Red Hat Single Sign-On 7.4

Eclipse OpenJ9 における Red Hat Single Sign-On for OpenShift

Red Hat Single Sign-On 7.4 向け

Last Updated: 2022-09-16

Red Hat Single Sign-On 7.4 Eclipse OpenJ9 における Red Hat Single Sign-On for OpenShift

Red Hat Single Sign-On 7.4 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

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Red_Hat_Single_Sign-On_for_OpenShift_on_Eclipse_OpenJ9.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドは、OpenJ9 において Red Hat Single Sign-On 7.4 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 4

第1章 はじめに 5
1.1. RED HAT SINGLE SIGN-ON とは? 5

第2章 はじめに 7
2.1. 比較： OPENSHIFT イメージおよび RED HAT SINGLE SIGN-ON 向けの RED HAT SINGLE SIGN-ON 7
2.2. バージョンの互換性とサポート 7
2.3. RED HAT SINGLE SIGN-ON FOR OPENSHIFT の非推奨のイメージストリームとアプリケーションテンプレート 7
2.4. 初期設定 (INITIAL SETUP) 7

第3章 使ってみる 8
3.1. 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 8
3.2. RED HAT SINGLE SIGN-ON イメージのデプロイ 9
3.2.1. デプロイメントの準備 9
3.2.2. 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 9
3.2.2.1. OpenShift CLI を使用したテンプレートのデプロイ 9
3.2.2.2. 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 10
3.2.2.3. 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 11
3.2.3. Red Hat Single Sign-On Pod の管理者コンソールへのアクセス 13

第4章 高度なコンセプト 15
4.1. RED HAT SINGLE SIGN-ON テンプレートの PASSTHROUGH TLS TERMINATION の要件およびデプロイ 15
4.1.1. デプロイメントの準備 15
4.1.2. HTTPS および JGroups キーストアの作成、および Red Hat Single Sign-On サーバーのトラストストアの作成 15
4.1.3. Secret 17
4.1.4. OpenShift CLI を使用した Chosen Red Hat Single Sign-On パススルー TLS テンプレートのデプロイ 17
4.2. RED HAT SINGLE SIGN-ON サーバーのホスト名のカスタマイズ 19
4.3. 外部データベースへの接続 20
4.4. カスタム JDBC ドライバーの使用 21
4.5. RED HAT SINGLE SIGN-ON サーバーの管理者アカウントの作成 22
4.5.1. 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 22
4.5.2. リモートシェルセッションを使用した Red Hat Single Sign-On Pod への管理者アカウントの作成 24
4.6. RED HAT SINGLE SIGN-ON イメージのデフォルト動作のカスタマイズ 25
4.7. デプロイメントプロセス 26
4.8. RED HAT SINGLE SIGN-ON クライアント 26
4.8.1. 自動および手動の Red Hat Single Sign-On クライアント登録方法 27
4.8.1.1. Red Hat Single Sign-On クライアントの自動登録 27
4.8.1.2. 手動 Red Hat Single Sign-On クライアント登録 29
4.9. OPENSHIFT シークレットでの RED HAT SINGLE SIGN-ON VAULT の使用 29
4.10. 制限事項 30

第5章 チュートリアル 31
5.1. ワークフローの例： 既存のデータベースを更新して、OPENSHIFT イメージバージョンの新しい RED HAT SINGLE SIGN-ON に移行する 31
5.1.1. データベースの自動移行 31
5.1.2. データベースの手動移行 32
5.2. ワークフローの例： RED HAT SINGLE SIGN-ON サーバーの環境全体でのデータベースの移行 40
5.2.1. 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 40
5.2.2. （オプション）追加の Red Hat Single Sign-On レルムおよびユーザーも エクスポートする 41

5.2.3. 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。 41
5.2.4. 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。 42
5.3. ワークフローの例： 認証に RED HAT SINGLE SIGN-ON を使用するように OPENSHIFT 3.11 を設定 44
5.3.1. Red Hat Single Sign-On 認証情報の設定 44
5.3.2. OpenShift マスターでの Red Hat Single Sign-On 認証の設定 45
5.3.3. OpenShift へのログイン 47
5.4. ワークフローの例： 既存の MAVEN BINARIES から OPENSHIFT アプリケーションを作成し、RED HAT SINGLE SIGN-ON を使用してこれのセキュリティーを保護する 47
5.4.1. 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 47
5.4.1.1. 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 47
5.4.1.2. レルム 管理ユーザーへのユーザー Red Hat Single Sign-On ロールの割り当て 49
5.4.1.3. 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 50
5.4.1.4. 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 51
5.4.1.5. アプリケーションへのアクセス 58
5.5. ワークフローの例： OPENID-CONNECT CLIENT 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 61
5.5.1. OpenShift デプロイメント用の Red Hat Single Sign On 認証の準備 62
5.5.2. Red Hat Single Sign-On 認証情報の準備 63
5.5.3. Red Hat Single Sign-On が有効な JBoss EAP イメージのデプロイ 65
5.5.4. Red Hat Single Sign-On を使用して JBoss EAP サーバーにログインします。 66
5.6. ワークフローの例： SAML クライアントでの RED HAT SINGLE SIGN-ON での EAP アプリケーションの手動登録 66
5.6.1. Red Hat Single Sign-On 認証情報の準備 67
5.6.2. OpenShift デプロイメント用の Red Hat Single Sign On 認証の準備 71
5.6.3. secure-saml-deployments ファイルの変更 72
5.6.4. アプリケーション web.xmlでの SAML クライアント登録の設定 73
5.6.5. アプリケーションのデプロイ 74

第6章 参照資料 75
6.1. アーティファクトリポジトリーミラー 75
6.2. 環境変数 76
6.2.1. 情報環境変数 76
6.2.2. 設定環境変数 77
6.2.3. すべての Red Hat Single Sign-On イメージのテンプレート変数 82
6.2.4. sso75-openj9-postgresql、sso75-openj9-postgresql-persistent、および sso75-openj9-x509-postgresql-persistent に固有のテンプレート変数 84
6.2.5. 一般的な eap64 および eap 71 S2I イメージのテンプレート変数 84
6.2.6. 自動クライアント登録の eap64-sso-s2i および eap71-sso-s2i に固有のテンプレート変数 86
6.2.7. SAML クライアントでの自動 クライアント登録に使用する eap64-sso-s2i および eap71-sso-s2i に固有のテンプレート変数 87
6.3. 公開されたポート 88

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。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社 の CTO、Chris Wright のメッセージ を参照してください。

第1章 はじめに

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。OpenShift イメージの Red Hat Single Sign-On 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、および RESTful Web サービス用のユーザーアカウントの一元管理を行う認証サーバーを提供します。

Eclipse OpenJ9 の Red Hat Single Sign-On for OpenShift は、**IBM Z/IBM Power Systems** のプラットフォームでのみ利用できます。その他の利用可能なプラットフォームは、Red Hat Single Sign-On for OpenShift on OpenJDKを参照してください。

Red Hat は、Red Hat Single Sign-On for OpenShift イメージバージョン番号 7.4.10.GA 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。これは、Red Hat Single Sign-On 7.4.10.GA 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、以下の 2 つのカテゴリーに分割できます。

- HTTPS および 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と、事前に準備されたテンプレート。これらは、パススルー TLS 終端を使用して TLS 通信を保護します。
    - **sso74-openj9-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 7.4.10.GA。
    - **sso74-openj9-postgresql**: 別の Pod の一時的な PostgreSQL データベースがサポートする Red Hat Single Sign-On 7.4.10.GA。
    - **sso74-openj9-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 7.4.10.GA。

注記

MySQL / MariaDB データベースで Red Hat Single Sign-On を使用するためのテンプレートは削除され、Red Hat Single Sign-On バージョン 7.4 以降は利用できません。

- OpenShift の内部 サービス提供の x509 証明書シークレットを使用するテンプレートは、 セキュアな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 クラスタートラフィックは **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 サーバートラストストアも、HTTPS キーストアの証明書の署名に使用される **/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt** CA 証明書ファイルを含む自動的に作成されます。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設定されます。これらのテンプレートは、re-encryption TLS termination を使用して TLS 通信を保護します。
    - **sso74-openj9-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 7.4.10.GA。**ASYM_ENCRYPT** JGroups プロトコルはクラスタートラフィックの暗号化に使用されます。
    - **sso74-openj9-x509-postgresql-persistent**: 自動生成された HTTPS キーストアと Red Hat Single Sign-On トラストストアを持つ Red Hat Single Sign-On 7.4.10.GA。永続的な PostgreSQL データベースでサポートされます。**ASYM_ENCRYPT** JGroups プロトコルはクラスタートラフィックの暗号化に使用されます。

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 が有効な Red Hat JBoss Enterprise Application Platform 6.4
- **eap71-sso-s2i**: Red Hat Single Sign-On が有効な Red Hat JBoss Enterprise Application Platform 7.1
- **datavirt63-secure-s2i**: Red Hat Single Sign-On が有効な Red Hat JBoss Data Virtualization 6.3

これらのテンプレートには、Red Hat Single Sign-On に固有の環境変数が含まれており、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します。

詳細は、「 Automatic and Manual Red Hat Single Sign-On Client registration Methods 」を参照してください。

第2章 はじめに

2.1. 比較： OPENSHIFT イメージおよび RED HAT SINGLE SIGN-ON 向けの RED HAT SINGLE SIGN-ON

Red Hat Single Sign-On for OpenShift イメージバージョン番号 7.4.10.GA は Red Hat Single Sign-On 7.4.10.GA をベースにしています。OpenShift イメージおよび Red Hat Single Sign-On 用の Red Hat Single Sign-On の機能には、いくつかの違いがあります。

- 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 が有効な JBoss EAP イメージは、適切な **web.xml** ファイルに **<auth-method>KEYCLOAK</auth-method> または <auth-method>KEYCLOAK-method>KEYCLOAK-method> が含まれる** OpenID Connect または SAML クライアント登録および設定を自動的に処理します。

2.2. 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についての詳細は、OpenShift および Atomic Platform Tested Integrations ページ の xPaaS 部分を参照してください。

2.3. RED HAT SINGLE SIGN-ON FOR OPENSHIFT の非推奨のイメージストリームとアプリケーションテンプレート

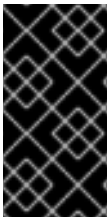
重要

7.0 から 7.3 までの OpenShift イメージバージョン番号の Red Hat Single Sign-On 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す。

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 7.4 または 7.4.10.GA と、これらのイメージバージョンに固有のアプリケーションテンプレートを使用することが推奨されます。

2.4. 初期設定 (INITIAL SETUP)

本書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したものと同じ OpenShift インスタンスと想定しています。



重要

Red Hat Single Sign-On for OpenShift イメージを以前のバージョンから 7.4.10.GA に移行する際の既存データベースの更新に関する情報は、「 Red Hat Single Sign-On for OpenShift Image を新しいバージョンに移行する際に既存のデータベースの更新」セクションを 参照してください。

第3章 使ってみる

3.1. 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

重要

Red Hat JBoss Middleware for OpenShift イメージは、認証が必要なセキュアな Red Hat レジストリー registry.redhat.io からオンデマンドでプルされます。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、管理者は個人ユーザーの Red Hat カスタマーポータル認証情報の代わりにレジストリーサービスアカウント（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。

レジストリーサービスアカウントを作成するには、レジストリーサービスアカウント に移動し、 必要に応じてログインします。

1. 「Registry **Service Accounts**」ページから、**Create Service Account**をクリックします。
2. サービスアカウントの名前を入力します（例： **registry.redhat.io-sa** ）。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（例： **サービスアカウントが registry.redhat.io から** コンテナーイメージを使用する）を入力します。
   b. **Create** をクリックします。
3. サービスアカウントの作成後に、Registry **Service Accounts** のページに表示されるテーブルの **Account name** 列にある **registry.redhat.io-sa** リンクをクリックします。
4. 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 Container Registry Authentication 」の記事を参照してください。

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 7.4.10.GA リソースのコアセットを更新します。

```
$ for resource in sso74-openj9-image-stream.json \
  sso74-openj9-https.json \
  sso74-openj9-postgresql.json \
  sso74-openj9-postgresql-persistent.json \
  sso74-openj9-x509-https.json \
  sso74-openj9-x509-postgresql-persistent.json
do
  oc replace -n openshift --force -f \
  https://raw.githubusercontent.com/jboss-container-images/redhat-sso-7-openshift-image/sso74-dev/templates/openj9/${resource}
done
```

4. 以下のコマンドを実行して、Red Hat Single Sign-On 7.4.10.GA 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/sso74-openj9-openshift-rhel8:7.4 --from=registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8:7.4 --confirm
```

3.2. RED HAT SINGLE SIGN-ON イメージのデプロイ

3.2.1. デプロイメントの準備

cluster:admin ロールを保持するユーザーで 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 sso-app-demo
```

2. **view** ロールを **default** 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は sso-app-demo namespace のすべてのリソースを表示できます。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
```

3.2.2. 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

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

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を一覧表示します。

```
$ oc get templates -n openshift -o name | grep -o 'sso74-openj9.\+'
sso74-openj9-https
sso74-openj9-postgresql
sso74-openj9-postgresql-persistent
sso74-openj9-x509-https
sso74-openj9-x509-postgresql-persistent
```

2. 選択したものをデプロイします。

```
$ oc new-app --template=sso74-openj9-x509-https
--> Deploying template "openshift/sso74-openj9-x509-https" to project sso-app-demo

    Red Hat Single Sign-On 7.4 (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.
```

または、OpenShift 3.x または 4.x Web コンソールを使用して Red Hat Single Sign-On テンプレートをデプロイするには、以下のいずれかの手順を実行します。

3.2.2.2. 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

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** をクリックしてから、**Browse Catalog** 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、リストを **sso** に一致するものに制限します。**Middleware** をクリックしてから **Integration** をクリックし、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 7.5 (Ephemeral)** を使用しています。

5. **Information** 手順で **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** プロジェクトスペースを選択します。**次へ** をクリックします。

7. **Binding** 手順で 、**この時点で Do not bind** ボタンを選択します。**Create** をクリックして続行します。

8. **結果** ステップで、**Continue to the project overview**リンクをクリックし、デプロイメントの状態を確認します。

3.2.2.3. 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

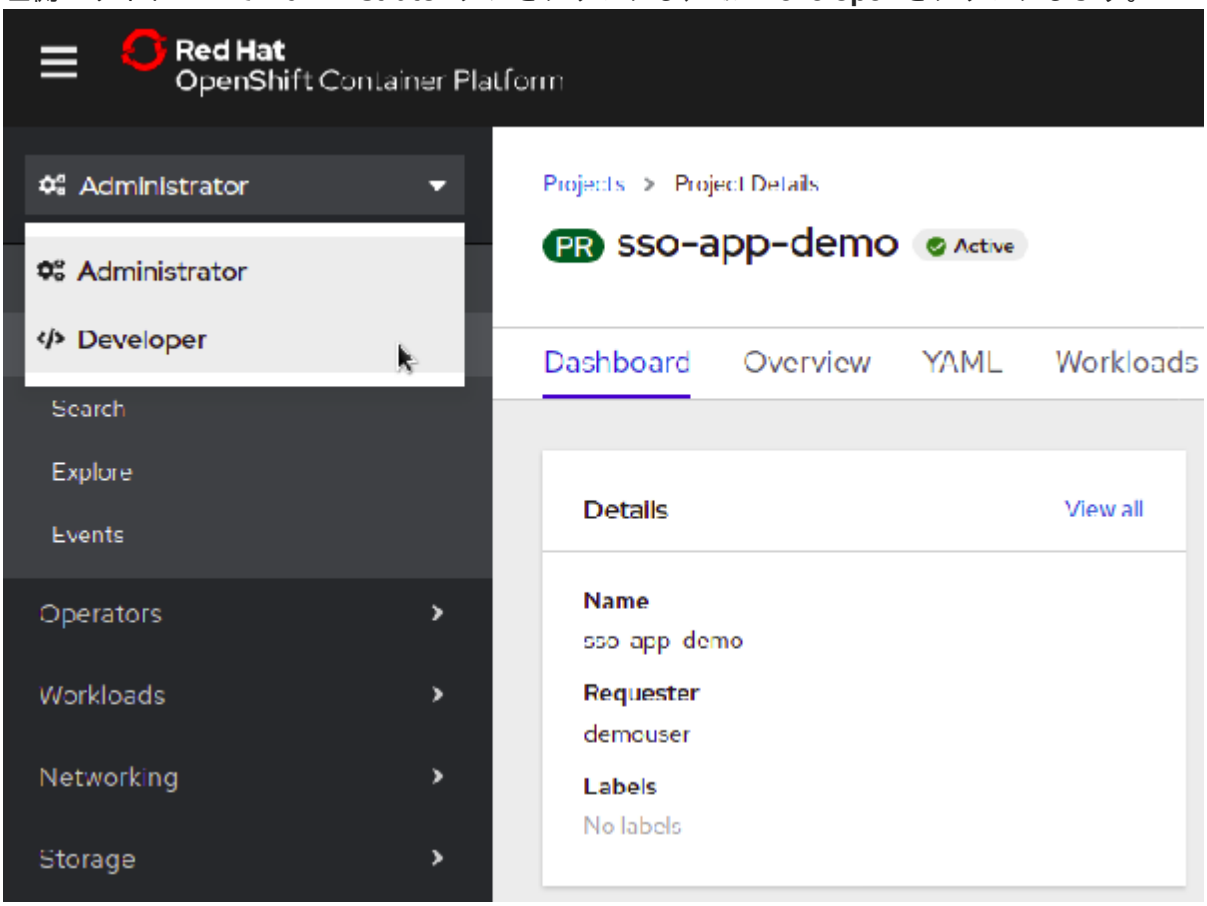
前提条件

- 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」で説明されている手順を実行します。

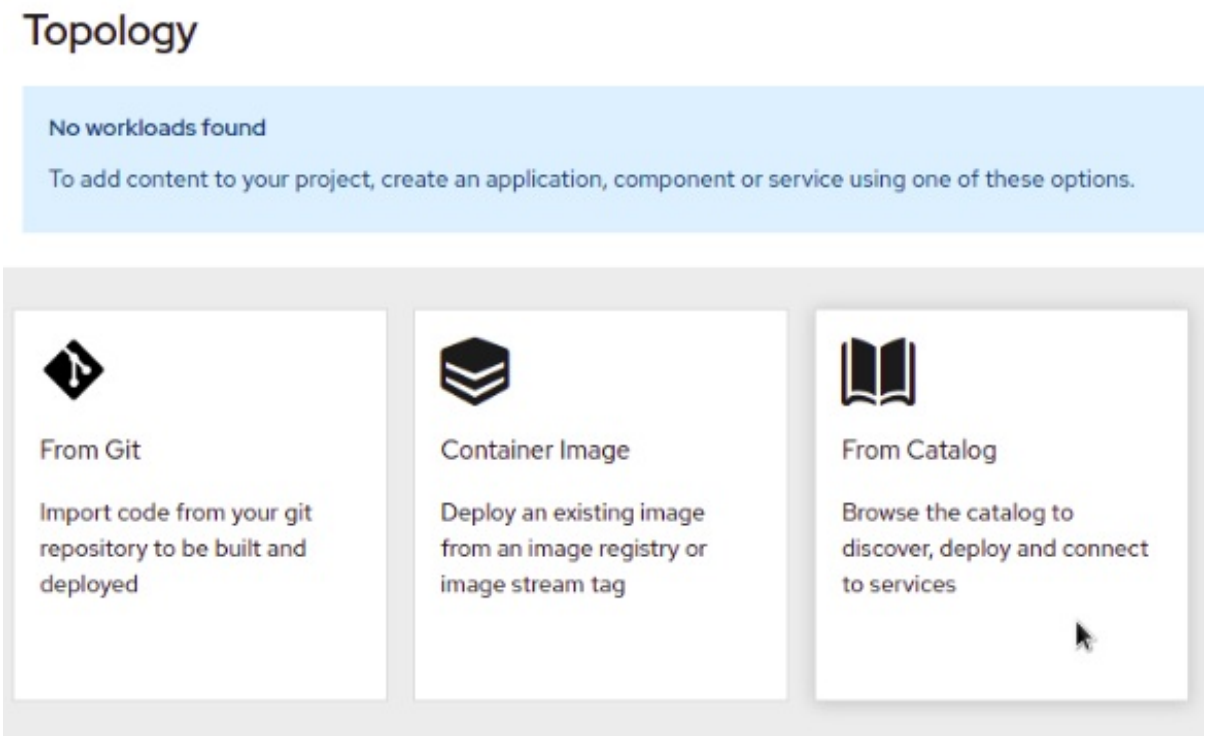
手順

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。

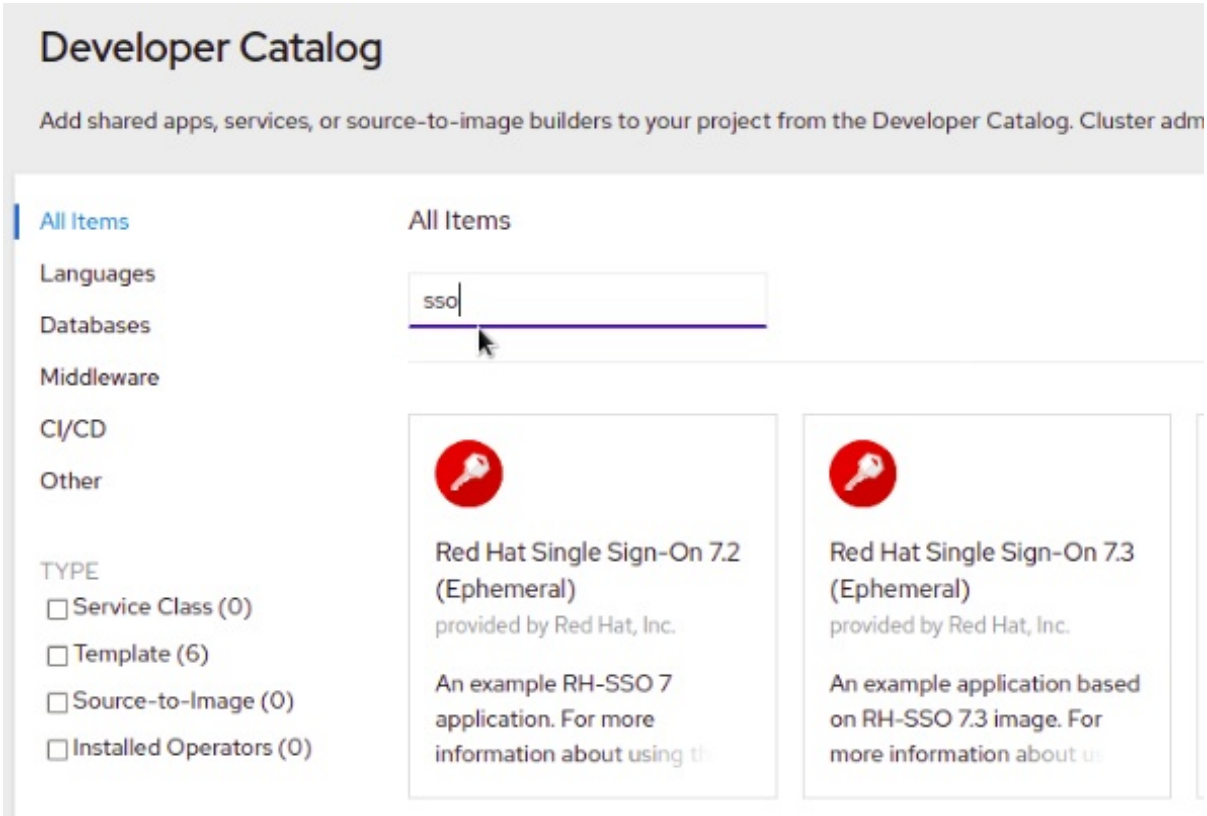
2. 左側のサイドバーで **Administrator** タブをクリックし、**</> Developer** をクリックします。



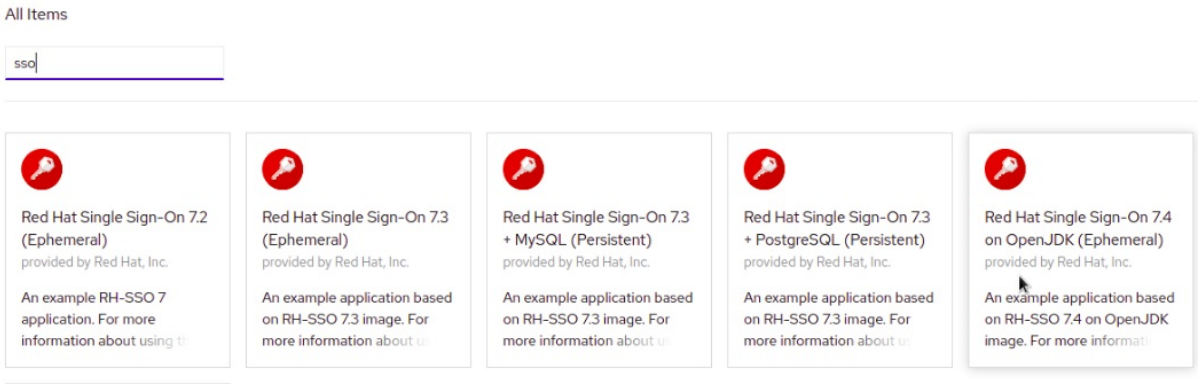
3. **From Catalog** をクリックします。



4. **sso** を検索します。



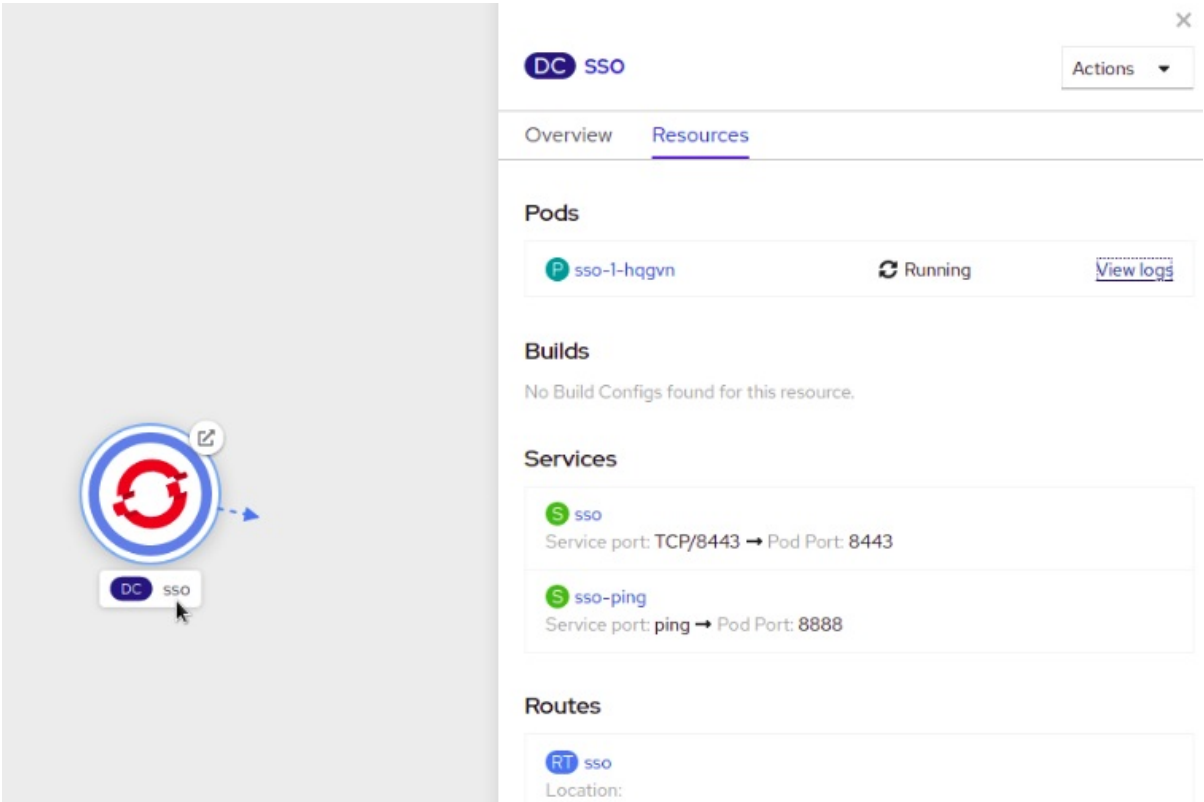
5. **Red Hat Single Sign-On 7.4 on OpenJDK (Ephemeral)** などのテンプレートを選択します。



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。



3.2.3. Red Hat Single Sign-On Pod の管理者コンソールへのアクセス

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
```

以下で 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**

管理者アカウントの使用

第4章 高度なコンセプト

これらには、Red Hat Single Sign-On サーバーのキーストアおよびトラストストアの設定、管理者アカウントの作成、利用可能な Red Hat Single Sign-On クライアント登録方法の概要、クラスタリングの設定に関するガイダンスなど、追加の設定トピックを説明します。

4.1. RED HAT SINGLE SIGN-ON テンプレートの PASSTHROUGH TLS TERMINATION の要件およびデプロイ

4.1.1. デプロイメントの準備

cluster:admin ロールを保持するユーザーで 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 sso-app-demo
```

2. **view** ロールを **default** 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は sso-app-demo namespace のすべてのリソースを表示できます。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
```

4.1.2. HTTPS および JGroups キーストアの作成、および Red Hat Single Sign-On サーバーのトラストストアの作成

passthrough TLS termination を使用する Red Hat Single Sign-On アプリケーションテンプレートには、以下が必要です。

- 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する Red Hat Single Sign-On サーバートラストストア

OpenShift イメージが適切にデプロイされる Red Hat Single Sign-On。

重要

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 **要求** も **期待** しません。テンプレートは OpenShift の内部 サービス提供の x509 証明書シークレットを使用して、HTTPS および JGroups キーストアを自動的に作成します。これらのクラスター証明書を作成するために使用される **/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt** CA 証明書ファイルを含む 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。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設定されます。

openssl ツールキットは、HTTPS キーストアに署名する CA 証明書を生成し、Red Hat Single Sign-On サーバーのトラストストアを作成するのに使用します。**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。

警告

実稼働環境では、SSL 暗号化接続(HTTPS)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。

自己署名または購入した SSL 証明書でキーストアを作成する方法は、JBoss Enterprise Application Platform セキュリティーガイド を参照してください。

HTTPS キーストアを作成します。

a. 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の署名時に、同一の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"
```

b. 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指定します。

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

c. 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指定します。

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

d. 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 に使用したのと同じパスワードを指定します。

```
$ openssl x509 -req -CA xpaas.crt -CAkey xpaas.key -in sso.csr -out sso.crt -days 365 -
CAcreateserial
```

e. 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指定します。**yes to Trust this certificate? [no]:** 質問：

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

f. 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指定します。

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

JGroups キーストアのセキュアなキーを生成します。

キーストアパスワードとして **password** 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 をトラストストアパスワードとして指定します。**yes to Trust this certificate? [no]:** 質問：

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

4.1.3. Secret

OpenShift は シークレット と呼ばれるオブジェクトを使用してパスワードやキーストアなどの機密情報を保持します。

1. 前のセクションで 生成される 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のシークレットを作成します。

   ```
   $ 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
   ```

4.1.4. OpenShift CLI を使用した Chosen Red Hat Single Sign-On パススルー TLS テンプレートのデプロイ

上記の キーストア および シークレット の作成後に、利用可能な passthrough TLS termination の一部を以下のようにデプロイします。

警告

簡単にするために、次のコマンド の **SSO_ADMIN_USERNAME**、**SSO_ADMIN_PASSWORD**、**HTTPS_PASSWORD**、**JGROU** および **SSO_TRUSTSTORE_PASSWORD** 変数の値が、Red Hat Single Sign-On アプリケーションテンプレート **sso75-openj9-https** の各パラメーターのデフォルト値に一致するように選択されています。

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に対して、十分な強固なユーザー名とパスワードを生成する方法、Red Hat Single Sign-On サーバーのトラストストア、および Red Hat Single Sign-On サーバーのトラストストアを生成する方法について、組織に固有のオンサイトポリシーを参照することが推奨されます。

テンプレートのプロビジョニング時に提供されたパスワードは、キーストアの作成時に提供されたパスワードと一致する必要があることに注意してください。異なるユーザー名とパスワードを使用する場合には、それぞれの テンプレートパラメーター の値をお使いの環境に合わせて変更します。

注記

Java Development Kit に含まれるパッケージである keytool を使用する以下のコマンドを使用すると、証明書に関連する名前を確認できます。

```
$ 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**、**および SSO_REALM** テンプレートパラメーターは任意です。

```
$ oc new-app --template=sso74-openj9-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/sso74-openj9-https" to project sso-app-demo

     Red Hat Single Sign-On 7.4.10.GA (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.
```

4.2. 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が **default** です。このプロバイダーは、現在非推奨となった **元のリクエスト** プロバイダーで強化された機能を提供します。追加の設定がない場合、要求ヘッダーを使用して元のリクエスト **プロバイダー** と同様のホスト名を判断します。

デフォルト プロバイダーの設定オプションは、『サーバーの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

後方互換性のために、SSO **_HOSTNAME も設定されている場合、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 **が修正され、固定の** ホスト名を設定できます。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。

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。

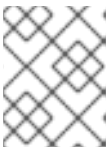
1. **SSO_HOSTNAME** 環境変数で Red Hat Single Sign-On for OpenShift イメージをデプロイすると、Red Hat Single Sign-On サーバーの必要なホスト名に設定されます。

```
$ oc new-app --template=sso74-openj9-x509-https \
  -p SSO_HOSTNAME="rh-sso-server.openshift.example.com"
```

2. Red Hat Single Sign-On サービスのルートの名前を特定します。

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

3. **host:** フィールドを、上記の **SSO_HOSTNAME** 環境変数の値として指定された hostname に変更します。



注記

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

```
$ 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
```

4.3. 外部データベースへの接続

Red Hat Single Sign-On は、外部の（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、適切なアドレスを参照するように **sso-{database name}** エンドポイントオブジェクトを変更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： 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のが最も簡単な方法です。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。

4.4. 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 を適切に設定する必要があります。現在、イメージで利用可能な唯一の JDBC ドライバーは PostgreSQL JDBC ドライバーです。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。

1. 空のディレクトリーを作成します。
2. JDBC ドライバーの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しい **Dockerfile** ファイルを作成します。その他のデータベースの場合は、**mariadb-java-client-2.5.4.jar** を該当するドライバーのファイル名に置き換えます。

```
FROM rh-sso-7/sso74-openj9-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 .
```

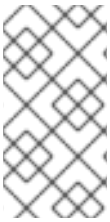
6. ビルドが完了したら、イメージを OpenShift で使用するレジストリーにプッシュしてイメージをデプロイします。詳細は、OpenShift ガイドを参照してください。

4.5. 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、追加設定なしで事前設定された管理アカウントを提供しません。この管理者アカウントは、**マスターレルム** の管理コンソールにログインし、レルムまたはユーザーの作成、Red Hat Single Sign-On によってセキュア化される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。

管理者アカウントを作成できます。

- SSO_ADMIN_USERNAME と SSO_ ADMIN_PASSWORD パラメーターの値 を提供することにより、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、または
- 特定の Red Hat Single Sign-On Pod へのリモートシェルセッション により、Red Hat Single Sign-On for OpenShift イメージがアプリケーションテンプレートなしでデプロイされます。



注記

Red Hat Single Sign-On では、Welcome Page の Web フォームで最初の管理者アカウントを作成できますが、Welcome Page が localhost からアクセスされる場合に限ります。この方法では、管理者アカウントの作成は、OpenShift イメージの Red Hat Single Sign-On には適用されません。

4.5.1. 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、SSO _ **ADMIN_USERNAME および SSO_ADMIN_PASSWORD** パラメーターは、**マスターレルム** 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およびパスワードを示します。

注記

これらの両方のパラメーターが必要です。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。

重要

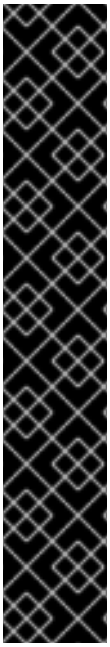
Red Hat Single Sign-On サーバーの管理者アカウントのライフサイクルは、Red Hat Single Sign-On サーバーのデータベースを保存するために使用するストレージタイプによって異なります。

- インメモリーデータベースモード (**sso75-openj9-https** テンプレートおよび **sso75-openj9-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 (**sso75-openj9-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。
- 永続的データベースモード (**sso75-openj9-postgresql-persistent** および **sso75-openj9-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の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、アプリケーションの対応する 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できます（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ります）。

警告

一時データベースモードの場合は、RH_SSO サーバーの管理者アカウントの作成後に、 デプロイメント設定から SSO_ADMIN_ **USERNAME および SSO_ADMIN_PASSWORD 変数を削除してから、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 SSO_ADMIN_** USERNAME および SSO_ADMIN_PASSWORD 変数を削除します。



重要

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** 変数設定を消去します。

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

4.5.2. リモートシェルセッションを使用した Red Hat Single Sign-On Pod への管理者アカウントの作成

Red Hat Single Sign-On アプリケーション Pod が起動した後、イメージストリームから (テンプレートを使用しないで) Red Hat Single Sign-On for OpenShift イメージを直接デプロイする際に、以下のコマンドを実行して、**master** レルムに対して管理者アカウントを作成します。

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' / 'sso_password' 認証情報はデモのみを目的としています。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、**マスターレルム** の Red Hat Single Sign-On サーバー管理者アカウントなしに、コンテナー全体を再起動すると、ゼロから再作成されるためです。

5. こ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 **マスターレルム** 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 **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 **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およびパスワードを指定します。

4.6. 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の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

本書では、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。

- **sso74-openj9-postgresql**
- **sso74-openj9-postgresql-persistent**
- **sso74-openj9-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、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、以下のように変数を設定します。

```
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
```

注記

上記のコマンドでは、PostgreSQL テンプレートは Red Hat Single Sign**-On for OpenShift イメージのデプロイに使用されたため、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、以下のように変数を設定します。

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

3. Red Hat Single Sign-On Pod をスケールアップします。

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

4. 選択した TechPreview 機能をテストします。

4.7. デプロイメントプロセス

デプロイが完了すると、**sso75-openj9-https** テンプレートおよび **sso75-openj9-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 Pod を作成します。sso75-openj9-postgresql、sso75-openj9-postgresql-persistent、および **sso75-openj9-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 サーバーの場合
- **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、**マスターレルム** の管理コンソールにログインします。

4.8. RED HAT SINGLE SIGN-ON クライアント

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され、サーバー設定を必要としない JavaScript アプリケーションには有用です。
- **機密**: ブラウザーログインの実行に必要な EAP Web アプリケーションなどのサーバー側のクライアントには便利です。
- **bearer-only**: ベアラートークン要求を許可するバックエンドサービスに便利です。

これは、アプリケーションの **web.xml** ファイルの **<auth-method>** キーに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** のスニペット例です。

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

4.8.1. 自動および手動の Red Hat Single Sign-On クライアント登録方法

クライアントアプリケーションは、eap **64-sso-s2i、eap71-sso-s2i、および datavirt63-secure-s2i テンプレートに固有の変数を使用して、 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。

4.8.1.1. Red Hat Single Sign-On クライアントの自動登録

Red Hat Single Sign-On クライアントの自動登録は、eap **64-sso-s2i、eap71-sso-s2i、および datavirt63-secure-s2i テンプレートに固有の Red Hat Single Sign-**On 環境変数によって決定されます。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 環境変数は次のとおりです。

変数	説明

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を使用する場合には空白にします。
HOSTNAME_HTTPS	https サービスルートのカスタムのホスト名。<application-name>.<project>.<default-domain-suffix> のデフォルトホスト名を使用する場合には空白にします。
SSO_URL	Red Hat Single Sign-On の Web サーバー認証アドレス： 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。
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 CORS(Cross-Origin Resource Sharing)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。

変数	説明
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た自動クライアント登録方法については、「OpenID-Connect Client 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。

4.8.1.2. 手動 Red Hat Single Sign-On クライアント登録

手動による Red Hat Single Sign-On クライアント登録は、クライアントアプリケーションの **../configuration/** ディレクトリーにデプロイメントファイルが存在することによって決定されます。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、アプリケーションがデプロイされる際に **standalone-openshift.xml の Red Hat Single Sign-** On アダプター設定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、Red Hat Single Sign-On アダプター設定を含むようにし、デプロイメントの **standalone-openshift.xml** ファイルに追加するか、または
- OpenID-Connect **keycloak.json** ファイルまたは SAML **keycloak-saml.xml** ファイルをクライアントアプリケーションの **../WEB-INF** ディレクトリーに手動で組み込みます。

SAML クライアント を使用した手動 Red Hat Single Sign-On クライアント登録メソッドのエンドツーエンドの SAML クライアントの使用例を参照するワークフローの例： Red Hat Single Sign-On 認証を使用するようにアプリケーションを手動で設定する。

4.9. 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するようになりました。『サーバー管理ガイド』を参照してください。以下の例は、OpenShift で平文の vault を設定し、SMTP パスワードの取得に使用するように設定する方法を示しています。

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。**SSO_VAULT_DIR** 環境変数をデプロイメント設定の環境に直接導入できます。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

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設定されます。

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、ボリュームをデプロイメント設定にマウントします。すでに実行されている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 **mySMTPsswd** 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 レルム内で ${vault.smtp-password}** を使用し、使用すると **mySMTPPswd** に置き換えられます。

4.10. 制限事項

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 が OpenShift の認証ゲートウェイとして使用される場合、Red Hat Single Sign-On で作成されたユーザーには、OpenShift Administrator **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>
```

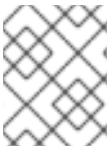
第5章 チュートリアル

5.1. ワークフローの例： 既存のデータベースを更新して、OPENSHIFT イメージバージョンの新しい RED HAT SINGLE SIGN-ON に移行する

重要

- 以前のバージョンの Red Hat Single Sign-On for OpenShift からバージョン 7.4.10.GA へのローリングアップデートは、データベースおよびキャッシュが後方互換性がないためです。
- アップグレードの前に、OpenShift の以前のバージョンの Red Hat Single Sign-On を実行するすべてのインスタンスを停止します。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。
- 事前に生成されたスクリプトは利用できず、データベースに応じて動的に生成されます。

Red Hat Single Sign-On 7.4.10.GA は、データベーススキーマを自動的に移行 するか、手動 で行うこともできます。



注記

デフォルトでは、Red Hat Single Sign-On 7.4.10.GA 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。

5.1.1. 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。https://access.redhat.com/documentation/ja-jp/red_hat_single_sign-on/7.4/html-single/red_hat_single_sign-on_for_openshift_on_eclipse_openj9/#Example-Deploying-SSO

重要

Red Hat Single Sign-On 7.4.10.GA にアップグレードする前に、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。これは、同じデータベースに対して同時に実行できないためです。

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 namespace 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
```

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 7.4.10.GA 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": "sso74-openj9-openshift-
rhel8:7.4"}]'
"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をもとに、新しい Red Hat Single Sign-On 7.4.10.GA 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 7.4.10.GA コンテナーをデプロイします。

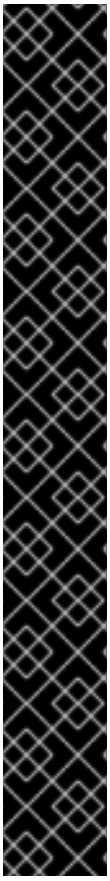
```
$ oc scale --replicas=1 dc/sso
deploymentconfig "sso" scaled
```

6. （オプション）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
```

5.1.2. データベースの手動移行



重要

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。Red Hat Single Sign-On 7.4.10.GA を使用すると、後でデータベースに手動で適用できる SQL ファイルにこれらのファイルを生成およびエクスポートできます。データベースの SQL 移行ファイルを動的に生成するには、以下を実行します。

1. Red Hat Single Sign-On 7.4.10.GA を正しいデータソースで設定します。
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**, and
    c. Pod のファイルシステム上の場所への **migrationExport**。出力 SQL 移行ファイルを保存する必要があります（例：**migrationExport="${jboss.home.dir}/keycloak-database-update.sql"**）。

詳細は、「Red Hat Single Sign-On 7.4.10.GA のデータベース設定 」を参照してください。

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、データの操作を実施するため、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 イメージを実行しているすべての Pod を停止します。

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。https://access.redhat.com/documentation/ja-jp/red_hat_single_sign-on/7.4/html-single/red_hat_single_sign-on_for_openshift_on_eclipse_openj9/#Example-Deploying-SSO

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、取得します。

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

```
$ cat job-to-migrate-db-to-sso74-openj9.yaml.orig
apiVersion: batch/v1
kind: Job
metadata:
  name: job-to-migrate-db-to-sso74-openj9
spec:
  autoSelector: true
  parallelism: 0
  completions: 1
  template:
    metadata:
      name: job-to-migrate-db-to-sso74-openj9
    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-sso74-openj9
      # 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-sso74-openj9.yaml.orig \
    job-to-migrate-db-to-sso74-openj9.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-sso75-openj9.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** 変数の値と、特定のデータソース固有の環境変数の値 (**<PREFIX>_JNDI**、**<PREFIX>_USERNAME**、**<PREFIX>_PASSWORD**、および **<PREFIX>_DATABASE**) をコピーします。

注記

DB_SERVICE_PREFIX_MAPPING 環境変数では、コンマ区切りの **<name>-<database_type>=<PREFIX>** トリプレットのリスト を使用できますが、このスクリプト例では、デモの目的で 1 つのデータソーストリプレットのみを受け入れます。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-sso74-openj9.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 7.4.10.GA 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 sso74-openj9 | cut -d ' ' -f1
sso74-openj9-openshift-rhel8
```

```
$ oc new-build sso74-openj9-openshift-rhel8:7.4~https://github.com/iankko/openshift-examples.git#KEYCLOAK-8500 \
  --context-dir=sso-manual-db-migration \
  --name=sso74-openj9-db-migration-image
--> Found image bf45ac2 (7 days old) in image stream "openshift/sso74-openj9-openshift-rhel8" under tag "7.4" for "sso74-openj9-openshift-rhel8:7.4"

    Red Hat SSO 7.4.10.GA
    ---------------
    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 "sso74-openj9-db-migration-image:latest"
      * Use 'start-build' to trigger a new build

--> Creating resources with label build=sso74-openj9-db-migration-image ...
    imagestream "sso74-openj9-db-migration-image" created
    buildconfig "sso74-openj9-db-migration-image" created
--> Success
    Build configuration "sso74-openj9-db-migration-image" created and build triggered.
    Run 'oc logs -f bc/sso74-openj9-db-migration-image' to stream the build progress.
```

```
$ oc logs -f bc/sso74-openj9-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-sso75-openj9.yaml**) をビルドされた **sso75-openj9-db-migration-image** イメージへの参照で更新します。

   a. イメージの docker プルリファレンスを取得します。

   ```
   $ PULL_REF=$(oc get istag -n $(oc project -q) --no-headers | grep sso74-openj9-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-sso74-openj9.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-sso74-openj9.yaml
job "job-to-migrate-db-to-sso74-openj9" 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、データの操作を実施するため、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 イメージを実行しているすべての 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 namespace 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
```

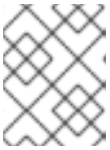
8. データベースの移行ジョブを実行し、Pod が正しく実行されるのを待機します。

```
$ oc get jobs
NAME                          DESIRED   SUCCESSFUL   AGE
job-to-migrate-db-to-sso74-openj9   1         0          3m
```

```
$ oc scale --replicas=1 job/job-to-migrate-db-to-sso74-openj9
job "job-to-migrate-db-to-sso74-openj9" scaled
```

```
$ oc get pods
NAME                          READY     STATUS      RESTARTS   AGE
sso-postgresql-1-n5p16              1/1       Running     1          19h
```

```
job-to-migrate-db-to-sso74-openj9-b87bb   1/1       Running     0          1m
sso74-openj9-db-migration-image-1-build   0/1       Completed   0          27m
```



注記

デフォルトでは、データベースの移行ジョブは、移行ファイル **の生成後に 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-sso74-openj9-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 7.4.10.GA 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。

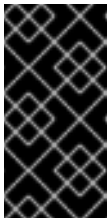
```
$ oc rsync --no-perms=true ./db-update/ sso-postgresql-1-n5p16:/tmp
sending incremental file list

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認証情報に置き換えます。また、& **lt;PREFIX>_PASSWORD** の値を要求されたら、データベースのパスワードとして使用します。

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ステップの更新」に進んでください。

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 7.4.10.GA 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": "sso74-openj9-openshift-
rhel8:7.4"}]'
"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をもとに、新しい Red Hat Single Sign-On 7.4.10.GA 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 7.4.10.GA コンテナーをデプロイします。

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

5.2. ワークフローの例： RED HAT SINGLE SIGN-ON サーバーの環境全体で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。「 OpenShift Deployment 向けの Red Hat Single Sign-On 認証の準備 」に記載されている手順がすでに実行されていることを前提とします。

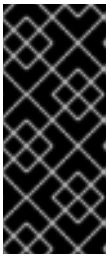
5.2.1. Red Hat Single Sign-On PostgreSQL アプリケーションテンプレートのデプロイ

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、リストを **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. Red Hat Single Sign-On アプリケーションテンプレート **sso75-openj9-postgresql** を選択します。テンプレートをデプロイする際に **SSO_REALM 変数の設定を解除し続けます（** デフォルト値）。



重要

Red Hat Single Sign-On 7.4.10.GA 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 つの Red Hat Single Sign-On サーバー起動時に可能な移行アクション（**エクスポート またはインポート**）のいずれかのみを実行することを意味します。

警告

SSO_REALM 設定変数が Red Hat Single Sign-On for OpenShift イメージに設定されている場合、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、SSO **_REALM** 設定変数をこのようなイメージに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。
次に、管理者アカウント を使用して **https://secure-sso-<sso-app-demo>.<openshift32.example.com> /auth/admin** にある Red Hat Single Sign-On 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、非セキュアな接続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。

5.2.2. （オプション）追加の Red Hat Single Sign-On レルムおよびユーザーも エクスポートする

Red Hat Single Sign-On 7.4.10.GA 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存在する Red Hat Single Sign-On 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、以下を最初に作成する必要があります。

1. レルムを新規作成します。
2. 新規ユーザーの作成

データベースの作成時には、データベースをエクスポートできます。

5.2.3. 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。

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 7.4.10.GA サーバーに、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をスケールアップします。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. （オプション）エクスポートが正常に完了したことを確認します。

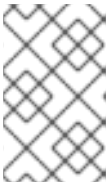
```
$ oc get pods
NAME                 READY     STATUS    RESTARTS   AGE
sso-4-ejr0k          1/1       Running   0          27m
sso-postgresql-1-ozzl0   1/1       Running   0          4h

$ 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
```

5.2.4. 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。

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. （オプション）Red Hat Single Sign-On データベースの JSON ファイルを、別の環境で実行されている Red Hat Single Sign-On サーバーにインポートします。



注記

OpenShift 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には、『RH SSO Server Administration Guide』 の「Export and Import 」セクションを参照してください。

Red Hat Single Sign-On サーバーが OpenShift の Red Hat Single Sign-On 7.4.10.GA コンテナーとして実行されている場合、Red Hat Single Sign-On サーバーの 管理コンソール 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 **マスターレルム** 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 **http://sso-<project-name>.<hostname>/auth/admin** に移動します。または、暗号化された Red Hat Single Sign-On Web サーバーの **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。

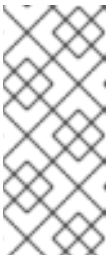
c. サイドバーの下部にある **Manage** セクションにある **Import** リンクをクリックします。

d. 開いているページで **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した **demorealm-export.json** JSON ファイルの場所を指定します。

e. **レルムからインポート** ドロップダウンメニューから、データをインポートする必要のある 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（デフォルトではインポートされます）。

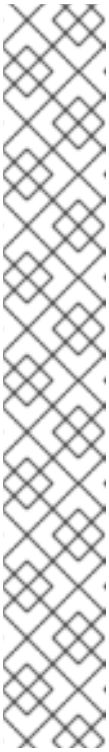
g. リソースがすでに存在する場合に、実行するストラテジーを選択します（ **Fail**、**Skip**、または Overwrite **のいずれか**）。



注記

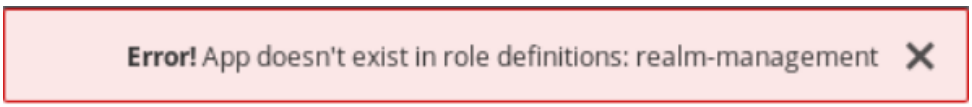
同じ識別子を持つオブジェクトが現在のデータベースにすでに存在すると、オブジェクト（ユーザー、クライアント、レルムロール、またはクライアントロール）のインポートに失敗します。**Skip** strategy を使用して **demorealm-export.json** ファイルにあるオブジェクトをインポートしますが、現在のデータベースには存在しません。

h. **Import** をクリックしてインポートを実行します。



注記

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。

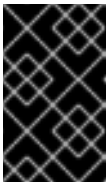


このような場合は、最初に **Access Type** を **bearer-only** に設定し、不足しているクライアントを作成する必要があります。これらのクライアントは、エクスポート JSON ファイルが作成されたソースの Red Hat Single Sign-On サーバーから、JSON ファイルがインポートされるターゲット Red Hat Single Sign-On サーバーに、その特性を手動でコピーして作成できます。必要なクライアントを作成したら、もう一度 **Import** ボタンをクリックします。

上記の エラーメッセージを抑制するには、不足している **realm-management** クライアント（ **ベアラーのみの アクセスタイプ** ）を作成し、**Import** ボタンを再びクリックする必要があります。

注記

Skip import ストラテジーの場合、新たに追加されたオブジェクトは **ADDED** とマークされ、省略されたオブジェクトは、インポート結果ページの **Action** コラムで **SKIPPED** とマークされます。



重要

管理コンソールのインポートにより、選択した場合はリソース **を上書きできます（上書き 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。

5.3. ワークフローの例： 認証に RED HAT SINGLE SIGN-ON を使用するように OPENSHIFT 3.11 を設定

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。これは 、ワークフローの例： Red Hat Single Sign-On が OpenShift にデプロイされた Red Hat Single Sign-On for OpenShift イメージの準備とデプロイ

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 アイデンティティープロバイダー と共に Red Hat Single Sign-On を認証方法として追加します。これを設定すると、OpenShift Web コンソールにログインするために、Red Hat Single Sign-On メソッド（設定されたアイデンティティープロバイダーとともに）も利用できます。

5.3.1. Red Hat Single Sign-On 認証情報の設定

Red Hat Single Sign-On の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-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 namespace（デフォルトは **Master**）にカーソルを合わせ、**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。

ユーザーを作成します。

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** sidebar 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** （この例では **testuser**）と追加の任意の情報を入力し、**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。

   b. **Temporary Password** オプションが **Off** に設定され、後でパスワード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より、追加の確認が求められます。

OpenID-Connect クライアントの作成および設定

詳細は、『Red Hat Single Sign-On サーバー管理ガイド』の「 クライアントの管理 」の章を参照してください。

1. **Manage** サイドバーの **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 を使用しています**。

3. ドロップダウンメニュー（この例では **openid-connect**を使用）から **Client Protocol** を選択し、**Save** をクリックします。**openshift-demo** クライアントの設定 **設定** 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 を入力します。この URI は **https://openshift.example.com:8443/*** になります。

次のセクションの 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。これで **Credentials** タブからコピーできます。シークレットは <**7b0384a2-b832-16c5-9d73-2957842e89h7**> になります。

5.3.2. OpenShift マスターでの Red Hat Single Sign-On 認証の設定

OpenShift マスター CLI にログインします。**/etc/origin/master/master-config.yaml** ファイルを編集するのに必要なパーミッションが必要です。

1. **/etc/origin/master/master-config.yaml** ファイルを編集し、**identityProviders** を検索します。たとえば、OpenShift マスターが HTPassword アイデンティティープロバイダー で設定されている場合には、**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(**Clients → openshift-demo → Credentials**)を参照してください。

b. url のエンドポイントは、Red Hat Single Sign-On アプリケーションで要求することで **見つけることができます**。以下は例になります。

```
<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
```

5.3.3. OpenShift へのログイン

OpenShift Web コンソールに移動します。この例では **https://openshift.example.com:8443/console** です。OpenShift ログインページには、htpasswd **_auth** または **rh-sso** のいずれかを使用するオプションが含まれるようになりました。前者は **/etc/origin/master/master-config.yaml** に存在するため、引き続き利用できます。

rh-sso を選択し、Red Hat Single Sign-On で作成した **testuser** ユーザーで OpenShift にログインします。**testuser は、OpenShift** CLI に追加されるまでプロジェクトを表示できません。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。

sso-app-demo に **testuser view** 権限を指定するには、OpenShift CLI を使用します。

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

5.4. ワークフローの例： 既存の MAVEN BINARIES から OPENSHIFT アプリケーションを作成し、RED HAT SINGLE SIGN-ON を使用してこれのセキュリティーを保護する

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

5.4.1. 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

以下の例では、app-jee-jsp と service-jee-jaxrs の両方のクイックスタートを使用して、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:

重要

本ガイドでは、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。

- **sso74-openj9-postgresql**
- **sso74-openj9-postgresql-persistent**
- **sso74-openj9-x509-postgresql-persistent**

5.4.1.1. 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 の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 namespace（デフォルトは **Master**）にカーソルを合わせ、**Add Realm** をクリックします。

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

公開鍵のコピー

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

注記

OpenShift イメージバージョン 7.4.10.GA 用の Red Hat Single Sign-On は、デフォルトで複数のキーを生成します（ **HS256**、**RS256**、**AES** など）。OpenShift 7.4.10.GA イメージの Red Hat Single Sign-On 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、キーテーブルのその行の **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 が有効な EAP 6.4 / 7.1 JSP アプリケーション をデプロイするには、公開鍵に関する情報が必要です。

Red Hat Single Sign-On ロールの作成

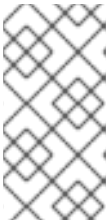
注記

service-jee-jaxrs クイックスタートは、サービスが 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** - ユーザー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 でユーザー および管理 ロールを作成します。これらのロールは、ユーザーアプリケーションへのアクセスを認証するために Red Hat Single Sign-On アプリケーションユーザーに割り当てられます。

1. **Configure サイドバーで 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、デフォルトの（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（ユーザー）を入力し、Save をクリックします。

admin ロールについてこ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 sidebar 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 （ この例ではユーザー appuserを使用）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を使用します）。

   b. Temporary Password オプションが Off に設定され、後でパスワード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。

5.4.1.2. レルム 管理ユーザーへのユーザー Red Hat Single Sign-On ロールの割り当て

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

1. Role Mappings をクリックして、レルムおよびクライアントロールの設定を一覧表示します。Available Roles で、先に作成した ユーザーロール 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 一覧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 1 度に選択できます。Ctrl キーと左マウスボタンを押さずに、リストの最後に移動し、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。

5.4.1.3. 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

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 default 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は eap-app-demo namespace のすべてのリソースを表示できます。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である 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 -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
```

5.4.1.4. 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

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
```

2. Configure the Red Hat JBoss Middleware Maven Repository

3. service-jee-jaxrs および 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] ------------------------------------------------------------------------
```

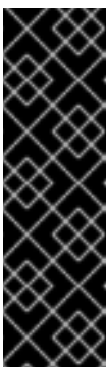
b. maven-enforcer -plugin プラグインの app-jee-jsp/config/keycloak.json 要件 をコメントアウトし、app-jee-jsp アプリケーションをビルドします。

```
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を設定し、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にある config/ ディレクトリーに存在する必要があります。しかし、この例では EAP 6.4 / 7.1 for OpenShift イメージで利用可能な選択した環境変数を介してアダプターを設定するため、現時点では 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 上にビルドされる イメージの標準デプロイメント 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 と app-jee- jsp クイックスタートの以前にビルドされた 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。

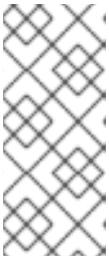
表5.1 デプロイメントディレクトリーの標準的な場所

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 および 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

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

```
$ 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 7.3
    -------------
    Platform for building and running JavaEE applications on JBoss EAP 7.3

    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は、以下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 JavaEE applications on JBoss EAP 7.3

    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 変数は、コピーされた demo レルムの RSA 公開鍵の実際のコンテンツ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
hXyKx97oloO6HwnV/MiX2EHO55Sn+ydsPzbjJevl5F31UvUco9uA8dGl6oM8HrnaW
Wv+i8PvmlaRMhhl6Xs68vJTEc6d0soP+6A+aExw0coNRp2PDwvzsXVWPvPQg3+iyt
Stxu3lcndx+gC0ZYnxoRqL7rY7zKcQBScGEr78Nw6vZDwfe6d/PQ6W4xVErNytX9Ky
LFVAE1VvhXALyqEM/EqYGLmpjw5bMGVKRXnhmVo9E88CkFDH8E+aPiApb/gFul1
GJOv+G8ySLoR1c8Y3L29F7C81odkVBp2yMm3RVFlGSPTjHqjO/nOtqYlfY4Wyw9m
RloY5SyW7044dZXRwIDAQAB" \
-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、デフォルトの 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（デフォルトの設定）。

```
$ 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
```

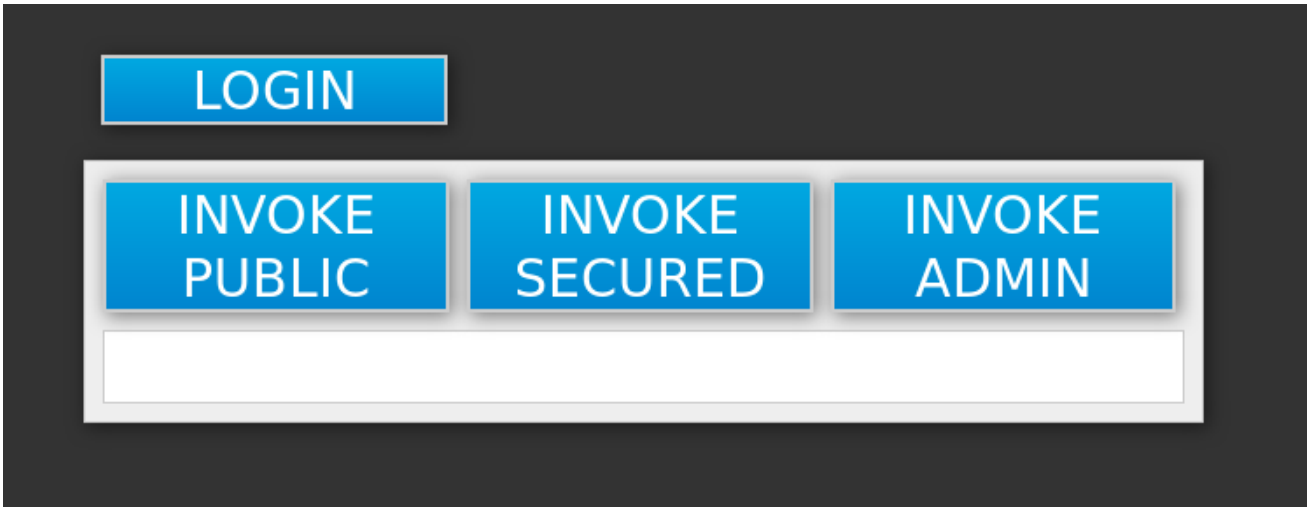
```
$ oc get route
No resources found.
```

```
$ 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
```

5.4.1.5. アプリケーションへのアクセス

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。以下の図のような出力が表示されるはずです。



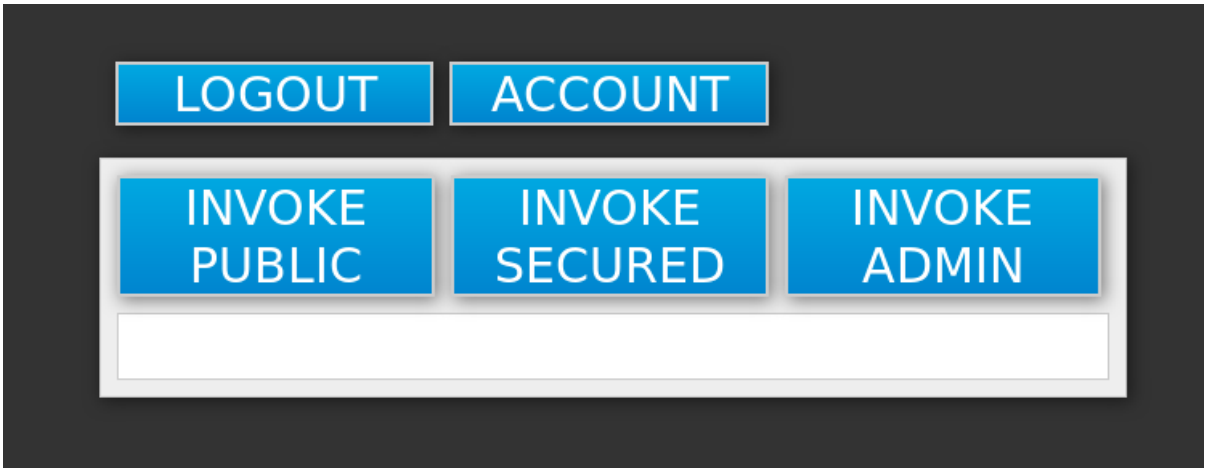
アプリケーションをテストするには、以下を実行します。

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

  Message: public 出力が表示されるはずです。

- **LOGIN** ボタンをクリックし、**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。

  先に設定した **Red Hat Single Sign-On** ユーザーのユーザー名およびパスワード(**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。



- INVOKE SECURED ボタンをクリックして、セキュアなエンドポイント にアクセスします。

  Message: secured 出力が表示されるはずです。

- INVOKE ADMIN ボタンをクリックして 管理エンドポイント にアクセスします。

  403 Forbidden 出力が表示されるはずです。

注記

管理エンドポイント 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、ユーザーロールの権限しか持たず、セキュアなエンドポイント へのアクセスを可能にするため、禁止されています。

appuser を admin Red Hat Single Sign-On ロールに追加するには、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 の管理者ユーザーの認証情報 を使用します。

2. Manage sidebar 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. アプリの ID リンクをクリックします 。 または、アクション コラムの 編集 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 の行にある Available Roles 一覧から admin エントリーを選択します。

7. Add selected> ボタンをクリックして 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。

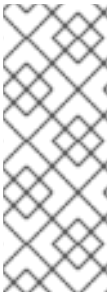
10. LOGIN ボタンを再度クリックし、プロバイダーアプリの 認証情報 をクリックします。

11. INVOKE ADMIN ボタンをもう一度クリックします。

    Message: admin 出力がすでに表示されます。

5.5. ワークフローの例： OPENID-CONNECT CLIENT 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

これは、ワークフローの例： Red Hat Single Sign-On が OpenShift にデプロイされた Red Hat Single Sign-On for OpenShift イメージの準備とデプロイ以下の例では、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。

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

cluster:admin ロールを保持するユーザーで 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 default 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は eap-app-demo namespace のすべてのリソースを表示できます。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である 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。次のコマンドは、パスワードを要求します。

   a. SSL キーストアのセキュアなキーを生成します。

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

   b. JGroups キーストアのセキュアなキーを生成します。

```
$ 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 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
```

5.5.2. Red Hat Single Sign-On 認証情報の準備

Red Hat Single Sign-On の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にカーソルを合わせ、Add Realm をクリックします。

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

公開鍵のコピー

新たに作成した 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、brevity 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。

ロールを作成します。

EAP アプリケーションの web.xml に定義されている JEE ロールに対応する名前で Red Hat Single Sign-On にロールを作成します。このロールは、ユーザーアプリケーションへのアクセスを認証するために Red Hat Single Sign-On アプリケーションユーザー に割り当てられます。

1. Configure サイドバーで 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（この例ではロール eap-user-roleを使用します）を入力し、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

2 人のユーザーを作成： Red Hat Single Sign -On サーバーで Red Hat Single Sign-On クライアントの自動登録を処理する レルム 管理ユーザー を割り当てます。- 前の手順で作成した アプリケーションユーザーに JEE ロールを割り当てて、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 sidebar 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 （ この例ではユーザー eap-mgmt-userを使用）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て確認します。Temporary を OFF に設定します。Reset Password をクリックして、ユーザーのパスワードを設定します。ポップアップウィンドウにより、追加の確認が求められます。**

5. **Role Mappings をクリックして、レルムおよびクライアントロールの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 sidebar 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** と、アプリケーションユーザーの 追加のオプション情報を入力して、**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。パスワードを確認したら、**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より、追加の確認が求められます。

5. **Role Mappings** をクリックして、レルムおよびクライアントロールの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。

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

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、リストを **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、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

変数	値の例
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 イメージのデプロイには数分の時間がかかる場合があります。

5.5.4. Red Hat Single Sign-On を使用して JBoss EAP サーバーにログインします。

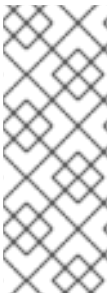
1. **JBoss EAP アプリケーションサーバーにアクセスし、Login をクリックします。Red Hat Single Sign-On ログインにリダイレクトされます。**

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。**

5.6. ワークフローの例： SAML クライアントでの RED HAT SINGLE SIGN-ON での EAP アプリケーションの手動登録

これは 、ワークフローの例： Red Hat Single Sign-On が OpenShift にデプロイされた Red Hat Single Sign-On for OpenShift イメージの準備とデプロイ

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。

5.6.1. Red Hat Single Sign-On 認証情報の準備

Red Hat Single Sign-On の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 namespace（デフォルトは Master）にカーソルを合わせ、Add Realm をクリックします。

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

公開鍵のコピー

新たに作成した 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、brevity 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。

ロールを作成します。

EAP アプリケーションの web.xml に定義されている JEE ロールに対応する名前で Red Hat Single Sign-On にロールを作成します。このロールは、ユーザーアプリケーションへのアクセスを認証するために Red Hat Single Sign-On アプリケーションユーザー に割り当てられます。

1. Configure サイドバーで 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（この例ではロール saml-user-roleを使用します）を入力し、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

2 人のユーザーを作成： Red Hat Single Sign -On サーバーで Red Hat Single Sign-On クライアントの自動登録を処理する レルム 管理ユーザー を割り当てます。- 前の手順で作成した アプリケーションユーザーに JEE ロールを割り当てて、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 sidebar 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 （ この例ではユーザー app-mgmt-userを使用）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。パスワードを確認したら、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より、追加の確認が求められます。

アプリケーションユーザーを作成します。

1. Manage sidebar 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 と、アプリケーションユーザーの 追加のオプション情報を入力して、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。パスワードを確認したら、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より、追加の確認が求められます。

5. Role Mappings をクリックして、レルムおよびクライアントロールの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。このセクションでは、手順の後半に必要な keystore.jks および keycloak-saml-subsystem.xml の 2 つの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、レルムに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 Client ID を入力します。この例では sso-saml-demo を使用します。

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 Root URL を入力します。この例では 、https://demoapp-eap-app-demo.openshift32.example.com を使用しています。

5. 保存 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、ルート URL と新しい sso-saml-demo クライアントの 有効なリダイレクト URL を設定します。

1. Root URL に、クライアントの作成時に使用される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 を使用しています。

2. 有効なリダイレクト URL については、ユーザーのログイン時にリダイレクトする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/* に対するリダイレクトアドレスを使用します。

SAML キー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、Archive Format を JKS のままにします。この例では、saml-demo の sso-saml-demo およびデフォルトのレルム 証明書エイリアス のデフォルト キーエイリアスを使用します。

3. Key Password と Store Password を入力します。この例では、両方の パスワードを使用します。

4. Download をクリックして、後で使用できるように keystore-saml.jks ファイルを保存します。

5. sso-saml-demo クライアントをクリックし、次の手順のために使用できるクライアントスペースに戻ります。

クライアントアダプターをダウンロードします。

1. インストール 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形式を選択します。この例では、Keycloak SAML Wildfly/JBoss サブシステム 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、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するために、次のセクションの他の EAP キーストアと使用されます。keystore-saml.jks ファイルを OpenShift ノードにコピーします。
keycloak-saml-subsystem.xml が変更され、アプリケーションのデプロイメントで使用されます。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。

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

cluster:admin ロールを保持するユーザーで 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 default 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は eap-app-demo namespace のすべてのリソースを表示できます。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である 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。次のコマンドは、パスワードを要求します。

a.
SSL キーストアのセキュアなキーを生成します。

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

b.
JGroups キーストアのセキュアなキーを生成します。

```
$ 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
```

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

前のセクションの 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 ディレクトリーにコピー済みで、secure-saml-deployments の名前が変更された必要があります。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。

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 名タグの YOUR- WAR.war の値を application .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 ページ タグの SPECIFY YOUR LOGOUT PAGE! の値を、アプリケーションからのログアウト時にユーザーをリダイレクトする url に置き換えます。この例では /index.jsp を使用します。

4.
<PrivateKeyPem> および <CertificatePem> タグおよびキーを削除し、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 （この例では /etc/eap-secret-volume/keystore-saml.jks）のマウントパスは、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 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ている場合に設定されます。

5. 2 番目の <CertificatePem> タグおよびキーを削除し、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 ファイルを保存して閉じます。

5.6.4. アプリケーション 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、アプリケーションの 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、以下が含まれていることを確認します。

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

5.6.5. 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6章 参照資料

6.1. 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 jar、ライブラリー jar、プラグイン、または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、ローカルカスタムミラーリポジトリ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、https://10.0.0.1:8443/repository/internal/ で外部アクセス可能な場合、以下のように 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することができます。

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
```

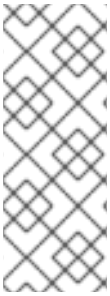
2. eap のビルド設定を MAVEN_MIRROR_URL 環境変数で更新します。

```
$ 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。

6.2. 環境変数

6.2.1. 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6.1 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/sso74-openj9-openshift-rhel8
JBOSS_IMAGE_VERSION	イメージのバージョン（"version" ラベルと同じ）	7.4
JBOSS_MODULES_SYSTEM_PKGS	-	org.jboss.logmanager,jdk.nashorn.api

6.2.2. 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6.2 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	設定された場合、(Jolokia の リファレンスマニュアル で説明されているように) (パスを含む) このファイルを Jolokia JVM エージェントプロパティーとして使用します。設定しないと、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。それ以外の場合は、本書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 です。	**true**
AB_JOLOKIA_HOST	バインド先のホストアドレス。デフォルトは 3.0.0 です。	127.0.0.1
AB_JOLOKIA_HTTPS	httpsとの安全な通信をオンにします。デフォルトでは、**AB_JOLOKIA_OPTS** で **serverCert** 設定の指定がないと、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場合、https は オフになります。値が空の文字列以外に設定されている場合は、https が有効になります。**	**true**

変数名	説明	値の例
AB_JOLOKIA_ID	使用するエージェント ID。デフォルトはコンテナー ID の ($HOSTNAME)。	openjdk-app-1-xqlsj
AB_JOLOKIA_OFF	設定されている場合、Jolokia のアクティベートを無効にします（例： 空の値）。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場合、https は オフになります。値が空の文字列以外に設定されている場合は、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。**key=value,key=value,...** の形式で表示される必要があります。	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	設定すると、AB_JOLOKIA_PASSWORD にランダムな値が生成され、**/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（デフォルトは 8778）。	5432
AB_JOLOKIA_USER	Basic 認証のユーザー。デフォルトは **jolokia** です。	myusername
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

変数名	説明	値の例
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に費やした時間に対する、ガベージコレクション外で費やした時間 (アプリケーションの実行に費やした時間など) 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、-Xms オプションは追加されません。	100
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、ここで設定されているコンテナの使用可能なメモリの比率に設定されます。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、**-Xmx** オプションは追加されません。	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

変数名	説明	値の例
MQ_SIMPLE_DEFAULT_PHYSICAL_DESTINATION	後方互換性を維持するには、**true** を設定し、**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 namespace。	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、実行と同時にコマンドとその引数が出力されます。	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
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

変数名	説明	値の例
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。設定されていない場合、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。設定されている場合、指定の変数の値に設定された Red Hat Single Sign-On サーバーのホスト名で、**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	この環境変数が指定されている場合は、Red Hat Single Sign-On サーバーで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
SSO_TRUSTSTORE_PASSWORD	トラストストアおよび証明書のパスワード。	mykeystorepass

変数名	説明	値の例
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 ボリュームに使用されます。	truststore-secret

Red Hat Single Sign-On for OpenShift で利用可能な アプリケーションテンプレートは、前述の設定変数を 共通の OpenShift 変数（ APPLICATION_NAME または SOURCE_REPOSITORY_URL など）、製品固有の変数（HORNET Q_CLUSTER_PASSWORDなど）、またはデータベースイメージ（例： POSTGRESQL_MAX_CONNECTIONS））に準拠している設定変数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、デプロイされた Red Hat Single Sign-On 対応アプリケーションが可能な限り目的のユースケースに合わせて調整できます。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。

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

表6.3 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。デフォルトホスト名の場合は空白にします (例: <application-name>-kieserver-<project>.<default-domain-suffix>。
HOSTNAME_HTTPS	https サービスルートのカスタムのホスト名。デフォルトホスト名の場合は空白にします (例: <application-name>-kieserver-<project>.<default-domain-suffix>。
HTTPS_KEYSTORE	シークレット内のキーストアファイル名HTTPS_PASSWORD および HTTPS_NAME と定義された場合、HTTPS を有効にし、SSL 証明書キーファイルを EAP_HOME/standalone/configuration 以下の相対パスに設定します。

変数	説明
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ている名前 HTTPS_PASSWORD および **HTTPS_KEYSTORE** と定義された場合、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（mykeystorepass など）。HTTPS_NAME および **HTTPS_KEYSTORE** と定義された場合、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を含むシークレット名
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 ImageStreams は通常 OpenShift の名前空間にインストールされています。ImageStream を異なる namespace/プロジェクトにインストールしている場合にのみ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名
JGROUPS_ENCRYPT_NAME	サーバー証明書に関連付けられている名前（例：**secret-key**）。
JGROUPS_ENCRYPT_PASSWORD	キーストアおよび証明書のパスワード（例：**パスワード**）。
JGROUPS_ENCRYPT_SECRET	キーストアファイルを含むシークレット名
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。
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。

変数	説明
SSO_REALM	この環境変数が指定されている場合は、Red Hat Single Sign-On サーバーで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** ボリュームに使用されます。
SSO_TRUSTSTORE_PASSWORD	トラストストアおよび証明書のパスワード。

6.2.4. sso75-openj9-postgresql、sso75-openj9-postgresql-persistent、および sso75-openj9-x509-postgresql-persistent に固有のテンプレート変数

表6.4 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	データソースを解決するためにアプリケーションによって使用されるデータベース JNDI 名（例：**java:/jboss/datasources/postgresql**）
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。
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。

6.2.5. 一般的な eap64 および eap 71 S2I イメージのテンプレート変数

表6.5 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	artifacts 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 クラスター管理者のパスワード。
HORNETQ_QUEUES	Queue name
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を使用する場合には空白にします（例：**<application-name>.<project>.<default-domain-suffix>** ）。
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を使用する場合には空白にします（例：**<application-name>.<project>.<default-domain-suffix>** ）。
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名**HTTPS_PASSWORD** および **HTTPS_NAME** と定義された場合、HTTPS を有効にし、SSL 証明書キーファイルを **EAP_HOME/standalone/configuration** 以下の相対パスに設定します。
HTTPS_NAME	サーバー証明書に関連付けられている名前 HTTPS_PASSWORD および **HTTPS_KEYSTORE** と定義された場合、HTTPS を有効にし、SSL 名を設定します。

変数	説明
HTTPS_PASSWORD	キーストアおよび証明書のパスワード（mykeystorepass など）。HTTPS_NAME および **HTTPS_KEYSTORE** と定義された場合、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を含むシークレット名
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 ImageStreams は通常 OpenShift の名前空間にインストールされています。ImageStream を異なる namespace/プロジェクトにインストールしている場合にのみ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名
JGROUPS_ENCRYPT_NAME	サーバー証明書に関連付けられている名前（例：**secret-key**）。
JGROUPS_ENCRYPT_PASSWORD	キーストアおよび証明書のパスワード（例：**パスワード**）。
JGROUPS_ENCRYPT_SECRET	キーストアファイルを含むシークレット名
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 URl。

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

表6.6 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
SSO_REALM	この環境変数が指定されている場合は、Red Hat Single Sign-On サーバーで作成されるレルムの名前。

変数	説明
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、**sso75-openj9-** 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** ボリュームに使用されます。
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	トラストストアおよび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、EAP と Red Hat Single Sign-On サーバー間の SSL 通信が安全ではない場合（つまり、証明書の検証は curl で無効になります）
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。

6.2.7. SAML クライアントでの自動 クライアント登録に使用する eap64-sso-s2i および eap71-sso-s2i に固有のテンプレート変数

表6.7 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ている名前
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名
SSO_SAML_KEYSTORE_SECRET	キーストアファイルを含むシークレット名
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。

6.3. 公開されたポート

ポート番号	説明
8443	HTTPS
8778	Jolokia の監視