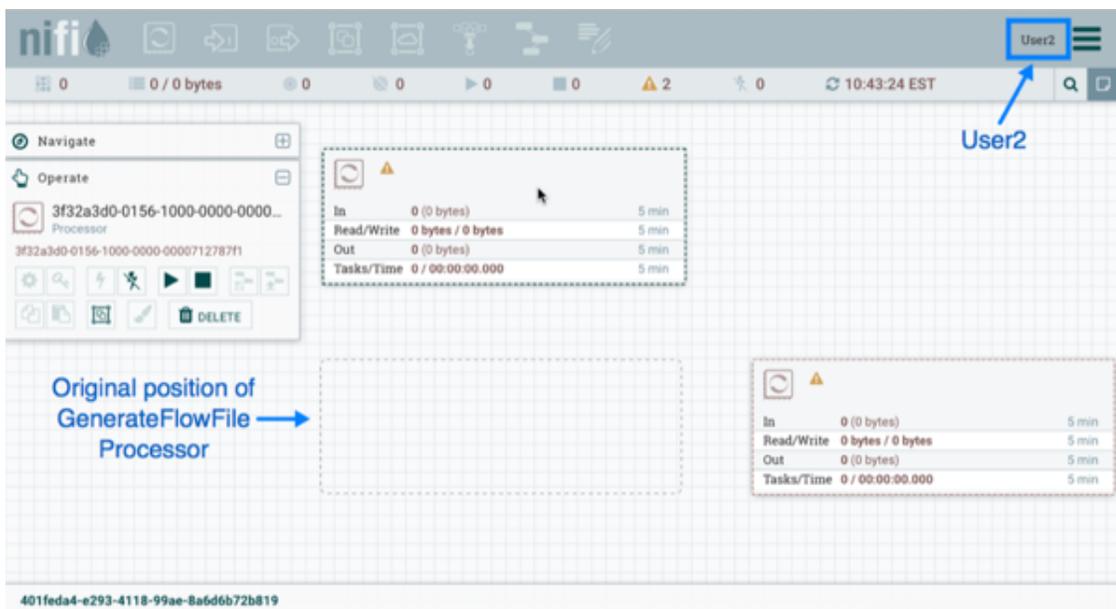
7.3.3.2. Editing a Processor

In the "Moving a Processor" example above, User2 was added to the "modify the component" policy for GenerateFlowFile. Without the ability to view the processor properties, User2 is unable to modify the processor's configuration. In order to edit a component, a user must be on both the "view the component" and "modify the component" policies. To implement this, User1 performs the following steps:

1. Select the GenerateFlowFile processor.

2. Select the Access Policies icon

    (   )
    from the Operate palette and the Access Policies dialog opens.

3. Select "view the component" from the policy drop-down.
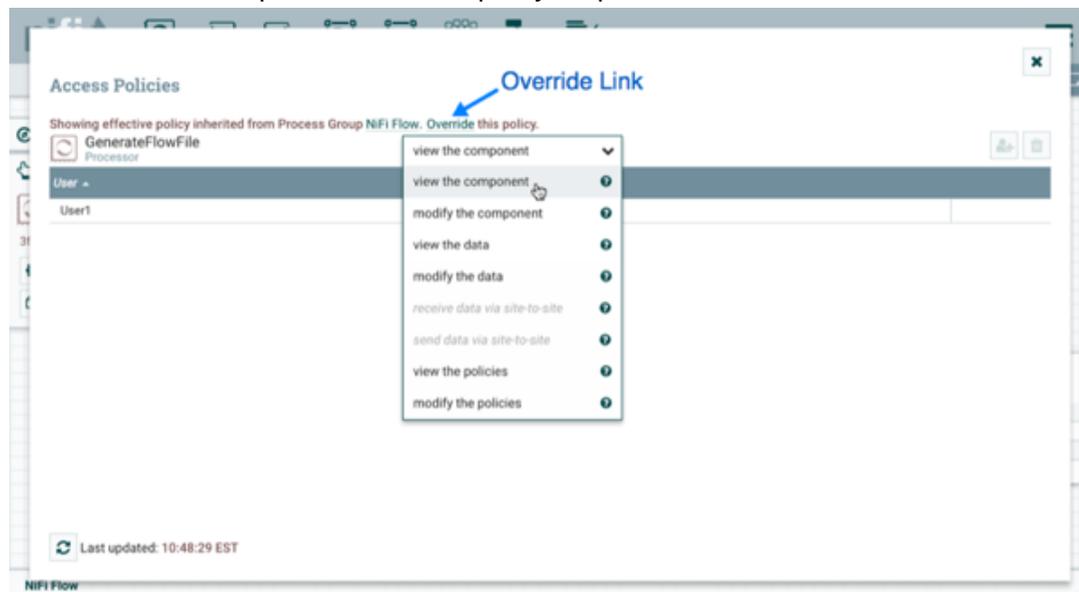
The view the component" policy that currently exists on the processor (child) is the "view the component" policy inherited from the root process group (parent) on which User1 has privileges.

4. Select the Override link in the policy inheritance message, keep the default of Copy policy and select the Override button.

5. On the override policy that is created, select the Add User icon

    (   ).
    Find or enter User2 in the User Identity field and select OK.
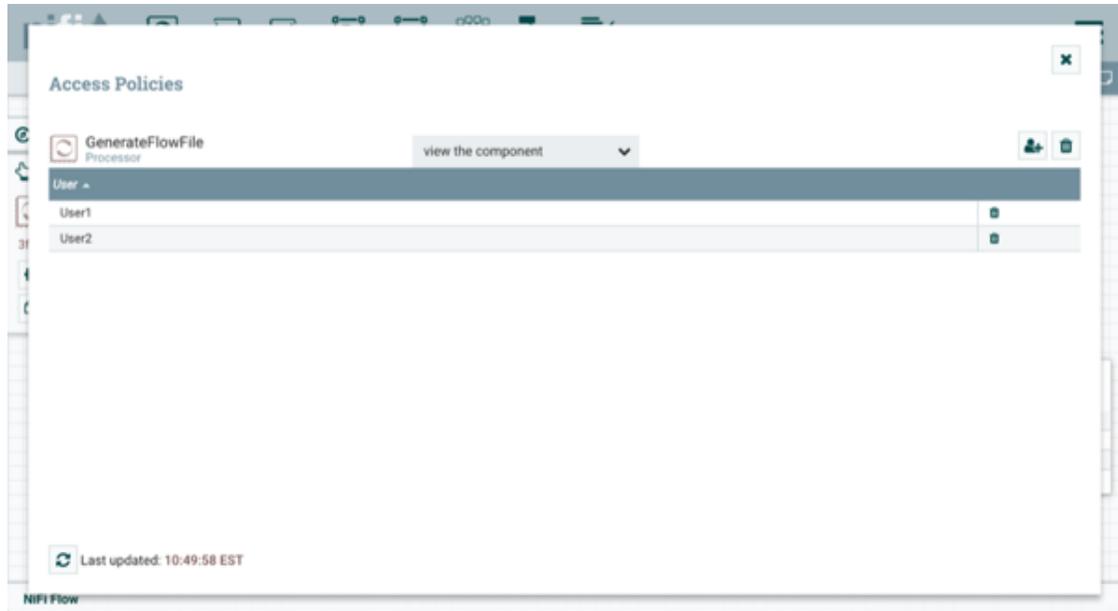
With these changes, User1 maintains the ability to view and edit the processors on the canvas. User2 can now view and edit the GenerateFlowFile processor.
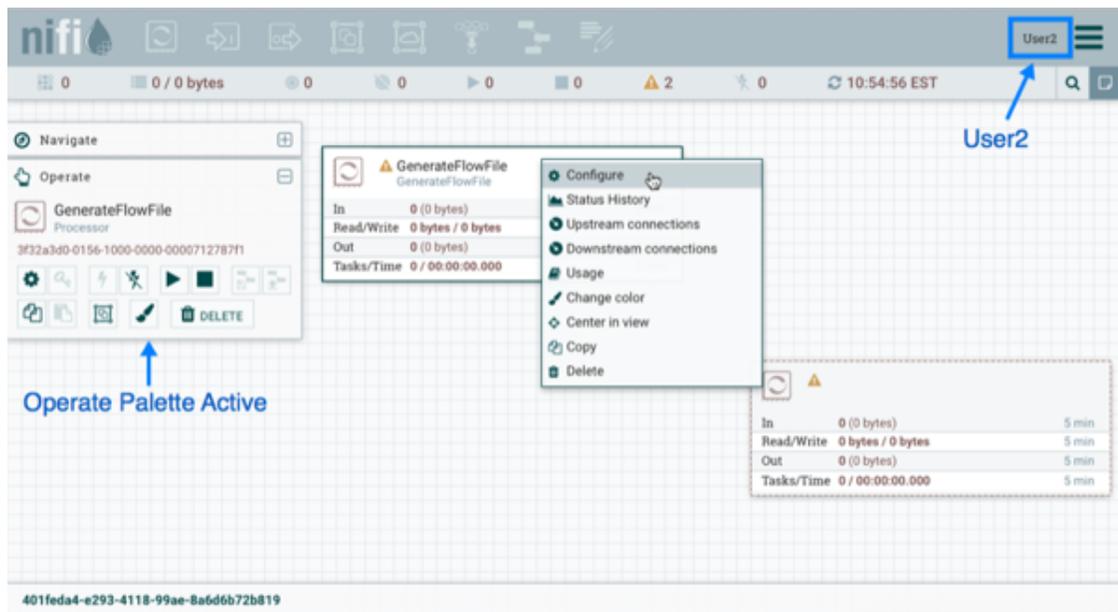
7.3.3.3. Creating a Connection

With the access policies configured as discussed in the previous two examples, User1 is able to connect GenerateFlowFile to LogAttribute:
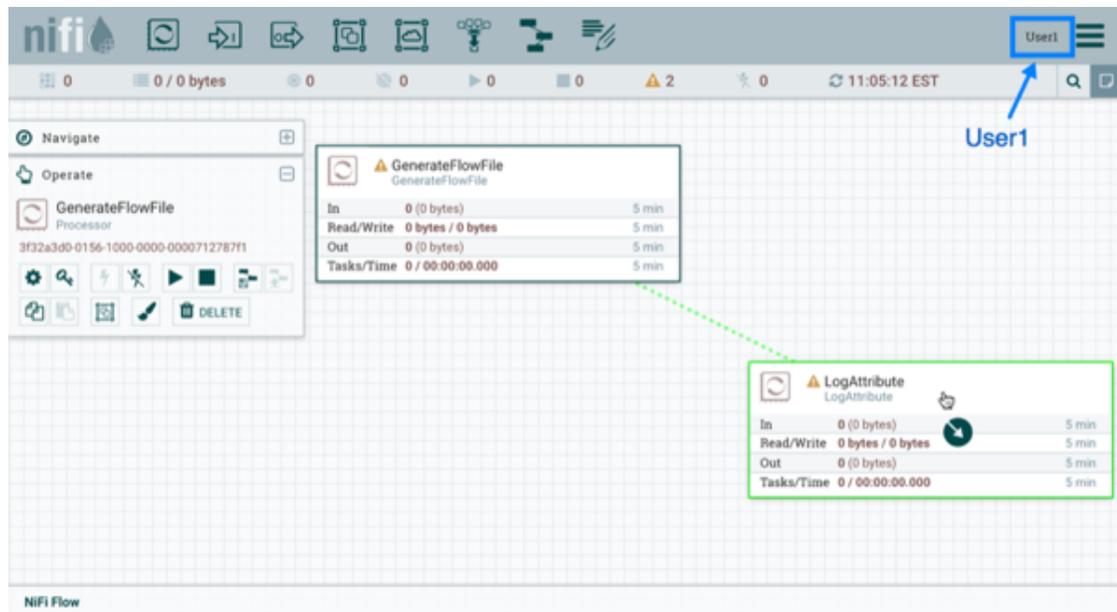
User2 cannot make the connection:
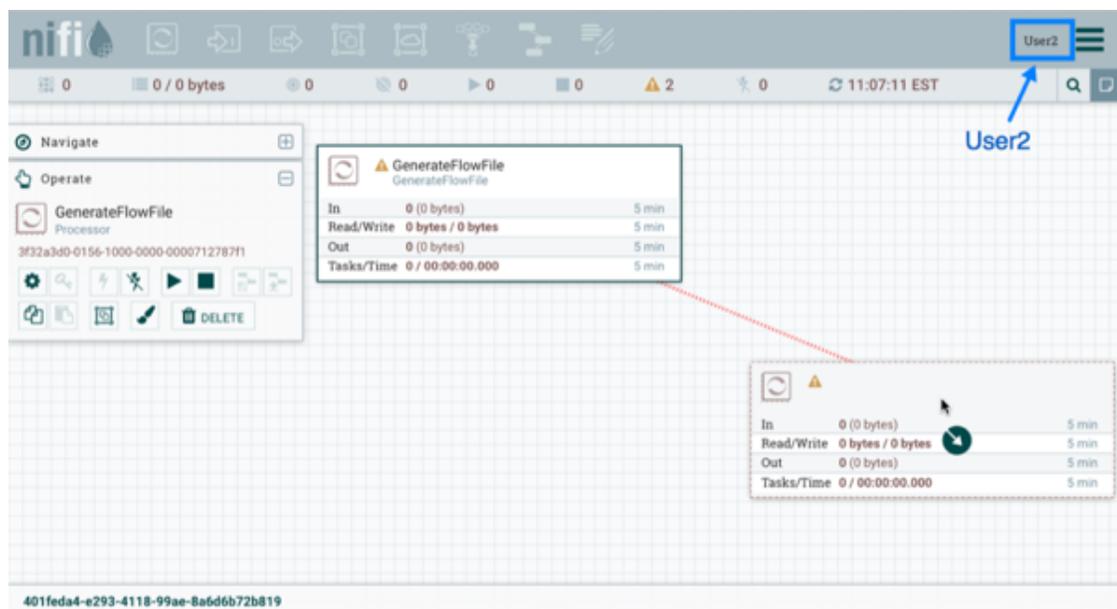
This is because:

- User2 does not have modify access on the process group.

- Even though User2 has view and modify access to the source component (GenerateFlowFile), User2 does not have an access policy on the destination component (LogAttribute).

To allow User2 to connect GenerateFlowFile to LogAttribute, as User1:

1. Select the root process group. The Operate palette is updated with details for the root process group.

2. Select the Access Policies icon

(   )
from the Operate palette and the Access Policies dialog opens.

3. Select "modify the component" from the policy drop-down.
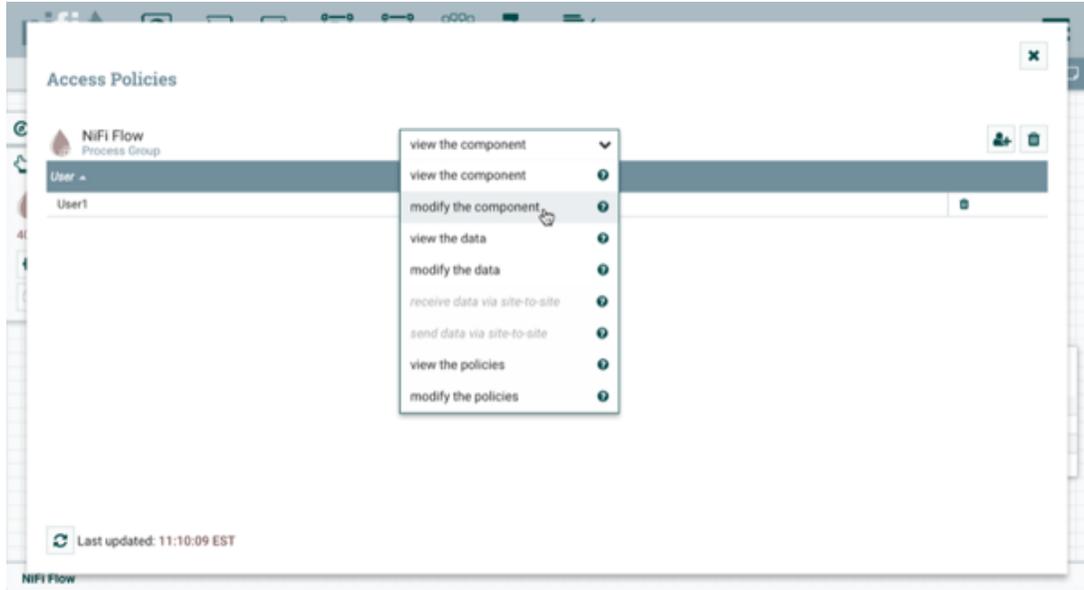
4. Select the Add User icon

(   ).
Find or enter User2 and select OK.
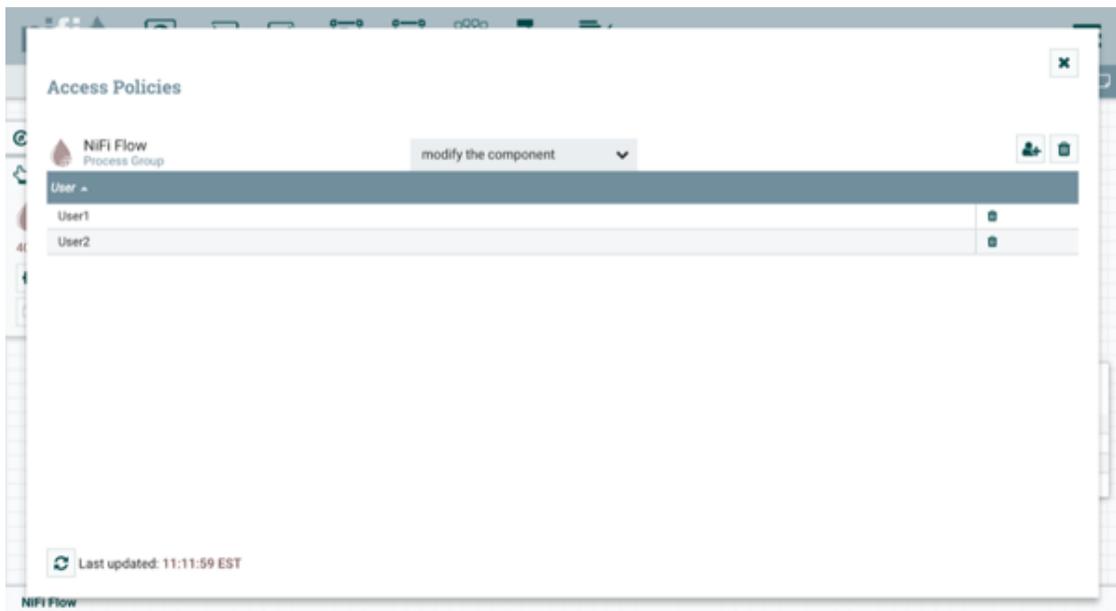
By adding User2 to the "modify the component" policy on the process
group, User2 is added to the "modify the component" policy on

the LogAttribute processor by policy inheritance. To confirm this, highlight the LogAttribute processor and select the Access Policies icon

(

)

from the Operate palette:
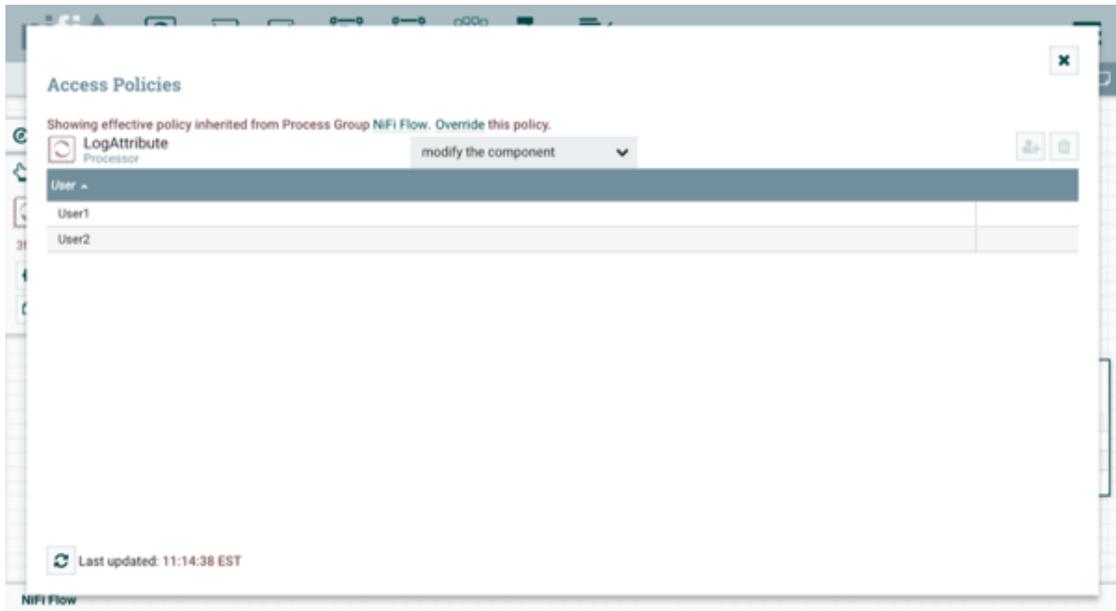
With these changes, User2 can now connect the GenerateFlowFile processor to the LogAttribute processor.
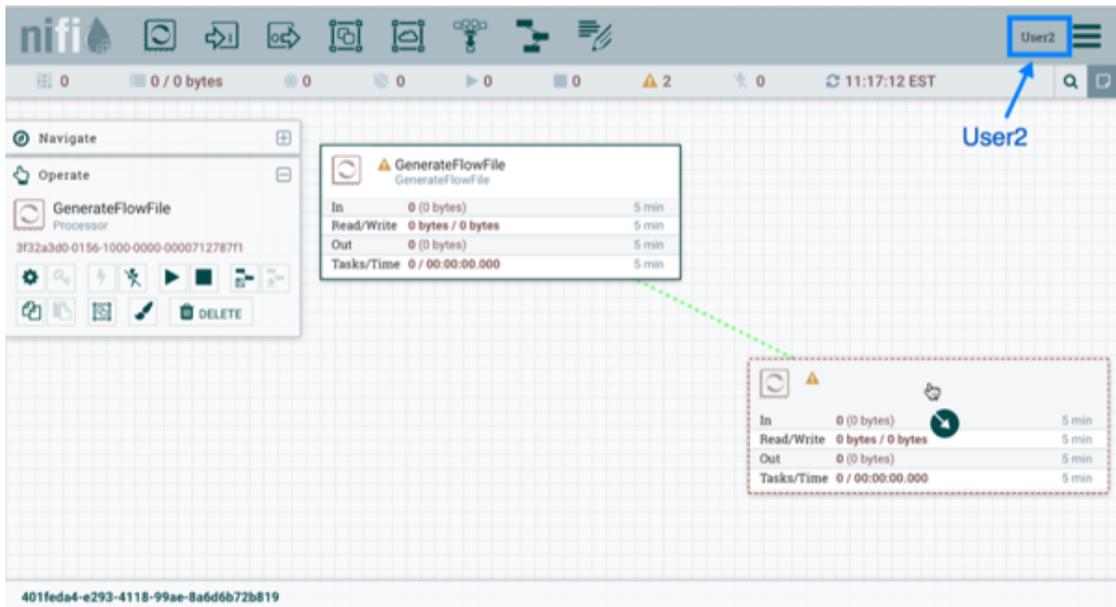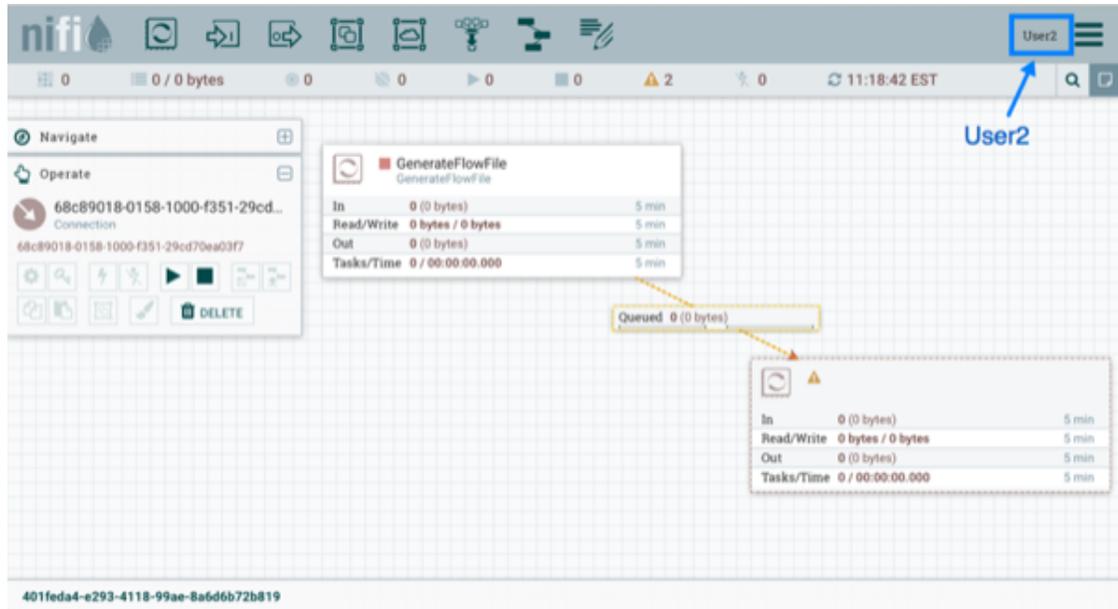
7.3.3.4. Editing a Connection

Assume User1 or User2 adds a ReplaceText processor to the root process group:
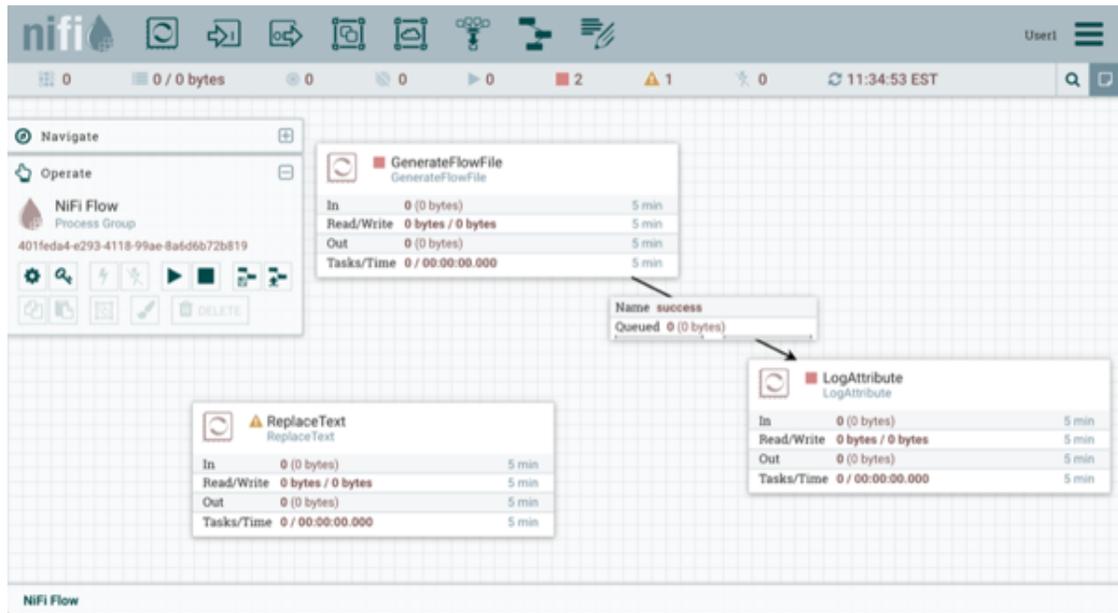
User1 can select and change the existing connection (between GenerateFlowFile to LogAttribute) to now connect GenerateFlowFile to ReplaceText:
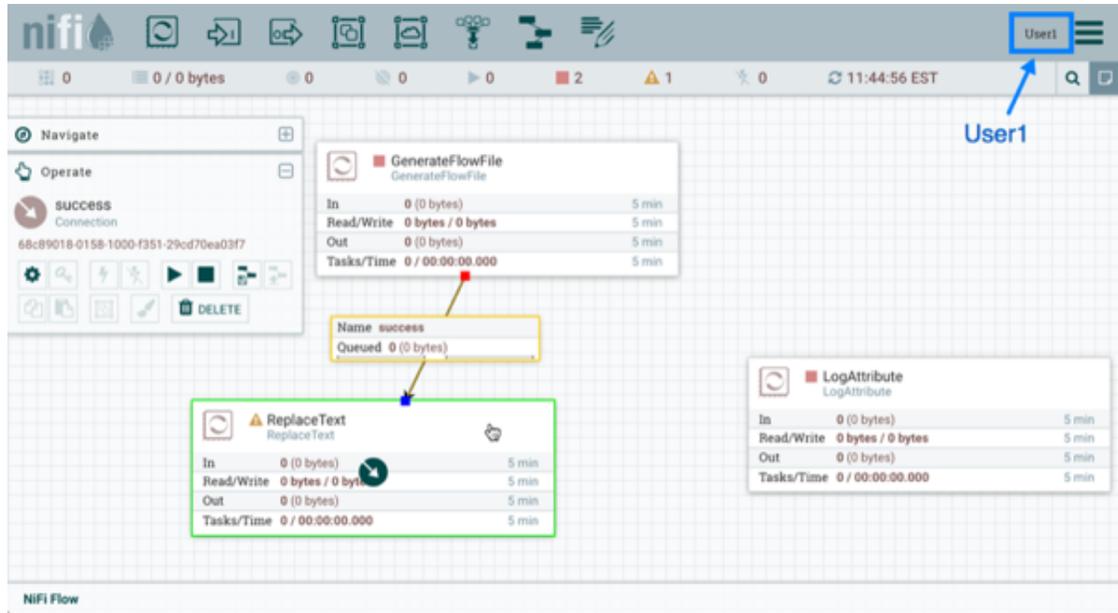
User 2 is unable to perform this action.
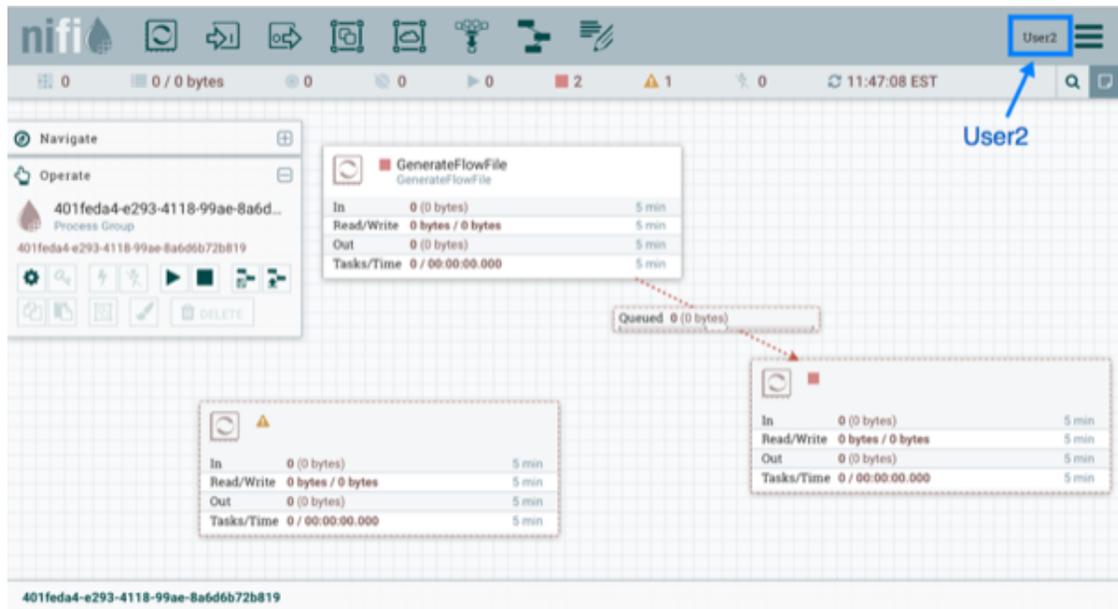
To allow User2 to connect GenerateFlowFile to ReplaceText, as User1:

1. Select the root process group. The Operate palette is updated with details for the root process group.

2. Select the Access Policies icon

   (   ).

3. Select "view the component" from the policy drop-down.

4. Select the Add User icon

().
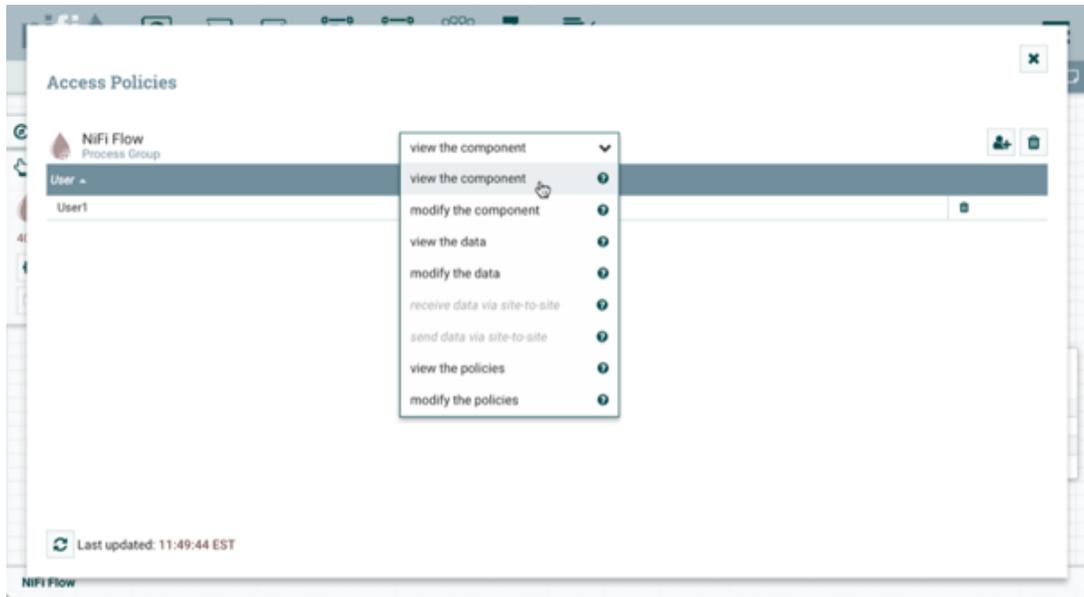
Find or enter User2 and select OK.
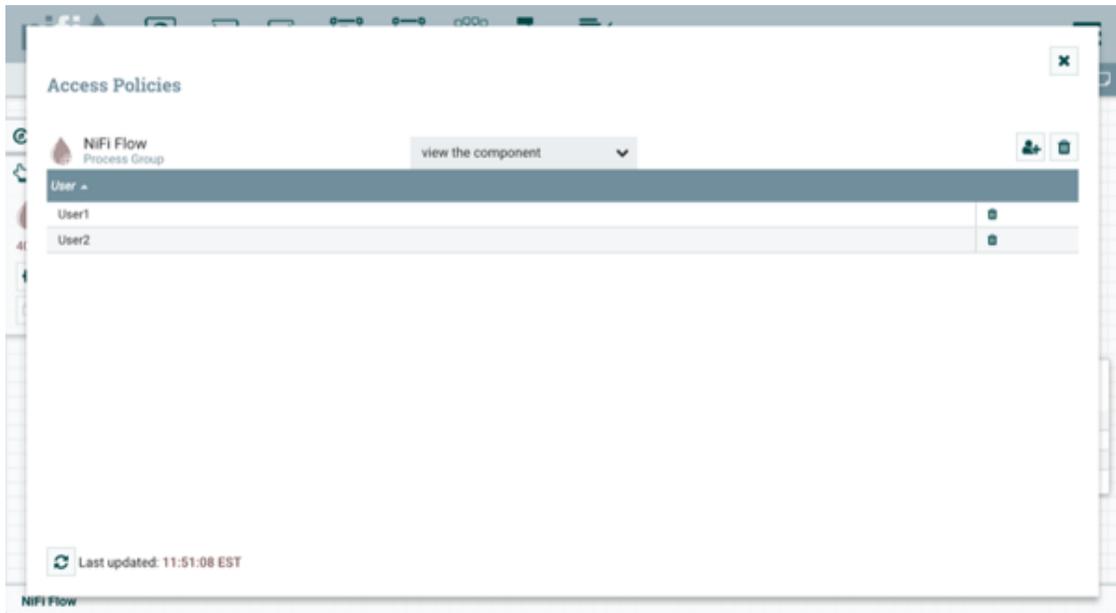
Being added to both the view and modify policies for the process group, User2 can now connect the GenerateFlowFile processor to the ReplaceText processor.

8. SAM Authorization

After you have logged in as the streamline user, you can access the SAM UI. The `streamline` user is assigned the **Admin** role and can manage users and security permissions. After logging in with this user, go to the menu item **Configuration** and select **Authorizer**.

You can use the **Authorizer** dialog to create users and assign them to roles.

8.1. Roles and Permissions

SAM provides four out of the box roles which map to the the 3 different personas that SAM provides capabilities for and then a Admin user.

• Admin Role – The Admin Role is a super user who has access to all of SAM's system roles and privileges.

• Application Developer Role – The Application Developer Role has the privileges necessary to create and submit applications.

- Operations Role – The Operations Role has the privileges necessary to create service pools and environments and to submit applications.

- Analyst Role – The Analyst Role has access to specific applications and dashboards.

A role provides permissions (Read,Write, Execute) to 5 different resources in SAM:

- Applications

- Service Pools

- Environments

- User Management / Security

- Dashboards

Table 8.1. Role and Permission Matrix

Resources	Admin Role Access	Application Developer Role Access	Operations Role Access	Analyst Role Access
Streamline Resources				
User Mgmt	All Access	No Access	No Access	No Access
Role Mgmt	All Access	No Access	No Access	No Access
Topology	All Access	U: R W X	All: R W X	No Access
Customer Processor	All Access	U: R W	All: R W	No Access
Service Pools	All Access	All: R	All: R W	No Access
Environments	All Access	U: R W	All: R W	No Access
System Artifacts: Notifier UDF UDAF Component Defs	All Access (includes edit access to component defs)	Read to All	Read to All	No Access
Custom Artifacts: Notifier UDF UDAF	All Access	U: R, W	All: R W	No Access
Dashboards				Has LInk to Menu
Schema Registry Resources				
Schemas		All: R W	All: R W	All: -
Model Registry Resources				
Models	All: R	U: R W O: R		

8.2. Creating Users and Assigning Them to Roles

Using the streamline user to initially go into SAM, you can create other admin roles in the system that can administer user accounts for the rest of the organization. To create new admin account for user gvetticaden, perform the following steps:

1. From Menu, select **Configuration** and then **Authorizer**.

2. From the **Users** tab, select the + icon.

3. Enter information about the new user account.

4. Click **Save**.

Result
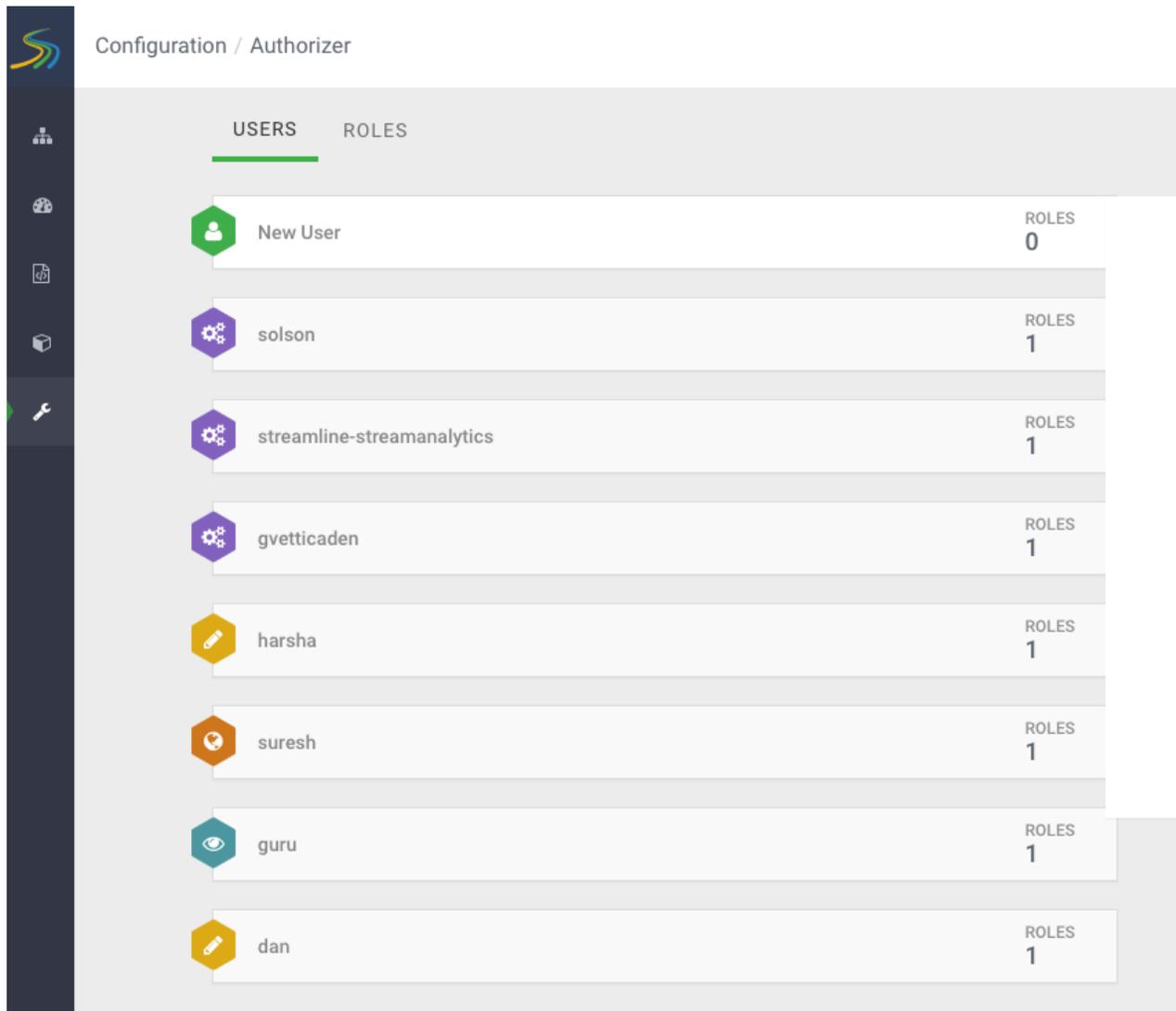
You are able to see the user you just created in the user list, on the left side of the **SAM Configuration | Authorizer** view.

You do not have to provide any password. This is because SAM relies on a Kerboeros/KDC to do the authentication. The principal is then passed to SAM when accessing the SAM UI. The principal name as part of the kerberos ticket must match a user in SAM. Then SAM looks up the role for that user and provides access based on the roles permissions.

8.3. Sharing Resources

SAM allows users to share different resources with other users to provide a collaborative team environment. A user who has edit access to a resource can share that resource with another user. When a resource is shared, the user can configure if the resource being shared can be just viewed or edited.

SAM allows the following resources to be shared:

• Environments

• Applications

8.3.1. Sharing an Environment

About This Task

By default, only the user who created an environment can see that environment. However, it is common for environments to be shared amongst a group of users. To do this, the user who created the environment must share that environment with other users.

Steps

1. From the left-hand menu select **Configuration**, then **Environment**.

2. From the environment you want to share, click the **Configuration** elipses at the top right and click **Share**.

3. In the **Share Environment** dialog, select the users with whom you want to share the environment.

4. Specify whether you want to give them **View** or **Edit** privileges and click **Ok**.

8.4. Sharing an Application

About This Task

By default, only the user who created an application can see that it. However, it is common for applications to be shared amongst a group of users. To do this, the user who created the application must share that it with other users.

Steps

1. From the left-hand menu select **My Applications**.

2. From the application you want to share, click the **Configuration** elipses at the top right and click **Share**.

3. In the **Share Application** dialog, select the users with whom you want to share the application.

4. Specify whether you want to give them **View** or **Edit** privileges and click **Ok**.

8.5. SAM Authorization Limitations

- SAM's roles and access control policies are maintained in SAM. Ranger support for SAM is not availble in HDF 3.0.

- Creation of users and assignment to roles must be done using the SAM UI. In the HDF 3.0 release, there is no support to import users from KDC/AD.

- Role assignment is at a user level. Assigning roles to a group is not supported in HDF 3.0 release.

- New Roles or editing the out of the box role cannot be allows. However, the collaboration sharing features allow you to share each of the 5 resources across users meeting most use case requirements.

9. 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.

9.1. Connecting to a Secure Service that Supports Delegation Tokens

About This Task

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 connecing 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 and Keytab path. SAM automatically populates all the principal
   and key tabs located on the Nimbus Host to make this easier. Then click **Ok**.
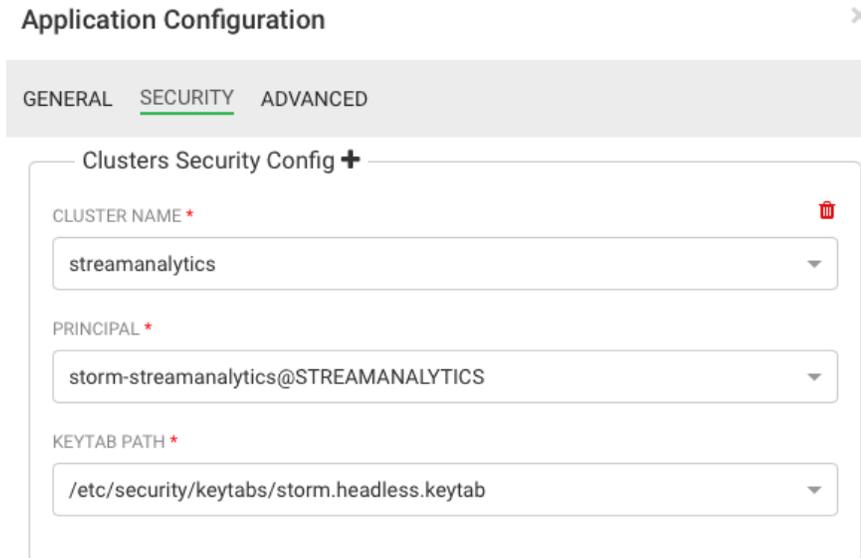
Result

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.

9.2. Connecting to Secure Kafka

About This Task

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.

Steps

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".

9.3. 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 kerborized.

- 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.

9.3.1. Understanding the End-to-End Workflow

The below image provides an explanation on how SAM functions for the above use case.
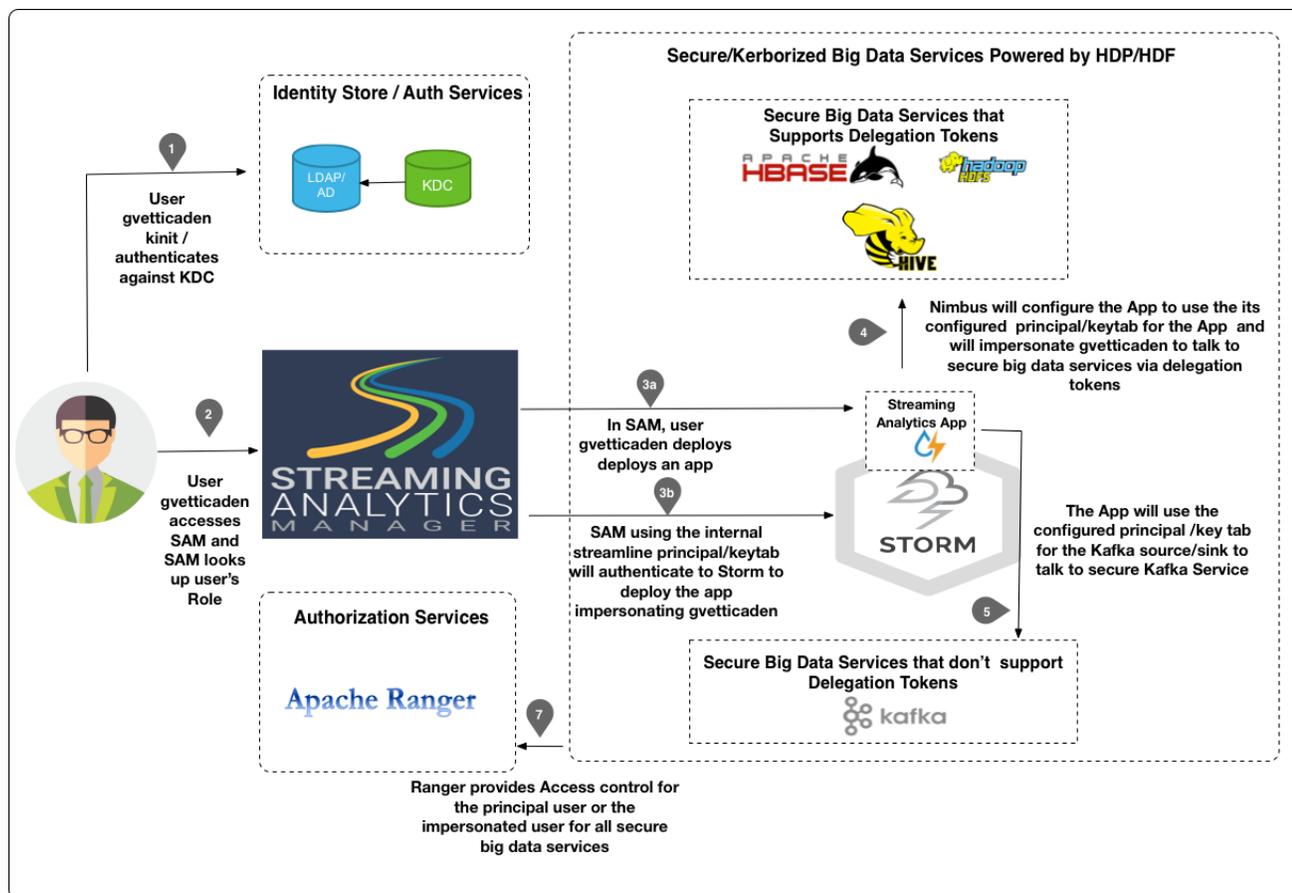
Step 1: Initial Login

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 usingthe streamline principal/keyab. 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.