Red Hat Single Sign-On 7.4

서버 설치 및 구성 가이드

Red Hat Single Sign-On 7.4와 함께 사용

Last Updated: 2023-06-13

Red Hat Single Sign-On 7.4 서버 설치 및 구성 가이드

Red Hat Single Sign-On 7.4와 함께 사용

법적 공지

초록

이 안내서는 Red Hat Single Sign-On 7.4를 설치하고 구성하는 정보로 구성되어 있습니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

1장. 가이드 개요

이 가이드의 목적은 Red Hat Single Sign-On 서버를 처음 부팅하기 전에 완료해야 하는 단계를 안내합니다. Red Hat Single Sign-On 드라이브를 테스트하려는 경우 자체 내장 및 로컬 전용 데이터베이스와 함께 거의 실행되지 않습니다. 프로덕션 환경에서 실행할 실제 배포의 경우 런타임(독립 실행형 또는 도메인 모드)에서 서버 구성을 관리하는 방법을 결정하고, Red Hat Single Sign-On 스토리지에 대한 공유 데이터베이스를 구성하고, 암호화 및 HTTPS를 설정하고 마지막으로 클러스터에서 실행되도록 Red Hat Single Sign-On을 설정해야 합니다. 이 가이드에서는 서버를 배포하기 전에 수행해야 하는 사전 부팅 결정 및 설정의 각 측면을 안내합니다.

특히, Red Hat Single Sign-On은 JBoss EAP 애플리케이션 서버에서 파생됩니다. Red Hat Single Sign-On 구성의 여러 측면은 JBoss EAP 구성 요소를 중심으로 합니다. 자세한 내용을 살펴보려면 이 가이드를 통해 설명서 외부에서 문서로 안내하는 경우가 많습니다.

1.1. 추가 추가 문서

Red Hat Single Sign-On은 JBoss EAP 애플리케이션 서버 및 Infinispan(캐스팅용) 및 Hibernate(속도용)와 같은 하위 프로젝트를 기반으로 구축됩니다. 이 가이드에서는 인프라 수준 구성의 기본 사항만 다룹니다. JBoss EAP 및 해당 하위 프로젝트에 대한 문서를 사용하는 것이 좋습니다. 문서에 대한 링크는 다음과 같습니다.

- *JBoss EAP 구성 가이드*

2장. 설치

ZIP 파일을 다운로드하고 압축을 풀거나 RPM을 사용하여 Red Hat Single Sign-On을 설치할 수 있습니다. 이 장에서는 시스템 요구 사항 및 디렉터리 구조를 검토합니다.

2.1. 시스템 요구 사항

다음은 Red Hat Single Sign-On 인증 서버를 실행하기 위한 요구 사항입니다.

- Java를 실행하는 모든 운영 체제에서 실행 가능
- Java 8 JDK
- zip 또는 gzip 및 tar
- 최소 512M의 RAM
- 최소 1G의 디스크 공간
- PostgreSQL, MySQL, Oracle 등과 같은 공유 외부 데이터베이스. Red Hat Single Sign-On에는 클러스터에서 실행하려면 외부 공유 데이터베이스가 필요합니다. 자세한 내용은 이 가이드의 데이터베이스 구성 섹션을 참조하십시오.
- 클러스터에서 실행하려는 경우 머신에서 네트워크 멀티 캐스트를 지원합니다. Red Hat Single Sign-On은 멀티 캐스트 없이 클러스터링할 수 있지만 이를 위해서는 다양한 구성 변경이 필요합니다. 자세한 내용은 이 가이드의 클러스터링 섹션을 참조하십시오.
- Linux에서는 **/dev/urandom** 을 임의의 데이터 소스로 사용하여 사용 가능한 엔트로피 부족으로 인해 Red Hat Single Sign-On이 보안 정책에서 요구하는 경우가 아니면 Red Hat Single Sign-On 중단을 방지하는 것이 좋습니다. **Oracle JDK 8 및 OpenJDK 8에서 시작 시 java.security.egd 시스템 속성을 file:/dev/urandom 로 설정합니다.**

2.2. ZIP 파일에서 RH-SSO 설치

Red Hat Single Sign-On 서버 다운로드 ZIP 파일에는 Red Hat Single Sign-On 서버를 실행하는 스크립트 및 바이너리가 포함되어 있습니다. 먼저 7.4.0.GA 서버를 설치한 다음 7.4.10.GA 서버 패치를 설치합니다.

절차

1. **Red Hat 고객 포털 로 이동하십시오.**
2. **Red Hat Single Sign-On 7.4.0.GA 서버를 다운로드합니다.**
3. **unzip 또는 Expand-Archive와 같은 적절한 unzip 유틸리티를 사용하여 ZIP 파일의 압축을 풉니다.**

4. Red Hat 고객 포털 로 돌아가기

5. 패치 탭을 클릭합니다.

6. Red Hat Single Sign-On 7.4.10.GA 서버 패치를 다운로드합니다.

7. 다운로드한 파일을 선택한 디렉터리에 배치합니다.

8. JBoss EAP의 bin 디렉토리로 이동합니다.

9. JBoss EAP 명령줄 인터페이스를 시작합니다.

Linux/Unix

```
$ jboss-cli.sh
```

Windows

```
> jboss-cli.bat
```

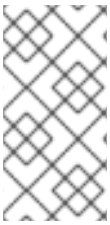
10. 패치를 적용합니다.

```
$ patch apply <path-to-zip>/rh-sso-7.4.10-patch.zip
```

추가 리소스

패치 적용에 대한 자세한 내용은 ZIP/Installer 설치 패치를 참조하십시오.

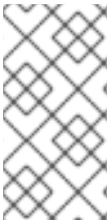
2.3. RPM에서 RH-SSO 설치



참고

Red Hat Enterprise Linux 7 및 8에서는 채널이 용어 리포지토리로 교체되었습니다. 이러한 명령에는 리포지토리라는 용어만 사용됩니다.

RPM에서 RH-SSO를 설치하려면 JBoss EAP 7.3 및 RH-SSO 7.4 리포지토리를 모두 구독해야 합니다.



참고

EAP RPM으로의 업그레이드를 계속 받을 수는 없지만 RH-SSO에 대한 업데이트는 수신하지 않습니다.

2.3.1. JBoss EAP 7.3 리포지토리 구독

사전 요구 사항

1. Red Hat Subscription Manager를 사용하여 Red Hat Enterprise Linux 시스템이 계정에 등록되어 있는지 확인합니다. 자세한 내용은 Red Hat 서브스크립션 관리 설명서를 참조하십시오.

2. 다른 JBoss EAP 리포지토리에 이미 등록한 경우 먼저 해당 리포지토리에서 구독을 취소해야 합니다.

Red Hat Enterprise Linux 6, 7의 경우: Red Hat Subscription Manager를 사용하여 다음 명령을 사용하여 JBoss EAP 7.3 리포지토리를 구독합니다. Red Hat Enterprise Linux 버전에 따라 <RHEL_VERSION>을 6 또는 7로 바꿉니다.

```
subscription-manager repos --enable=jb-eap-7.3-for-rhel-<RHEL_VERSION>-server-rpms --
enable=rhel-<RHEL_VERSION>-server-rpms
```

Red Hat Enterprise Linux 8의 경우: Red Hat Subscription Manager를 사용하여 다음 명령을 사용하여 JBoss EAP 7.3 리포지토리를 구독하십시오.

```
subscription-manager repos --enable=jb-eap-7.3-for-rhel-8-x86_64-rpms --enable=rhel-8-for-x86_64-baseos-rpms --enable=rhel-8-for-x86_64-appstream-rpms
```

2.3.2. RH-SSO 7.4 Repository 및 Installing RH-SSO 7.4 구독

사전 요구 사항

1. Red Hat Subscription Manager를 사용하여 Red Hat Enterprise Linux 시스템이 계정에 등록되어 있는지 확인합니다. 자세한 내용은 Red Hat 서브스크립션 관리 설명서를 참조하십시오.

2. 이미 JBoss EAP 7.3 리포지토리를 구독했는지 확인합니다. 자세한 내용은 JBoss EAP 7.3 리포지토리 서브스크립션을 참조하십시오.

RH-SSO 7.4 리포지토리를 구독하고 RH-SSO 7.4를 설치하려면 다음 단계를 완료하십시오.

1. Red Hat Enterprise Linux 6, 7의 경우: Red Hat Subscription Manager를 사용하여 다음 명령을 사용하여 RH-SSO 7.4 리포지토리에 등록합니다. Red Hat Enterprise Linux 버전에 따라 <RHEL_VERSION>을 6 또는 7로 바꿉니다.

```
subscription-manager repos --enable=rh-sso-7.4-for-rhel-<RHEL-VERSION>-server-rpms
```

2. Red Hat Enterprise Linux 8의 경우: Red Hat Subscription Manager를 사용하여 다음 명령을 사용하여 RH-SSO 7.4 리포지토리에 등록합니다.

```
subscription-manager repos --enable=rh-sso-7.4-for-rhel-8-x86_64-rpms
```

3. Red Hat Enterprise Linux 6의 경우 7: 다음 명령을 사용하여 구독한 RH-SSO 7.4 리포지토리에서 RH-SSO를 설치합니다.

```
yum groupinstall rh-sso7
```

4. Red Hat Enterprise Linux 8의 경우 다음 명령을 사용하여 구독한 RH-SSO 7.4 리포지토리에서 RH-SSO를 설치합니다.

```
dnf groupinstall rh-sso7
```

설치가 완료되었습니다. RPM 설치의 기본 RH-SSO_HOME 경로는 /opt/rh/rh-sso7/root/usr/share/keycloak입니다.

추가 리소스

Red Hat Single Sign-On의 7.4.10.GA 패치 설치에 대한 자세한 내용은 RPM 패치 를 참조하십시오.

2.4. 배포 디렉터리 STRUCTURE

이 장에서는 서버 배포의 디렉터리 구조를 안내합니다.

일부 디렉토리의 목적을 살펴보겠습니다.

bin/

여기에는 서버를 부팅하거나 서버에서 다른 관리 작업을 수행하는 다양한 스크립트가 포함되어 있습니다.

domain/

여기에는 도메인 모드에서 Red Hat Single Sign-On을 실행할 때 구성 파일 및 작업 디렉터리가 포함되어 있습니다.

modules/

이는 서버에서 사용하는 모든 Java 라이브러리입니다.

standalone/

여기에는 독립 실행형 모드에서 Red Hat Single Sign-On을 실행할 때 구성 파일과 작업 디렉터리가 포함되어 있습니다.

standalone/deployments/

Red Hat Single Sign-On에 확장을 작성하는 경우 여기에 확장을 배치할 수 있습니다. 이에 대한 자세한 내용은 서버 개발자 가이드를 참조하십시오.

Themes/

이 디렉터리에는 서버에서 표시하는 모든 UI 화면을 표시하는 데 사용되는 모든 html, 스타일 시트, JavaScript 파일 및 이미지가 포함되어 있습니다. 여기에서 기존 항목을 수정하거나 직접 만들 수 있습니다. 이에 대한 자세한 내용은 서버 개발자 가이드를 참조하십시오.

3장. 운영 모드 선택

프로덕션 환경에 **Red Hat Single Sign-On**을 배포하기 전에 사용할 운영 모드 유형을 결정해야 합니다. 클러스터 내에서 **Red Hat Single Sign-On**을 실행하시겠습니까? 서버 구성을 관리하는 중앙 집중식 방법이 필요하십니까? 선택한 운영 모드는 데이터베이스를 구성하고 캐싱을 구성하고 서버를 부팅하는 방법에도 영향을 미칩니다.

작은 정보

Red Hat Single Sign-On은 **JBoss EAP Application Server**를 기반으로 구축됩니다. 이 가이드에서는 특정 모드 내에서 배포의 기본 사항만 다룹니다. 이에 대한 자세한 내용을 보려면 ***JBoss EAP 구성 가이드***.

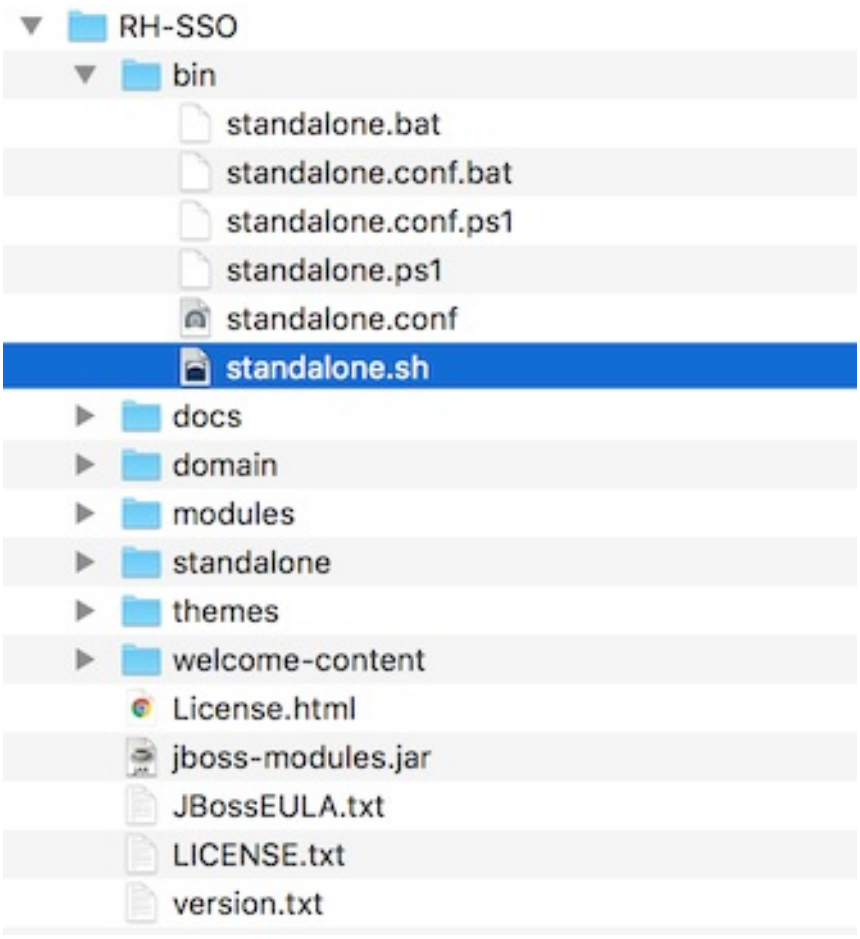
3.1. 독립 실행형 모드

독립 실행형 운영 모드는 하나의 **Red Hat Single Sign-On** 서버 인스턴스만 실행하려는 경우에만 유용합니다. 클러스터형 배포에는 사용할 수 없으며 모든 캐시는 배포되지 않고 로컬 전용입니다. 단일 장애 지점이 있으므로 프로덕션에서 독립 실행형 모드를 사용하지 않는 것이 좋습니다. 독립 실행형 모드 서버가 다운되면 사용자가 로그인할 수 없습니다. 이 모드는 **Red Hat Single Sign-On**의 기능을 사용하여 드라이브를 테스트하고 플레이하는 데에만 유용합니다.

3.1.1. 독립 실행형 부팅 스크립트

독립 실행형 모드에서 서버를 실행하는 경우 운영 체제에 따라 서버를 부팅하려면 특정 스크립트를 실행해야 합니다. 이러한 스크립트는 서버 배포의 ***bin/*** 디렉터리에 있습니다.

독립 실행형 부팅 스크립트



서버를 부팅하려면 다음을 수행합니다.

Linux/Unix

```
$ .../bin/standalone.sh
```

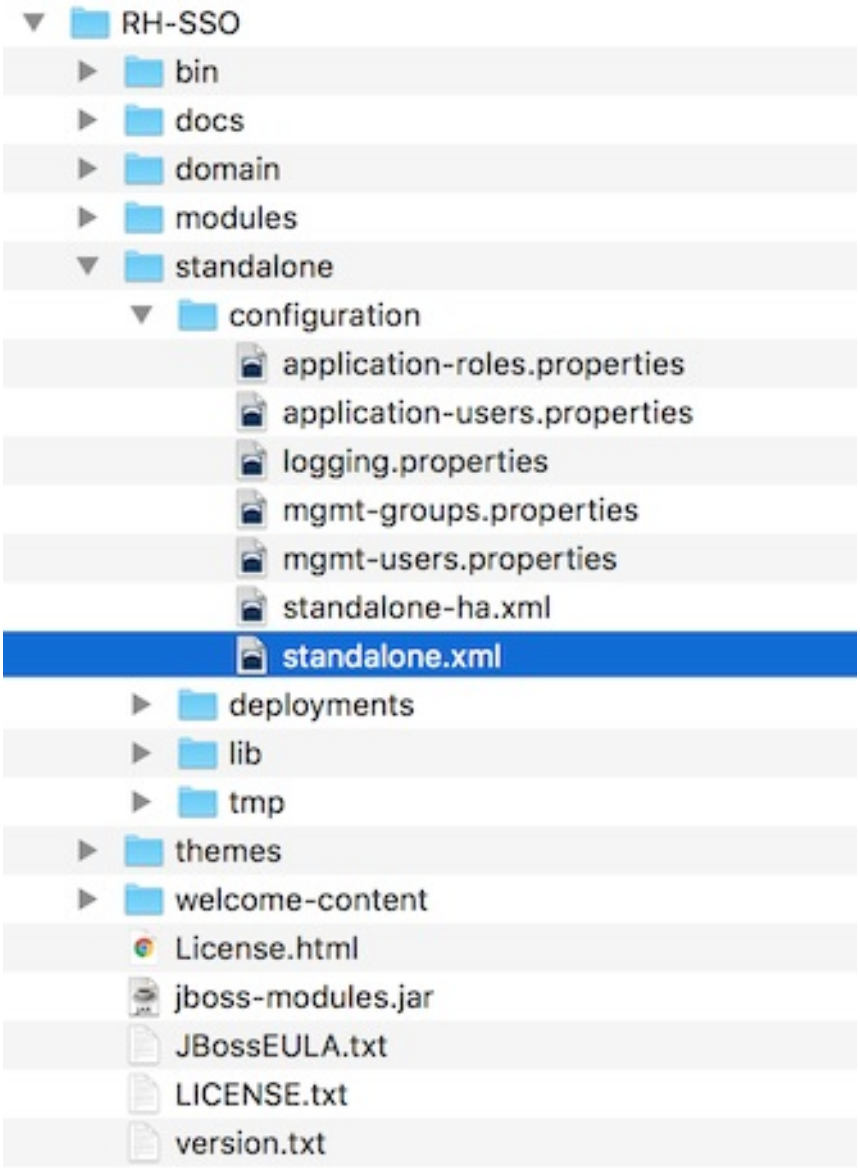
Windows

```
> ...\bin\standalone.bat
```

3.1.2. 독립 실행형 구성

이 가이드의 대부분은 **Red Hat Single Sign-On**의 인프라 수준 측면을 구성하는 방법을 안내합니다. 이러한 측면은 **Red Hat Single Sign-On**이 파생된 애플리케이션 서버에 고유한 구성 파일에 구성됩니다. 독립 실행형 작업 모드에서 이 파일은 ***.../standalone/configuration/standalone.xml*** 에 있습니다. 이 파일은 **Red Hat Single Sign-On** 구성 요소에 특정한 비인프라 수준 항목을 구성하는 데도 사용됩니다.

독립 실행형 구성 파일



주의

서버를 실행하는 동안 이 파일에 대한 변경 사항은 적용되지 않으며 서버에서 덮어쓸 수도 있습니다. 대신 명령줄 스크립팅 또는 JBoss EAP의 웹 콘솔을 사용합니다. 자세한 내용은 *JBoss EAP 구성 가이드를 참조하십시오*.

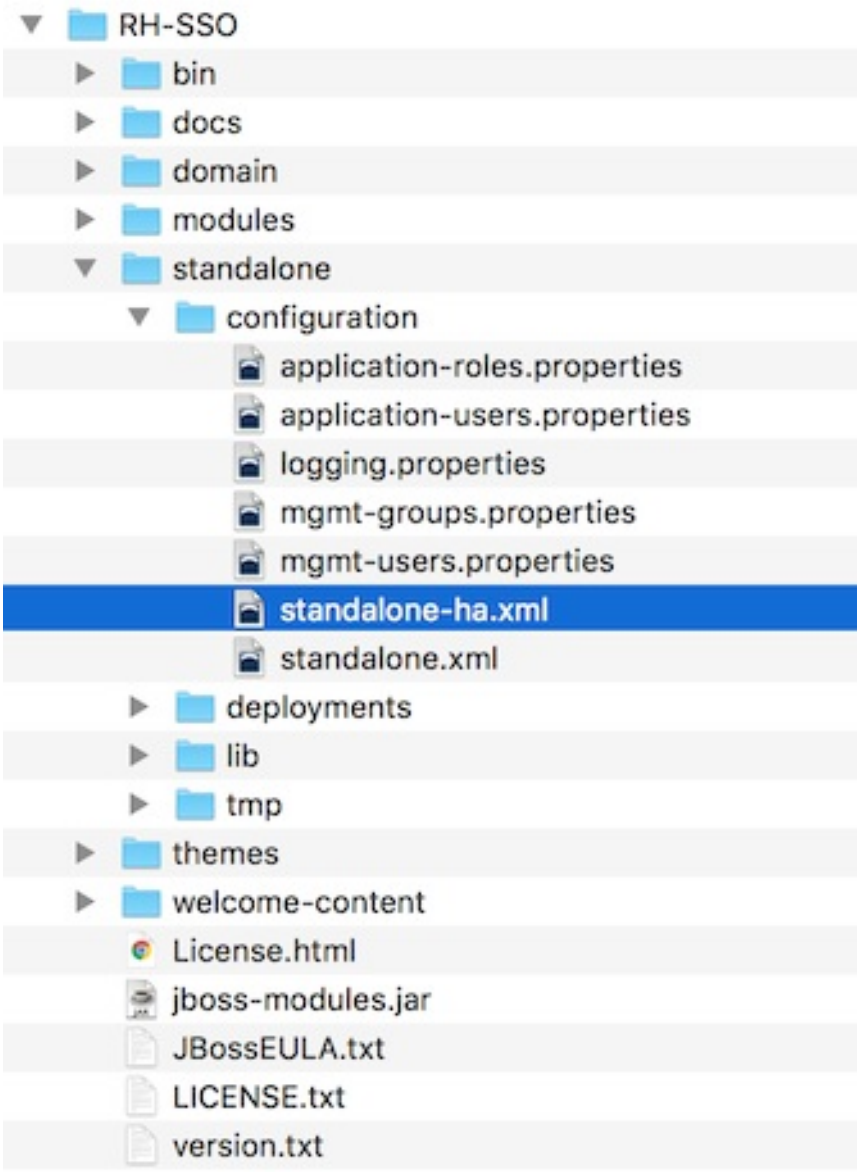
3.2. 독립 실행형 클러스터 모드

독립 실행형 클러스터형 작동 모드는 클러스터 내에서 Red Hat Single Sign-On을 실행하려는 경우입니다. 이 모드에서는 서버 인스턴스를 실행하려는 각 머신에 Red Hat Single Sign-On 배포 사본이 있어야 합니다. 이 모드는 처음에 배포하기가 매우 쉽지만 매우 번거로울 수 있습니다. 구성을 변경하려면 각 시스템에서 각 배포를 수정해야 합니다. 대규모 클러스터의 경우 시간이 오래 걸릴 수 있으며 오류가 발생하기 쉽습니다.

3.2.1. 독립 실행형 클러스터 구성

배포에는 클러스터 내에서 실행하기 위해 대부분 사전 구성된 앱 서버 구성 파일이 있습니다. 네트워킹, 데이터베이스, 캐시 및 검색에 대한 모든 특정 인프라 설정이 있습니다. 이 파일은 *.../standalone/configuration/standalone-ha.xml* 에 있습니다. 이 구성에는 몇 가지 누락 사항이 있습니다. 공유 데이터베이스 연결을 구성하지 않고 클러스터에서 Red Hat Single Sign-On을 실행할 수 없습니다. 또한 클러스터 앞에 특정 유형의 로드 밸런서를 배포해야 합니다. 이 가이드의 클러스터링 및 데이터베이스 섹션에서는 이러한 사항을 안내합니다.

독립 실행형 HA 구성



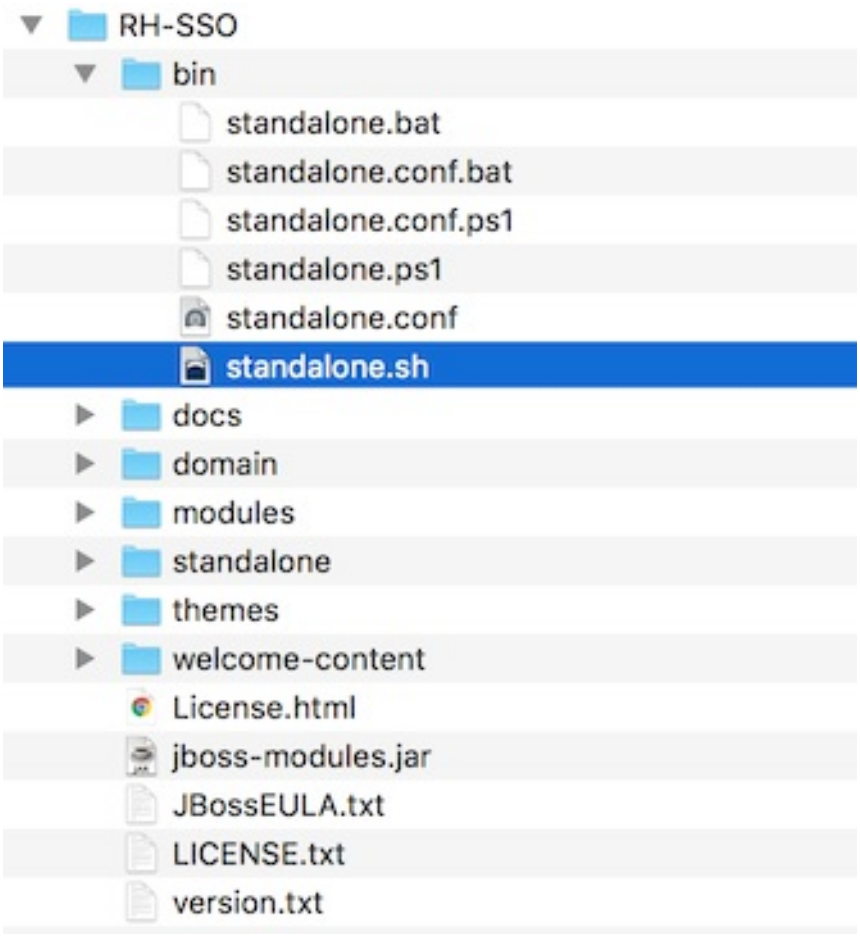
주의

서버를 실행하는 동안 이 파일에 대한 변경 사항은 적용되지 않으며 서버에서 덮어쓸 수도 있습니다. 대신 명령줄 스크립팅 또는 **JBoss EAP**의 웹 콘솔을 사용합니다. 자세한 내용은 *JBoss EAP 구성 가이드를 참조하십시오*.

3.2.2. 독립 실행형 클러스터 부팅 스크립트

독립 실행형 모드에서와 동일한 부팅 스크립트를 사용하여 **Red Hat Single Sign-On**을 시작합니다. 차이점은 **HA** 구성 파일을 가리키도록 추가 플래그를 전달하는 것입니다.

독립 실행형 클러스터 부팅 스크립트



서버를 부팅하려면 다음을 수행합니다.

Linux/Unix

```
$ .../bin/standalone.sh --server-config=standalone-ha.xml
```

Windows

```
> ...\bin\standalone.bat --server-config=standalone-ha.xml
```

3.3. 도메인 클러스터 모드

도메인 모드는 서버의 구성을 중앙에서 관리하고 게시하는 방법입니다.

표준 모드에서 클러스터를 실행하면 클러스터가 크기가 증가함에 따라 빠르게 악화될 수 있습니다. 구성을 변경해야 할 때마다 클러스터의 각 노드에서 수행해야 합니다. 도메인 모드는 구성을 저장하고 게시하는 중앙 위치를 제공하여 이 문제를 해결합니다. 설정하는 것은 매우 복잡할 수 있지만 결국 가치가 있습니다. 이 기능은 **Red Hat Single Sign-On**이 파생되는 **JBoss EAP Application Server**에 빌드됩니다.

참고

가이드는 도메인 모드의 기본 사항을 설명합니다. 클러스터에서 도메인 모드를 설정하는 방법에 대한 자세한 단계는 ***JBoss EAP** 구성 가이드에서* 가져와야 합니다.

다음은 도메인 모드에서 실행의 몇 가지 기본 개념입니다.

도메인 컨트롤러

도메인 컨트롤러는 클러스터의 각 노드에 대한 일반 구성을 저장, 관리 및 게시하는 프로세스입니다. 이 프로세스는 클러스터의 노드가 구성을 가져오는 중앙 지점입니다.

호스트 컨트롤러

호스트 컨트롤러는 특정 시스템에서 서버 인스턴스를 관리합니다. 하나 이상의 서버 인스턴스를 실행하도록 구성합니다. 도메인 컨트롤러는 각 시스템의 호스트 컨트롤러와 상호 작용하여 클러스터를 관리할 수도 있습니다. 실행 중인 프로세스의 수를 줄이기 위해 도메인 컨트롤러는 실행 중인 시스템에서 호스트 컨트롤러 역할을 합니다.

도메인 프로필

도메인 프로필은 서버에서 부팅하는 데 사용할 수 있는 이름이 지정된 구성 세트입니다. 도메인 컨트롤러는 다른 서버에서 사용하는 여러 도메인 프로필을 정의할 수 있습니다.

서버 그룹

서버 그룹은 서버 컬렉션입니다. 구성 요소는 하나로 관리 및 구성됩니다. 도메인 프로필을 서버 그룹에 할당할 수 있으며 해당 그룹의 모든 서비스에서 해당 도메인 프로필을 구성으로 사용합니다.

도메인 모드에서는 마스터 노드에서 도메인 컨트롤러가 시작됩니다. 클러스터 구성은 도메인 컨트롤러에 있습니다. 다음으로 클러스터의 각 시스템에서 호스트 컨트롤러가 시작됩니다. 각 호스트 컨트롤러 배포 구성은 해당 시스템에서 시작할 **Red Hat Single Sign-On** 서버 인스턴스 수를 지정합니다. 호스트 컨트롤러가 부팅되면 이를 수행하도록 구성된 만큼 **Red Hat Single Sign-On** 서버 인스턴스가 시작됩니다. 이러한 서버 인스턴스는 도메인 컨트롤러에서 구성을 가져옵니다.

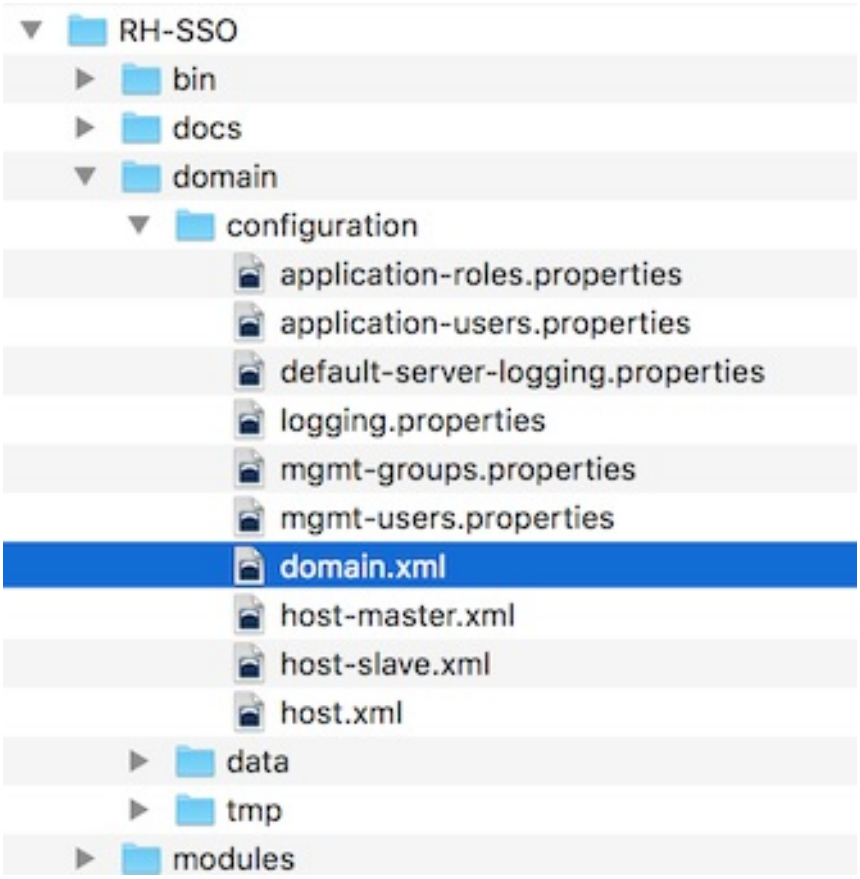
참고

Microsoft Azure와 같은 일부 환경에서는 도메인 모드가 적용되지 않습니다. **JBoss EAP** 설명서를 참조하십시오.

3.3.1. 도메인 구성

이 가이드의 다른 여러 장에서는 데이터베이스, **HTTP** 네트워크 연결, 캐시 및 기타 인프라 관련 항목과 같은 다양한 측면을 구성하는 방법을 설명합니다. 독립 실행형 모드는 ***standalone.xml*** 파일을 사용하여 이러한 작업을 구성하는 반면 도메인 모드는 ***…/domain/configuration/domain.xml*** 구성 파일을 사용합니다. 여기에서 **Red Hat Single Sign-On** 서버의 도메인 프로필 및 서버 그룹이 정의됩니다.

domain.xml



주의

도메인 컨트롤러가 실행되는 동안 이 파일에 대한 변경 사항은 적용되지 않으며 서버에서 덮어쓸 수도 있습니다. 대신 명령줄 스크립팅 또는 **JBoss EAP**의 웹 콘솔을 사용합니다. 자세한 내용은 ***JBoss EAP** 구성 가이드를 참조하십시오*.

이 ***domain.xml*** 파일의 몇 가지 측면을 살펴보겠습니다. **auth-server-standalone** 및 **auth-server-clustered** 프로필 **XML** 블록을 통해 대량의 구성 결정을 내릴 수 있습니다. 네트워크 연결, 캐시 및 데이터베이스 연결과 같은 항목을 구성합니다.

auth-server 프로필

```
<profiles>
  <profile name="auth-server-standalone">
    ...
  </profile>
  <profile name="auth-server-clustered">
    ...
  </profile>
```

auth-server-standalone 프로필은 클러스터되지 않은 설정입니다. **auth-server-clustered** 프로필은 클러스터형 설정입니다.

아래로 스크롤하면 다양한 **socket-binding-groups** 가 정의되어 있습니다.

socket-binding-groups

```
<socket-binding-groups>
  <socket-binding-group name="standard-sockets" default-interface="public">
    ...
  </socket-binding-group>
  <socket-binding-group name="ha-sockets" default-interface="public">
    ...
```

```
        </socket-binding-group>
        <!-- load-balancer-sockets should be removed in production systems and replaced with a
better software or hardware based one -->
        <socket-binding-group name="load-balancer-sockets" default-interface="public">
          ...
        </socket-binding-group>
    </socket-binding-groups>
```

이 구성에서는 각 **Red Hat Single Sign-On** 서버 인스턴스에서 열리는 다양한 커넥터의 기본 포트 매핑을 정의합니다. **${...}** 을 포함하는 모든 값은 **-D** 스위치(예:)를 사용하여 명령줄에서 재정의할 수 있는 값입니다.

```
$ domain.sh -Djboss.http.port=80
```

Red Hat Single Sign-On의 서버 그룹은 **server-groups XML** 블록에 있습니다. 호스트 컨트롤러가 인스턴스를 부팅할 때 사용되는 도메인 프로필(기본값)과 **Java VM**의 일부 기본 부팅 인수도 지정합니다. 또한 **socket-binding-group** 을 **server** 그룹에 바인딩합니다.

서버 그룹

```
    <server-groups>
        <!-- load-balancer-group should be removed in production systems and replaced with a
better software or hardware based one -->
        <server-group name="load-balancer-group" profile="load-balancer">
            <jvm name="default">
                <heap size="64m" max-size="512m"/>
            </jvm>
            <socket-binding-group ref="load-balancer-sockets"/>
        </server-group>
        <server-group name="auth-server-group" profile="auth-server-clustered">
            <jvm name="default">
                <heap size="64m" max-size="512m"/>
            </jvm>
            <socket-binding-group ref="ha-sockets"/>
        </server-group>
    </server-groups>
```

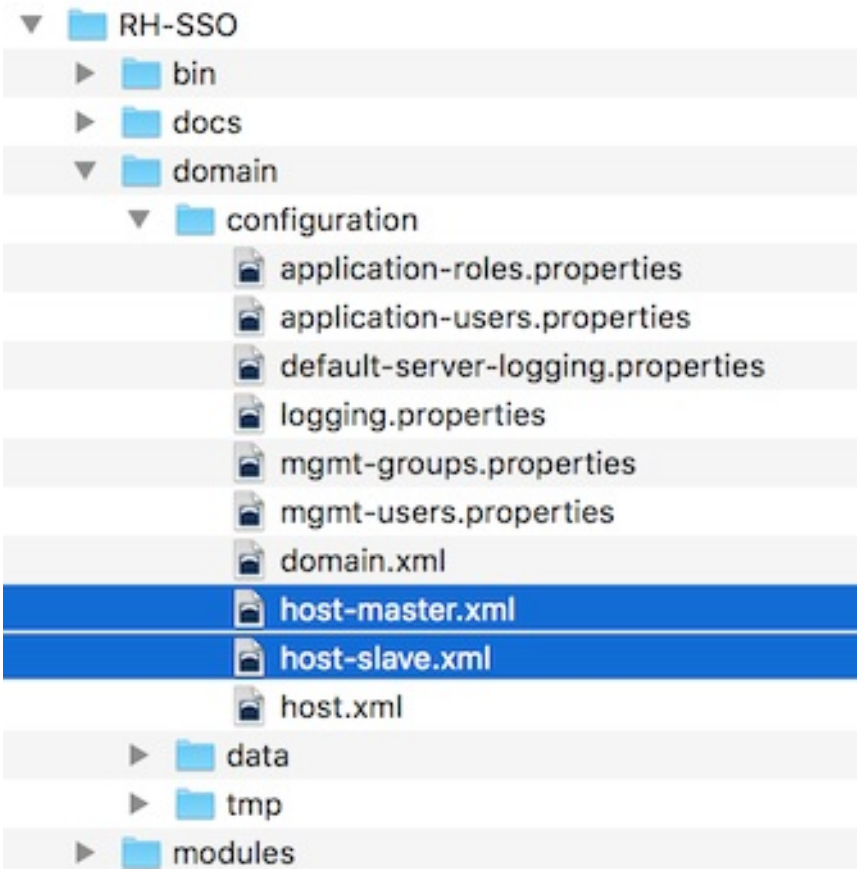
3.3.2. 호스트 컨트롤러 구성

Red Hat Single Sign-On에는 *.../domain/configuration/* 디렉터리에 있는 두 개의 호스트 컨트롤러 구성 파일이 포함되어 있습니다. *host-master.xml* 및 *host-slave.xml*. *host-master.xml* 은 도메인 컨트롤러, 로드 밸런서 및 하나의 Red Hat Single Sign-On 서버 인스턴스를 부팅하도록 구성되어 있습니다. *host-slave.xml* 은 도메인 컨트롤러와 통신하고 하나의 Red Hat Single Sign-On 서버 인스턴스를 부팅하도록 구성되어 있습니다.

참고

로드 밸런서는 필수 서비스가 아닙니다. 개발 시스템에서 드라이브 클러스터링을 쉽게 테스트할 수 있도록 존재합니다. 프로덕션 환경에서 사용할 수 있지만 사용하려는 하드웨어 또는 소프트웨어 기반 로드 밸런서가 다른 경우 이를 교체할 수 있습니다.

호스트 컨트롤러 구성



로드 밸런서 서버 인스턴스를 비활성화하려면 *host-master.xml* 을 편집하고 "load-balancer" 항목을 주석 처리하거나 제거합니다.

```
<servers>
   <!-- remove or comment out next line -->
   <server name="load-balancer" group="loadbalancer-group"/>
   ...
</servers>
```

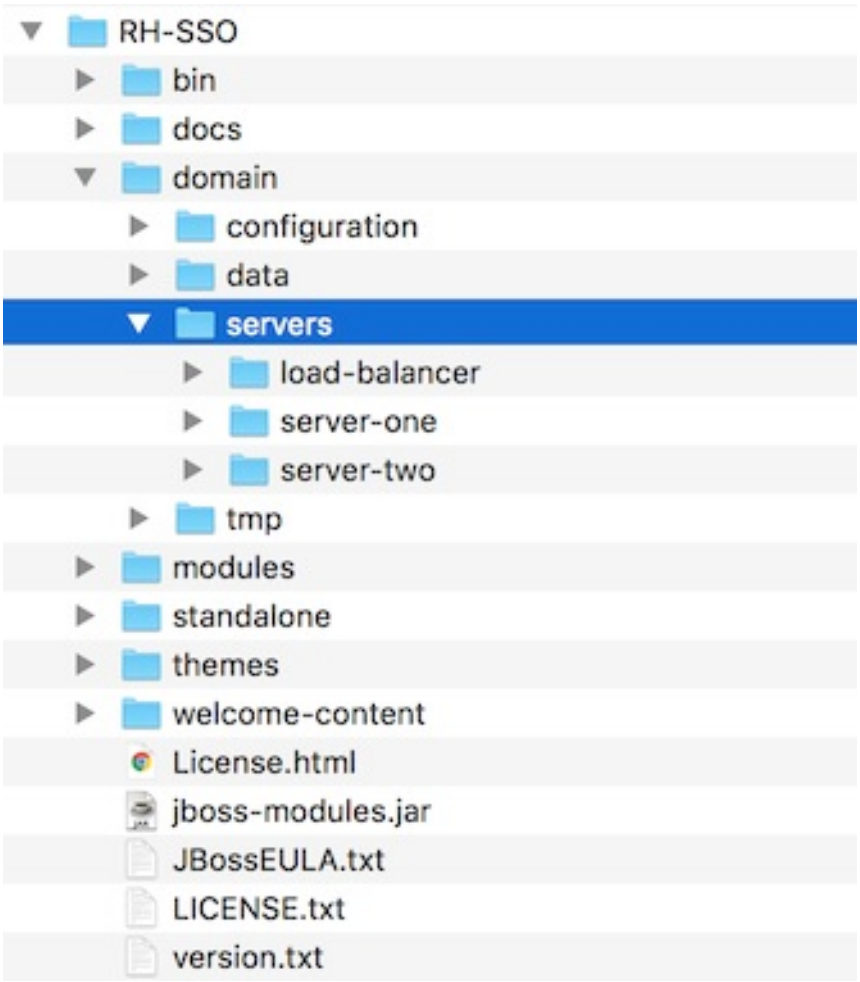
이 파일에 대해 주목해야 할 또 다른 흥미로운 점은 인증 서버 인스턴스를 선언하는 것입니다. **port-offset** 설정이 있습니다. ***domain.xml*** **socket-binding-group** 또는 **server group**에 정의된 네트워크 포트에는 **port-offset** 값이 추가됩니다. 이 예제 도메인 설정에서는 로드 밸런서 서버에서 열고 있는 포트가 시작된 인증 서버 인스턴스와 충돌하지 않도록 이 작업을 수행합니다.

```
<servers>
   ...
   <server name="server-one" group="auth-server-group" auto-start="true">
       <socket-bindings port-offset="150"/>
   </server>
</servers>
```

3.3.3. 서버 인스턴스 작업 디렉터리

호스트 파일에 정의된 각 **Red Hat Single Sign-On** 서버 인스턴스는 ***.../domain/servers/{SERVER NAME}*** 에 작업 디렉터리를 생성합니다. 추가 구성이 있고 서버 인스턴스에 필요하거나 생성되는 임시, 로그 또는 데이터 파일도 배치할 수 있습니다. 서버당 이러한 구조의 구조는 다른 **JBoss EAP** 부팅 서버처럼 보입니다.

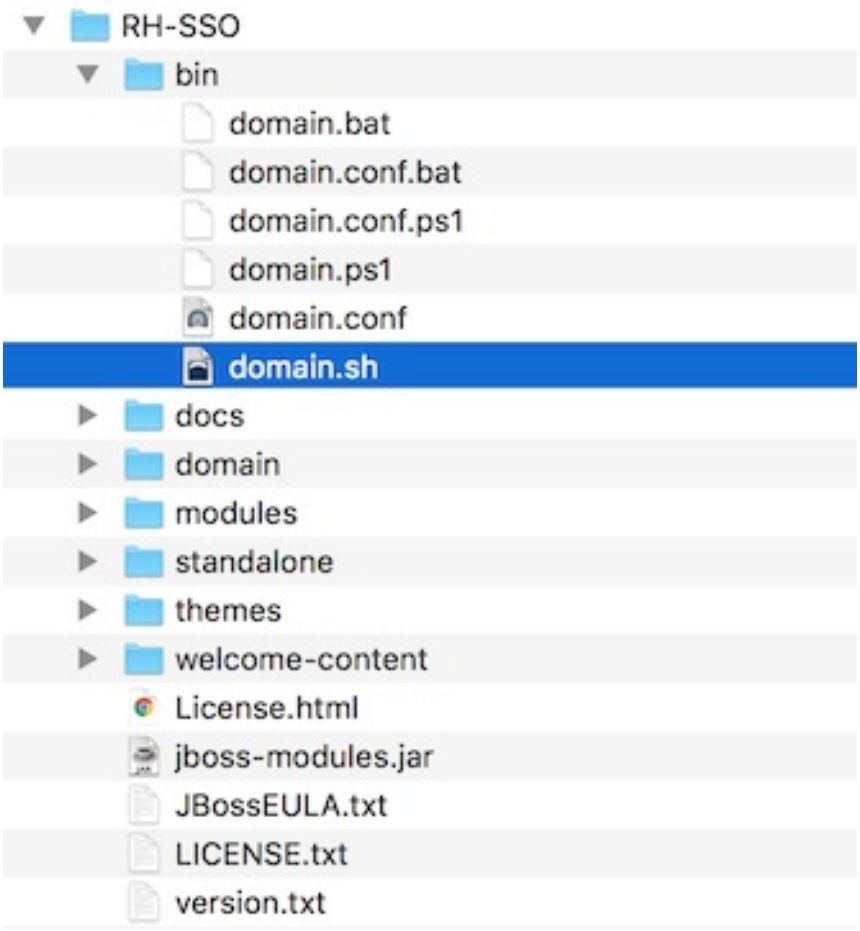
작업 디렉터리



3.3.4. Domain Boot Script

서버를 도메인 모드에서 실행하는 경우 운영 체제에 따라 서버를 부팅하기 위해 실행해야 하는 특정 스크립트가 있습니다. 이러한 스크립트는 서버 배포의 ***bin/*** 디렉터리에 있습니다.

Domain Boot Script



서버를 부팅하려면 다음을 수행합니다.

Linux/Unix

```
$ .../bin/domain.sh --host-config=host-master.xml
```

Windows

```
> ...\bin\domain.bat --host-config=host-master.xml
```

부팅 스크립트를 실행할 때 **--host-config** 스위치를 통해 사용할 호스트 제어 구성 파일을 전달해야 합니다.

3.3.5. 클러스터형 도메인 예

out-of-the-box ***domain.xml*** 구성을 사용하여 드라이브 클러스터링을 테스트할 수 있습니다. 이 예제 도메인은 하나의 시스템에서 실행하고 부팅하기 위한 것입니다.

- 도메인 컨트롤러
- **HTTP** 로드 밸런서
- **Red Hat Single Sign-On** 서버 인스턴스 **2**개

두 시스템에서 클러스터 실행을 시뮬레이션하려면 **domain.sh** 스크립트를 두 번 실행하여 두 개의 별도의 호스트 컨트롤러를 시작해야 합니다. 첫 번째는 도메인 컨트롤러, **HTTP** 로드 밸런서 및 하나의 **Red Hat Single Sign-On** 인증 서버 인스턴스를 시작하는 마스터 호스트 컨트롤러입니다. 두 번째는 인증 서버 인스턴스만 시작하는 슬레이브 호스트 컨트롤러가 됩니다.

3.3.5.1. 도메인 컨트롤러에 슬라브 연결 설정

그러나 작업을 부팅하려면 먼저 슬레이브 호스트 컨트롤러를 구성하여 도메인 컨트롤러에 안전하게 통신할 수 있습니다. 이 작업을 수행하지 않으면 슬레이브 호스트에서 도메인 컨트롤러에서 중앙 집중식 구성을 가져올 수 없습니다. 보안 연결을 설정하려면 마스터와 슬레이브 간에 공유할 서버 **admin** 사용자와 시크릿을 생성해야 합니다. **.../bin/add-user.sh** 스크립트를 실행하여 이 작업을 수행합니다.

스크립트를 실행하면 **Management User** 를 선택하고 **yes** 로 응답하면 하나의 **AS** 프로세스가 다른 프로세스에 연결할 때 새 사용자를 사용할지 묻는 메시지가 표시됩니다. 이렇게 하면 ***.../domain/configuration/host-slave.xml*** 파일을 잘라내어 붙여넣어야 하는 보안이 생성됩니다.

앱 서버 관리자 추가

```
$ add-user.sh
 What type of user do you wish to add?
  a) Management User (mgmt-users.properties)
  b) Application User (application-users.properties)
 (a): a
 Enter the details of the new user to add.
 Using realm 'ManagementRealm' as discovered from the existing property files.
 Username : admin
 Password recommendations are listed below. To modify these restrictions edit the add-
user.properties configuration file.
  - The password should not be one of the following restricted values {root, admin, administrator}
  - The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-
alphanumeric symbol(s)
  - The password should be different from the username
 Password :
 Re-enter Password :
 What groups do you want this user to belong to? (Please enter a comma separated list, or leave
blank for none)[ ]:
 About to add user 'admin' for realm 'ManagementRealm'
 Is this correct yes/no? yes
 Added user 'admin' to file '/.../standalone/configuration/mgmt-users.properties'
 Added user 'admin' to file '/.../domain/configuration/mgmt-users.properties'
 Added user 'admin' with groups to file '/.../standalone/configuration/mgmt-groups.properties'
 Added user 'admin' with groups to file '/.../domain/configuration/mgmt-groups.properties'
 Is this new user going to be used for one AS process to connect to another AS process?
 e.g. for a slave host controller connecting to the master or for a Remoting connection for server to
server EJB calls.
 yes/no? yes
 To represent the user add the following to the server-identities definition <secret
value="bWdtdDEyMyE=" />
```

참고

add-user.sh는 Red Hat Single Sign-On 서버에 사용자를 추가하지 않고 기본 JBoss Enterprise Application Platform에 사용자를 추가합니다. 위 스크립트에서 사용되고 생성된 인증 정보는 예시용으로만 사용됩니다. 시스템에서 생성된 항목을 사용하십시오.

다음으로 다음과 같이 시크릿 값을 ***…/domain/configuration/host-slave.xml*** 파일에 잘라내어 붙여 넣습니다.

```
    <management>
        <security-realms>
```

```
        <security-realm name="ManagementRealm">
            <server-identities>
                <secret value="bWdtdDEyMyE="/>
            </server-identities>
```

또한 생성된 사용자의 사용자 *이름을 .../domain/configuration/host-slave.xml* 파일에 추가해야 합니다.

```
    <remote security-realm="ManagementRealm" username="admin">
```

3.3.5.2. 부팅 스크립트 실행

하나의 개발 시스템에서 두 노드 클러스터를 시뮬레이션하므로 부팅 스크립트를 두 번 실행합니다.

마스터 부팅

```
$ domain.sh --host-config=host-master.xml
```

슬레이브 부팅

```
$ domain.sh --host-config=host-slave.xml
```

이를 시도하려면 브라우저를 열고 http://localhost:8080/auth 로 이동합니다.

3.4. 데이터센터 간 복제 모드

Red Hat Single Sign-On 7.2에서 기술 프리뷰 기능으로 도입된 교차 사이트 복제는 최신 **RH-SSO 7.6** 릴리스를 포함한 **Red Hat SSO 7.x** 릴리스에서 지원되는 기능으로 더 이상 사용할 수 없습니다. **Red Hat**은 지원되지 않으므로 고객 구현 또는 해당 환경에서 이 기능을 사용하지 않는 것이 좋습니다. 또한 이 기능에 대한 지원 예외는 더 이상 고려되거나 허용되지 않습니다.

사이트 간 복제를 위한 새로운 솔루션은 Red Hat SSO 8 대신 도입될 제품인RHBK(Red Hat build of Keycloak)의 향후 릴리스에 대해 지속적으로 고려하고 있습니다. 자세한 내용은 곧 제공될 예정입니다.

4장. CRYOSTAT 구성 관리

Red Hat Single Sign-On의 하위 수준 구성은 배포에서 **standalone.xml,standalone-ha.xml** 또는 **domain.xml** 파일을 편집하여 수행됩니다. 이 파일의 위치는 작동 모드에 따라 다릅니다.

여기에서 구성할 수 있는 끝없는 설정이 있지만 이 섹션에서는 ***keycloak-server*** 하위 시스템의 구성에 중점을 둡니다. 사용 중인 구성 파일에 관계없이 ***keycloak-server*** 하위 시스템의 구성은 동일합니다.

keycloak-server 하위 시스템은 일반적으로 다음과 같이 파일의 끝에 선언됩니다.

```
<subsystem xmlns="urn:jboss:domain:keycloak-server:1.1">
  <web-context>auth</web-context>
  ...
</subsystem>
```

이 하위 시스템에서 변경된 내용은 서버가 재부팅될 때까지 적용되지 않습니다.

4.1. SPI 공급자 구성

각 구성 설정의 세부 사항은 해당 설정과 관련된 다른 위치에서 설명합니다. 그러나 **SPI** 공급자의 설정을 선언하는 데 사용되는 형식을 이해하는 것이 유용합니다.

Red Hat Single Sign-On은 뛰어난 유연성을 제공하는 모듈식 시스템입니다. 50개 이상의 **SPI**(서비스 공급자 인터페이스)가 있으며 각 **SPI** 구현을 스왑 아웃할 수 있습니다. **SPI**의 구현은 *공급자* 라고 합니다.

SPI 선언의 모든 요소는 선택 사항이지만 전체 **SPI** 선언은 다음과 같습니다.

```
<spi name="myspi">
   <default-provider>myprovider</default-provider>
   <provider name="myprovider" enabled="true">
      <properties>
         <property name="foo" value="bar"/>
      </properties>
   </provider>
   <provider name="mysecondprovider" enabled="true">
      <properties>
         <property name="foo" value="foo"/>
      </properties>
   </provider>
</spi>
```

여기 **SPI myspi** 에 대해 정의된 두 개의 공급자가 있습니다. **default-provider** 는 **myprovider** 로 나열됩니다. 그러나 **SPI**는 이 설정을 처리하는 방법을 결정합니다. 일부 **SPI**는 두 개 이상의 공급자를 허용하고 일부는 제공하지 않습니다. 따라서 **default-provider** 는 **SPI**를 선택하는 데 도움이 될 수 있습니다.

또한 각 공급자는 고유한 구성 속성 세트를 정의합니다. 위의 두 공급자에 모두 **foo** 라는 속성이 있다는 사실은 단순히 비만입니다.

각 속성 값의 유형은 공급자가 해석합니다. 그러나 한 가지 예외가 있습니다. **eventsStore SPI**의 **jpa** 공급자를 고려하십시오.

```
<spi name="eventsStore">
   <provider name="jpa" enabled="true">
      <properties>
         <property name="exclude-events" value="[&quot;EVENT1&quot;,
                                    &quot;EVENT2&quot;]"/>
      </properties>
   </provider>
</spi>
```

이 값은 대괄호로 시작하고 대괄호로 끝나는 것을 알 수 있습니다. 즉, 값이 공급자에게 목록으로 전달됩니다. 이 예제에서 시스템은 두 개의 요소 값이 있는 목록을 공급자에게 전달합니다. *목록에 값을 더 추가하려면 각 목록 요소를 쉼표로 구분합니다. 안타깝게도 각 목록 요소 옆에 있는 따옴표를 이스케이프해야 합니다.*

4.2. JBOSS EAP CLI 시작

*구성을 직접 편집하는 것 외에도 **jboss-cli** 툴을 통해 명령을 실행하여 구성을 변경할 수도 있습니다. **CLI**를 사용하면 서버를 로컬 또는 원격으로 구성할 수 있습니다. 스크립팅과 결합할 때 특히 유용합니다.*

***JBoss EAP CLI**를 시작하려면 **jboss-cli** 를 실행해야 합니다.*

Linux/Unix

```
$ .../bin/jboss-cli.sh
```

Windows

```
> ...\bin\jboss-cli.bat
```

이렇게 하면 다음과 같은 프롬프트가 표시됩니다.

prompt

```
[disconnected /]
```

실행 중인 서버에서 명령을 실행하려면 먼저 **connect** 명령을 실행합니다.

연결

```
[disconnected /] connect
connect
[standalone@localhost:9990 /]
```

자신을 생각하고 있을 수 있습니다. "사용자 이름이나 비밀번호를 입력하지 않았습니다!" 실행 중인 독립 실행형 서버 또는 도메인 컨트롤러와 동일한 시스템에서 **jboss-cli** 를 실행하고 계정에 적절한 파일 권한이 있는 경우 관리자 사용자 이름과 암호를 설정하거나 입력할 필요가 없습니다. 해당 설정을 사용하지 않는 경우 보안을 강화하는 방법에 대한 자세한 내용은 **JBoss EAP** 구성 가이드를 참조하십시오.

4.3. CLI CRYOSTAT 모드

독립 실행형 서버와 동일한 머신에 있고 서버가 활성 상태가 아닌 동안 명령을 실행하려는 경우 서버를 **CLI**에 삽입하고 들어오는 요청을 허용하지 않는 특수 모드를 변경할 수 있습니다. 이렇게 하려면 변경하려는 구성 파일을 사용하여 **embed-server** 명령을 실행합니다.

embed-server

```
[disconnected /] embed-server --server-config=standalone.xml
[standalone@embedded /]
```

4.4. CLI GUI 모드

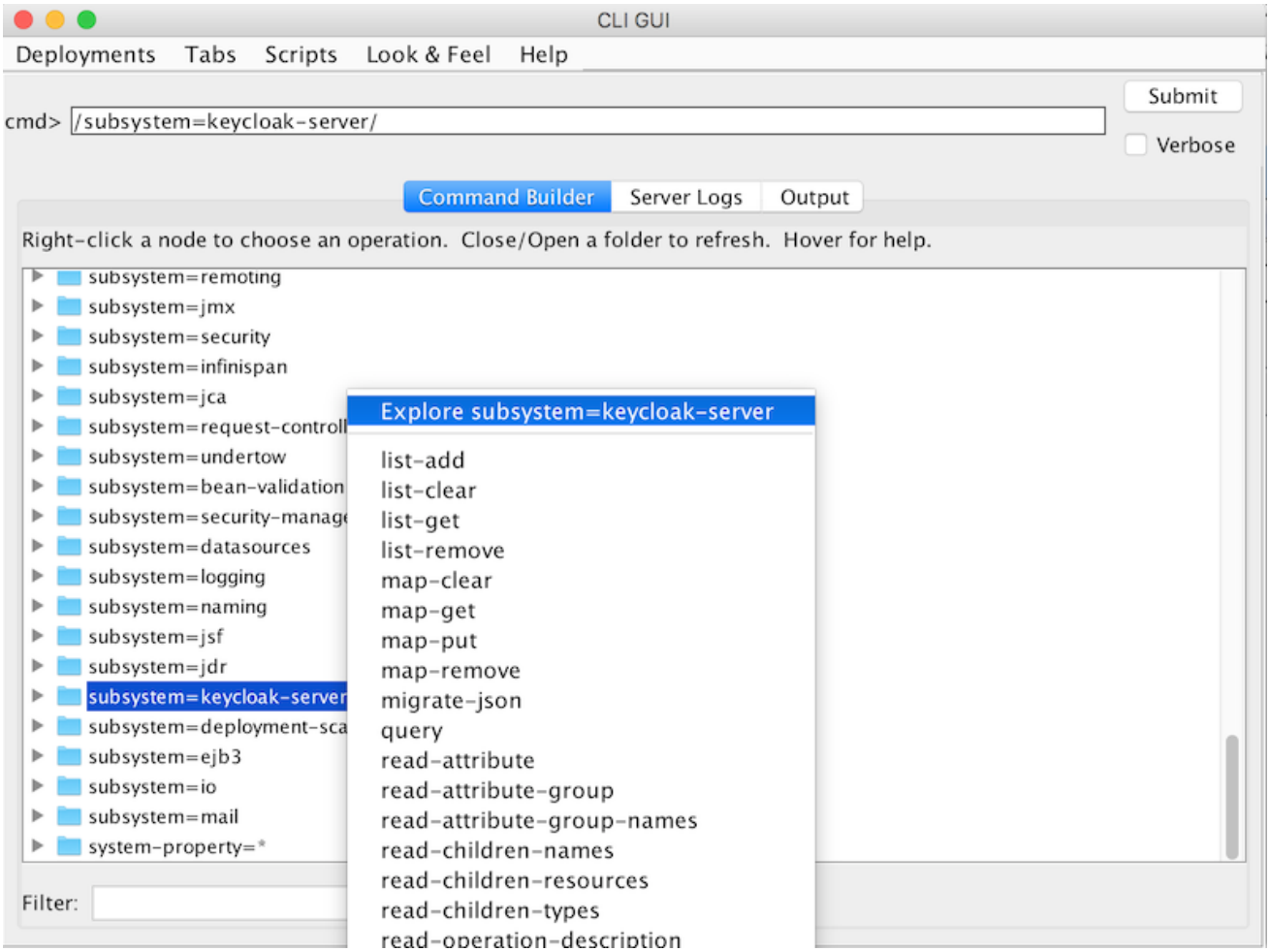
***CLI**는 **GUI** 모드에서도 실행할 수 있습니다. **GUI** 모드는 실행 중인 서버의 전체 관리 모델을 그래픽으로 보고 편집할 수 있는 **Swing** 애플리케이션을 시작합니다. **GUI** 모드는 **CLI** 명령을 포맷하고 사용 가능한 옵션에 대해 학습하는 데 도움이 필요한 경우 특히 유용합니다. **GUI**는 로컬 또는 원격 서버에서 서버 로그를 검색할 수도 있습니다.*

***GUI** 모드에서 시작*

```
$ .../bin/jboss-cli.sh --gui
```

*참고: 원격 서버에 연결하려면 **--connect** 옵션도 전달합니다. 자세한 내용은 **--help** 옵션을 사용합니다.*

***GUI** 모드를 시작한 후 아래로 스크롤하여 하위 시스템**=keycloak-server** 를 찾습니다. 노드를 마우스 오른쪽 버튼으로 클릭하고 **Explore subsystem=keycloak-server** 를 클릭하면 **keycloak-server** 하위 시스템만 표시되는 새 탭이 표시됩니다.*



4.5. CLI 스크립팅

CLI에는 광범위한 스크립팅 기능이 있습니다. 스크립트는 CLI 명령이 포함된 텍스트 파일일 뿐입니다. 테마 및 템플릿 캐싱을 해제하는 간단한 스크립트를 고려하십시오.

turn-off-caching.cli

```
/subsystem=keycloak-server/theme=defaults/:write-attribute(name=cacheThemes,value=false)
/subsystem=keycloak-server/theme=defaults/:write-attribute(name=cacheTemplates,value=false)
```

스크립트를 실행하려면 CLI GUI의 스크립트 메뉴를 따르거나 명령줄에서 다음과 같이 스크립트를 실행할 수 있습니다.

```
$ .../bin/jboss-cli.sh --file=turn-off-caching.cli
```

4.6. CLI RECIPES

다음은 일부 구성 작업과 **CLI** 명령을 사용하여 수행하는 방법입니다. 첫 번째 예제를 제외한 첫 번째 예제에서는 와일드카드 경로 ** 를 사용하여 대체해야 하거나 **keycloak-server** 하위 시스템 경로를 사용해야 합니다.

독립 실행형의 경우 다음을 의미합니다.

**** = /subsystem=keycloak-server**

도메인 모드의 경우 다음과 같습니다.

**** = /profile=auth-server-clustered/subsystem=keycloak-server**

4.6.1. 서버의 웹 컨텍스트 변경

```
/subsystem=keycloak-server/:write-attribute(name=web-context,value=myContext)
```

4.6.2. 글로벌 기본 테마 설정

```
**/theme=defaults/:write-attribute(name=default,value=myTheme)
```

4.6.3. 새 SPI 및 공급자 추가

```
**/spi=mySPI/:add
**/spi=mySPI/provider=myProvider/:add(enabled=true)
```

4.6.4. 공급자 비활성화

```
**/spi=mySPI/provider=myProvider/:write-attribute(name=enabled,value=false)
```

4.6.5. SPI의 기본 공급자 변경

```
**/spi=mySPI/:write-attribute(name=default-provider,value=myProvider)
```

4.6.6. dblock SPI 구성

```
**/spi=dblock/:add(default-provider=jpa)
**/spi=dblock/provider=jpa/:add(properties={lockWaitTimeout => "900"},enabled=true)
```

4.6.7. 공급자의 단일 속성 값 추가 또는 변경

```
**/spi=dblock/provider=jpa/:map-put(name=properties,key=lockWaitTimeout,value=3)
```

4.6.8. 공급자에서 단일 속성 제거

```
**/spi=dblock/provider=jpa/:map-remove(name=properties,key=lockRecheckTime)
```

4.6.9. 공급자 속성의 목록값 설정

```
**/spi=eventsStore/provider=jpa/:map-put(name=properties,key=exclude-events,value=
[EVENT1,EVENT2])
```

5장. PROFILES

Red Hat Single Sign-On에는 기본적으로 활성화되어 있지 않은 기능이 있습니다. 여기에는 완전히 지원되지 않는 기능이 포함되어 있습니다. 또한 기본적으로 활성화되어 있지만 비활성화할 수 있는 몇 가지 기능이 있습니다.

활성화 및 비활성화할 수 있는 기능은 다음과 같습니다.

이름	설명	기본적으로 활성화되어 있음	지원 수준
account2	새 계정 관리 콘솔	없음	프리뷰
account_api	계정 관리 REST API	없음	프리뷰
admin_fine_grained_authz	세분화된 관리자 권한	없음	프리뷰
docker	Docker 레지스트리 프로토콜	없음	지원됨
impersonation	관리자가 사용자를 가장할 수 있는 기능	제공됨	지원됨
openshift_integration	OpenShift 보안을 활성화하는 확장	없음	프리뷰
스크립트	JavaScript를 사용하여 사용자 정의 인증 작성	없음	프리뷰
token_exchange	토큰 교환 서비스	없음	프리뷰
upload_scripts	Red Hat Single Sign-On REST API를 통해 스크립트 업로드	없음	더 이상 사용되지 않음
web_authn	W3C 웹 인증 (WebAuthn)	없음	프리뷰

모든 프리뷰 기능을 활성화하려면 다음을 사용하여 서버를 시작합니다.

```
bin/standalone.sh|bat -Dkeycloak.profile=preview
```

*도메인 모드에서 **server-one** 의 **standalone/configuration/profile.properties** (또는 **domain/servers/server-one/configuration/profile.properties**) 파일을 생성하여 영구적으로 이 값을 설정할 수 있습니다. 파일에 다음을 추가합니다.*

```
profile=preview
```

특정 기능을 활성화하려면 다음을 사용하여 서버를 시작합니다.

```
bin/standalone.sh|bat -Dkeycloak.profile.feature.<feature name>=enabled
```

*예를 들어 **Docker use -Dkeycloak.profile.feature.docker=enabled.***

***profile.properties** 파일에서 이 값을 영구적으로 설정할 수 있습니다.*

```
feature.docker=enabled
```

특정 기능을 비활성화하려면 다음을 사용하여 서버를 시작합니다.

```
bin/standalone.sh|bat -Dkeycloak.profile.feature.<feature name>=disabled
```

*예를 들어 **Impersonation**을 사용하지 않도록 설정하려면 **-Dkeycloak.profile.feature.impersonation=disabled.***

***profile.properties** 파일에서 이 값을 영구적으로 설정할 수 있습니다.*

```
feature.impersonation=disabled
```

6장. 관계형 데이터베이스 설정

Red Hat Single Sign-On은 H2라는 자체 임베디드 Java 기반 관계형 데이터베이스와 함께 제공됩니다. 이는 Red Hat Single Sign-On이 데이터를 저장하는 데 사용할 기본 데이터베이스이며 실제로 인증 서버를 즉시 실행할 수 있도록만 존재합니다. 프로덕션 준비가 된 외부 데이터베이스로 교체하는 것이 좋습니다. H2 데이터베이스는 높은 동시성 상황에서 실행 가능하지 않으며 클러스터에서도 사용해서는 안 됩니다. 이 장의 목적은 Red Hat Single Sign-On을 보다 성숙한 데이터베이스에 연결하는 방법을 보여주는 것입니다.

Red Hat Single Sign-On은 두 가지 계층화된 기술을 사용하여 관계형 데이터를 유지합니다. 아래 계층화된 기술은 JDBC입니다. JDBC는 RDBMS 연결에 사용되는 Java API입니다. 데이터베이스 벤더가 제공하는 데이터베이스 유형당 다양한 JDBC 드라이버가 있습니다. 이 장에서는 이러한 벤더별 드라이버 중 하나를 사용하도록 Red Hat Single Sign-On을 구성하는 방법에 대해 설명합니다.

지속성을 위한 최상위 계층 기술은 Hibernate JPA입니다. Java 개체를 관계형 데이터에 매핑하는 관계형 매핑 API에 대한 개체입니다. Red Hat Single Sign-On의 대부분의 배포는 Hibernate의 구성 측면에 접할 필요가 없지만 드문 경우 이 작업을 수행하는 방법에 대해 논의할 것입니다.

참고

데이터 소스 구성은 JBoss EAP 구성 가이드 의 데이터 소스 구성 장에서 훨씬 더 자세히 다룹니다.

6.1. RDBMS 설정 체크리스트

다음은 Red Hat Single Sign-On에 대해 RDBMS를 구성하기 위해 수행해야 하는 단계입니다.

1. 데이터베이스에 대한 JDBC 드라이버 찾기 및 다운로드
2. 드라이버 JAR을 모듈에 패키지하고 이 모듈을 서버에 설치합니다.
3. 서버의 구성 프로필에서 JDBC 드라이버 선언
4. 데이터베이스의 JDBC 드라이버를 사용하도록 데이터 소스 구성 수정

5. *데이터 소스 구성을 수정하여 데이터베이스에 대한 연결 매개 변수를 정의합니다.*

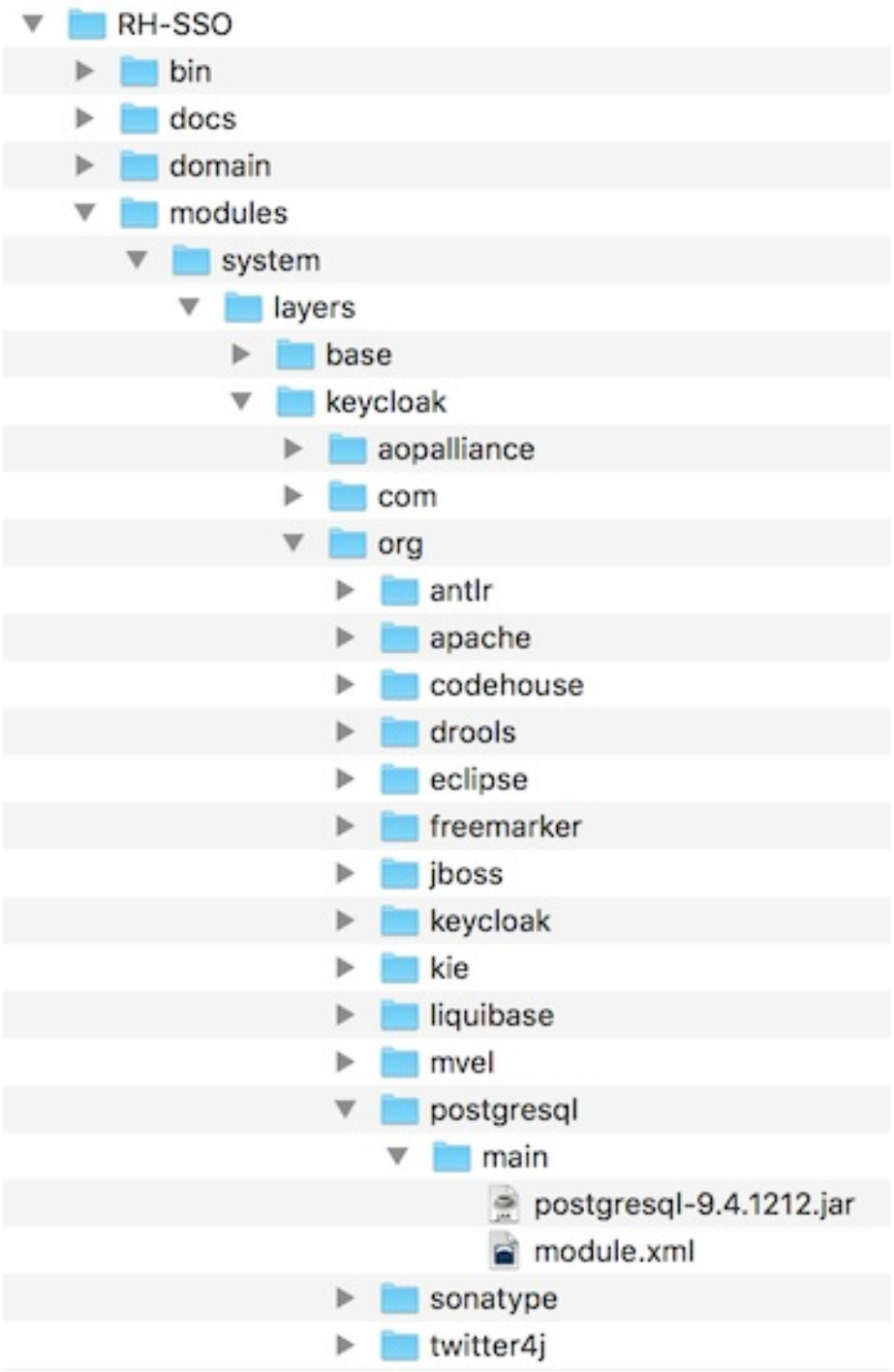
*이 장에서는 모든 예제에 **PostgresSQL**을 사용합니다. 다른 데이터베이스는 설치를 위해 동일한 단계를 따릅니다.*

6.2. JDBC 드라이버 패키지

*RDBMS용 **JDBC** 드라이버 **JAR**을 찾아 다운로드합니다. 이 드라이버를 사용하려면 먼저 모듈에 패키지하여 서버에 설치해야 합니다. 모듈은 **Red Hat Single Sign-On** 클래스 경로에 로드되는 **JAR**과 다른 모듈에 있는 **JAR**의 종속성을 정의합니다. 쉽게 설정할 수 있습니다.*

*Red Hat Single Sign-On 배포의 **…/modules/** 디렉터리에 모듈 정의를 유지할 디렉터리 구조를 생성해야 합니다. 규칙은 디렉터리 구조 이름에 **JDBC** 드라이버의 **Java** 패키지 이름을 사용합니다. **PostgreSQL**의 경우 **org/postgresql/main** 디렉터리를 생성합니다. 데이터베이스 드라이버 **JAR**을 이 디렉터리에 복사하고 여기에 빈 **module.xml** 파일도 생성합니다.*

모듈 디렉토리



*이 작업을 수행한 후 **module.xml** 파일을 열고 다음 **XML**을 생성합니다.*

*모듈 **XML***

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.3" name="org.postgresql">

    <resources>
        <resource-root path="postgresql-9.4.1212.jar"/>
```

```
    </resources>

    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

모듈 이름은 모듈의 디렉터리 구조와 일치해야 합니다. 따라서 **org/postgresql** 는 **org.postgresql** 에 매핑됩니다. **resource-root path** 속성은 드라이버의 **JAR** 파일 이름을 지정해야 합니다. 나머지 부분은 **JDBC** 드라이버 **JAR**에 있는 일반 종속 항목일 뿐입니다.

6.3. JDBC 드라이버 선언 및 로드

다음 작업은 새로 패키징된 **JDBC** 드라이버를 배포 프로필에 선언하여 서버가 부팅될 때 로드되고 사용 가능하게 하는 것입니다. 이 작업을 수행하는 위치는 작동 모드에 따라 다릅니다. 표준 모드로 배포하는 경우 **…/standalone/configuration/standalone.xml** 을 편집합니다. 표준 클러스터링 모드로 배포하는 경우 **…/standalone/configuration/standalone-ha.xml** 을 편집합니다. 도메인 모드로 배포하는 경우 **…/domain/configuration/domain.xml** 을 편집합니다. 도메인 모드에서는 사용 중인 프로필을 편집해야 합니다. **auth-server-standalone** 또는 **auth-server-clustered**

프로필 내에서 데이터 소스 하위 시스템 내에서 **drivers XML** 블록을 검색합니다. **H2 JDBC** 드라이버에 대해 선언된 사전 정의 드라이버가 표시됩니다. 여기에서 외부 데이터베이스에 대한 **JDBC** 드라이버를 선언합니다.

JDBC 드라이버

```
<subsystem xmlns="urn:jboss:domain:datasources:5.0">
    <datasources>
        ...
        <drivers>
            <driver name="h2" module="com.h2database.h2">
                <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
            </driver>
        </drivers>
    </datasources>
</subsystem>
```

drivers XML 블록 내에서 추가 JDBC 드라이버를 선언해야 합니다. 원하는 대로 선택할 수 있는 이름이 있어야 합니다. 드라이버 JAR에 대해 이전에 생성한 모듈 패키지를 가리키는 module 속성을 지정합니다. 마지막으로 드라이버의 Java 클래스를 지정해야 합니다. 다음은 이 장의 앞부분에서 정의한 모듈 예제에 있는 PostgreSQL 드라이버 설치의 예입니다.

JDBC 드라이버 선언

```
<subsystem xmlns="urn:jboss:domain:datasources:5.0">
  <datasources>
    ...
    <drivers>
      <driver name="postgresql" module="org.postgresql">
          <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
      </driver>
      <driver name="h2" module="com.h2database.h2">
          <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
      </driver>
    </drivers>
  </datasources>
</subsystem>
```

6.4. RED HAT SINGLE SIGN-ON 데이터 소스 수정

JDBC 드라이버를 선언한 후에는 Red Hat Single Sign-On에서 새 외부 데이터베이스에 연결하는 데 사용하는 기존 데이터 소스 구성을 수정해야 합니다. JDBC 드라이버를 등록한 것과 동일한 구성 파일 및 XML 블록 내에서 이 작업을 수행합니다. 다음은 새 데이터베이스에 대한 연결을 설정하는 예입니다.

JDBC 드라이버 선언

```
<subsystem xmlns="urn:jboss:domain:datasources:5.0">
  <datasources>
    ...
    <datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-name="KeycloakDS"
enabled="true" use-java-context="true">
      <connection-url>jdbc:postgresql://localhost/keycloak</connection-url>
      <driver>postgresql</driver>
      <pool>
        <max-pool-size>20</max-pool-size>
      </pool>
      <security>
        <user-name>William</user-name>
```

```
            <password>password</password>
        </security>
    </datasource>
    ...
  </datasources>
</subsystem>
```

KeycloakDS 에 대한 데이터 소스 정의를 검색합니다. 먼저 connection-url 을 수정해야 합니다. 벤더의 JDBC 구현에 대한 문서는 이 연결 URL 값의 형식을 지정해야 합니다.

다음으로 사용할 드라이버 를 정의합니다. 이 장의 이전 섹션에서 선언한 JDBC 드라이버의 논리적 이름입니다.

트랜잭션을 수행하려는 때마다 데이터베이스에 대한 새 연결을 여는 것은 비용이 많이 듭니다. 이를 보완하기 위해 데이터 소스 구현에서는 오픈 연결 풀을 유지 관리합니다. max-pool-size 는 풀할 최대 연결 수를 지정합니다. 시스템의 부하에 따라 이 값을 변경할 수 있습니다.

마지막으로 PostgreSQL을 사용하여 데이터베이스에 연결하는 데 필요한 데이터베이스 사용자 이름과 암호를 정의해야 합니다. 이 예제에서는 명확한 텍스트로 표시된다고 우려할 수 있습니다. 이를 난독화하는 방법이 있지만 이 가이드의 범위를 벗어납니다.

참고

데이터 소스 기능에 대한 자세한 내용은 JBoss EAP 구성 가이드의 데이터 소스 구성 장을 참조하십시오.

6.5. 데이터베이스 설정

이 구성 요소의 구성은 배포판의 standalone.xml,standalone-ha.xml 또는 domain.xml 파일에 있습니다. 이 파일의 위치는 작동 모드에 따라 다릅니다.

데이터베이스 구성

```
<subsystem xmlns="urn:jboss:domain:keycloak-server:1.1">
  ...
  <spi name="connectionsJpa">
```

```
    <provider name="default" enabled="true">
      <properties>
        <property name="dataSource" value="java:jboss/datasources/KeycloakDS"/>
        <property name="initializeEmpty" value="false"/>
        <property name="migrationStrategy" value="manual"/>
        <property name="migrationExport" value="${jboss.home.dir}/keycloak-database-update.sql"/>
      </properties>
    </provider>
  </spi>
  ...
</subsystem>
```

가능한 구성 옵션은 다음과 같습니다.

dataSource

dataSource의 JNDI 이름

jta

데이터 소스가 JTA 가능 여부를 지정하는 부울 속성

driverDialect

데이터베이스 대화 상자의 값입니다. 대부분의 경우 이 속성을 지정할 필요가 없습니다.linect는 Hibernate에 의해 자동으로 감지됩니다.

initializeEmpty

비어 있는 경우 데이터베이스를 초기화합니다. false로 설정하면 데이터베이스를 수동으로 초기화해야 합니다. 데이터베이스 세트 migrationStrategy를 manual 로 직접 초기화하려면 데이터베이스를 초기화할 SQL 명령으로 파일을 생성합니다. 기본값은 true입니다.

migrationStrategy

데이터베이스를 마이그레이션하는 데 사용하는 전략입니다. 유효한 값은 update,manual 및 validate 입니다. 업데이트하면 데이터베이스 스키마가 자동으로 마이그레이션됩니다. Manual은 데이터베이스에서 수동으로 실행할 수 있는 SQL 명령을 사용하여 필요한 변경 사항을 파일로 내보냅니다. 유효성 검사는 단순히 데이터베이스가 최신 상태인지 확인합니다.

migrationExport

수동 데이터베이스 초기화/마이그레이션 파일을 작성할 위치에 대한 경로입니다.

showSql

Hibernate에 콘솔의 모든 SQL 명령(기본적으로 false)을 표시할지 여부를 지정합니다. 이것은 매우 자세한 것입니다!

formatSql

Hibernate가 SQL 명령을 포맷해야 하는지 여부를 지정합니다(기본적으로 true).

globalStatsInterval

실행된 DB 쿼리 및 기타 사항에 대해 Hibernate의 글로벌 통계를 기록합니다. 통계는 항상 지정된 간격(초)에 서버 로그에 보고되며 각 보고서 후에는 지워집니다.

스키마

사용할 데이터베이스 스키마 지정

참고

이러한 구성 스위치는 JBoss EAP 개발 가이드에 설명되어 있습니다.

6.6. 데이터베이스에 대한 유니코드 고려 사항

Red Hat Single Sign-On의 데이터베이스 스키마는 다음 특수 필드의 유니코드 문자열만 고려합니다.

- *realms: 표시 이름, HTML 표시 이름*
- *페더레이션 공급자: 표시 이름*
- *사용자 이름: 사용자 이름, 이름, 성, 특성 이름 및 값*
- *groups: 이름, 속성 이름 및 값*
- *역할: 이름*
- *오브젝트 설명*

그렇지 않으면 문자는 종종 8비트인 데이터베이스 인코딩에 포함된 문자로 제한됩니다. 그러나 일부 데이터베이스 시스템의 경우 유니코드 문자의 UTF-8 인코딩을 활성화하고 모든 텍스트 필드에서 전체 유니코드 문자를 사용할 수 있습니다. 종종 이는 8비트 인코딩의 경우보다 짧은 최대 문자열 길이만큼 균형을 이루는 경우가 많습니다.

일부 데이터베이스는 유니코드 문자를 처리할 수 있도록 데이터베이스 및/또는 JDBC 드라이버에 특수 설정이 필요합니다. 아래에서 데이터베이스에 대한 설정을 찾으십시오. 데이터베이스가 여기에 나열되어 있는 경우에도 데이터베이스 및 JDBC 드라이버 수준에서 UTF-8 인코딩을 올바르게 처리할 수 있습니다.

기술적으로 모든 필드에 대한 유니코드 지원의 핵심 기준은 데이터베이스가 VARCHAR 및 Cryostat 필드에 설정된 유니코드 문자 설정을 허용하는지 여부입니다. 예인 경우 일반적으로 유니코드가 필드 길이를 희생할 가능성이 높습니다. NVARCHAR 및 N CHAR 필드에서 유니코드만 지원하는 경우 모든 텍스트 필드에 대한 유니코드 지원은 Keycloak 스키마에서 VARCHAR 및 Cryostat 필드를 광범위하게 사용하므로 거의 지원되지 않습니다.

6.6.1. Oracle Database

유니코드 문자는 적절하게 처리되면 데이터베이스가 VARCHAR 및 Cryostat 필드에서 유니코드 지원을 사용하여 생성되었습니다(예: AL32UTF8 문자 세트를 데이터베이스 문자 집합으로 사용). JDBC 드라이버에는 특별한 설정이 필요하지 않습니다.

데이터베이스 문자 집합이 유니코드가 아닌 경우 특수 필드에서 유니코드 문자를 사용하려면 연결 속성 oracle.jdbc.defaultNChar 를 true 로 설정하여 JDBC 드라이버를 구성해야 합니다. oracle.jdbc.convertNcharLiterals 연결 속성을 true 로 설정하는 것은 엄격하게 필요하지는 않지만 의미가 있을 수 있습니다. 이러한 속성은 시스템 속성 또는 연결 속성으로 설정할 수 있습니다. oracle.jdbc.defaultNChar 설정은 성능에 부정적인 영향을 미칠 수 있습니다. 자세한 내용은 Oracle JDBC 드라이버 구성 설명서를 참조하십시오.

6.6.2. Microsoft SQL Server Database

유니코드 문자는 특수 필드에 대해서만 올바르게 처리됩니다. JDBC 드라이버 또는 데이터베이스의 특수 설정은 필요하지 않습니다.

6.6.3. MySQL 데이터베이스

유니코드 문자가 올바르게 처리되면 데이터베이스가 CREATE DATABASE 명령의 VARCHAR 및 Cryostat 필드에서 유니코드 지원을 사용하여 생성되었습니다(예: MySQL 5.5에 설정된 기본 데이터베이스 문자로 설정된 utf8 문자 집합을 사용하는 경우). utf8mb4 문자 세트는 utf8 문자 세트의 다른 스토리지 요구 사항으로 인해 작동하지 않습니다. [1]). 이 경우 바이트가 아닌 지정된 양의 문자를 수용하도록 열이 생성되므로 특수하지 않은 필드에 대한 길이 제한이 적용되지 않습니다. 데이터베이스 기본 문자 집합에서 유니코드를 저장할 수 없는 경우 특수 필드만 유니코드 값을 저장할 수 있습니다.

JDBC 드라이버 설정 측면에서 JDBC 연결 설정에 connection 속성 characterEncoding=UTF-8 을 추가해야 합니다.

6.6.4. PostgreSQL 데이터베이스

데이터베이스 문자 집합이 UTF8 인 경우 유니코드가 지원됩니다. 이 경우 유니코드 문자를 어떤 필드에서든 사용할 수 있으며, 특수하지 않은 필드에 대한 필드 길이는 감소되지 않습니다. JDBC 드라이버의 특수 설정은 필요하지 않습니다.

PostgreSQL 데이터베이스의 문자 세트는 생성 시 결정됩니다. SQL 명령을 사용하여 PostgreSQL 클러스터에 설정된 기본 문자를 확인할 수 있습니다.

```
show server_encoding;
```

기본 문자 세트가 UTF 8이 아닌 경우 다음과 같이 UTF8을 사용하여 데이터베이스를 생성할 수 있습니다.

```
create database keycloak with encoding 'UTF8';
```

---

[1] https://issues.redhat.com/browse/KEYCLOAK-3873추적

7장. 호스트 이름

Red Hat Single Sign-On은 여러 가지 사항에 공개 호스트 이름을 사용합니다. 예를 들어, 암호 재설정 이메일에 전송된 토큰 발행자 필드 및 URL에서 다음을 수행합니다.

Hostname SPI는 요청에 대한 호스트 이름을 구성하는 방법을 제공합니다. 기본 공급자를 사용하면 요청 URI를 기반으로 백엔드 요청을 허용하는 동안 프런트 엔드 요청에 대한 고정 URL을 설정할 수 있습니다. 또한 기본 제공 공급자가 필요한 기능을 제공하지 않는 경우 자체 공급자를 개발할 수도 있습니다.

7.1. 기본 공급자

기본 호스트 이름 공급자는 구성된 frontendUrl 을 frontend 요청의 기본 URL(user-agents의 요청)으로 사용하고 요청 URL을 백엔드 요청(클라이언트의 직접 요청)으로 사용합니다.

프런트 엔드 요청은 Keycloak 서버와 동일한 컨텍스트 경로를 가질 필요가 없습니다. 즉, 내부 URL이 https://10.0.0.10:8080/auth 일 수 있는 반면, 예를 들어 https://example.org/keycloak 또는 https://example.org/keycloak 에 Keycloak을 노출할 수 있습니다.

이렇게 하면 사용자 에이전트(브라우저)에서 공용 도메인 이름을 통해 ${project.name}에 요청을 보낼 수 있지만 내부 클라이언트는 내부 도메인 이름 또는 IP 주소를 사용할 수 있습니다.

이는 authorization_endpoint 에서 프런트 엔드 URL을 사용하는 반면 token_endpoint 는 백엔드 URL을 사용하는 경우 OpenID Connect Discovery 끝점에 반영됩니다. 여기에 인스턴스의 공용 클라이언트가 공용 끝점을 통해 Keycloak에 문의하므로 authorization_endpoint 및 token_endpoint 의 기반이 동일합니다.

Keycloak에 대한 frontendUrl을 설정하려면 add -Dkeycloak.frontendUrl=https://auth.example.org 를 시작에 전달하거나 standalone.xml 에서 구성할 수 있습니다. 아래 예제를 참조하십시오.

```
<spi name="hostname">
    <default-provider>default</default-provider>
    <provider name="default" enabled="true">
        <properties>
            <property name="frontendUrl" value="https://auth.example.com"/>
            <property name="forceBackendUrlToFrontendUrl" value="false"/>
        </properties>
    </provider>
</spi>
```

jboss-cli로 frontendUrl 을 업데이트하려면 다음 명령을 사용합니다.

```
/subsystem=keycloak-server/spi=hostname/provider=fixed:write-
attribute(name=properties.frontendUrl,value="https://auth.example.com")
```

모든 요청이 공용 도메인 이름을 통과하도록 하려면 백엔드 요청에서 프런트 엔드 URL을 사용하도록 강제 적용하고 forceBackendUrlToFrontendUrl 을 true 로 설정하여 .

개별 영역에 대한 기본 프런트 엔드 URL을 덮어쓸 수도 있습니다. 이 작업은 관리자 콘솔에서 수행할 수 있습니다.

공용 도메인에 관리 끝점 및 콘솔을 노출하지 않으려면 속성 adminUrl 을 사용하여 frontendUrl 과 다른 관리자 콘솔의 고정 URL을 설정합니다. 또한 서버 관리 가이드를 참조하는 방법에 대한 자세한 내용은 외부에서 /auth/admin 에 대한 액세스를 차단해야 합니다.

7.2. 사용자 정의 공급자

사용자 정의 호스트 이름 공급자를 개발하려면 org.keycloak.urls.HostnameProvider y 및 org.keycloak.urls.HostnameProvider 를 구현해야 합니다.

사용자 지정 공급자를 개발하는 방법에 대한 자세한 내용은 서버 개발자 가이드 의 서비스 공급자 인터페이스 섹션의 지침을 따르십시오.

8장. 네트워크 설정

Red Hat Single Sign-On은 일부 네트워킹 제한 사항으로 즉시 실행될 수 있습니다. 하나의 경우 모든 네트워크 끝점이 localhost 에 바인딩되므로 인증 서버는 하나의 로컬 시스템에서만 사용할 수 있습니다. HTTP 기반 연결의 경우 80 및 443과 같은 기본 포트를 사용하지 않습니다. HTTPS/SSL은 즉시 구성되지 않고 Red Hat Single Sign-On에는 많은 보안 취약점이 있습니다. 마지막으로 Red Hat Single Sign-On 은 외부 서버에 대한 보안 SSL 및 HTTPS 연결을 수행해야 하는 경우가 있으므로 끝점을 올바르게 검증할 수 있도록 신뢰 저장소가 설정되어야 하는 경우가 많습니다. 이 장에서는 이러한 모든 사항에 대해 설명합니다.

8.1. 주소 바인딩

기본적으로 Red Hat Single Sign-On은 localhost 루프백 주소 127.0.0.1 에 바인딩됩니다. 네트워크에서 인증 서버를 사용할 수 있도록 하려면 매우 유용한 기본값이 아닙니다. 일반적으로 공용 네트워크에 역방향 프록시 또는 로드 밸런서를 배포하고 트래픽을 사설 네트워크의 개별 Red Hat Single Sign-On 서버 인스턴스로 라우팅하는 것이 좋습니다. 두 경우 모두 localhost 가 아닌 다른 항목에 바인딩하도록 네트워크 인터페이스를 설정해야 합니다.

bind 주소를 설정하는 것은 매우 쉬우며 명령행에서 운영 모드 선택 장에 설명된 standalone.sh 또는 domain.sh 부팅 스크립트를 사용하여 수행할 수 있습니다.

```
$ standalone.sh -b 192.168.0.5
```

-b 스위치는 모든 공용 인터페이스의 IP 바인딩 주소를 설정합니다.

또는 명령줄에서 바인딩 주소를 설정하지 않으려면 배포의 프로필 구성을 편집할 수 있습니다. 프로필 구성 파일(운영 모드에따라standalone.xml 또는 domain.xml)을 열고 인터페이스 XML 블록을 찾습니다.

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
    </interface>
    <interface name="public">
        <inet-address value="${jboss.bind.address:127.0.0.1}"/>
    </interface>
</interfaces>
```

공용 인터페이스는 공개적으로 사용할 수 있는 소켓을 생성하는 하위 시스템에 해당합니다. 이러한 하위 시스템 중 하나의 예로는 Red Hat Single Sign-On의 인증 끝점을 제공하는 웹 계층이 있습니다. 관리 인터페이스는 JBoss EAP의 관리 계층에 의해 열리는 소켓에 해당합니다. 특히 jboss-cli.sh 명령줄 인터페이스와 JBoss EAP 웹 콘솔을 사용할 수 있는 소켓입니다.

공용 인터페이스를 보면 특수 문자열 **${jboss.bind.address:127.0.0.1}** 이 있습니다. 이 문자열은 **Java** 시스템 속성을 설정하여 명령줄에서 재정의할 수 있는 **127.0.0.1** 값을 나타냅니다. 예를 들면 다음과 같습니다.

```
$ domain.sh -Djboss.bind.address=192.168.0.5
```

b 는 이 명령에 대한 간략한 표기법일 뿐입니다. 따라서 프로필 구성에서 직접 **bind address** 값을 변경하거나 부팅 시 명령줄에서 이 값을 변경할 수 있습니다.

참고

인터페이스 정의를 설정할 때 사용할 수 있는 더 많은 옵션이 있습니다. 자세한 내용은 **JBoss EAP** 구성 가이드의 네트워크 인터페이스를 참조하십시오.

8.2. 소켓 포트 바인딩

각 소켓에 대해 열린 포트에는 명령줄 또는 구성에서 재정의할 수 있는 사전 정의된 기본값이 있습니다. 이 구성을 설명하기 위해 독립 실행형 모드로 실행 중이어서 **.../standalone/configuration/standalone.xml** 을 엽니다. **socket-binding-group** 을 검색합니다.

```
    <socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
        <socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}"/>
        <socket-binding name="management-https" interface="management"
port="${jboss.management.https.port:9993}"/>
        <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
        <socket-binding name="http" port="${jboss.http.port:8080}"/>
        <socket-binding name="https" port="${jboss.https.port:8443}"/>
        <socket-binding name="txn-recovery-environment" port="4712"/>
        <socket-binding name="txn-status-manager" port="4713"/>
        <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
        </outbound-socket-binding>
    </socket-binding-group>
```

socket-bindings 는 서버에서 열 소켓 연결을 정의합니다. 이러한 바인딩은 사용하는 인터페이스 (바인드 주소)와 열 포트 번호를 지정합니다. 관심 있는 대상은 다음과 같습니다.

http

Red Hat Single Sign-On HTTP 연결에 사용되는 포트를 정의합니다.

https

Red Hat Single Sign-On HTTPS 연결에 사용되는 포트 정의

ajp

이 소켓 바인딩은 **Cryostat** 프로토콜에 사용되는 포트를 정의합니다. 이 프로토콜은 **Apache HTTPD**를 로드 밸런서로 사용할 때 **mod-cluster** 를 함께 사용하여 **Apache HTTPD** 서버에서 사용합니다.

management-http

JBoss EAP CLI 및 웹 콘솔에서 사용하는 **HTTP** 연결을 정의합니다.

예제 **domain.xml** 파일에 여러 **socket-binding-groups** 가 정의되어 있으므로 도메인 모드를 설정하는 경우 소켓 구성은 비트 레시티입니다. **server-group** 정의까지 아래로 스크롤하면 각 **server-group** 에 사용되는 **socket-binding-group** 을 확인할 수 있습니다.

도메인 소켓 바인딩

```
<server-groups>
    <server-group name="load-balancer-group" profile="load-balancer">
        ...
        <socket-binding-group ref="load-balancer-sockets"/>
    </server-group>
    <server-group name="auth-server-group" profile="auth-server-clustered">
        ...
        <socket-binding-group ref="ha-sockets"/>
    </server-group>
</server-groups>
```

참고

socket-binding-group 정의를 설정할 때 더 많은 옵션을 사용할 수 있습니다. 자세한 내용은 **JBoss EAP** 구성 가이드 의 소켓 바인딩 그룹을 참조하십시오.

8.3. HTTPS/SSL 설정

주의

Red Hat Single Sign-On은 기본적으로 **SSL/HTTPS**를 처리하도록 설정되어 있지 않습니다. **Red Hat Single Sign-On** 서버 자체에서 또는 **Red Hat Single Sign-On** 서버 앞에 있는 역방향 프록시에서 **SSL**을 활성화하는 것이 좋습니다.

이 기본 동작은 각 **Red Hat Single Sign-On** 영역의 **SSL/HTTPS** 모드로 정의됩니다. 이 내용은 서버 관리 가이드에서 자세히 설명하지만 이러한 모드에 대한 몇 가지 컨텍스트와 간략한 개요를 제공합니다.

외부 요청

Red Hat Single Sign-On은 **localhost,127.0.0.1,10.x.x ,192.168.x.x, 172.16.x.x** 와 같은 개인 **IP** 주소를 고정하면 **SSL** 없이 즉시 실행할 수 있습니다. 서버에 **SSL/HTTPS**가 구성되어 있지 않거나 비개인 **IP** 주소에서 **HTTP**를 통해 **Red Hat Single Sign-On**에 액세스하려고 하면 오류가 발생합니다.

none

Red Hat Single Sign-On에는 **SSL**이 필요하지 않습니다. 이것은 실제로 사물과 함께 플레이 할 때 개발에만 사용해야합니다.

모든 요청

Red Hat Single Sign-On은 모든 **IP** 주소에 대해 **SSL**이 필요합니다.

각 영역의 **SSL** 모드는 **Red Hat Single Sign-On** 관리 콘솔에서 구성할 수 있습니다.

8.3.1. Red Hat Single Sign-On 서버에 대해 SSL/HTTPS 활성화

HTTPS 트래픽을 처리하기 위해 역방향 프록시 또는 로드 밸런서를 사용하지 않는 경우 **Red Hat Single Sign-On** 서버에 **HTTPS**를 활성화해야 합니다. 여기에는 다음이 포함됩니다.

1. **SSL/HTTP** 트래픽에 대한 개인 키 및 인증서가 포함된 키 저장소 가져오기 또는 생성
2. 이 키 쌍 및 인증서를 사용하도록 **Red Hat Single Sign-On** 서버 구성.

8.3.1.1. 인증서 및 Java 키 저장소 생성

*HTTPS 연결을 허용하려면 **Red Hat Single Sign-On** 서버를 배포하는 웹 컨테이너에서 **HTTPS**를 활성화하기 전에 자체 서명된 인증서 또는 타사 서명된 인증서를 가져와서 **Java** 키 저장소로 가져와야 합니다.*

8.3.1.1.1. 자체 서명된 인증서

*개발 중에 **Red Hat Single Sign-On** 배포를 테스트할 수 있는 타사 서명된 인증서가 없으므로 **Java JDK**와 함께 제공되는 **keytool** 유틸리티를 사용하여 자체 서명된 인증서를 생성해야 합니다.*

```
$ keytool -genkey -alias localhost -keyalg RSA -keystore keycloak.jks -validity 10950
  Enter keystore password: secret
  Re-enter new password: secret
  What is your first and last name?
  [Unknown]:  localhost
  What is the name of your organizational unit?
  [Unknown]:  Keycloak
  What is the name of your organization?
  [Unknown]:  Red Hat
  What is the name of your City or Locality?
  [Unknown]:  Westford
  What is the name of your State or Province?
  [Unknown]:  MA
  What is the two-letter country code for this unit?
  [Unknown]:  US
  Is CN=localhost, OU=Keycloak, O=Test, L=Westford, ST=MA, C=US correct?
  [no]:  yes
```

*서버를 설치하는 시스템의 **DNS** 이름과 관련된 첫 번째 및 성이란 무엇입니까**?** 질문에 대답해야 합니다. 테스트 목적으로 **localhost** 를 사용해야 합니다. 이 명령을 실행하면 에서 **keytool** 명령을 실행한 것과 동일한 디렉터리에 **keycloak.jks** 파일이 생성됩니다.*

*타사 서명된 인증서를 원하지만 없는 경우 **cacert.org** 에서 무료로 인증서를 받을 수 있습니다. 이 작업을 수행하기 전에 먼저 설정해야합니다.*

가장 먼저 해야 할 일은 인증서 요청을 생성하는 것입니다.

```
$ keytool -certreq -alias yourdomain -keystore keycloak.jks > keycloak.careq
```

*여기서 **yourdomain** 은 이 인증서가 생성되는 **DNS** 이름입니다. **keytool**은 요청을 생성합니다.*

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIC2jCCAcICAQAwZTELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk1BMREwDwYDVQQHEwhXZX
N0Zm9y
```

```
ZDEQMA4GA1UEChMHUmVkIEhhdDEQMA4GA1UECxMHUmVkIEhhdDESMBAGA1UEAxMJbG9jY
Wxob3N0
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAr7kck2TaavlEOGbcpi9c0rncY4HhdzmY
Ax2nZfq1eZEaIPqI5aTxwQZzzLDK9qbeAd8Ji79HzSqnRDxNYaZu7mAYhFKHgixsolE3o5Yfzbw1
29RvyeUVe+WZxv5oo9wolVVpdSINIMEL2LaFhtX/c1dqiqYVpfnvFshZQaIg2nL8juzZcBjj4as
H98gIS7khql/dkZKsw9NLvyxgJvp7PaXurX29fNf3ihG+oFrL22oFyV54BWWxXCKU/GPn61EGZGw
Ft2qSIGLdctpMD1aJR2bcnlhEjZKDksjQZoQ5YMXaAGkcYkG6QkgrocDE2YXDbi7GIdf9MegVJ35
2DQMpwIDAQABoDAwLgYJKoZIhvcNAQkOMSEwHzAdBgNVHQ4EFgQUQwlZJBA+fjiDdiVzaO9vrE/i

n2swDQYJKoZIhvcNAQELBQADggEBAC5FRvMkhal3q86tHPBYWBuTtmcSjs4qUm6V6f63frhveWHf
PzRrI1xH272XUIeBk0gtzWo0nNZnf0mMCtUBbHhhDcG82xolikfqibZijoQZCiGiedVjHJFtniDQ
9bMDUOXEMQ7gHZg5q6mJfNG9MbMpQaUVEEFvfGEQQxbiFK7hRWU8S23/d80e8nExgQxdJWJ6v
d0X
MzzFK6j4Dj55bJVuM7GFmfdNC52pNOD5vYe47Aqh8oajHX9XTycVtPXl45rrWAH33ftbrS8SrZ2S
vqIFQeuLL3BaHwpl3t7j2lMWcK1p80laAxEASib/fAwrRHpLHBXRcq6uALUOZl4Alt8=
-----END NEW CERTIFICATE REQUEST-----
```

*이 **ca** 요청을 **CA**로 보냅니다. **CA**는 서명된 인증서를 발급하여 사용자에게 보냅니다. 새 인증서를 가져오기 전에 **CA**의 루트 인증서를 가져와서 가져와야 합니다. 다음과 같이 **CA**에서 인증서를 다운로드하고 (예: **root.crt**) 가져올 수 있습니다.*

```
$ keytool -import -keystore keycloak.jks -file root.crt -alias root
```

*마지막 단계는 새 **CA**가 생성한 인증서를 키 저장소로 가져오는 것입니다.*

```
$ keytool -import -alias yourdomain -keystore keycloak.jks -file your-certificate.cer
```

8.3.1.2. 키 저장소를 사용하도록 Red Hat Single Sign-On 구성

*이제 적절한 인증서가 있는 **Java** 키 저장소가 있으므로 이를 사용하려면 **Red Hat Single Sign-On** 설치를 구성해야 합니다. 먼저 **keystore**를 사용하고 **HTTPS**를 활성화하려면 **standalone.xml,standalone-ha.xml** 또는 **host.xml** 파일을 편집해야 합니다. 그런 다음 키 저장소 파일을 배포의 구성/ 디렉터리로 이동하거나 선택한 위치에 있는 파일을 이동하고 절대 경로를 제공할 수 있습니다. 절대 경로를 사용하는 경우 구성에서 **relative-to** 매개변수를 제거합니다(* 운영 모드 *참조).*

***CLI**를 사용하여 새로운 **security-realm** 요소를 추가합니다.*

```
$ /core-service=management/security-realm=UndertowRealm:add()

$ /core-service=management/security-realm=UndertowRealm/server-identity=ssl:add(keystore-
path=keycloak.jks, keystore-relative-to=jboss.server.config.dir, keystore-password=secret)
```

*도메인 모드를 사용하는 경우 **/host=<host_name>/** 접두사를 사용하여 모든 호스트에서 명령을 실행해야 합니다(모든 호스트에서 **security-realm** 을 생성하기 위해).*

```
$ /host=<host_name>/core-service=management/security-realm=UndertowRealm/server-identity=ssl:add(keystore-path=keycloak.jks, keystore-relative-to=jboss.server.config.dir, keystore-password=secret)
```

독립 실행형 또는 호스트 구성 파일에서 **security-realms** 요소는 다음과 같아야 합니다.

```
<security-realm name="UndertowRealm">
    <server-identities>
        <ssl>
            <keystore path="keycloak.jks" relative-to="jboss.server.config.dir" keystore-password="secret" />
        </ssl>
    </server-identities>
</security-realm>
```

다음으로 독립 실행형 또는 각 도메인 구성 파일에서 **security-realm** 의 인스턴스를 검색합니다. 생성된 영역을 사용하도록 **https-listener** 를 수정합니다.

```
$ /subsystem=undertow/server=default-server/https-listener=https:write-attribute(name=security-realm, value=UndertowRealm)
```

도메인 모드를 사용하는 경우 **/profile=<profile_name>/**과 함께 사용되는 프로필로 명령에 접두사를 붙입니다.

결과 요소 **server name="default-server".**은 하위 시스템 **xmlns="urn:domain:undertow:10.0"**의 하위 요소인 다음 스탠자를 포함해야 합니다.

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0">
  <buffer-cache name="default"/>
  <server name="default-server">
    <https-listener name="https" socket-binding="https" security-realm="UndertowRealm"/>
  ...
</subsystem>
```

8.4. 발신 HTTP 요청

Red Hat Single Sign-On 서버는 안전한 애플리케이션 및 서비스에 대한 브라우저 이외의 **HTTP** 요청을 수행해야 하는 경우가 많습니다. 인증 서버는 **HTTP** 클라이언트 연결 풀을 유지 관리하여 이러한 발신 연결을 관리합니다. **standalone.xml,standalone-ha.xml** 또는 **domain.xml** 에서 구성해야 하는 몇 가지 사항이 있습니다. 이 파일의 위치는 작동 모드에 따라 다릅니다.

HTTP 클라이언트 구성 예

```
<spi name="connectionsHttpClient">
  <provider name="default" enabled="true">
    <properties>
      <property name="connection-pool-size" value="256"/>
    </properties>
  </provider>
</spi>
```

가능한 구성 옵션은 다음과 같습니다.

establish-connection-timeout-millis

소켓 연결을 설정하는 제한 시간입니다.

socket-timeout-millis

발신 요청이 이 시간 동안 데이터를 수신하지 않으면 연결 시간이 초과됩니다.

connection-pool-size

풀에 있을 수 있는 연결 수(기본적으로 **128**개)입니다.

max-pooled-per-route

호스트당 풀링할 수 있는 연결 수(기본적으로 **64**개)입니다.

connection-ttl-millis

밀리초 단위의 최대 연결 시간입니다. 기본적으로 설정되지 않습니다.

max-connection-idle-time-millis

연결 풀에서 유휴 상태로 유지될 수 있는 최대 시간(기본적으로 **900**초)입니다. **Apache HTTP** 클라이언트의 배경 정리 스레드를 시작합니다. 이 검사 및 백그라운드 스레드를 비활성화하려면 **-1** 로 설정합니다.

disable-cookies

기본적으로 **true** 입니다. **true**로 설정하면 쿠키 캐싱이 비활성화됩니다.

client-keystore

Java 키 저장소 파일의 파일 경로입니다. 이 키 저장소에는 양방향 SSL을 위한 클라이언트 인증서가 포함되어 있습니다.

client-keystore-password

클라이언트 키 저장소의 암호입니다. client-keystore 가 설정된 경우 이는 REQUIRED 입니다.

client-key-password

클라이언트 키의 암호입니다. client-keystore 가 설정된 경우 이는 REQUIRED 입니다.

proxy-mappings

발신 HTTP 요청에 대한 프록시 구성을 나타냅니다. 자세한 내용은 프록시 매핑 섹션을 참조하십시오.

disable-trust-manager

발신 요청에 HTTPS가 필요하고 이 config 옵션이 true 로 설정된 경우 truststore를 지정할 필요가 없습니다. 이 설정은 개발 중에만 사용해야 하며 SSL 인증서 확인을 비활성화하므로 프로덕션에서는 사용하지 않아야 합니다. 이는 선택사항 입니다. 기본값은 false입니다.

8.4.1. Outgoing HTTP 요청에 대한 프록시 매핑

Red Hat Single Sign-On에서 보낸 발신 HTTP 요청은 쉼표로 구분된 프록시 매핑 목록을 기반으로 프록시 서버를 선택적으로 사용할 수 있습니다. proxy-mapping은 regex 기반 호스트 이름 패턴의 조합과 hostnamePattern;proxyUri 의 형태로 proxy-uri를 나타냅니다. 예를 들면 다음과 같습니다.

```
.*\.(google|googleapis)\.com;http://www-proxy.acme.com:8080
```

발신 HTTP 요청에 대한 프록시를 확인하려면 대상 호스트 이름이 구성된 호스트 이름 패턴과 일치합니다. 첫 번째 일치 패턴은 사용할 proxy-uri를 결정합니다. 지정된 호스트 이름에 대해 구성된 패턴이 일치하지 않으면 프록시가 사용되지 않습니다.

프록시 서버에 인증이 필요한 경우 이 형식 username:password@ 형식으로 프록시 사용자의 인증 정보를 포함합니다. 예를 들면 다음과 같습니다.

```
.*\.(google|googleapis)\.com;http://user01:pas2w0rd@www-proxy.acme.com:8080
```

proxy-uri의 특수 값 NO_PROXY 를 사용하여 연결된 호스트 이름과 일치하는 호스트에 프록시를 사용하지 않아야 함을 나타낼 수 있습니다. 프록시 매핑 끝에 catch-all 패턴을 지정하여 나가는 모든 요청에 대한 기본 프록시를 정의할 수 있습니다.

다음 예제에서는 프록시 매핑 구성을 보여줍니다.

```
# All requests to Google APIs should use http://www-proxy.acme.com:8080 as proxy
.*\.(google|googleapis)\.com;http://www-proxy.acme.com:8080

# All requests to internal systems should use no proxy
.*\.acme\.com;NO_PROXY

# All other requests should use http://fallback:8080 as proxy
.*;http://fallback:8080
```

이는 다음 **jboss-cli** 명령을 통해 구성할 수 있습니다. 아래 표시된 대로 **regex-pattern**을 올바르게 이스케이프해야 합니다.

```
echo SETUP: Configure proxy routes for HttpClient SPI

# In case there is no connectionsHttpClient definition yet
/subsystem=keycloak-server/spi=connectionsHttpClient/provider=default:add(enabled=true)

# Configure the proxy-mappings
/subsystem=keycloak-server/spi=connectionsHttpClient/provider=default:write-
attribute(name=properties.proxy-mappings,value=[".*\\.(google|googleapis)\\.com;http://www-
proxy.acme.com:8080",".*\\.acme\\.com;NO_PROXY",".*;http://fallback:8080"])
```

jboss-cli 명령을 실행하면 다음 하위 시스템 구성이 생성됩니다. " 문자를 " .로 인코딩해야 합니다.

```
<spi name="connectionsHttpClient">
    <provider name="default" enabled="true">
        <properties>
            <property
            name="proxy-mappings"
            value="[&quot;.*\\.(google|googleapis)\\.com;http://www-
proxy.acme.com:8080&quot;,&quot;.*\\.acme\\.com;NO_PROXY&quot;,&quot;.*;http://fallback:8080&qu
ot;]"/>
        </properties>
    </provider>
</spi>
```

8.4.2. 발신 HTTPS 요청 신뢰 저장소

Red Hat Single Sign-On이 원격 **HTTPS** 엔드포인트에서 호출할 때 신뢰할 수 있는 서버에 연결하기 위해 원격 서버의 인증서를 검증해야 합니다. 이는 메시지 가로채기**(man-in-the-middle)** 공격을 방지하기 위해 필요합니다. 이러한 원격 서버의 인증서 또는 이러한 인증서에 서명한 **CA**의 인증서를 신뢰 저장소에 배치해야 합니다. 이 신뢰 저장소는 **Red Hat Single Sign-On** 서버에서 관리합니다.

신뢰 저장소는 **ID** 브로커, **LDAP ID** 공급자, 이메일을 보낼 때 및 클라이언트 애플리케이션과의 백채널 통신에 안전하게 연결할 때 사용됩니다.

주의

기본적으로 신뢰 저장소 공급자가 구성되지 않았으며 모든 **https** 연결이 **Java**의 **JSSE** 참조 가이드에 설명된 대로 표준 **java truststore** 구성으로 대체됩니다. 신뢰가 설정되지 않은 경우 이러한 발신 **HTTPS** 요청이 실패합니다.

keytool 을 사용하여 새 **truststore** 파일을 생성하거나 신뢰할 수 있는 호스트 인증서를 기존에 추가할 수 있습니다.

```
$ keytool -import -alias HOSTDOMAIN -keystore truststore.jks -file host-certificate.cer
```

truststore는 배포의 **standalone.xml,standalone-ha.xml** 또는 **domain.xml** 파일 내에 구성됩니다. 이 파일의 위치는 작동 모드에 따라 다릅니다. 다음 템플릿을 사용하여 신뢰 저장소 구성을 추가할 수 있습니다.

```
<spi name="truststore">
   <provider name="file" enabled="true">
      <properties>
         <property name="file" value="path to your .jks file containing public certificates"/>
         <property name="password" value="password"/>
         <property name="hostname-verification-policy" value="WILDCARD"/>
         <property name="disabled" value="false"/>
      </properties>
   </provider>
</spi>
```

이 설정에 사용할 수 있는 구성 옵션은 다음과 같습니다.

file

Java 키 저장소 파일의 경로입니다. **HTTPS** 요청은 통신 중인 서버의 호스트를 확인하는 방법이 필요합니다. 이것이 신뢰자가 하는 것입니다. 키 저장소에는 신뢰할 수 있는 호스트 인증서 또는 인증 기관이 하나 이상 포함되어 있습니다. 이 **truststore** 파일에는 보안 호스트의 공용 인증서만 포함되어야 합니다. **disabled** 가 **true**가 아닌 경우 이는 **REQUIRED** 입니다.

암호

truststore의 암호입니다. disabled 가 true가 아닌 경우 이는 REQUIRED 입니다.

hostname-verification-policy

*기본적으로 WILDCARD 입니다. HTTPS 요청의 경우 서버 인증서의 호스트 이름을 확인합니다. ANY 은 호스트 이름이 확인되지 않았음을 의미합니다. WILDCARD 는 하위 도메인 이름(예: *.foo.com)의 와일드카드를 허용합니다. CN 은 호스트 이름과 정확히 일치해야 합니다.*

비활성화됨

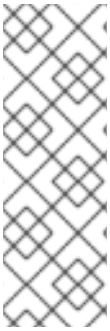
true(기본값)인 경우 truststore 구성이 무시되고 인증서 검사가 설명된 대로 JSSE 구성으로 대체됩니다. false로 설정하는 경우 truststore에 대한 파일 및 암호를 구성해야 합니다.

9장. 클러스터링

이 섹션에서는 클러스터에서 실행되도록 **Red Hat Single Sign-On** 구성에 대해 설명합니다. 특히 클러스터를 설정할 때 수행해야 하는 몇 가지 작업이 있습니다.

- 작업 모드 선택
- 공유 외부 데이터베이스 구성
- 로드 밸런서 설정
- **IP** 멀티 캐스트를 지원하는 프라이빗 네트워크 제공

작업 모드를 선택하고 공유 데이터베이스를 구성하는 방법에 대해서는 이 가이드의 앞부분에서 설명합니다. 이 장에서는 로드 밸런서 설정 및 사설 네트워크 공급에 대해 설명합니다. 또한 클러스터에서 호스트를 부팅할 때 알아야 할 몇 가지 문제에 대해 설명합니다.



참고

IP 멀티 캐스트 없이 **Red Hat Single Sign-On**을 클러스터할 수 있지만 이 가이드의 범위를 벗어납니다. 자세한 내용은 **JBoss EAP** 구성 가이드의 **Cryostat** 장을 참조하십시오. https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/configuration_guide/#cluster_communication_jgroups

9.1. 권장되는 네트워크 아키텍처

Red Hat Single Sign-On 배포를 위해 권장되는 네트워크 아키텍처는 프라이빗 네트워크에 있는 **Red Hat Single Sign-On** 서버로 요청을 라우팅하는 공용 **IP** 주소에 **HTTP/HTTPS** 로드 밸런서를 설정하는 것입니다. 이는 모든 클러스터링 연결을 분리하고 서버를 보호하는 좋은 수단을 제공합니다.

참고

기본적으로 권한이 없는 노드가 클러스터에 가입하고 멀티 캐스트 메시지를 브로드캐스트하지 못하도록 할 수 없습니다. 이로 인해 클러스터 노드가 사설 네트워크에 있어야 하며 방화벽이 외부 공격으로부터 보호되어야 합니다.

9.2. 클러스터링 예

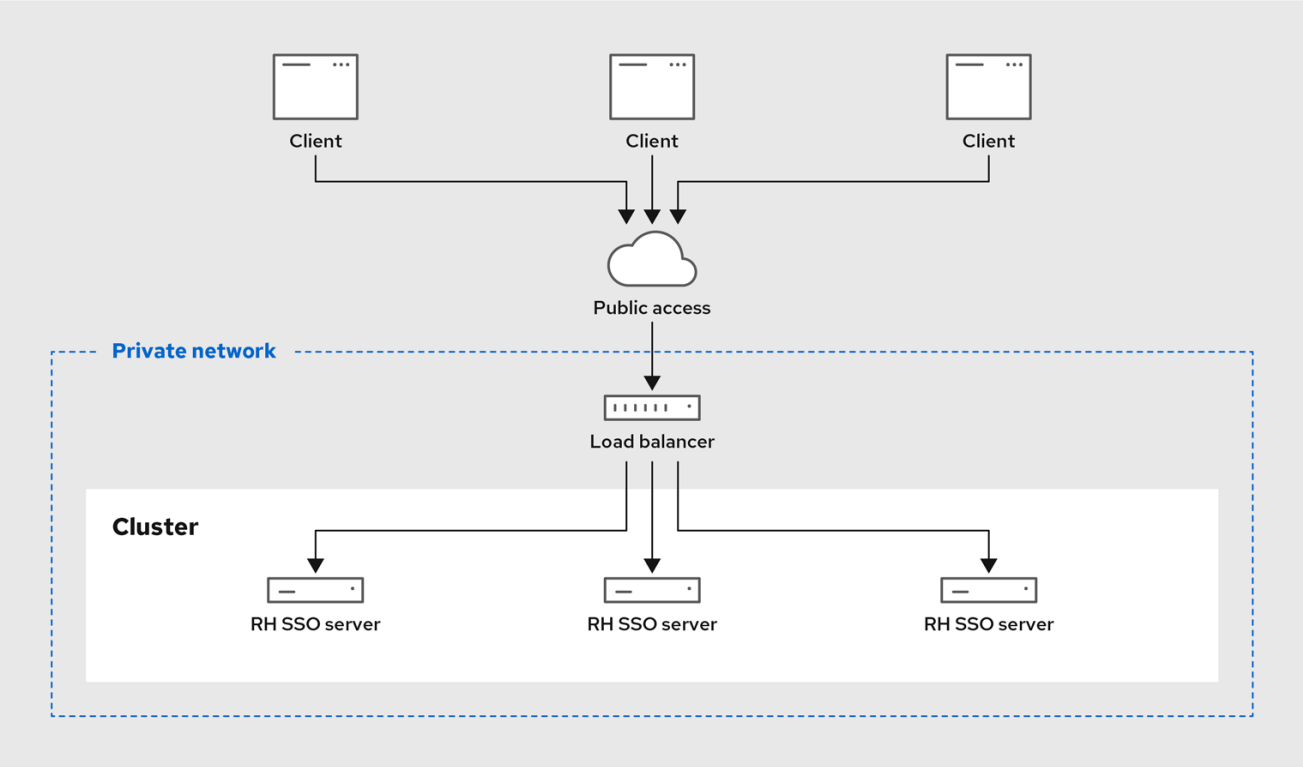
Red Hat Single Sign-On에는 도메인 모드를 활용하는 박스 클러스터링 데모가 포함되어 있습니다. 자세한 내용은 클러스터된 도메인 예제 장을 검토하십시오.

9.3. 로드 밸런서 또는 프록시 설정

이 섹션에서는 클러스터형 **Red Hat Single Sign-On** 배포 앞에 역방향 프록시 또는 로드 밸런서를 배치하기 전에 구성해야 하는 여러 사항에 대해 설명합니다. 또한 클러스터된 도메인 예제 를 사용한 기본 제공 로드 밸런서 구성에 대해 설명합니다.

다음 다이어그램에서는 로드 밸런서를 사용하는 방법을 보여줍니다. 이 예에서 로드 밸런서는 **3**개의 클라이언트와 **3**개의 **Red Hat Single Sign-On** 서버 클러스터 간의 역방향 프록시 역할을 합니다.

로드 밸런서 다이어그램 예



70_RHSSO_0320

9.3.1. 클라이언트 IP 주소 식별

Red Hat Single Sign-On의 몇 가지 기능은 인증 서버에 연결하는 HTTP 클라이언트의 원격 주소가 클라이언트 시스템의 실제 IP 주소입니다. 예를 들면 다음과 같습니다.

- *이벤트 로그 - 실패한 로그인 시도가 잘못된 소스 IP 주소로 기록됨*
- *SSL 필요 - 필요한 SSL이 외부(기본값)로 설정된 경우 모든 외부 요청에 대해 SSL이 필요합니다.*
- *인증 흐름 - IP 주소를 사용하여 외부 요청에 대해서만 OTP를 표시하는 사용자 정의 인증 흐름*
- *동적 클라이언트 등록*

이는 Red Hat Single Sign-On 인증 서버 앞에 역방향 프록시 또는 로드 밸런서가 있는 경우 문제가 될 수 있습니다. 일반적인 설정은 부하 분산 및 요청을 사설 네트워크에 있는 백엔드 Red Hat Single Sign-On 서버 인스턴스로 전달하는 공용 네트워크에 프런트 엔드 프록시가 있다는 것입니다. 이 시나리오에서는 실제 클라이언트 IP 주소가 Red Hat Single Sign-On 서버 인스턴스에서 전달 및 처리하도록 몇 가지 추가 구성이 있습니다. 특히 다음 내용에 유의하십시오.

- *역방향 프록시 또는 로드 밸런서를 구성하여 X-Forwarded-For 및 X-Forwarded-Proto HTTP 헤더를 올바르게 설정합니다.*
- *원래 'Host' HTTP 헤더를 유지하도록 역방향 프록시 또는 로드 밸런서를 구성합니다.*
- *X-Forwarded-For 헤더에서 클라이언트의 IP 주소를 읽도록 인증 서버를 구성합니다.*

X-Forwarded-For 및 X-Forwarded-Proto HTTP 헤더를 생성하고 원래 Host HTTP 헤더를 보존하도록 프록시를 구성하는 것은 이 가이드의 범위를 벗어납니다. 프록시를 통해 X-Forwarded-For 헤더가 설정되어 있는지 확인하기 위해 추가 예방 조치를 취합니다. 프록시가 올바르게 구성되지 않은 경우 악성 클라이언트는 이 헤더를 설정하고 Red Hat Single Sign-On을 사용하여 클라이언트가 실제와 다른 IP 주소에서 연결된다고 생각할 수 있습니다. IP 주소의 검정색 또는 흰색 목록을 수행하는 경우 이는 실제로 중요합니다.

프록시 자체 외에도 Red Hat Single Sign-On 측에서 구성해야 하는 몇 가지 사항이 있습니다. 프록시가 HTTP 프로토콜을 통해 요청을 전달하는 경우 네트워크 패킷 대신 X-Forwarded-For 헤더에서 클라이언트의 IP 주소를 가져오도록 Red Hat Single Sign-On을 구성해야 합니다. 이렇게 하려면 운영 모드에 따

*라 프로필 구성 파일(**standalone.xml**,**standalone-ha.xml** 또는 **domain.xml**)을 열고 **urn:jboss:domain:undertow:10.0 XML** 블록을 찾습니다.*

X-Forwarded-For HTTP Config

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0">
  <buffer-cache name="default"/>
  <server name="default-server">
    <ajp-listener name="ajp" socket-binding="ajp"/>
    <http-listener name="default" socket-binding="http" redirect-socket="https"
       proxy-address-forwarding="true"/>
    ...
  </server>
  ...
</subsystem>
```

***http-listener** 요소에 **proxy-address-forwarding** 속성을 추가합니다. 값을 **true** 로 설정합니다.*

*프록시가 **HTTP** 대신 **requests(**즉, **Apache HTTPD + mod-cluster)**를 사용하는 경우 약간 다르게 구성해야 합니다. **http-listener** 를 수정하는 대신, 이 정보를 패킷에서 가져오기 위해 필터를 추가해야 합니다.*

X-Forwarded-For Cryostat Config

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0">
    <buffer-cache name="default"/>
    <server name="default-server">
       <ajp-listener name="ajp" socket-binding="ajp"/>
       <http-listener name="default" socket-binding="http" redirect-socket="https"/>
       <host name="default-host" alias="localhost">
          ...
          <filter-ref name="proxy-peer"/>
       </host>
    </server>
      ...
    <filters>
       ...
       <filter name="proxy-peer"
             class-name="io.undertow.server.handlers.ProxyPeerAddressHandler"
```

```
            module="io.undertow.core" />
    </filters>
  </subsystem>
```

9.3.2. 역방향 프록시로 HTTPS/SSL 활성화

역방향 프록시에서 **SSL**에 포트 **8443**을 사용하지 않는 경우 리디렉션되는 포트 **HTTPS** 트래픽을 구성해야 합니다.

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0">
    ...
    <http-listener name="default" socket-binding="http"
        proxy-address-forwarding="true" redirect-socket="proxy-https"/>
    ...
</subsystem>
```

http-listener 요소에 **redirect-socket** 속성을 추가합니다. 값은 또한 정의해야 하는 소켓 바인딩을 가리키는 **proxy-https** 여야 합니다.

그런 다음 **socket-binding - group** 요소에 새 **socket-binding** 요소를 추가합니다.

```
<socket-binding-group name="standard-sockets" default-interface="public"
    port-offset="${jboss.socket.binding.port-offset:0}">
    ...
    <socket-binding name="proxy-https" port="443"/>
    ...
</socket-binding-group>
```

9.3.3. 설정 확인

역방향 프록시를 통해 **/auth/realms/master/.well-known/openid-configuration** 경로를 열어 역방향 프록시 또는 로드 밸런서 구성을 확인할 수 있습니다. 예를 들어 역방향 프록시 주소가 **https://acme.com/**인 경우 **URL https://acme.com/auth/realms/master/.well-known/openid-configuration** 을 엽니다. 그러면 **Red Hat Single Sign-On**의 여러 끝점이 나열된 **JSON** 문서가 표시됩니다. 끝점이 역방향 프록시 또는 로드 밸런서의 주소**(scheme,** 도메인 및 포트**)**로 시작하는지 확인합니다. 이렇게 하면 **Red Hat Single Sign-On**이 올바른 엔드포인트를 사용하고 있는지 확인합니다.

또한 **Red Hat Single Sign-On**에 요청에 대한 올바른 소스 **IP** 주소가 표시되는지 확인해야 합니다. 이를 확인하려면 잘못된 사용자 이름 및/또는 암호를 사용하여 관리자 콘솔에 로그인할 수 있습니다. 서버 로그에 다음과 같은 경고가 표시됩니다.

> *08:14:21,287 WARN XNIO-1 task-45 [org.keycloak.events] type=LOGIN_ERROR, realmId=master, clientId=security-admin-console, userId=8f20d7ba-4974-4811-a695-242c8fbd1bf8, ipAddress=X.X.X.X, error=invalid_user_credentials, auth_method=openid-connect, auth_type=code, redirect_uri=http://localhost:8080/auth/admin/master/console/?redirect_fragment=%2Frealms%2Fmaster%2Fevents-settings, code_id=a3d48b67-a439-4546-b992-e93311d6493e, username=admin*

***ipAddress** 값이 역방향 프록시 또는 로드 밸런서의 **IP** 주소가 아닌 로그인하려는 시스템의 **IP** 주소인지 확인합니다.*

9.3.4. Built-In Load Balancer 사용

이 섹션에서는 클러스터형 도메인 예제에 설명된 기본 제공 로드 밸런서 구성에 대해 설명합니다.

***Clustered Domain Example** 은 하나의 시스템에서만 실행되도록 설계되었습니다. 다른 호스트에 슬레이브를 가져오려면 다음을 수행해야 합니다.*

1. *새 호스트 슬레이브를 가리키도록 **domain.xml** 파일을 편집합니다.*
2. *서버 배포를 복사합니다. **domain.xml,host.xml** 또는 **host-master.xml** 파일이 필요하지 않습니다. 또한 **standalone/** 디렉터리도 필요하지 않습니다.*
3. ***host-slave.xml** 파일을 편집하여 사용된 바인드 주소를 변경하거나 명령줄에서 재정의합니다.*

9.3.4.1. 로드 밸런서를 사용하여 새 호스트 등록

*먼저 **domain.xml** 의 로드 밸런서 구성에 새 호스트 슬레이브를 등록합니다. 이 파일을 열고 **load-balancer** 프로필의 **undertow** 구성으로 이동합니다. **reverse-proxy XML** 블록 내에 **remote- host 3** 이라는 새 호스트 정의를 추가합니다.*

domain.xml reverse-proxy config

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0">
 ...
 <handlers>
    <reverse-proxy name="lb-handler">
      <host name="host1" outbound-socket-binding="remote-host1" scheme="ajp" path="/" instance-
```

```
id="myroute1"/>
        <host name="host2" outbound-socket-binding="remote-host2" scheme="ajp" path="/" instance-
id="myroute2"/>
        <host name="remote-host3" outbound-socket-binding="remote-host3" scheme="ajp" path="/"
instance-id="myroute3"/>
      </reverse-proxy>
  </handlers>
  ...
</subsystem>
```

output-socket-binding 은 **domain.xml** 파일에서 나중에 구성된 **socket-binding** 을 가리키는 논리 이름입니다. **instance-id** 속성은 로드 밸런싱 시 고정 세션을 활성화하기 위해 쿠키를 사용하여 이 값을 사용하므로 새 호스트에도 고유해야 합니다.

다음으로 **load-balancer-sockets socket-binding-group** 으로 이동하고 **remote-host3** 에 대한 **outbound-socket-binding** 을 추가합니다. 이 새 바인딩은 새 호스트의 호스트와 포트를 가리켜야 합니다.

domain.xml outbound-socket-binding

```
<socket-binding-group name="load-balancer-sockets" default-interface="public">
    ...
    <outbound-socket-binding name="remote-host1">
        <remote-destination host="localhost" port="8159"/>
    </outbound-socket-binding>
    <outbound-socket-binding name="remote-host2">
        <remote-destination host="localhost" port="8259"/>
    </outbound-socket-binding>
    <outbound-socket-binding name="remote-host3">
        <remote-destination host="192.168.0.5" port="8259"/>
    </outbound-socket-binding>
</socket-binding-group>
```

9.3.4.2. 마스터 바인딩 주소

다음으로 해야 할 일은 마스터 호스트의 공용 및 관리 바인딩 주소를 변경하는 것입니다. 주소 바인딩 장에 설명된 대로 **domain.xml** 파일을 편집하거나 다음과 같이 명령줄에서 이러한 바인딩 주소를 지정합니다.

```
$ domain.sh --host-config=host-master.xml -Djboss.bind.address=192.168.0.2 -
Djboss.bind.address.management=192.168.0.2
```

9.3.4.3. 호스트 슬라브 바인딩 주소

다음으로 공용,관리 및 도메인 컨트롤러 바인딩 주소 (**jboss.domain.master-address**)를 변경해야 합니다. **host-slave.xml** 파일을 편집하거나 다음과 같이 명령줄에서 지정합니다.

```
$ domain.sh --host-config=host-slave.xml
    -Djboss.bind.address=192.168.0.5
     -Djboss.bind.address.management=192.168.0.5
      -Djboss.domain.master.address=192.168.0.2
```

jboss.bind.address 및 **jboss.bind.address.management** 의 값은 호스트 슬레이브의 **IP** 주소와 관련이 있습니다. **jboss.domain.master.address** 의 값은 마스터 호스트의 관리 주소인 도메인 컨트롤러의 **IP** 주소여야 합니다.

9.3.5. 다른 로드 밸런서 구성

다른 소프트웨어 기반 로드 밸런서 사용 방법에 대한 자세한 내용은 **JBoss EAP** 구성 가이드 의 로드 밸런싱 섹션을 참조하십시오.

9.4. 고정 세션

일반적인 클러스터 배포는 로드 밸런서(**reverse proxy**)와 사설 네트워크의 **Red Hat Single Sign-On** 서버 **2**개로 구성됩니다. 성능을 위해 로드 밸런서에서 특정 브라우저 세션과 관련된 모든 요청을 동일한 **Red Hat Single Sign-On** 백엔드 노드로 전달하는 경우 유용할 수 있습니다.

그 이유는 **Red Hat Single Sign-On**이 현재 인증 세션 및 사용자 세션과 관련된 데이터를 저장하기 위해 적용 대상에서 **Infinispan** 분산 캐시를 사용하고 있기 때문입니다. **Infinispan** 분산 캐시는 기본적으로 하나의 소유자로 구성됩니다. 즉, 특정 세션이 하나의 클러스터 노드에만 저장되고 다른 노드는 액세스하려는 경우 원격으로 세션을 조회해야 합니다.

예를 들어 **ID 123** 이 있는 인증 세션이 **node1** 의 **Infinispan** 캐시에 저장된 다음 **node2** 에서 이 세션을 조회해야 하는 경우 특정 세션 엔터티를 반환하려면 네트워크를 통해 **node1** 에 요청을 보내야 합니다.

특정 세션 엔터티를 로컬에서 항상 사용할 수 있는 경우 유용합니다. 이 경우 고정 세션의 도움을 받아 수행할 수 있습니다. 공용 프런트 엔드 로드 밸런서가 있는 클러스터 환경의 워크플로우와 **Red Hat Single Sign-On** 노드 두 개는 다음과 같을 수 있습니다.

- Red Hat Single Sign-On 로그인 화면을 보려면 초기 요청 전송

- 이 요청은 프런트 엔드 로드 밸런서에서 제공하며 일부 임의 노드(예: node1)로 전달합니다. 엄격하게 말하면, 노드는 임의적 일 필요는 없지만 다른 기준 (클라이언트 IP 주소 등)에 따라 선택할 수 있습니다. 모두 기본 로드 밸런서(reverse proxy)의 구현 및 구성에 따라 달라집니다.

- Red Hat Single Sign-On은 임의의 ID(예: 123)를 사용하여 인증 세션을 생성하여 Infinispan 캐시에 저장합니다.

- Infinispan 분산 캐시는 세션 ID의 해시를 기반으로 세션의 기본 소유자를 할당합니다. 이에 대한 자세한 내용은 Infinispan 문서를 참조하십시오. Infinispan이 이 세션의 소유자가 node2 로 할당되었다고 가정하겠습니다.

- Red Hat Single Sign-On은 < session-id>.<owner-node-id >와 같은 형식으로 AUTH_SESSION_ID 쿠키를 생성합니다. 이 예에서는 123.node2 입니다.

- 브라우저에서 Red Hat Single Sign-On 로그인 화면과 AUTH_SESSION_ID 쿠키를 사용하여 응답이 사용자에게 반환됩니다.

이 시점에서 로드 밸런서가 ID 123 이 있는 인증 세션의 소유자이므로 모든 다음 요청을 node2 로 전달하는 경우 유용합니다. 따라서 Infinispan은 이 세션을 로컬로 조회할 수 있습니다. 인증이 완료되면 인증 세션이 사용자 세션으로 변환되며 ID 123 이 동일하기 때문에 node2 에도 저장됩니다.

고정 세션은 클러스터 설정에 필수는 아니지만 위에서 언급한 이유로 성능에 적합합니다. AUTH_SESSION_ID 쿠키를 고정하도록 로드 밸런서를 구성해야 합니다. 이 작업은 로드 밸런서에 따라 어떻게 됩니까.

시작 중에 시스템 속성 jboss.node.name 을 경로 이름에 해당하는 값과 함께 사용하려면 Red Hat Single Sign-On 측에서 사용하는 것이 좋습니다. 예를 들어 -Djboss.node.name=node1 은 node1 을 사용하여 경로를 식별합니다. 이 경로는 노드가 특정 키의 소유자인 경우 Infinispan 캐시에서 사용되며 AUTH_SESSION_ID 쿠키 에 연결됩니다. 다음은 이 시스템 속성을 사용하는 시작 명령의 예입니다.

```
cd $RHSSO_NODE1
./standalone.sh -c standalone-ha.xml -Djboss.socket.binding.port-offset=100 -
Djboss.node.name=node1
```

일반적으로 프로덕션 환경에서 경로 이름은 백엔드 호스트와 동일한 이름을 사용해야 하지만 필수는

아닙니다. 다른 경로 이름을 사용할 수 있습니다. 예를 들어 사설 네트워크 내에서 **Red Hat Single Sign-On** 서버의 호스트 이름을 숨기려면 다음을 수행합니다.

9.4.1. 경로 추가 비활성화

일부 로드 밸런서는 백엔드 **Red Hat Single Sign-On** 노드를 사용하는 대신 경로 정보를 직접 추가하도록 구성할 수 있습니다. 그러나 위에서 설명한 대로 **Red Hat Single Sign-On**에서 경로를 추가하는 것이 좋습니다. 이는 **Red Hat Single Sign-On**은 특정 세션의 소유자인 엔티티를 인식하고 해당 노드로 라우팅할 수 있기 때문에 성능이 향상됩니다. 이는 반드시 로컬 노드가 아닙니다.

Red Hat Single Sign-On의 **AUTH_SESSION_ID** 쿠키의 경로 정보를 원하는 경우 **Red Hat Single Sign-On**의 **RHSSO_HOME/standalone/configuration/standalone-ha.xml** 파일에 추가하여 다음을 비활성화할 수 있습니다.

```
<subsystem xmlns="urn:jboss:domain:keycloak-server:1.1">
 ...
   <spi name="stickySessionEncoder">
      <provider name="infinispan" enabled="true">
         <properties>
            <property name="shouldAttachRoute" value="false"/>
         </properties>
      </provider>
   </spi>

</subsystem>
```

9.5. 멀티 캐스트 네트워크 설정

기본적으로 클러스터링 지원에는 **IP** 멀티 캐스트가 필요합니다. 멀티 캐스트는 네트워크 브로드캐스트 프로토콜입니다. 이 프로토콜은 부팅 시 클러스터를 검색하고 참여하는데 사용됩니다. 또한 **Red Hat Single Sign-On**에서 사용하는 분산 캐시의 복제 및 무효화에 대한 메시지를 브로드캐스트하는 데도 사용됩니다.

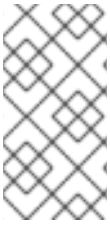
Red Hat Single Sign-On의 클러스터링 하위 시스템은 **Cryostat** 스택에서 실행됩니다. 기본적으로 클러스터링의 바인딩 주소는 **127.0.0.1**을 기본 **IP** 주소로 사용하는 사설 네트워크 인터페이스에 바인딩됩니다. 주소 바인딩 장에 설명된 **standalone-ha.xml** 또는 **domain.xml** 섹션을 편집해야 합니다.

프라이빗 네트워크 구성

```
<interfaces>
   ...
   <interface name="private">
```

```
            <inet-address value="${jboss.bind.address.private:127.0.0.1}"/>
        </interface>
    </interfaces>
    <socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
        ...
        <socket-binding name="jgroups-mping" interface="private" port="0" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
        <socket-binding name="jgroups-tcp" interface="private" port="7600"/>
        <socket-binding name="jgroups-tcp-fd" interface="private" port="57600"/>
        <socket-binding name="jgroups-udp" interface="private" port="55200" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
        <socket-binding name="jgroups-udp-fd" interface="private" port="54200"/>
        <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105" multicast-
port="23364"/>
        ...
    </socket-binding-group>
```

*구성하려는 항목은 **jboss.bind.address.private** 및 **jboss.default.multicast.address** 및 클러스터링 스택의 서비스 포트입니다.*



참고

***IP** 멀티 캐스트 없이 **Red Hat Single Sign-On**을 클러스터할 수 있지만 이 가이드의 범위를 벗어납니다. 자세한 내용은 **JBoss EAP** 구성 가이드 의 **Cryostat**를 참조하십시오.*

9.6. 클러스터 통신 보안

*프라이빗 네트워크에서 클러스터 노드를 분리하면 클러스터에 참여하거나 클러스터의 통신을 볼 수 있도록 프라이빗 네트워크에 액세스해야 합니다. 또한 클러스터 통신에 대한 인증 및 암호화를 활성화할 수도 있습니다. 사설 네트워크가 보안되는 한 인증 및 암호화를 활성화할 필요가 없습니다. **Red Hat Single Sign-On**은 어떠한 경우에도 클러스터에 대한 매우 민감한 정보를 보내지 않습니다.*

*클러스터링 통신에 대한 인증 및 암호화를 활성화하려면 **JBoss EAP** 구성 가이드 의 클러스터 보안 을 참조하십시오.*

9.7. 직렬화된 클러스터 시작

Red Hat Single Sign-On** 클러스터 노드는 동시에 부팅할 수 있습니다. **Red Hat Single Sign-On** 서버 인스턴스가 부팅되면 일부 데이터베이스 마이그레이션, 가져오기 또는 초기화를 수행할 수 있습니다. **DB

잠금은 클러스터 노드를 동시에 부팅할 때 시작 작업이 서로 충돌하지 않도록 하는 데 사용됩니다.

기본적으로 이 잠금의 최대 시간 초과는 **900**초입니다. 노드가 시간 초과보다 이 잠금에서 대기 중인 경우 부팅에 실패합니다. 일반적으로 기본값을 증가/분리할 필요는 없지만 배포에서 **standalone.xml,standalone-ha.xml** 또는 **domain.xml** 파일에서 구성할 수 있는 경우에만 기본값을 늘릴 수 있습니다. 이 파일의 위치는 *작동 모드에* 따라 다릅니다.

```
<spi name="dblock">
    <provider name="jpa" enabled="true">
        <properties>
            <property name="lockWaitTimeout" value="900"/>
        </properties>
    </provider>
</spi>
```

9.8. 클러스터 부팅

클러스터에서 **Red Hat Single Sign-On** 부팅은 *운영 모드에*따라 다릅니다.

독립 실행형 모드

```
$ bin/standalone.sh --server-config=standalone-ha.xml
```

도메인 모드

```
$ bin/domain.sh --host-config=host-master.xml
$ bin/domain.sh --host-config=host-slave.xml
```

추가 매개변수 또는 시스템 속성을 사용해야 할 수도 있습니다. 예를 들어 바인딩 호스트 또는 시스템 속성 **jboss.node.name** 의 매개 변수 **-b** 는 *세션 섹션에 설명된 대로 경로 이름을* 지정합니다.

9.9. 문제 해결

- 클러스터를 실행할 때 두 클러스터 노드의 로그에 다음과 유사한 메시지가 표시됩니다.

```
INFO  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (Incoming-
10,shared=udp)
ISPN000094: Received new cluster view: [node1/keycloak|1] (2) [node1/keycloak,
node2/keycloak]
```

언급된 노드가 하나만 표시되면 클러스터 호스트가 함께 결합되지 않을 수 있습니다.

일반적으로 네트워크 간에 통신하기 위해 방화벽 없이 클러스터 노드를 사설 네트워크에 두는 것이 좋습니다. 대신 공용 액세스 지점에서 방화벽을 활성화할 수 있습니다. 어떤 이유로 클러스터 노드에서 방화벽을 활성화해야 하는 경우 일부 포트를 열어야 합니다. 기본값은 멀티캐스트 주소 **230.0.0.4**가 있는 **UDP** 포트 **55200** 및 멀티 캐스트 포트 **45688**입니다. **Cryostat** 스택에 대한 진단과 같은 추가 기능을 활성화하려면 더 많은 포트가 열려 있어야 할 수 있습니다. **Red Hat Single Sign-On**은 대부분의 클러스터링이 **Infinispan/JGroups**에 작동합니다. 자세한 내용은 **JBoss EAP** 구성 가이드 의 **Cryostat**를 참조하십시오.

- 페일오버 지원(고가용성), 제거, 만료 및 캐시 튜닝에 관심이 있는 경우 **10장. 서버 캐시 구성**을 참조하십시오.

10장. 서버 캐시 구성

Red Hat Single Sign-On에는 두 가지 유형의 캐시가 있습니다. 하나의 캐시 유형은 데이터베이스 앞에 있어 **DB**의 부하를 줄이고 데이터를 메모리에 보관하여 전체 응답 시간을 줄입니다. **realm, client, role, user metadata**는 이러한 유형의 캐시에 보관됩니다. 이 캐시는 로컬 캐시입니다. 더 많은 **Red Hat Single Sign-On** 서버가 있는 경우에도 로컬 캐시는 복제를 사용하지 않습니다. 대신 로컬로 복사본만 보관하고 항목이 업데이트된 경우 클러스터의 나머지 부분으로 전송되고 항목이 제거됩니다. 별도의 복제된 캐시 작업이 있습니다. 로컬 캐시에서 제거해야 하는 항목에 대해 무효화 메시지를 전체 클러스터에 보내는 작업은 무엇입니까. 이렇게 하면 네트워크 트래픽을 크게 줄이고 효율적으로 수행하며, 유선을 통해 민감한 메타데이터를 전송하는 것을 방지할 수 있습니다.

두 번째 유형의 캐시는 사용자 세션 관리, 오프라인 토큰, 로그인 실패 추적을 처리하여 서버가 암호 피싱 및 기타 공격을 감지할 수 있도록 합니다. 이러한 캐시에 저장된 데이터는 메모리에서만 임시이지만 클러스터에 복제될 수 있습니다.

이 장에서는 클러스터형 배포 및 비클러스터형 배포 모두에 대해 이러한 캐시에 대한 몇 가지 구성 옵션에 대해 설명합니다.

참고

이러한 캐시의 고급 구성은 **JBoss EAP** 구성 가이드 의 **Infinispan** 섹션에서 확인할 수 있습니다.

10.1. 제거 및 만료

Red Hat Single Sign-On에 대해 여러 다른 캐시가 구성되어 있습니다. 보안 애플리케이션, 일반 보안 데이터 및 구성 옵션에 대한 정보를 보유하는 영역 캐시가 있습니다. 사용자 메타데이터를 포함하는 사용자 캐시도 있습니다. 둘 다 기본값을 최대 **10000**개의 항목으로 캐시하고 최근에 사용된 제거 전략을 사용합니다. 또한 각 오브젝트는 클러스터형 설정의 제거를 제어하는 오브젝트 리버전 캐시에도 연결됩니다. 이 캐시는 암시적으로 생성되며 구성된 크기가 두 배입니다. 권한 부여 데이터를 보유하는 권한 부여 캐시에도 동일하게 적용됩니다. 키 캐시는 외부 키에 대한 데이터를 보유하고 있으며 전용 버전 캐시가 필요하지 않습니다. 대신 명시적으로 선언된 만료 가 있으므로 키가 주기적으로 만료되고 외부 클라이언트 또는 **ID** 공급자에서 주기적으로 다운로드해야 합니다.

이러한 캐시의 제거 정책 및 **max** 항목은 운영 모드에 따라 **standalone.xml,standalone-ha.xml** 또는 **domain.xml** 에서 구성할 수 있습니다. 구성 파일에는 다음과 유사한 **infinispan** 하위 시스템이 있는 부분이 있습니다.

```
<subsystem xmlns="urn:jboss:domain:infinispan:9.0">
    <cache-container name="keycloak">
        <local-cache name="realms">
            <object-memory size="10000"/>
```

```
    </local-cache>
    <local-cache name="users">
        <object-memory size="10000"/>
    </local-cache>
    ...
    <local-cache name="keys">
        <object-memory size="1000"/>
        <expiration max-idle="3600000"/>
    </local-cache>
    ...
</cache-container>
```

*허용된 항목 수를 제한하거나 확장하려면 오브젝트 요소 또는 특정 캐시 구성의 **expiration** 요소를 추가하거나 편집합니다.*

*또한 별도의 캐시 세션,**clientSessions**,**offlineSessions**,**offlineClientSessions**,**loginFailures** 및 **actionTokens** 도 있습니다. 이러한 캐시는 클러스터 환경에 배포되며 기본적으로 크기가 바인딩되지 않습니다. 바인딩된 경우 일부 세션이 손실될 수 있습니다. 만료된 세션은 **Red Hat Single Sign-On** 자체에서 내부적으로 지워지므로 제한 없이 이러한 캐시의 크기가 늘어나는 것을 방지할 수 있습니다. 많은 세션으로 인해 메모리 문제가 표시되는 경우 다음을 시도할 수 있습니다.*

- *클러스터 크기를 늘립니다. (클러스터의 더 많은 노드는 세션들이 노드 간에 더 균등하게 분배됨을 의미합니다)*
- ***Red Hat Single Sign-On** 서버 프로세스의 메모리 증가*
- *캐시가 한 곳에 저장되도록 소유자 수를 줄입니다. 자세한 내용은 **10.2절. “복제 및 Cryostat”** 에서 참조하십시오.*
- *분산 캐시의 **l1-lifespan**을 비활성화합니다. 자세한 내용은 **Infinispan** 문서를 참조하십시오.*
- *세션 시간 초과 감소: **Red Hat Single Sign-On** 관리 콘솔의 각 영역에 대해 개별적으로 수행할 수 있습니다. 그러나 이는 최종 사용자의 유용성에 영향을 미칠 수 있습니다. 자세한 내용은 시간 초과를 참조하십시오.*

*클러스터 노드 간에 메시지를 보내는 데 주로 사용되는 추가 복제 캐시인 **work** 가 있습니다. 이 캐시는 기본적으로 바인딩되지 않습니다. 그러나 이 캐시의 항목이 매우 짧은 경우 메모리 문제가 발생하지 않아야 합니다.*

10.2. 복제 및 CRYOSTAT

클러스터형 설정을 사용할 때 분산 캐시로 구성된 세션,**authenticationSessions**,**offlineSessions**, **loginFailure** 및 몇 가지 다른 캐시(자세한 내용은 **10.1**절. “제거 및 만료” 참조)와 같은 캐시가 있습니다. 항목이 모든 단일 노드에 복제되지는 않지만 대신 하나 이상의 노드가 해당 데이터의 소유자로 선택됩니다. 노드가 특정 캐시 항목의 소유자가 아닌 경우 해당 노드를 가져오도록 클러스터를 쿼리합니다. 즉, 장애 조치의 의미는 데이터를 소유한 모든 노드가 다운되면 해당 데이터가 영구적으로 손실된다는 것입니다. 기본적으로 **Red Hat Single Sign-On**은 데이터에 대해 하나의 소유자만 지정합니다. 따라서 하나의 노드가 해당 데이터를 중단하면 손실됩니다. 이는 일반적으로 사용자가 로그아웃되고 다시 로그인해야 함을 의미합니다.

distributed-cache 선언의 **owners** 속성을 변경하여 데이터를 복제하는 노드 수를 변경할 수 있습니다.

소유자

```
<subsystem xmlns="urn:jboss:domain:infinispan:9.0">
  <cache-container name="keycloak">
    <distributed-cache name="sessions" owners="2"/>
...
```

여기에서 두 개 이상의 노드가 하나의 특정 사용자 로그인 세션을 복제합니다.

작은 정보

권장되는 소유자 수는 배포에 따라 다릅니다. 노드가 다운될 때 사용자가 로그아웃한 경우 중요하지 않은 경우 한 명의 소유자가 충분하고 복제를 방지할 수 있습니다.

작은 정보

일반적으로 고정 세션과 함께 로드 밸런서를 사용하도록 환경을 구성하는 것이 좋습니다. 특정 요청이 제공되는 **Red Hat Single Sign-On** 서버로는 일반적으로 분산 캐시의 데이터 소유자가 되어 데이터를 로컬에서 검색할 수 있습니다. 자세한 내용은 **9.4**절. “고정 세션”를 참조하십시오.

10.3. 캐싱 비활성화

영역 또는 사용자 캐시를 비활성화하려면 배포에서 **standalone.xml**,**standalone-ha.xml** 또는 **domain.xml** 파일을 편집해야 합니다. 이 파일의 위치는 작동 모드에 따라 다릅니다. 처음에 구성의 모양은 다음과 같습니다.

```
<spi name="userCache">
    <provider name="default" enabled="true"/>
</spi>

<spi name="realmCache">
    <provider name="default" enabled="true"/>
</spi>
```

*비활성화하려는 캐시에 대해 캐시를 비활성화하려면 **enabled** 속성을 **false**로 설정합니다. 이 변경 사항을 적용하려면 서버를 재부팅해야 합니다.*

10.4. 런타임에 캐시 삭제

*영역 또는 사용자 캐시를 지우려면 **Red Hat Single Sign-On** 관리자 콘솔 **settings**으로 이동하여 **Cache Config** 페이지로 이동합니다. 이 페이지에서는 영역 캐시, 사용자 캐시 또는 외부 공개 키의 캐시를 지울 수 있습니다.*

참고

모든 영역에 대해 캐시가 지워집니다!

11장. RED HAT SINGLE SIGN-ON OPERATOR

참고

Red Hat Single Sign-On Operator는 기술 프리뷰 이며 완전히 지원되지 않습니다.

Red Hat Single Sign-On Operator는 **Openshift**에서 **Red Hat Single Sign-On** 관리를 자동화합니다. 이 **Operator**를 사용하여 관리 작업을 자동화하는 **CR**(사용자 정의 리소스)을 생성합니다. 예를 들어 **Red Hat Single Sign-On** 관리 콘솔에서 클라이언트 또는 사용자를 생성하는 대신 사용자 지정 리소스를 생성하여 해당 작업을 수행할 수 있습니다. 사용자 정의 리소스는 관리 작업에 대한 매개 변수를 정의하는 **YAML** 파일입니다.

사용자 정의 리소스를 생성하여 다음 작업을 수행할 수 있습니다.

- **Install Red Hat Single Sign-On**
- 영역 생성
- 클라이언트 생성
- 사용자 생성
- 외부 데이터베이스에 연결
- 데이터베이스 백업 예약
- 확장 기능 및 테마 설치

참고

영역, 클라이언트 및 사용자에 대한 사용자 정의 리소스를 생성한 후 **Red Hat Single Sign-On** 관리 콘솔을 사용하거나 **oc** 명령을 사용하여 사용자 지정 리소스로 관리할 수 있습니다. 그러나 **Operator**는 수정한 사용자 정의 리소스에 대해 한 가지 동기화를 수행하기 때문에 두 가지 방법을 모두 사용할 수 없습니다. 예를 들어 **realm** 사용자 정의 리소스를 수정하면 변경 사항이 관리자 콘솔에 표시됩니다. 그러나 관리 콘솔을 사용하여 영역을 수정하면 이러한 변경 사항이 사용자 정의 리소스에 영향을 미치지 않습니다.

클러스터에 **Red Hat Single Sign-On Operator**를 설치하여 **Operator**를 사용하기 시작합니다.

11.1. 클러스터에 RED HAT SINGLE SIGN-ON OPERATOR 설치

Red Hat Single Sign-On Operator를 설치하려면 다음을 사용합니다.

- OLM(Operator Lifecycle Manager)
- 명령줄 설치

11.1.1. Operator Lifecycle Manager를 사용하여 설치

사전 요구 사항

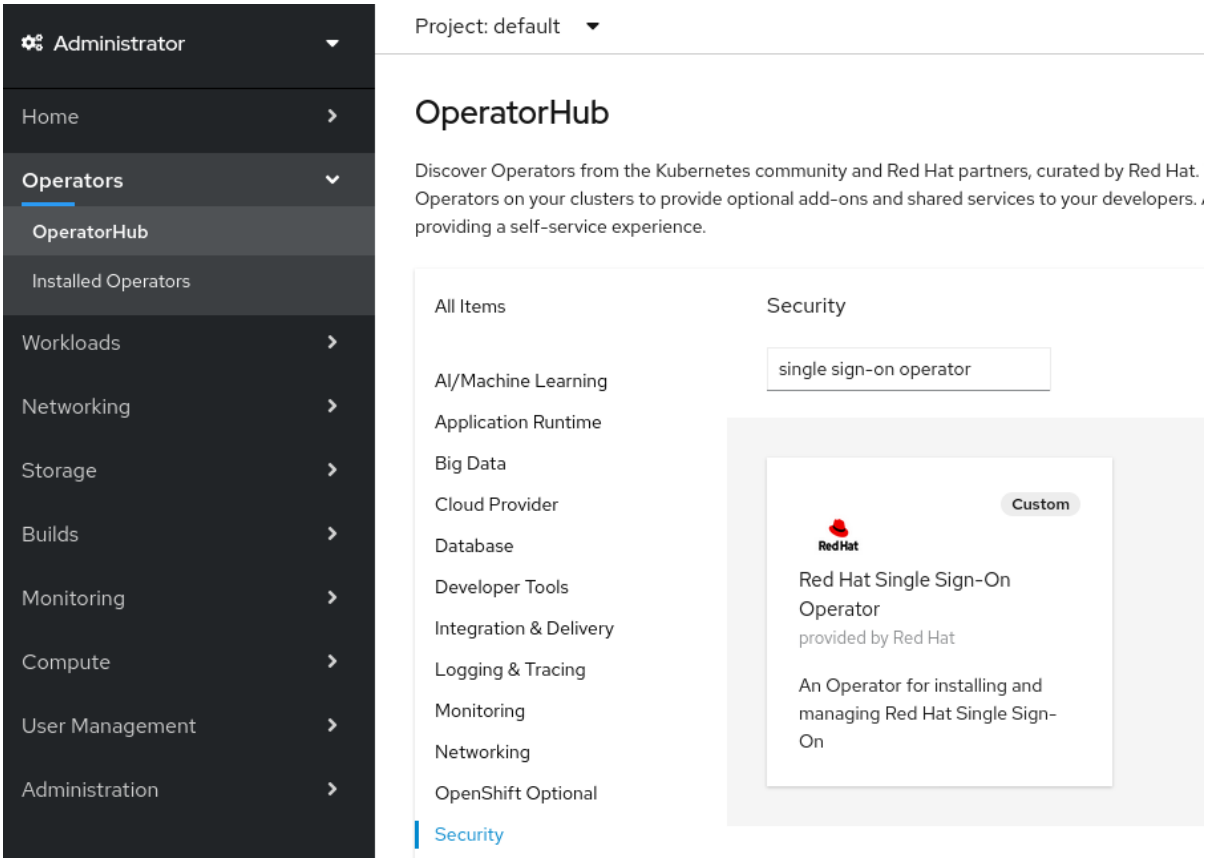
- **cluster-admin** 권한 또는 관리자가 부여한 동등한 수준의 권한이 있어야 합니다.

절차

OpenShift 4.4 클러스터에서 다음 절차를 수행합니다.

1. **OpenShift Container Platform** 웹 콘솔을 엽니다.
2. 왼쪽 열에서 **Operator, OperatorHub**를 클릭합니다.
3. **Red Hat Single Sign-On Operator**를 검색합니다.

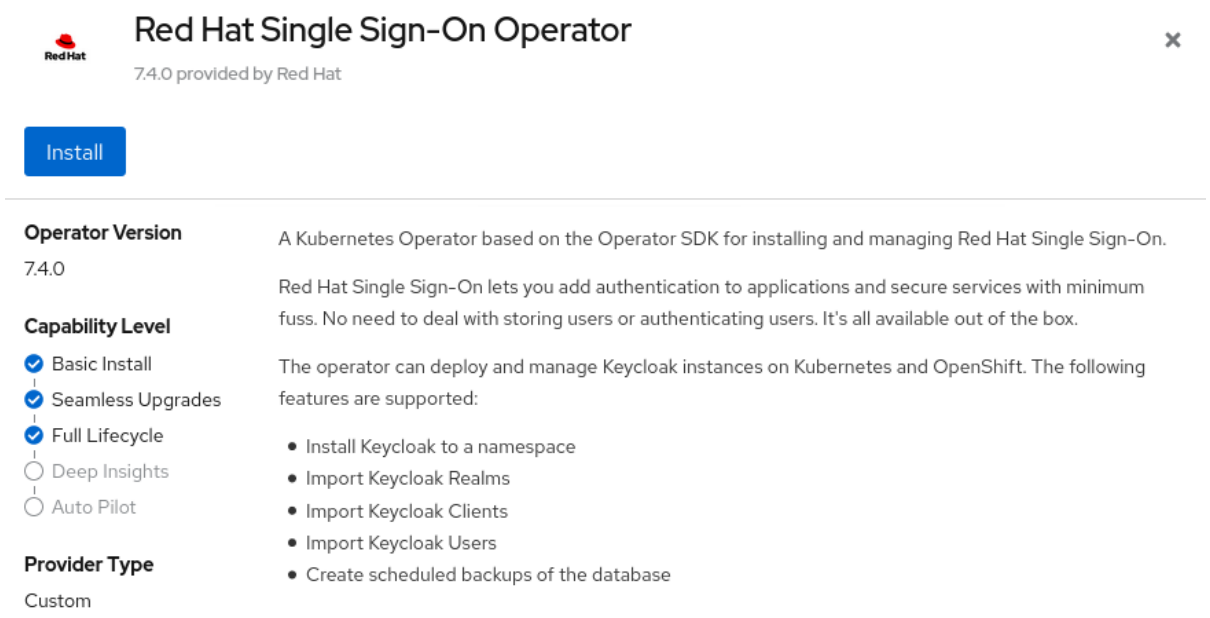
OpenShift의 OperatorHub 탭



4. ***Red Hat Single Sign-On Operator** 아이콘을 클릭합니다.*

   설치 페이지가 열립니다.

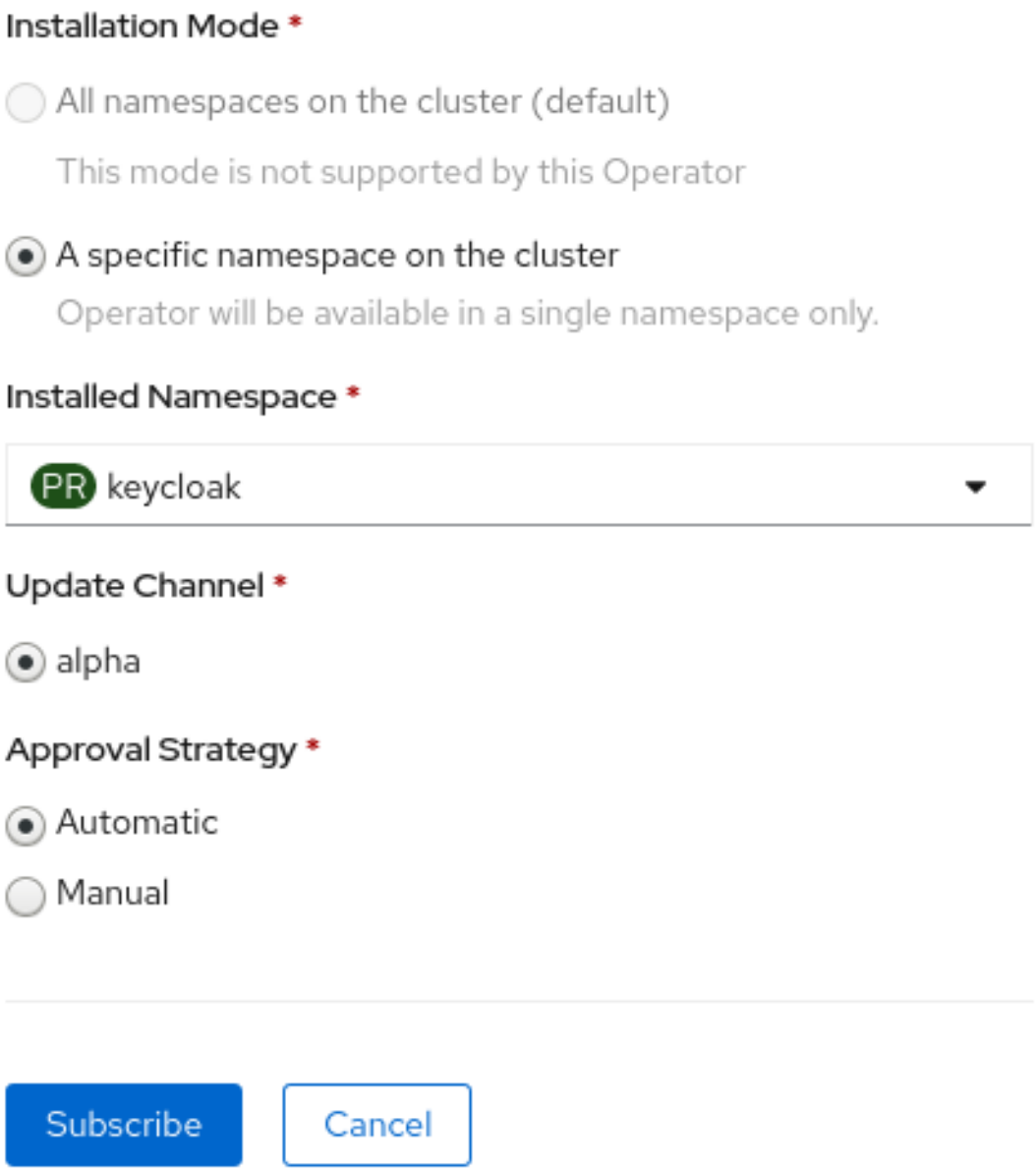
OpenShift에 Operator 설치 페이지



5. *설치를 클릭합니다.*

6. *네임스페이스를 선택하고 서브스크립션을 클릭합니다.*

***OpenShift*의 네임스페이스 선택**



Operator가 설치를 시작합니다.

추가 리소스

- *Operator 설치가 완료되면 첫 번째 사용자 정의 리소스를 생성할 준비가 된 것입니다. 사용자 정의 리소스를 사용한 Red Hat Single Sign-On 설치를 참조하십시오.*
- *OpenShift Operator에 대한 자세한 내용은 OpenShift Operator 가이드를 참조하십시오.*

11.1.2. 명령줄에서 설치

명령줄에서 **Red Hat Single Sign-On Operator**를 설치할 수 있습니다.

사전 요구 사항

- **cluster-admin** 권한 또는 관리자가 부여한 동등한 수준의 권한이 있어야 합니다.

절차

1. 이 위치에서 설치할 소프트웨어를 받으십시오: **Github repo**.
2. 필요한 모든 사용자 정의 리소스 정의를 설치합니다.

```
$ oc create -f deploy/crds/
```

3. 네임스페이스 **myproject** 와 같은 새 네임스페이스를 생성하거나 기존 네임스페이스를 재사용합니다.

```
$ oc create namespace myproject
```

4. **Operator**의 역할, 역할 바인딩 및 서비스 계정을 배포합니다.

```
$ oc create -f deploy/role.yaml -n myproject
$ oc create -f deploy/role_binding.yaml -n myproject
$ oc create -f deploy/service_account.yaml -n myproject
```

5. **Operator**를 배포합니다.

```
$ oc create -f deploy/operator.yaml -n myproject
```

6. **Operator**가 실행 중인지 확인합니다.

```
$ oc get deployment keycloak-operator
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
keycloak-operator   1/1     1            1           41s
```

추가 리소스

-

Operator 설치가 완료되면 첫 번째 사용자 정의 리소스를 생성할 준비가 된 것입니다. 사용자 정의 리소스를 사용한 Red Hat Single Sign-On 설치를 참조하십시오.

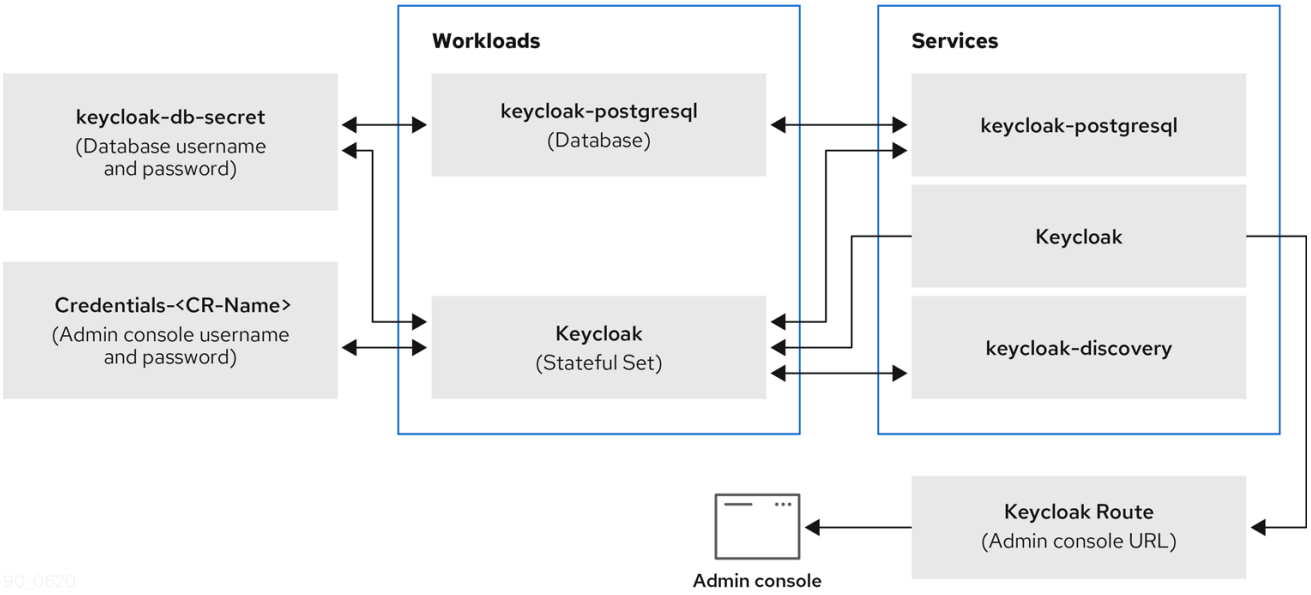
- OpenShift Operator에 대한 자세한 내용은 OpenShift Operator 가이드를 참조하십시오.

11.2. 사용자 정의 리소스를 사용한 RED HAT SINGLE SIGN-ON 설치

Operator를 사용하여 Keycloak 사용자 정의 리소스를 생성하여 Red Hat Single Sign-On 설치를 자동화할 수 있습니다. 사용자 지정 리소스를 사용하여 Red Hat Single Sign-On을 설치할 때 여기에 설명된 구성 요소 및 서비스를 생성하고 다음 그래픽에 설명되어 있습니다.

- Keycloak-db-secret - 데이터베이스 사용자 이름, 암호 및 외부 주소와 같은 속성 저장(외부 데이터베이스에 연결하는 경우)
- credentials-<CR-Name > - Red Hat Single Sign-On 관리자 콘솔에 로그인하기 위한 관리자 사용자 이름과 암호(< CR-Name >은 Keycloak 사용자 정의 리소스 이름을 기반으로 함)
- Keycloak - 고가용성 지원을 통한 StatefulSet으로 구현된 Keycloak 배포 사양
- Keycloak-postgresql - PostgreSQL 데이터베이스 설치 시작
- Keycloak-discovery Service - JDBC_PING 검색을 수행합니다.
- K eycloak 서비스 - HTTPS를 통해 Red Hat Single Sign-On에 연결(HTTP는 지원되지 않음)
- Keycloak-postgresql 서비스 - 내부 및 외부 연결(사용되는 경우 데이터베이스 인스턴스)
- K eycloak 경로 - OpenShift에서 Red Hat Single Sign-On 관리 콘솔에 액세스하기 위한 URL

Operator 구성 요소 및 서비스가 상호 작용하는 방법



11.2.1. Keycloak 사용자 정의 리소스

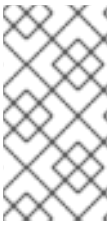
Keycloak 사용자 정의 리소스는 설치에 대한 매개변수를 정의하는 YAML 파일입니다. 이 파일에는 세 개의 속성이 포함되어 있습니다.

- *instances - 고가용성 모드에서 실행 중인 인스턴스 수를 제어합니다.*
- *externalAccess - 활성화된 것이 True 이면 Operator는 Red Hat Single Sign-On 클러스터에 대한 OpenShift의 경로를 생성합니다.*
- *externalDatabase - 외부 호스팅 데이터베이스를 연결하려는 경우에만 적용됩니다. 해당 주제는 이 가이드의 외부 데이터베이스 섹션에서 다룹니다.*

Keycloak 사용자 정의 리소스의 YAML 파일의 예

```
apiVersion: keycloak.org/v1alpha1
kind: Keycloak
metadata:
  name: example-sso
  labels:
    app: sso
spec:
```

```
instances: 1
externalAccess:
  enabled: True
```



참고

YAML 파일을 업데이트할 수 있으며 Red Hat Single Sign-On 관리 콘솔에 변경 사항이 표시되지만 관리 콘솔의 변경 사항은 사용자 정의 리소스를 업데이트하지 않습니다.

11.2.2. OpenShift에서 Keycloak 사용자 정의 리소스 생성

OpenShift에서 사용자 정의 리소스를 사용하여 관리 콘솔의 URL인 경로를 생성하고 관리 콘솔의 사용자 이름과 암호가 있는 시크릿을 찾습니다.

사전 요구 사항

- 이 사용자 정의 리소스에 대한 YAML 파일이 있습니다.
- cluster-admin 권한 또는 관리자가 부여한 동등한 수준의 권한이 있어야 합니다.

절차

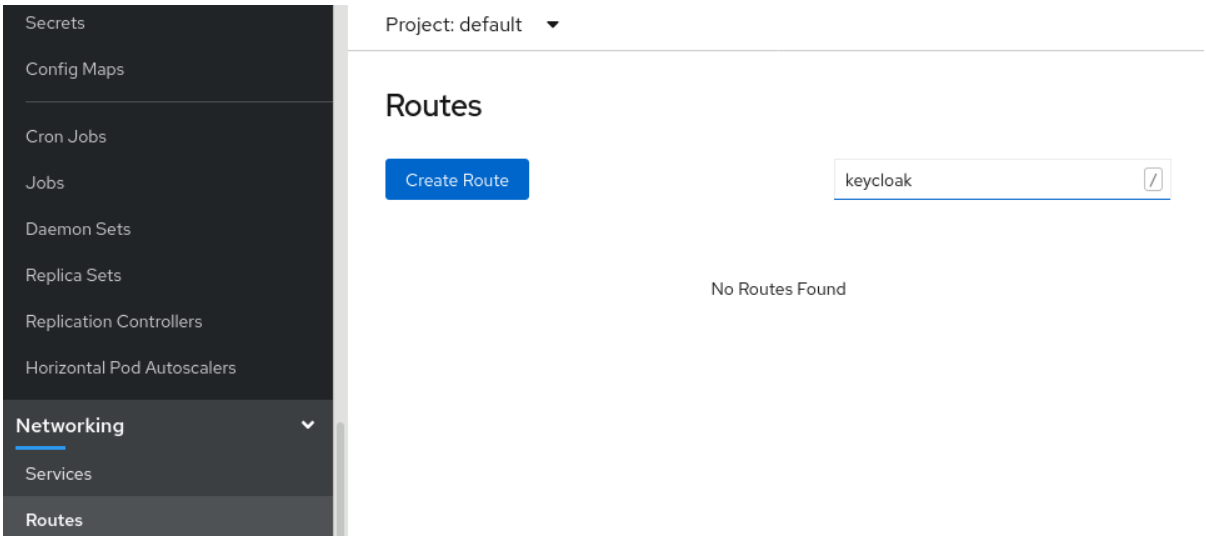
1. YAML 파일 oc create -f <filename>.yaml -n <namespace>를 사용하여 경로를 생성합니다. 예를 들면 다음과 같습니다.

```
$ oc create -f sso.yaml -n sso
keycloak.keycloak.org/example-sso created
```

경로는 OpenShift에서 생성됩니다.

2. OpenShift 웹 콘솔에 로그인합니다.
3. Networking,Routes 를 선택하고 Keycloak을 검색합니다.

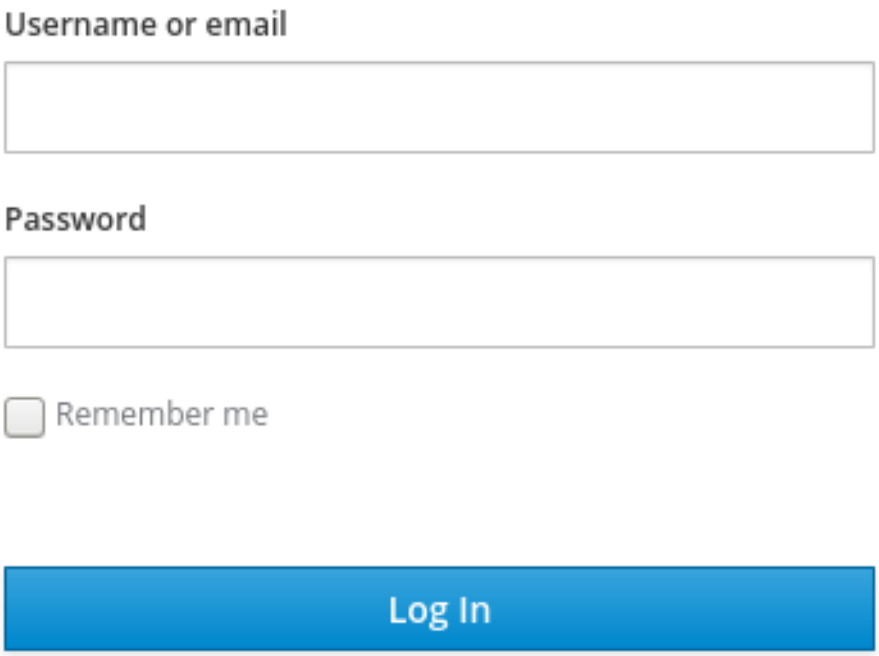
OpenShift 웹 콘솔의 경로 화면



4. ***Keycloak** 경로가 있는 화면에서 위치 아래의 **URL**을 클릭합니다.*

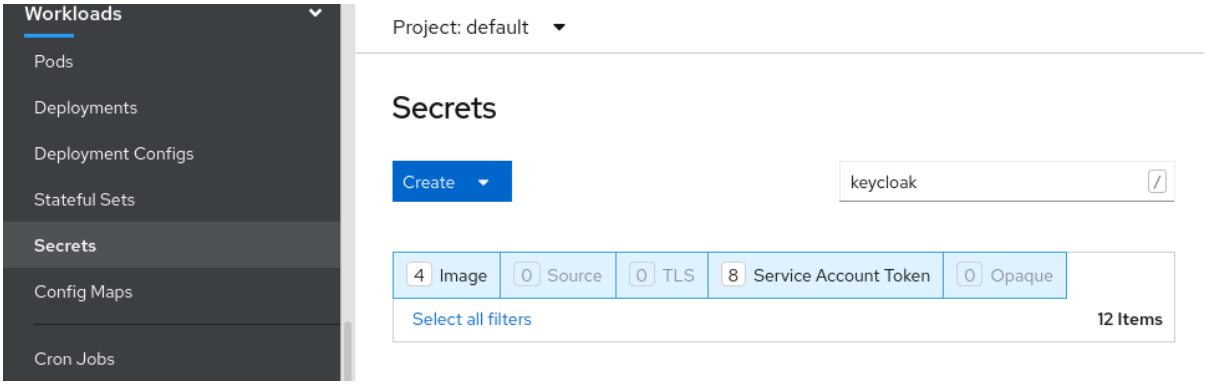
   ***Red Hat Single Sign-On** 관리자 콘솔 로그인 화면이 나타납니다.*

관리자 콘솔 로그인 화면



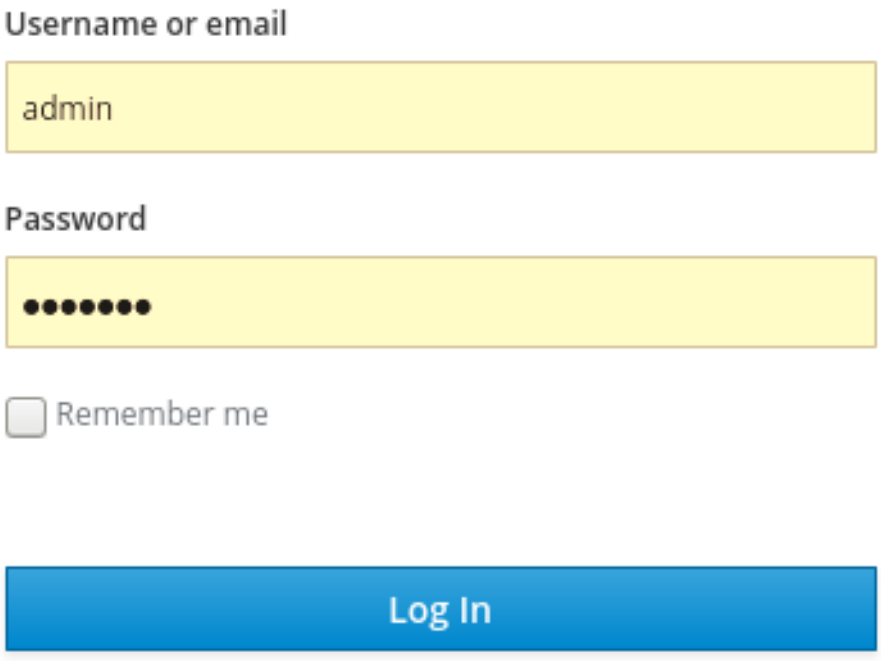
5. ***OpenShift** 웹 콘솔에서 관리자 콘솔의 사용자 이름과 암호를 찾습니다. 워크로드 에서 시크릿을 클릭하고 **Keycloak**을 검색합니다.*

OpenShift 웹 콘솔의 시크릿 화면



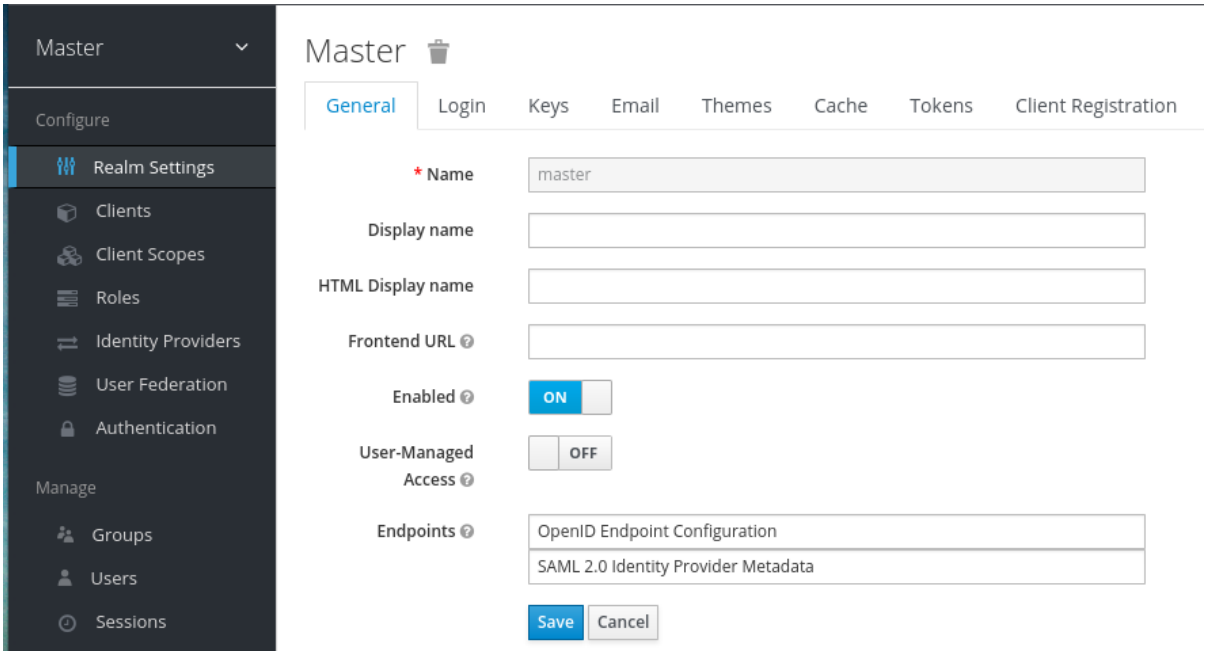
6. *관리자 콘솔 로그인 화면에 사용자 이름과 암호를 입력합니다.*

관리자 콘솔 로그인 화면



*이제 **Keycloak** 사용자 정의 리소스에서 설치한 **Red Hat Single Sign-On** 인스턴스에 로그인했습니다. 영역, 클라이언트 및 사용자에 대한 사용자 정의 리소스를 생성할 준비가 되었습니다.*

Red Hat Single Sign-On 마스터 영역



7. *사용자 정의 리소스의 상태를 확인합니다.*

```
$ oc describe keycloak <CR-name>
```

결과

Operator에서 사용자 정의 리소스를 처리한 후 다음 명령으로 상태를 확인합니다.

```
$ oc describe keycloak <CR-name>
```

Keycloak 사용자 정의 리소스 상태

```
Name:         example-keycloak
Namespace:    keycloak
Labels:       app=sso
Annotations:  <none>
API Version:  keycloak.org/v1alpha1
Kind:         Keycloak
Spec:
  External Access:
    Enabled:  true
  Instances:  1
Status:
  Credential Secret:  credential-example-keycloak
  Internal URL:       https://<External URL to the deployed instance>
  Message:
  Phase:              reconciling
```

```
  Ready:          true
 Secondary Resources:
  Deployment:
   keycloak-postgresql
  Persistent Volume Claim:
   keycloak-postgresql-claim
  Prometheus Rule:
   keycloak
  Route:
   keycloak
  Secret:
   credential-example-keycloak
   keycloak-db-secret
  Service:
   keycloak-postgresql
   keycloak
   keycloak-discovery
  Service Monitor:
   keycloak
  Stateful Set:
   keycloak
 Version:
Events:
```

추가 리소스

- Red Hat Single Sign-On 설치가 완료되면 realm 사용자 정의 리소스를 생성할 준비가 된 것입니다.
- 외부 데이터베이스가 있는 경우 Keycloak 사용자 지정 리소스를 수정하여 이를 지원할 수 있습니다. 외부 데이터베이스에 연결을 참조하십시오.

11.3. REALM 사용자 정의 리소스 생성

Operator를 사용하여 사용자 정의 리소스에서 정의한 대로 Red Hat Single Sign-On에서 영역을 생성할 수 있습니다. YAML 파일에서 realm 사용자 정의 리소스의 속성을 정의합니다.

참고

YAML 파일을 업데이트하고 변경 사항이 Red Hat Single Sign-On 관리 콘솔에 표시될 수 있지만 관리 콘솔의 변경 사항은 사용자 정의 리소스를 업데이트하지 않습니다.

Cryostat 사용자 정의 리소스에 대한 YAML 파일의 예

```
apiVersion: keycloak.org/v1alpha1
kind: KeycloakRealm
metadata:
  name: test
  labels:
    app: sso
spec:
  realm:
    id: "basic"
    realm: "basic"
    enabled: True
    displayName: "Basic Realm"
  instanceSelector:
    matchLabels:
      app: sso
```

사전 요구 사항

- *이 사용자 정의 리소스에 대한 YAML 파일이 있습니다.*
- *YAML 파일에서 instanceSelector 아래의 앱은 Keycloak 사용자 정의 리소스의 레이블과 일치합니다. 이러한 값과 일치하면 Red Hat Single Sign-On의 올바른 인스턴스에 영역을 생성할 수 있습니다.*
- *cluster-admin 권한 또는 관리자가 부여한 동등한 수준의 권한이 있어야 합니다.*

절차

1. *생성한 YAML 파일에서 이 명령을 사용합니다. oc create -f <realm-name>.yaml. 예를 들면 다음과 같습니다.*

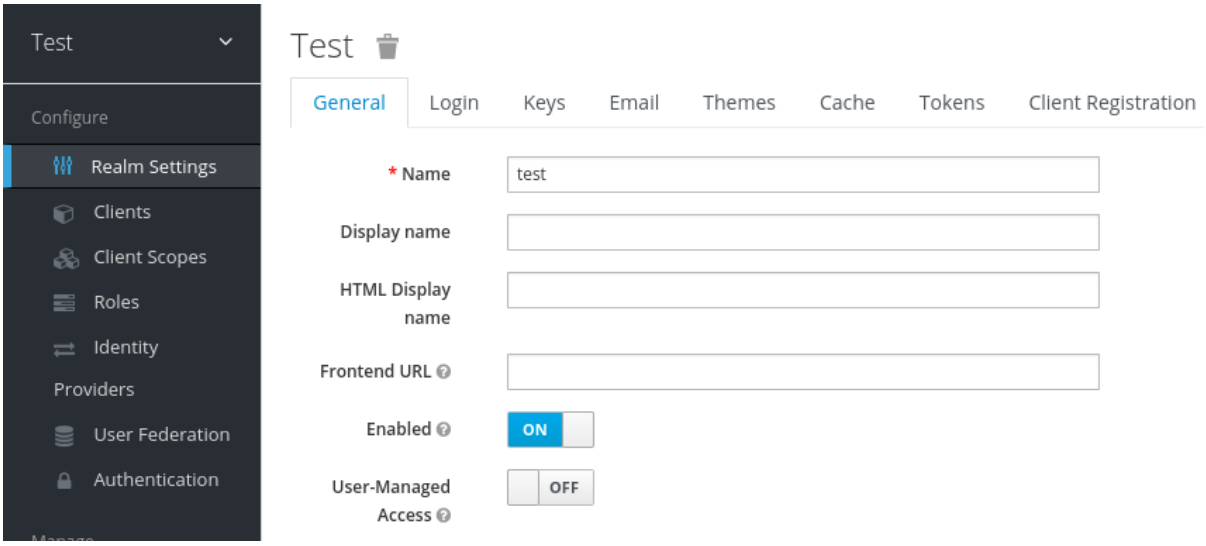
```
$ oc create -f initial_realm.yaml
keycloak.keycloak.org/test created
```

2. *Red Hat Single Sign-On의 관련 인스턴스에 대한 관리자 콘솔에 로그인합니다.*

3. ***Select Cryostat**를 클릭하고 생성한 영역을 찾습니다.*

   새 영역이 열립니다.

관리 콘솔 마스터 영역



결과

***Operator**에서 사용자 정의 리소스를 처리한 후 다음 명령으로 상태를 확인합니다.*

```
$ oc describe keycloak <CR-name>
```

***realm** 사용자 정의 리소스 상태*

```
Name:         example-keycloakrealm
Namespace:    keycloak
Labels:       app=sso
Annotations:  <none>
API Version:  keycloak.org/v1alpha1
Kind:         KeycloakRealm
Metadata:
  Creation Timestamp:  2019-12-03T09:46:02Z
  Finalizers:
    realm.cleanup
  Generation:        1
  Resource Version:  804596
  Self Link:         /apis/keycloak.org/v1alpha1/namespaces/keycloak/keycloakrealms/example-keycloakrealm
```

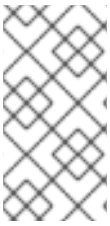
```
  UID:              b7b2f883-15b1-11ea-91e6-02cb885627a6
Spec:
  Instance Selector:
    Match Labels:
      App: sso
  Realm:
    Display Name:  Basic Realm
    Enabled:       true
    Id:            basic
    Realm:         basic
Status:
  Login URL:
  Message:
  Phase:      reconciling
  Ready:      true
Events:       <none>
```

추가 리소스

- 영역 생성이 완료되면 클라이언트 사용자 정의 리소스를 생성할 준비가 된 것입니다.

11.4. 클라이언트 사용자 정의 리소스 생성

Operator를 사용하여 사용자 정의 리소스에서 정의한 대로 Red Hat Single Sign-On에서 클라이언트를 생성할 수 있습니다. YAML 파일에서 영역의 속성을 정의합니다.



참고

YAML 파일을 업데이트하고 변경 사항이 Red Hat Single Sign-On 관리 콘솔에 표시될 수 있지만 관리 콘솔의 변경 사항은 사용자 정의 리소스를 업데이트하지 않습니다.

Client 사용자 정의 리소스에 대한 YAML 파일의 예

```
apiVersion: keycloak.org/v1alpha1
kind: KeycloakClient
metadata:
  name: example-client
  labels:
    app: sso
spec:
```

```
realmSelector:
  matchLabels:
    app: <matching labels for KeycloakRealm custom resource>
client:
 # auto-generated if not supplied
 #id: 123
 clientId: client-secret
 secret: client-secret
 # ...
 # other properties of Keycloak Client
```

사전 요구 사항

- *이 사용자 정의 리소스에 대한 **YAML** 파일이 있습니다.*
- ***cluster-admin** 권한 또는 관리자가 부여한 동등한 수준의 권한이 있어야 합니다.*

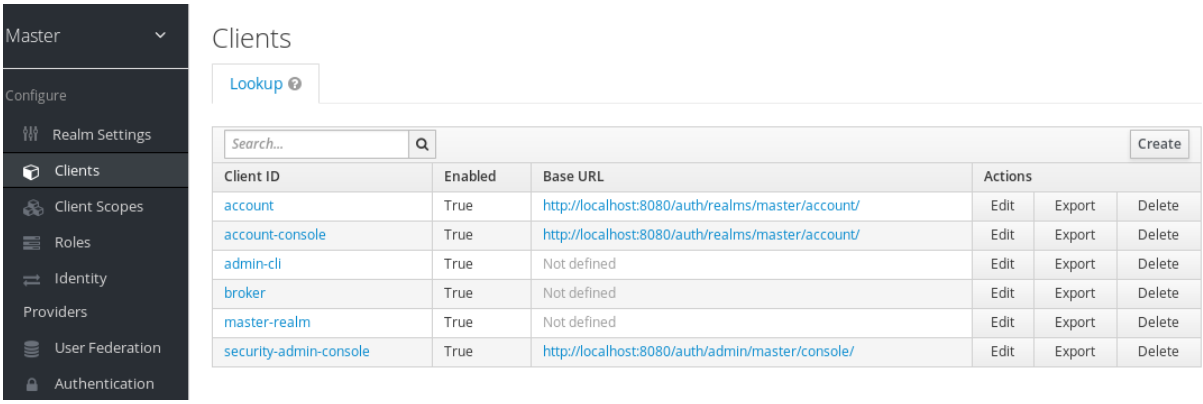
절차

1. *생성한 **YAML** 파일에서 이 명령을 사용합니다. **oc create -f <client-name>.yaml**. 예를 들면 다음과 같습니다.*

```
$ oc create -f initial_client.yaml
keycloak.keycloak.org/example-client created
```

2. ***Red Hat Single Sign-On**의 관련 인스턴스에 대해 **Red Hat Single Sign-On** 관리 콘솔에 로그인합니다.*
3. *클라이언트를 클릭합니다.*

   새 클라이언트가 클라이언트 목록에 나타납니다.



결과

*클라이언트가 생성되면 **Operator**에서 다음 이름 지정 패턴을 사용하여 클라이언트 **ID**와 클라이언트의 시크릿을 포함하는 보안을 생성합니다. **keycloak-client-secret-<custom resource name > .** 예를 들면 다음과 같습니다.*

클라이언트의 시크릿

```
apiVersion: v1
data:
  CLIENT_ID: <base64 encoded Client ID>
  CLIENT_SECRET: <base64 encoded Client Secret>
kind: Secret
```

Operator에서 사용자 정의 리소스를 처리한 후 다음 명령으로 상태를 확인합니다.

```
$ oc describe keycloak <CR-name>
```

클라이언트 사용자 정의 리소스 상태

```
Name:         client-secret
Namespace:    keycloak
Labels:       app=sso
API Version:  keycloak.org/v1alpha1
Kind:         KeycloakClient
Spec:
  Client:
    Client Authenticator Type:     client-secret
```

```
    Client Id:              client-secret
    Id:                     keycloak-client-secret
  Realm Selector:
    Match Labels:
      App:  sso
Status:
  Message:
  Phase:    reconciling
  Ready:    true
  Secondary Resources:
    Secret:
      keycloak-client-secret-client-secret
Events:  <none>
```

추가 리소스

- 클라이언트 생성이 완료되면 사용자 정의 리소스를 생성할 준비가 된 것입니다.

11.5. 사용자 정의 리소스 생성

Operator를 사용하여 사용자 정의 리소스에서 정의한 대로 Red Hat Single Sign-On에서 사용자를 생성할 수 있습니다. YAML 파일에서 사용자 정의 리소스의 속성을 정의합니다.

참고

암호를 제외하고 속성을 업데이트할 수 있습니다. YAML 파일 및 변경 사항은 Red Hat Single Sign-On 관리 콘솔에 표시되지만 관리 콘솔을 변경하면 사용자 정의 리소스가 업데이트되지 않습니다.

사용자 정의 리소스에 대한 YAML 파일의 예

```
apiVersion: keycloak.org/v1alpha1
kind: KeycloakUser
metadata:
  name: example-user
spec:
  user:
    username: "realm_user"
    firstName: "John"
    lastName: "Doe"
    email: "user@example.com"
```

```
  enabled: True
  emailVerified: False
  realmRoles:
    - "offline_access"
  clientRoles:
    account:
      - "manage-account"
    realm-management:
      - "manage-users"
realmSelector:
  matchLabels:
    app: sso
```

사전 요구 사항

- *이 사용자 정의 리소스에 대한 **YAML** 파일이 있습니다.*
- ***realmSelector** 는 기존 영역 사용자 정의 리소스의 레이블과 일치합니다.*
- ***cluster-admin** 권한 또는 관리자가 부여한 동등한 수준의 권한이 있어야 합니다.*

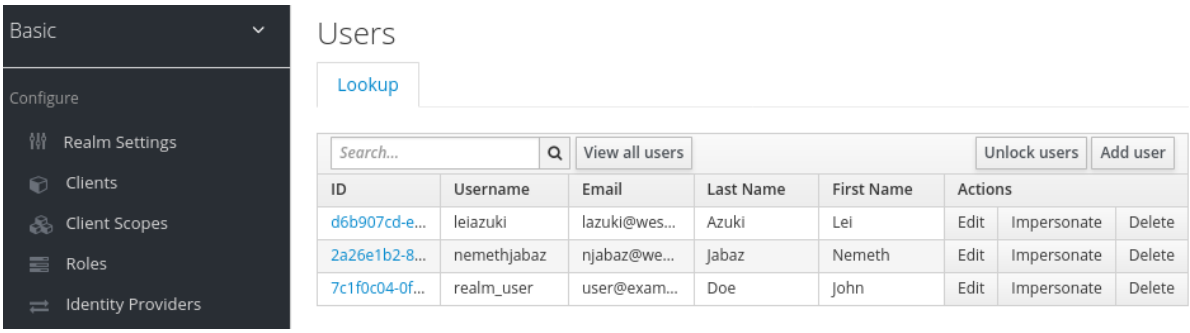
절차

1. *생성한 **YAML** 파일에서 이 명령을 사용합니다. **oc create -f <user_cr>.yaml**. 예를 들면 다음과 같습니다.*

```
$ oc create -f initial_user.yaml
keycloak.keycloak.org/example-user created
```

2. ***Red Hat Single Sign-On**의 관련 인스턴스에 대한 관리자 콘솔에 로그인합니다.*
3. *사용자를 클릭합니다.*
4. ***YAML** 파일에 정의된 사용자를 검색합니다.*

   사용자를 찾으려면 다른 영역으로 전환해야 할 수 있습니다.



결과

*사용자가 생성되면 **Operator**는 다음 이름 지정 패턴을 사용하여 사용자 이름과 암호가 모두 포함된 보안을 생성합니다. **credentials -<realm name>-<username>-<namespace > .** 예를 들면 다음과 같습니다.*

KeycloakUser Secret

```
kind: Secret
apiVersion: v1
data:
  password: <base64 encoded password>
  username: <base64 encoded username>
type: Opaque
```

Operator에서 사용자 정의 리소스를 처리하면 다음 명령으로 상태를 확인합니다.

```
$ oc describe keycloak <CR-name>
```

사용자 정의 리소스 상태

```
Name:         example-realm-user
Namespace:    keycloak
Labels:       app=sso
API Version:  keycloak.org/v1alpha1
Kind:         KeycloakUser
Spec:
  Realm Selector:
```

```
    Match Labels:
      App: sso
  User:
    Email:         realm_user@redhat.com
    Credentials:
      Type:        password
      Value:       <user password>
    Email Verified:  false
    Enabled:         true
    First Name:      John
    Last Name:       Doe
    Username:        realm_user
Status:
  Message:
  Phase:    reconciled
Events:     <none>
```

추가 리소스

- 외부 데이터베이스가 있는 경우 **Keycloak** 사용자 지정 리소스를 수정하여 이를 지원할 수 있습니다. 외부 데이터베이스에 연결을 참조하십시오.

- 사용자 지정 리소스를 사용하여 데이터베이스를 백업하려면 데이터베이스 백업 일정을 참조하십시오.

11.6. 외부 데이터베이스에 연결

Operator를 사용하여 **Keycloak** 사용자 정의 리소스를 수정하고 **keycloak-db-secret YAML** 파일을 생성하여 외부 **PostgreSQL** 데이터베이스에 연결할 수 있습니다. 값은 **Base64**로 인코딩됩니다.

keycloak-db-secret에 대한 **YAML** 파일의 예

```
apiVersion: v1
kind: Secret
metadata:
    name: keycloak-db-secret
    namespace: keycloak
stringData:
    POSTGRES_DATABASE: <Database Name>
    POSTGRES_EXTERNAL_ADDRESS: <External Database IP or URL (resolvable by K8s)>
    POSTGRES_EXTERNAL_PORT: <External Database Port>
    # Strongly recommended to use <'Keycloak CR-Name'-postgresql>
```

```
    POSTGRES_HOST: <Database Service Name>
    POSTGRES_PASSWORD: <Database Password>
    # Required for AWS Backup functionality
    POSTGRES_SUPERUSER: true
    POSTGRES_USERNAME: <Database Username>
  type: Opaque
```

다음 속성은 데이터베이스의 호스트 이름 또는 IP 주소와 포트를 설정합니다.

- POSTGRES_EXTERNAL_ADDRESS - IP 주소 또는 외부 데이터베이스의 호스트 이름입니다.
- POSTGRES_EXTERNAL_PORT - (선택 사항) 데이터베이스 포트입니다.

다른 속성은 호스트 또는 외부 데이터베이스에 대해 동일한 방식으로 작동합니다. 다음과 같이 설정합니다.

- POSTGRES_DATABASE - 사용할 데이터베이스 이름입니다.
- POSTGRES_HOST - 데이터베이스와 통신하는 데 사용되는 서비스 의 이름입니다. 일반적으로 keycloak-postgresql.
- POSTGRES_USERNAME - 데이터베이스 사용자 이름
- POSTGRES_PASSWORD - 데이터베이스 암호
- POSTGRES_SUPERUSER - 백업을 슈퍼 사용자로 실행해야 하는지 여부를 나타냅니다. 일반적으로 true.

Keycloak 사용자 정의 리소스에는 외부 데이터베이스 지원을 활성화하기 위해 업데이트가 필요합니다.

외부 데이터베이스를 지원하는 Keycloak 사용자 정의 리소스의 YAML 파일의 예

```
apiVersion: keycloak.org/v1alpha1
kind: Keycloak
metadata:
  labels:
     app: sso
  name: example-keycloak
  namespace: keycloak
spec:
  externalDatabase:
    enabled: true
  instances: 1
```

사전 요구 사항

- *keycloak-db-secret 에 대한 YAML 파일이 있습니다.*
- *Keycloak 사용자 지정 리소스를 수정하여 externalDatabase 를 true 로 설정했습니다.*
- *cluster-admin 권한 또는 관리자가 부여한 동등한 수준의 권한이 있어야 합니다.*

절차

1. *PostgreSQL 데이터베이스의 시크릿을 찾습니다. oc get secret <secret_for_db> -o yaml. 예를 들면 다음과 같습니다.*

```
$ oc get secret keycloak-db-secret -o yaml
apiVersion: v1
data
  POSTGRES_DATABASE: cm9vdA==
  POSTGRES_EXTERNAL_ADDRESS: MTcyLjE3LjAuMw==
  POSTGRES_EXTERNAL_PORT: NTQzMg==
```

POSTGRES_EXTERNAL_ADDRESS 는 Base64 형식입니다.

2. *secret의 값을 디코딩합니다. echo "<encoded_secret>" | base64 -decode. 예를 들면 다음과 같습니다.*

```
$ echo "MTcyLjE3LjAuMw==" | base64 -decode
192.0.2.3
```

3. 디코딩된 값이 데이터베이스의 IP 주소와 일치하는지 확인합니다.

```
$ oc get pods -o wide
NAME                    READY  STATUS    RESTARTS   AGE   IP
keycloak-0              1/1    Running   0          13m   192.0.2.0
keycloak-postgresql-c8vv27m 1/1    Running   0          24m   192.0.2.3
```

4. keycloak-postgresql 가 실행 중인 서비스 목록에 표시되는지 확인합니다.

```
$ oc get svc
NAME              TYPE       CLUSTER-IP     EXTERNAL-IP  PORT(S)   AGE
keycloak          ClusterIP  203.0.113.0    <none>       8443/TCP  27m
keycloak-discovery   ClusterIP  None           <none>       8080/TCP  27m
keycloak-postgresql  ClusterIP  203.0.113.1    <none>       5432/TCP  27m
```

keycloak-postgresql 서비스는 백엔드의 IP 주소 집합에 요청을 보냅니다. 이러한 IP 주소를 끝점이라고 합니다.

5. keycloak-postgresql 서비스에서 사용하는 끝점을 보고 데이터베이스에 IP 주소를 사용하는지 확인합니다.

```
$ oc get endpoints keycloak-postgresql
NAME                ENDPOINTS         AGE
keycloak-postgresql   192.0.2.3.5432    27m
```

6. Red Hat Single Sign-On이 외부 데이터베이스로 실행되고 있는지 확인합니다. 이 예에서는 모든 항목이 실행 중임을 보여줍니다.

```
$ oc get pods
NAME                    READY  STATUS    RESTARTS   AGE   IP
keycloak-0              1/1    Running   0          26m   192.0.2.0
keycloak-postgresql-c8vv27m 1/1    Running   0          36m   192.0.2.3
```

11.7. 데이터베이스 백업 예약

Operator를 사용하여 사용자 정의 리소스에서 정의한 대로 데이터베이스의 자동 백업을 예약할 수 있습니다. 사용자 정의 리소스는 백업 작업을 트리거하고 상태를 다시 보고합니다.

*Operator*를 사용하여 로컬 영구 볼륨에 대한 일회성 백업을 수행하는 백업 작업을 생성할 수 있습니다.

Backup 사용자 정의 리소스에 대한 **YAML** 파일의 예

```
apiVersion: keycloak.org/v1alpha1
kind: KeycloakBackup
metadata:
  name: test-backup
```

사전 요구 사항

- 이 사용자 정의 리소스에 대한 **YAML** 파일이 있습니다.
- **Red Hat Single Sign-On Operator**가 생성한 **PersistentVolumeClaim** 에만 예약할 **claimRef** 가 있는 **PersistentVolume** 이 있습니다.

절차

1. 백업 작업을 생성합니다. **oc create -f <backup_crname>.** 예를 들면 다음과 같습니다.

```
$ oc create -f one-time-backup.yaml
keycloak.keycloak.org/test-backup
```

Operator는 다음 이름 지정 스키마를 사용하여 **PersistentVolumeClaim** 을 생성합니다. **Keycloak-backup-<CR-name>.**

2. 볼륨 목록을 확인합니다.

```
$ oc get pvc
NAME                        STATUS   VOLUME
keycloak-backup-test-backup   Bound    pvc-e242-ew022d5-093q-3134n-41-adff
keycloak-postresql-claim      Bound    pvc-e242-vs29202-9bcd7-093q-31-zadj
```

3. 백업 작업 목록을 확인합니다.

```
$ oc get jobs
NAME          COMPLETIONS    DURATION    AGE
test-backup   0/1            6s          6s
```

4. 실행된 백업 작업 목록을 확인합니다.

```
$ oc get pods
NAME                          READY   STATUS      RESTARTS   AGE
test-backup-5b4rf             0/1     Completed   0          24s
keycloak-0                    1/1     Running     0          52m
keycloak-postgresql-c824c6-vv27m  1/1     Running     0          71m
```

5. 완료된 백업 작업의 로그를 확인합니다.

```
$ oc logs test-backup-5b4rf
==> Component data dump completed
.
.
.
.
```

추가 리소스

- 영구 볼륨에 대한 자세한 내용은 영구 스토리지 이해 를 참조하십시오.

11.8. 확장 및 테마 설치

Operator를 사용하여 회사 또는 조직에 필요한 확장 기능 및 테마를 설치할 수 있습니다. 확장 또는 주제는 **Red Hat Single Sign-On**에서 사용할 수 있는 모든 항목이 될 수 있습니다. 예를 들어 메트릭 확장을 추가할 수 있습니다. **Keycloak** 사용자 정의 리소스에 확장 또는 테마를 추가합니다.

Keycloak 사용자 정의 리소스의 **YAML** 파일의 예

```
apiVersion: keycloak.org/v1alpha1
kind: Keycloak
metadata:
  name: example-keycloak
  labels:
    app: sso
spec:
  instances: 1
  extensions:
```

```
 - <url_for_extension_or_theme>
externalAccess:
  enabled: True
```

사전 요구 사항

- *Keycloak 사용자 정의 리소스에 대한 YAML 파일이 있습니다.*
- *cluster-admin 권한 또는 관리자가 부여한 동등한 수준의 권한이 있어야 합니다.*

절차

1. *Keycloak 사용자 정의 리소스의 YAML 파일을 편집합니다. oc edit <CR-name>*
2. *instances 라인 뒤에 extensions: 행을 추가합니다.*
3. *사용자 정의 확장 또는 주제의 JAR 파일에 URL을 추가합니다.*
4. *파일을 저장합니다.*

Operator는 확장 또는 주제를 다운로드하여 설치합니다.

11.9. 사용자 정의 리소스 관리를 위한 명령 옵션

사용자 지정 요청을 생성한 후 oc 명령을 사용하여 편집하거나 삭제할 수 있습니다.

- *사용자 정의 요청을 편집하려면 다음 명령을 사용하십시오. oc edit <CR-name>*
- *사용자 정의 요청을 삭제하려면 다음 명령을 사용하십시오. oc delete <CR-name>*

*예를 들어 **test-realm** 이라는 **realm** 사용자 지정 요청을 편집하려면 다음 명령을 사용합니다.*

```
$ oc edit test-realm
```

변경할 수 있는 창이 열립니다.

참고

***YAML** 파일을 업데이트하고 변경 사항이 **Red Hat Single Sign-On** 관리 콘솔에 표시될 수 있지만 관리 콘솔의 변경 사항은 사용자 정의 리소스를 업데이트하지 않습니다.*