Configuring Apache Knox SSO

Date of Publish: 2018-07-15

http://docs.hortonworks.com

Contents

Setting Up Knox SSO

Knox SSO provides web UI SSO (Single Sign-on) capabilities to your cluster. Knox SSO enables your users to login once and gain access to cluster resources. To set up Knox SSO, you will configure an identity provider, enable SSO using the Ambari CLI, and then manually configure various component settings.

Context

The flexibility of the Apache Knox authentication and federation providers allows KnoxSSO to provide normalization of authentication events through token exchange, resulting in a common JWT (JSON WebToken)-based token.

KnoxSSO provides an abstraction for integrating any number of authentication systems and SSO solutions, and enables participating web applications to scale to those solutions more easily. Without the token exchange capabilities offered by KnoxSSO, each component UI would need to integrate with each desired solution on its own. With KnoxSSO, they only need to integrate with the single solution and common token.

Configuring Knox SSO Workflow Overview

There are two ways to set up Knox SSO:

- LDAP/AD: Uses the default form-based identity provider, Shiro. Use this if you have an Active Directory deployment.
- SAML: Uses the pac4j provider and integrates with the identity provider Okta. Use this if you integrate with an external identity provider.

To set up Knox SSO with LDAP/AD, complete the following workflow:

1. Install Knox.
2. Configure Ambari Authentication for LDAP/AD.
3. Configure an LDAP/AD Identity Provider (IdP).
4. Enable Knox SSO using the Ambari CLI.
5. Configure Knox SSO for HDFS, Oozie, MapReduce2, Zeppelin, or YARN.
6. Restart all services that require a restart via Ambari.

To set up Knox SSO with SAML/Okta, complete the following workflow:

1. Install Knox.
2. Configure an Okta Identity Provider (IdP).
3. Set up Knox SSO via the Ambari CLI.
4. Set up Knox SSO via Component Config Files.
5. Restart all services that require a restart via Ambari.

For information on what services are supported for Knox SSO, see the “Knox Supported Services Matrix”.

Related Information

Knox Supported Services Matrix

Configuring an Identity Provider (IdP)

Knox has two identity providers: form-based (for LDAP/AD) and SAML (for Okta).

Configuring an LDAP/AD Identity Provider (IdP)

The form-based identity provider (IdP) is the default identity provider for KnoxSSO out of the box and is installed by Ambari. This form-based IdP is used for LDAP/AD.

Pre-requisites

LDAP authentication must be configured for Ambari and that it be the same LDAP server as Knox SSO is using for form-based IdP. See “Configuring Ambari Authentication for LDAP/AD”.

Reference

The installed configuration of the provider leverages the Shiro provider which attempts to authenticate a request by looking for HTTP basic credentials.

Instead of responding with an HTTP basic challenge, however, the browser is redirected to the KnoxAuth application to present the user with a form in which to submit username and password.

knoxsso.xml with Shiro provider

The following knosso.xml topology file illustrates the use of the Shiro provider, the hosting of the knoxauth application, and the configuration of the KNOXSSO service itself.

The typical Shiro provider configuration is augmented with new parameters for achieving the behavior described above.

The restrictedCookies parameter is used to add the WWW-Authenticate header in order to suppress the HTTP basic challenge.

The redirectToUrl parameter is used to indicate where to redirect the browser rather, than issuing an HTTP basic challenge.

Note, also, the application element which is used to indicate that a given application is to be hosted by the Knox gateway and how it relates to the redirectToUrl parameter in the Shiro provider.

The knoxsso.xml topology describes the manner in which a client acquires a KnoxSSO websso cookie/token. The Shiro provider allows the integration LDAP/AD with HTTP Basic Auth credentials.

```
<topology>
    <gateway>
      <provider>
        <role>webappsec</role>
        <name>WebAppSec</name>
        <enabled>true</enabled>
        <param>
            <name>xframe.options.enabled</name>
            <value>true</value>
        </param>
      </provider>
        <provider>
            <role>authentication</role>
            <name>ShiroProvider</name>
            <enabled>true</enabled>
            <param>
                <name>sessionTimeout</name>
                <value>30</value>
            </param>
            <param>
                <name>redirectToUrl</name>
                <value>/gateway/knoxsso/knoxauth/login.html</value>
            </param>
            <param>
                <name>restrictedCookies</name>
                <value>rememberme,WWW-Authenticate</value>
            </param>
            <param>
                <name>main.ldapRealm</name>
                <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm</
value>
            </param>
            <param>
                <name>main.ldapContextFactory</name>

 <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapContextFactory</value>
            </param>
            <param>
                <name>main.ldapRealm.contextFactory</name>
                <value>$ldapContextFactory</value>
            </param>
            <param>
                <name>main.ldapRealm.userDnTemplate</name>
                <value>uid={0},ou=people,dc=hadoop,dc=apache,dc=org</value>
            </param>
            <param>
                <name>main.ldapRealm.contextFactory.url</name>
                <value>ldap://localhost:33389</value>
            </param>
            <param>
                <name>main.ldapRealm.authenticationCachingEnabled</name>
                <value>false</value>
            </param>
            <param>

 <name>main.ldapRealm.contextFactory.authenticationMechanism</name>
                <value>simple</value>
            </param>
            <param>
                <name>urls./**</name>
                <value>authcBasic</value>
            </param>
        </provider>
        <provider>
            <role>identity-assertion</role>
            <name>Default</name>
            <enabled>true</enabled>
        </provider>
        <provider>
            <role>hostmap</role>
            <name>static</name>
            <enabled>true</enabled>
            <param><name>localhost</
name><value>sandbox,sandbox.hortonworks.com</value></param>
        </provider>
    </gateway>

    <application>
      <name>knoxauth</name>
    </application>

    <service>
        <role>KNOXSSO</role>
        <param>
            <name>knoxsso.cookie.secure.only</name>
            <value>true</value>
        </param>
        <param>
            <name>knoxsso.token.ttl</name>
```

```
             <value>30000</value>
         </param>
         <param>
            <name>knoxsso.redirect.whitelist.regex</name>
            <value>^https?:\/\/(c64\d\d\.ambari\.apache\.org|localhost|
127\.0\.0\.1|0:0:0:0:0:0:0:1|::1):[0-9].*$</value>
         </param>
     </service>
</topology>
```

The following parameters must be updated to match your environment:

- redirectToUrl
- main.ldapRealm.userDnTemplate
- main.ldapRealm.contextFactory.url
    - You can point this to the demo LDAP server (ldap://localhost:33389) or your own LDAP server.
- knoxsso.redirect.whitelist.regex

Related Information

Configuring Ambari Authentication for LDAP/AD

Configuring an Okta Identity Provider (IdP)

Apache Knox with KnoxSSO + pac4j provider enables the use of a number of new authentication and SSO solutions for accessing and developing KnoxSSO-enabled applications (including Ambari, Ranger, service UIs and custom built applications that utilize REST APIs through Knox.) This section illustrates the integration of the Okta identity service offering by leveraging the pac4j provider SAML capabilities in Apache Knox.

Reference

A similar flow to what is described below would be available for Ambari and Ranger, or any KnoxSSO participating application.

As opposed to the KnoxSSO form-based IdP, where the actual form is hosted by a Knox hosted authentication app, SAML IdPs need to have KnoxSSO instances configured within the SAML solution as participating in the SAML SSO flow as a service provider. This generally requires the creation of an “Application” in Okta and other providers which will contain the required endpoints and metadata required for verifying requests for login and redirecting users back to KnoxSSO after the authentication event is successful.

knoxsso.xml with Okta

The knoxsso.xml topology file will need to be changed from the form-based IdP configuration to the SAML-based IdP by swapping the Shiro provider with the pac4j provider for Knox.

The knoxsso.xml topology describes the manner in which a client acquires a KnoxSSO websso cookie/token. The pac4j federation provider allows the integration of a number of authentication solutions. In this case, the openid connect capability is being leveraged to integrate the cloud-based PrivaKey identity service.

The following topology file is an example for use with Okta.

```
<topology>
   <gateway>
     <provider>
         <role>federation</role>
         <name>pac4j</name>
         <enabled>true</enabled>
         <param>
           <name>pac4j.callbackUrl</name>
      <value>https://www.local.com:8443/gateway/knoxsso/api/v1/websso</
value>
```

```
         </param>

         <param>
           <name>clientName</name>
           <value>SAML2Client</value>
         </param>

         <param>
           <name>saml.identityProviderMetadataPath</name>
           <value>https://dev-122415.oktapreview.com/app/
exk5nc5z1xbFKb7nH0h7/sso/saml/metadata</value>
         </param>

         <param>

           <name>saml.serviceProviderMetadataPath</name>
           <value>/tmp/sp-metadata.xml</value>
         </param>

         <param>
           <name>saml.serviceProviderEntityId</name>
           <value>https://www.local.com:8443/gateway/knoxsso/api/v1/websso?
pac4jCallback=true&amp;client_name=SAML2Client</value>
         </param>
     </provider>
     <provider>
         <role>identity-assertion</role>
         <name>Default</name>
         <enabled>true</enabled>
         <param>
           <name>principal.mapping</name>
           <value>guest@example.com=guest;</value>
         </param>
     </provider>
   </gateway>

   <service>
       <role>KNOXSSO</role>
       <param>
         <name>knoxsso.cookie.secure.only</name>
         <value>true</value>
      </param>
      <param>
        <name>knoxsso.token.ttl</name>
        <value>100000</value>
      </param>
      <param>
         <name>knoxsso.redirect.whitelist.regex</name>
         <value>^https?:\/\/(www\.local\.com|localhost|127\.0\.0\.1|
0:0:0:0:0:0:0:1|::1):[0-9].*$</value>
      </param>
   </service>
</topology>
```

In the above example, we have configured the Okta application to assert the user's ID as their email address. In order to leverage this as the identity, we need to map it to a username that will be valid within the cluster. The identity assertion provider above does a simple mapping from a known email address to a known username. More appropriate assertion provider usage would likely be to use the regex assertion provider that would allow you to extract the username from the email address.

We currently do not have support for establishing groups from the SAML assertion and must rely on the participating applications to do a group lookup based on the username.

Related Information

Okta information on SAML-based SSO

Configuring the SAML application using the Okta SAML App Wizard

Enable Knox SSO Using the Ambari CLI

To enable Knox SSO (Single Sign-on) for your cluster, you must begin with the Ambari CLI wizard. It prompts you for SSO settings and propagates them across your cluster. The wizard then configures SSO for Atlas, Ambari, and Ranger UIs.

About this task

When you enable SSO, unauthenticated users who try to access a service (e.g., Ambari, Atlas, etc), are redirected to the Knox SSO login page for authentication. This makes signing into services faster and easier, with fewer credentials to remember.

Before you begin

The Ambari Server must be running and you must be logged in as root.

Procedure

1. From the command line, begin the SSO setup wizard: ambari-server setup-sso.
2. When prompted, enter your Ambari Admin credentials.
3. Depending on your configuration, choose a path:
   - If SSO is not configured, it prompts Do you want to configure SSO authentication.
     - Enter y to continue through the wizard.
     - Enter n to exit the wizard.
   - If SSO is already configured, it prompts Do you want to disable SSO authentication.
     - Enter y to disable SSO for Ambari and the services (if services were being managed). Then it exits the wizard.
     - Enter n to continue through the wizard.
4. Enter the provider URL using the format: https://<hostname>:8443/gateway/knoxsso/api/v1/websso. https://dw-weekly.field.hortonworks.com:8443/gateway/knoxsso/api/v1/websso
5. Populate the Public Certificate PEM:

   a) Export the Knox certificate: ./knoxcli.sh export-cert --type PEM

   ```
   [root@dw-weekly bin]# ./knoxcli.sh export-cert --type PEM
   Certificate gateway-identity has been successfully exported to: /usr/
   $REPO/$VERSION/knox/data/security/keystores/gateway-identity.pem
   ```

   b) Copy the contents of the file, excluding the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----.

   ```
   [root@dw-weekly bin]# ./knoxcli.sh export-cert --type PEM
   Certificate gateway-identity has been successfully exported to: /usr/
   $REPO/$VERSION/knox/data/security/keystores/gateway-identity.pem
   [root@dw-weekly bin]# vi /usr/$REPO/$VERSION/knox/data/security/
   keystores/gateway-identity.pem

   -----BEGIN CERTIFICATE-----
   MIIoqToofo6gfwIffgIIfJG6+oql7YUwGQYJKoqIhvoNfQZFfQfwGTZLMfkGf1UZfhMoVVMxGTfL
   fgNVffgTfFRlo3QxGTfLfgNVffoTfFRlo3QxGqfNfgNVffoTfkhhqG9voGZNMfsGf1UZoxMZVGVq
   GGZoMoYGf1UZfxMfqHotG2Vlf2x5LmqpqWxkLmhvonRvfnGvomtqLmNvfTfZFw0xOGf2MTUxNjU0
   ```

```
MjffFw0xOTf2MTUxNjU0MjffMHUxoqfJfgNVffYTflVTMQ0wowYGVQQIZwRUqXN0MQ0wowYGVQQH
ZwRUqXN0MQ8wGQYGVQQKZwqIYWRvf3fxGTfLfgNVffsTfFRlo3QxKGfmfgNVffMTH2R3LXGlqWts
ZS5mfWVsqo5of3J0f253f3Jroy5jf20wgq8wGQYJKoqIhvoNfQZffQfGgY0fMIGJfoGffMjs9Q6M
f4f4Ussf/Yffpfr7k3Gx8v0/
Vlum6OL3Mr0vYQFtNSvGMZTZ25QQ8YHOvGf4frqi9lqwj6qwZYWf
RQUTIxuiOGPiMhK70onmLflmqpoGYmSJ3/shfOUoyN7+JiImYYn/
rJvt4Yt362gGvJynfsZGGKko
johF4v0FLoqGfgMfffZwGQYJKoqIhvoNfQZFfQfGgYZfZomm8ZTJJufW4vfp8O51Qx7J4ioY6G69
qgf76j4Oh8fqGqRVfoKYvrIZuJsZKHpIPGhtnVtqHG8YYf6vffSXoMmGpp5qfvZLfqnR1HNl6oZq
qf7J9qn9MPZqlrf5/kOGY85w0UUkVqotRLjsK/niHhojGKffJrok7hMUo7TYwfQ
-----END CERTIFICATE-----
```

c) When prompted Public Certificate PEM (empty line to finish input), paste the contents of the cert.pem file.

```
MIIoqToofo6gfwIffgIIfJG6+oql7YUwGQYJKoqIhvoNfQZFfQfwGTZLMfkGf1UZfhMoVVMxGTfL
fgNVffgTfFRlo3QxGTfLfgNVffoTfFRlo3QxGqfNfgNVffoTfkhhqG9voGZNMfsGf1UZoxMZVGVq
GGZoMoYGf1UZfxMfqHotG2Vlf2x5LmqpqWxkLmhvonRvfnGvomtqLmNvfTfZFw0xOGf2MTUxNjU0
MjffFw0xOTf2MTUxNjU0MjffMHUxoqfJfgNVffYTflVTMQ0wowYGVQQIZwRUqXN0MQ0wowYGVQQH
ZwRUqXN0MQ8wGQYGVQQKZwqIYWRvf3fxGTfLfgNVffsTfFRlo3QxKGfmfgNVffMTH2R3LXGlqWts
ZS5mfWVsqo5of3J0f253f3Jroy5jf20wgq8wGQYJKoqIhvoNfQZffQfGgY0fMIGJfoGffMjs9Q6M
f4f4Ussf/Yffpfr7k3Gx8v0/
Vlum6OL3Mr0vYQFtNSvGMZTZ25QQ8YHOvGf4frqi9lqwj6qwZYWf
RQUTIxuiOGPiMhK70onmLflmqpoGYmSJ3/shfOUoyN7+JiImYYn/
rJvt4Yt362gGvJynfsZGGKko
johF4v0FLoqGfgMfffZwGQYJKoqIhvoNfQZFfQfGgYZfZomm8ZTJJufW4vfp8O51Qx7J4ioY6G69
qgf76j4Oh8fqGqRVfoKYvrIZuJsZKHpIPGhtnVtqHG8YYf6vffSXoMmGpp5qfvZLfqnR1HNl6oZq
qf7J9qn9MPZqlrf5/kOGY85w0UUkVqotRLjsK/niHhojGKffJrok7hMUo7TYwfQ
```

6. When prompted Use SSO for Ambari [y/n] (n)?, enter Y to use or N to not use SSO for Ambari.

   Ambari does not need to be configured for SSO in order for the services to be configured for SSO (and vice-versa).

7. When prompted Manage SSO configurations for eligible services [y/n] (n)?, enter your selection.

   - y begins the service SSO setup wizard.
   - n exits the SSO setup wizard, saving your PEM setup and Ambari SSO selections.

   If you choose Y, the configurations for each eligible service are changed depending on the your selection when prompted.

   If you choose N, Ambari does not alter the existing configuration for any service. This is important if the cluster was set up using Blueprints and you do not want Ambari to change the SSO settings explicitly set.

8. If you chose y, you are prompted Use SSO for all services [y/n] (y)?.

   - y automatically sets up SSO for all available services.
   - n enters SSO set up for each individual service, allowing you to choose for which services you wish to enable SSO.

9. For the JWT Cookie name (), hadoop-jwt is the default.
10. Leave JWT audiences list empty.
    The prompt returns Ambari Server 'setup-sso' completed successfully.
11. Select **Ambari** > **Actions** > **Restart All Required** to restart all other services that require a restart.

Example

Example Knox SSO via ambari-server setup-sso

```
[root@dw-weekly ~]# $JAVA_HOME/bin/keytool -export -alias gateway-identity
 -rfc -file cert.pem -keystore /usr/$REPO/current/knox-server/data/security/
keystores/gateway.jks
[root@dw-weekly ~]# cd /usr/$REPO/current/knox-server/bin
[root@dw-weekly bin]# ./knoxcli.sh export-cert --type PEM
Certificate gateway-identity has been successfully exported to: /usr/$REPO/
$VERSION/knox/data/security/keystores/gateway-identity.pem
```

```
[root@dw-weekly bin]# vi /usr/$REPO/$VERSION/knox/data/security/keystores/
gateway-identity.pem
// <copy the certificate>
```

```
[root@dw-weekly ~]# ambari-server setup-sso
Using python  /usr/bin/python
Setting up SSO authentication properties...
Enter Ambari Admin login: admin
Enter Ambari Admin password:

SSO is currently not configured
Do you want to configure SSO authentication [y/n] (y)? y
Provider URL (https://knox.example.com:8443/gateway/knoxsso/api/v1/websso):
 https://dw-weekly.field.hortonworks.com:8443/gateway/knoxsso/api/v1/websso
Public Certificate PEM (empty line to finish input):
MIIoqToofo6gfwIffgIIfJG6+oql7YUwGQYJKoqIhvoNfQZFfQfwGTZLMfkGf1UZfhMoVVMxGTfL
fgNVffgTfFRlo3QxGTfLfgNVffoTfFRlo3QxGqfNfgNVffoTfkhhqG9voGZNMfsGf1UZoxMZVGVq
GGZoMoYGf1UZfxMfqHotG2Vlf2x5LmqpqWxkLmhvonRvfnGvomtqLmNvfTfZFw0xOGf2MTUxNjU0
MjffFw0xOTf2MTUxNjU0MjffMHUxoqfJfgNVffYTflVTMQ0wowYGVQQIZwRUqXN0MQ0wowYGVQQH
ZwRUqXN0MQ8wGQYGVQQKZwqIYWRvf3fxGTfLfgNVffsTfFRlo3QxKGfmfgNVffMTH2R3LXGlqWts
ZS5mfWVsqo5of3J0f253f3Jroy5jf20wgq8wGQYJKoqIhvoNfQZffQfGgY0fMIGJfoGffMjs9Q6M
f4f4Ussf/Yffpfr7k3Gx8v0/Vlum6OL3Mr0vYQFtNSvGMZTZ25QQ8YHOvGf4frqi9lqwj6qwZYWf
RQUTIxuiOGPiMhK70onmLflmqpoGYmSJ3/shfOUoyN7+JiImYYn/rJvt4Yt362gGvJynfsZGGKko
johF4v0FLoqGfgMfffZwGQYJKoqIhvoNfQZFfQfGgYZfZomm8ZTJJufW4vfp8O51Qx7J4ioY6G69
qgf76j4Oh8fqGqRVfoKYvrIZuJsZKHpIPGhtnVtqHG8YYf6vffSXoMmGpp5qfvZLfqnR1HNl6oZq
qf7J9qn9MPZqlrf5/kOGY85w0UUkVqotRLjsK/niHhojGKffJrok7hMUo7TYwfQ=

Use SSO for Ambari [y/n] (n)? y
Manage SSO configurations for eligible services [y/n] (n)? y
 Use SSO for all services [y/n] (n)? y
JWT Cookie name (hadoop-jwt): hadoop-jwt
JWT audiences list (comma-separated), empty for any ():
Ambari Server 'setup-sso' completed successfully.
```

```
[root@dw-weekly ~]# ambari-server restart
```

What to do next

You must next manually configure Knox SSO by using component configuration files. These steps are documented in “Set up Knox SSO via Component Config Files”.

Related Information

Configure Knox SSO for HDFS, Oozie, MapReduce2, Zeppelin, or YARN

Ambari CLI Wizard for Knox SSO Reference

Configure Knox SSO for HDFS, Oozie, MapReduce2, Zeppelin, or YARN

As of HDP-3.0.0, SSO is enabled using the ambari-server setup-sso wizard. SSO for Ambari, Atlas, and Ranger is automatically enabled by the wizard. To enable SSO for HDFS, Oozie, MapReduce2, Zeppelin, or YARN, you must manually change their configuration files. Users who try to access these components will be redirected to the Knox SSO login page for authentication.

Before you begin

You must be running Ambari 2.7.0.0 with HDP-3.0.0 or higher.

You must have already enabled SSO using ambari-server setup-sso.

Procedure

1. In Ambari, set the following properties for your components:

- HDFS: core-site.xml

```
"hadoop.http.authentication.type":
 "org.apache.hadoop.security.authentication.server.JWTRedirectAuthenticationHandler
"hadoop.http.authentication.public.key.pem": “$SSOPUBLICKEY"
"hadoop.http.authentication.authentication.provider.url":
 "$SSOPROVIDERURL"
```

- Oozie: oozie-site.xml

```
oozie.authentication.type=org.apache.hadoop.security.authentication.server.JWTRedir
oozie.authentication.authentication.provider.url=https://
$KNOX_HOST:8443/gateway/knoxsso/api/v1/websso
oozie.authentication.public.key.pem=$KNOX_PUBLIC_KEY
optional: oozie.authentication.expected.jwt.audiences=$AUDIENCES
 (default: EMPTY; which means ALL)
optional: oozie.authentication.jwt.cookie=$COOKIE-NAME (default: hadoop-
jwt)
```

- MapReduce2: core-site.xml

```
"hadoop.http.authentication.type":
 "org.apache.hadoop.security.authentication.server.JWTRedirectAuthenticationHandler
"hadoop.http.authentication.public.key.pem": “$SSOPUBLICKEY"
"hadoop.http.authentication.authentication.provider.url":
 "$SSOPROVIDERURL"
```

- Zeppelin: Advanced zeppelin-shiro-ini > shiro_ini_content

```
knoxJwtRealm = org.apache.zeppelin.realm.jwt.KnoxJwtRealm
knoxJwtRealm.providerUrl = $PROVIDERURL
knoxJwtRealm.login = gateway/knoxsso/knoxauth/login.html
knoxJwtRealm.publicKeyPath = $PATH_OF_KNOX-SSO.PEM
knoxJwtRealm.logoutAPI = false
knoxJwtRealm.logout = gateway/knoxssout/api/v1/webssout
knoxJwtRealm.cookieName = hadoop-jwt
knoxJwtRealm.redirectParam = originalUrl
knoxJwtRealm.groupPrincipalMapping = group.principal.mapping
knoxJwtRealm.principalMapping = principal.mapping
authc = org.apache.zeppelin.realm.jwt.KnoxAuthenticationFilter
```

- Zeppelin: Advanced spark2-env, for SPARK_HISTORY_OPTS

```
export SPARK_HISTORY_OPTS=’
-
Dspark.ui.filters=org.apache.hadoop.security.authentication.server.AuthenticationFi
-
Dspark.org.apache.hadoop.security.authentication.server.AuthenticationFilter.params
 ="type=org.apache.hadoop.security.authentication.server.JWTRedirectAuthenticationH
kerberos.principal=$SPARK_HISTORY_KERBEROS_PRINCIPAL,
kerberos.keytab=$SPNEGO_KEYTAB,
authentication.provider.url=$PROVIDER_URL ,
public.key.pem=$PUBLIC_KEY”’
```

- YARN: core-site.xml

```
"hadoop.http.authentication.type":
 "org.apache.hadoop.security.authentication.server.JWTRedirectAuthenticationHandler
"hadoop.http.authentication.public.key.pem": “$SSOPUBLICKEY"
```

```
"hadoop.http.authentication.authentication.provider.url":
 "$SSOPROVIDERURL"
```

2. Click Save and confirm subsequent prompts.
3. Click **Ambari** > **Actions** > **Restart All Required** to restart all other services that require a restart.