



# Red Hat Single Sign-On 7.4

## 服务器安装和配置指南

用于 Red Hat Single Sign-On 7.4



# Red Hat Single Sign-On 7.4 服务器安装和配置指南

---

用于 Red Hat Single Sign-On 7.4

## 法律通告

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## 摘要

本指南包含安装和配置 Red Hat Single Sign-On 7.4 的信息

# 目录

使开源包含更多 .....	4
<b>第 1 章 指南概述</b> .....	<b>5</b>
1.1. 推荐的额外外部文档 .....	5
<b>第 2 章 安装</b> .....	<b>6</b>
2.1. 系统要求 .....	6
2.2. 从 ZIP 文件安装 RH-SSO .....	6
2.3. 从 RPM 安装 RH-SSO .....	7
2.4. 分发目录结构 .....	8
<b>第 3 章 选择操作模式</b> .....	<b>10</b>
3.1. 独立模式 .....	10
3.2. 独立集群模式 .....	12
3.3. 域集群模式 .....	14
3.4. 跨数据中心复制模式 .....	22
<b>第 4 章 管理子系统配置</b> .....	<b>23</b>
4.1. 配置 SPI 供应商 .....	23
4.2. 启动 JBOSS EAP CLI .....	24
4.3. CLI 嵌入式模式 .....	25
4.4. CLI GUI 模式 .....	25
4.5. CLI 脚本 .....	26
4.6. CLI RECIPES .....	27
<b>第 5 章 配置集</b> .....	<b>29</b>
<b>第 6 章 关系的数据库设置</b> .....	<b>31</b>
6.1. RDBMS 设置清单 .....	31
6.2. 打包 JDBC 驱动程序 .....	32
6.3. 声明和加载 JDBC 驱动程序 .....	34
6.4. 修改 RED HAT SINGLE SIGN-ON 数据源 .....	35
6.5. 数据库配置 .....	36
6.6. 数据库的 UNICODE 注意事项 .....	38
<b>第 7 章 主机名</b> .....	<b>41</b>
7.1. 默认供应商 .....	41
7.2. 自定义供应商 .....	42
<b>第 8 章 网络设置</b> .....	<b>43</b>
8.1. 绑定地址 .....	43
8.2. 套接字端口绑定 .....	44
8.3. 设置 HTTPS/SSL .....	45
8.4. 传出 HTTP 请求 .....	49
<b>第 9 章 集群</b> .....	<b>55</b>
9.1. 推荐的网络架构 .....	55
9.2. 集群示例 .....	55
9.3. 设置负载均衡器或代理 .....	56
9.4. 粘性会话 .....	62
9.5. 多播网络设置 .....	63
9.6. 保护集群通信 .....	64
9.7. 序列化集群启动 .....	65
9.8. 引导集群 .....	65

9.9. 故障排除	66
<b>第 10 章 服务器缓存配置</b> .....	<b>67</b>
10.1. 驱除和过期	67
10.2. 复制和故障切换	68
10.3. 禁用缓存	69
10.4. 在运行时清除缓存	70
<b>第 11 章 RED HAT SINGLE SIGN-ON OPERATOR</b> .....	<b>71</b>
11.1. 在集群上安装 RED HAT SINGLE SIGN-ON OPERATOR	72
11.2. 使用自定义资源进行 RED HAT SINGLE SIGN-ON 安装	77
11.3. 创建 REALM 自定义资源	83
11.4. 创建客户端自定义资源	86
11.5. 创建用户自定义资源	89
11.6. 连接到外部数据库	92
11.7. 调度数据库备份	95
11.8. 安装扩展和主题	97
11.9. 管理自定义资源的命令选项	98



## 使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅[我们的首席技术官 Chris Wright 提供的消息](#)。



## 第 1 章 指南概述

本指南的目的是完成第一次引导 Red Hat Single Sign-On 服务器前需要完成的步骤。如果您只想测试驱动器 Red Hat Single Sign-On，它只会耗尽其自身的嵌入式和本地数据库。对于要在生产中运行的实际部署，您需要确定如何在运行时管理服务器配置(standalone 或 domain mode)，为 Red Hat Single Sign-On 存储配置共享数据库，设置加密和 HTTPS，最后设置 Red Hat Single Sign-On 在集群中运行。本指南介绍了任何预引导决策和设置的各个方面，您必须在部署服务器之前完成。

特别注意的是 Red Hat Single Sign-On 派生自 JBoss EAP Application Server。配置 Red Hat Single Sign-On 的许多方面会围绕 JBoss EAP 配置元素而解决。通常，如果想要详细介绍，本指南会将您定向到手册之外的文档。

### 1.1. 推荐的额外外部文档

Red Hat Single Sign-On 基于 JBoss EAP 应用服务器及其子项目（如 Infinispan（用于缓存）和 Hibernate（用于持久性）。本指南仅涵盖基础架构级配置的基础知识。强烈建议您使用 JBoss EAP 及其子项目的文档。以下是文档的链接：

- [JBoss EAP 配置指南](#)

## 第 2 章 安装

您可以通过下载 ZIP 文件和解压缩或使用 RPM 来安装 Red Hat Single Sign-On。本章回顾了系统要求以及目录结构。

### 2.1. 系统要求

以下是运行 Red Hat Single Sign-On 身份验证服务器的要求：

- 可以在运行 Java 的任何操作系统上运行
- Java 8 JDK
- zip 或 gzip 和 tar
- 至少 512M RAM
- 至少 1G 的磁盘空间
- 共享外部数据库，如 PostgreSQL、MySQL、Oracle 等。如果要在集群中运行，Red Hat Single Sign-On 需要外部共享数据库。如需更多信息，请参阅本指南的 [数据库配置](#) 部分。
- 如果要在集群中运行，在您的机器上支持网络多播。Red Hat Single Sign-On 可以在没有多播的情况下进行集群，但这需要大量配置更改。如需更多信息，请参阅本指南的集群部分。
- 在 Linux 上，建议使用 `/dev/urandom` 作为随机数据源，以防止因为缺少可用的熵而导致 Red Hat Single Sign-On 挂起，除非安全策略强制使用 `/dev/random`。要在 Oracle JDK 8 和 OpenJDK 8 上达到此目的，请在启动时将 `java.security.egd` 系统属性设置为 `file:/dev/urandom`。

### 2.2. 从 ZIP 文件安装 RH-SSO

Red Hat Single Sign-On 服务器下载 ZIP 文件包含运行 Red Hat Single Sign-On 服务器的脚本和二进制文件。您首先安装 7.4.0.GA 服务器，然后安装 7.4.10.GA 服务器补丁。

#### 流程

1. 访问红帽客户门户。<https://access.redhat.com/jbossnetwork/restricted/listSoftware.html?downloadType=distributions&product=core.service.rhssso>
2. 下载 Red Hat Single Sign-On 7.4.0.GA 服务器。
3. 使用适当的 **unzip** 实用程序（如 unzip 或 Expand-Archive）解包 ZIP 文件。
4. 返回到 [红帽客户门户](#)。
5. 点 **Patches** 选项卡。
6. 下载 Red Hat Single Sign-On 7.4.10.GA 服务器补丁。
7. 将下载的文件放在您选择的目录中。
8. 前往 JBoss EAP 的 **bin** 目录。
9. 启动 JBoss EAP 命令行界面。

## Linux/Unix

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

## Windows

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

10. 应用补丁。

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

## 其他资源

有关应用补丁的详情，请参阅 [对 ZIP/Installer 安装打补丁](#)。

## 2.3. 从 RPM 安装 RH-SSO



### 注意

在 Red Hat Enterprise Linux 7 和 8 中，术语频道被术语仓库替代。在这些说明中，只使用存储库术语。

您必须订阅 JBoss EAP 7.3 和 RH-SSO 7.4 存储库，然后才能从 RPM 安装 RH-SSO。



### 注意

您无法继续接收到 EAP RPM 的升级，但停止接收 RH-SSO 的更新。

### 2.3.1. 订阅 JBoss EAP 7.3 存储库

#### 先决条件

1. 使用 Red Hat Subscription Manager 验证您的 Red Hat Enterprise Linux 系统已注册到您的帐户。如需更多信息，请参阅 [Red Hat Subscription Management 文档](#)。
2. 如果您已经订阅了另一个 JBoss EAP 存储库，您必须首先取消订阅该存储库。

对于 Red Hat Enterprise Linux 6,7：使用 Red Hat Subscription Manager，使用以下命令订阅 JBoss EAP 7.3 存储库。根据您的 Red Hat Enterprise Linux 版本，将 <RHEL\_VERSION> 替换为 6 或 7。

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

对于 Red Hat Enterprise Linux 8：使用 Red Hat Subscription Manager，使用以下命令订阅 JBoss EAP 7.3 存储库：

```
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. 订阅 RH-SSO 7.4 存储库并安装 RH-SSO 7.4

## 先决条件

1. 使用 Red Hat Subscription Manager 验证您的 Red Hat Enterprise Linux 系统已注册到您的帐户。如需更多信息，请参阅 [Red Hat Subscription Management 文档](#)。
2. 确保您已订阅了 JBoss EAP 7.3 存储库。如需更多信息，请参阅 [订阅 JBoss EAP 7.3 存储库](#)。

要订阅 RH-SSO 7.4 存储库并安装 RH-SSO 7.4，请完成以下步骤：

1. 对于 Red Hat Enterprise Linux 6, 7：使用 Red Hat Subscription Manager，使用以下命令订阅 RH-SSO 7.4 存储库。根据您的 Red Hat Enterprise Linux 版本，将 <RHEL\_VERSION> 替换为 6 或 7。

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

2. 对于 Red Hat Enterprise Linux 8：使用 Red Hat Subscription Manager，使用以下命令订阅 RH-SSO 7.4 存储库：

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

3. 对于 Red Hat Enterprise Linux 6,7：使用以下命令从您订阅的 RH-SSO 7.4 存储库安装 RH-SSO：

```
yum groupinstall rh-ss-7
```

4. 对于 Red Hat Enterprise Linux 8：使用以下命令从您订阅的 RH-SSO 7.4 存储库安装 RH-SSO：

```
dnf groupinstall rh-ss-7
```

您的安装已完成。RPM 安装的默认 RH-SSO\_HOME 路径为 /opt/rh/rh-ss-7/root/usr/share/keycloak。

## 其他资源

有关为 Red Hat Single Sign-On 安装 7.4.10.GA 补丁的详情，请参考 [RPM 补丁](#)。

## 2.4. 分发目录结构

本章介绍了服务器分发的目录结构。

我们来检查某些目录的目的：

### *bin/*

它包含引导服务器或在服务器上执行某些其他管理操作的各种脚本。

### *domain/*

它包含在 [域模式下运行](#) Red Hat Single Sign-On 时的配置文件和工作目录。

### *modules/*

这些是服务器使用的所有 Java 库。

### *standalone/*

它包含 [以独立模式运行](#) Red Hat Single Sign-On 时的配置文件和工作目录。

### *standalone/deployments/*

如果您要向 Red Hat Single Sign-On 编写扩展，您可以在这里放置扩展。有关此 [内容的更多信息](#)，请参阅 [服务器开发人员指南](#)。

---

### *themes/*

此目录包含所有 html、样式表、JavaScript 文件和映像，用于显示服务器显示的任何 UI 屏幕。您可以在[此处](#)修改现有主题或创建自己的主题。有关此[内容的更多信息](#)，请[参阅服务器开发人员指南](#)。

## 第 3 章 选择操作模式

在生产环境中部署 Red Hat Single Sign-On 之前，您需要决定您要使用的工作模式。是否在集群中运行 Red Hat Single Sign-On？您是否想要通过集中的方式管理服务器配置？您选择的操作模式会影响您配置数据库、配置缓存甚至如何引导服务器。

### 提示

Red Hat Single Sign-On 基于 JBoss EAP Application Server 构建。本指南仅了解在特定模式中部署的基础知识。如果您需要有关此特定信息，最好是 [JBoss EAP 配置指南](#)。

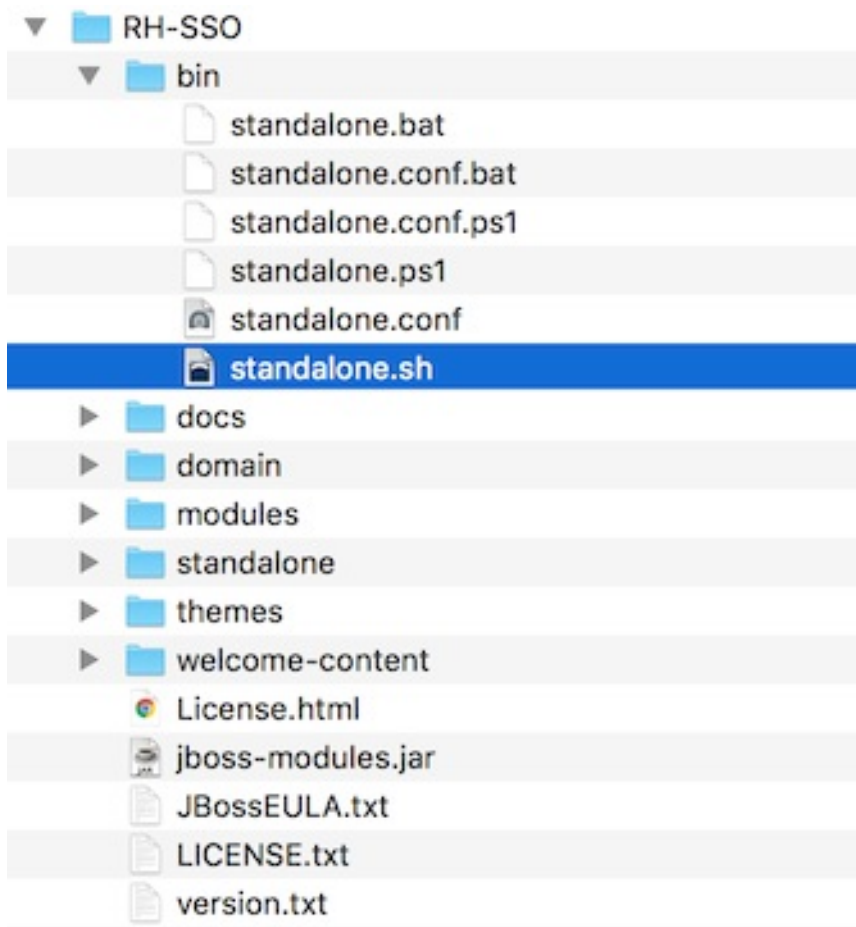
### 3.1. 独立模式

单机操作模式仅在您要运行一个且只有一个 Red Hat Single Sign-On 服务器实例时才有用。它不适用于集群部署，所有缓存都是非分布式且本地的。不建议在生产环境中使用独立模式，因为您将存在单点故障。如果您的单机模式服务器停机，用户将无法登录。此模式对测试驱动器和 Red Hat Single Sign-On 的功能有用

#### 3.1.1. 独立启动脚本

在独立模式中运行服务器时，您需要运行特定的脚本来根据您的操作系统引导服务器。这些脚本位于服务器分发的 `bin/` 目录中。

#### 独立启动脚本



引导服务器：

Linux/Unix

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

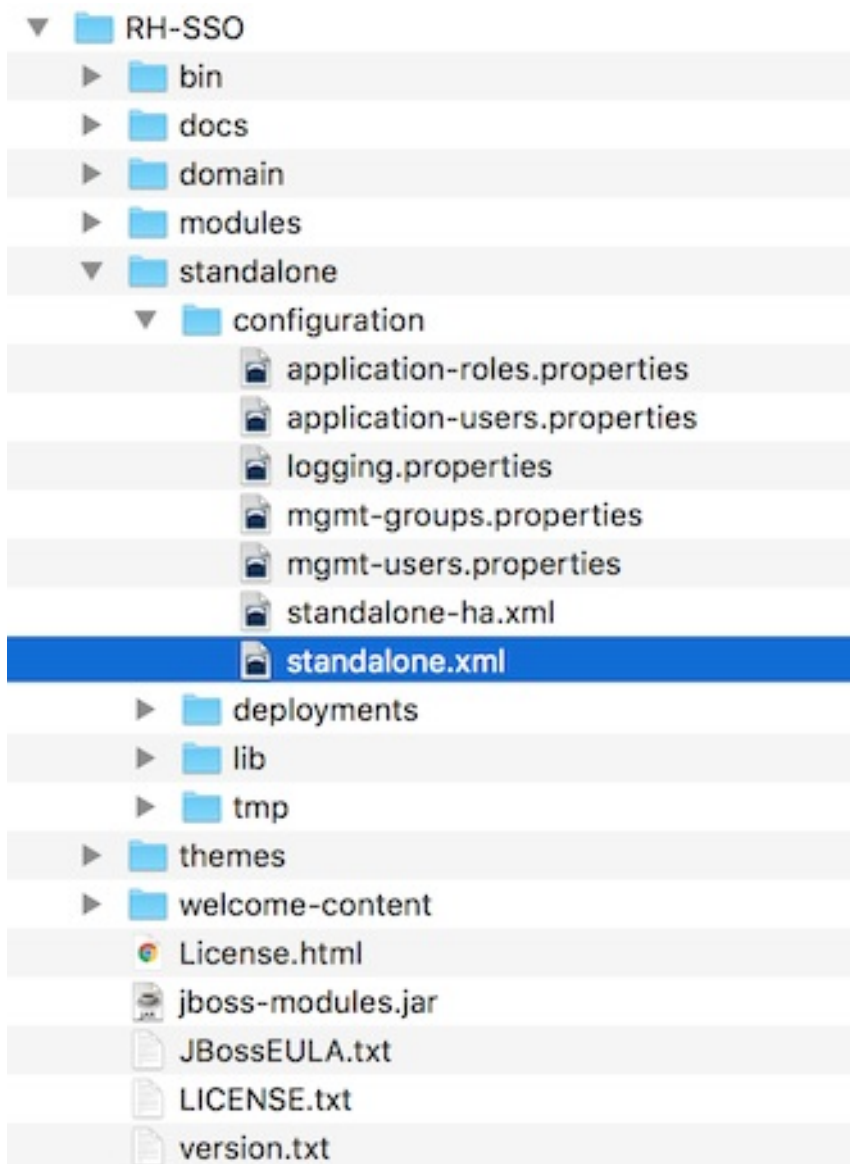
## Windows

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

### 3.1.2. 独立配置

本指南批量指导您如何配置 Red Hat Single Sign-On 的基础架构级别方面。这些方面是在特定于 Red Hat Single Sign-On 的一个应用程序服务器的配置文件中配置。在独立操作模式中，此文件在 `.../standalone/configuration/standalone.xml` 中。此文件还用于配置特定于 Red Hat Single Sign-On 组件的非基础架构级别操作。

#### 独立配置文件





### 警告

在服务器运行时，您对此文件所做的任何更改都不会生效，甚至可能被服务器覆盖。而是使用命令行脚本或 JBoss EAP 的 Web 控制台。如需更多信息，请参阅 [JBoss EAP 配置指南](#)。

## 3.2. 独立集群模式

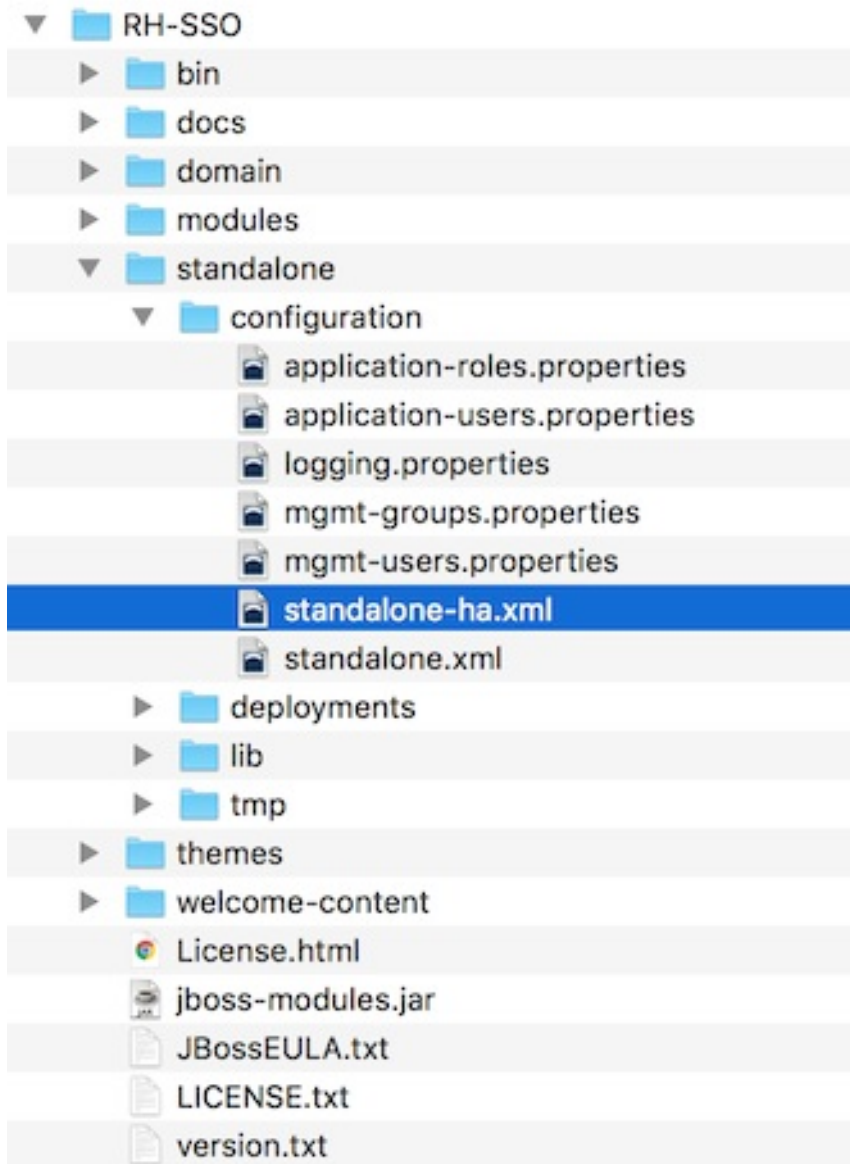
独立集群操作模式用于在集群中运行 Red Hat Single Sign-On。这个模式要求您在您要运行服务器实例的每台机器上有一个 Red Hat Single Sign-On 分发的副本。最初部署此模式可能非常容易，但可能会变得非常繁琐。要进行配置更改，您必须修改每台机器上的每个发行版。对于大型集群，这可能会变得耗时且容易出错。

### 3.2.1. 独立集群配置

发行版主要有一个预配置的应用服务器配置文件，可在集群内运行。它具有所有特定的基础架构设置，用于网络、数据库、缓存和发现。此文件位于 `.../standalone/configuration/standalone-ha.xml` 中。此配置中缺少一些内容。您无法在不配置共享数据库连接的情况下在集群中运行 Red Hat Single Sign-On。您还需要在集群前面部署某种类型的负载均衡器。本指南的 [集群](#) 和 [数据库](#) 部分将指导您完成这些事务。

### 独立 HA 配置





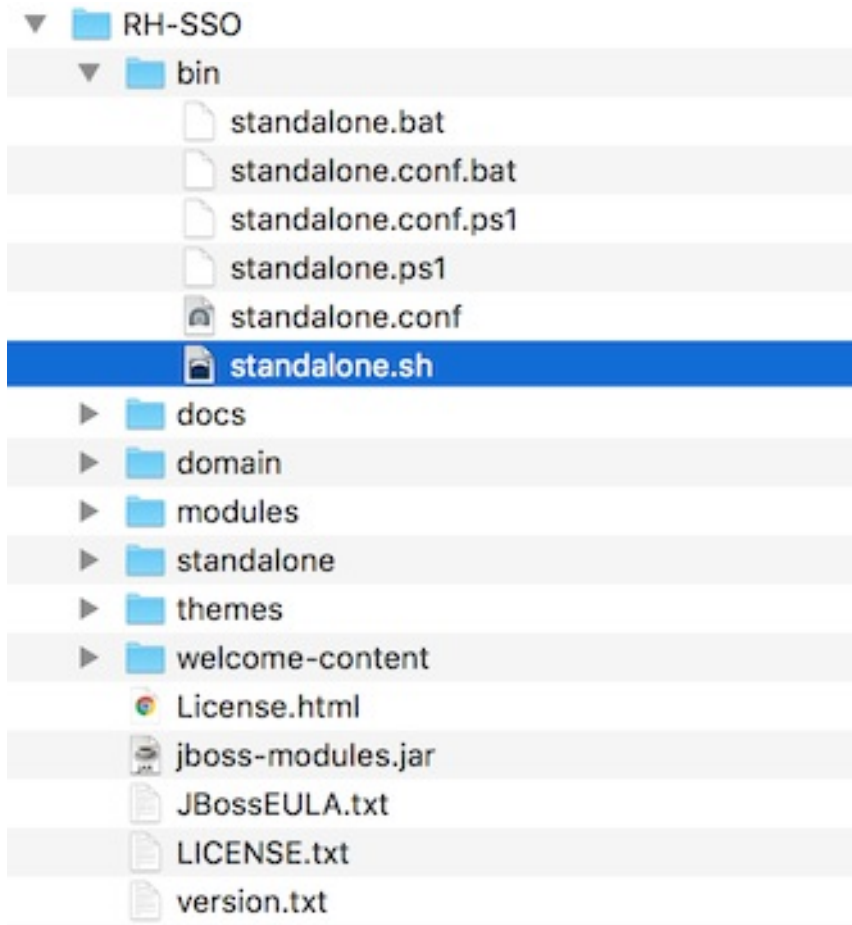
### 警告

在服务器运行时，您对此文件所做的任何更改都不会生效，甚至可能被服务器覆盖。而是使用命令行脚本或 JBoss EAP 的 Web 控制台。如需更多信息，请参阅 [JBoss EAP 配置指南](#)。

## 3.2.2. 独立集群启动脚本

您可以使用相同的引导脚本来启动 Red Hat Single Sign-On，如独立模式下操作。区别在于，您传递了一个额外的标记以指向 HA 配置文件。

### 独立集群启动脚本



引导服务器：

### Linux/Unix

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

### Windows

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

## 3.3. 域集群模式

域模式是集中管理和发布服务器的配置的一种方式。

以标准模式运行集群可能会快速变为聚合，因为集群大小会增大。每次需要更改配置更改时，都必须在集群的每个节点中执行它。域模式通过提供存储和发布配置的中心来解决此问题。设置可能非常复杂，但在最后需要它。此功能内置在 Red Hat Single Sign-On 派生的 JBoss EAP 应用服务器中。



### 注意

本指南将介绍域模式的基本知识。应从 [JBoss EAP 配置指南](#) 中获取集群中如何设置域模式的详细步骤。

以下是在域模式下运行的一些基本概念。

### 域控制器

域控制器是一种负责存储、管理和发布集群中每个节点的常规配置的进程。此过程是从哪些节点获取其配置的核心点。

### 主机控制器

主机控制器负责管理特定计算机上的服务器实例。您可以将其配置为运行一个或多个服务器实例。域控制器也可以与每台机器上的主机控制器交互，以管理集群。为减少正在运行的进程的数量，域控制器在其上运行的计算机上也充当主机控制器。

### 域配置文件

域配置文件是一组指定的配置，供服务器用于从中启动。域控制器可以定义由不同服务器使用的多个域配置文件。

### 服务器组

服务器组是服务器的集合。它们作为一个进行管理和配置。您可以为服务器组分配域配置文件，并且该组中的每个服务都将使用该域配置文件作为其配置。

在域模式中，主控节点上启动域控制器。集群的配置驻留在域控制器中。接下来，在集群中的每个机器上启动主机控制器。每个主机控制器部署配置指定在该计算机上启动多少个 Red Hat Single Sign-On 服务器实例。当主机控制器启动时，它会启动尽可能多的 Red Hat Single Sign-On 服务器实例。这些服务器实例从域控制器拉取其配置。



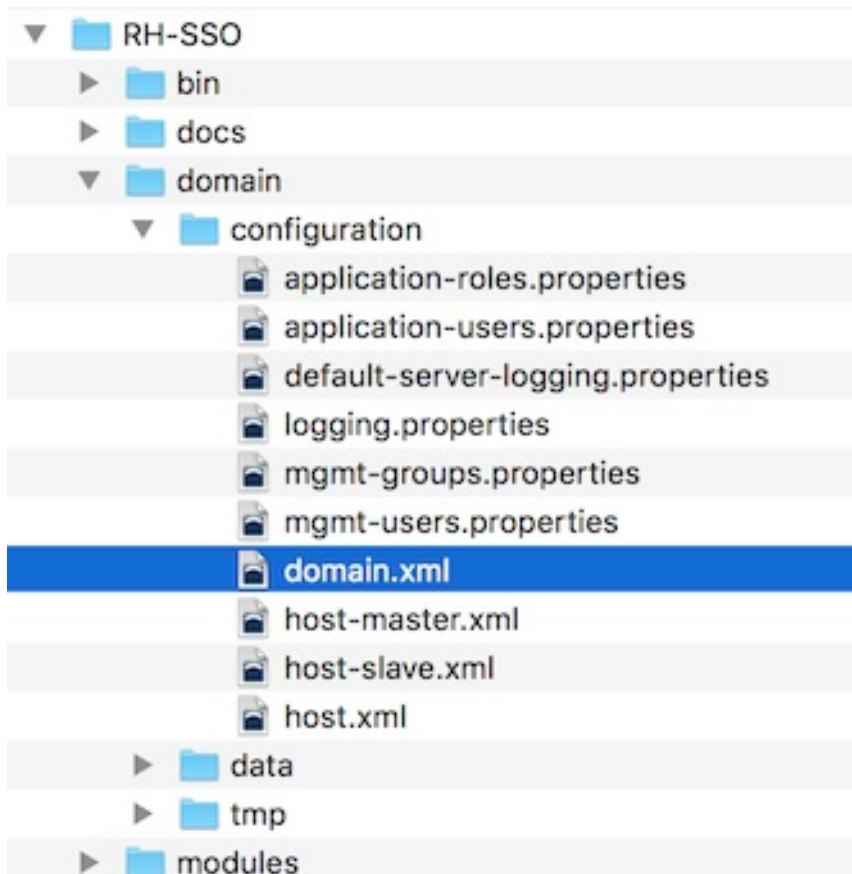
#### 注意

在一些环境中，如 Microsoft Azure，域模式不适用。请参阅 JBoss EAP 文档。

### 3.3.1. 域配置

本指南中的各种其他章节将指导您配置数据库、HTTP 网络连接、缓存和其他基础架构相关事项等各个方面。虽然独立模式使用 *standalone.xml* 文件来配置这些内容，但域模式使用 ...  
*/domain/configuration/domain.xml* 配置文件。这是定义 Red Hat Single Sign-On 服务器的域配置文件和服务器组的位置。

#### domain.xml



### 警告

您在域控制器运行时对此文件所做的任何更改都不会生效，甚至可能被服务器覆盖。而是使用命令行脚本或 JBoss EAP 的 Web 控制台。如需更多信息，请参阅 [JBoss EAP 配置指南](#)。

让我们来看看这个 `domain.xml` 文件的一些方面。**auth-server-standalone** 和 **auth-server-clustered** 配置集 XML 块是您要做出大量配置决策的位置。您要在此处配置诸如网络连接、缓存和数据库连接等内容。

### auth-server 配置集

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

**auth-server-standalone** 配置集是一个非集群的设置。**auth-server-clustered** 配置集是集群设置。

如果您进一步向下滚动，您将看到定义了各种 **socket-binding-groups**。

### 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>

```

此配置定义了与每个 Red Hat Single Sign-On 服务器实例一起打开的各种连接器的默认端口映射。包含 `#{...}` 的任何值都是可在命令行中使用 `-D` 开关覆盖的值，即。

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

Red Hat Single Sign-On 的服务器组的定义位于 **server-groups** XML 块中。它指定使用的域配置文件(默认)，以及主机控制器引导实例时 Java 虚拟机的一些默认引导参数。它还将 **socket-binding-group** 绑定到服务器组。

## 服务器组

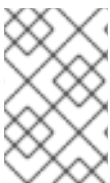
```

<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. 主机控制器配置

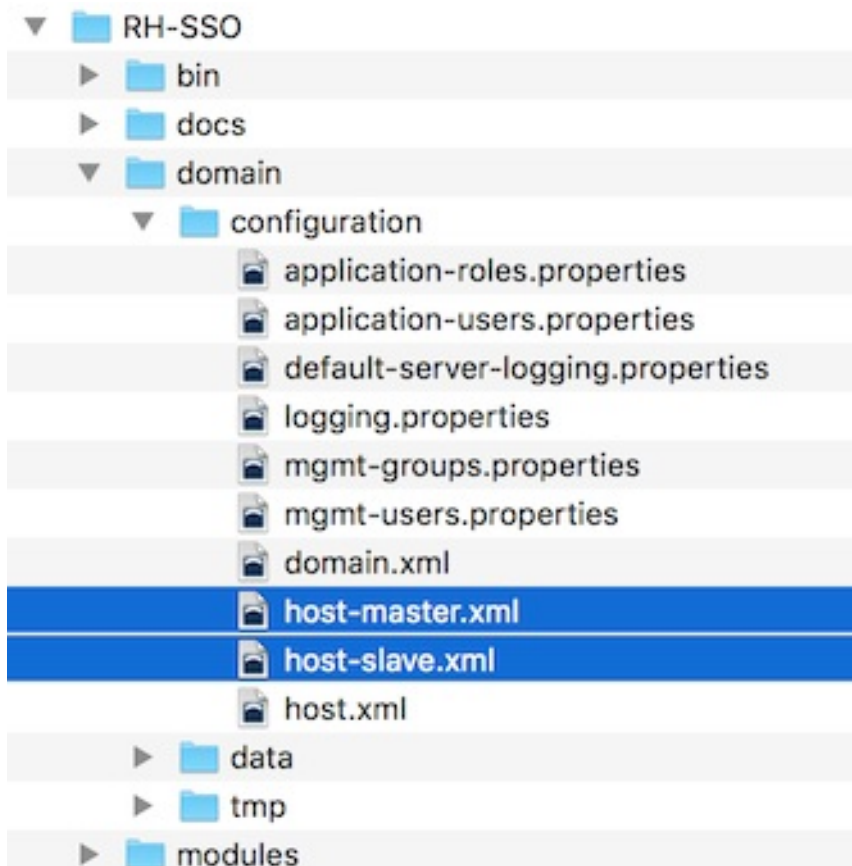
Red Hat Single Sign-On 附带两个主机控制器配置文件，它们位于 `.../domain/configuration/` 目录中：`host-master.xml` 和 `host-slave.xml`。`host-master.xml` 配置为引导域控制器、负载均衡器和一个 Red Hat Single Sign-On 服务器实例。`host-slave.xml` 配置为与域控制器通信并引导一个 Red Hat Single Sign-On 服务器实例。



#### 注意

负载均衡器不是所需的服务。它已存在，以便您可以在开发机器上轻松测试驱动集群。虽然在生产环境中可用，但如果您有您要使用的不同硬件或软件负载均衡器，您可以选择替换它。

## 主机控制器配置



要禁用负载均衡器服务器实例，请编辑 `host-master.xml` 并注释掉或删除 **"load-balancer"** 条目。

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

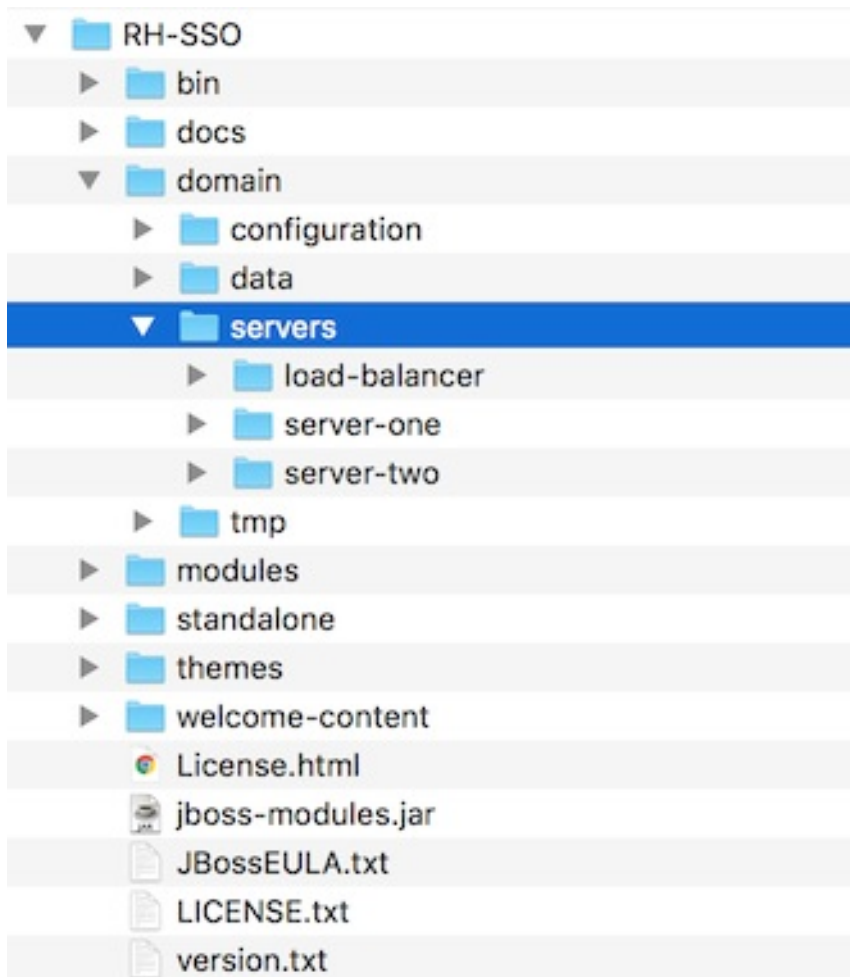
关于此文件的另一个值得注意的是，身份验证服务器实例的声明。它有一个 **port-offset** 设置。 `domain.xml` **socket-binding-group** 或服务器组中定义的任何网络端口都会向其中添加 **port-offset** 值。在这个示例域设置中，我们这样做，因此负载均衡器服务器打开的端口不会与启动的身份验证服务器实例冲突。

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

### 3.3.3. 服务器实例工作目录

主机文件中定义的每个 Red Hat Single Sign-On 服务器实例都会在 `.../domain/servers/{SERVER NAME}` 下创建一个工作目录。可以有一些其他配置，以及服务器实例需要或创建任何临时、日志或数据文件。这些服务器目录的结构最终像任何其他 JBoss EAP 引导服务器一样。

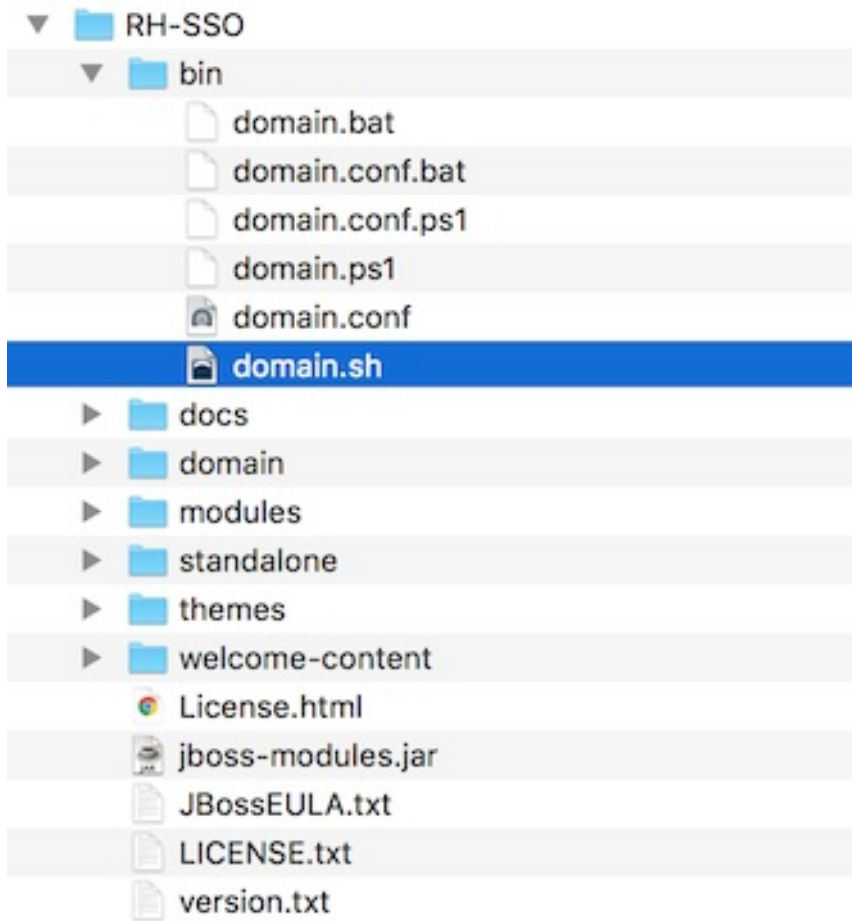
#### 工作目录



### 3.3.4. 域启动脚本

在域模式下运行服务器时，您需要运行特定的脚本来根据您的操作系统引导服务器。这些脚本位于服务器分发的 *bin/* 目录中。

#### 域启动脚本



引导服务器：

### Linux/Unix

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

### Windows

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

运行引导脚本时，您需要传递要通过 **--host-config** 参数使用的主机控制配置文件。

## 3.3.5. 集群域示例

您可以使用开箱即用的 *domain.xml* 配置测试驱动器集群。这个示例域旨在在一台机器上运行并引导：

- 域控制器
- HTTP 负载均衡器
- 2 Red Hat Single Sign-On 服务器实例

要在两台机器上模拟运行集群，您需要运行 **domain.sh** 脚本两次以启动两个单独的主机控制器。第一个是 master 主机控制器，它将启动域控制器、HTTP 负载均衡器和一个 Red Hat Single Sign-On 身份验证服务器实例。第二个是从属主机控制器，仅启动身份验证服务器实例。

### 3.3.5.1. 设置到域控制器的连接



在引导前，您必须配置 slave 主机控制器，以便它可以安全地与域控制器通信。如果不执行此操作，则从主机将无法从域控制器获取集中式配置。要设置安全连接，您必须创建一个服务器 admin 用户和在主设备和从设备之间共享的机密。您可以通过运行 `.../bin/add-user.sh` 脚本来完成此操作。

当您运行脚本时，选择 **Management User** 并回答 **yes**，当它询问新用户是否将用于一个 AS 进程连接到另一个进程时，请回答 **yes**。这将生成一个 secret，您需要剪切并粘贴到 `.../domain/configuration/host-slave.xml` 文件中。

## 添加 App Server Admin

```
$ 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=" />
```



### 注意

add-user.sh 不将用户添加到 Red Hat Single Sign-On 服务器，而是添加到底层 JBoss Enterprise Application Platform 中。以上脚本中使用和生成的凭证仅用于示例目的。请使用系统上生成的。

接下来，将 secret 值剪切并粘贴到 `.../domain/configuration/host-slave.xml` 文件中，如下所示：

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

您还需要在 `.../domain/configuration/host-slave.xml` 文件中添加创建用户的用户名：

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

### 3.3.5.2. 运行启动脚本

由于我们在一台开发机器上模拟两个节点集群，因此您将运行启动脚本两次：

引导 master

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

引导从设备

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

要试用，请打开浏览器并访问 <http://localhost:8080/auth>。

## 3.4. 跨数据中心复制模式

在 Red Hat Single Sign-On 7.2 中作为技术预览功能引进的跨站点复制，在任何 Red Hat SSO 7.x 版本中不再作为支持的功能提供，包括最新的 RH-SSO 7.6 版本。红帽不推荐在自己的环境中实施或使用此功能，因为它不被支持。另外，这个功能的支持例外不再被视为或接受。

讨论了跨站点复制的新解决方案，并特别考虑红帽构建的 Keycloak (RHBK) 的未来发行版本，这是将要引入的产品，而不是 Red Hat SSO 8。很快将提供更多详细信息。

## 第 4 章 管理子系统配置

Red Hat Single Sign-On 的低级别配置是通过编辑您发行版中的 **standalone.xml**、**standalone-ha.xml** 或 **domain.xml** 文件来完成的。此文件的位置取决于您的 [操作模式](#)。

虽然您可以在此处配置无限设置，但本节将专注于 keycloak-server 子系统的配置。无论您使用哪个配置文件，keycloak-server 子系统的配置都相同。

keycloak-server 子系统通常声明在文件末尾，如下所示：

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

请注意，此子系统中的任何更改都不会生效，直到服务器重启为止。

### 4.1. 配置 SPI 供应商

每个配置设置的具体内容会在其他上下文中通过此设置进行讨论。但是，了解用于声明 SPI 提供程序上的设置的格式会很有用。

Red Hat Single Sign-On 是一个高度模块化的系统，具有很大的灵活性。有 50 多个服务提供商接口 (SPI)，您可以交换每个 SPI 的实现。SPI 的实现称为 [提供程序](#)。

SPI 声明中的所有元素都是可选的，但完整的 SPI 声明如下所示：

```
<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>
```

在这里，我们为 SPI 定义了两个 [供应商](#)。**default-provider** 列为 **myprovider**。但是，最多是 SPI 来确定它如何处理此设置。有些 SPI 允许多个供应商，有些供应商不允许。因此 **default-provider** 可以帮助 SPI 选择。

另请注意，每个提供程序定义了自己的一组配置属性。以上两个提供程序都有一个名为 **foo** 的属性，只是一个共同的。

每个属性值的类型由供应商解释。但是，有一个例外。考虑 **eventsStore** SPI 的 **jpa** 供应商：

```
<spi name="eventsStore">
  <provider name="jpa" enabled="true">
    <properties>
      <property name="exclude-events" value="['EVENT1']>
```

```
                &quot;EVENT2&quot;]"/>
    </properties>
</provider>
</spi>
```

我们看到该值以方括号开头和结束。这意味着该值将作为列表传递给提供程序。在本例中，系统将提供程序传递有两个元素值 EVENT1 和 EVENT2 的列表。要向列表添加更多值，请仅使用逗号分隔每个 list 元素。不幸的是，您需要使用 `"` 来转义各个列表元素周围的引号。

## 4.2. 启动 JBOSS EAP CLI

除了手动编辑配置外，您还可以选择通过 `jboss-cli` 工具发出命令来更改配置。CLI 允许您在本地或远程配置服务器。和脚本相结合时，它特别有用。

要启动 JBoss EAP CLI，您需要运行 `jboss-cli`。

### Linux/Unix

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

### Windows

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

这将使您进入类似如下的提示：

### 提示

```
[disconnected /]
```

如果要在正在运行的服务器上执行命令，您首先要执行 `connect` 命令。

## 连接

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

您可以考虑自己，"我不在任何用户名或密码中输入！"如果您在与正在运行的独立服务器或域控制器相同的计算机上运行 `jboss-cli`，并且您的帐户有适当的文件权限，则不必设置或输入管理员用户名和密码。如需了解如何使事情更加安全（如果使用该设置），请参阅 [JBoss EAP 配置指南](#)。

### 4.3. CLI 嵌入式模式

如果这样做与单机服务器位于同一个机器上，并且您希望在服务器未激活时发出命令，您可以将服务器嵌入到 CLI 中，并在禁止传入的请求的特殊模式下进行更改。要做到这一点，首先使用您要更改的配置文件执行 `embed-server` 命令。

#### `embed-server`

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

### 4.4. CLI GUI 模式

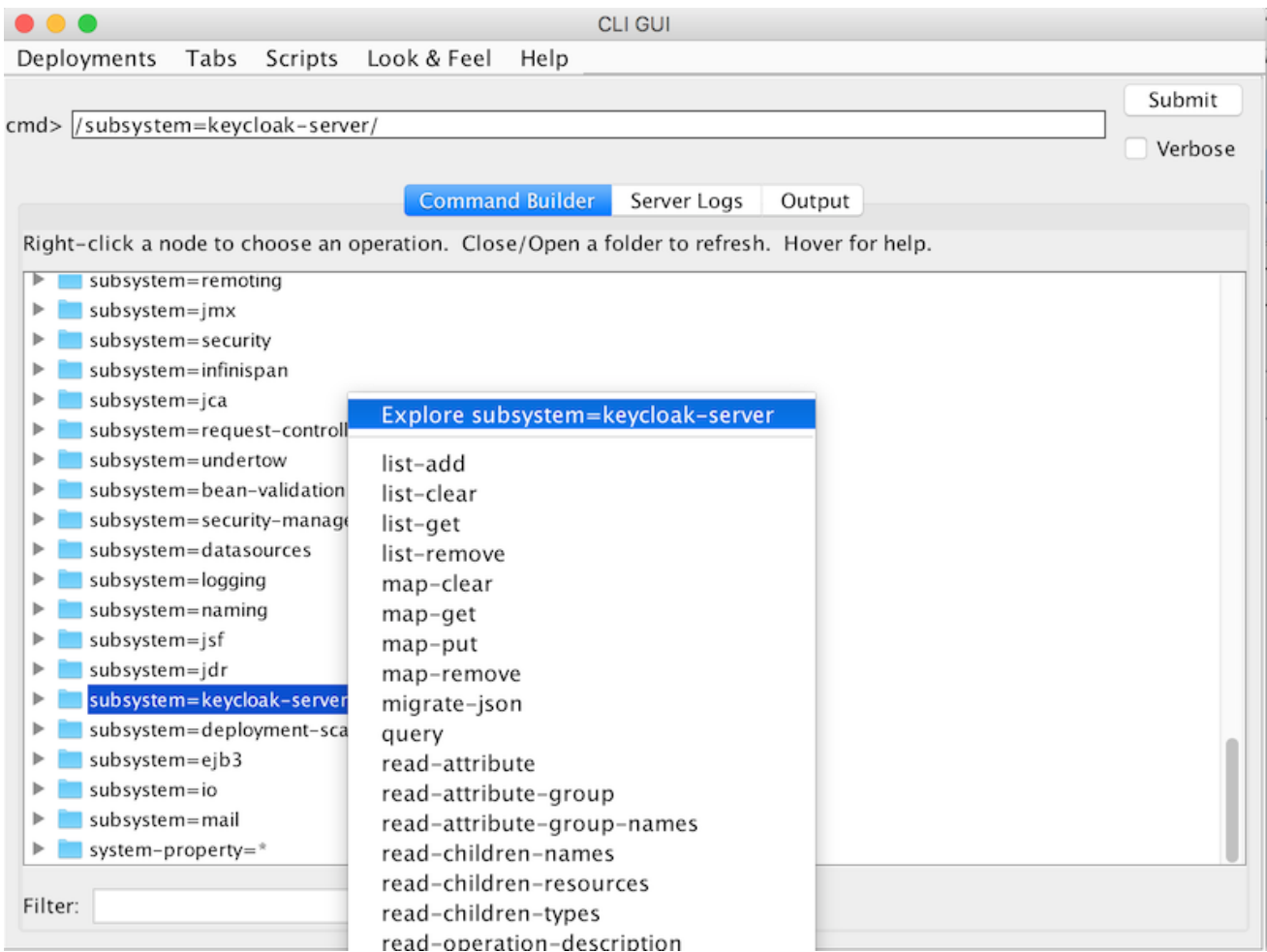
CLI 也可以以 GUI 模式运行。GUI 模式启动 `Swing` 应用程序，允许您以图形方式查看和编辑正在运行的服务器的整个管理模型。当您需要帮助格式化 CLI 命令并了解可用选项时，GUI 模式特别有用。GUI 还可以从本地或远程服务器检索服务器日志。

## 在 GUI 模式中启动

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

**注意：**要连接到远程服务器，您也要传递 `--connect` 选项。详情请查看 `--help` 选项。

启动 GUI 模式后，您可能想要向下滚动以查找节点 `subsystem=keycloak-server`。如果右键单击节点，然后单击 `Explore subsystem=keycloak-server`，您将获得一个新标签页，其中仅显示 `keycloak-server` 子系统。



## 4.5. CLI 脚本

CLI 具有广泛的脚本功能。脚本只是一个文本文件，其中带有 CLI 命令。考虑关闭主题和模板缓存的简单脚本。

### 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)
```

要执行脚本，我可以遵循 CLI GUI 中的 **Scripts** 菜单，或者在命令行中执行脚本，如下所示：

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

## 4.6. CLI RECIPES

以下是一些配置任务，以及如何使用 CLI 命令执行这些任务。请注意，在除第一个示例中，我们使用通配符路径 **\*\*** 表示您应该替换或到 **keycloak-server** 子系统的路径。

对于独立，这只是：

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

对于域模式，这意味着如下：

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

### 4.6.1. 更改服务器的 Web 上下文

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

### 4.6.2. 设置全局默认主题

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

#### 4.6.3. 添加新 SPI 和供应商

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

#### 4.6.4. 禁用供应商

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

#### 4.6.5. 更改 SPI 的默认供应商

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

#### 4.6.6. 配置 dblock SPI

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

#### 4.6.7. 为供应商添加或更改单个属性值

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

#### 4.6.8. 从供应商中删除单个属性

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

#### 4.6.9. 对类型 List 的 provider 属性设置值

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



## 第 5 章 配置集

**Red Hat Single Sign-On 中默认没有启用的功能，其中包括不完全支持的功能。此外，还有一些默认启用的功能，但可以禁用。**

可以启用和禁用的功能有：

Name	描述	默认启用	支持级别
account2	新帐户管理控制台	否	预览
account_api	帐户管理 REST API	否	预览
admin_fine_grained_authz	精细的管理权限	否	预览
docker	Docker Registry 协议	否	支持
模拟	管理员能够模拟用户	是	支持
openshift_integration	扩展以启用 OpenShift 保护	否	预览
scripts	使用 JavaScript 编写自定义验证器	否	预览
token_exchange	令牌交换服务	否	预览
upload_scripts	通过 Red Hat Single Sign-On REST API 上传脚本	否	已弃用
web_authn	W3C Web 身份验证 (WebAuthn)	否	预览

启用所有预览功能，请使用以下内容启动服务器：

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

您可以通过在域模式中为 **server-one** 创建文件 **standalone/configuration/profile.properties**（或 **domain/servers/server-one/configuration/profile.properties**）来永久设置。在文件中添加以下内容：

```
profile=preview
```

要启用特定的功能，请使用以下内容启动服务器：

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

例如，启用 Docker 使用 **-Dkeycloak.profile.feature.docker=enabled**。

您可以通过添加以下内容在 `profile.properties` 文件中永久设置：

```
feature.docker=enabled
```

要禁用特定的功能，请使用以下内容启动服务器：

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

例如，禁用 Impersonation use 使用 **-Dkeycloak.profile.feature.impersonation=disabled**。

您可以通过添加以下内容在 `profile.properties` 文件中永久设置：

```
feature.impersonation=disabled
```

## 第 6 章 关系的数据库设置

**Red Hat Single Sign-On 附带自己的嵌入式基于 Java 的关系数据库，称为 H2。这是 Red Hat Single Sign-On 用来持久保留数据的默认数据库，实际上仅存在，以便您可以开箱即用地运行身份验证服务器。我们强烈建议您将其替换为更多生产就绪的外部数据库。在高并发情况下，H2 数据库无法非常可行，且不能在集群中使用。本章的目的是向您展示如何将 Red Hat Single Sign-On 连接到更成熟的数据库。**

**Red Hat Single Sign-On 使用两种层次技术来持久保留其关系数据。底层技术是 JDBC。JDBC 是一个 Java API，用于连接 RDBMS。每个数据库类型都有不同的 JDBC 驱动程序，这些驱动程序由您的数据库供应商提供。本章讨论了如何配置 Red Hat Single Sign-On 以使用这些特定供应商的驱动程序之一。**

**用于持久性的顶级技术是 Hibernate JPA。这是关系映射 API 的对象，用于将 Java 对象映射到关系数据。Red Hat Single Sign-On 的大多数部署都不会涉及 Hibernate 的配置方面，但我们将讨论当您遇到的意外情况时如何完成。**



**注意**

**在 JBoss EAP 配置指南中的 [数据源配置一章](#)中更加全面地介绍数据源配置。**

### 6.1. RDBMS 设置清单

以下是获取为 Red Hat Single Sign-On 配置 RDBMS 所需的步骤。

1. **为您的数据库找到并下载 JDBC 驱动程序**
2. **将驱动程序 JAR 打包成模块，并将此模块安装到服务器中**
3. **在服务器的配置中声明 JDBC 驱动程序**
4. **修改数据源配置以使用数据库的 JDBC 驱动程序**
5. **修改数据源配置，以定义与数据库的连接参数**

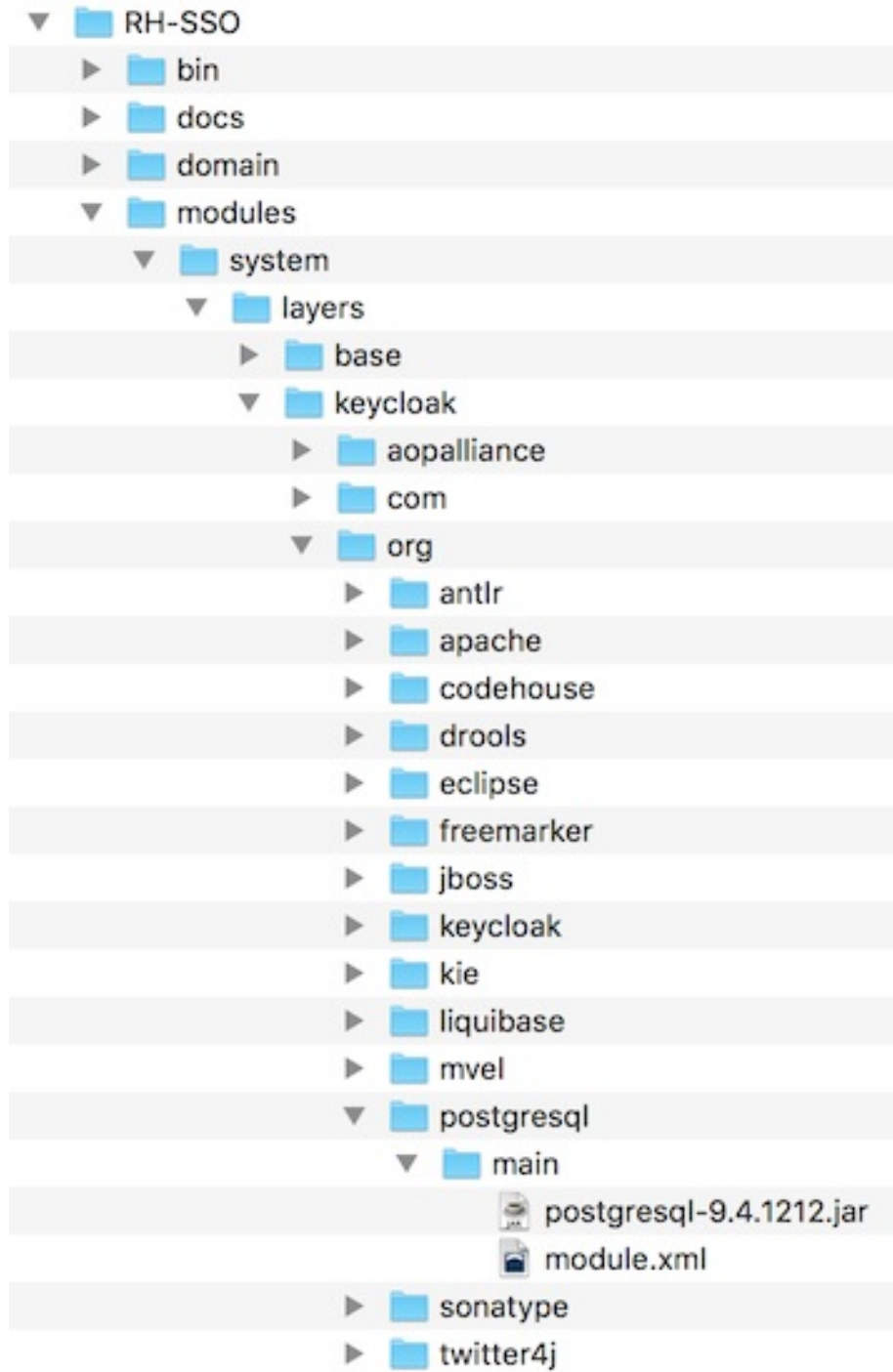
本章将使用 PostgreSQL 作为其所有示例。其他数据库遵循相同的安装步骤。

## 6.2. 打包 JDBC 驱动程序

查找并下载用于您的 RDBMS 的 JDBC 驱动程序 JAR。在使用这个驱动程序前，您必须将它打包到模块中，并将其安装到服务器中。模块定义加载到 Red Hat Single Sign-On 类路径中的 JAR，以及那些 JAR 在其他模块上具有的依赖项。它们易于设置。

在 Red Hat Single Sign-On 发行版本的 `.../modules/` 目录中，您需要创建一个目录结构来保存您的模块定义。其惯例使用 JDBC 驱动程序的 Java 软件包名称作为目录结构的名称。对于 PostgreSQL，创建目录 `org/postgresql/main`。将数据库驱动程序 JAR 复制到这个目录中，并在其中创建一个空的 `module.xml` 文件。

模块目录



完成后，打开 `module.xml` 文件并创建以下 XML：

### 模块 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>
</module>
```

```

</resources>

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

```

模块名称应与模块的目录结构匹配。因此，`org/postgresql` 映射到 `org.postgresql`。 `resource-root path` 属性应指定驱动程序的 JAR 文件名。其余仅是任何 JDBC 驱动程序 JAR 存在的正常依赖项。

### 6.3. 声明和加载 JDBC 驱动程序

接下来，您需要做的是，将新打包的 JDBC 驱动程序声明成您的部署配置文件，使它加载并在服务器启动时可用。执行此操作的位置取决于您的 [操作模式](#)。如果您要以标准模式部署，请编辑 ... `/standalone/configuration/standalone.xml`。如果您要以标准集群模式部署，请编辑 ... `/standalone/configuration/standalone-ha.xml`。如果您要以域模式部署，请编辑 ... `/domain/configuration/domain.xml`。在域模式中，您需要确保编辑您要使用的配置集：`auth-server-standalone` 或 `auth-server-clustered`

在配置集中，搜索 `datasources` 子系统下的驱动程序 XML 块。您应该会看到为 H2 JDBC 驱动程序声明的预定义驱动程序。这是您将为外部数据库声明 JDBC 驱动程序的位置。

#### JDBC 驱动程序

```

<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>

```

在驱动程序 XML 块中，您需要声明额外的 JDBC 驱动程序。它需要一个名称，您可以选择该名称作

为您想要的任何内容。您可以指定指向您之前为驱动程序 JAR 创建的模块软件包的模块属性。最后，您必须指定驱动程序的 Java 类。以下是在本章前面定义的模块示例中安装 PostgreSQL 驱动程序的示例。

### 声明您的 JDBC 驱动程序

```
<subsystem xmlns="urn:jboss:domain:datasources:5.0">
  <datasources>
    ...
    <drivers>
      <driver name="postgresql" module="org.postgresql">
        <xa-datasource-class>org.postgresql.xa.PGXADDataSource</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. 修改 RED HAT SINGLE SIGN-ON 数据源

在声明 JDBC 驱动程序后，您必须修改 Red Hat Single Sign-On 用来将其连接到您的新外部数据库的现有数据源配置。您将在您注册 JDBC 驱动程序的同一配置文件和 XML 块中执行此操作。下面是设置与新数据库的连接示例：

### 声明您的 JDBC 驱动程序

```
<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>
```

为 **KeycloakDS** 搜索数据源定义。首先需要修改 `connection-url`。您的供应商 **JDBC** 实施的文档应指定此连接 **URL** 值的格式。

接下来定义您要使用的 **驱动程序**。这是您在本章上一节中声明的 **JDBC 驱动程序** 的逻辑名称。

每次您要执行事务时，都必须打开与数据库的新连接。为了补偿，数据源实施会维护一个开放连接池。`max-pool-size` 指定它将池的最大连接数。您可能希望根据系统的负载更改此值。

最后，在使用 **PostgreSQL** 时，您需要定义连接到数据库所需的数据库用户名和密码。您可能会担心在示例中是明文。有方法模糊处理，但这超出了本指南的范围。



#### 注意

有关数据源功能的更多信息，请参阅 **JBoss EAP 配置指南** 中的数据源配置章节。[https://access.redhat.com/documentation/zh-cn/red\\_hat\\_jboss\\_enterprise\\_application\\_platform/7.3/html-single/configuration\\_guide/#datasource\\_management](https://access.redhat.com/documentation/zh-cn/red_hat_jboss_enterprise_application_platform/7.3/html-single/configuration_guide/#datasource_management)

## 6.5. 数据库配置

此组件的配置可在您的发行版本中的 `standalone.xml`、`standalone-ha.xml` 或 `domain.xml` 文件中找到。此文件的位置取决于您的 **操作模式**。

### 数据库配置

```
<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>

```

可能的配置选项有：

### **dataSource**

**dataSource** 的 JNDI 名称

### **jta**

指定数据源是否能够 JTA 的布尔值属性

### **driverDialect**

数据库划分的值。在大多数情况下，您不需要将此属性指定为 **dialect** 将由 **Hibernate** 自动探测到。

### **initializeEmpty**

如果为空，则初始化数据库。如果设置为 **false**，则必须手动初始化数据库。如果要手动将数据库设置 **migrationStrategy** 改为 **manual**，这会使用 **SQL** 命令创建文件来初始化数据库。默认值为 **true**。

### **migrationStrategy**

用于迁移数据库的策略。有效值为 **update**、**manual** 和 **validate**。**update** 将自动迁移数据库架构。手动将使用 **SQL** 命令将所需的更改导出到文件，您可以手动对数据库执行。验证将简单检查数据库是最新的。

### **migrationExport**

编写手动数据库初始化/迁移文件的路径。

### **showSql**

指定 **Hibernate** 是否应在控制台中显示所有 **SQL** 命令（默认为 **false**）。这非常详细！

### **formatSql**

指定 *Hibernate* 是否应该格式化 SQL 命令 (默认为 *true*)

### *globalStatsInterval*

将记录 *Hibernate* 有关已执行 DB 查询和其他事务的全局统计信息。统计数据始终以指定间隔 (以秒为单位) 报告到服务器日志, 并在每次报告后清除。

### *schema*

指定要使用的数据库架构



注意

[JBoss EAP 开发指南](#) 中介绍了这些配置交换机等。

## 6.6. 数据库的 UNICODE 注意事项

*Red Hat Single Sign-On* 中的数据库模式仅在以下特殊字段中考虑 *Unicode* 字符串 :

- *realms* : 显示名称、HTML 显示名称
- 联邦供应商 : 显示名称
- *Users*: *username*, *given name*, *last name*, *attribute name* and *values*
- *groups*: *name*, *attribute name* 和 *values*
- *roles*: *name*
- 对象的描述

否则, 字符仅限于包含在数据库编码中的字符, 通常是 8 位。但是, 对于某些数据库系统, 可以启用 *Unicode* 字符的 UTF-8 编码, 并在所有文本字段中使用完整的 *Unicode* 字符集。通常, 这以比 8 位编码更短的字符串的最大长度相平衡。

某些数据库需要特殊设置数据库和/或 JDBC 驱动程序才能处理 Unicode 字符。请在下面找到您的数据库的设置。请注意，如果此处列出了数据库，它仍然可以在数据库和 JDBC 驱动程序级别上正确处理 UTF-8 编码。

从技术上讲，对于 Unicode 支持所有字段的关键条件是数据库是否允许为 VARCHAR 和 CHAR 字段设置 Unicode 字符。如果是，则 Unicode 将存在的高可能性，通常以字段长度为代价。如果它只在 NVARCHAR 和 NCHAR 字段中支持 Unicode，则对所有文本字段的 Unicode 支持不太可能，因为 Keycloak 模式会广泛使用 VARCHAR 和 CHAR 字段。

### 6.6.1. Oracle 数据库

Unicode 字符可以被正确处理，提供了数据库在 VARCHAR 和 CHAR 字段中支持（例如，使用 AL32UTF8 字符集作为数据库字符集）创建的 Unicode 字符。JDBC 驱动程序不需要特殊设置。

如果数据库字符集不是 Unicode，则在特殊字段中使用 Unicode 字符，则需要配置 JDBC 驱动程序，并将连接属性 `oracle.jdbc.defaultNChar` 设置为 `true`。这可能是明智的，但并非严格必要，也要将 `oracle.jdbc.convert.convertNcharLiterals` 连接属性设为 `true`。这些属性可以设置为系统属性或连接属性。请注意，设置 `oracle.jdbc.defaultNChar` 可能会对性能造成负面影响。详情请查看 Oracle JDBC 驱动程序配置文档。

### 6.6.2. Microsoft SQL Server 数据库

Unicode 字符仅针对特殊字段正确处理。不需要特殊设置 JDBC 驱动程序或数据库。

### 6.6.3. MySQL 数据库

Unicode 字符可以被正确处理，提供了数据库是使用 CREATE DATABASE 命令的 VARCHAR 和 CHAR 字段中的 Unicode 支持创建的（例如，使用 utf8 字符集作为 MySQL 5.5 中设置的默认数据库字符）。请注意，utf8mb4 字符集无法正常工作，因为不同的存储要求被设置为 utf8 字符集<sup>[1]</sup>。请注意，在这种情况下，非特殊字段的长度限制不适用，因为创建了列来容纳给定的字符数，而不是字节。如果数据库默认字符集不允许存储 Unicode，则只有特殊字段允许存储 Unicode 值。

在 JDBC 驱动程序设置一侧，需要将连接属性 `characterEncoding=UTF-8` 添加到 JDBC 连接设置中。

### 6.6.4. PostgreSQL 数据库

当数据库字符设置为 UTF8 时，支持 Unicode。在这种情况下，任何字段中都可以使用 Unicode 字符，非特殊字段没有减少字段长度。不需要特殊设置 JDBC 驱动程序。

PostgreSQL 数据库的字符集在创建时决定。您可以使用 SQL 命令确定 PostgreSQL 集群的默认字符集

```
show server_encoding;
```

如果默认字符集不是 UTF 8，您可以使用 UTF8 创建数据库作为其字符集，如下所示：

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

---

[1]

Tracked as <https://issues.redhat.com/browse/KEYCLOAK-3873>

## 第 7 章 主机名

**Red Hat Single Sign-On** 将公共主机名用于多个内容。例如，在令牌签发者字段中，在密码重置电子邮件中发送。

**Hostname SPI** 提供了一种为请求配置主机名的方法。默认提供程序允许为 **frontend** 请求设置固定 URL，同时允许后端请求基于请求 URI。如果内置供应商不提供所需的功能，也可以自行开发自己的供应商。

### 7.1. 默认供应商

默认主机名提供程序使用配置的 **frontendUrl** 作为 **frontend** 请求（来自 **user-agents** 的请求）的基本 URL，并使用请求 URL 作为后端请求（直接来自客户端的请求）的基础。

**frontend** 请求不必与 **Keycloak** 服务器具有相同的 **context-path**。这意味着您可以在上公开 **Keycloak**，例如 <https://auth.example.org> 或 <https://example.org/keycloak>，而内部的 URL 可以是 <https://10.0.0.10:8080/auth>。

这使得 **user-agents (browsers)** 可以通过公共域名向 `${project.name}` 发送请求，而内部客户端可以使用内部域名或 IP 地址。

这反映在 **OpenID Connect Discovery** 端点中，例如 **authorization\_endpoint** 使用 **frontend URL**，而 **token\_endpoint** 使用后端 URL。请注意，此处实例的公共客户端将通过公共端点联系 **Keycloak**，这会导致 **authorization\_endpoint** 和 **token\_endpoint** 的基本是相同的。

要为 **Keycloak** 设置 **frontendUrl**，您可以将 `add -Dkeycloak.frontendUrl=https://auth.example.org` 传递给启动，或者在 `standalone.xml` 中配置它。请参见以下示例：

```
<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>
```

要使用 `jboss-cli` 更新 **frontendUrl**，请使用以下命令：

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

如果您希望所有请求都经过公共域名，您可以通过将 `forceBackendUrlToFrontendUrl` 设置为 `true` 来强制后端请求使用 `frontend URL`。

也可以覆盖各个域的默认 `frontend URL`。这可以在管理控制台中完成。

如果您不想在公共域中公开管理端点和控制台，请使用 `adminUrl` 属性为 `admin` 控制台设置固定 `URL`，这与 `frontendUrl` 不同。有关如何进行此操作的详情，还需要阻止从外部访问 `/auth/admin`。

## 7.2. 自定义供应商

要开发自定义主机名供应商，您需要实施 `org.keycloak.urls.HostnameProviderFactory` 和 `org.keycloak.urls.HostnameProvider`。

有关如何开发自定义供应商的更多信息，请参阅 [Server Developer Guide](#) 中的 `Service Provider Interfaces` 部分中的说明。

## 第 8 章 网络设置

**Red Hat Single Sign-On 可以开箱即用，但有一些网络限制。对于一个，所有网络端点都绑定到 localhost，因此 auth 服务器实际上仅在一台本地计算机上可用。对于基于 HTTP 的连接，不使用默认端口，如 80 和 443。HTTPS/SSL 没有配置开箱即用，如果没有它，Red Hat Single Sign-On 有很多安全漏洞。最后，Red Hat Single Sign-On 可能需要向外部服务器进行安全 SSL 和 HTTPS 连接，因此需要设置信任存储，以便正确验证端点。本章讨论了所有这些内容。**

### 8.1. 绑定地址

默认情况下，Red Hat Single Sign-On 绑定到 localhost 回送地址 127.0.0.1。如果您想在网络上可用的身份验证服务器，则这不是非常有用的默认值。通常，我们建议您在公共网络上部署反向代理或负载均衡器，并将流量路由到私有网络上的单个 Red Hat Single Sign-On 服务器实例。在这两种情况下，您仍然需要设置网络接口，以绑定到 localhost 以外的其他接口。

设置绑定地址非常简单，可以在命令行中使用 standalone.sh 或 domain.sh boot 脚本（在 [Choosing an Operating Mode](#) 章节中讨论）在命令行中完成。

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

**-b** 交换机为任何公共接口设置 IP 绑定地址。

或者，如果您不希望在命令行中设置绑定地址，您可以编辑部署的配置集配置。打开 profile 配置文件 (standalone.xml 或 domain.xml，具体取决于您的 [操作模式](#))，并查找 接口 XML 块。

```
<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>
```

公共接口 对应于创建公开可用的套接字的子系统。其中其中一个子系统的示例是提供 Red Hat Single Sign-On 身份验证端点的 Web 层。管理接口 对应于 JBoss EAP 管理层打开的套接字。特别是允许您使用 jboss-cli.sh 命令行界面和 JBoss EAP Web 控制台的套接字。

在查看 公共接口 中，您会看到它具有特殊字符串 {jboss.bind.address:127.0.0.1}。此字符串表示值 127.0.0.1，可通过设置 Java 系统属性（例如：

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

**-b** 只是此命令的一个简写表示法。因此，您可以直接在配置集配置中更改 **bind address** 值，或者在引导时在命令行中更改它。



### 注意

设置接口定义时，还有更多可用选项。有关更多信息，请参阅 [JBoss EAP 配置指南](#) 中的 [网络接口](#)。

## 8.2. 套接字端口绑定

为每个套接字打开的端口具有预定义的默认值，可以在命令行或配置中覆盖。为了说明此配置，我们预先 [以独立模式运行](#)，并打开 `.../standalone/configuration/standalone.xml`。搜索 `socket-binding-group`。

```
<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` 定义将由服务器打开的套接字连接。这些绑定指定了它们使用的接口（绑定地址），以及它们将打开的端口号。您最感兴趣的是：

### http

定义用于 Red Hat Single Sign-On HTTP 连接的端口

### https

定义用于 Red Hat Single Sign-On HTTPS 连接的端口

### ajp



此套接字绑定定义用于 AJP 协议的端口。当您将 Apache HTTPD 服务器用作负载均衡器时，Apache HTTPD 服务器会结合使用此协议。

## management-http

定义 JBoss EAP CLI 和 Web 控制台使用的 HTTP 连接。

当在域模式下运行时，套接字配置会稍微复杂，因为示例 domain.xml 文件定义了多个 socket-binding-groups。如果向下滚动到 server-group 定义，您可以看到每个 server-group 使用了哪些 socket-binding-group。

### 域套接字绑定

```
<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>
```

### 注意

设置 socket-binding-group 定义时，还有更多可用选项。有关更多信息，请参阅 JBoss EAP 配置指南中的 [套接字绑定组](#)。

## 8.3. 设置 HTTPS/SSL

**警告**

默认情况下，Red Hat Single Sign-On 没有设置为处理 SSL/HTTPS。强烈建议您在 Red Hat Single Sign-On 服务器本身或 Red Hat Single Sign-On 服务器前在反向代理中启用 SSL。

这个默认行为由每个 Red Hat Single Sign-On 域的 SSL/HTTPS 模式定义。[服务器管理指南中的](#) 更为详细地讨论，但让我们提供了一些上下文以及这些模式的简要概述。

**外部请求**

Red Hat Single Sign-On 可以在没有 SSL 的情况下运行，只要您坚持使用私有 IP 地址（如 localhost、127.0.0.1、10.x.x.x、192.168.x.x 和 172.16.x.x）。如果您没有在服务器上配置 SSL/HTTPS，或者您试图通过 HTTP 从非专用 IP 地址访问 Red Hat Single Sign-On，则会出现错误。

**none**

Red Hat Single Sign-On 不需要 SSL。您应该真正在开发过程中使用。

**所有请求**

Red Hat Single Sign-On 需要 SSL 用于所有 IP 地址。

每个域的 SSL 模式可以在 Red Hat Single Sign-On 管理控制台中配置。

**8.3.1. 为 Red Hat Single Sign-On 服务器启用 SSL/HTTPS**

如果您不使用反向代理或负载均衡器来处理 HTTPS 流量，则需要为 Red Hat Single Sign-On 服务器启用 HTTPS。这涉及

1. 获取或生成包含 SSL/HTTP 流量的私钥和证书的密钥存储
2. 配置 Red Hat Single Sign-On 服务器以使用此密钥对和证书。

**8.3.1.1. 创建证书和 Java 密钥存储**

为了允许 HTTPS 连接，您需要获取自签名或第三方签名的证书，并将其导入到 Java 密钥存储中，然后才能在您将 Red Hat Single Sign-On 服务器部署到 web 容器中启用 HTTPS。

### 8.3.1.1.1. 自签名证书

在开发中，您可能没有第三方签名的证书来测试 Red Hat Single Sign-On 部署，因此您需要使用与 Java JDK 附带的 `keytool` 程序生成自签名证书。

```
$ 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
```

您应该回答您的名字和姓氏是什么？问题是您要安装该服务器的计算机的 DNS 名称。出于测试目的，应使用 `localhost`。执行此命令后，`keycloak.jks` 文件将生成在您执行 `keytool` 命令相同的目录中。

如果您希望第三方签名的证书，但没有一个证书，您可以在 [cacert.org](http://cacert.org) 获取一个空闲证书。但是，在执行此操作之前，必须先进行一些设置。

首先要做的是生成一个证书请求：

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

其中 `yourdomain` 是为其生成此证书的 DNS 名称。 `keytool` 生成请求：

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIC2jCCAcICAQAwZTElMAkGA1UEBhMCVVMxCzAJBgNVBAgTAK1BMREwDwYDVQQHEwhXZX
N0Zm9y
ZDEQMA4GA1UEChMHUUmVkiEhhdDEQMA4GA1UECxMHUUmVkiEhhdDESMBAGA1UEAxMJbG9jY
Wxob3N0
```

```

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAr7kck2TaavIEOGbcpi9c0rncY4HhdzmY
Ax2nZfq1eZEaIPqI5aTxwQZzzLDK9qbeAd8Ji79HzSqnRDxNYaZu7mAYhFKHgixsolE3o5Yfzbw1
29RvyeUVe+WZxv5oo9wolVvpdSINIMEL2LaFhtX/c1dqiyVpfnvFshZQalg2nL8juzZcBij4as
H98glS7khql/dkZKsw9NLvyxgJvp7PaXurX29fNf3ihG+oFrL22oFyV54BWWxXCKU/GPn61EGZGw
Ft2qSIGLdctpMD1aJR2bcnlhEjZKDKsjQZoQ5YMXaAGkcYkG6QkgrocDE2YXDbi7Gldf9MegVJ35
2DQMpwIDAQABoDAwLgYJKoZIhvcNAQkOMSEwHzAdBgNVHQ4EFgQUQwLZJBA+fjiDdiVzaO9vrE/i

n2swDQYJKoZIhvcNAQELBQADggEBAC5FRvMkhal3q86tHPBYWBUtTmcSjs4qUm6V6f63frhveWHf
PzRrI1xH272XUleBk0gtzWo0nNZnf0mMctUBbHhhDcG82xolikfqibZijoQZCiGiedVjHJFtniDQ
9bMDUOXEMQ7gHZg5q6mJfNG9MbMpQaUVEEFvfGEQQxbiFK7hRWU8S23/d80e8nExgQxdJWJ6v
d0X
MzzFK6j4Dj55bJVuM7GFmfdNC52pNOD5vYe47Aqh8oajHX9XTycVtPXI45rrWAH33ftbrS8SrZ2S
vqIFQeuLL3BaHwpl3t7j2IMWcK1p80laAxEASib/fAwrrHhPLHBXRcq6uALUOZI4Alt8=
-----END NEW CERTIFICATE REQUEST-----

```

将此 **ca** 请求发送到您的 **CA**。**CA** 将发出您签名的证书并将其发送给您。在导入新证书前，您必须获取并导入 **CA** 的 **root** 证书。您可以从 **CA** 下载证书（例如：**root.crt**）并导入，如下所示：

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

最后一步是将生成的新 **CA** 证书导入到密钥存储中：

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

### 8.3.1.2. 配置红帽单点登录以使用密钥存储

现在，您已有一个带有适当证书的 **Java** 密钥存储，您需要配置 **Red Hat Single Sign-On** 安装以使用它。首先，您必须编辑 **standalone.xml**、**standalone-ha.xml** 或 **host.xml** 文件，以使用密钥存储并启用 **HTTPS**。然后，您可以将密钥存储文件移到部署的 **configuration/** 目录中，也可以是您选择的位置的文件，并提供绝对路径。如果您使用绝对路径，请从您的配置中删除可选的 **relative-to** 参数(See [操作模式](#))。

使用 **CLI** 添加新的 **security-realm** 元素：

```
$ /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)
```

如果使用域模式，则应在使用 **/host=<host\_name>/** 前缀的每个主机上执行命令（以便在其中创建 **security-realm**），如下所示，您要为每个主机重复这些命令：

```
$ /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)
```

在独立或主机配置文件中，`security-realms` 元素应该类似如下：

```
<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>
```

接下来，在独立或每个域配置文件中搜索任何 `security-realm` 实例。修改 `https-listener` 以使用创建的域：

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

如果使用域模式，请在命令前添加用于的配置文件的：`/profile=<profile_name>/。`

生成的元素 `server name="default-server"`，它是子系统 `xmlns="urn:jboss:domain:undertow:10.0"` 的子元素，应包含以下小节：

```
<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"/>
    ...
  </server>
</subsystem>
```

#### 8.4. 传出 HTTP 请求

**Red Hat Single Sign-On 服务器**通常需要对安全的应用程序和服务进行非浏览器 HTTP 请求。`auth` 服务器通过维护 HTTP 客户端连接池来管理这些传出连接。在 `standalone.xml`、`standalone-ha.xml` 或 `domain.xml` 中需要配置一些内容。此文件的位置取决于您的 [操作模式](#)。

#### HTTP 客户端配置示例

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

可能的配置选项有：

#### **establish-connection-timeout-millis**

建立套接字连接的超时。

#### **socket-timeout-millis**

如果传出请求没有收到这段时间的数据，请超时连接。

#### **connection-pool-size**

池中可以有多个连接（默认为 128）。

#### **max-pooled-per-route**

可以为每个主机池多少个连接（默认为 64）。

#### **connection-ttl-millis**

以毫秒为单位进行的最大连接时间。默认不设置。

#### **max-connection-idle-time-millis**

连接在连接池中可能闲置的最长时间（默认为900秒）。将启动 Apache HTTP 客户端的后台清理线程。设置为 -1，以禁用此检查和后台线程。

#### **disable-cookies**

默认情况下为 true。当设置为 true 时，这将禁用任何 Cookie 缓存。

#### **client-keystore**

这是 Java 密钥存储文件的文件路径。此密钥存储包含双向 SSL 的客户端证书。

#### **client-keystore-password**

客户端密钥存储的密码。如果设置了 `client-keystore`，则这是 **REQUIRED**。

#### `client-key-password`

客户端密钥的密码。如果设置了 `client-keystore`，则这是 **REQUIRED**。

#### `proxy-mappings`

表示传出 HTTP 请求的代理配置。如需了解更多详细信息，请参阅有关 [Proxy Mappings for Outgoing HTTP Requests](#) 部分。

#### `disable-trust-manager`

如果传出请求需要 HTTPS，且此配置选项被设置为 `true`，则不必指定信任存储。此设置应只在开发期间使用，且永远不会在生产环境中使用，因为它将禁用 SSL 证书的验证。这是 **OPTIONAL**。默认值为 `false`。

### 8.4.1. 传出 HTTP 请求的代理映射

Red Hat Single Sign-On 发送的传出 HTTP 请求可以选择使用基于以逗号分隔的 `proxy-mapping` 的代理服务器。`proxy-mapping` 以 `hostnamePattern;proxyUri` 的形式表示基于 `regex` 的主机名模式和 `proxy-uri` 的组合，如：

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

要确定传出 HTTP 请求的代理，目标主机名与配置的主机名模式匹配。第一个匹配模式决定了要使用的 `proxy-uri`。如果没有为给定主机名配置任何配置的模式匹配，则不会使用代理。

如果代理服务器需要身份验证，请将代理用户的凭据包含在以下格式 `username:password@`。例如：

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

`proxy-uri` 的特殊值 `NO_PROXY` 可用于指示不应将代理用于与关联的主机名模式匹配的主机。可以在 `proxy-mappings` 的末尾指定一个 `catch-all` 模式，为所有传出请求定义默认代理。

以下示例演示了 `proxy-mapping` 配置。

```
# 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
```

这可以通过以下 `jboss-cli` 命令进行配置。请注意，您需要正确转义 `regex-pattern`，如下所示。

```
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` 命令生成以下子系统配置：请注意，需要对 " 字符 " 进行编码。

```
<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. 传出 HTTPS 请求 Truststore

当 Red Hat Single Sign-On 在远程 HTTPS 端点上调用时，必须验证远程服务器的证书，以确保它连接到可信服务器。这是为了防止中间人攻击所必需的。这些远程服务器的证书或签署这些证书的 CA 必须放在信任存储中。此信任存储由 Red Hat Single Sign-On 服务器管理。

在安全地连接到身份代理、LDAP 身份提供程序、发送电子邮件和与客户端应用程序的后端通道通信时，使用 `truststore`。



**警告**

默认情况下，不会配置信任存储提供程序，任何 https 连接都回退到标准的 java 信任存储配置，如 [Java 的 JSSE 参考指南](#) 中所述。如果没有建立信任，则这些传出 HTTPS 请求将失败。

您可以使用 `keytool` 创建新的信任存储文件，或向现有主机添加可信主机证书：

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

`truststore` 在您分发的 `standalone.xml`、`standalone-ha.xml` 或 `domain.xml` 文件中配置。此文件的位置取决于您的 [操作模式](#)。您可以使用以下模板添加信任存储配置：

```
<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>
```

此设置可能的配置选项有：

**file**

Java 密钥存储文件的路径。HTTPS 请求需要一种方法来验证它们要与之通信的服务器的主机。这是信任者的作用。密钥存储包含一个或多个可信主机证书或证书颁发机构。此 `truststore` 文件应该只包含您的安全主机的公共证书。如果禁用，则这是 **REQUIRED**。

**password**

`truststore` 的密码。如果禁用，则这是 **REQUIRED**。

**hostname-verification-policy**

默认 **WILDCARD**。对于 HTTPS 请求，这会验证服务器证书的主机名。**ANY** 表示主机名不会被验证。**WILDCARD** Allows wildcard in subdomain name i.e. `lfoo.com`。**STRICT** CN 必须完全匹配

主机名。

**disabled**

如果为 **true**（默认值），则忽略信任存储配置，证书检查将回退到 **JSSE** 配置，如下所述。如果设置为 **false**，则必须为信任存储 配置文件，和密码。

## 第 9 章 集群

本节论述了将 Red Hat Single Sign-On 配置为在集群中运行。设置集群时需要执行很多操作，特别是：

- [选择操作模式](#)
- [配置共享外部数据库](#)
- [设置负载均衡器](#)
- [提供支持 IP 多播的专用网络](#)

本指南前面讨论了一种操作模式和配置共享数据库。在本章中，我们将讨论设置负载均衡器并提供专用网络。我们还将讨论在引导集群中的主机时需要注意的一些问题。

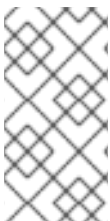


### 注意

可以在没有 IP 多播的情况下集群 Red Hat Single Sign-On，但本主题超出了本指南的范围。有关更多信息，请参阅 [JBoss EAP 配置指南中的 JGroups 章节](#)。

### 9.1. 推荐的网络架构

部署红帽单点登录的建议网络架构是在公共 IP 地址上设置 HTTP/HTTPS 负载均衡器，该地址将请求路由到位于专用网络上的 Red Hat Single Sign-On 服务器。这会隔离所有群集连接，并提供保护服务器的 nice 方法。



### 注意

默认情况下，无法防止未授权节点加入集群并广播多播信息。这就是为什么集群节点应该位于私有网络中，使用防火墙保护它们免受外部攻击。

### 9.2. 集群示例

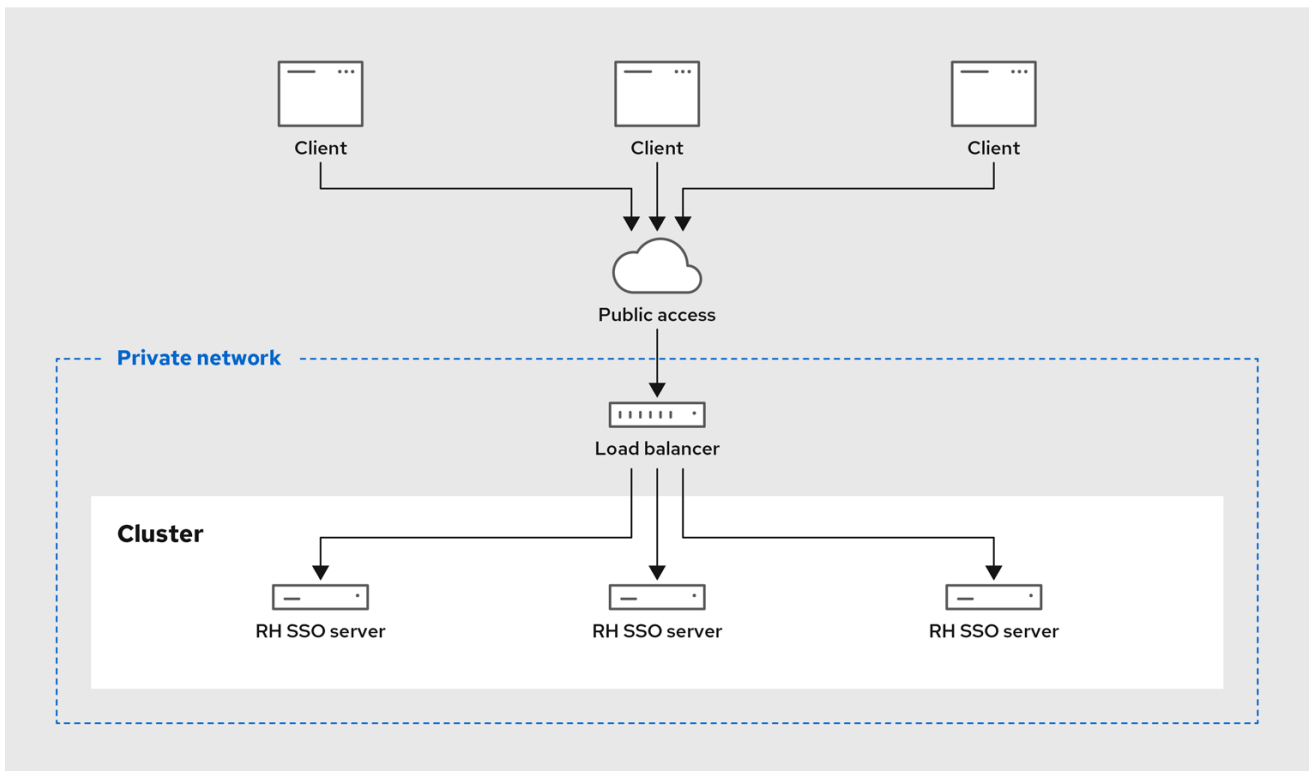
Red Hat Single Sign-On 附带一个 box 集群演示，它利用域模式。详情请查看[集群域示例](#) 章节。

### 9.3. 设置负载均衡器或代理

本节讨论在集群 Red Hat Single Sign-On 部署前放置反向代理或负载均衡器前需要配置的一些内容。它还涵盖配置[集群域示例](#) 的内置负载均衡器。

下图演示了负载均衡器的使用。在本例中，负载均衡器充当三个客户端和三个 Red Hat Single Sign-On 服务器的集群之间的反向代理。

#### Load Balancer Diagram 示例



70\_RHSSO\_0320

#### 9.3.1. 识别客户端 IP 地址

Red Hat Single Sign-On 中的一些功能依赖于连接到身份验证服务器的 HTTP 客户端的远程地址是客户端机器的实际 IP 地址。示例包括：

- 事件日志 - 使用错误的源 IP 地址记录失败的登录尝试
- SSL 必需 - 如果需要 SSL 设置为 `external` (默认值), 则所有外部请求都需要 SSL
- 身份验证流 - 使用 IP 地址的自定义身份验证流, 如 `show OTP` 仅适用于外部请求
- 动态客户端注册

当您在 Red Hat Single Sign-On 身份验证服务器前面有一个反向代理或 loadbalancer 时, 这可能会造成问题。常见的设置是, 您有一个 `frontend` 代理位于公共网络上, 该代理负载平衡并将请求转发到专用网络中的后端 Red Hat Single Sign-On 服务器实例。在这种情况下, 您必须执行一些额外的配置, 以便实际的客户端 IP 地址被 Red Hat Single Sign-On 服务器实例转发到和处理。具体来说:

- 将您的反向代理或负载均衡器配置为正确设置 `X-Forwarded-For` 和 `X-Forwarded-Proto` HTTP 标头。
- 将您的反向代理或负载均衡器配置为保留原始的 'Host' HTTP 标头。
- 配置身份验证服务器, 以从 `X-Forwarded-For` 标头读取客户端的 IP 地址。

配置代理以生成 `X-Forwarded-For` 和 `X-Forwarded-Proto` HTTP 标头, 并保留原始主机 HTTP 标头超出了本指南的范围。采取额外的措施来确保代理设置了 `X-Forwarded-For` 标头。如果您的代理没有正确配置, 则恶意客户端可以自行设置此标头, 并欺骗 Red Hat Single Sign-On 认为客户端正在从不同的 IP 地址连接。如果您正在执行 IP 地址的黑色或白名单, 这非常重要。

除了代理本身外, 您需要在 Red Hat Single Sign-On 端配置一些操作。如果您的代理通过 HTTP 协议转发请求, 则需要配置 Red Hat Single Sign-On, 以从 `X-Forwarded-For` 标头而不是从网络数据包中提取客户端的 IP 地址。为此, 请打开配置文件 `standalone.xml`、`standalone-ha.xml` 或 `domain.xml`, 具体取决于您的 [操作模式](#), 并查找 `urn:jboss:domain:undertow:10.0` XML 块。

### X-Forwarded-For HTTP 配置

```
<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>

```

将 `proxy-address-forwarding` 属性添加到 `http-listener` 元素。将值设为 `true`。

如果您的代理使用 **AJP** 协议而不是 **HTTP** 来转发请求（即 **Apache HTTPD + mod-cluster**），那么您必须配置一些不同的内容。您需要添加一个过滤器来从 **AJP** 数据包中提取此信息，而不是修改 `http-listener`。

### X-Forwarded-For AJP 配置

```

<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. 使用 Reverse 代理启用 HTTPS/SSL

假设您的反向代理没有为 **SSL** 使用端口 **8443**，您还需要配置将哪些端口 **HTTPS** 流量重定向到其中。

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

将 `redirect-socket` 属性添加到 `http-listener` 元素。该值应该是 `proxy-https`，它指向您还需要定义的套接字绑定。

然后，在 `socket-binding - group` 元素中添加一个新的 `socket-binding` 元素：

```
<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. 验证配置

您可以通过反向代理打开路径 `/auth/realms/master/.well-known/openid-configuration` 来验证反向代理或负载均衡器配置。例如，如果反向代理地址是 `https://acme.com/`，则打开 URL `https://acme.com/auth/realms/master/.well-known/openid-configuration`。这将显示一个 JSON 文档，其中列出了 Red Hat Single Sign-On 的多个端点。确保端点以反向代理或负载均衡器的地址 (`scheme`、`domain` 和 `port`) 开头。通过这样做，您可以确保 Red Hat Single Sign-On 使用正确的端点。

您还应验证 Red Hat Single Sign-On 是否看到请求的正确源 IP 地址。要检查这一点，您可以尝试使用无效的用户名和/或密码登录到管理控制台。这应该会在服务器日志中显示警告信息，如下所示：

```
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
```

检查 `ipAddress` 的值是您尝试使用登录的机器的 IP 地址，而不是反向代理或负载均衡器的 IP 地址。

### 9.3.4. 使用 Built-In Load Balancer

本节介绍配置群集 [域示例](#) 中讨论的内置负载均衡器。

**集群域示例** 仅设计为在一台计算机上运行。要在另一主机上启动从设备，您需要：

1. 编辑 `domain.xml` 文件以指向您的新主机从设备
2. 复制服务器分发。您不需要 `domain.xml`、`host.xml` 或 `host-master.xml` 文件。不需要 `standalone/` 目录。
3. 编辑 `host-slave.xml` 文件，以在命令行中更改使用或覆盖它们的绑定地址

#### 9.3.4.1. 使用 Load Balancer 注册新主机

我们首先在 `domain.xml` 中使用负载均衡器配置注册新主机 `slave`。打开此文件，再前往 `load-balancer` 配置文件中的 `undertow` 配置。在 `reverse-proxy` XML 块中添加名为 `remote-host 3` 的新主机定义。

##### `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` 是指向稍后在 `domain.xml` 文件中配置的 `socket-binding` 的逻辑名称。`instance-id` 属性还必须对新主机唯一，因为该值供 `Cookie` 用于在负载平衡时启用粘性会话。

接下来，向下滚动到 `load-balancer-sockets socket-binding-group`，再为 `remote-host3` 添加



**outbound-socket-binding**。这个新绑定需要指向新主机的主机和端口。

#### 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. Master 绑定地址

接下来，您需要更改 master 主机的公共和管理绑定地址。按照 [Bind Addresses](#) 章节中所述编辑 domain.xml 文件，或者在命令行中指定这些绑定地址，如下所示：

```
$ 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. 主机 Slave Bind Addresses

接下来，您必须更改公共、管理和域控制器绑定地址(jboss.domain.master-address)。编辑 host-slave.xml 文件，或在命令行中指定它们，如下所示：

```
$ 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
```

与主机从 IP 地址相关的 jboss.bind.address 和 jboss.bind.address.management 的值。jboss.domain.master.address 的值必须是域控制器的 IP 地址，这是 master 主机的管理地址。

#### 9.3.5. 配置其他 Load Balancer

有关如何使用其他基于软件的 [负载均衡器的信息](#)，请参阅 [JBoss EAP 配置指南中的 负载均衡](#) 部分。

#### 9.4. 粘性会话

典型的集群部署由专用网络上的负载均衡器（反向代理）和 2 个或更多 Red Hat Single Sign-On 服务器组成。出于性能的需要，如果负载均衡器将与特定浏览器会话相关的所有请求转发到同一 Red Hat Single Sign-On 后端节点，这可能很有用。

原因在于，Red Hat Single Sign-On 在覆盖的下面使用 Infinispan 分布式缓存，以保存与当前身份验证会话和用户会话相关的数据。Infinispan 分布式缓存默认配置有一个所有者。这意味着，特定的会话只保存在一个集群节点上，而其他节点需要远程查找会话（如果想要访问它）。

例如，如果带有 ID 为 123 的验证会话保存在 node1 上的 Infinispan 缓存中，那么 node2 需要通过网络将请求发送到 node1，以返回特定的会话实体。

如果特定的会话实体始终在本地可用，这可以通过粘性会话的帮助来完成。集群环境中的工作流带有公共前端负载均衡器和两个后端 Red Hat Single Sign-On 节点，如下所示：

- 用户发送初始请求以查看 Red Hat Single Sign-On 登录屏幕
- 此请求由 frontend 负载均衡器提供，该负载均衡器将其转发到一些随机节点（例如 node1）。严格说，节点不需要随机，但可以根据某些其他条件（客户端 IP 地址等）进行选择。它都取决于底层负载均衡器的实施和配置（反向代理）。
- Red Hat Single Sign-On 创建带有随机 ID（如 123）的验证会话，并将其保存到 Infinispan 缓存。
- Infinispan 分布式缓存根据会话 ID 的哈希值分配会话的主要所有者。有关此问题的更多详细信息，请参阅 [Infinispan 文档](#)。假设 Infinispan 分配 node2 是此会话的所有者。
- Red Hat Single Sign-On 创建 cookie AUTH\_SESSION\_ID，格式为 < session-id>. <owner-node-id >。在我们的示例中，它将为 123.node2。
- 使用红帽单点登录登录屏幕和浏览器中的 AUTH\_SESSION\_ID cookie 返回用户的响应

从此时，如果负载均衡器将所有下一个请求转发到 `node2`，因为这是 ID 为 123 的验证会话的所有者，因此 `Infinispan` 可以在本地查找此会话。身份验证完成后，身份验证会话将转换为用户会话，该会话也会保存在 `node2` 中，因为它具有相同的 ID 123。

集群设置的粘性会话不是强制的，但由于上述原因，最好是性能。您需要通过 `AUTH_SESSION_ID` cookie 将 `loadbalancer` 配置为粘性。具体操作取决于您的负载均衡器。

建议在 `Red Hat Single Sign-On` 端使用系统属性 `jboss.node.name`，其值对应于您的路由名称。例如，`-Djboss.node.name=node1` 将使用 `node1` 来识别路由。此路由将由 `Infinispan` 缓存使用，当节点是特定密钥的所有者时，将附加到 `AUTH_SESSION_ID` cookie。以下是使用此系统属性启动命令的示例：

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

通常，在生产环境中，路由名称应该使用与后端主机相同的名称，但这不是必需的。您可以使用不同的路由名称。例如，如果您要在专用网络中隐藏 `Red Hat Single Sign-On` 服务器的主机名。

#### 9.4.1. 禁用添加路由

有些负载均衡器可以配置为自行添加路由信息，而不依赖于后端 `Red Hat Single Sign-On` 节点。但是，如上所述，建议通过 `Red Hat Single Sign-On` 添加路由。这是因为，当以性能提高时，因为 `Red Hat Single Sign-On` 知道作为特定会话所有者的实体，并可路由到该节点，这不一定是本地节点。

如果需要，可以通过将以下内容添加到 `Red Hat Single Sign-On` 的 `RHSSO_HOME/standalone/configuration/standalone-ha.xml` 文件中，禁用在 `Red Hat Single Sign-On` 子系统配置中的 `AUTH_SESSION_ID` cookie 中添加路由信息：

```
<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. 多播网络设置

开箱即用的集群支持需要 IP 多播。多播是网络广播协议。此协议在引导时用于发现和加入集群。它还用于广播复制和 Red Hat Single Sign-On 使用的分布式缓存无效消息。

Red Hat Single Sign-On 的集群子系统在 JGroups 堆栈上运行。开箱即用，集群的绑定地址绑定到带有 127.0.0.1 作为默认 IP 地址的私有网络接口。您必须编辑 [Bind Address](#) 章节中讨论的 `standalone-ha.xml` 或 `domain.xml` 部分。

### 专用网络配置

```
<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>
```

您需要配置的内容是 `jboss.bind.address.private` 和 `jboss.default.multicast.address`，以及集群堆栈上服务的端口。



#### 注意

可以在没有 IP 多播的情况下集群 Red Hat Single Sign-On，但本主题超出了本指南的范围。有关更多信息，请参阅 [JBoss EAP 配置指南中的 JGroups](#)。

## 9.6. 保护集群通信

当集群节点在私有网络中隔离时，它需要访问专用网络才能加入集群或查看集群中的通信。另外，您还可以为集群通信启用身份验证和加密。只要您的专用网络是安全的，就不需要启用身份验证和加密。Red Hat Single Sign-On 在这两种情况下不会在集群中发送非常敏感信息。

如果要为集群通信启用身份验证和加密，请参阅 [JBoss EAP 配置指南中的保护集群](#)。

## 9.7. 序列化集群启动

Red Hat Single Sign-On 集群节点允许同时引导。当 Red Hat Single Sign-On 服务器实例引导时，可能需要进行一些数据库迁移、导入或首次初始化。DB 锁定用于防止在集群节点同时引导时与另一个操作冲突。

默认情况下，这个锁定的最大超时时间为 900 秒。如果节点正在等待这个锁定超过超时时间，它将无法引导。通常，您不需要增加/减少默认值，但仅在您的发行版中的 `standalone.xml`、`standalone-ha.xml` 或 `domain.xml` 文件中配置它。此文件的位置取决于您的 [操作模式](#)。

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

## 9.8. 引导集群

在集群中引导 Red Hat Single Sign-On 取决于您的 [操作模式](#)

### 独立模式

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

### 域模式

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

您可能需要使用其他参数或系统属性。例如，绑定主机或系统属性 `jboss.node.name` 的参数 `-b` 指定路由的名称，如 [Sticky Sessions](#) 部分所述。

## 9.9. 故障排除

- 请注意，当运行集群时，您应该会在两个集群节点的日志中看到类似如下的消息：

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

如果您只看到提到的一个节点，您的集群主机可能无法连接在一起。

通常，最好在私有网络上具有集群节点，而无需防火墙以用于它们之间的通信。防火墙只能对您的网络的公共访问点启用。如果出于某种原因，您仍需要在集群节点上启用防火墙，则需要打开一些端口。默认值为 UDP 端口 55200 和多播端口 45688，多播地址为 230.0.0.4。请注意，如果您想要为 JGroups 堆栈启用诊断等其他功能，您可能需要打开更多端口。Red Hat Single Sign-On 将大多数集群工作委托给 Infinispan/JGroups。有关更多信息，请参阅 [JBoss EAP 配置指南中的 JGroups](#)。

- 如果您对故障转移支持（高可用性）、驱除、过期和缓存调整感兴趣，请参阅 [第 10 章 服务器缓存配置](#)。

## 第 10 章 服务器缓存配置

Red Hat Single Sign-On 有两种类型的缓存。一种类型的缓存位于数据库前面，以减少 DB 的负载，并通过在内存中保留数据来降低总体响应时间。realm、client、role 和 user 元数据保存在这种类型的缓存中。这个缓存是本地缓存。即使位于具有更多 Red Hat Single Sign-On 服务器的集群中，本地缓存也不会使用复制。相反，它们只会本地保留副本，如果条目更新了一个 invalidation 消息，则消息会发送到集群的其余部分，该条目将被驱除。有单独的复制缓存工作，该任务是将失效消息发送到整个集群，其中哪些条目应从本地缓存驱除。这可大大减少网络流量，使效率更高，并避免通过线路传输敏感元数据。

第二类型的缓存处理管理用户会话、离线令牌，并跟踪登录失败，以便服务器可以检测密码临时和其他攻击。这些缓存中保存的数据只是临时的，但可能在集群中复制。

本章讨论了用于集群和非集群部署的这些缓存的一些配置选项。



### 注意

有关这些缓存的更多高级配置，请参阅 JBoss EAP 配置指南中的 [Infinispan](#) 部分。

### 10.1. 驱除和过期

为红帽单点登录配置了多个不同的缓存。有一个域缓存，其中包含有关安全应用程序、通用安全数据和配置选项的信息。另外，还有一个包含用户元数据的用户缓存。两者都会默认缓存到最多 10000 个条目，并使用最早使用的驱除策略。它们也与控制集群设置中驱除的对象修订缓存相关联。这个缓存会被隐式创建，且配置的大小已两倍。这同样适用于包含授权数据的授权缓存。密钥缓存包含有关外部密钥的数据，不需要具有专用的修订缓存。相反，它已明确声明了过期时间，因此密钥会定期过期，并强制从外部客户端或身份提供程序定期下载。

这些缓存的驱除策略和最大条目可以在 standalone.xml、standalone-ha.xml 或 domain.xml 中配置，具体取决于您的 [操作模式](#)。在配置文件中，其中包含 infinispan 子系统的部分，它类似如下：

```
<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>
</subsystem>
```

```

</local-cache>
...
</cache-container>

```

要限制或扩展允许的条目数量，只需添加或编辑 `object` 元素或特定缓存配置的 `expiration` 元素。

此外，还有单独的缓存会话，`clientSessions`、`offlineSessions`、`offline ClientSessions`、`loginFailures` 和 `actionTokens`。这些缓存在集群环境中分发，它们默认不绑定。如果绑定了它们，则可能存在一些会话丢失。Red Hat Single Sign-On 本身内部清除过期的会话，以避免在不限量的情况下增大这些缓存的大小。如果您看到因为大量会话导致的内存问题，您可以尝试：

- 增加集群大小（集群中更多节点意味着会话在节点间平均分配）
- 为 Red Hat Single Sign-On 服务器进程增加内存
- 减少所有者数量，以确保缓存保存在一个位置。详情请查看 [第 10.2 节“复制和故障切换”](#)
- 为分布式缓存禁用 `l1-lifespan`。如需了解更多详细信息，请参阅 [Infinispan 文档](#)
- 减少会话超时，可以针对 Red Hat Single Sign-On 管理控制台中的每个域单独完成。但是，这可能会影响最终用户的可用性。如需了解更多详细信息，请参阅 [超时](#)。

还有额外的复制缓存 `work`，它主要用于在集群节点之间发送消息；默认情况下，也会绑定它。但是，这个缓存不应该导致任何内存问题，因为此缓存中的条目会非常短。

## 10.2. 复制和故障切换

有一些缓存，如会话、`authenticationSessions`、`offlineSessions`、`login Failure` 和一些其他一些（更多详情请参阅 [第 10.1 节“驱除和过期”](#)），在使用集群设置时将它们配置为分布式缓存。条目不会复制到每个节点，而是选择一个或多个节点作为该数据的所有者。如果节点不是特定缓存条目的所有者，它会查询集群来获取它。这意味着，如果拥有数据的所有节点都停机，则数据将永久丢失。默认情况下，Red Hat Single Sign-On 只为数据指定一个所有者。因此，如果一个节点停止那个数据，则该节点会丢失。这通常意味着用户将被注销，必须再次登录。

您可以通过更改 `distributed-cache` 声明中的 `owners` 属性来更改复制数据的节点数量。



**owners**

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

此处我们更改了它，因此至少两个节点将复制一个特定的用户登录会话。

**提示**

建议的所有者数量实际上取决于您的部署。如果没有小心，如果在节点停机时用户注销，则一个所有者就足够了，您将避免复制。

**提示**

通常最好将您的环境配置为使用带粘性会话的 loadbalancer。作为提供特定请求的 Red Hat Single Sign-On 服务器而言，这对性能很有用，通常是分布式缓存中数据的所有者，因此可以在本地查找数据。详情请查看 [第 9.4 节“粘性会话”](#)。

**10.3. 禁用缓存**

要禁用 realm 或用户缓存，您必须编辑发行版中的 standalone.xml、standalone-ha.xml 或 domain.xml 文件。此文件的位置取决于您的 [操作模式](#)。首先，配置是什么样子。

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

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

要禁用缓存，将您要禁用的缓存的 enabled 属性设置为 false。您必须重新引导服务器才能使此更改生效。

#### 10.4. 在运行时清除缓存

要清除 realm 或 user 缓存，请转至 Red Hat Single Sign-On admin console Realm Settings → Cache Config 页面。在此页面中，您可以清除域缓存、用户缓存或外部公钥的缓存。



**注意**

**将为所有域清除缓存！**

## 第 11 章 RED HAT SINGLE SIGN-ON OPERATOR

**注意**

*Red Hat Single Sign-On Operator 只是一个技术预览，不被支持。*

*Red Hat Single Sign-On Operator 在 Openshift 中自动化 Red Hat Single Sign-On 管理。您可以使用此 Operator 创建自定义资源(CR)，以自动化管理任务。例如，您可以创建自定义资源来执行这些任务，而不是在 Red Hat Single Sign-On 管理控制台中创建客户端或用户。自定义资源是一个 YAML 文件，用于定义管理任务的参数。*

您可以创建自定义资源来执行以下任务：

- [安装 Red Hat Single Sign-On](#)
- [创建域](#)
- [创建客户端](#)
- [创建用户](#)
- [连接到外部数据库](#)
- [调度数据库备份](#)
- [安装扩展和主题](#)



## 注意

为域、客户端和用户创建自定义资源后，您可以使用 Red Hat Single Sign-On admin 控制台或使用 `oc` 命令作为自定义资源来管理它们。但是，您不能同时使用这两种方法，因为 Operator 会对您修改的自定义资源执行一种同步。例如，如果您修改了 `realm` 自定义资源，则更改将显示在管理控制台中。但是，如果您使用 admin 控制台修改域，这些更改对自定义资源没有影响。

通过在集群中安装 [Red Hat Single Sign-On Operator](#) 开始使用 Operator。

## 11.1. 在集群上安装 RED HAT SINGLE SIGN-ON OPERATOR

要安装 Red Hat Single Sign-On Operator，您可以使用：

- [Operator Lifecycle Manager \(OLM\)](#)
- [命令行安装](#)

### 11.1.1. 使用 Operator Lifecycle Manager 安装

#### 先决条件

- 有 `cluster-admin` 权限或管理员授予的等效权限级别。

#### 流程

在 OpenShift 4.4 集群中执行此步骤。

1. 打开 OpenShift Container Platform Web 控制台。
2. 在左侧列中，点 Operators OperatorHub。
3. 搜索 Red Hat Single Sign-On Operator。

[OpenShift 中的 OperatorHub 标签页](#)

Project: default ▾

## OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. Operators on your clusters to provide optional add-ons and shared services to your developers. , providing a self-service experience.

All Items

Security

single sign-on operator

Custom

**Red Hat**

**Red Hat Single Sign-On Operator**  
provided by Red Hat

An Operator for installing and managing Red Hat Single Sign-On


Security

4.

点 *Red Hat Single Sign-On Operator* 图标。

此时会打开 *Install* 页面。

*OpenShift* 上的 *Operator Install* 页面



## Red Hat Single Sign-On Operator

7.4.0 provided by Red Hat

[Install](#)

---

**Operator Version**  
7.4.0

**Capability Level**

- Basic Install
- Seamless Upgrades
- Full Lifecycle
- Deep Insights
- Auto Pilot

**Provider Type**  
Custom

A Kubernetes Operator based on the Operator SDK for installing and managing Red Hat Single Sign-On.

Red Hat Single Sign-On lets you add authentication to applications and secure services with minimum fuss. No need to deal with storing users or authenticating users. It's all available out of the box.

The operator can deploy and manage Keycloak instances on Kubernetes and OpenShift. The following features are supported:

- Install Keycloak to a namespace
- Import Keycloak Realms
- Import Keycloak Clients
- Import Keycloak Users
- Create scheduled backups of the database

5.

**点 *Install*。**

6.

**选择一个命名空间并点 *Subscribe*。**

**OpenShift 中的命名空间选择**

**Installation Mode \***

All namespaces on the cluster (default)

This mode is not supported by this Operator

A specific namespace on the cluster

Operator will be available in a single namespace only.

**Installed Namespace \*****Update Channel \***

alpha

**Approval Strategy \***

Automatic

Manual

Subscribe

Cancel

*Operator 开始安装。*

**其他资源**

- 当 Operator 安装完成后，您已准备好创建第一个自定义资源。请参阅 [使用自定义资源的 Red Hat Single Sign-On 安装](#)。
- 如需有关 OpenShift Operator 的更多信息，请参阅 [OpenShift Operator 指南](#)。

**11.1.2. 从命令行安装**

您可以从命令行安装 **Red Hat Single Sign-On Operator**。

### 先决条件

- 有 `cluster-admin` 权限或管理员授予的等效权限级别。

### 流程

1. 从此位置获取要安装的软件：[Github repo](#)。

2. 安装所有所需的自定义资源定义：

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

3. 创建新命名空间（或重复使用现有命名空间），如 `myproject` 命名空间：

```
$ oc create namespace myproject
```

4. 为 `Operator` 部署角色、角色绑定和服务帐户：

```
$ 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. 部署 `Operator`：

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

6. 确认 `Operator` 正在运行：

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

### 其他资源

- 当 `Operator` 安装完成后，您已准备好创建第一个自定义资源。请参阅 [使用自定义资源的](#)



## Red Hat Single Sign-On 安装。

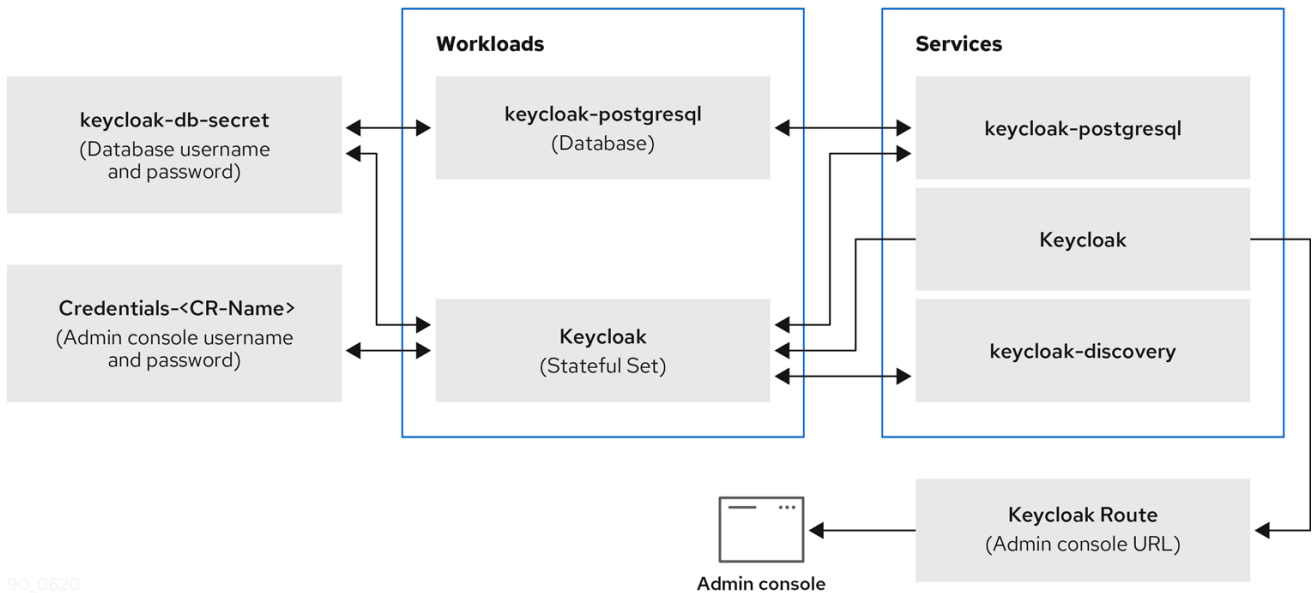
- 如需有关 OpenShift Operator 的更多信息，请参阅 [OpenShift Operator 指南](#)。

### 11.2. 使用自定义资源进行 RED HAT SINGLE SIGN-ON 安装

您可以通过创建 Keycloak 自定义资源来使用 Operator 自动安装 Red Hat Single Sign-On。当您使用自定义资源安装 Red Hat Single Sign-On 时，您可以创建此处描述的组件和服务，并在下图中所示。

- **keycloak-db-secret** - 存储数据库用户名、密码和外部地址等属性（如果您连接到外部数据库）
- **credentials-<CR-Name >** - 用于登录到 Red Hat Single Sign-On 管理控制台的 Admin 用户名和密码(& lt;CR-Name > 基于 Keycloak 自定义资源名称)
- **Keycloak** - Keycloak 部署规格，该规格作为具有高可用性支持的 StatefulSet 实现
- **keycloak-postgresql** - 启动 PostgreSQL 数据库安装
- **keycloak-discovery Service** - 执行 JDBC\_PING 发现
- **Keycloak Service** - 通过 HTTPS 连接到 Red Hat Single Sign-On（不支持 HTTP）
- **keycloak-postgresql Service** - 如果使用的数据库实例，请连接内部和外部
- **Keycloak Route** - 从 OpenShift 访问 Red Hat Single Sign-On 管理控制台的 URL

Operator 组件和服务如何交互



90\_0620

### 11.2.1. Keycloak 自定义资源

**Keycloak 自定义资源**是一个 YAML 文件，用于定义安装的参数。此文件包含三个属性：

- **实例** - 控制在高可用性模式下运行的实例数量。
- **externalAccess** - 如果启用为 `True`，Operator 会为 Red Hat Single Sign-On 集群创建一个路由。
- **ExternalDatabase** - 仅在您要连接外部托管数据库时才应用。该主题包括在本指南的 [外部数据库](#) 部分。

#### Keycloak 自定义资源的 YAML 文件示例

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



### 注意

您可以更新 YAML 文件，且 Red Hat Single Sign-On admin 控制台中显示的更改，但对管理控制台的更改不会更新自定义资源。

## 11.2.2. 在 OpenShift 中创建 Keycloak 自定义资源

在 OpenShift 上，您可以使用自定义资源来创建路由，这是管理控制台的 URL，并查找包含管理控制台的用户名和密码的机密。

### 先决条件

- 您有一个用于此自定义资源的 YAML 文件。
- 有 `cluster-admin` 权限或管理员授予的等效权限级别。

### 流程

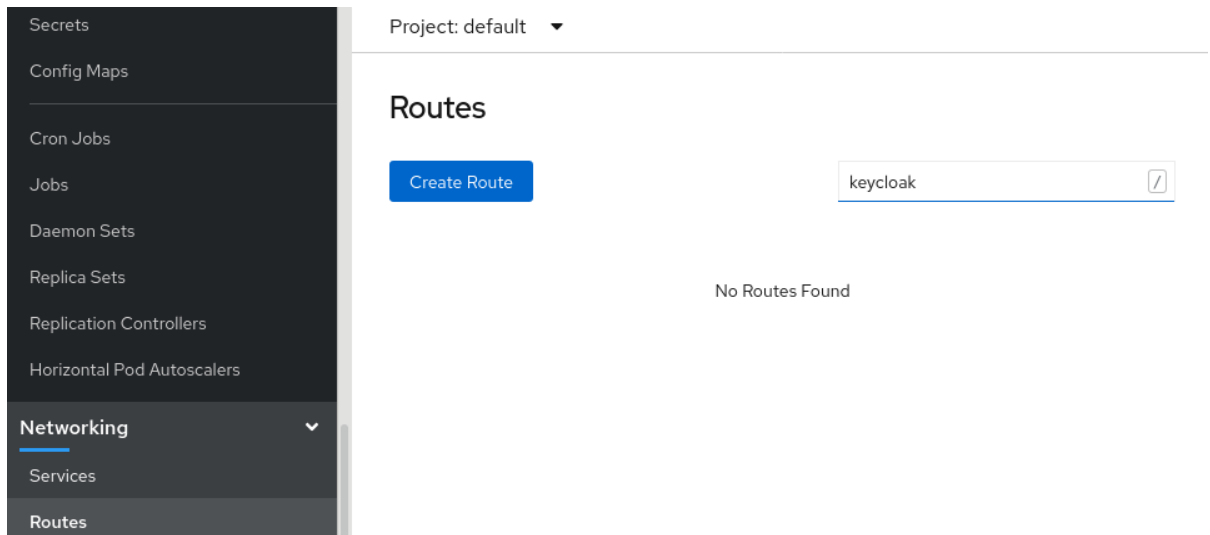
1. 使用 YAML 文件创建一个路由：`oc create -f <filename>.yaml -n <namespace>`。例如：

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

在 OpenShift 中创建路由。

2. 登录 OpenShift Web 控制台。
3. 选择 **Networking, Routes and search for Keycloak**。

OpenShift Web 控制台中的路由屏幕



4.

在带有 Keycloak 路由的屏幕上，点 Location 下的 URL。

此时会出现 Red Hat Single Sign-On admin 控制台登录屏幕。

#### 管理控制台登录屏幕

Username or email

Password

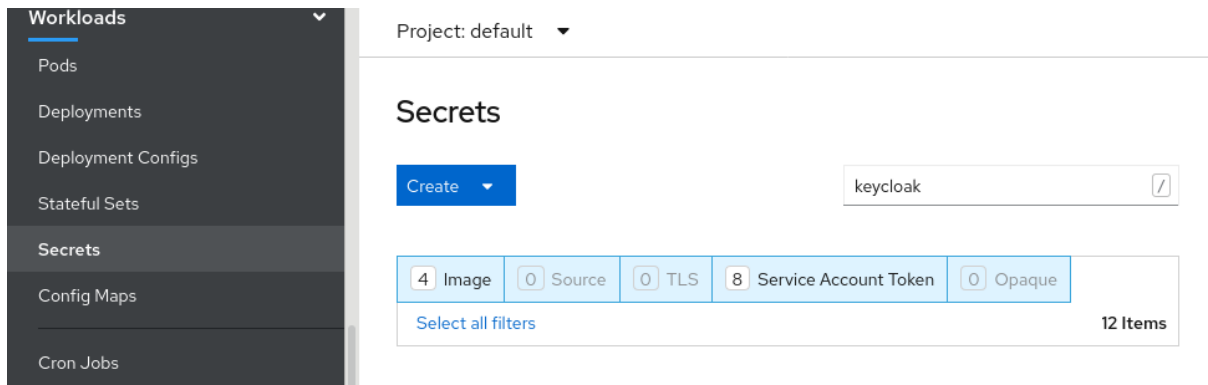
Remember me

Log In

5.

在 OpenShift Web 控制台中找到管理控制台的用户名和密码；在 Workloads 下，单击 Secrets 并搜索 Keycloak。

#### OpenShift Web 控制台中的 secret 屏幕



6.

在管理控制台登录屏幕中输入用户名和密码。

### 管理控制台登录屏幕

Username or email

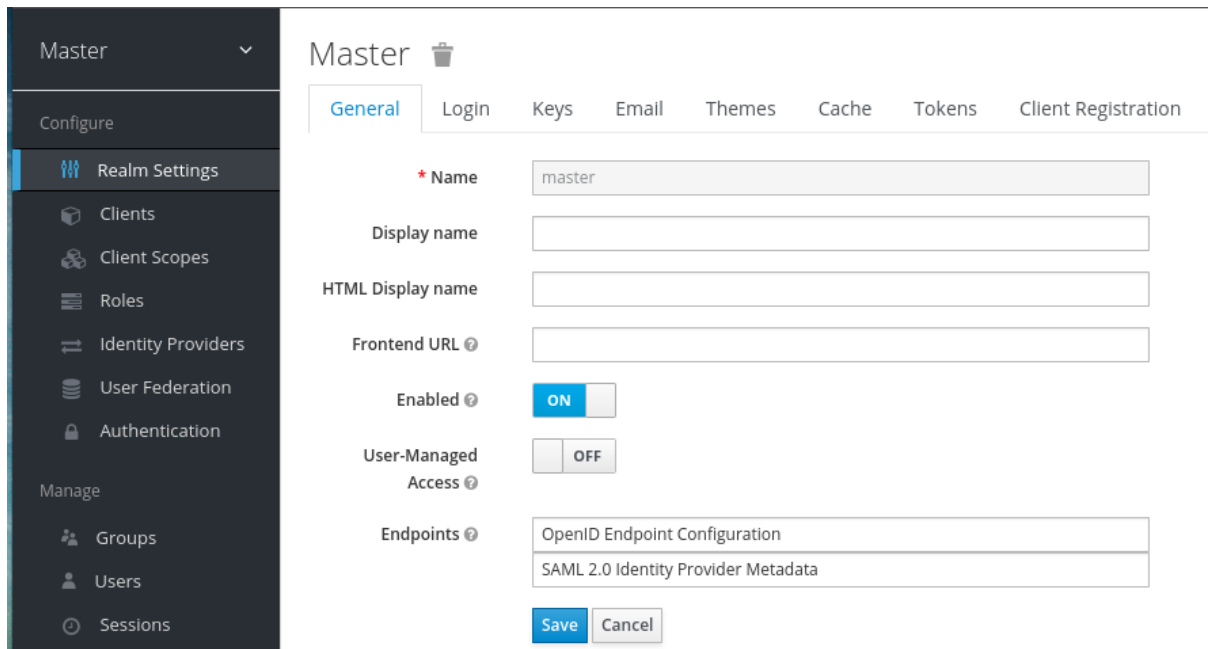
Password

Remember me

Log In

现在，您已登录到由 Keycloak 自定义资源安装的 Red Hat Single Sign-On 实例。您已准备好为域、客户端和用户创建自定义资源。

### Red Hat Single Sign-On master realm



7.

检查自定义资源的状态：

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

结果

*Operator* 处理自定义资源后，使用以下命令查看状态：

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

**Keycloak 自定义资源状态**

```
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:

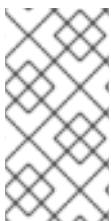
```

### 其他资源

- 安装 Red Hat Single Sign-On 后，就可以 [创建一个 realm 自定义资源](#)。
- 如果您有外部数据库，您可以修改 Keycloak 自定义资源来支持它。请[参阅连接到外部数据库](#)。

### 11.3. 创建 REALM 自定义资源

您可以使用 Operator 在 Red Hat Single Sign-On 中创建由自定义资源定义的域。您可以在 YAML 文件中定义 realm 自定义资源的属性。



#### 注意

您可以更新 YAML 文件并出现在 Red Hat Single Sign-On admin 控制台中，但对管理控制台的更改不会更新自定义资源。

### Realm 自定义资源的 YAML 文件示例

```
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
```

### 先决条件

- 您有一个用于此自定义资源的 YAML 文件。
- 在 YAML 文件中，instanceSelector 下的 app 与 Keycloak 自定义资源的标签匹配。匹配这些值可确保您在 Red Hat Single Sign-On 右侧实例中创建域。
- 有 cluster-admin 权限或管理员授予的等效权限级别。

### 流程

1. 在您创建的 YAML 文件中使用以下命令：`oc create -f <realm-name>.yaml`。例如：

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

2. 登录到 Red Hat Single Sign-On 相关实例的管理控制台。
3. 单击 **Select Realm**，再找到您创建的域。



新域将打开。

### 管理控制台 master realm

The screenshot shows the Keycloak management console interface. On the left is a dark sidebar menu with options: Test (dropdown), Configure, Realm Settings (selected), Clients, Client Scopes, Roles, Identity, Providers, User Federation, and Authentication. The main content area is titled 'Test' and has a trash icon. Below the title are tabs for 'General' (selected), Login, Keys, Email, Themes, Cache, Tokens, and Client Registration. The 'General' tab contains the following configuration fields:

- Name:** test (text input)
- Display name:** (text input)
- HTML Display name:** (text input)
- Frontend URL:** (text input)
- Enabled:** ON (toggle switch)
- User-Managed Access:** OFF (toggle switch)

### 结果

Operator 处理自定义资源后，使用以下命令查看状态：

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

realm 自定义资源状态

```
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>

```

## 其他资源

- 当域创建完成后，您已准备好 [创建客户端自定义资源](#)。

### 11.4. 创建客户端自定义资源

您可以使用 Operator 在 Red Hat Single Sign-On 中创建由自定义资源定义的客户端。您可以在 YAML 文件中定义域的属性。



#### 注意

您可以更新 YAML 文件并出现在 Red Hat Single Sign-On admin 控制台中，但对管理控制台的更改不会更新自定义资源。

#### 客户端自定义资源的 YAML 文件示例

```

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

```

## 先决条件

- 您有一个用于此自定义资源的 **YAML** 文件。
- 有 **cluster-admin** 权限或管理员授予的等效权限级别。

## 流程

1. 在您创建的 **YAML** 文件中使用以下命令：`oc create -f <client-name>.yaml`。例如：

```

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

```

2. 登录到 Red Hat Single Sign-On 相关实例的 Red Hat Single Sign-On 管理控制台。
3. 点 **Clients**。

新客户端会出现在客户端列表中。

Client ID

Lookup ?

Client ID	Enabled	Base URL	Actions		
account	True	<a href="http://localhost:8080/auth/realms/master/account/">http://localhost:8080/auth/realms/master/account/</a>	Edit	Export	Delete
account-console	True	<a href="http://localhost:8080/auth/realms/master/account/">http://localhost:8080/auth/realms/master/account/</a>	Edit	Export	Delete
admin-cli	True	Not defined	Edit	Export	Delete
broker	True	Not defined	Edit	Export	Delete
master-realm	True	Not defined	Edit	Export	Delete
security-admin-console	True	<a href="http://localhost:8080/auth/admin/master/console/">http://localhost:8080/auth/admin/master/console/</a>	Edit	Export	Delete

## 结果

创建客户端后，Operator 会使用以下命名模式创建一个包含客户端 ID 和客户端的 secret 的 Secret：`keycloak-client-secret-<custom resource name>`。例如：

### 客户端的 Secret

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

Operator 处理自定义资源后，使用以下命令查看状态：

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

### 客户端自定义资源状态

```
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:
```

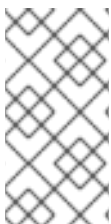
**Secret:**  
keycloak-client-secret-client-secret  
**Events:** <none>

## 其他资源

- 客户端创建完成后，您已准备好 [创建用户自定义资源](#)。

### 11.5. 创建用户自定义资源

您可以使用 Operator 在 Red Hat Single Sign-On 中创建由自定义资源定义的用户。您可以在 YAML 文件中定义用户自定义资源的属性。



#### 注意

您可以更新除 YAML 文件中除密码以外的属性，并在 Red Hat Single Sign-On admin 控制台中显示更改，但对管理控制台的更改不会更新自定义资源。

#### 用户自定义资源的 YAML 文件示例

```
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
```

### 先决条件

- 您有一个用于此自定义资源的 YAML 文件。
- `realmSelector` 与现有 `realm` 自定义资源的标签匹配。
- 有 `cluster-admin` 权限或管理员授予的等效权限级别。

### 流程

1. 在您创建的 YAML 文件中使用以下命令：`oc create -f <user_cr>.yaml`。例如：

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

2. 登录到 Red Hat Single Sign-On 相关实例的管理控制台。
3. 点 **Users**。
4. 搜索您在 YAML 文件中定义的用户。

您可能需要切换到其他域来查找用户。

ID	Username	Email	Last Name	First Name	Actions		
d6b907cd-e...	leiazuki	lazuki@wes...	Azuki	Lei	Edit	Impersonate	Delete
2a26e1b2-8...	nemethjabaz	njabaz@we...	Jabaz	Nemeth	Edit	Impersonate	Delete
7c1f0c04-0f...	realm_user	user@exam...	Doe	John	Edit	Impersonate	Delete

## 结果

创建用户后，Operator 会创建一个 Secret，其中包含使用以下命名模式的用户名和密码：  
`credential-<realm name>-<username>-<namespace>`。例如：

### KeycloakUser Secret

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

Operator 处理自定义资源后，使用以下命令查看状态：

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

用户自定义资源状态

```
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>

```

### 其他资源

- 如果您有外部数据库，您可以修改 Keycloak 自定义资源来支持它。请参阅[连接到外部数据库](#)。
- 要使用自定义资源备份数据库，请参阅[调度数据库备份](#)。

## 11.6. 连接到外部数据库

您可以通过修改 Keycloak 自定义资源并创建 `keycloak-db-secret` YAML 文件来使用 Operator 连接到外部 PostgreSQL 数据库。请注意，值采用 Base64 编码。

### keycloak-db-secret 的 YAML 文件示例

```

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

```

以下属性设置数据库的主机名或 IP 地址和端口。

- **POSTGRES\_EXTERNAL\_ADDRESS** - 一个 IP 地址或外部数据库的主机名。
- **POSTGRES\_EXTERNAL\_PORT** - (可选) 一个数据库端口。

其他属性的工作方式与托管或外部数据库相同。按如下所示设置它们：

- **POSTGRES\_DATABASE** - 要使用的数据库名称。
- **POSTGRES\_HOST** - 用于与数据库通信的服务名称。通常 `keycloak-postgresql`。
- **POSTGRES\_USERNAME** - 数据库用户名
- **POSTGRES\_PASSWORD** - 数据库密码
- **POSTGRES\_SUPERUSER** - 表示备份是否应该以超级用户身份运行。通常为 `true`。

Keycloak 自定义资源需要更新才能启用外部数据库支持。

支持外部数据库的 Keycloak 自定义资源的 YAML 文件示例

```

apiVersion: keycloak.org/v1alpha1
kind: Keycloak
metadata:
  labels:

```

```

app: sso
name: example-keycloak
namespace: keycloak
spec:
  externalDatabase:
    enabled: true
  instances: 1

```

### 先决条件

- 您有一个用于 `keycloak-db-secret` 的 YAML 文件。
- 您已修改了 Keycloak 自定义资源，将 `externalDatabase` 设置为 `true`。
- 有 `cluster-admin` 权限或管理员授予的等效权限级别。

### 流程

1. 找到 PostgreSQL 数据库的 `secret` : `oc get secret <secret_for_db> -o yaml`。例如 :

```

$ 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` 是 Base64 格式。

2. 解码 `secret` 的值 : `echo "<encoded_secret>" | base64 -decode`。例如 :

```

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

```

3. 确认解码的值与数据库的 IP 地址匹配 :

```

$ 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.

确认 `keycloak-postgresql` 出现在正在运行的服务列表中：

```

$ 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` 服务将请求发送到后端的一组 IP 地址。这些 IP 地址称为端点。

5.

查看 `keycloak-postgresql` 服务使用的端点，以确认它们为您的数据库使用 IP 地址：

```

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

```

6.

确认 Red Hat Single Sign-On 使用外部数据库运行。本例显示一切都在运行：

```

$ 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. 调度数据库备份

您可以使用 Operator 调度自定义资源定义的数据库的自动备份。自定义资源会触发备份作业，并报告返回其状态。

您可以使用 Operator 创建对本地持久性卷执行一次性备份的备份作业。

### Backup 自定义资源的 YAML 文件示例

```

apiVersion: keycloak.org/v1alpha1
kind: KeycloakBackup
metadata:

```

**name: test-backup**

### 先决条件

- 您有一个用于此自定义资源的 YAML 文件。
- 您有一个带有 `claimRef` 的 `PersistentVolume`，以便仅为 Red Hat Single Sign-On Operator 创建的 `PersistentVolumeClaim` 保留它。

### 流程

1.

创建备份作业：`oc create -f <backup_crname>`。例如：

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

Operator 使用以下命名方案创建一个 `PersistentVolumeClaim`：`Keycloak-backup-<CR-name >`。

2.

查看卷列表：

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

3.

查看备份作业列表：

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

4.

查看执行的备份作业列表：

```
$ 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.

查看已完成的备份作业的日志：

```

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

```

### 其他资源

- 有关持久性卷的详情，[请参阅了解持久性存储](#)。

## 11.8. 安装扩展和主题

您可以使用操作器安装扩展，并满足公司或机构所需的扩展。扩展或主题可以是 Red Hat Single Sign-On 可消耗的任何内容。例如，您可以添加指标扩展。您可以将扩展或主题添加到 Keycloak 自定义资源中。

### Keycloak 自定义资源的 YAML 文件示例

```

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

```

### 先决条件

- 您有一个 Keycloak 自定义资源的 YAML 文件。
- 有 `cluster-admin` 权限或管理员授予的等效权限级别。

### 流程

1. 编辑 Keycloak 自定义资源的 YAML 文件：`oc edit <CR-name>`
2. 在 `instances` 行后，添加名为 `extensions:` 的行。
3. 添加 URL 到 JAR 文件，用于自定义扩展或主题。
4. 保存该文件。

**Operator** 下载扩展或主题并安装它。

## 11.9. 管理自定义资源的命令选项

创建自定义请求后，您可以使用 `oc` 命令编辑或删除它。

- 要编辑自定义请求，请使用以下命令：`oc edit <CR-name>`
- 要删除自定义请求，请使用以下命令：`oc delete <CR-name>`

例如，要编辑名为 `test-realm` 的 `realm` 自定义请求，请使用以下命令：

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

此时会打开一个窗口，您可以在其中进行更改。



### 注意

您可以更新 YAML 文件并出现在 Red Hat Single Sign-On admin 控制台中，但对管理控制台的更改不会更新自定义资源。