Red Hat Single Sign-On 7.6

Red Hat Single Sign-On for OpenShift

Red Hat Single Sign-On 7.6 向け

Last Updated: 2022-07-20

Red Hat Single Sign-On 7.6 Red Hat Single Sign-On for OpenShift

Red Hat Single Sign-On 7.6 向け

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本ガイドは、Red Hat Single Sign-On 7.6 for OpenShift を使い始めるための基本的な情報および手順で構成されています。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。これは大規模な取り組みであるため、これらの変更は今後の複数のリリースで段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 OPENSHIFT の RED HAT SINGLE SIGN-ON の概要

1.1. RED HAT SINGLE SIGN-ON とは?

Red Hat Single Sign-On は、Red Hat JBoss Middleware for OpenShift のコンテナー化イメージとして利用できる統合されたシングルサインオンソリューションです。Red Hat Single Sign-On for OpenShift イメージは、ユーザーが Web アプリケーション、モバイルアプリケーション、RESTful Web サービスのユーザーアカウントを一元的にログイン、ログアウト、登録、および管理するための認証サーバーを提供します。

OpenShift の Red Hat Single Sign-On は、x86_64、IBM Z、および IBM Power Systems のプラットフォームで使用できます。

1.2. 比較： RED HAT SINGLE SIGN-ON FOR OPENSHIFT IMAGE と RED HAT SINGLE SIGN-ON

Red Hat Single Sign-On for OpenShift イメージバージョン番号 7.6.0 は Red Hat Single Sign-On 7.6.0 をベースにしています。OpenShift イメージと Red Hat Single Sign-On の Red Hat Single Sign-On には、考慮する必要のある機能にはいくつかの重要な違いがあります。

OpenShift イメージの Red Hat Single Sign-On には、Red Hat Single Sign-On のすべての機能が含まれています。さらに、Red Hat Single Sign-On 対応の JBoss EAP イメージは、各 **web.xml** ファイルに **<auth-method>KEYCLOAK</auth-method>** または **<auth-method>KEYCLOAK-SAML</auth-method>** を含む **.war** に対する OpenID Connect または SAML クライアント登録を自動的に処理します。

1.3. このソフトウェアで使用するためのテンプレート

Red Hat は、Red Hat Single Sign-On for OpenShift イメージバージョン番号 7.6.0 を使用して複数の OpenShift アプリケーションテンプレートを提供します。これらのテンプレートは、Red Hat Single Sign-On 7.6.0 サーバーベースのデプロイメントの開発に必要なリソースを定義します。テンプレートは、主にパススルーテンプレートと再暗号化テンプレートの 2 つのカテゴリに分けることができます。その他のテンプレートもいくつか存在します。

1.3.1. パススルーテンプレート

これらのテンプレートでは、HTTPS、JGroups キーストア、および Red Hat Single Sign-On サーバー用のトラストストアが事前に存在している必要があります。これにより、passthrough TLS termination を使用して TLS 通信のセキュリティーを保護します。

- **sso76-https**: 同じ Pod の内部 H2 データベースでサポートされる Red Hat Single Sign-On 7.6.0。
- **sso76-postgresql**: 別の Pod の一時 PostgreSQL データベースがサポートする Red Hat Single Sign-On 7.6.0。
- **sso76-postgresql-persistent**: 別の Pod の永続的な PostgreSQL データベースがサポートする Red Hat Single Sign-On 7.6.0。

注記

MySQL または MariaDB データベースで Red Hat Single Sign-On を使用するテンプレートが削除されており、Red Hat Single Sign-On バージョン 7.4 以降は利用できません。

1.3.2. 再暗号化テンプレート

OpenShift の内部 service serving x509 certificate secrets を使用して、セキュリティーが保護されたコンテンツを提供するために使用される HTTPS キーストアを自動的に作成します。JGroups クラストラフィックは **AUTH** プロトコルを使用して認証され、**ASYM_ENCRYPT** プロトコルを使用して暗号化されます。Red Hat Single Sign-On サーバーのトラストストアも自動的に作成されます。このファイルには、HTTPS キーストアの証明書の署名に使用されるCA 証明書ファイル **/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt** が含まれます。

さらに、Red Hat Single Sign-On サーバーのトラストストアには、Java システムパスにある既知の信頼できる CA 証明書ファイルすべてが事前に入力されています。以下のテンプレートは、re-encryption TLS termination を使用してTLS 通信のセキュリティーを保護します。

- **sso76-x509-https**: 内部 H2 データベースでサポートされる自動生成された HTTPS キーストアおよび Red Hat Single Sign-On トラストストアを持つ Red Hat Single Sign-On 7.6.0。JGroups プロトコル **ASYM_ENCRYPT** はクラストラフィックの暗号化に使用されます。

- **sso76-x509-postgresql-persistent**: 自動生成された HTTPS キーストアおよび Red Hat Single Sign-On トラストストアを持つ Red Hat Single Sign-On 7.6.0。永続的な PostgreSQL データベースでサポートされます。JGroups プロトコル **ASYM_ENCRYPT** はクラストラフィックの暗号化に使用されます。

1.3.3. その他のテンプレート

Red Hat Single Sign-On と統合する他のテンプレートも利用できます。

- **eap64-sso-s2i**: Red Hat Single Sign-On が有効になっている Red Hat JBoss Enterprise Application Platform 6.4

- **eap71-sso-s2i**: Red Hat Single Sign-On が有効になっている Red Hat JBoss Enterprise Application Platform 7.1

- **datavirt63-secure-s2i**: Red Hat Single Sign-On が有効になっている Red Hat JBoss Data Virtualization 6.3

このテンプレートには、デプロイ時に Red Hat Single Sign-On クライアントの自動登録を有効にする Red Hat Single Sign-On 固有の環境変数が含まれます。

関連情報

- Red Hat Single Sign-On Client の自動および手動での登録方法
- Passthrough TLS termination
- Re-encryption TLS termination

1.4. バージョンの互換性とサポート

OpenShift イメージバージョンの互換性の詳細については、Supported Configurations のページを参照してください。

注記

7.0 から 7.4 までの Red Hat Single Sign-On for OpenShift イメージバージョン番号は非推奨となり、イメージおよびアプリケーションテンプレートの更新を受け取らなくなりました。

新規アプリケーションをデプロイするには、Red Hat Single Sign-On for OpenShift イメージのバージョン 7.5 または 7.6.0 と、これらのイメージバージョン固有のアプリケーションテンプレートを使用します。

第2章 RED HAT SINGLE SIGN-ON FOR OPENSHIFT の設定

2.1. RED HAT SINGLE SIGN-ON FOR OPENSHIFT IMAGE STREAMS およびアプリケーションテンプレートの使用

Red Hat JBoss Middleware for OpenShift イメージは、認証を必要とするセキュリティー保護された Red Hat Registry registry.redhat.io からオンデマンドで取得されます。コンテンツを取得するには、Red Hat アカウントを使用してレジストリーにログインする必要があります。

registry.redhat.io からのコンテナーイメージを共有環境 (OpenShift など) で使用するには、管理者が個人の Red Hat カスタマーポータルの認証情報の代わりに Registry Service Account (認証トークンとも呼ばれる) を使用することを推奨します。

手順

1. レジストリーサービスアカウントを作成するには、Registry Service Account Management Application に移動して、必要に応じてログインします。
2. **Registry Service Accounts**ページから、**Create Service Account**をクリックします。
3. サービスアカウントの名前 (**registry.redhat.io-sa**) を指定します。これには、固定されたランダムな文字列が追加されます。
    a. サービスアカウントの説明を入力します (例: **Service account to consume container images from registry.redhat.io.**)。
    b. **Create** をクリックします。
4. サービスアカウントの作成後に、**Registry Service Accounts**ページに表示される表の **Account name** 列の **registry.redhat.io-sa** をクリックします。
5. 最後に、**OpenShift Secret** タブをクリックし、そのページに一覧表示されているすべての手順を実行します。

詳細は、「Red Hat コンテナーレジストリーの認証」を参照してください。

手順

1. クラスター管理者またはグローバル **openshift** プロジェクトへのプロジェクト管理者アクセスを持つユーザーとしてログインしていることを確認します。
2. OpenShift Container Platform のバージョンに基づいてコマンドを選択します。
    a. マスターホストの（一部）で OpenShift Container Platform v3 ベースのクラスターインスタンスを実行している場合、以下を実行します。

```
$ oc login -u system:admin
```

    b. OpenShift Container Platform v4 ベースのクラスターインスタンスを実行している場合は、CLI に kubeadmin ユーザーとしてログインします。

```
$ oc login -u kubeadmin -p password https://openshift.example.com:6443
```

3. 以下のコマンドを実行して、**openshift** プロジェクトで Red Hat Single Sign-On 7.6.0 リソースのコアセットを更新します。

```
$ for resource in sso76-image-stream.json \
  sso76-https.json \
  sso76-postgresql.json \
  sso76-postgresql-persistent.json \
  sso76-x509-https.json \
  sso76-x509-postgresql-persistent.json
do
  oc replace -n openshift --force -f \
  https://raw.githubusercontent.com/jboss-container-images/redhat-sso-7-openshift-image/sso76-dev/templates/${resource}
done
```

4. 以下のコマンドを実行して、Red Hat Single Sign-On 7.6.0 OpenShift イメージストリームを **openshift** プロジェクトにインストールします。

```
$ oc -n openshift import-image rh-sso-7/sso76-openshift-rhel8:7.6 --from=registry.redhat.io/rh-sso-7/sso76-openshift-rhel8:7.6 --confirm
```

2.2. RED HAT SINGLE SIGN-ON イメージのデプロイ

2.2.1. デプロイメントの準備

手順

1. **cluster:admin** ロールを持つユーザーとして OpenShift CLI にログインします。
2. 新しいプロジェクトを作成します。

```
$ oc new-project sso-app-demo
```

3. **view** ロールを **デフォルト** のサービスアカウントに追加します。これにより、サービスアカウントが **sso-app-demo** 名前空間のすべてのリソースを表示できるようになります。これは、クラスターの管理に必要です。

```
$ oc policy add-role-to-user view system:serviceaccount:$(oc project -q):default
```

2.2.2. アプリケーションテンプレートを使用した Red Hat Single Sign-On イメージのデプロイ

次のいずれかのインターフェイスを使用して、テンプレートをデプロイできます。

- OpenShift CLI
- OpenShift 3.x web console
- OpenShift 4.x web console

2.2.2.1. OpenShift CLI を使用したテンプレートのデプロイ

前提条件

- Using the Red Hat Single Sign-On for OpenShift Image Streams and application templates で説明されている手順を実行します。

手順

1. 利用可能な Red Hat Single Sign-On アプリケーションテンプレートの一覧を表示します。

```
$ oc get templates -n openshift -o name | grep -o 'sso76.\+'
sso76-https
sso76-postgresql
sso76-postgresql-persistent
sso76-x509-https
sso76-x509-postgresql-persistent
```

2. 選択したものをデプロイします。

```
$ oc new-app --template=sso76-x509-https
--> Deploying template "openshift/sso76-x509-https" to project sso-app-demo

    Red Hat Single Sign-On 7.6 (Ephemeral)
    ---------
    An example Red Hat Single Sign-On 7 application. For more information about using this
template, see <link xlink:href="https://github.com/jboss-openshift/application-
templates">https://github.com/jboss-openshift/application-templates</link>.

    A new Red Hat Single Sign-On service has been created in your project. The admin
username/password for accessing the master realm using the Red Hat Single Sign-On
console is IACfQO8v/nR7llVSVb4Dye3TNRbXoXhRpAKTmiCRc. The HTTPS keystore used
for serving secure content, the JGroups keystore used for securing JGroups
communications, and server truststore used for securing Red Hat Single Sign-On requests
were automatically created using OpenShift's service serving x509 certificate secrets.

    * With parameters:
      * Application Name=sso
      * JGroups Cluster Password=jg0Rssom0gmHBnooDF3Ww7V4Mu5RymmB #
generated
      * Datasource Minimum Pool Size=
      * Datasource Maximum Pool Size=
      * Datasource Transaction Isolation=
      * ImageStream Namespace=openshift
      * Red Hat Single Sign-On Administrator Username=IACfQO8v # generated
      * Red Hat Single Sign-On Administrator
Password=nR7llVSVb4Dye3TNRbXoXhRpAKTmiCRc # generated
      * Red Hat Single Sign-On Realm=
      * Red Hat Single Sign-On Service Username=
      * Red Hat Single Sign-On Service Password=
      * Container Memory Limit=1Gi

--> Creating resources ...
  service "sso" created
  service "secure-sso" created
  service "sso-ping" created
  route "sso" created
  route "secure-sso" created
```

```
    deploymentconfig "sso" created
--> Success
    Run 'oc status' to view your app.
```

2.2.2.2. OpenShift 3.x Web コンソールを使用したテンプレートのデプロイ

前提条件

- Using the Red Hat Single Sign-On for OpenShift Image Streams and application templates で説明されている手順を実行します。

手順

1. OpenShift Web コンソールにログインし、**sso-app-demo** プロジェクトスペースを選択します。
2. **Add to Project** をクリックした後、**Browse Catalog** をクリックしてデフォルトのイメージストリームおよびテンプレートを一覧表示します。
3. **Filter by Keyword** 検索バーを使用して、一覧を **sso** に一致するものに制限します。**Middleware** をクリックした後、**Integration** をクリックして必要なアプリケーションテンプレートを表示する必要がある場合があります。
4. Red Hat Single Sign-On アプリケーションテンプレートを選択します。この例では、**Red Hat Single Sign-On 7.6(Ephemeral)** を使用します。
5. **Information** ステップで **Next** をクリックします。
6. **Add to Project** ドロップダウンメニューから、**sso-app-demo** プロジェクト領域を選択します。**Next** をクリックします。
7. **Binding** ステップで **Do not bind at this time** ラジオボタンを選択します。**Create** をクリックして次に進みます。
8. **Results** 手順で **Continue to the project overview**をクリックし、デプロイメントのステータスを確認します。

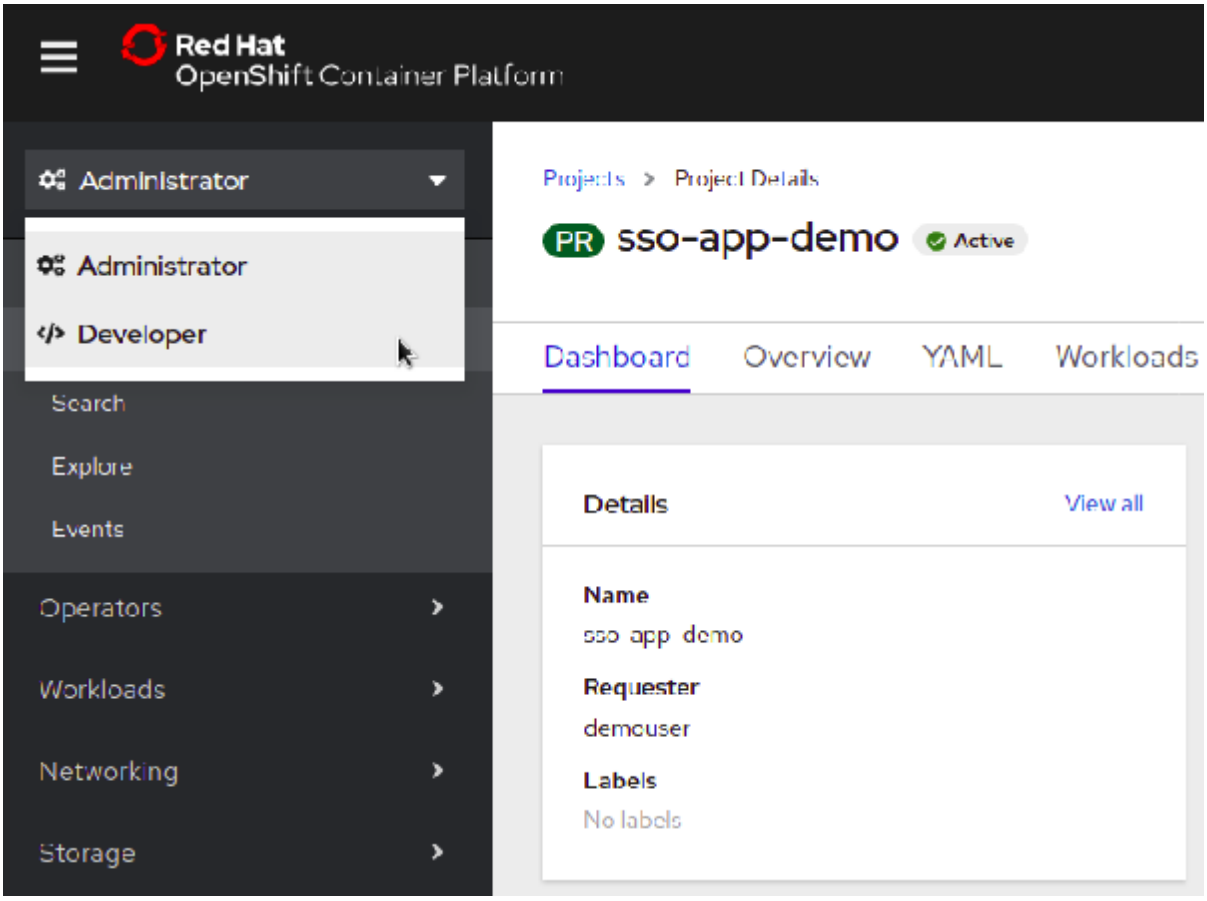
2.2.2.3. OpenShift 4.x Web コンソールを使用したテンプレートのデプロイ

前提条件

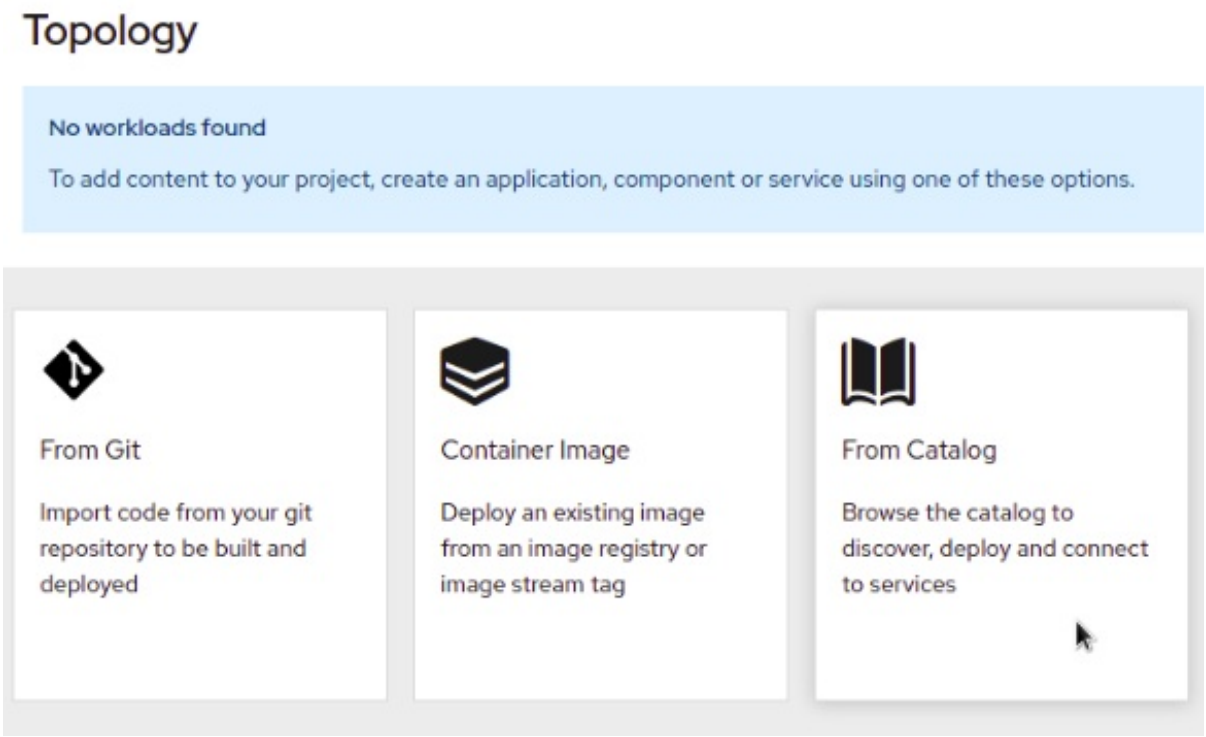
- Using the Red Hat Single Sign-On for OpenShift Image Streams and application templates で説明されている手順を実行します。

手順

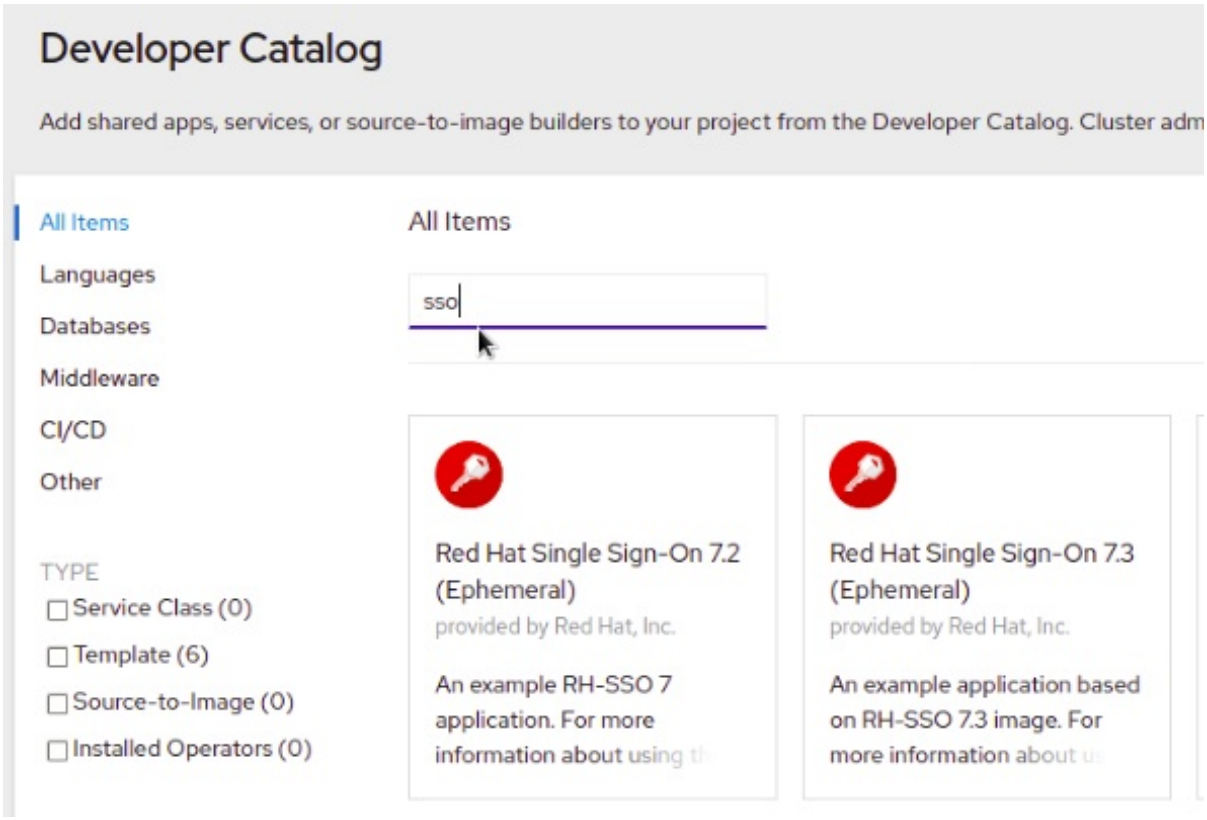
1. OpenShift Web コンソールにログインし、**sso-app-demo** プロジェクト領域を選択します。
2. 左側のサイドバーで **Administrator** タブをクリックし、**</> Developer** をクリックします。



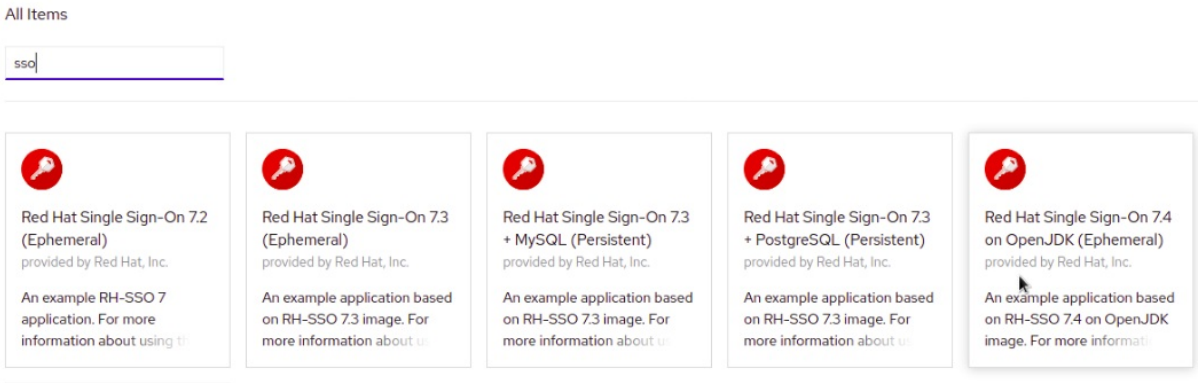
3. **From Catalog** をクリックします。



4. **sso** を検索します。



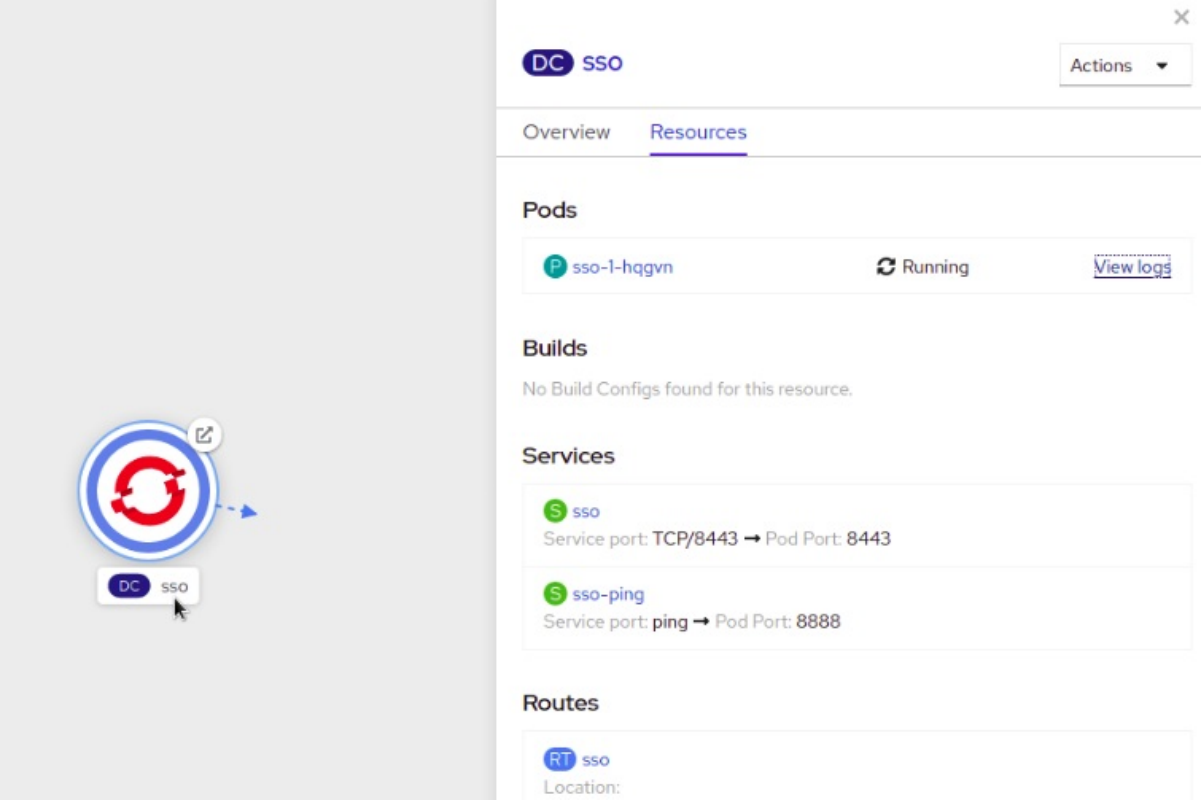
5. **OpenJDK(Ephemeral)上の Red Hat Single Sign-On 7.6**などのテンプレートを選択します。



6. **Instantiate Template** をクリックします。

7. 必要に応じてテンプレートパラメーターを変更し、**Create** をクリックします。

8. OpenShift イメージの Red Hat Single Sign-On がデプロイされていることを確認します。



2.3. RED HAT SINGLE SIGN-ON POD の管理者コンソールへのアクセス

手順

1. テンプレートのデプロイ後に、利用可能なルートを特定します。

```
$ oc get routes
NAME     HOST/PORT
sso      sso-sso-app-demo.openshift.example.com
```

2. Red Hat Single Sign-On 管理コンソールにアクセスします。

```
https://sso-sso-app-demo.openshift.example.com/auth/admin
```

3. administrator account のログイン資格情報を入力します。

第3章 高度な手順の実行

この章では、Red Hat Single Sign-On サーバーのキーストアとトラストストアの設定、管理者アカウントの作成、使用可能な Red Hat Single Sign-On クライアント登録方法の概要、およびクラスターリングの設定に関するガイダンスなどの高度な手順について説明します。

3.1. パススルー TLS ターミネーションテンプレートの展開

これらのテンプレートを使用してデプロイできます。HTTPS、JGroups キーストア、および Red Hat Single Sign-On サーバートラストストアがすでに存在している必要があるため、カスタム HTTPS、JGroups キーストア、および Red Hat Single Sign-On サーバートラストストアを使用して Red Hat Single Sign-On サーバー Pod をインスタンス化するために使用できます。

3.1.1. デプロイメントの準備

手順

1. **cluster:admin** ロールを持つユーザーとして OpenShift CLI にログインします。
2. 新しいプロジェクトを作成します。

```
$ oc new-project sso-app-demo
```

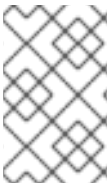
3. **view** ロールを **デフォルト** のサービスアカウントに追加します。これにより、サービスアカウントが sso-app-demo 名前空間のすべてのリソースを表示できるようになります。これは、クラスターの管理に必要です。

```
$ oc policy add-role-to-user view system:serviceaccount:$(oc project -q):default
```

3.1.2. Red Hat Single Sign-On サーバーの HTTPS および JGroups キーストアの作成

この手順では、**openssl** ツールキットを使用して、HTTPS キーストアに署名するための CA 証明書を生成し、Red Hat Single Sign-On サーバー用のトラストストアを作成します。**Java Development Kit に含まれる** パッケージの **keytool** を使用して、これらのキーストアの自己署名証明書を生成します。

re-encryption TLS termination を使用した Red Hat Single Sign-On アプリケーションテンプレートは、前述の HTTPS および JGroups キーストアおよび Red Hat Single Sign-On サーバートラストストアを事前に準備する **必要**はなく、**期待**もしません。



注記

既存の HTTPS/JGroups キーストアを使用して Red Hat Single Sign-On サーバーをプロビジョニングする場合は、代わりにいくつかのパススルーテンプレートを使用してください。

再暗号化テンプレートは、OpenShift の内部 service serving x509 certificate secrets を使用して HTTPS および JGroups キーストアを自動的に作成します。

Red Hat Single Sign-On サーバートラストストアも自動的に作成されます。このファイルには、これらのクラスター証明書の作成に使用される **/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt** CA 証明書ファイルが含まれます。さらに、Red Hat Single Sign-On サーバーのトラストストアには、Java システムパスにある既知の信頼できる CA 証明書ファイルすべてが事前に入力されています。

前提条件

パススルー TLS ターミネーションを使用する Red Hat Single Sign-On アプリケーションテンプレートでは、以下をデプロイする必要があります。

- https トラフィックの暗号化に使用される HTTPS キーストア
- クラスターのノード間の JGroups 通信の暗号化に使用される JGroups キーストア
- Red Hat Single Sign-On 要求のセキュリティー保護に使用される Red Hat Single Sign-On サーバートラストストア

注記

実稼働環境の場合、Red Hat は、SSL 暗号化接続 (HTTPS) 用に検証された認証局 (CA) から購入した独自の SSL 証明書を使用することを推奨します。

自己署名証明書または購入した SSL 証明書でキーストアを作成する方法に関する詳細は、「SSL 暗号化キーおよび証明書の生成」を参照してください。

HTTPS キーストアを作成します。

手順

1. CA 証明書を生成します。パスワードを選択し、忘れないようにしてください。以下の CA 証明書で証明書署名要求に署名する際に、同じパスワードを指定します。

```
$ openssl req -new -newkey rsa:4096 -x509 -keyout xpaas.key -out xpaas.crt -days 365 -subj
"/CN=xpaas-sso-demo.ca"
```

2. HTTPS キーストアの秘密鍵を生成する**mykeystorepass** をキーストアパスワードとして提供します。

```
$ keytool -genkeypair -keyalg RSA -keysize 2048 -dname "CN=secure-sso-sso-app-
demo.openshift.example.com" -alias jboss -keystore keystore.jks
```

3. HTTPS キーストアの証明書署名要求を生成します。**mykeystorepass** をキーストアパスワードとして提供します。

```
$ keytool -certreq -keyalg rsa -alias jboss -keystore keystore.jks -file sso.csr
```

4. CA 証明書を使用して証明書署名要求に署名します。CA 証明書の生成 に使用されたパスワードを指定します。

```
$ openssl x509 -req <(printf "subjectAltName=DNS:secure-sso-sso-app-
demo.openshift.example.com") -CA xpaas.crt -CAkey xpaas.key -in sso.csr -out sso.crt -days
365 -CAcreateserial
```

注記

上記のコマンドを 1 行で機能させるために、コマンドにはプロセス置換 (**<() 構文**) が含まれています。現在のシェル環境がそのような構文をサポートしていることを確認してください。そうしないと、**syntax error near unexpected token `('** が発生する可能性があります。

5. CA 証明書を HTTPS キーストアにインポートします。**mykeystorepass** をキーストアパスワードとして提供します。**Trust this certificate? [no]:** の質問に **yes** を返信します。

```
$ keytool -import -file xpaas.crt -alias xpaas.ca -keystore keystore.jks
```

6. 署名済み証明書署名要求を HTTPS キーストアにインポートします。**mykeystorepass** をキーストアパスワードとして提供します。

```
$ keytool -import -file sso.crt -alias jboss -keystore keystore.jks
```

JGroups キーストアのセキュアキーを生成します。

キーストアパスワードとして **パスワード** を指定します。

```
$ keytool -genseckey -alias secret-key -storetype JCEKS -keystore jgroups.jceks
```

CA 証明書を新しい Red Hat Single Sign-On サーバートラストストアにインポートします。

mykeystorepass をトラストストアのパスワードとして指定します。**Trust this certificate? [no]:** の質問に **yes** を返信します。

```
$ keytool -import -file xpaas.crt -alias xpaas.ca -keystore truststore.jks
```

3.1.3. シークレットの作成

手順

OpenShift がパスワードやキーストアなどの機密情報を保持するために使用するシークレットと呼ばれるオブジェクトを作成します。

1. 前のセクション で生成された HTTPS および JGroups キーストアのシークレットと Red Hat Single Sign-On サーバートラストストアを作成します。

```
$ oc create secret generic sso-app-secret --from-file=keystore.jks --from-file=jgroups.jceks --from-file=truststore.jks
```

2. これらのシークレットを、Red Hat Single Sign-On Pod の実行に使用されるデフォルトのサービスアカウントにリンクします。

```
$ oc secrets link default sso-app-secret
```

関連情報

- シークレットの概要
- デフォルトのプロジェクトサービスアカウントおよびロール

3.1.4. OpenShift CLI を使用したパススルー TLS テンプレートのデプロイ

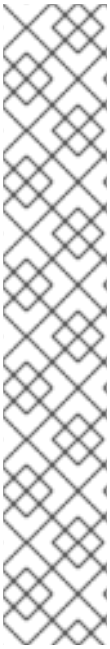
keystores と secrets を作成したら、**oc** コマンドを使用してパススルー TLS ターミネーションテンプレートをデプロイします。

3.1.4.1. oc コマンドガイドライン

以下の **oc** コマンドで
は、**SSO_ADMIN_USERNAME**、**SSO_ADMIN_PASSWORD**、**HTTPS_PASSWORD**、**JGROUPS_ENCRY**
および **SSO_TRUSTSTORE_PASSWORD** 変数の値は、**sso76-https** Red Hat Single Sign-On アプリケーションテンプレートのデフォルト値に一致します。

実稼働環境では、Red Hat Single Sign-On サーバーの管理者ユーザーアカウントの強力なユーザー名とパスワード、および HTTPS と JGroups キーストア、および Red Hat Single Sign-On サーバーのトラストストアのパスワードを生成するためのガイダンスについて、組織のオンサイトポリシーを参照することを Red Hat は推奨します。

また、テンプレートを作成するときは、パスワードをキーストアの作成時に提供されたパスワードと一致させてください。別のユーザー名またはパスワードを使用した場合は、環境に合わせてテンプレートのパラメーターの値を変更してください。



注記

次の **keytool** コマンドを使用して、証明書に関連付けられているエイリアス名を確認できます。**keytool** は、Java Development Kit に含まれているパッケージです。

```
$ keytool -v -list -keystore keystore.jks | grep Alias
Enter keystore password:  mykeystorepass
Alias name: xpaas.ca
Alias name: jboss
```

```
$ keytool -v -list -keystore jgroups.jceks -storetype jceks | grep Alias
Enter keystore password:  password
Alias name: secret-key
```

最後に、次のコマンドの
SSO_ADMIN_USERNAME、**SSO_ADMIN_PASSWORD**、**SSO_REALM** テンプレートパラメーターは任意です。

3.1.4.2. サンプル oc コマンド

```
$ oc new-app --template=sso76-https \
 -p HTTPS_SECRET="sso-app-secret" \
 -p HTTPS_KEYSTORE="keystore.jks" \
 -p HTTPS_NAME="jboss" \
 -p HTTPS_PASSWORD="mykeystorepass" \
 -p JGROUPS_ENCRYPT_SECRET="sso-app-secret" \
 -p JGROUPS_ENCRYPT_KEYSTORE="jgroups.jceks" \
 -p JGROUPS_ENCRYPT_NAME="secret-key" \
 -p JGROUPS_ENCRYPT_PASSWORD="password" \
 -p SSO_ADMIN_USERNAME="admin" \
 -p SSO_ADMIN_PASSWORD="redhat" \
 -p SSO_REALM="demorealm" \
 -p SSO_TRUSTSTORE="truststore.jks" \
 -p SSO_TRUSTSTORE_PASSWORD="mykeystorepass" \
 -p SSO_TRUSTSTORE_SECRET="sso-app-secret"
--> Deploying template "openshift/sso76-https" to project sso-app-demo

     Red Hat Single Sign-On 7.6.0 (Ephemeral with passthrough TLS)
```

```
    ---------
    An example Red Hat Single Sign-On 7 application. For more information about using this
template, see https://github.com/jboss-openshift/application-templates.

    A new Red Hat Single Sign-On service has been created in your project. The admin
username/password for accessing the master realm via the Red Hat Single Sign-On console is
admin/redhat. Please be sure to create the following secrets: "sso-app-secret" containing the
keystore.jks file used for serving secure content; "sso-app-secret" containing the jgroups.jceks file
used for securing JGroups communications; "sso-app-secret" containing the truststore.jks file used
for securing Red Hat Single Sign-On requests.

    * With parameters:
      * Application Name=sso
      * Custom http Route Hostname=
      * Custom https Route Hostname=
      * Server Keystore Secret Name=sso-app-secret
      * Server Keystore Filename=keystore.jks
      * Server Keystore Type=
      * Server Certificate Name=jboss
      * Server Keystore Password=mykeystorepass
      * Datasource Minimum Pool Size=
      * Datasource Maximum Pool Size=
      * Datasource Transaction Isolation=
      * JGroups Secret Name=sso-app-secret
      * JGroups Keystore Filename=jgroups.jceks
      * JGroups Certificate Name=secret-key
      * JGroups Keystore Password=password
      * JGroups Cluster Password=yeSppLfp # generated
      * ImageStream Namespace=openshift
      * Red Hat Single Sign-On Administrator Username=admin
      * Red Hat Single Sign-On Administrator Password=redhat
      * Red Hat Single Sign-On Realm=demorealm
      * Red Hat Single Sign-On Service Username=
      * Red Hat Single Sign-On Service Password=
      * Red Hat Single Sign-On Trust Store=truststore.jks
      * Red Hat Single Sign-On Trust Store Password=mykeystorepass
      * Red Hat Single Sign-On Trust Store Secret=sso-app-secret
      * Container Memory Limit=1Gi

--> Creating resources ...
    service "sso" created
    service "secure-sso" created
    service "sso-ping" created
    route "sso" created
    route "secure-sso" created
    deploymentconfig "sso" created
--> Success
    Run 'oc status' to view your app.
```

関連情報

- Passthrough TLS Termination

3.2. RED HAT SINGLE SIGN-ON サーバーのホスト名のカスタマイズ。

ホスト名 SPI では、Red Hat Single Sign-On サーバーのホスト名を設定するための柔軟な方法が導入されました。デフォルトのホスト名プロバイダーの 1 つが **デフォルト** です。このプロバイダーは、現在は非推奨となった元の **リクエスト** プロバイダーよりも強化された機能を提供します。追加の設定がないと、要求ヘッダーを使用して元の **要求** プロバイダーと同様にホスト名を判別します。

デフォルト プロバイダーの設定オプションについては、『サーバー インストールおよび設定ガイド』を参照してください。**frontendUrl** オプションは、**SSO_FRONTEND_URL** 環境変数で設定できます。

注記

後方互換性を確保するために、**SSO_HOSTNAME** も設定されている場合は **SSO_FRONTEND_URL** 設定は無視されます。

ホスト名プロバイダーの別のオプションも **修正され**、固定ホスト名の設定が可能になります。後者では、有効なホスト名のみを使用でき、内部アプリケーションが代替 URL を使用して Red Hat Single Sign-On サーバーを呼び出すことができます。

手順

以下のコマンドを実行して、Red Hat Single Sign-On サーバーの **固定** ホスト名 SPI プロバイダーを設定します。

1. **SSO_HOSTNAME** 環境変数を Red Hat Single Sign-On サーバーの必要なホスト名に設定して、Red Hat Single Sign-On for OpenShift イメージをデプロイします。

```
$ oc new-app --template=sso76-x509-https \
  -p SSO_HOSTNAME="rh-sso-server.openshift.example.com"
```

2. Red Hat Single Sign-On サービスのルート名を特定します。

```
$ oc get routes
NAME     HOST/PORT
sso      sso-sso-app-demo.openshift.example.com
```

3. 上記の **SSO_HOSTNAME** 環境変数の値として指定されたホスト名に一致するように **host:** フィールドを変更します。

注記

必要に応じて、以下のコマンドで **rh-sso-server.openshift.example.com** 値を調整します。

```
$ oc patch route/sso --type=json -p '[{"op": "replace", "path": "/spec/host", "value": "rh-sso-server.openshift.example.com"}]'
```

成功すると、直前のコマンドは以下の出力を返します。

```
route "sso" patched
```

3.3. 外部データベースへの接続

Red Hat Single Sign-On は、外部 (OpenShift クラスター) データベースに接続するように設定できます。そのためには、**sso-{database name}** Endpoints オブジェクトを変更して適切なアドレスを参照する必要があります。手順は、OpenShift のマニュアル に記載されています。

ヒント: 最も簡単な方法は、テンプレートから Red Hat Single Sign-On をデプロイしてから Endpoints オブジェクトを変更することです。DeploymentConfig の一部のデータソース設定変数を更新する必要がある場合があります。完了したら、新しいデプロイメントをロールアウトするだけで実行します。

3.4. カスタム JDBC ドライバーの使用

データベースに接続するには、そのデータベースの JDBC ドライバーが存在し、Red Hat Single Sign-On が適切に設定されている必要があります。現在、イメージで利用可能な JDBC ドライバーは PostgreSQL JDBC ドライバーのみです。他のデータベースの場合は、カスタム JDBC ドライバーと CLI スクリプトを使用して Red Hat Single Sign-On イメージを拡張し、接続プロパティーを設定する必要があります。以下の手順は、MariaDB ドライバーを例として取る方法を示しています。それに応じて、その他のデータベースドライバーの例を更新します。

手順

1. 空のディレクトリーを作成します。
2. JDBC ドライバーバイナリーをこのディレクトリーにダウンロードします。
3. 以下の内容で、このディレクトリーに新規の **Dockerfile** ファイルを作成します。他のデータベースの場合は、**mariadb-java-client-2.5.4.jar** を適切なドライバーのファイル名に置き換えます。

```
FROM rh-sso-7/sso76-openshift-rhel8:latest

COPY sso-extensions.cli /opt/eap/extensions/
COPY mariadb-java-client-2.5.4.jar /opt/eap/extensions/jdbc-driver.jar
```

4. 以下の内容で、このディレクトリーに新しい **sso-extensions.cli** ファイルを作成します。デプロイメントのニーズに応じて、イタリック体の変数の値を更新します。

```
batch

set DB_DRIVER_NAME=mariadb
set DB_USERNAME=username
set DB_PASSWORD=password
set DB_DRIVER=org.mariadb.jdbc.Driver
set DB_XA_DRIVER=org.mariadb.jdbc.MariaDbDataSource
set DB_JDBC_URL=jdbc:mariadb://jdbc-host/keycloak
set DB_EAP_MODULE=org.mariadb

set FILE=/opt/eap/extensions/jdbc-driver.jar

module add --name=$DB_EAP_MODULE --resources=$FILE --
dependencies=javax.api,javax.resource.api
/subsystem=datasources/jdbc-driver=$DB_DRIVER_NAME:add( \
  driver-name=$DB_DRIVER_NAME, \
  driver-module-name=$DB_EAP_MODULE, \
  driver-class-name=$DB_DRIVER, \
  driver-xa-datasource-class-name=$DB_XA_DRIVER \
)
```

```
/subsystem=datasources/data-source=KeycloakDS:remove()
/subsystem=datasources/data-source=KeycloakDS:add( \
  jndi-name=java:jboss/datasources/KeycloakDS, \
  enabled=true, \
  use-java-context=true, \
  connection-url=$DB_JDBC_URL, \
  driver-name=$DB_DRIVER_NAME, \
  user-name=$DB_USERNAME, \
  password=$DB_PASSWORD \
)

run-batch
```

5. このディレクトリーで以下のコマンドを入力し、**project/name:tag** を任意の名前に置き換えてイメージをビルドします。**docker** は、**podman** の代わりに使用できます。

```
podman build -t docker-registry-default/project/name:tag .
```

6. ビルドが完了したら、イメージをデプロイするために OpenShift で使用されるレジストリーにイメージをプッシュします。詳細は、OpenShift ガイド を参照してください。

3.5. RED HAT SINGLE SIGN-ON サーバーの管理者アカウントの作成

Red Hat Single Sign-Onは、事前設定された管理アカウントをそのまま提供しません。この管理者アカウントは、**master** レルムの管理コンソールにログインして、レルムやユーザーの作成 Red Hat Single Sign-On でセキュリティー保護するためのアプリケーションの登録などのサーバーメンテナンス操作を実行するために必要です。

管理者アカウントは、以下の方法で作成できます。

- Red Hat Single Sign-On アプリケーションテンプレートのデプロイ時に **SSO_ADMIN_USERNAME** および **SSO_ADMIN_PASSWORD** パラメーター の値を提供することで。
- OpenShift イメージの Red Hat Single Sign-On がアプリケーションテンプレートなしでデプロイされる場合に、特定の Red Hat Single Sign-On Pod へのリモートシェルセッション で。

注記

Red Hat Single Sign-On では、最初の管理者アカウントを Welcome Page の Web フォームで作成できますが、Welcome ページが localhost からアクセスされている場合に限ります。この管理者アカウント作成方法は、Red Hat Single Sign-On for OpenShift イメージには適用されません。

3.5.1. テンプレートパラメーターを使用した管理者アカウントの作成

Red Hat Single Sign-On アプリケーションテンプレートをデプロイする場合は、**SSO_ADMIN_USERNAME** パラメーターおよび **SSO_ADMIN_PASSWORD** パラメーターは、**master** レルム用に作成される Red Hat Single Sign-On サーバーの管理者アカウントのユーザー名とパスワードを示します。

これらのパラメーターはいずれも必須です。指定されていない場合、それらはテンプレートがインスタンス化される時に OpenShift 命令メッセージとして自動生成され、表示されます。

Red Hat Single Sign-On サーバーの管理者アカウントのライフサイクルは、Red Hat Single Sign-On サーバーのデータベースを保存するために使用するストレージタイプによって異なります。

- インメモリーデータベースモード（**sso76-https テンプレートおよび sso76-x509-https** テンプレート）の場合、アカウントは特定の Red Hat Single Sign-On Pod のライフサイクル全体で存在します（保存されたアカウントデータは Pod の破棄時に失われます）。
- 一時データベースモード（ **sso76-postgresql** テンプレート）の場合、アカウントはデータベース Pod のライフサイクル全体で存在します。Red Hat Single Sign-On Pod が破壊的な場合でも、保存されたアカウントデータは、データベース Pod が実行中であることを前提として保持されます。
- 永続データベースモード（**sso76-postgresql-persistent および sso76- x509-postgresql-persistent** テンプレート）の場合、アカウントは、データベースデータを保持するために使用される永続的メディアのライフサイクル全体で存在します。つまり、Red Hat Single Sign-On とデータベース Pod の両方が破壊されても、保存されたアカウントデータは保持されます。

Red Hat Single Sign-On アプリケーションテンプレートをデプロイして、(新しい Red Hat Single Sign-On アプリケーションをインスタンス化する必要があるたびに) アプリケーションに対応する OpenShift デプロイメント設定を取得し、そのデプロイメント設定を複数回再利用することが一般的です。

RH_SSO サーバーの管理者アカウントを作成したら **一時的または永続的データベースモード** の場合、新しい Red Hat Single Sign-On アプリケーションをデプロイする前に、デプロイメント設定から **SSO_ADMIN_USERNAME** 変数および **SSO_ADMIN_PASSWORD** 変数を削除します。

手順

以下のコマンドを実行して、管理者アカウントの作成後に再利用できるように、Red Hat Single Sign-On アプリケーションのデプロイメント設定を作成します。

1. Red Hat Single Sign-On アプリケーションのデプロイメント設定を特定します。

```
$ oc get dc -o name
deploymentconfig/sso
deploymentconfig/sso-postgresql
```

2. **SSO_ADMIN_USERNAME** および **SSO_ADMIN_PASSWORD** 変数の設定を削除します。

```
$ oc set env dc/sso \
  -e SSO_ADMIN_USERNAME="" \
  -e SSO_ADMIN_PASSWORD=""
```

3.5.2. リモートシェルセッションによる Red Hat Single Sign-On Pod への管理者アカウントの作成

テンプレートを使用せずにイメージストリームから直接 Red Hat Single Sign-On for OpenShift イメージをデプロイする場合は、以下のコマンドを使用して、Red Hat Single Sign-On サーバーの **master** レルムの管理者アカウントを作成します。

前提条件

- Red Hat Single Sign-On アプリケーション Pod が開始されました。

手順

1. Red Hat Single Sign-On アプリケーション Pod を特定します。

```
$ oc get pods
NAME                  READY     STATUS    RESTARTS   AGE
sso-12-pt93n          1/1       Running   0          1m
sso-postgresql-6-d97pf   1/1       Running   0          2m
```

2. OpenShift コンテナー用に Red Hat Single Sign-On にリモートシェルセッションを開きます。

```
$ oc rsh sso-12-pt93n
sh-4.2$
```

3. **add-user-keycloak.sh** スクリプトを使用して、コマンドラインで **master** レルムの Red Hat Single Sign-On サーバー管理者アカウントを作成します。

```
sh-4.2$ cd /opt/eap/bin/
sh-4.2$ ./add-user-keycloak.sh \
      -r master \
      -u sso_admin \
      -p sso_password
Added 'sso_admin' to '/opt/eap/standalone/configuration/keycloak-add-user.json', restart
server to load user
```

注記

上記の例の「sso_admin」または「sso_password」 認証情報はデモ目的にのみ使用されます。安全なユーザー名とパスワードの作成方法は、組織内で適用されるパスワードポリシーを参照してください。

4. 基盤の JBoss EAP サーバーインスタンスを再起動して、新たに追加したユーザーアカウントをロードします。サーバーが正しく再起動するまで待ちます。

```
sh-4.2$ ./jboss-cli.sh --connect ':reload'
{
   "outcome" => "success",
   "result" => undefined
}
```

警告

サーバーを再起動する場合は、コンテナー全体ではなく、実行中の Red Hat Single Sign-On コンテナー内で JBoss EAP プロセスを再起動することが重要です。これは、**master** レルムの Red Hat Single Sign-On サーバー管理アカウントなしに、コンテナー全体の再起動がゼロから再作成されるためです。

5. 上記の手順で作成した認証情報を使用して、Red Hat Single Sign-On サーバーの **master** レルムの管理コンソールにログインします。ブラウザーで、Red Hat Single Sign-On Web サーバー

の場合は **http://sso-<project-name>.<hostname>/auth/admin** に移動するか、暗号化された Red Hat Single Sign-On Web サーバーの場合は **https://secure-sso-<project-name>.<hostname>/auth/admin** に移動し、管理者ユーザーの作成に使用するユーザー名とパスワードを指定します。

関連情報

- このソフトウェアで使用するためのテンプレート

3.6. RED HAT SINGLE SIGN-ON イメージのデフォルト動作のカスタマイズ

TechPreview 機能の有効化やデバッグの有効化など、Red Hat Single Sign-On イメージのデフォルト動作を変更できます。本セクションでは、JAVA_OPTS_APPEND 変数を使用してこの変更を行う方法を説明します。

前提条件

この手順では、Red Hat Single Sign-On for OpenShift イメージが、以下のテンプレートのいずれかを使用してデプロイされている ことを前提としています。

- **sso76-postgresql**
- **sso76-postgresql-persistent**
- **sso76-x509-postgresql-persistent**

手順

OpenShift Web コンソールまたは CLI を使用してデフォルトの動作を変更できます。

OpenShift Web コンソールを使用する場合は、JAVA_OPTS_APPEND 変数を sso デプロイメント設定に追加します。たとえば、TechPreview 機能を有効にするには、変数を以下のように設定します。

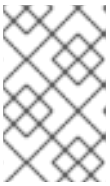
```
JAVA_OPTS_APPEND="-Dkeycloak.profile=preview"
```

CLI を使用する場合は、以下のコマンドを使用して、「前提条件」に記載されているテンプレートを使用して Red Hat Single Sign-On Pod がデプロイされた場合に TechPreview 機能を有効にします。

1. Red Hat Single Sign-On Pod を縮小します。

```
$ oc get dc -o name
deploymentconfig/sso
deploymentconfig/sso-postgresql

$ oc scale --replicas=0 dc sso
deploymentconfig "sso" scaled
```



注記

上記のコマンドでは、PostgreSQL テンプレートを使用して Red Hat Single Sign-On for OpenShift イメージをデプロイするため、**sso-postgresql** が表示されます。

2. デプロイメント設定を編集して JAVA_OPTS_APPEND 変数を設定します。たとえば、TechPreview 機能を有効にするには、変数を以下のように設定します。

```
oc env dc/sso -e "JAVA_OPTS_APPEND=-Dkeycloak.profile=preview"
```

3. Red Hat Single Sign-On Pod を拡大します。

```
$ oc scale --replicas=1 dc sso
deploymentconfig "sso" scaled
```

4. 選択した TechPreview 機能をテストします。

3.7. デプロイメントプロセス

デプロイが完了すると、**sso76-https** テンプレートおよび **sso76-x509-https** テンプレートは、データベースと Red Hat Single Sign-On サーバーの両方を含む単一の Pod を作成します。**sso76-postgresql**、**sso76-postgresql-persistent**、および **sso76-x509-postgresql-persistent** テンプレートは 2 つの Pod を作成します。1 つはデータベースサーバー用で、もう 1 つは Red Hat Single Sign-On Web サーバー用です。

Red Hat Single Sign-On Web サーバー Pod が起動した後に、カスタム設定のホスト名、またはデフォルトのホスト名からアクセスできます。

- **http://sso-<project-name>.<hostname>/auth/admin**: Red Hat Single Sign-On Web サーバー用
- **https://secure-sso-<project-name>.<hostname>/auth/admin**: 暗号化された Red Hat Single Sign-On Web サーバー用

管理者ユーザーの認証情報 を使用して、**master** レルムの管理コンソールにログインします。

3.8. RED HAT SINGLE SIGN-ON CLIENT

クライアントは、ユーザー認証を要求する Red Hat Single Sign-On エンティティーです。クライアントは、ユーザー認証を提供するために Red Hat Single Sign-On を要求するアプリケーションを使用することも、認証ユーザーの代わりにサービスを開始するためのアクセストークンの要求を行うことができます。詳細は、Red Hat Single Sign-On ドキュメントの「クライアントの管理」の章 を参照してください。

Red Hat Single Sign-On は、OpenID-Connect および SAML クライアントプロトコルを提供します。

OpenID-Connect が推奨されるプロトコルとなり、3 つの異なるアクセス種別が使用されます。

- **public**: ブラウザーで直接実行し、サーバー構成が必要ない JavaScript アプリケーションに役立ちます。
- **confidential**: ブラウザーログインを実行する必要がある EAP Web アプリケーションなどのサーバー側クライアントに役立ちます。
- **bearer-only**: ベアラートークンリクエストを許可するバックエンドサービスに役立ちます。

アプリケーションの **web.xml** ファイルの **<auth-method>** キーでクライアントタイプを指定する必要があります。このファイルは、デプロイ時にイメージにより読み取られます。**<auth-method>** 要素の値を以下のように設定します。

- OpenID Connect クライアントの **KEYCLOAK**。
- SAML クライアントの **KEYCLOAK-SAML**。

以下は、OIDC クライアントを設定するアプリケーションの **web.xml** のスニペットの例です。

```
...
<login-config>
        <auth-method>KEYCLOAK</auth-method>
</login-config>
...
```

3.8.1. Red Hat Single Sign-On Client の自動および手動での登録方法

eap64-sso-s2i、**eap71-sso-s2i**、および **datavirt63-secure-s2i** に固有の変数に渡される認証情報を使用すると、クライアントアプリケーションを Red Hat Single Sign-On レルムに自動的に登録できます。

または、Red Hat Single Sign-On クライアントアダプターを設定してエクスポートし、クライアントアプリケーション設定に追加することで、クライアントアプリケーションを手動で登録できます。

3.8.1.1. Red Hat Single Sign-On Client の自動登録

Red Hat Single Sign-On クライアントの自動登録は、**eap64-sso-s2i**、**eap71-sso-s2i**、および **datavirt63-secure-s2i** に固有の Red Hat Single Sign-On 環境変数により決定されます。テンプレートに指定された Red Hat Single Sign-On 認証情報は、クライアントアプリケーションのデプロイメント時にクライアントを Red Hat Single Sign-On レルムに登録するのに使用されます。

eap64-sso-s2i、**eap71-sso-s2i**、および **datavirt63-secure-s2i** テンプレートに含まれる Red Hat Single Sign-On の環境変数は次のとおりです。

変数	説明
HOSTNAME_HTTP	http サービスルートのカスタムホスト名。<application-name>.<project>.<default-domain-suffix> のデフォルトホスト名には空白のままにします。
HOSTNAME_HTTPS	https サービスルートのカスタムホスト名。<application-name>.<project>.<default-domain-suffix> のデフォルトホスト名には空白のままにします。
SSO_URL	Red Hat Single Sign-On の Web サーバーの認証アドレス: https://secure-sso-**<project-name>**.**<hostname>**/auth
SSO_REALM	この手順用に作成された Red Hat Single Sign-On レルム。
SSO_USERNAME	**レルム管理ユーザー** の名前。
SSO_PASSWORD	ユーザーのパスワード。

変数	説明
SSO_PUBLIC_KEY	レルムによって生成された公開鍵。これは、Red Hat Single Sign-On コンソールの **Realm Settings** の **Keys** タブにあります。
SSO_BEARER_ONLY	**True** に設定すると、OpenID Connect クライアントは bearer-only として登録されます。
SSO_ENABLE_CORS	**True** に設定すると、Red Hat Single Sign-On アダプターは、Cross-Origin Resource Sharing (CORS) を有効にします。

Red Hat Single Sign-On クライアントが SAML プロトコルを使用する場合は、以下の追加変数を設定する必要があります。

変数	説明
SSO_SAML_KEYSTORE_SECRET	SAML キーストアへのアクセスに使用するシークレット。デフォルトは **sso-app-secret** です。
SSO_SAML_KEYSTORE	SAML キーストアシークレットのキーストアファイル名。デフォルトは **keystore.jks** です。
SSO_SAML_KEYSTORE_PASSWORD	SAML のキーストアパスワード。デフォルトは **mykeystorepass** です。
SSO_SAML_CERTIFICATE_NAME	SAML に使用するキー/証明書のエイリアス。デフォルトは **jboss** です。

OpenID-Connect クライアントを使用してクライアントの自動登録方法のエンドツーエンドの例は、「ワークフローの例: OpenID-Connect クライアントを使用した Red Hat Single Sign-On での EAP アプリケーションの自動登録」を参照してください。

3.8.1.2. Red Hat Single Sign-On Client の手動登録

Red Hat Single Sign-On におけるクライアントの手動登録は、クライアントアプリケーションの **../configuration/** ディレクトリーにおけるデプロイメントファイルの存在によって判断されます。これらのファイルは、Red Hat Single Sign-On Web コンソールのクライアントアダプターからエクスポートされます。このファイルの名前は、OpenID-Connect および SAML クライアントの場合とは異なります。

openid-Connect	../configuration/secure-deployments
SAML	../configuration/secure-saml-deployments

これらのファイルは、アプリケーションのデプロイ時に **standalone-openshift.xml** の Red Hat Single Sign-On アダプター構成セクションにコピーされます。

Red Hat Single Sign-On アダプター設定をクライアントアプリケーションに渡す方法は 2 つあります。

- デプロイメントファイルを変更して Red Hat Single Sign-On アダプター設定が含まれるように修正し、デプロイメント時に **standalone-openshift.xml** ファイルに含まれるようにします。
- クライアントアプリケーションの **../WEB-INF** ディレクトリーに OpenID-Connect の **keycloak.json** ファイル、または SAML **keycloak-saml.xml** ファイルを手動で追加します。

SAML クライアントを使用した手動の Red Hat Single Sign-On クライアント登録方法のエンドツーエンドの例は、「ワークフローのサンプル: SAML クライアントを使用して、Red Hat Single Sign-On 認証を使用するアプリケーションを手動で設定」を参照してください。

3.9. OPENSHIFT シークレットでの RED HAT SINGLE SIGN-ON VAULT の使用

Red Hat Single Sign-On 管理における複数のフィールドでは、外部 vault からシークレットの値の取得をサポートします。『サーバー管理 ガイド』 を参照してください。以下の例は、OpenShift にプレーンテキスト Vault ファイルベースを設定し、SMTP パスワードを取得するように設定する方法を示しています。

手順

1. **SSO_VAULT_DIR** 環境変数を使用して Vault のディレクトリーを指定します。デプロイメント設定の環境に **SSO_VAULT_DIR** 環境変数を直接導入することができます。また、テンプレート内の適切な場所に以下のスニペットを追加して、テンプレートに追加することもできます。

```
"parameters": [
    ...
    {
        "displayName": "RH-SSO Vault Secret directory",
        "description": "Path to the RH-SSO Vault directory.",
        "name": "SSO_VAULT_DIR",
        "value": "",
        "required": false
    }
    ...
]

env: [
    ...
    {
        "name": "SSO_VAULT_DIR",
        "value": "${SSO_VAULT_DIR}"
    }
    ...
]
```

注記

ファイルのプレーンテキスト Vault プロバイダーは、**SSO_VAULT_DIR** 環境変数を設定した場合のみ構成されます。

2. OpenShift クラスターにシークレットを作成します。

```
$ oc create secret generic rhsso-vault-secrets --from-literal=master_smtp-password=mySMTPPsswd
```

3. **${SSO_VAULT_DIR}** をパスとして使用して、デプロイメント設定にボリュームをマウントします。すでに実行中のデプロイメントの場合:

```
oc set volume dc/sso --add --mount-path=${SSO_VAULT_DIR} --secret-name=rhsso-vault-secrets
```

4. Pod の作成後に、Red Hat Single Sign-On 設定内でカスタマイズされた文字列を使用してシークレットを参照できます。たとえば、このチュートリアルで作成した **mySMTPPsswd** シークレットを使用する場合は、smtp パスワードの設定で **master** レルム内の **${vault.smtp-password}** を使用し、使用すると **mySMTPPsswd** に置き換えられます。

3.10. 制限

現時点で、OpenShift は外部プロバイダーからの OpenShift のロールマッピングを受け入れません。Red Hat Single Sign-On を OpenShift の認証ゲートウェイとして使用する場合は、Red Hat Single Sign-On で作成したユーザーに、OpenShift 管理者の **oc adm policy** コマンドを使用してロールを追加する必要があります。

たとえば、Red Hat Single Sign-On で作成したユーザーが OpenShift でプロジェクトの namespace を表示できるようにするには、以下を実行します。

```
$ oc adm policy add-role-to-user view <user-name> -n <project-name>
```

第4章 チュートリアル

この章のチュートリアルは、OpenShift Container Platform クラスターのインストール を実行して作成されたものと同様の OpenShift インスタンスがあることを前提としています。

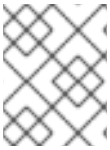
4.1. OPENSHIFT イメージバージョンの新しい RED HAT SINGLE SIGN-ON 用のデータベースの更新

更新に関連する次の点に注意してください。

- データベースおよびキャッシュは後方互換性がないため、以前のバージョンの Red Hat Single Sign-On for OpenShift からバージョン 7.6.0 へのローリングアップデートはサポートされません。
- Red Hat Single Sign-On for OpenShift のバージョンのインスタンスは、アップグレード前に実行できません。同じデータベースに対して同時に実行することはできません。
- 事前に生成されたスクリプトは利用できません。それらはデータベースに応じて動的に生成されます。

データベースを更新するには、次の 2 つの選択肢があります。

- Red Hat Single Sign-On 7.6.0 が データベーススキーマを自動的に移行できるようにします。
- データベースを 手動で 更新する



注記

デフォルトでは、Red Hat Single Sign-On 7.6.0 を初めて起動すると、データベースは自動的に移行されます。

4.1.1. データベースの自動移行

このプロセスは、PostgreSQL データベース (一時的または永続的モードでデプロイ) がサポートする Red Hat Single Sign-On for OpenShift イメージの以前のバージョンを別の Pod で実行することを前提としています。

前提条件

- Preparing Red Hat Single Sign-On Authentication for OpenShift Deployment で説明されている手順を実行します。

手順

データベーススキーマを自動的に移行するには、以下の手順に従います。

1. Red Hat Single Sign-On for OpenShift イメージの以前のバージョンを実行して、コンテナーをデプロイするために使用されるデプロイメント設定を特定します。

```
$ oc get dc -o name --selector=application=sso
deploymentconfig/sso
deploymentconfig/sso-postgresql
```

2. 現在の名前空間で、以前のバージョンの Red Hat Single Sign-On for OpenShift イメージを実行するすべての Pod を停止します。同じデータベースに対して同時に実行することはできません。

```
$ oc scale --replicas=0 dc/sso
deploymentconfig "sso" scaled
```

3. 既存のデプロイメント設定のイメージ変更トリガーを更新して、Red Hat Single Sign-On 7.6.0 イメージを参照します。

```
$ oc patch dc/sso --type=json -p '[{"op": "replace", "path":
"/spec/triggers/0/imageChangeParams/from/name", "value": "sso76-openshift-rhel8:7.6"}]'
"sso" patched
```

4. イメージ変更トリガーで定義した最新のイメージに基づいて、新しい Red Hat Single Sign-On 7.6.0 イメージのロールアウトを開始します。

```
$ oc rollout latest dc/sso
deploymentconfig "sso" rolled out
```

5. 変更したデプロイメント設定を使用して、Red Hat Single Sign-On 7.6.0 コンテナーをデプロイします。

```
$ oc scale --replicas=1 dc/sso
deploymentconfig "sso" scaled
```

6. (必要に応じて) データベースが正常に更新されたことを確認します。

```
$ oc get pods --selector=application=sso
NAME                  READY     STATUS    RESTARTS   AGE
sso-4-vg21r           1/1       Running   0          1h
sso-postgresql-1-t871r   1/1       Running   0          2h
```

```
$ oc logs sso-4-vg21r | grep 'Updating'
11:23:45,160 INFO
[org.keycloak.connections.jpa.updater.liquibase.LiquibaseJpaUpdaterProvider]
(ServerService Thread Pool -- 58) Updating database. Using changelog META-INF/jpa-
changelog-master.xml
```

4.1.2. データベースの手動移行

データベース移行プロセスは、データスキーマを更新し、データの操作を実行します。このプロセスは、SQL 移行ファイルを動的に生成する前に、Red Hat Single Sign-On for OpenShift イメージの以前のバージョンを実行しているすべての Pod も停止します。

注記

このプロセスは、PostgreSQL データベース (一時的または永続的モードでデプロイ) がサポートする Red Hat Single Sign-On for OpenShift イメージの以前のバージョンを別の Pod で実行することを前提としています。

手順

スクリプト生成のために環境を準備します。

1. 適切なデータソースで Red Hat Single Sign-On 7.6.0 を設定します。
2. **standalone-openshift.xml** ファイルに以下の設定オプションを設定します。
   a. **initializeEmpty=false**
   b. **migrationStrategy=manual**
   c. **migrationExport** は、出力 SQL 移行ファイルが保存される Pod のファイルシステム上の場所に移行します (例: **migrationExport="${jboss.home.dir}/keycloak-database-update.sql"**)。

関連情報

- データベースの設定

手順

次のコマンドを実行して、データベースの SQL 移行ファイルを生成します。

1. OpenShift データベース移行ジョブ のテンプレートを準備し、SQL ファイルを生成します。

```
$ cat job-to-migrate-db-to-sso76.yaml.orig
apiVersion: batch/v1
kind: Job
metadata:
  name: job-to-migrate-db-to-sso76
spec:
  autoSelector: true
  parallelism: 0
  completions: 1
  template:
    metadata:
      name: job-to-migrate-db-to-sso76
    spec:
      containers:
      - env:
        - name: DB_SERVICE_PREFIX_MAPPING
          value: <<DB_SERVICE_PREFIX_MAPPING_VALUE>>
        - name: <<PREFIX>>_JNDI
          value: <<PREFIX_JNDI_VALUE>>
        - name: <<PREFIX>>_USERNAME
          value: <<PREFIX_USERNAME_VALUE>>
        - name: <<PREFIX>>_PASSWORD
          value: <<PREFIX_PASSWORD_VALUE>>
        - name: <<PREFIX>>_DATABASE
          value: <<PREFIX_DATABASE_VALUE>>
        - name: TX_DATABASE_PREFIX_MAPPING
          value: <<TX_DATABASE_PREFIX_MAPPING_VALUE>>
        - name: <<SERVICE_HOST>>
          value: <<SERVICE_HOST_VALUE>>
        - name: <<SERVICE_PORT>>
          value: <<SERVICE_PORT_VALUE>>
        image: <<SSO_IMAGE_VALUE>>
        imagePullPolicy: Always
```

```
      name: job-to-migrate-db-to-sso76
      # Keep the pod running after the SQL migration
      # file was generated, so we can retrieve it
      command:
        - "/bin/bash"
        - "-c"
        - "/opt/eap/bin/openshift-launch.sh || sleep 600"
    restartPolicy: Never
```

```
$ cp job-to-migrate-db-to-sso76.yaml.orig \
    job-to-migrate-db-to-sso76.yaml
```

2. 以前のバージョンの Red Hat Single Sign-On for OpenShift イメージを実行するのに使用されるデプロイメント設定から、データソース定義およびデータベースアクセス認証情報をデータベース移行ジョブのテンプレートの適切な場所にコピーします。
以下のスクリプトを使用して、sso という名前のデプロイメント設定から、**job-to-migrate-db-to- sso 76.yaml** という名前のデータベースジョブ移行テンプレートに **DB _SERVICE_PREFIX_MAPPING変数および TX_DATABASE_ PREFIX_MAPPING** 変数の値をコピーします。

注記

DB_SERVICE_PREFIX_MAPPING 環境変数では、<name>-<database_type>=<PREFIX> トリプレットのコンマ区切りリスト を値として指定することができますが、この例のスクリプトでは、デモのために 1 つのデータソースのトリプレット定義のみを受け付けます複数のデータソース定義トリプレットを処理するスクリプトを変更できます。

```
$ cat mirror_sso_dc_db_vars.sh
#!/bin/bash

# IMPORTANT:
#
# If the name of the SSO deployment config differs from 'sso'
# or if the file name of the YAML definition of the migration
# job is different, update the following two variables
SSO_DC_NAME="sso"
JOB_MIGRATION_YAML="job-to-migrate-db-to-sso76.yaml"

# Get existing variables of the $SSO_DC_NAME deployment config
# in an array
declare -a SSO_DC_VARS=( \
  $(oc set env dc/${SSO_DC_NAME} --list \
  | sed '/^#/d') \
)

# Get the PREFIX used in the names of environment variables
PREFIX=$( \
  grep -oP 'DB_SERVICE_PREFIX_MAPPING=[^ ]+' \
  <<< "${SSO_DC_VARS[@]}" \
)
```

```
PREFIX=${PREFIX##*=}

# Substitute:
# * <<PREFIX>> with actual $PREFIX value and
# * <<PREFIX with "<<$PREFIX" value
# The order in which these replacements are made is important!
sed -i "s#<<PREFIX>>#${PREFIX}#g" ${JOB_MIGRATION_YAML}
sed -i "s#<<PREFIX#<<${PREFIX}#g" ${JOB_MIGRATION_YAML}

# Construct the array of environment variables
# specific to the datasource
declare -a DB_VARS=(JNDI USERNAME PASSWORD DATABASE)

# Prepend $PREFIX to each item of the datasource array
DB_VARS=( "${DB_VARS[@]/#/${PREFIX}_}" )

# Add DB_SERVICE_PREFIX_MAPPING and TX_DATABASE_PREFIX_MAPPING
# variables to datasource array
DB_VARS=( \
  "${DB_VARS[@]}" \
  DB_SERVICE_PREFIX_MAPPING \
  TX_DATABASE_PREFIX_MAPPING \
)

# Construct the SERVICE from DB_SERVICE_PREFIX_MAPPING
SERVICE=$( \
  grep -oP 'DB_SERVICE_PREFIX_MAPPING=[^ ]' \
  <<< "${SSO_DC_VARS[@]}" \
)
SERVICE=${SERVICE#*=}
SERVICE=${SERVICE%=*}
SERVICE=${SERVICE^^}
SERVICE=${SERVICE//-/_}

# If the deployment config contains <<SERVICE>>_SERVICE_HOST
# and <<SERVICE>>_SERVICE_PORT variables, add them to the
# datasource array. Their values also need to be propagated into
# yaml definition of the migration job.
HOST_PATTERN="${SERVICE}_SERVICE_HOST=[^ ]"
PORT_PATTERN="${SERVICE}_SERVICE_PORT=[^ ]"
if
  grep -Pq "${HOST_PATTERN}" <<< "${SSO_DC_VARS[@]}" &&
  grep -Pq "${PORT_PATTERN}" <<< "${SSO_DC_VARS[@]}"
then
  DB_VARS=( \
    "${DB_VARS[@]}" \
    "${SERVICE}_SERVICE_HOST" \
    "${SERVICE}_SERVICE_PORT" \
  )
# If they are not defined, delete their placeholder rows in
# yaml definition file (since if not defined they are not
# expanded which make the yaml definition invalid).
else
  for KEY in "HOST" "PORT"
  do
    sed -i "/SERVICE_${KEY}/d" ${JOB_MIGRATION_YAML}
```

```
  done
fi

# Substitute:
# * <<SERVICE_HOST>> with ${SERVICE}_SERVICE_HOST and
# * <<SERVICE_HOST_VALUE>> with <<${SERVICE}_SERVICE_HOST_VALUE>>
# The order in which replacements are made is important!
# Do this for both "HOST" and "PORT"
for KEY in "HOST" "PORT"
do
  PATTERN_1="<<SERVICE_${KEY}>>"
  REPL_1="${SERVICE}_SERVICE_${KEY}"
  sed -i "s#${PATTERN_1}#${REPL_1}#g" ${JOB_MIGRATION_YAML}
  PATTERN_2="<<SERVICE_${KEY}_VALUE>>"
  REPL_2="<<${SERVICE}_SERVICE_${KEY}_VALUE>>"
  sed -i "s#${PATTERN_2}#${REPL_2}#g" ${JOB_MIGRATION_YAML}
done

# Propagate the values of the datasource array items into
# yaml definition of the migration job
for VAR in "${SSO_DC_VARS[@]}"
do
  IFS=$'=' read KEY VALUE <<< $VAR
  if grep -q $KEY <<< ${DB_VARS[@]}
  then
    KEY+="_VALUE"
    # Enwrap integer port value with double quotes
    if [[ ${KEY} =~ ${SERVICE}_SERVICE_PORT_VALUE ]]
    then
      sed -i "s#<<${KEY}>>#\"${VALUE}\"#g" ${JOB_MIGRATION_YAML}
    # Character values do not need quotes
    else
      sed -i "s#<<${KEY}>>#${VALUE}#g" ${JOB_MIGRATION_YAML}
    fi
    # Verify that the value has been successfully propagated.
    if
      grep -q '(JNDI|USERNAME|PASSWORD|DATABASE)' <<< "${KEY}" &&
      grep -q "<<PREFIX${KEY#${PREFIX}}" ${JOB_MIGRATION_YAML} ||
      grep -q "<<${KEY}>>" ${JOB_MIGRATION_YAML}
    then
      echo "Failed to update value of ${KEY%_VALUE}! Aborting."
      exit 1
    else
      printf '%-60s%-40s\n' \
            "Successfully updated ${KEY%_VALUE} to:" \
            "$VALUE"
    fi
  fi
done
```

スクリプトを実行します。

```
$ chmod +x ./mirror_sso_dc_db_vars.sh
$ ./mirror_sso_dc_db_vars.sh
```

```
Successfully updated DB_SERVICE_PREFIX_MAPPING to:        sso-postgresql=DB
Successfully updated DB_JNDI to:                          java:jboss/datasources/KeycloakDS
Successfully updated DB_USERNAME to:                      userxOp
Successfully updated DB_PASSWORD to:                      tsWNhQHK
Successfully updated DB_DATABASE to:                      root
Successfully updated TX_DATABASE_PREFIX_MAPPING to:       sso-postgresql=DB
```

3. 事前設定されたソース を使用して Red Hat Single Sign-On 7.6.0 データベース移行イメージをビルドし、ビルドが完了するまで待ちます。

```
$ oc get is -n openshift | grep sso76 | cut -d ' ' -f1
sso76-openshift-rhel8
```

```
$ oc new-build sso76-openshift-rhel8:7.6~https://github.com/iankko/openshift-examples.git#KEYCLOAK-8500 \
  --context-dir=sso-manual-db-migration \
  --name=sso76-db-migration-image
--> Found image bf45ac2 (7 days old) in image stream "openshift/sso76-openshift-rhel8" under tag "7.6" for "sso76-openshift-rhel8:7.6"

    Red Hat SSO 7.6.0
    -----------------
    Platform for running Red Hat SSO

    Tags: sso, sso7, keycloak

    * A source build using source code from https://github.com/iankko/openshift-examples.git#KEYCLOAK-8500 will be created
      * The resulting image will be pushed to image stream "sso76-db-migration-image:latest"
      * Use 'start-build' to trigger a new build

--> Creating resources with label build=sso76-db-migration-image ...
    imagestream "sso76-db-migration-image" created
    buildconfig "sso76-db-migration-image" created
--> Success
    Build configuration "sso76-db-migration-image" created and build triggered.
    Run 'oc logs -f bc/sso76-db-migration-image' to stream the build progress.
```

```
$ oc logs -f bc/sso76-db-migration-image --follow
Cloning "https://github.com/iankko/openshift-examples.git#KEYCLOAK-8500" ...
...
Push successful
```

4. データベース移行ジョブのテンプレート(job-to-migrate-db-to-sso76.yaml)をビルドされた sso76- db-migration-image イメージ への参照で更新します。

   a. イメージの docker pull 参照を取得します。

```
$ PULL_REF=$(oc get istag -n $(oc project -q) --no-headers | grep sso76-db-migration-image | tr -s ' ' | cut -d ' ' -f 2)
```

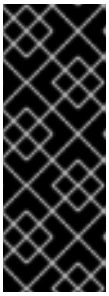
b. ジョブテンプレートの <<SSO_IMAGE_VALUE>> フィールドを、プル仕様に置き換えます。

```
$ sed -i "s#<<SSO_IMAGE_VALUE>>#$PULL_REF#g" job-to-migrate-db-to-sso76.yaml
```

c. フィールドが更新されていることを確認します。

5. ジョブテンプレートからデータベース移行ジョブをインスタンス化します。

```
$ oc create -f job-to-migrate-db-to-sso76.yaml
job "job-to-migrate-db-to-sso76" created
```



重要

データベースの移行プロセスでは、データスキーマの更新を処理し、データの操作を実行します。したがって、SQL 移行ファイルを動的に生成する前に、以前のバージョンの Red Hat Single Sign-On for OpenShift イメージを実行するすべての Pod を停止します。

6. Red Hat Single Sign-On for OpenShift イメージの以前のバージョンを実行して、コンテナーをデプロイするために使用されるデプロイメント設定を特定します。

```
$ oc get dc -o name --selector=application=sso
deploymentconfig/sso
deploymentconfig/sso-postgresql
```

7. 現在の名前空間で、以前のバージョンの Red Hat Single Sign-On for OpenShift イメージを実行するすべての Pod を停止します。

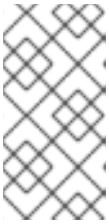
```
$ oc scale --replicas=0 dc/sso
deploymentconfig "sso" scaled
```

8. データベースの移行ジョブを実行し、Pod が正しく実行されるのを待機します。

```
$ oc get jobs
NAME                          DESIRED   SUCCESSFUL   AGE
job-to-migrate-db-to-sso76   1         0            3m
```

```
$ oc scale --replicas=1 job/job-to-migrate-db-to-sso76
job "job-to-migrate-db-to-sso76" scaled
```

```
$ oc get pods
NAME                              READY     STATUS      RESTARTS   AGE
sso-postgresql-1-n5p16            1/1       Running     1          19h
job-to-migrate-db-to-sso76-b87bb   1/1       Running     0          1m
sso76-db-migration-image-1-build      0/1       Completed   0          27m
```



注記

デフォルトでは、データベース移行ジョブは、移行ファイルの生成後 600 秒 後に自動的に終了します。この期間を調整できます。

9. Pod から動的に生成された SQL データベース移行ファイルを取得します。

```
$ mkdir -p ./db-update
$ oc rsync job-to-migrate-db-to-sso76-b87bb:/opt/eap/keycloak-database-update.sql
./db-update
receiving incremental file list
keycloak-database-update.sql

sent 30 bytes  received 29,726 bytes  59,512.00 bytes/sec
total size is 29,621  speedup is 1.00
```

10. keycloak-database-update.sql ファイルを検査して、Red Hat Single Sign-On 7.6.0 バージョンへの手動データベース更新内で実行される変更を確認します。

11. データベースの更新を手動で適用します。

- PostgreSQL データベース (一時的または永続的モードでデプロイ) がサポートする Red Hat Single Sign-On for OpenShift イメージの以前のバージョンを別の Pod で実行する場合は、以下のコマンドを実行します。

  i. 生成された SQL 移行ファイルを PostgreSQL Pod にコピーします。

```
$ oc rsync --no-perms=true ./db-update/ sso-postgresql-1-n5p16:/tmp
sending incremental file list
```

```
sent 77 bytes  received 11 bytes  176.00 bytes/sec
total size is 26,333  speedup is 299.24
```

ii. PostgreSQL Pod へのシェルセッションを開始します。

```
$ oc rsh sso-postgresql-1-n5p16
sh-4.2$
```

iii. psql ツールを使用して、データベースの更新を手動で適用します。

```
sh-4.2$ alias psql="/opt/rh/rh-postgresql95/root/bin/psql"
sh-4.2$ psql --version
psql (PostgreSQL) 9.5.4
sh-4.2$ psql -U <PREFIX>_USERNAME -d <PREFIX>_DATABASE -W -f
/tmp/keycloak-database-update.sql
Password for user <PREFIX>_USERNAME:
INSERT 0 1
INSERT 0 1
...
```

重要

<PREFIX>_USERNAME および <PREFIX>_DATABASE を、前のセクション で取得した実際のデータベースの認証情報に置き換えます。また、プロンプトが表示されたら、データベースのパスワードとして <PREFIX>_PASSWORD の値を使用します。

iv. PostgreSQL Pod へのシェルセッションを閉じます。「イメージ変更トリガーの更新手順」に進みます。

12. 既存のデプロイメント設定のイメージ変更トリガーを更新して、Red Hat Single Sign-On 7.6.0 イメージを参照します。

```
$ oc patch dc/sso --type=json -p '[{"op": "replace", "path":
"/spec/triggers/0/imageChangeParams/from/name", "value": "sso76-openshift-
rhel8:7.6"}]'
"sso" patched
```

13. イメージ変更トリガーで定義した最新のイメージに基づいて、新しい Red Hat Single Sign-On 7.6.0 イメージのロールアウトを開始します。

```
$ oc rollout latest dc/sso
deploymentconfig "sso" rolled out
```

14. 変更したデプロイメント設定を使用して、Red Hat Single Sign-On 7.6.0 コンテナーをデプロイします。

```
$ oc scale --replicas=1 dc/sso
deploymentconfig "sso" scaled
```

4.2. 環境間での RED HAT SINGLE SIGN-ON サーバーのデータベースの移行

このチュートリアルでは、Red Hat Single Sign-On サーバーデータベースをある環境から別の環境への移行、または別のデータベースへの移行に重点を置いています。

Red Hat Single Sign-On 7.6.0 データベースのエクスポートおよびインポートは、Red Hat Single Sign-On サーバーの起動時にトリガーされ、そのパラメーターは Java システムプロパティーを介して渡されます。これは、1 回の Red Hat Single Sign-On サーバーの起動中に、可能な移行アクションの 1 つ (エクスポート または インポート) のみが実行されることを意味します。

4.2.1. Red Hat Single Sign-On PostgreSQL アプリケーションテンプレートのデプロイ

前提条件

- Preparing Red Hat Single Sign-On Authentication for OpenShift Deployment で説明されている手順を前提としています。

手順

1. OpenShift Web コンソールにログインし、sso-app-demo プロジェクトスペースを選択します。
2. Add to project をクリックし、デフォルトのイメージストリームおよびテンプレートを一覧表示します。
3. Filter by keyword 検索バーを使用して、一覧を sso に一致するものに制限します。See all をクリックして、必要なアプリケーションテンプレートを表示する必要がある場合があります。
4.

sso76-postgresql Red Hat Single Sign-On アプリケーションテンプレートを選択します。テンプレートをデプロイする際には、必ず SSO_REALM 変数が設定されていない (デフォルト値) ことを確認してください。

警告

Red Hat Single Sign-On for OpenShift イメージに SSO_REALM 設定変数が設定されていると、変数で要求される Red Hat Single Sign-On サーバーレルムを作成するためにデータベースのインポートが実行されます。データベースのエクスポートを正しく実行するには、そのイメージに SSO_REALM 設定変数を同時に定義することはできません。

5. Create をクリックしてアプリケーションテンプレートをデプロイし、Pod デプロイメントを開始します。これには数分の時間がかかる場合があります。

その後、管理者アカウント を使用して https://secure-sso-<sso-app-demo>.<openshift32.example.com>/auth/admin で Red Hat Single Sign-On の Web コンソールにアクセスします。

注記

このワークフロー例では、自己生成された CA を使用して、デモ目的でエンドツーエンドのワークフローを提供します。Red Hat Single Sign-On Web コンソールにアクセスすると、セキュリティーが保護されていない接続に関する警告が表示されます。
実稼働環境では、検証された認証局から購入した SSL 証明書を使用することを推奨します。

関連情報

- データベースのインポートおよびエクスポート

4.2.2. (オプション) エクスポートする追加のレルムとユーザーの作成

Red Hat Single Sign-On 7.6.0 サーバーデータベースのエクスポートを実行すると、現在データベースにあるレルムおよびユーザーのみがエクスポートされます。エクスポートした JSON ファイルに追加の Red Hat Single Sign-On レルムとユーザーも含まれる場合は、以下を作成する必要があります。これらの手順を使用してください。

1. レルムを作成します。

2. ユーザーを作成します。

これが作成されると、データベースをエクスポートできます。

関連情報

- データベースのインポートおよびエクスポート

4.2.3. OpenShift Pod 上の JSON ファイルとして Red Hat Single Sign-On データベースをエクスポートします。

前提条件

- 新しいレルムとユーザーが作成されます。

手順

1. Red Hat Single Sign-On デプロイメント設定を取得し、これをゼロにスケールダウンします。

```
$ oc get dc -o name
deploymentconfig/sso
deploymentconfig/sso-postgresql

$ oc scale --replicas=0 dc sso
deploymentconfig "sso" scaled
```

2. Red Hat Single Sign-On for OpenShift イメージにデプロイされた Red Hat Single Sign-On 7.6.0 サーバーに対して、Red Hat Single Sign-On サーバーの起動時にデータベースのエクスポートを実行するように指示します。

```
$ oc set env dc/sso \
  -e "JAVA_OPTS_APPEND= \
    -Dkeycloak.migration.action=export \
    -Dkeycloak.migration.provider=singleFile \
    -Dkeycloak.migration.file=/tmp/demorealm-export.json"
```

3.

Red Hat Single Sign-On デプロイメント設定のバックアップを拡張します。これにより、Red Hat Single Sign-On サーバーが起動し、そのデータベースをエクスポートします。

```
$ oc scale --replicas=1 dc sso
deploymentconfig "sso" scaled
```

4. (必要に応じて) エクスポートが成功したことを確認します。

```
$ oc get pods
NAME                 READY    STATUS    RESTARTS   AGE
sso-4-ejr0k          1/1      Running   0          27m
sso-postgresql-1-ozzl0   1/1      Running   0          4h

$ oc logs sso-4-ejr0k | grep 'Export'
09:24:59,503 INFO  [org.keycloak.exportimport.singlefile.SingleFileExportProvider]
(ServerService Thread Pool -- 57) Exporting model into file /tmp/demorealm-
export.json
09:24:59,998 INFO  [org.keycloak.services] (ServerService Thread Pool -- 57) KC-
SERVICES0035: Export finished successfully
```

4.2.4. エクスポートした JSON ファイルを取得してインポートします。

手順

1. Pod から Red Hat Single Sign-On データベースの JSON ファイルを取得します。

```
$ oc get pods
NAME                 READY    STATUS    RESTARTS   AGE
sso-4-ejr0k          1/1      Running   0          2m
sso-postgresql-1-ozzl0   1/1      Running   0          4h

$ oc rsync sso-4-ejr0k:/tmp/demorealm-export.json .
```

2. (必要に応じて) Red Hat Single Sign-On データベースの JSON ファイルを、別の環境で実行している Red Hat Single Sign-On サーバーにインポートします。

注記

OpenShift 上で実行していない Red Hat Single Sign-On サーバーにインポートする場合は、「データベースのインポートおよびエクスポート」を参照してください。

Red Hat Single Sign-On サーバーが OpenShift で Red Hat Single Sign-On 7.6.0 コンテ

ナーとして実行されている場合、管理コンソールの Export/Import 関数を使用して、以前にエクスポートした JSON ファイルから Red Hat Single Sign-On サーバーのデータベースにリソースをインポートします。

a. 管理者ユーザーの作成に使用する認証情報を使用して、Red Hat Single Sign-On サーバーの master レルムの管理コンソールにログインします。ブラウザーで、Red Hat Single Sign-On Web サーバーの場合は http://sso-<project-name>.<hostname>/auth/admin に移動し、暗号化された Red Hat Single Sign-On Web サーバーの場合は https://secure-sso-<project-name>.<hostname>/auth/admin に移動します。

b. サイドバーの上部に、Red Hat Single Sign-On レルム名、ユーザー、クライアント、レルムロール、およびクライアントロールのインポート先を選択します。この例では、master レルムを使用します。

c. サイドバーの下部にある Manage セクションにある Import のリンクをクリックします。

d. 開いたページで Select file をクリックし、ローカルファイルシステムでエクスポートされた JSON ファイル demorealm-export.json の場所を指定します。

e. Import from realm ドロップダウンメニューから、データのインポート元となる Red Hat Single Sign-On レルムの名前を選択します。この例では、master レルムを使用します。

f. インポートするユーザー、クライアント、レルムロール、およびクライアントロールを選択します (すべてのユーザーはデフォルトでインポートされます)。

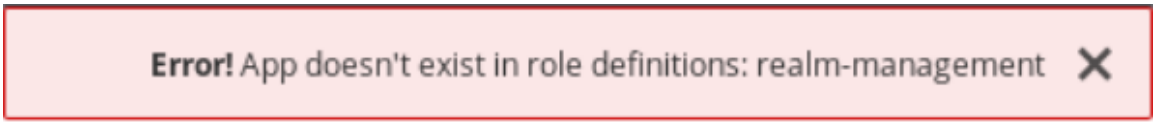
g. リソースがすでに存在する場合 (Fail、Skip、または Overwriteのいずれか) に実行するストラテジーを選択します。

注記

オブジェクト (ユーザー、クライアント、レルムロール、またはクライアントロール) をインポートしようとすると、現在のデータベースに同じ識別子を持つオブジェクトがすでに存在する場合に失敗します。Skip ストラテジーを使用して、demorealm-export.json ファイルにあるが、現在のデータベースに存在しないオブジェクトをインポートします。

h. インポート をクリックしてインポートを実行します。

Import ボタンをクリックすると、master レルム以外のレルムから master レルムにオブジェクトをインポートする場合や、その逆の場合、以下のようなエラーが発生することがあります。



この場合、まず、Access Type を bearer-only に設定して、不足しているクライアントを作成する必要があります。これらのクライアントは、エクスポート JSON ファイルが作成されたソースの Red Hat Single Sign-On サーバーから、JSON ファイルがインポートされるターゲット Red Hat Single Sign-On サーバーに、その特性を手動でコピーして作成できます。必要なクライアントを作成したら、再度 Import ボタンをクリックします。

上記の エラーメッセージを非表示にするには、Access Type が bearer-only の不足している realm-management クライアントを作成して、Import ボタンを再度クリックする必要があります。

Skip インポートストラテジーでは、新たに追加されたオブジェクトは ADDED としてマークされ、スキップされたオブジェクトが、インポート結果ページの Action 列の SKIPPED としてマークされます。

管理コンソールのインポートにより、選択した場合にリソースを 上書き できます (Overwrite ストラテジー)。実稼働システムでは、この機能を注意して使用します。

4.3. 認証に RED HAT SINGLE SIGN-ON を使用するための OPENSHIFT3.11 の設定

OpenShift 3.11 を、OpenShift の承認ゲートウェイとして Red Hat Single Sign-On デプロイメントを使用するように設定します。

この例では、OpenShift Container Platform クラスターのインストール時に設定された ID プロバイダーとともに認証方法として Red Hat Single Sign-On を追加します。設定が完了すると、ユーザーが OpenShift Web コンソールにログインできるように Red Hat Single Sign-On メソッドも (設定されたアイデンティティープロバイダーとともに) 利用できます。

関連情報

- アイデンティティープロバイダー
- OpenShift Container Platform クラスターのインストール

4.3.1. Red Hat Single Sign-On の認証情報の設定

前提条件

- Preparing Red Hat Single Sign-On Authentication for OpenShift Deployment で説明されている手順を前提としています。

手順

Red Hat Single Sign-On デプロイメント時に作成された xref:sso-administrator-setup 管理者アカウントを使用して、https://secure-sso-sso-app-demo.openshift32.example.com/auth/admin で暗号化された Red Hat Single Sign-On Web サーバーにログインします。

レルムの作成

1. カーソルをサイドバーの上部にあるレルム名前空間 (デフォルトは Master) の上に置き、Add Realm をクリックします。
2. レルム名 (この例では OpenShiftを使用) を入力し、Create をクリックします。

ユーザーの作成

Red Hat Single Sign-On が有効な OpenShift ログインを実証するのに使用できるテストユーザーを作成します。

1. Manage サイドバーで Users をクリックし、レルムのユーザー情報を表示します。
2. Add User をクリックします。

3. 有効な ユーザー名 (この例では testuser を使用し) と任意の追加情報を入力し、Save をクリックします。

4. ユーザー設定を編集します。

   a. ユーザー空間の Credentials タブをクリックし、ユーザーのパスワードを入力します。

   b. Temporary Password オプションを Off に設定して、後でパスワードの変更を要求しないようにし、Reset Password をクリックしてユーザーパスワードを設定します。ポップアップウィンドウに追加の確認を求めるプロンプトが表示されます。

OpenID-Connect クライアントの作成および設定

1. Manage サイドバーで Clients をクリックし、Create をクリックします。

2. Client ID を入力します。この例では openshift-demo を使用します。

3. ドロップダウンメニューから クライアントプロトコル を選択し (この例では openid-connectを使用)、Save をクリックします。openshift-demo クライアントの Settings ページに移動します。

4. Access Type ドロップダウンメニューから、confidential を選択します。これは、サーバー側のアプリケーションのアクセスタイプです。

5. Valid Redirect URIs ダイアログで、OpenShift Web コンソールの URI (この例では https://openshift.example.com:8443/*) を入力します。

次のセクションで、OpenShift マスターで OpenID-Connect を設定するには、クライアント シークレット が必要です。Credentials タブの下にコピーできます。この例では、シークレットは <7b0384a2-b832-16c5-9d73-2957842e89h7> です。

関連情報

- OpenID Connect および SAML クライアントの管理

4.3.2. Red Hat Single Sign-On 認証用の OpenShift マスターの設定

OpenShift マスター CLI にログインします。

前提条件

/etc/origin/master/master-config.yaml ファイルを編集するために必要なパーミッションが必要です。

手順

1. /etc/origin/master/master-config.yaml ファイルを編集し、identityProviders セクションを見つけます。たとえば、OpenShift マスターが HTPassword ID プロバイダー で設定されている場合、identityProviders セクションは以下のようになります。

```
identityProviders:
- challenge: true
  login: true
  name: htpasswd_auth
  provider:
    apiVersion: v1
    file: /etc/origin/openshift-passwd
    kind: HTPasswdPasswordIdentityProvider
```

以下のスニペットのような内容で、Red Hat Single Sign-On をセカンダリーアイデンティティープロバイダーとして追加します。

```
- name: rh_sso
  challenge: false
  login: true
  mappingMethod: add
  provider:
    apiVersion: v1
    kind: OpenIDIdentityProvider
    clientID: openshift-demo
    clientSecret: 7b0384a2-b832-16c5-9d73-2957842e89h7
    ca: xpaas.crt
    urls:
      authorize: https://secure-sso-sso-app-
demo.openshift32.example.com/auth/realms/OpenShift/protocol/openid-connect/auth
      token: https://secure-sso-sso-app-
demo.openshift32.example.com/auth/realms/OpenShift/protocol/openid-connect/token
      userInfo: https://secure-sso-sso-app-
demo.openshift32.example.com/auth/realms/OpenShift/protocol/openid-
```

```
connect/userinfo
  claims:
    id:
    - sub
    preferredUsername:
    - preferred_username
    name:
    - name
    email:
    - email
```

a. clientSecret の Red Hat Single Sign-On Secret ハッシュは、Red Hat Single Sign-On Web コンソールの Clients → openshift-demo → Credentials にあります。

b. urls のエンドポイントは、Red Hat Single Sign-On アプリケーションでリクエストを行うことで確認できます。以下に例を示します。

```
<curl -k https://secure-sso-sso-app-
demo.openshift32.example.com/auth/realms/OpenShift/.well-known/openid-
configuration | python -m json.tool>
```

応答には、authorization_endpoint、token_endpoint、および userinfo_endpoint が含まれます。

c. このワークフロー例では、自己生成された CA を使用して、デモ目的でエンドツーエンドのワークフローを提供します。このため、ca は <ca: xpaas.crt> として提供されます。この CA 証明書も /etc/origin/master ディレクトリーにコピーする必要があります。検証済みの認証局から購入した証明書を使用する場合、この作業は必要ありません。

2. 設定を保存して OpenShift マスターを再起動します。

```
$ systemctl restart atomic-openshift-master
```

4.3.3. OpenShift へのログイン

手順

1. OpenShift Web コンソールに移動します。この例では https://openshift.example.com:8443/console が使用されています。

OpenShift ログインページに、htpasswd_auth または rh-sso ID プロバイダーのいずれかを使用してログインするためのオプションが提供されるようになりました

か?/etc/origin/master/master-config.yaml に存在するため、以前のバージョンは引き続き利用できます。

2. rh-sso を選択し、Red Hat Single Sign-On に先に作成した testuser ユーザーを使用して OpenShift にログインします。

   OpenShift CLI に追加されるまで、testuser にはプロジェクトは表示されません。これは、現在外部ロールマッピングを受け入れないため、OpenShift のユーザー特権を提供する唯一の方法です。

3. sso-app-demo に testuser view 権限を提供するには、OpenShift CLI を使用します。

```
$ oc adm policy add-role-to-user view testuser -n sso-app-demo
```

4.4. MAVEN バイナリーからの OPENSHIFT アプリケーションの作成と、RED HAT SINGLE SIGN-ON を使用した保護

OpenShift に既存のアプリケーションをデプロイするには、バイナリーソースの機能を使用できます。

4.4.1. EAP 6.4 / 7.1 JSPサービス呼び出しアプリケーションのバイナリビルドをデプロイし、Red Hat Single Sign-On を使用して保護

以下の例は、app-jee-jsp および service-jee-jaxrs クイックスタートの両方を使用して、Red Hat Single Sign-On を使用して認証する EAP 6.4 / 7.1 JSP サービスアプリケーションをデプロイします。

前提条件

- Red Hat Single Sign-On for OpenShift イメージは、以前に以下のテンプレートのいずれかを使用してデプロイされています。
- sso76-postgresql
- sso76-postgresql-persistent
- sso76-x509-postgresql-persistent

4.4.1.1. EAP 6.4 / 7.1 JSP アプリケーションの Red Hat Single Sign-On レルム、ロール、およびユーザーの作成

EAP 6.4 / 7.1 JSP サービスアプリケーションには、Red Hat Single Sign-On を使用して認証できるように専用の Red Hat Single Sign-On レルム、ユーザー名、およびパスワードが必要です。Red Hat Single Sign-On for OpenShift イメージをデプロイした後に、以下の手順を実行します。

Red Hat Single Sign-On レルムの作成

1. Red Hat Single Sign-On サーバーの管理コンソールにログインします。

   https://secure-sso-sso-app-demo.openshift.example.com/auth/admin

   Red Hat Single Sign-On 管理者ユーザーの認証情報 を使用します。

2. カーソルをサイドバーの上部にあるレルム名前空間 (デフォルトは Master) の上に置き、Add Realm をクリックします。

3. レルム名を入力して (この例では demo を使用)、Create をクリックします。

公開鍵のコピー

新規に作成された demo レルムで Keys タブをクリックしてから Active タブを選択し、生成したタイプ RSA の公開鍵をコピーします。

注記

Red Hat Single Sign-On for OpenShift イメージバージョン 7.6.0 は、デフォルトで複数のキーを生成します（ HS256、RS256、AES など）。Red Hat Single Sign-On for OpenShift 7.6.0 イメージの公開鍵情報をコピーするには、Keys タブをクリックしてから Active タブを選択し、Keys テーブルでその行の Public key ボタンをクリックします。ここで、キーのタイプは RSA に一致します。次に、表示されるポップアップウィンドウの内容を選択し、コピーします。

Red Hat Single Sign-On 対応 EAP 6.4 / 7.1 JSP アプリケーションを 後でデプロイ するには、公開鍵に関する情報が必要になります。

Red Hat Single Sign-On のロールの作成

service-jee-jaxrs クイックスタートはサービスによって 3 つのエンドポイントを公開します。

- public - 認証は必要ありません。
- secured - user ロールを持つユーザーが呼び出すことができます。
- admin - admin ロールを持つユーザーが呼び出すことができます。

Red Hat Single Sign-On で user ロールおよび admin ロールを作成します。これらのロールは Red Hat Single Sign-On アプリケーションユーザーに割り当てられ、ユーザーアプリケーションへのアクセスを認証します。

1. Configure サイドバーの Roles をクリックし、このレルムのロールを一覧表示します。

注記

これは新しいレルムであるため、デフォルト (offline_access および uma_authorization) ロールのみが必要です。

2. Add Role をクリックします。
3. ロール名 (user) を入力し、Save をクリックします。

admin ロールについてこれらの手順を繰り返します。

Red Hat Single Sign-On レルム管理ユーザーの作成

1. Manage サイドバーで Users をクリックし、レルムのユーザー情報を表示します。

2. Add User をクリックします。

3. 有効な Username を入力し (この例ではユーザー appuser を使用)、Save をクリックします。

4. ユーザー設定を編集します。

   a. ユーザースペースの Credentials タブをクリックして、ユーザーのパスワードを入力します (この例ではパスワード apppassword を使用しています)。

   b. Temporary Password オプションを Off に設定して、後でパスワードの変更を要求しないようにし、Reset Password をクリックしてユーザーパスワードを設定します。ポップアップウィンドウが表示され、確認を求められます。

4.4.1.2. レルム管理ユーザーにユーザーロールを割り当てます

以前に作成した appuser を、Red Hat Single Sign-On ロール user と関連付けるには、以下の手順を実行します。

1. Role Mappings をクリックし、レルムおよびクライアントロール設定を一覧表示します。Available Roles で、以前に作成した user ロールを選択し、Add selected> をクリックします。

2. Client Roles をクリックし、一覧から realm-management エントリーを選択し、Available Roles のリストで各レコードを選択します。

注記

Ctrl キーを保持し、最初の 偽装 エントリーを同時にクリックすることで、複数の項目を一度に選択できます。Ctrl キーとマウスの左ボタンを押したままにしておくと、一覧の最後を view-clients エントリーに移動し、各レコードが選択されていることを確認します。

3. Add selected> をクリックしてロールをクライアントに割り当てます。

4.4.1.3. EAP 6.4 / 7.1 JSP アプリケーションの OpenShift デプロイメント向けの Red Hat Single Sign-On 認証の準備

手順

1. EAP 6.4 / 7.1 JSP アプリケーションの新しいプロジェクトを作成します。

```
$ oc new-project eap-app-demo
```

2. view ロールを デフォルト のサービスアカウントに追加します。これにより、サービスアカウントが eap-app-demo 名前空間のすべてのリソースを表示できるようになります。これは、クラスターの管理に必要です。

```
$ oc policy add-role-to-user view system:serviceaccount:$(oc project -q):default
```

3. EAP テンプレートには、SSL キーストアと JGroups キーストア が必要です。この例では、Java Development Kit に含まれるパッケージの keytool を使用して、これらのキーストアの自己署名証明書を生成します。

   a. SSL キーストアのセキュアなキーを生成します (この例では、キーストアのパスワードとして password を使用します)。

```
$ keytool -genkeypair \
-dname "CN=secure-eap-app-eap-app-demo.openshift.example.com" \
-alias https \
-storetype JKS \
-keystore eapkeystore.jks
```

   b. JGroups キーストアに、セキュリティーで保護されたキーを生成します (この例では、キーストアのパスワードとして password を使用します)。

```
$ keytool -genseckey \
-alias jgroups \
-storetype JCEKS \
-keystore eapjgroups.jceks
```

   c. SSL および JGroup キーストアファイルで、OpenShift シークレットの EAP 6.4 / 7.1 を生成します。

```
$ oc create secret generic eap-ssl-secret --from-file=eapkeystore.jks
```

```
$ oc create secret generic eap-jgroup-secret --from-file=eapjgroups.jceks
```

d. EAP アプリケーションシークレットを default サービスアカウントに追加します。

```
$ oc secrets link default eap-ssl-secret eap-jgroup-secret
```

4.4.1.4. EAP 6.4 / 7.1 JSP アプリケーションのバイナリービルドのデプロイ

手順

1. ソースコードのクローンを作成します。

```
$ git clone https://github.com/keycloak/keycloak-quickstarts.git
```

Red Hat JBoss Middleware Maven リポジトリー を設定します。

2. service-jee-jaxrs アプリケーションと app-jee-jsp アプリケーションの両方をビルドします。

a. service-jee-jaxrs アプリケーションをビルドします。

```
$ cd keycloak-quickstarts/service-jee-jaxrs/
```

```
$ mvn clean package -DskipTests
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Keycloak Quickstart: service-jee-jaxrs 3.1.0.Final
[INFO] ------------------------------------------------------------------------
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.153 s
[INFO] Finished at: 2017-06-26T12:06:12+02:00
[INFO] Final Memory: 25M/241M
[INFO] ------------------------------------------------------------------------
```

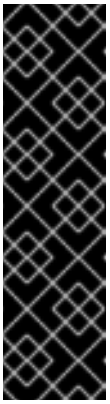
b. maven-enforcer-plugin プラグインの app-jee-jsp/config/keycloak.json 要件を コメントアウト し、app-jee-jsp アプリケーションをビルドします。

```
service-jee-jaxrs]$ cd ../app-jee-jsp/
```

```
app-jee-jsp]$ sed -i /\<executions\>/s/^/\<\!--/ pom.xml
```

```
app-jee-jsp]$ sed -i '/\(<\/executions>\)/a\-->' pom.xml
```

```
app-jee-jsp]$ mvn clean package -DskipTests
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Keycloak Quickstart: app-jee-jsp 3.1.0.Final
[INFO] ------------------------------------------------------------------------
...
[INFO] Building war: /tmp/github/keycloak-quickstarts/app-jee-jsp/target/app-jsp.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.018 s
[INFO] Finished at: 2017-06-26T12:22:25+02:00
[INFO] Final Memory: 35M/310M
[INFO] ------------------------------------------------------------------------
```



重要

app-jee-jsp クイックスタートではアダプターの設定が必要で、アダプター設定ファイル (keycloak.json) がクイックスタートのルートの config/ ディレクトリーにあり、クイックスタートを正常にビルドする必要があります。ただし、この例では、EAP 6.4 / 7.1 for OpenShift イメージで利用可能な選択された環境変数を使用して後でアダプターを設定するため、現時点で keycloak.json アダプター設定ファイルの形式を指定する必要はありません。

4. ローカルファイルシステムでディレクトリー構造を準備します。

メインのバイナリービルドディレクトリーの deployments/ サブディレクトリーにあるアプリケーションアーカイブは、OpenShift でビルドされるイメージの 標準のデプロイメントディレクトリー に直接コピーされます。アプリケーションをデプロイするには、web アプリケーションデータが含まれるディレクトリー階層が正しく構成される必要があります。

ローカルファイルシステム上にバイナリービルド用のメインディレクトリーと、そのディレクトリー内に deployments/ サブディレクトリーを作成します。service-jee-jaxrs および app-jee-jsp クイックスタートの両方のビルドされた WAR アーカイブを deployments/ サブディレクトリーにコピーします。

```
app-jee-jsp]$ ls
config  pom.xml  README.md  src  target
```

```
app-jee-jsp]$ mkdir -p sso-eap7-bin-demo/deployments
```

```
app-jee-jsp]$ cp target/app-jsp.war sso-eap7-bin-demo/deployments/
```

```
app-jee-jsp]$ cp ../service-jee-jaxrs/target/service.war sso-eap7-bin-
demo/deployments/
```

```
app-jee-jsp]$ tree sso-eap7-bin-demo/
sso-eap7-bin-demo/
|__ deployments
    |__ app-jsp.war
    |__ service.war

1 directory, 2 files
```

注記

標準の **deployments** ディレクトリーの場所は、アプリケーションのデプロイに使用された基礎となるベースイメージによって異なります。以下の表を参照してください。

表4.1 デプロイメントディレクトリーの標準的な場所

基礎となるベースイメージの名前	デプロイメントディレクトリーの標準的な場所
EAP for OpenShift 6.4 and 7.1	$JBOSS_HOME/standalone/deployments
Java S2I for OpenShift	/deployments
JWS for OpenShift	$JWS_HOME/webapps

5. **EAP 6.4 / 7.1 イメージのイメージストリームを特定します。**

```
$ oc get is -n openshift | grep eap | cut -d ' ' -f 1
jboss-eap64-openshift
jboss-eap71-openshift
```

6. イメージストリームおよびアプリケーション名を指定して、新しいバイナリービルドを作成します。

注記

--image-stream=jboss-eap71-openshift パラメーターを、以下の oc コマンドの --image-stream=jboss-eap64-openshift に置き換え、JBoss EAP 6.4 for OpenShift イメージに JSP アプリケーションをデプロイします。

```
$ oc new-build --binary=true \
--image-stream=jboss-eap71-openshift \
--name=eap-app
--> Found image 31895a4 (3 months old) in image stream "openshift/jboss-eap71-
openshift" under tag "latest" for "jboss-eap71-openshift"

    JBoss EAP 7.4
    -------------
    Platform for building and running Jakarta EE applications on JBoss EAP 7.4

    Tags: builder, javaee, eap, eap7

    * A source build using binary input will be created
      * The resulting image will be pushed to image stream "eap-app:latest"
      * A binary build was created, use 'start-build --from-dir' to trigger a new build

--> Creating resources with label build=eap-app ...
    imagestream "eap-app" created
    buildconfig "eap-app" created
--> Success
```

7. バイナリービルドを開始します。直前の手順で 作成したバイナリービルドのメインディレクトリーを OpenShift ビルドのバイナリー入力が含まれるディレクトリーとして使用するよう oc 実行ファイルに指示します。app-jee-jsp の作業ディレクトリーで、次のコマンドを実行します。

```
app-jee-jsp]$ oc start-build eap-app \
--from-dir=./sso-eap7-bin-demo/ \
--follow
Uploading directory "sso-eap7-bin-demo" as binary input for the build ...
build "eap-app-1" started
Receiving source from STDIN as archive ...
Copying all war artifacts from /home/jboss/source/. directory into
/opt/eap/standalone/deployments for later deployment...
Copying all ear artifacts from /home/jboss/source/. directory into
/opt/eap/standalone/deployments for later deployment...
Copying all rar artifacts from /home/jboss/source/. directory into
/opt/eap/standalone/deployments for later deployment...
Copying all jar artifacts from /home/jboss/source/. directory into
/opt/eap/standalone/deployments for later deployment...
Copying all war artifacts from /home/jboss/source/deployments directory into
/opt/eap/standalone/deployments for later deployment...
'/home/jboss/source/deployments/app-jsp.war' ->
'/opt/eap/standalone/deployments/app-jsp.war'
'/home/jboss/source/deployments/service.war' ->
```

```
'/opt/eap/standalone/deployments/service.war'
Copying all ear artifacts from /home/jboss/source/deployments directory into
/opt/eap/standalone/deployments for later deployment...
Copying all rar artifacts from /home/jboss/source/deployments directory into
/opt/eap/standalone/deployments for later deployment...
Copying all jar artifacts from /home/jboss/source/deployments directory into
/opt/eap/standalone/deployments for later deployment...
Pushing image 172.30.82.129:5000/eap-app-demo/eap-app:latest ...
Pushed 6/7 layers, 86% complete
Pushed 7/7 layers, 100% complete
Push successful
```

8. ビルドに基づいて新規の OpenShift アプリケーションを作成します。

```
$ oc new-app eap-app
--> Found image 6b13d36 (2 minutes old) in image stream "eap-app-demo/eap-app"
under tag "latest" for "eap-app"

    eap-app-demo/eap-app-1:aa2574d9
    ------------------------------
    Platform for building and running Jakarta EE applications on JBoss EAP 7.4

    Tags: builder, javaee, eap, eap7

    * This image will be deployed in deployment config "eap-app"
    * Ports 8080/tcp, 8443/tcp, 8778/tcp will be load balanced by service "eap-app"
      * Other containers can access this service through the hostname "eap-app"

--> Creating resources ...
    deploymentconfig "eap-app" created
    service "eap-app" created
--> Success
    Run 'oc status' to view your app.
```

9. 現在の名前空間で EAP 6.4 / 7.1 JSP アプリケーションの実行中のコンテナーをすべて停止します。

```
$ oc get dc -o name
deploymentconfig/eap-app
```

```
$ oc scale dc/eap-app --replicas=0
deploymentconfig "eap-app" scaled
```

10. デプロイメントの前に EAP 6.4 / 7.1 JSP アプリケーションをさらに設定します。

    a. Red Hat Single Sign-On サーバーインスタンスに関する適切な詳細でアプリケーションを設定します。

警告

以下の SSO_PUBLIC_KEY 変数の値を、コピー された demo レルムの RSA 公開鍵の実際の内容に置き換えるようにしてください。

```
$ oc set env dc/eap-app \
-e HOSTNAME_HTTP="eap-app-eap-app-demo.openshift.example.com" \
-e HOSTNAME_HTTPS="secure-eap-app-eap-app-demo.openshift.example.com" \
-e SSO_DISABLE_SSL_CERTIFICATE_VALIDATION="true" \
-e SSO_USERNAME="appuser" \
-e SSO_PASSWORD="apppassword" \
-e SSO_REALM="demo" \
-e SSO_URL="https://secure-sso-sso-app-demo.openshift.example.com/auth" \
-e
SSO_PUBLIC_KEY="MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAkd
hXyKx97oIoO6HwnV/MiX2EHO55Sn+ydsPzbjJevI5F31UvUco9uA8dGl6oM8HrnaW
Wv+i8PvmIaRMhhl6Xs68vJTEc6d0soP+6A+aExw0coNRp2PDwvzsXVWPvPQg3+iyt
Stxu3lcndx+gC0ZYnxoRqL7rY7zKcQBScGEr78Nw6vZDwfe6d/PQ6W4xVErNytX9Ky
LFVAE1VvhXALyqEM/EqYGLmpjw5bMGVKRXnhmVo9E88CkFDH8E+aPiApb/gFuI1
GJOv+G8ySLoR1c8Y3L29F7C81odkVBp2yMm3RVFIGSPTjHqjO/nOtqYIfY4Wyw9m
RIoY5SyW7044dZXRwIDAQAB" \
-e SSO_SECRET="0bb8c399-2501-4fcd-a183-68ac5132868d"
deploymentconfig "eap-app" updated
```

b.

SSL および JGroups キーストアの両方の詳細を使用してアプリケーションを設定します。

```
$ oc set env dc/eap-app \
-e HTTPS_KEYSTORE_DIR="/etc/eap-secret-volume" \
-e HTTPS_KEYSTORE="eapkeystore.jks" \
-e HTTPS_PASSWORD="password" \
-e JGROUPS_ENCRYPT_SECRET="eap-jgroup-secret" \
-e JGROUPS_ENCRYPT_KEYSTORE_DIR="/etc/jgroups-encrypt-secret-volume" \
-e JGROUPS_ENCRYPT_KEYSTORE="eapjgroups.jceks" \
-e JGROUPS_ENCRYPT_PASSWORD="password"
deploymentconfig "eap-app" updated
```

c.

先に作成した SSL および JGroups のシークレット両方に対して OpenShift ボリュームを定義します。

```
$ oc volume dc/eap-app --add \
--name="eap-keystore-volume" \
--type=secret \
```

```
--secret-name="eap-ssl-secret" \
--mount-path="/etc/eap-secret-volume"
deploymentconfig "eap-app" updated
```

```
$ oc volume dc/eap-app --add \
--name="eap-jgroups-keystore-volume" \
--type=secret \
--secret-name="eap-jgroup-secret" \
--mount-path="/etc/jgroups-encrypt-secret-volume"
deploymentconfig "eap-app" updated
```

d.

アプリケーションのデプロイメント設定を、default の OpenShift サービスアカウントでアプリケーション Pod を実行するように設定します (デフォルト設定)。

```
$ oc patch dc/eap-app --type=json \
-p '[{"op": "add", "path": "/spec/template/spec/serviceAccountName", "value":
"default"}]'
"eap-app" patched
```

11.

変更されたデプロイメント設定を使用して EAP 6.4 / 7.1 JSP アプリケーションのコンテナーをデプロイします。

```
$ oc scale dc/eap-app --replicas=1
deploymentconfig "eap-app" scaled
```

12.

サービスをルートとして公開します。

```
$ oc get svc -o name
service/eap-app
```

```
$ oc get route
No resources found.
```

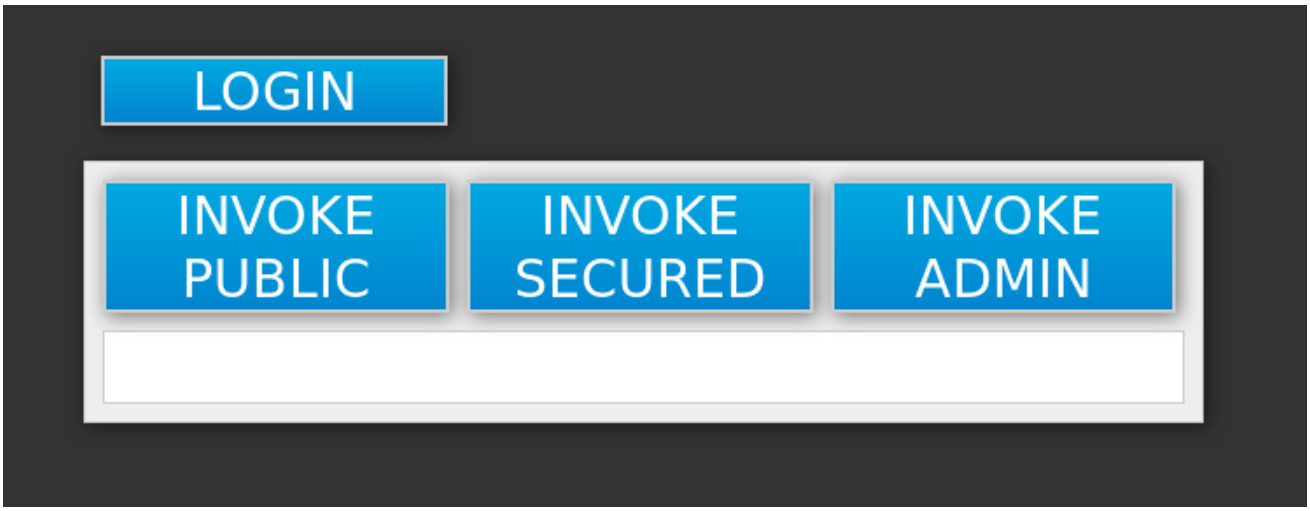
```
$ oc expose svc/eap-app
route "eap-app" exposed
```

```
$ oc get route
NAME      HOST/PORT                                PATH      SERVICES   PORT
TERMINATION   WILDCARD
eap-app   eap-app-eap-app-demo.openshift.example.com          eap-app    8080-tcp
None
```

4.4.1.5. アプリケーションにアクセスします。

URL http://eap-app-eap-app-demo.openshift.example.com/app-jsp を使用して、ブラウザーでア

プリケーションにアクセスします。次の画像のような出力が表示されます。



手順

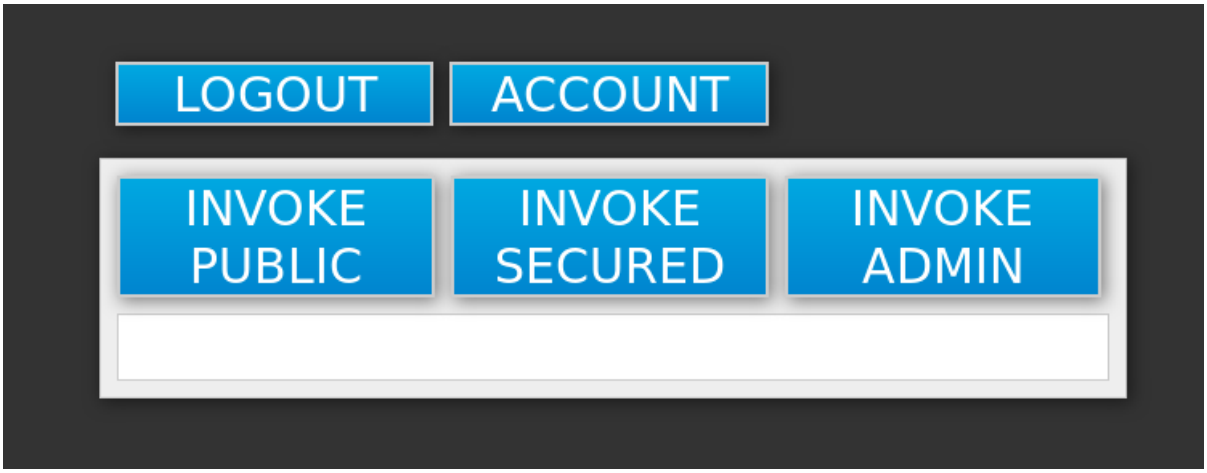
アプリケーションをテストするには、以下を実行します。

1. INVOKE PUBLIC ボタンをクリックして、認証を必要としない public エンドポイントにアクセスします。

   Message: public 出力が表示されるはずです。

2. LOGIN ボタンをクリックして、ユーザー認証用に demo レルムに対して Red Hat Single Sign-On サーバーインスタンスにリダイレクトされます。

   以前に設定した Red Hat Single Sign-On ユーザーのユーザー名とパスワード (appuser / apppassword) を指定します。Log in をクリックします。以下のイメージで説明されているように、アプリケーションの変更を確認できます。

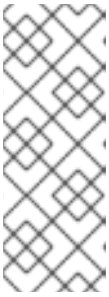


3. セキュリティー保護された エンドポイントにアクセスするには、**INVOKE SECURED** ボタンをクリックします。

   Message: secured 出力が表示されるはずです。

4. **INVOKE ADMIN** ボタンをクリックして **admin** エンドポイントにアクセスします。

   403 Forbidden 出力が表示されるはずです。



注記

admin エンドポイントでは、**admin Red Hat Single Sign-On** ロールを持つユーザーが適切に呼び出す必要があります。**appuser** へのアクセスは、**user** ロール権限のみで禁止され、セキュリティー保護された エンドポイントへのアクセスが可能になります。

手順

appuser を **Red Hat Single Sign-On** ロール **admin** に追加するには、以下の手順を実行します。

1. **Red Hat Single Sign-On** サーバーのインスタンスの管理コンソールにアクセスします。

   https://secure-sso-sso-app-demo.openshift.example.com/auth/admin.

   Red Hat Single Sign-On 管理者ユーザーの認証情報 を使用します。

2. Manage サイドバーで Users をクリックし、demo レルムのユーザー情報を表示します。

3. View all users ボタンをクリックします。

4. appuser の ID リンクをクリックします。あるいは、Actions 列の Edit ボタンをクリックします。

5. Role Mappings タブをクリックします。

6. Realm Roles 行の Available Roles リストから admin エントリーを選択します。

7. Add selected> ボタンをクリックし、admin ロールをユーザーに追加します。

8. EAP 6.4 / 7.1 JSP サービスアプリケーションに戻ります。

   http://eap-app-eap-app-demo.openshift.example.com/app-jsp.

9. LOGOUT ボタンをクリックして、appuser のロールマッピングを再読み込みします。

10. 再度 LOGIN ボタンをクリックし、プロバイダー appuser 認証情報をクリックします。

11. 再度 INVOKE ADMIN ボタンをクリックします。

    Message: admin というメッセージが表示されるはずです。

4.5. OPENID-CONNECT クライアントを使用した RED HAT SINGLE SIGN-ON への EAP アプリケーションの自動登録

以下の例では、OpenID-Connect クライアントアダプターを使用して、EAP プロジェクトの Red Hat Single Sign-On レルム、ロール、およびユーザー認証情報を準備します。これらの認証情報は、Red Hat Single Sign-On クライアントの自動登録のために EAP for OpenShift テンプレートに提供さ

れます。デプロイ後、Red Hat Single Sign-On ユーザーを使用して JBoss EAP の認証およびアクセスが可能になります。

注記

以下の例では OpenID-Connect クライアントを使用していますが、SAML クライアントも使用できます。OpenID-Connect と SAML クライアントの違いに関する詳細は、「Red Hat Single Sign-On Clients」および「Red Hat Single Sign-On Client の自動および手動での登録方法」を参照してください。

前提条件

- Preparing Red Hat Single Sign-On Authentication for OpenShift Deployment で説明されている手順を前提としています。

4.5.1. OpenShift デプロイメント用の Red Hat Single Sign On 認証の準備

cluster:admin ロールを持つユーザーとして OpenShift CLI にログインします。

1. 新しいプロジェクトを作成します。

```
$ oc new-project eap-app-demo
```

2. view ロールを デフォルト のサービスアカウントに追加します。これにより、サービスアカウントが eap-app-demo 名前空間のすべてのリソースを表示できるようになります。これは、クラスターの管理に必要です。

```
$ oc policy add-role-to-user view system:serviceaccount:$(oc project -q):default
```

3. EAP テンプレートには、SSL キーストアと JGroups キーストア が必要です。
この例では、Java Development Kit に含まれるパッケージの keytool を使用して、これらのキーストアの自己署名証明書を生成します。以下のコマンドはパスワードを要求します。

   a. SSL キーストアのセキュアキーを生成します。

```
$ keytool -genkeypair -alias https -storetype JKS -keystore eapkeystore.jks
```

b. JGroups キーストアのセキュアキーを生成します。

```
$ keytool -genseckey -alias jgroups -storetype JCEKS -keystore eapjgroups.jceks
```

4. SSL および JGroup キーストアファイルで EAP for OpenShift シークレットを生成します。

```
$ oc create secret generic eap-ssl-secret --from-file=eapkeystore.jks
$ oc create secret generic eap-jgroup-secret --from-file=eapjgroups.jceks
```

5. EAP シークレットを default のサービスアカウントに追加します。

```
$ oc secrets link default eap-ssl-secret eap-jgroup-secret
```

4.5.2. Red Hat Single Sign-On 認証情報の準備

Red Hat Single Sign-On デプロイメント時に作成された 管理者アカウント を使用して、https://secure-sso-<project-name>.<hostname>/auth/admin で暗号化された Red Hat Single Sign-On Web サーバーにログインします。

手順

レルムの作成

1. サイドバーの上部にあるレルム名前空間にカーソルを合わせ、Add Realm をクリックします。

2. レルム名を入力して (この例では eap-demoを使用)、Create をクリックします。

公開鍵のコピー

新規作成された eap-demo レルムで Keys タブをクリックして、生成された公開鍵をコピーします。この例では、簡潔に変数 <realm-public-key> を使用しています。後で Red Hat Single Sign-On 対応の JBoss EAP イメージをデプロイするのに使用されます。

ロールの作成

サンプル EAP アプリケーションの web.xml で定義された JEE ロールに対応する名前で、Red Hat Single Sign-On でロールを作成します。このロールは Red Hat Single Sign-On アプリケーションユーザー に割り当てられ、ユーザーアプリケーションへのアクセスを認証します。

1. Configure サイドバーの Roles をクリックし、このレルムのロールを一覧表示します。これは新しいレルムであるため、デフォルトの offline_access ロールのみが必要です。
2. Add Role をクリックします。
3. ロール名を入力して (この例ではロール eap-user-role を使用)、Save をクリックします。

ユーザーの作成およびロールの割り当て

レルム管理ユーザーを割り当て、realm-management ロールを割り当て、Red Hat Single Sign-On サーバーで自動 Red Hat Single Sign-On クライアント登録を処理します。アプリケーションユーザーには、前のステップで作成した JEE ロールを割り当て、ユーザーアプリケーションへのアクセスを認証します。

レルム管理ユーザー を作成します。

1. Manage サイドバーで Users をクリックし、レルムのユーザー情報を表示します。
2. Add User をクリックします。
3. 有効なユーザー名 を入力して (この例ではユーザー eap-mgmt-user を使用)、Save をクリックします。
4. ユーザー設定を編集します。ユーザー空間の Credentials タブをクリックし、ユーザーのパスワードを入力します。パスワードを確認した後、Reset Password をクリックしてユーザーパスワードを設定できます。ポップアップウィンドウに追加の確認を求めるプロンプトが表示されます。

5. Role Mappings をクリックし、レルムおよびクライアントロール設定を一覧表示します。Client Roles ドロップダウンメニューで realm-management を選択し、利用可能なロールをすべてユーザーに追加します。これにより、JBoss EAP イメージによるクライアント作成に使用できる Red Hat Single Sign-On サーバーの権限が提供されます。

アプリケーションユーザー を作成します。

1. Manage サイドバーで Users をクリックし、レルムのユーザー情報を表示します。

2. Add User をクリックします。

3. アプリケーションユーザー の有効な ユーザー名 と任意の追加情報を入力し、Save をクリックします。

4. ユーザー設定を編集します。ユーザー空間の Credentials タブをクリックし、ユーザーのパスワードを入力します。パスワードを確認した後、Reset Password をクリックしてユーザーパスワードを設定できます。ポップアップウィンドウに追加の確認を求めるプロンプトが表示されます。

5. Role Mappings をクリックし、レルムおよびクライアントロール設定を一覧表示します。Available Roles で、先に作成したロールを追加します。

4.5.3. Red Hat Single Sign-On が有効な JBoss EAP イメージのデプロイ

手順

1. OpenShift Web コンソールに戻り、Add to project をクリックしてデフォルトのイメージストリームおよびテンプレートを一覧表示します。

2. Filter by keyword 検索バーを使用して、一覧を sso に一致するものに制限します。See all をクリックして、必要なアプリケーションテンプレートを表示する必要がある場合があります。

3. eap71-sso-s2i イメージを選択して、すべてのデプロイメントパラメーターを一覧表示します。EAP ビルド時に Red Hat Single Sign-On 認証情報を設定するには、以下の Red Hat

Single Sign-On パラメーターを追加します。

変数	値の例
APPLICATION_NAME	sso
HOSTNAME_HTTPS	secure-sample-jsp.eap-app-demo.openshift32.example.com
HOSTNAME_HTTP	sample-jsp.eap-app-demo.openshift32.example.com
SOURCE_REPOSITORY_URL	https://repository-example.com/developer/application
SSO_URL	https://secure-sso-sso-app-demo.openshift32.example.com/auth
SSO_REALM	eap-demo
SSO_USERNAME	eap-mgmt-user
SSO_PASSWORD	password
SSO_PUBLIC_KEY	<realm-public-key>
HTTPS_KEYSTORE	eapkeystore.jks
HTTPS_PASSWORD	password
HTTPS_SECRET	eap-ssl-secret
JGROUPS_ENCRYPT_KEYSTORE	eapjgroups.jceks
JGROUPS_ENCRYPT_PASSWORD	password
JGROUPS_ENCRYPT_SECRET	eap-jgroup-secret

4. Create をクリックして JBoss EAP イメージをデプロイします。

JBoss EAP イメージがデプロイされるまでに数分間がかかる場合があります。

4.5.4. Red Hat Single Sign-On を使用した JBoss EAP サーバーへのログイン

手順

1. JBoss EAP アプリケーションサーバーにアクセスし、Login をクリックします。Red Hat Single Sign-On ログインにリダイレクトされます。

2. この例で作成した Red Hat Single Sign-On ユーザーを使用してログインします。Red Hat Single Sign-On サーバーに対して認証され、JBoss EAP アプリケーションサーバーに返されます。

4.6. SAML クライアントを使用した RED HAT SINGLE SIGN-ON への EAP アプリケーションの手動登録

この例では、EAP プロジェクトの Red Hat Single Sign-On レルム、ロール、およびユーザー認証情報を準備し、OpenShift デプロイメント用の EAP を設定します。デプロイ後、Red Hat Single Sign-On ユーザーを使用して JBoss EAP の認証およびアクセスが可能になります。

注記

この例では、SAML クライアントを使用していますが、OpenID-Connect クライアントも使用できます。SAML と OpenID-Connect クライアントの相違点の詳細は、「Red Hat Single Sign-On Clients」および「Red Hat Single Sign-On Client の自動および手動での登録方法」を参照してください。

前提条件

- Preparing Red Hat Single Sign-On Authentication for OpenShift Deployment で説明されている手順を前提としています。

4.6.1. Red Hat Single Sign-On 認証情報の準備

手順

Red Hat Single Sign-On デプロイメント時に作成された 管理者アカウント を使用して、https://secure-sso-<project-name>.<hostname>/auth/admin で暗号化された Red Hat Single Sign-On Web サーバーにログインします。

レルムの作成

1. カーソルをサイドバーの上部にあるレルム名前空間 (デフォルトは Master) の上に置

き、**Add Realm** をクリックします。

2. レルム名を入力して (この例では **saml-demo**を使用)、**Create** をクリックします。

公開鍵のコピー

新規作成された **saml-demo** レルムで **Keys** タブをクリックして、生成された公開鍵をコピーします。この例では、簡潔に変数 **realm-public-key** を使用します。これは後で **Red Hat Single Sign-On** 対応 **JBoss EAP** イメージをデプロイする必要があります。

ロールの作成

サンプル **EAP** アプリケーションの **web.xml** で定義された **JEE** ロールに対応する名前で、**Red Hat Single Sign-On** でロールを作成します。このロールは **Red Hat Single Sign-On** アプリケーションユーザー に割り当てられ、ユーザーアプリケーションへのアクセスを認証します。

1. **Configure** サイドバーの **Roles** をクリックし、このレルムのロールを一覧表示します。これは新しいレルムであるため、デフォルトの **offline_access** ロールのみが必要です。

2. **Add Role** をクリックします。

3. ロール名を入力して (この例では、**saml-user-role** ロールを使用)、**Save** をクリックします。

ユーザーの作成およびロールの割り当て

レルム管理ユーザーを割り当て、**realm-management** ロールを割り当て、**Red Hat Single Sign-On** サーバーで自動 **Red Hat Single Sign-On** クライアント登録を処理します。アプリケーションユーザーには、前のステップで作成した **JEE** ロールを割り当て、ユーザーアプリケーションへのアクセスを認証します。

レルム管理ユーザー を作成します。

1. **Manage サイドバーで Users をクリックし、レルムのユーザー情報を表示します。**

2. **Add User をクリックします。**

3. **有効なユーザー名 を入力して (この例ではユーザー app-mgmt-user を使用)、Save をクリックします。**

4. **ユーザー設定を編集します。ユーザー空間の Credentials タブをクリックし、ユーザーのパスワードを入力します。パスワードを確認した後、Reset Password をクリックしてユーザーパスワードを設定できます。ポップアップウィンドウに追加の確認を求めるプロンプトが表示されます。**

アプリケーションユーザー を作成します。

1. **Manage サイドバーで Users をクリックし、レルムのユーザー情報を表示します。**

2. **Add User をクリックします。**

3. **アプリケーションユーザー の有効な ユーザー名 と任意の追加情報を入力し、Save をクリックします。**

4. **ユーザー設定を編集します。ユーザー空間の Credentials タブをクリックし、ユーザーのパスワードを入力します。パスワードを確認した後、Reset Password をクリックしてユーザーパスワードを設定できます。ポップアップウィンドウに追加の確認を求めるプロンプトが表示されます。**

5. **Role Mappings をクリックし、レルムおよびクライアントロール設定を一覧表示します。Available Roles で、先に作成したロールを追加します。**

SAML クライアントを作成および設定します。

クライアントは、ユーザー認証を要求する Red Hat Single Sign-On エンティティーです。この例では、EAP アプリケーションの認証を処理する SAML クライアントを設定します。本セクションでは、

この手順の後半に必要な keystore.jks ファイルおよび keycloak-saml-subsystem.xml ファイルを保存します。

SAML クライアントを作成します。

1. Configure サイドバーの Clients をクリックし、レルム内のクライアントを一覧表示します。Create をクリックします。
2. 有効な クライアント ID を入力します。この例では sso-saml-demo を使用しています。
3. Client Protocol ドロップダウンメニューで、saml を選択します。
4. アプリケーションの ルート URL を入力します。この例では、https://demoapp-eap-app-demo.openshift32.example.com を使用します。
5. Save をクリックします。

SAML クライアントを設定します。

Settings タブで、新しい sso-saml-demo クライアントの ルート URL および 有効なリダイレクト URL を設定します。

1. Root URL には、クライアントの作成時に使用したアドレスと同じアドレスを入力します。この例では、https://demoapp-eap-app-demo.openshift32.example.com を使用します。
2. Valid Redirect URLs には、ログイン時またはログアウト時にリダイレクトされるユーザーのアドレスを入力します。以下の例では、ルート https://demoapp-eap-app-demo.openshift32.example.com/* に関連してリダイレクトアドレスを使用しています。

SAML 鍵をエクスポートします。

1. sso-saml-demo クライアント領域の SAML Keys タブをクリックし、Export をクリック

します。

2. この例では、アーカイブ形式 を JKS のままにします。この例では、デフォルトの キーエイリアス sso-saml-demo と、デフォルトのレルム証明書エイリアス saml-demo を使用しています。

3. Key Password と Store Password。この例では、両方の パスワード を使用しています。

4. 後で使用できるように keystore-saml.jks ファイルを ダウンロード して保存します。

5. sso-saml-demo クライアントをクリックして、次の手順の準備が整ったクライアント領域に戻ります。

クライアントアダプターをダウンロードします。

1. Installation をクリックします。

2. Format Option ドロップダウンメニューを使用してフォーマットを選択します。この例では、Keycloak SAML Wildfly/JBoss Subsystem を使用します。

3. Download をクリックして keycloak-saml-subsystem.xml ファイルを保存します。

keystore-saml.jks は、次のセクションの他の EAP キーストアとともに使用され、EAP アプリケーションプロジェクトの OpenShift シークレットを作成します。keystore-saml.jks ファイルを OpenShift ノードにコピーします。
keycloak-saml-subsystem.xml は変更され、アプリケーションデプロイメントで使用されます。それを secure-saml-deployments としてアプリケーションの /configuration フォルダーにコピーします。

4.6.2. OpenShift デプロイメント用の Red Hat Single Sign On 認証の準備

cluster:admin ロールを持つユーザーとして OpenShift CLI にログインします。

手順

1. 新しいプロジェクトを作成します。

```
$ oc new-project eap-app-demo
```

2. view ロールを デフォルト のサービスアカウントに追加します。これにより、サービスアカウントが eap-app-demo 名前空間のすべてのリソースを表示できるようになります。これは、クラスターの管理に必要です。

```
$ oc policy add-role-to-user view system:serviceaccount:$(oc project -q):default
```

3. EAP テンプレートには、SSL キーストアと JGroups キーストア が必要です。
この例では、Java Development Kit に含まれるパッケージの keytool を使用して、これらのキーストアの自己署名証明書を生成します。以下のコマンドはパスワードを要求します。

   a. SSL キーストアのセキュアキーを生成します。

```
$ keytool -genkeypair -alias https -storetype JKS -keystore eapkeystore.jks
```

   b. JGroups キーストアのセキュアキーを生成します。

```
$ keytool -genseckey -alias jgroups -storetype JCEKS -keystore eapjgroups.jceks
```

4. SSL および JGroup キーストアファイルで EAP for OpenShift シークレットを生成します。

```
$ oc create secret generic eap-ssl-secret --from-file=eapkeystore.jks
$ oc create secret generic eap-jgroup-secret --from-file=eapjgroups.jceks
```

5. EAP アプリケーションシークレットを、以前に作成した EAP サービスアカウントに追加します。

```
$ oc secrets link default eap-ssl-secret eap-jgroup-secret
```

4.6.3. secure-saml-deployments ファイルの変更

前提条件

-

以前のセクションで Red Hat Single Sign-On クライアントからエクスポートされた keycloak-saml-subsystem.xml は、アプリケーションの /configuration フォルダーにコピーされ、secure-saml-deployments に変更されているはずです。EAP は起動時にこのファイルを検索し、Red Hat Single Sign-On SAML アダプター設定内の standalone-openshift.xml ファイルにコピーします。

手順

1. テキストエディターで /configuration/secure-saml-deployments ファイルを開きます。

2. secure-deployment name タグの YOUR-WAR.war 値を、アプリケーションの .war ファイルに置き換えます。この例では sso-saml-demo.war を使用します。

3. logout page タグの SPECIFY YOUR LOGOUT PAGE! の値を url に置き換え、ユーザーがアプリケーションからログアウトしたときにユーザーをリダイレクトします。この例では、/index.jsp を使用します。

4. <PrivateKeyPem> および <CertificatePem> のタグキーを削除し、これをキーストア情報に置き換えます。

```
...
<Keys>
  <Key signing="true">
    <KeyStore file= "/etc/eap-secret-volume/keystore-saml.jks" password="password">
      <PrivateKey alias="sso-saml-demo" password="password"/>
      <Certificate alias="sso-saml-demo"/>
    </KeyStore>
  </Key>
</Keys>
```

keystore-saml.jks のマウントパス (この例では /etc/eap-secret-volume/keystore-saml.jks) は、EAP_HTTPS_KEYSTORE_DIR パラメーターを使用してアプリケーションテンプレートで指定できます。
PrivateKey と 証明書 のエイリアスおよびパスワードは、SAML キーが Red Hat Single Sign-On クライアントからエクスポートされたときに設定されていました。

5. 2 番目の <CertificatePem> タグおよびキーを削除し、これをレルム証明書情報に置き換えます。

```
...
<Keys>
  <Key signing="true">
    <KeyStore file="/etc/eap-secret-volume/keystore-saml.jks" password="password">
```

```
      <Certificate alias="saml-demo"/>
    </KeyStore>
  </Key>
</Keys>
...
```

証明書エイリアスとパスワードは、SAML キーが Red Hat Single Sign-On クライアントからエクスポートされている場合に設定されます。

6. /configuration/secure-saml-deployments ファイルを保存し、閉じます。

4.6.4. アプリケーションの web.xml での SAML クライアント登録の設定

クライアントタイプは、アプリケーション web.xml の <auth-method> キーでも指定する必要があります。このファイルは、デプロイ時にイメージにより読み取られます。

アプリケーションの web.xml ファイルを開き、以下が含まれることを確認します。

```
...
<login-config>
  <auth-method>KEYCLOAK-SAML</auth-method>
</login-config>
...
```

4.6.5. アプリケーションのデプロイ

アプリケーション自体に設定されているため、イメージの Red Hat Single Sign-On 設定を含める必要はありません。アプリケーションのログインページに移動すると、Red Hat Single Sign-On ログインにリダイレクトされます。先に作成した アプリケーションユーザー を使用して、Red Hat Single Sign-On でアプリケーションにログインします。

第5章 参照

5.1. アーティファクトリポジトリーミラー

Maven のリポジトリーは、さまざまなタイプのビルドアーティファクトおよび依存関係を保持します (すべてのプロジェクト jars、ライブラリー jar、プラグイン、その他のプロジェクト固有のアーティファクト)。また、S2I ビルドの実行中にアーティファクトのダウンロード元となる場所も指定します。組織では、中央リポジトリーを使用する他に、ローカルのカスタムリポジトリー (ミラー) をデプロイすることが一般的です。

ミラーを使用する利点は次のとおりです。

- 地理的に近く、高速な同期ミラーを使用できる。
- リポジトリーの内容をより良く制御できる。
- パブリックサーバーおよびリポジトリーに依存する必要なく、異なるチーム (開発者、CI) 全体でアーティファクトを共有できる。
- ビルド時間が改善される。

多くの場合で、リポジトリーマネージャーはミラーへのローカルキャッシュとして機能できます。リポジトリーマネージャーがすでにデプロイされ、http://10.0.0.1:8080/repository/internal/ で外部から到達可能な場合は、以下の手順で MAVEN_MIRROR_URL 環境変数をアプリケーションのビルド設定に提供すると、S2I ビルドはこのマネージャーを使用できます。

手順

1. MAVEN_MIRROR_URL 変数を適用するビルド設定の名前を特定します。

```
$ oc get bc -o name
buildconfig/sso
```

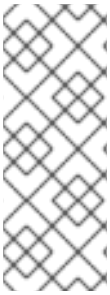
2. MAVEN_MIRROR_URL 環境変数で sso のビルド設定を更新します。

```
$ oc set env bc/sso \
  -e MAVEN_MIRROR_URL="http://10.0.0.1:8080/repository/internal/"
buildconfig "sso" updated
```

3. 設定を確認します。

```
$ oc set env bc/sso --list
# buildconfigs sso
MAVEN_MIRROR_URL=http://10.0.0.1:8080/repository/internal/
```

4. アプリケーションの新しいビルドをスケジュールします。



注記

アプリケーションのビルド中、Maven 依存関係はデフォルトのパブリックリポジトリーではなく、リポジトリーマネージャーからプルされることを確認できます。またビルドの完了後、ビルド中に取得および使用されたすべての依存関係がミラーに追加されたことが確認できます。

5.2. 環境変数

5.2.1. 情報環境変数

以下の情報は、イメージに関する情報を伝えるように設計されており、ユーザーが変更すべきではありません。

表5.1 情報環境変数

変数名	説明	値の例
AB_JOLOKIA_AUTH_OPENSHIFT	-	true
AB_JOLOKIA_HTTPS	-	true
AB_JOLOKIA_PASSWORD_RANDOM	-	true
JBOSS_IMAGE_NAME	イメージ名（"name" ラベルと同じ）	rh-sso-7/sso76-openshift-rhel8
JBOSS_IMAGE_VERSION	イメージのバージョン（"version" ラベルと同じ）	7.6

変数名	説明	値の例
JBOSS_MODULES_SYSTEM_PKGS	-	org.jboss.logmanager,jdk.nashorn.api

5.2.2. 設定環境変数

構成環境変数は、再ビルドを必要とせずにイメージを便利に調整するように設計されており、必要に応じてユーザーが設定する必要があります。

表5.2 設定環境変数

変数名	説明	値の例
AB_JOLOKIA_AUTH_OPENSHIFT	OpenShift TLS 通信のクライアント認証を切り替えます。このパラメーターの値は、提供されるクライアントの証明書に含まれる必要がある相対識別名になります。このパラメーターを有効にすると、Jolokia が自動的に https 通信モードに切り替わります。デフォルトの CA 証明書は **/var/run/secrets/kubernetes.io/serviceaccount/ca.crt** に設定されます。	true
AB_JOLOKIA_CONFIG	設定されている場合には、このファイル (パスを含む) を Jolokia JVM エージェントプロパティーとして使用します (Jolokia の reference manual を参照)。設定されていない場合は、本書に定義されている設定を使用して **/opt/jolokia/etc/jolokia.properties** ファイルが作成されます。それ以外の場合、本ガイドの残りの設定は無視されます。	/opt/jolokia/custom.properties
AB_JOLOKIA_DISCOVERY_ENABLED	Jolokia の検索を有効にします。デフォルトは **false** に設定されます。	true
AB_JOLOKIA_HOST	バインド先のホストアドレス。デフォルトは 0.0.0.0 です。	127.0.0.1

変数名	説明	値の例
AB_JOLOKIA_HTTPS	https との安全な通信を有効にします。デフォルトでは、**AB_JOLOKIA_OPTS** に serverCert 設定が指定されていないと、自己署名サーバー証明書が生成されます。**注記: 値が空の文字列に設定されていると、https は オフ になります。値が空の文字列に設定されていると、https が 有効 になります。**	true
AB_JOLOKIA_ID	使用するエージェント ID (デフォルトではコンテナー ID である $HOSTNAME)。	openjdk-app-1-xqlsj
AB_JOLOKIA_OFF	設定すると、Jolokia のアクティベートが無効になります (つまり、空の値をエコーします)。Jolokia はデフォルトで有効になります。**注記: 値が空の文字列に設定されていると、https は オフ になります。値が空の文字列に設定されていると、https が 有効 になります。**	true
AB_JOLOKIA_OPTS	エージェント設定に追加されるその他のオプション。これらは **"key=value, key=value, … <200b> "** の形式で指定する必要があります。	backlog=20
AB_JOLOKIA_PASSWORD	Basic 認証のパスワード。デフォルトでは認証は無効になっています。	mypassword
AB_JOLOKIA_PASSWORD_RANDOM	これが設定されていると、**AB_JOLOKIA_PASSWORD** 用に無作為に生成され、**/opt/jolokia/etc/jolokia.pw** ファイルに保存されます。	true
AB_JOLOKIA_PORT	使用するポート (デフォルト: 8778)。	5432
AB_JOLOKIA_USER	Basic 認証のユーザー。デフォルトは **jolokia** に設定されます。	myusername

変数名	説明	値の例
CONTAINER_CORE_LIMIT	「CFS Bandwidth Control」で説明されているように、計算されたコア制限。	2
GC_ADAPTIVE_SIZE_POLICY_WEIGHT	現在のガベージコレクション (GC) 時間と以前の GC 時間に指定される重み。	90
GC_MAX_HEAP_FREE_RATIO	縮小を回避するための GC 後のヒープ解放の最大パーセンテージ。	40
GC_MAX_METASPACE_SIZE	メタスペースの最大サイズ。	100
GC_TIME_RATIO_MIN_HEAP_FREE_RATIO	拡大を回避するための GC 後のヒープ解放の最小パーセンテージ。	20
GC_TIME_RATIO	ガベージコレクションで費やされた時間に対する、ガベージコレクションの外で費やされた時間 (たとえば、アプリケーションの実行に費やされた時間) の比率を指定します。	4
JAVA_DIAGNOSTICS	これを設定して、問題が発生したときのいくつかの診断情報を標準化します。	true
JAVA_INITIAL_MEM_RATIO	これは、最大ヒープメモリーを基にデフォルトの初期ヒープメモリーを算出するために使用されます。デフォルトは 100 で、初期ヒープサイズに最大ヒープの 100% が使用されることを意味します。このメカニズムを省略するには、この値を 0 に設定します。この場合は、**-Xms** オプションを追加しません。	100

変数名	説明	値の例
JAVA_MAX_MEM_RATIO	これは、コンテナーの制限に基づいてデフォルトの最大ヒープメモリーを算出するために使用されます。コンテナーのメモリー制約のない Docker コンテナーで使用されると、このオプションは影響を受けません。メモリー制約がある場合、**-Xmx** はコンテナーで利用可能なメモリーの比率をここで設定します。デフォルト値の 50 は、利用可能なメモリーの 50% が上限として使用されることを意味します。このメカニズムを省略するには、この値を 0 に設定します。この場合は、**-Xmx** オプションを追加しません。	40
JAVA_OPTS_APPEND	サーバー起動オプション。	-Dkeycloak.migration.action=export -Dkeycloak.migration.provider=dir -Dkeycloak.migration.dir=/tmp
MQ_SIMPLE_DEFAULT_PHYSICAL_DESTINATION	後方互換性を維持するには、true に設定して、**queue/MyQueue** および **topic/MyTopic** の代わりに **MyQueue** および **MyTopic** を物理宛先名のデフォルトとして使用します。	false
OPENSHIFT_KUBE_PING_LABELS	クラスタリングのラベルセレクター。	app=sso-app
OPENSHIFT_KUBE_PING_NAMESPACE	クラスタリングプロジェクトの名前空間。	myproject
SCRIPT_DEBUG	**true** に設定すると、bash スクリプトが **-x** オプションで実行され、コマンドとその引数が実行時に出力されます。	true
SSO_ADMIN_PASSWORD	Red Hat Single Sign-On サーバーの **master** レルムの管理者アカウントのパスワード**必須。**値が指定されていないとこれは自動生成され、テンプレートがインスタンス化される時に OpenShift の命令メッセージとして表示されます。	adm-password

変数名	説明	値の例
SSO_ADMIN_USERNAME	Red Hat Single Sign-On サーバーの **master** レルムの管理者アカウントのユーザー名、**必須。**値が指定されていないとこれは自動生成され、テンプレートがインスタンス化される時に OpenShift の命令メッセージとして表示されます。	admin
SSO_HOSTNAME	Red Hat Single Sign-On サーバーのカスタムホスト名。**デフォルトでは設定されません。**設定されていないと、要求ヘッダーを使用して Red Hat Single Sign-On サーバーのホスト名を判断する **要求** ホスト名 SPI プロバイダーが使用されます。設定すると、指定した変数値に設定された Red Hat Single Sign-On サーバーのホスト名を持つ **固定** ホスト名 SPI プロバイダーが使用されます。**SSO_HOSTNAME** 変数が設定されている場合に追加の手順を実行する場合は、Customizing Hostname for the Red Hat Single Sign-On Serverセクションを参照してください。	rh-sso-server.openshift.example.com
SSO_REALM	この環境変数を指定すると、Red Hat Single Sign-On サーバーに作成されるレルムの名前。	demo
SSO_SERVICE_PASSWORD	Red Hat Single Sign-On サービスユーザーのパスワード	mgmt-password
SSO_SERVICE_USERNAME	Red Hat Single Sign-On サービスへのアクセスに使用されるユーザー名。これは、指定の Red Hat Single Sign-On レルム内にアプリケーションクライアントを作成するためにクライアントによって使用されます。この環境変数が指定されている場合は、このユーザーが作成されます。	sso-mgmtuser
SSO_TRUSTSTORE	シークレット内のトラストストアファイルの名前。	truststore.jks
SSO_TRUSTSTORE_DIR	トラストストアディレクトリー。	/etc/sso-secret-volume

変数名	説明	値の例
SSO_TRUSTSTORE_PASSWORD	トラストストアと証明書のパスワード。	mykeystorepass
SSO_TRUSTSTORE_SECRET	トラストストアファイルが含まれるシークレットの名前。sso-truststore-volume ボリュームに使用します。	truststore-secret

OpenShift 向けのRed Hat Single Sign-On の利用可能な アプリケーションテンプレート は、前述の構成変数 を一般的な OpenShift 変数 (たとえば APPLICATION_NAME または SOURCE_REPOSITORY_URL)、製品固有の変数 (たとえば HORNETQ_CLUSTER_PASSWORD)、またはデータベースイメージに典型的な構成変数 (たとえば POSTGRESQL_MAX_CONNECTIONS) と組み合わせることができます。これらの異なるタイプの設定変数はすべて、Red Hat Single Sign-On 対応アプリケーションを展開して、可能な限り意図したユースケースに沿ったものにするために、必要に応じて調整できます。Red Hat Single Sign-On が有効なアプリケーションのアプリケーションテンプレートのカテゴリーごとに利用可能な設定変数のリストを以下に示します。

5.2.3. すべての Red Hat Single Sign-On イメージのテンプレート変数

表5.3 すべての Red Hat Single Sign-On イメージで利用可能な設定変数

変数	説明
APPLICATION_NAME	アプリケーションの名前。
DB_MAX_POOL_SIZE	設定されたデータソースに xa-pool/max-pool-size を設定します。
DB_TX_ISOLATION	設定されたデータソースの transaction-isolation を設定します。
DB_USERNAME	データベースのユーザー名。
HOSTNAME_HTTP	http サービスルートのカスタムホスト名。デフォルトのホスト名 (例: <application-name>.<project>.<default-domain-suffix>) は空白のままにします。
HOSTNAME_HTTPS	https サービスルートのカスタムホスト名。デフォルトのホスト名 (例: <application-name>.<project>.<default-domain-suffix>) は空白のままにします。

変数	説明
HTTPS_KEYSTORE	シークレット内のキーストアファイルの名前。**HTTPS_PASSWORD** および **HTTPS_NAME** とともに定義されている場合は、HTTPS を有効にし、SSL 証明書の鍵ファイルを **$JBOSS_HOME/standalone/configuration** 下の相対パスに設定します。
HTTPS_KEYSTORE_TYPE	キーストアファイルのタイプ (JKS または JCEKS)。
HTTPS_NAME	サーバー証明書に関連付けられた名前 (例: **jboss**)。**HTTPS_PASSWORD** および **HTTPS_KEYSTORE** とともに定義された場合は、HTTPSを有効にし、SSL 名を設定します。
HTTPS_PASSWORD	キーストアおよび証明書のパスワード (例: **mykeystorepass**)。**HTTPS_NAME** および **HTTPS_KEYSTORE** とともに定義された場合は、HTTPS を有効にし、SSL キーパスワードを設定します。
HTTPS_SECRET	キーストアファイルが含まれるシークレットの名前。
IMAGE_STREAM_NAMESPACE	Red Hat ミドルウェアイメージの ImageStreams がインストールされている namespaceこれらの ImageStream は通常 **openshift** 名前空間にインストールされます。ImageStreams を別の名前空間またはプロジェクトでインストールした場合に限り、これを変更する必要があります。
JGROUPS_CLUSTER_PASSWORD	JGroups クラスターパスワード。
JGROUPS_ENCRYPT_KEYSTORE	シークレット内のキーストアファイルの名前。
JGROUPS_ENCRYPT_NAME	サーバー証明書に関連付けられた名前 (例: **secret-key**)。
JGROUPS_ENCRYPT_PASSWORD	キーストアおよび証明書のパスワード (**password** など)。
JGROUPS_ENCRYPT_SECRET	キーストアファイルが含まれるシークレットの名前。
SSO_ADMIN_USERNAME	Red Hat Single Sign-On サーバーの **master** レルムの管理者アカウントのユーザー名、**必須。**値が指定されていないとこれは自動生成され、テンプレートがインスタンス化される時に OpenShift 命令メッセージとして表示されます。

変数	説明
SSO_ADMIN_PASSWORD	Red Hat Single Sign-On サーバーの **master** レルムの管理者アカウントのパスワード**必須。**値が指定されていないとこれは自動生成され、テンプレートがインスタンス化される時に OpenShift 命令メッセージとして表示されます。
SSO_REALM	この環境変数を指定すると、Red Hat Single Sign-On サーバーに作成されるレルムの名前。
SSO_SERVICE_USERNAME	Red Hat Single Sign-On サービスへのアクセスに使用されるユーザー名。これは、指定の Red Hat Single Sign-On レルム内にアプリケーションクライアントを作成するためにクライアントによって使用されます。この環境変数が指定されている場合は、このユーザーが作成されます。
SSO_SERVICE_PASSWORD	Red Hat Single Sign-On サービスユーザーのパスワード
SSO_TRUSTSTORE	シークレット内のトラストストアファイルの名前。
SSO_TRUSTSTORE_SECRET	トラストストアファイルが含まれるシークレットの名前。**sso-truststore-volume** ボリュームに使用します。
SSO_TRUSTSTORE_PASSWORD	トラストストアと証明書のパスワード。

5.2.4. sso76-postgresql、sso76 -postgresql -persistent、および sso76-x509-postgresql-persistent に固有のテンプレート変数

表5.4 一時または永続ストレージを使用した Red Hat Single Sign-On 対応の PostgreSQL アプリケーションに固有の設定変数

変数	説明
DB_USERNAME	データベースのユーザー名。
DB_PASSWORD	データベースユーザーのパスワード。
DB_JNDI	**java:/jboss/datasources/postgresql** などのデータソースを解決するアプリケーションによって使用されるデータベースの JNDI 名
POSTGRESQL_MAX_CONNECTIONS	許可されるクライアント接続の最大数。これにより、準備済みトランザクションの最大数も設定します。

変数	説明
POSTGRESQL_SHARED_BUFFERS	データのキャッシュに使用する PostgreSQL 専用のメモリー容量を設定します。

5.2.5. 一般的な S2I イメージ (eap64 および eap71) 用のテンプレート変数

表5.5 S2I を介して構成された EAP 6.4 および EAP 7 アプリケーションの設定変数

変数	説明
APPLICATION_NAME	アプリケーションの名前。
ARTIFACT_DIR	アーティファクトディレクトリー。
AUTO_DEPLOY_EXPLODED	展開形式のデプロイメントコンテンツが自動的にデプロイされるかどうかを制御します。
CONTEXT_DIR	ビルドする Git プロジェクト内のパス。ルートプロジェクトディレクトリーの場合は空になります。
GENERIC_WEBHOOK_SECRET	汎用ビルドのトリガーシークレット。
GITHUB_WEBHOOK_SECRET	GitHub トリガーシークレット。
HORNETQ_CLUSTER_PASSWORD	HornetQ クラスター管理者パスワード。
HORNETQ_QUEUES	キューの名前。
HORNETQ_TOPICS	トピック名。
HOSTNAME_HTTP	https サービスルートのカスタムホスト名。デフォルトのホスト名 (例: **<application-name>.<project>.<default-domain-suffix>**) は空白のままにします。
HOSTNAME_HTTPS	https サービスルートのカスタムホスト名。デフォルトのホスト名 (例: **<application-name>.<project>.<default-domain-suffix>**) は空白のままにします。
HTTPS_KEYSTORE_TYPE	キーストアファイルのタイプ (JKS または JCEKS)。
HTTPS_KEYSTORE	シークレット内のキーストアファイルの名前。**HTTPS_PASSWORD** および **HTTPS_NAME** とともに定義されている場合は、HTTPS を有効にし、SSL 証明書の鍵ファイルを **$JBOSS_HOME/standalone/configuration** 下の相対パスに設定します。

変数	説明
HTTPS_NAME	サーバー証明書に関連付けられた名前 (例: **jboss**)。**HTTPS_PASSWORD** および **HTTPS_KEYSTORE** とともに定義された場合は、HTTPSを有効にし、SSL 名を設定します。
HTTPS_PASSWORD	キーストアおよび証明書のパスワード (例: **mykeystorepass**)。**HTTPS_NAME** および **HTTPS_KEYSTORE** とともに定義された場合は、HTTPS を有効にし、SSL キーパスワードを設定します。
HTTPS_SECRET	キーストアファイルが含まれるシークレットの名前。
IMAGE_STREAM_NAMESPACE	Red Hat ミドルウェアイメージの ImageStreams がインストールされている namespaceこれらの ImageStream は通常 **openshift** 名前空間にインストールされます。ImageStreams を別の名前空間またはプロジェクトでインストールした場合に限り、これを変更する必要があります。
JGROUPS_CLUSTER_PASSWORD	JGroups クラスターパスワード。
JGROUPS_ENCRYPT_KEYSTORE	シークレット内のキーストアファイルの名前。
JGROUPS_ENCRYPT_NAME	サーバー証明書に関連付けられた名前 (例: **secret-key**)。
JGROUPS_ENCRYPT_PASSWORD	キーストアおよび証明書のパスワード (**password** など)。
JGROUPS_ENCRYPT_SECRET	キーストアファイルが含まれるシークレットの名前。
SOURCE_REPOSITORY_REF	Git ブランチ/タグ参照。
SOURCE_REPOSITORY_URL	アプリケーションの Git ソース URI。

5.2.6. 自動クライアント登録用に、 eap64-sso-s2i および eap71-sso-s2i に固有のテンプレート変数

表5.6 S2I を介して構成された EAP 6.4 および EAP 7 Red Hat Single Sign-On 対応アプリケーションの設定変数

変数	説明
SSO_URL	Red Hat Single Sign-On サーバーの場所。

変数	説明
SSO_REALM	この環境変数を指定すると、Red Hat Single Sign-On サーバーに作成されるレルムの名前。
SSO_USERNAME	Red Hat Single Sign-On サービスへのアクセスに使用されるユーザー名。これは、指定の Red Hat Single Sign-On レルム内にアプリケーションクライアントを作成するために使用されます。これは、**sso76** - テンプレートのいずれかで指定された **SSO_SERVICE_USERNAME** と一致する必要があります。
SSO_PASSWORD	Red Hat Single Sign-On サービスユーザーのパスワード
SSO_PUBLIC_KEY	Red Hat Single Sign-On 公開鍵。公開鍵は、中間者によるセキュリティー攻撃を回避するために、テンプレートに渡すことが推奨されます。
SSO_SECRET	機密アクセスのための Red Hat Single Sign-On クライアントシークレット。
SSO_SERVICE_URL	Red Hat Single Sign-On サービスの場所。
SSO_TRUSTSTORE_SECRET	トラストストアファイルが含まれるシークレットの名前。**sso-truststore-volume** ボリュームに使用します。
SSO_TRUSTSTORE	シークレット内のトラストストアファイルの名前。
SSO_TRUSTSTORE_PASSWORD	トラストストアと証明書のパスワード。
SSO_BEARER_ONLY	Red Hat Single Sign-On クライアントアクセスタイプ。
SSO_DISABLE_SSL_CERTIFICATE_VALIDATION	true の場合は、EAP と Red Hat Single Sign-On Server 間の SSL 通信が安全ではない場合 (つまり、curl で証明書の検証が無効なっている場合)
SSO_ENABLE_CORS	Red Hat Single Sign-On アプリケーションの CORS を有効にします。

5.2.7. SAML クライアントでの自動クライアント登録用に、eap64-sso-s2i および eap71-sso-s2i に固有のテンプレート変数

表5.7 SAML プロトコルを使用して S2I を介して構築された EAP 6.4 および EAP 7 の Red Hat Single Sign-On 対応アプリケーション向け設定変数

変数	説明
SSO_SAML_CERTIFICATE_NAME	サーバー証明書に関連付けられた名前。
SSO_SAML_KEYSTORE_PASSWORD	キーストアおよび証明書のパスワード。
SSO_SAML_KEYSTORE	シークレット内のキーストアファイルの名前。
SSO_SAML_KEYSTORE_SECRET	キーストアファイルが含まれるシークレットの名前。
SSO_SAML_LOGOUT_PAGE	SAML アプリケーションの Red Hat Single Sign-On のログアウトページ。

5.3. 公開されたポート

ポート番号	説明
8443	HTTPS
8778	Jolokia の監視