Red Hat Single Sign-On 7.6

服务器安装和配置指南

适用于 Red Hat Single Sign-On 7.6

Last Updated: 2024-06-05

Red Hat Single Sign-On 7.6 服务器安装和配置指南

适用于 Red Hat Single Sign-On 7.6

法律通告

摘要

本指南包含安装和配置红帽单点登录 7.6 的信息

目录

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

本指南的目的是浏览在首次引导 Red Hat Single Sign-On 服务器之前完成的步骤。如果您只想测试驱动器 Red Hat Single Sign-On，它明显耗尽了其自身的嵌入式和仅本地数据库。对于将要在生产环境中运行的实际部署，您需要决定如何在运行时（standalone 或 domain 模式）管理服务器配置，为红帽单点登录存储配置共享数据库，设置加密和 HTTPS，最后设置红帽单点登录以在集群中运行。本指南介绍了部署服务器前必须执行的任何预引导决策和设置的每个方面。

特别需要注意的是，红帽单点登录来源于 JBoss EAP 应用服务器。配置红帽单点登录与 JBoss EAP 配置元素相关的许多方面。通常，如果您想深入了解更多详情，此指南会将您定向到手册之外的文档。

1.1. 建议额外的外部文档

Red Hat Single Sign-On 基于 JBoss EAP 应用服务器及其子项目（用于缓存）和 Hibernate（用于持久性）。本指南仅涵盖基础架构级别配置的基础知识。强烈建议您仔细阅读 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 安装软件

您可以通过下载 ZIP 文件并解压缩它，或使用 RPM 来安装 Red Hat Single Sign-On。本章论述了系统要求和目录结构。

2.1. 安装先决条件

安装 Red Hat Single Sign-On 服务器需要满足这些先决条件：

- Java 8 JRE 或 Java 11 JRE
- 支持您选择的 Java 版本的操作系统。请参阅 支持的配置。
- zip 或 gzip 和 tar
- 至少 512M RAM
- 至少 1G 磁盘空间
- 共享的外部数据库，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，在机器上进行网络多播支持。红帽单点登录可在不多播的情况下进行集群，但这需要进行大量配置更改。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出现 Red Hat Single Sign-On 挂起，除非您的安全策略需要 **/dev/random** 使用。要在 Oracle JDK 8 和 OpenJDK 8 上实现这一点，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 7.6 服务器，然后安装 7.6.9 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.rhsso
2. 下载红帽单点登录 7.6 服务器。
3. 使用适当的解压缩程序（如 **unzip**、tar 或 Expand-Archive）解压缩 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 7.6.9 服务器补丁。
   将下载的 ZIP 文件放在您选择的目录中。
7. 进入 Red Hat Single Sign-On 服务器的根目录。
8. 启动 JBoss EAP 命令行界面。

Linux/Unix

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

Windows

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

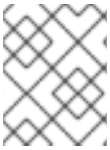
9. 应用补丁。

```
$ patch apply <path-to-zip>/rh-sso-7.6.9-patch.zip
```

其他资源

有关应用补丁的详情，请参阅 补丁 ZIP/Installer 安装。

2.3. 从 RPM 安装 RH-SSO



注意

使用 Red Hat Enterprise Linux 7 和 8 时，术语频道被一个术语仓库替代。这些说明中，仅使用术语库。

您必须订阅 JBoss EAP 7.4 和 RH-SSO 7.6 存储库，然后才能从 RPM 安装 RH-SSO。

注意

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

2.3.1. 订阅 JBoss EAP 7.4 软件仓库

先决条件

1. 确定使用 Red Hat Subscription Manager 将 Red Hat Enterprise Linux 系统注册到您的帐户。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，必须先退出该存储库。

对于 Red Hat Enterprise Linux 6：使用 Red Hat Subscription Manager，使用以下命令订阅 JBoss EAP 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=jb-eap-7.4-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 7.4 存储库：

```
subscription-manager repos --enable=jb-eap-7.4-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 7.6 存储库并安装 RH-SSO 7.6

先决条件

1. 确定使用 Red Hat Subscription Manager 将 Red Hat Enterprise Linux 系统注册到您的帐户。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 7.4 存储库。如需更多信息，请参阅订阅 JBoss EAP 7.4 存储库。

流程

1. 对于 Red Hat Enterprise Linux 6：使用 Red Hat Subscription Manager，使用以下命令订阅 RH-SSO 7.6 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-sso-7.6-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 7.6 存储库：

```
subscription-manager repos --enable=rh-sso-7.6-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 7.6 存储库安装 RH-SSO 7.6：

```
yum groupinstall rh-sso7
```

4. 对于 Red Hat Enterprise Linux 8：使用以下命令从您订阅的 RH-SSO 7.6 存储库安装 RH-SSO 7.6：

```
dnf groupinstall rh-sso7
```

您的安装已完成。RPM 安装的默认 RH-SSO_HOME 路径为 /opt/rh/rh-sso7/root/usr/share/keycloak。

其他资源

有关为 Red Hat Single Sign-On 安装 7.6.9 补丁的详情，请参考 RPM 补丁。

2.4. 重要目录

以下是服务器分发中的一些重要目录：

bin/

它包含引导服务器或对服务器执行其他一些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/

这些都是服务器使用的所有 Java 库。

standalone/

这在 以独立 模式运行 Red Hat Single Sign-On 时包含配置文件和工作目录。

standalone/deployments/

如果您要向红帽单点登录编写扩展，您可以在此处放置扩展。有关此问题的更多信息，请参阅 服务器开发人员指南。

themes/

该目录包含用于显示服务器显示的任何 UI 屏幕的所有 html、风格表、Java 文件和镜像。在这里，您可以修改现有主题或自行创建。有关此问题的更多信息，请参阅 服务器开发人员指南。

第 3 章 使用操作模式

在生产环境中部署 Red Hat Single Sign-On 之前，您需要决定要使用的操作模式。

- 您将在集群中运行 Red Hat Single Sign-On？
- 您是否希望集中管理服务器配置？

您选择的运营模式会影响配置数据库、配置缓存甚至如何引导服务器。

提示

红帽单点登录构建于 JBoss EAP 应用服务器之上。本指南将仅介绍特定模式下部署的基本知识。如果您需要有关此信息的具体信息，最好是 *JBoss EAP 配置指南*。

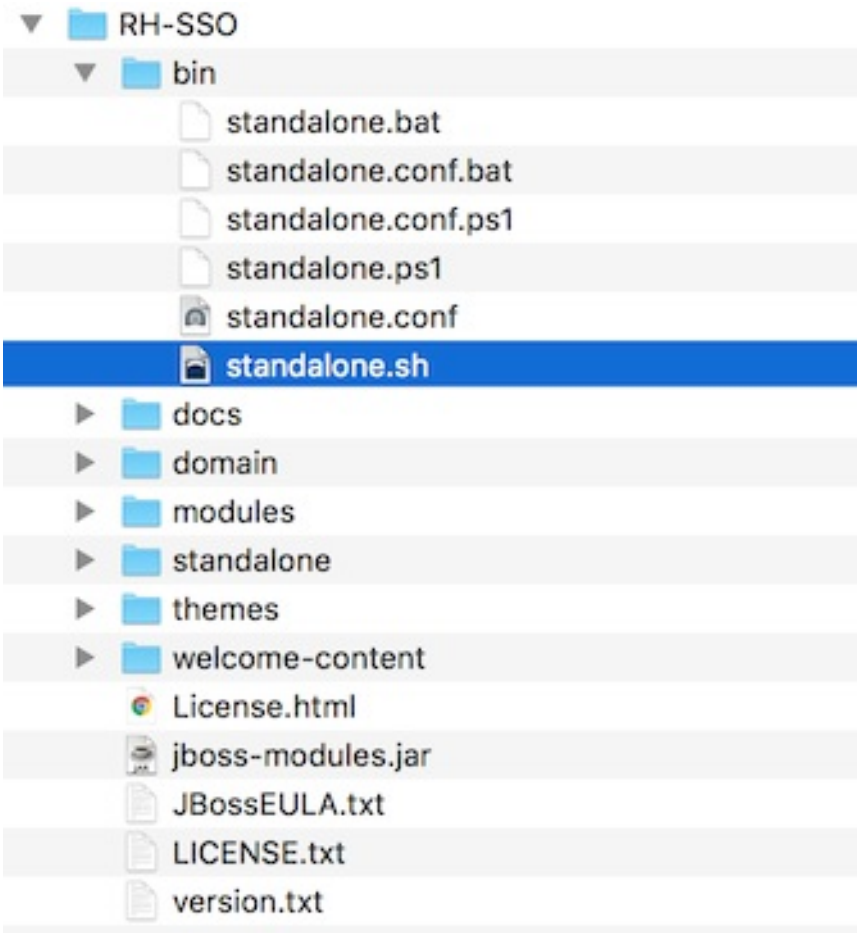
3.1. 使用独立模式

独立操作模式仅在您要运行一个且只有一个红帽单点登录服务器实例时很有用。它不适用于集群部署，所有缓存都不是分布式且仅本地的缓存。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。如果您的独立模式服务器停机，用户将无法登录。这个模式对使用 Red Hat Single Sign-On 的功能进行测试和 play 非常有用。

3.1.1. 以独立模式引导

在单机模式下运行服务器时，您需要有一个特定的脚本来引导服务器，具体取决于您的操作系统。这些脚本位于服务器分发的 *bin/* 目录中。

独立引导脚本



引导服务器：

Linux/Unix

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

Windows

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

警告

要使用 Java SE 17 以独立模式运行 Red Hat Single Sign-On，应修改配置执行捆绑的脚本 **enable-elytron-se17.cli**。

Linux/Unix

```
$ ./bin/jboss-cli.sh --file=docs/examples/enable-elytron-se17.cli
```

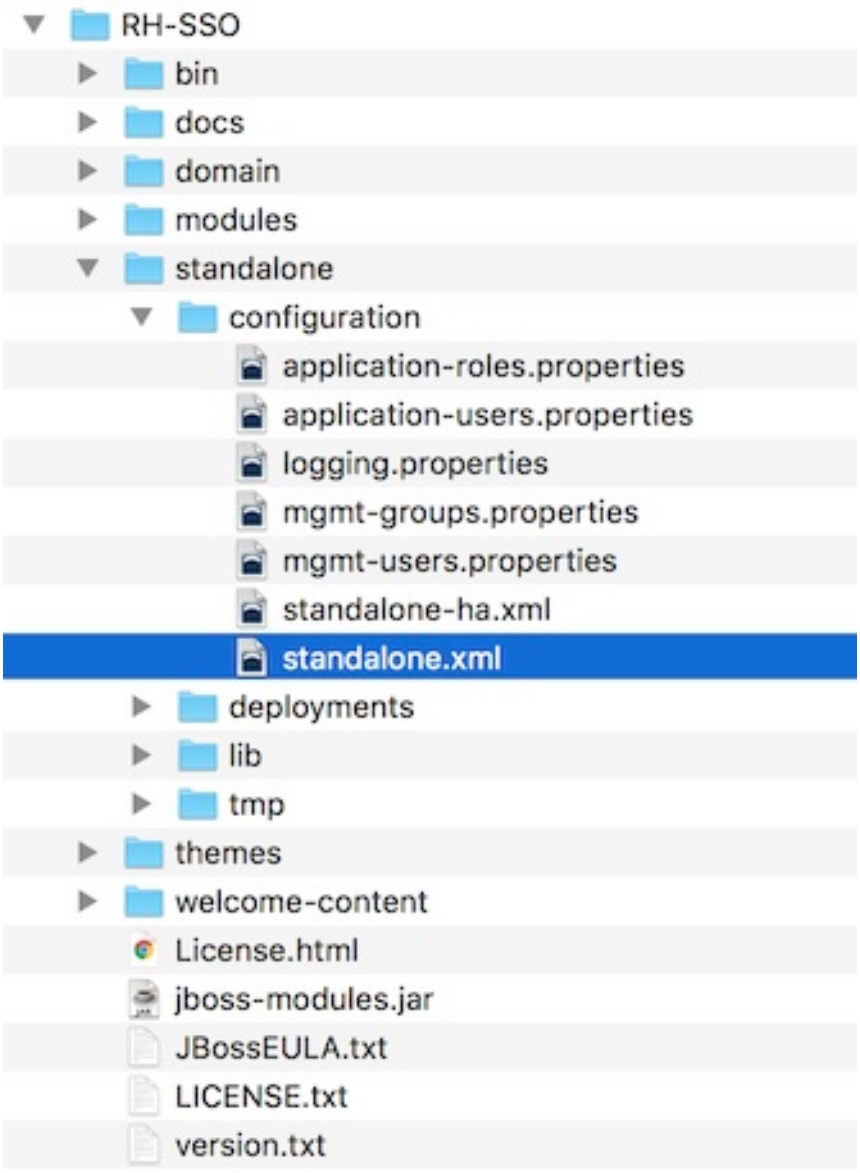
Windows

```
> .\bin\jboss-cli.bat --file=docs\examples\enable-elytron-se17.cli
```

3.1.2. 独立配置

本指南中大部分介绍了如何配置 Red Hat Single Sign-On 的基础架构组件。这些方面在特定于 Red Hat Single Sign-On 是一个衍生的应用程序服务器的配置文件中配置。在单机操作模式中，该文件在 *.../standalone/configuration/standalone.xml* 中有效。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功能。

独立配置文件



警告

您在服务器运行时对此文件进行的任何更改都不会生效，甚至可由服务器覆盖。应使用命令行脚本或 JBoss EAP Web 控制台。如需更多信息，请参阅 *JBoss EAP 配置指南*。

3.2. 使用独立集群模式

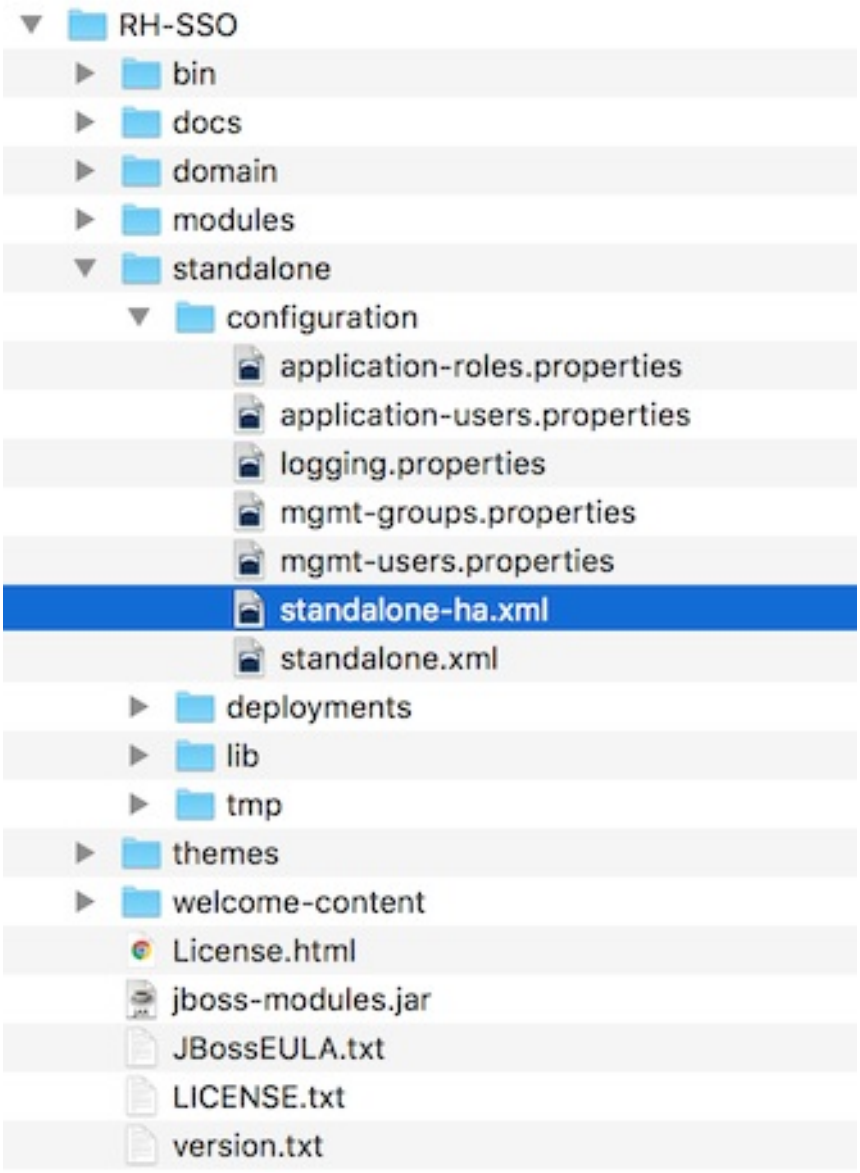
当您要在集群中运行 Red Hat Single Sign-On 时，会应用独立集群操作模式。这个模式要求您在要运行服务器实例的每台机器上具有 Red Hat Single Sign-On 分发的副本。该模式最初部署可能非常容易，但可能会变得非常繁琐。要进行配置更改，您可以修改每台机器上的每个分发版本。对于大型集群来说，这个模式可能会变得耗时，且容易出错。

3.2.1. 独立集群配置

此发行版主要有一个预先配置的应用服务器配置文件，以便在集群中运行。它具有适用于网络、数据库、

缓存和发现的所有特定基础架构设置。此文件位于 *.../standalone/configuration/standalone-ha.xml* 中。此配置中缺少了一些问题。在不配置共享数据库连接的情况下，您无法在集群中运行 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 和数据库部分指导您完成这些内容。???

独立 HA 配置



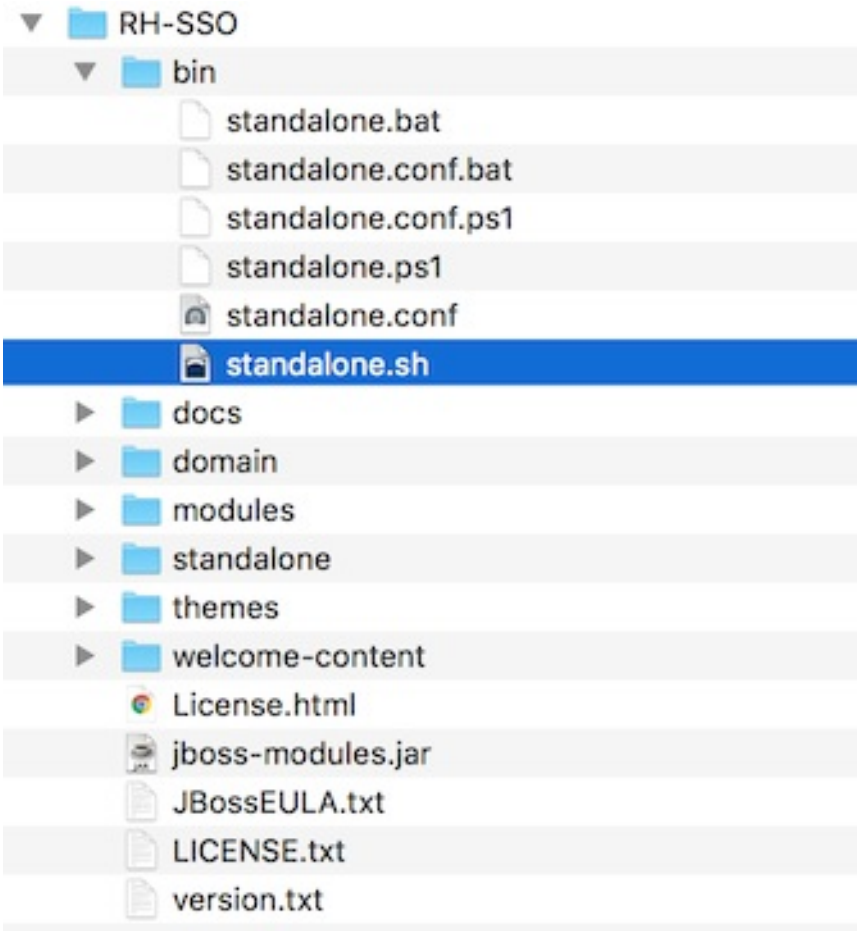
警告

您在服务器运行时对此文件进行的任何更改都不会生效，甚至可由服务器覆盖。应使用命令行脚本或 JBoss EAP Web 控制台。如需更多信息，请参阅 *JBoss EAP 配置指南*。

3.2.2. 以独立集群模式引导

您使用相同的引导脚本启动 Red Hat Single Sign-On，就像您在单机模式中一样。区别在于，您通过额外的标记以指向 HA 配置文件。

独立集群引导脚本



引导服务器：

Linux/Unix

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

Windows

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

警告

要使用 Java SE 17 在独立集群模式下运行 Red Hat Single Sign-On，应修改配置执行捆绑的脚本 **enable-elytron-se17.cli**。

Linux/Unix

```
$ ./bin/jboss-cli.sh --file=docs/examples/enable-elytron-se17.cli -
Dconfig=standalone-ha.xml
```

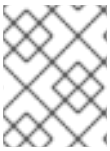
Windows

```
> .\bin\jboss-cli.bat --file=docs\examples\enable-elytron-se17.cli "-
Dconfig=standalone-ha.xml"
```

3.3. 使用域集群模式

域模式是集中管理并发布服务器配置的方法。

以标准模式运行集群可能会很快就会成为随着集群大小增长的聚合。每次您需要进行配置更改时，您都会在集群的每个节点上执行它。域模式通过提供用于存储和发布配置的核心位置来解决这个问题。设置起来可能非常复杂，但最后很值得设置。该功能内置于红帽单点登录的 JBoss EAP 应用服务器中。



注意

本指南将介绍域模式的基本知识。应该在 *JBoss EAP 配置指南* 中获得如何在群集中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

域控制器是一个负责存储、管理和发布集群中每个节点的常规配置的过程。这个过程是集群中的节点获取其配置的中央点。

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。您要将它配置为运行一个或多个服务器实例。域控制器也可与每台计算机上的主机控制器交互来管理集群。为减少正在运行的进程数量，域控制器也充当其所运行计算机上的主机控制器。

域配置集

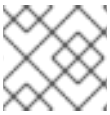
域配置文件是一组指定的配置，供服务器用来引导。域控制器可以定义由不同服务器使用的多个域配置文件。

服务器组

服务器组是服务器的集合。它们作为.您可以将域配置文件分配到服务器组，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，在主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，在集群的每个计算机上启动主机控制器。每个主机控制器部署配置都指定在该计算机上启动多少个红帽单点登录服务器实例。当主机控制器启动时，它会启动任意数量的 Red Hat Single Sign-On 服务器实例，就像被配置为这样做。这

些服务器实例从域控制器拉取其配置。



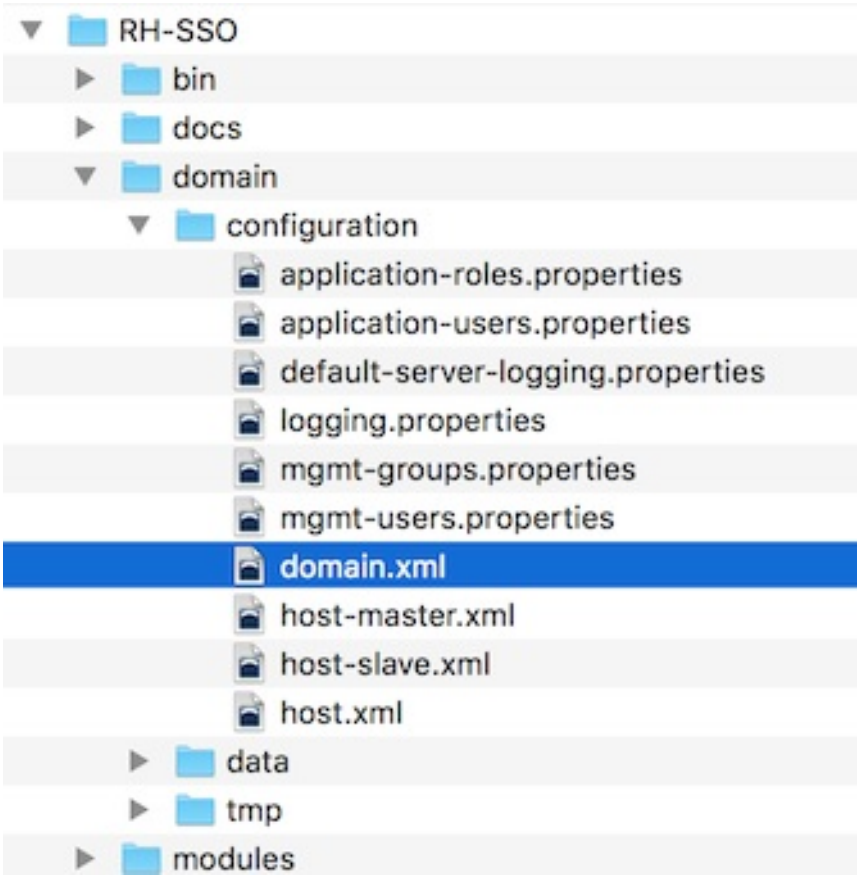
注意

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

3.3.1. 域配置

本指南中的各个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的内容。虽然 standalone 模式使用 *standalone.xml* 文件来配置这些内容，而域模式使用 *.../domain/configuration/domain.xml* 配置文件。这是定义红帽单点登录服务器的域配置文件和服务器组的位置。

domain.xml



警告

您在域控制器运行期间您对此文件所做的任何更改都不会生效，甚至可由服务器覆盖。应使用命令行脚本或 JBoss EAP Web 控制台。如需更多信息，请参阅 *JBoss EAP 配置指南*。

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 块是您将要进行大量配置决策的位置。您可以在此处配置内容，如网络连接、缓存和数据库连接。

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设置。**auth-server-clustered** 配置集是集群的设置。

如果进一步向下滚动，您会看到定义了各种 **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>
```

此配置定义每个红帽单点登录服务器实例打开的各种连接器的默认端口映射。包含 **${...}** 的任何值都是一个值，可通过 **-D** 开关（例如 ）在命令行中覆盖该值。

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

红帽单点登录的服务器组定义位于 **服务器**组 XML 块中。它指定使用的域配置文件（**默**认），以及在主机控制器引导实例时，以及 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>
```

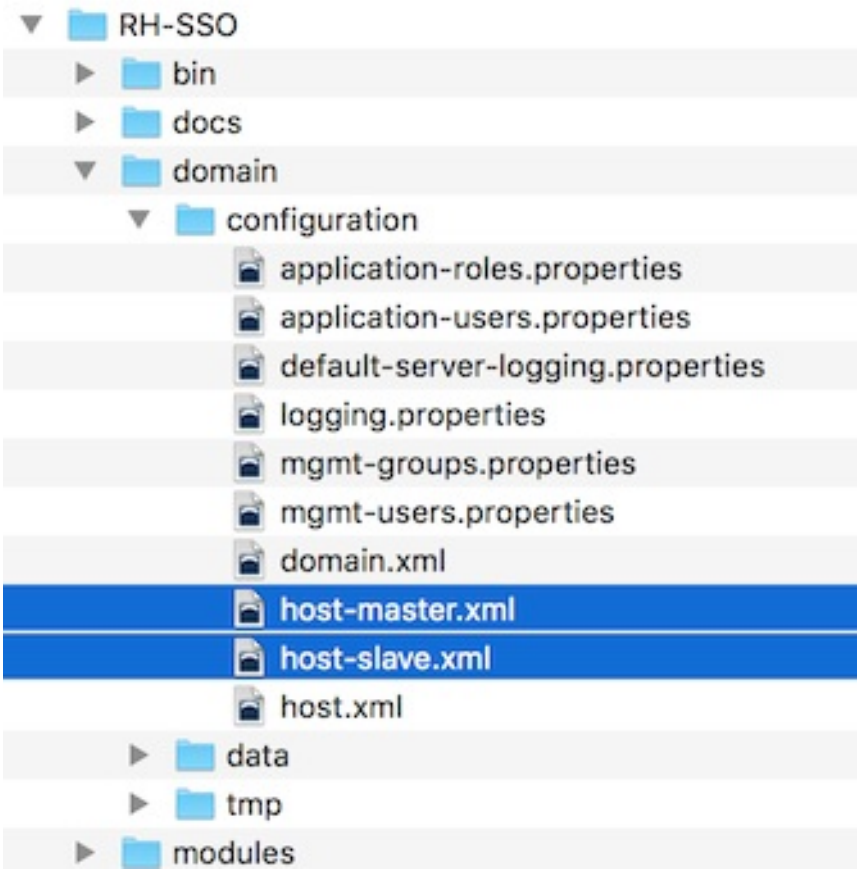
3.3.2. 主机控制器配置

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、负载平衡器和一个红帽单点登录服务器实例。*host-slave.xml* 配置为与域控制器对话并引导一个红帽单点登录服务器实例。

注意

负载均衡器不是必需的服务。它存在，以便您可以在开发机器中轻松测试驱动器集群。虽然可用于生产环境，但如果您有不同的硬件或基于软件的负载均衡器，则可以选择替换它。

主机操作系统配置



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

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

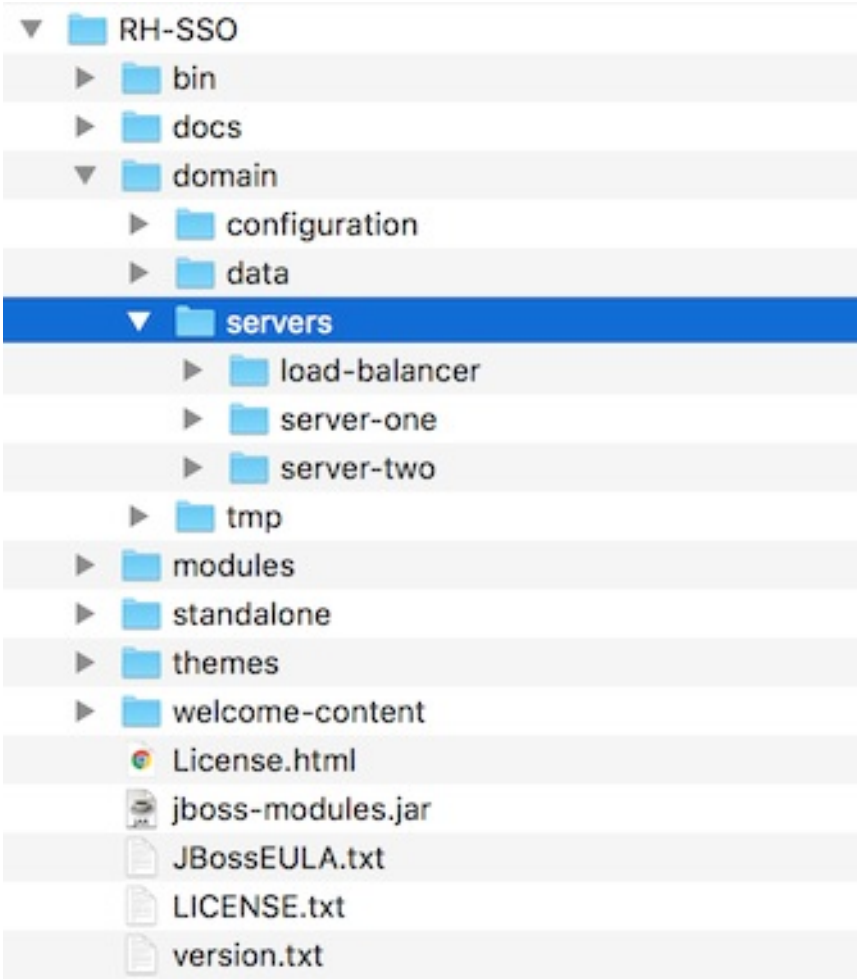
有关此文件的另一个好处在于身份验证服务器实例的声明。它具有 **port-offset** 设置。*domain.xml* **socket-binding-group** 或 server group 中定义的任何网络端口都将添加 **port-offset** 值。在本示例域设置中，我们执行此操作，以便负载均衡器服务器打开的端口与启动的身份验证服务器实例没有冲突。

```
<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红帽单点登录服务器实例会在 *.../domain/servers/{SERVER NAME}* 下创建一个工作目录。可以放置额外的配置，以及服务器实例需要的任何临时、日志或数据文件，也可以创建这些配置。每个服务器目录的结构与任何其他 JBoss EAP 启动服务器类似。

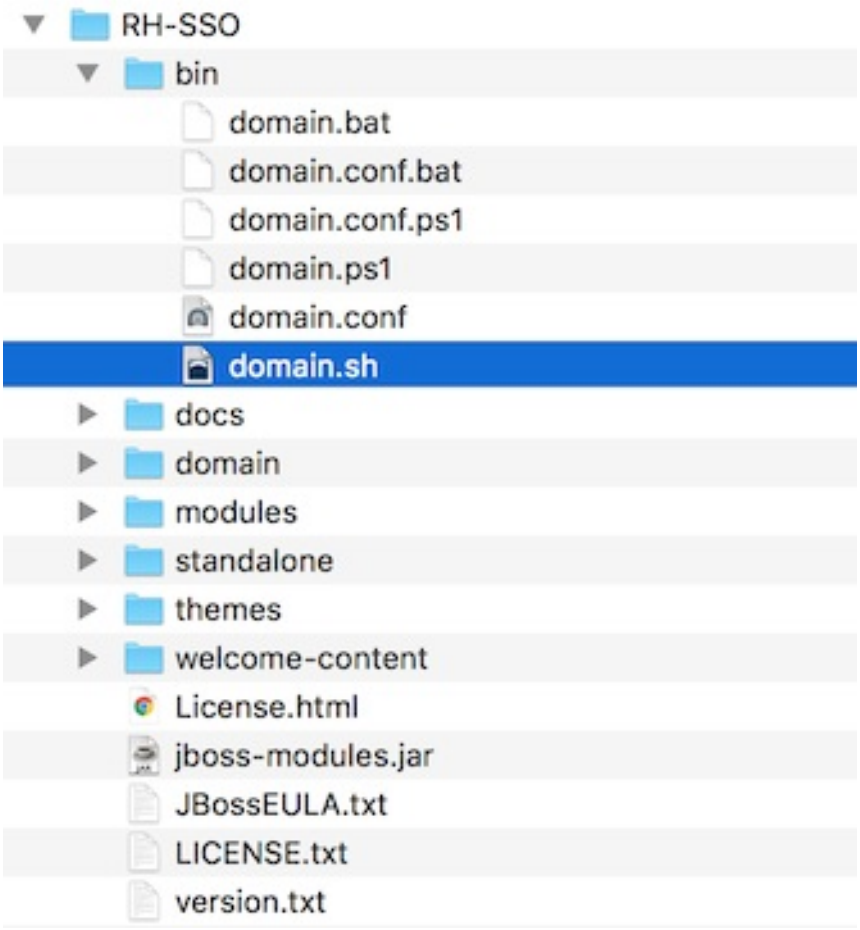
工作目录



3.3.4. 以域集群模式引导

以域模式运行服务器时，您需要运行 特定的脚本来引导服务器，具体取决于您的操作系统。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
```

在运行启动脚本时，您将需要传递要通过 **--host-config** 参数使用的主机控制配置文件。

警告

要使用 Java SE 17 在域模式下运行 Red Hat Single Sign-On，应修改配置执行捆绑的脚本 **enable-keycloak-se17-domain.cli**。

Linux/Unix

```
$ ./bin/jboss-cli.sh --file=docs/examples/enable-keycloak-se17-domain.cli
```

Windows

```
> .\bin\jboss-cli.bat --file=docs\examples\enable-keycloak-se17-domain.cli
```

3.3.5. 使用示例集群域进行测试

您可以使用示例 *domain.xml* 配置测试驱动器集群。这个示例域是在一台机器中运行并引导：

- 一个域控制器
- HTTP 负载均衡器
- 两个红帽单点登录服务器实例

流程

1. 运行 **domain.sh** 脚本两次，以启动两个独立的主机控制器。
   第一个是启动域控制器、HTTP 负载均衡器和一个红帽单点登录身份验证服务器实例的主主机控制器。第二个是从属主机控制器，仅启动身份验证服务器实例。

2. 配置从属主机控制器，以使其安全地与域控制器通信。执行以下步骤：
   如果省略这些步骤，则从属主机无法从域控制器获取集中配置。

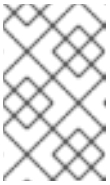
   a. 通过创建一个服务器 admin 用户和一个在主服务器和从卷之间共享的 secret 来设置安全连接。
   运行 **.../bin/add-user.sh** 脚本。

   b. 当脚本询问要添加的用户类型时，选择 **Management User**。
   这个选择会生成一个 secret，供您剪切并粘贴到 *.../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 脚本不会将用户添加到红帽单点登录服务器，而是添加到底层 JBoss 企业应用平台中。这个脚本中使用的和生成的凭证仅用于演示目的。请使用在您的系统中生成的它们。

3. 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>
```

4. 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">
```

5. *运行引导脚本两次，以在一个开发机器中模拟两个节点集群。*

引导 master

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

启动从卷

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

6. 打开浏览器，再转到 http://localhost:8080/auth 以试用。

3.4. 使用跨站点复制模式

在 Red Hat Single Sign-On 7.2 中作为技术预览功能引入的跨站点复制不再作为任何 Red Hat SSO 7.x 发行版本中支持的功能提供，包括最新的 RH-SSO 7.6 发行版本。红帽不推荐任何客户在其环境中实施或使用此功能，因为它不被支持。另外，对这个功能的支持例外不再被视为或接受。

将讨论跨站点复制的新解决方案，并在未来的 Red Hat build of Keycloak (RHBK) 中考虑，这是引入的产品，而不是 Red Hat SSO 8。稍后会提供更详细的信息。

第 4 章 管理子系统配置

可以通过编辑发行版中的 **standalone.xml**、**standalone-ha.xml** 或 **domain.xml** 文件来实现 Red Hat Single Sign-On 的低级别配置。这个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外的设置，但本节将重点介绍 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 子系统通常会声明在文件末尾，如下所示：

```
<subsystem xmlns="urn:jboss:domain:keycloak-server:1.2">
  <web-context>auth</web-context>
  ...
</subsystem>
```

请注意，这个子系统中的任何更改在服务器重启后不会生效。

4.1. 配置 SPI 供应商

使用该设置的上下文中讨论每个配置设置的具体细节。但是，了解用于声明 SPI 提供程序设置的格式很有用。

Red Hat Single Sign-On 是一个高度模块化的系统，实现了极大的灵活性。超过 50 个服务提供商接口 (SPI)，您可以交换出每个 SPI 的实现。SPI 的实施称为 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>
   <properties>
      <property name="spi-foo" value="spi-bar"/>
   </properties>
   <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 **myspi** 定义了两个提供程序。**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 允许多个提供程序，有些则不允许。因此，**default-provider** 可以帮助 SPI 选择。

SPI **属性** 可以用来指定 SPI 特定的配置属性。例如，**用户**、**client** 和 **role** SPI 允许通过 **storageProviderTimeout** 属性以毫秒为单位配置存储供应商超时，如下所示：

```
<spi name="user">
   <properties>
      <property name="storageProviderTimeout" value="10000"/>
```

```
    </properties>
</spi>
```

另请注意，每个供应商都定义了自己的一组配置属性。上述两个提供程序的事实具有名为 **foo** 的属性，只是一个统一的显示。

每个属性值的类型由提供程序来解释。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="[&amp;quot;EVENT1&amp;quot;,
                                                   &amp;quot;EVENT2&amp;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到提供程序。在本例中，系统将传递提供程序一个列表，其两个元素值为 EVENT1 和 EVENT2。要在列表中添加更多值，只需使用逗号分隔各个列表元素。不幸的是，您需要使用 " 转义引号括起 **每个列表元素**。

有关自定义提供程序和配置的详情，请遵循 服务器开发人员指南。

4.2. 启动 JBOSS EAP CLI

除了手动编辑配置之外，您还可以通过 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。在结合脚本时，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
```

这将会出现类似如下的提示：

提示

```
[disconnected /]
```

如果您想在运行的服务器上执行命令，则首先执行 **连接** 命令。

connect

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

*您可以考虑自己，"我没有输入任何用户名或密码！"。如果在与运行独立服务器或域控制器相同的机器上运行 **jboss-cli**，并且您的帐户有适当的文件权限，则不必在管理用户名和密码中设置或输入。如果您对此设置不熟悉，请参阅 JBoss EAP 配置指南 以了解更多详细信息。*

4.3. CLI 嵌入式模式

*如果您发生在与独立服务器相同的机器上，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中进行更改。为此，请先执行 **embed-server** 命令以及您要更改的配置文件。*

embed-server

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

4.4. 使用 CLI GUI 模式

CLI 也可以在 GUI 模式下运行。GUI 模式启动 Swing 应用程序，可让您以图形方式查看并编辑 正在运行的 服务器的完整管理模式。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。

流程

1. *以 GUI 模式启动 CLI*

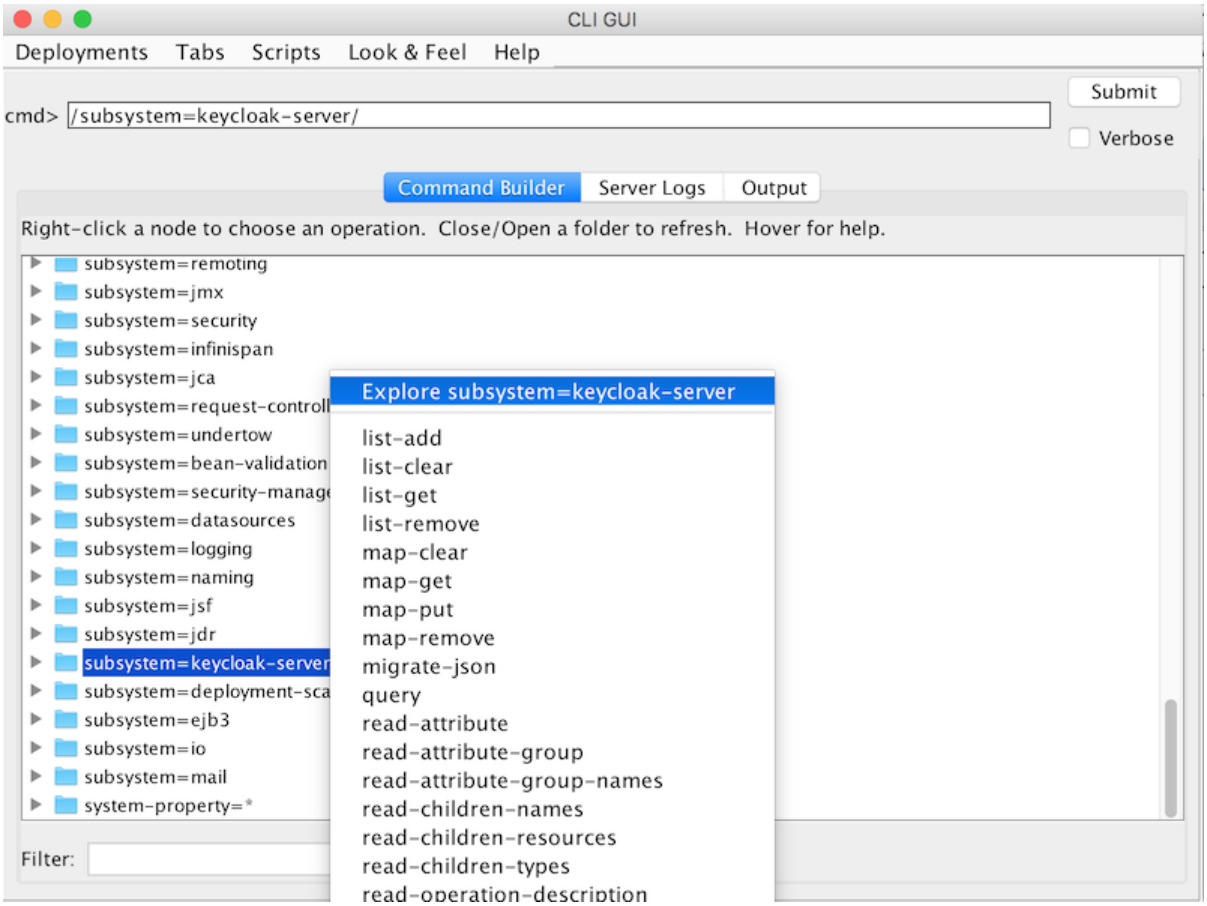
   ```
   $ .../bin/jboss-cli.sh --gui
   ```

   *注：要连接到远程服务器，也传递 **--connect** 选项。详情请使用 --help 选项。*

2. *向下滚动，以找到节点 **子系统=keycloak-server**。*

3. *右键单击节点，再选择 **Explore subsystem=keycloak-server**。*
   新标签页仅显示 keycloak-server 子系统。

   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。请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，您可以使用 CLI GUI 中的 **Scripts** 菜单，或者从命令行执行脚本，如下所示：*

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

4.6. CLI 方案

*以下是一些配置任务，以及如何使用 CLI 命令执行它们。请注意，在所有示例中，我们使用通配符路径 ** 表示您应该替换或者 keycloak-server 子系统的路径。*

对于独立，这只意味着：

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

对于域模式，这意味着类似：

***** = **/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. 设置全局默认值 theme

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

4.6.3. 添加新的 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. 为 SPI 添加或更改单个属性值

```
**/spi=mySPI/:map-put(name=properties, key=storageProviderTimeout, value=10000)
```

4.6.7. 从 SPI 中删除单个属性

```
**/spi=mySPI/:map-remove(name=properties, key=storageProviderTimeout)
```

4.6.8. 配置 dblock SPI

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

4.6.9. 为提供程序添加或更改单个属性值

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

4.6.10. 从供应商中删除单个属性

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

4.6.11. 在类型为 List 的供应商属性上设置值

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

第 5 章 PROFILES

Red Hat Single Sign-On 中没有默认启用的功能，其中包括不被支持的功能。此外，还有一些默认启用的功能，但可以禁用这些功能。

可启用和禁用的功能有：

Name	描述	默认启用	支持级别
account2	新的帐户管理控制台	是	支持
account_api	帐户管理 REST API	是	支持
admin_fine_grained_authz	精细授予的管理权限	否	预览
ciba	OpenID Connect Client Initiated Backchannel 身份验证(CIBA)	是	支持
client_policies	添加客户端配置策略	是	支持
client_secret_rotation	为机密客户端启用客户端 secret 轮转	是	预览
par	OAuth 2.0 推送授权请求 (PAR)	是	支持
declarative_user_profile	使用声明式风格配置用户配置集	否	预览
docker	Docker Registry 协议	否	支持
模拟(imimimation)	管理员能够模拟用户	是	支持
openshift_integration	扩展功能以启用 OpenShift 的安全	否	预览
recovery_codes	用于身份验证的恢复代码	否	预览
脚本	使用 JavaScript 编写自定义验证器	否	预览
step_up_authentication	步骤验证	是	支持
token_exchange	令牌交换服务	否	预览
upload_scripts	上传脚本	否	已弃用

Name	描述	默认启用	支持级别
web_authn	W3C Web 身份验证 (WebAuthn)	是	支持
update_email	更新电子邮件工作流	否	预览

启用所有技术预览功能以启动服务器：

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

您可以通过创建文件 ***standalone/configuration/profile.properties*** *（或* ***domain/servers/server-one/configuration/profile.properties*** *for* ***server-one*** *in domain-one）来永久进行设置。在文件中添加以下内容：*

```
profile=preview
```

启用特定功能以通过以下方式启动服务器：

```
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
```

要禁用特定功能启动服务器，使用以下命令：

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

例如，要禁用 Impersonation use ***-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 设置关系数据库

Red Hat Single Sign-On 随附其自身的嵌入式基于 Java 的关系数据库，称为 H2。这是 Red Hat Single Sign-On 将用来持久数据并仅存在的默认数据库，以便您可以默认运行身份验证服务器。

H2 数据库仅用于示例目的。它不是受支持的数据库，因此没有为数据库迁移进行测试。我们强烈建议您将其替换为更生产就绪的外部数据库。在高并发情形中，H2 数据库无法非常可行，不应在集群中使用。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 使用两个层次技术来保持其关系数据。底层技术是 JDBC。JDBC 是一个 Java API，用于连接到 RDBMS。您的数据库供应商提供了不同的 JDBC 驱动程序。本章论述了如何配置红帽单点登录以使用这些特定于供应商的驱动程序之一。

持久性的顶级技术是 Hibernate JPA。这是一个指向关系映射 API 的对象，可将 Java 对象映射到关系数据。Red Hat Single Sign-On 的大多数部署不必接触 Hibernate 的配置方面，但我们将讨论在出现很少情况时如何完成该操作。

注意

JBoss EAP 配置指南 的 数据源配置章节将更加 详尽地阐述数据源配置。

6.1. 数据库设置检查列表

以下是您要执行的步骤，以获取为红帽单点登录配置的 RDBMS。

1. 为您的数据库找到并下载 JDBC 驱动程序
2. 将驱动程序 JAR 打包到模块中，并将此模块安装到服务器中
3. 在服务器的配置配置集中声明 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，以定义数据库的连接参数

本章将在所有示例中使用 PostgresSQL。其他数据库遵循相同的安装步骤。

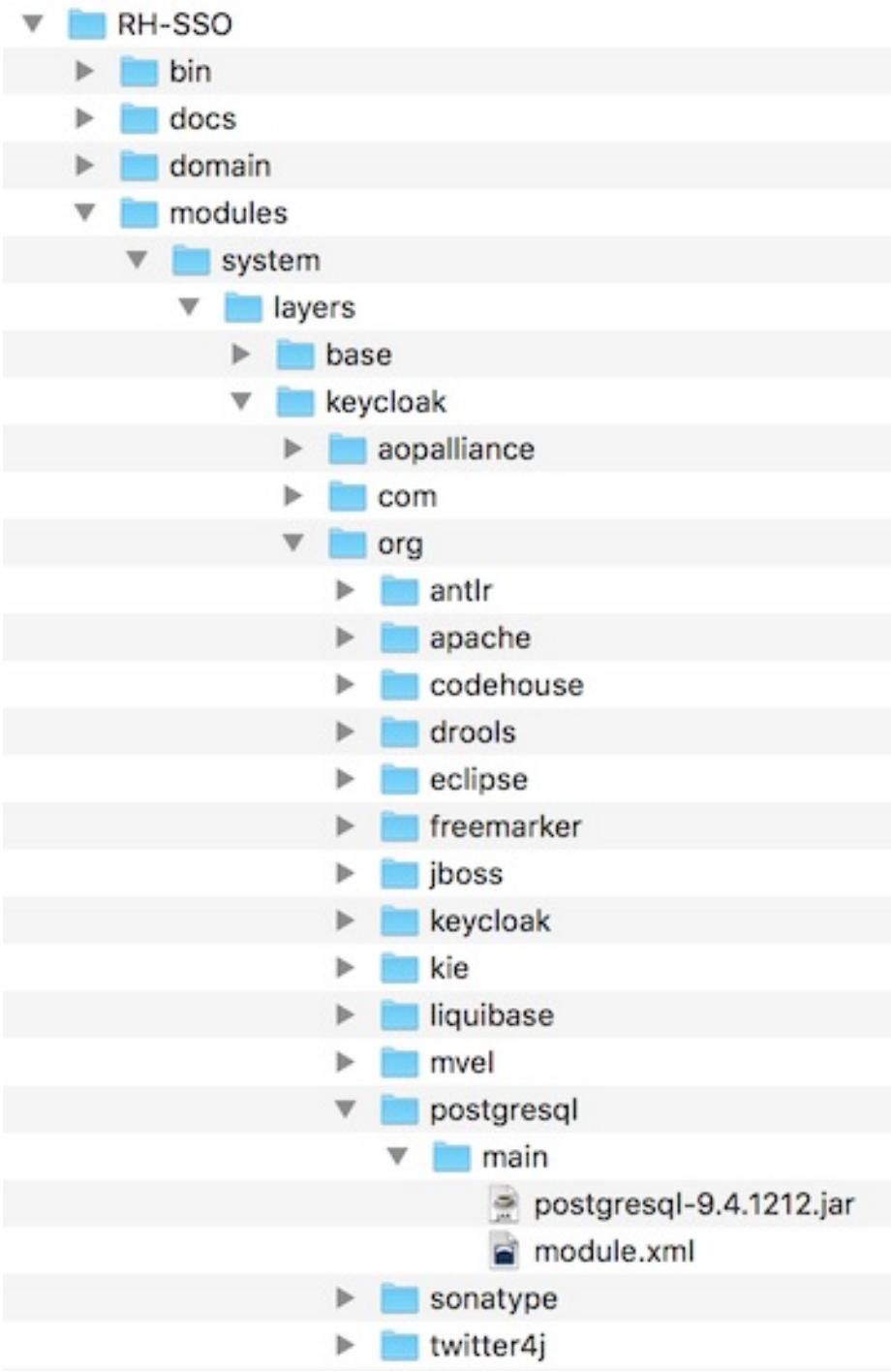
6.2. 打包 JDBC 驱动程序

为您的 RDBMS 查找并下载 JDBC 驱动程序 JAR。在使用这个驱动程序前，必须将其打包为模块并将其安装到服务器中。模块定义了挂载到红帽单点登录类路径中的 JAR，这些 JAR 依赖于其他模块的依赖项。

流程

1. 创建一个目录结构，在 Red Hat Single Sign-On 分发的 .../modules/ 目录中保存您的模块定义。其约定是将 JDBC 驱动程序的 Java 软件包名称用作目录结构的名称。对于 PostgreSQL，创建目录 org/postgresql/main。
2. 将数据库驱动程序 JAR 复制到此目录中，并在其中创建一个空的 module.xml 文件。

   模块目录



3. *打开 module.xml 文件并创建以下 XML ：*

模块 XML

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.3" name="org.postgresql">

    <resources>
        <resource-root path="postgresql-VERSION.jar"/>
    </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，组织/postgresql 映射到 **org.postgresql**。
- **resource-root path** 属性应该指定驱动程序的 JAR 文件名。
- 其余的是任何 JDBC 驱动程序 JAR 都有的普通依赖项。

6.3. 声明并加载 JDBC 驱动程序

将 JDBC 声明到您的部署配置文件中，使它在服务器引导时加载并可用。

先决条件

您已打包了 JDBC 驱动程序。

流程

1. 根据您的部署模式编辑其中一个文件来声明 JDBC 驱动程序：
   - 对于 standalone 模式，编辑 .../standalone/configuration/standalone.xml。
   - 对于单机集群模式，请编辑 .../standalone/configuration/standalone-ha.xml。
   - 对于域模式，编辑 .../domain/configuration/domain.xml。
     在域模式中，确保编辑正在使用的配置集： **auth-server-standalone** 或 **auth-server-clustered**
2. 在配置集内，在 **datasources** 子系统中搜索 **驱动程序** XML 块。
   您应该会看到为 H2 JDBC 驱动程序声明了预定义的驱动程序。这是您将声明用于外部数据库的 JDBC 驱动程序的位置。

   JDBC 驱动程序

   ```
   <subsystem xmlns="urn:jboss:domain:datasources:6.0">
     <datasources>
       ...
       <drivers>
         <driver name="h2" module="com.h2database.h2">
             <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
         </driver>
       </drivers>
     </datasources>
   </subsystem>
   ```

3. 在 **驱动程序** XML 块中，声明额外的 JDBC 驱动程序。
   - 为 **这个驱动程序分配任何名称**。
   - 指定 **module** 属性，它指向您为驱动程序 JAR 之前创建的 **模块** 软件包。
   - 指定驱动程序的 Java 类。
     下面是安装一个在本章前面定义的模块示例中的 PostgreSQL 驱动程序的示例。

声明您的 JDBC 驱动程序

```
<subsystem xmlns="urn:jboss:domain:datasources:6.0">
  <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. 修改红帽单点登录数据源

您修改 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。您可以在您注册 JDBC 驱动程序的同一配置文件和 XML 块中执行此操作。以下是设置到新数据库的连接的示例：

声明您的 JDBC 驱动程序

```
<subsystem xmlns="urn:jboss:domain:datasources:6.0">
  <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>
```

先决条件

- 您已声明了 JDBC 驱动程序。

流程

1. 搜索 **KeycloakDS 的数据源** 定义。
   您首先需要修改 **connection-url**。您供应商的 JDBC 实施的文档应该指定这个连接 URL 值的格式。

2. 定义您要使用的 **驱动程序**。

这是您在本章前部分中声明的 JDBC 驱动程序的逻辑名称。

每次您要执行事务时，打开与数据库的新连接非常昂贵。为了弥合，数据源实现维护一个开放连接池。**max-pool-size** 指定它池的最大连接数。您可以根据系统负载更改这个值。

3. 定义连接数据库所需的数据库用户名和密码。这一步至少需要 PostgreSQL。您可能会担心这些凭据在示例中使用明文。存在模糊处理这些凭据的方法，但这些方法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。

6.5. 数据库配置

此组件的配置可在您的分发的 **standalone.xml**、**standalone-ha.xml** 或 **domain.xml** 文件中找到。这个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:1.2">
    ...
    <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

指定数据源是否能力为 JTA 的布尔值属性

driverDialect

数据库减弱的值。在大多数情况下，您不需要指定此属性，因为 Hibernate 将自动探测到这个属性。

initializeEmpty

初始化数据库（如果为空）。如果设置为 false，则必须手动初始化数据库。如果要手动初始化数据库设置 migrationStrategy 以手动，这样将使用 SQL 命令创建文件以初始化数据库。**默认值为 true。**

migrationStrategy

用于迁移数据库的策略。有效值是 更新、手动 和验证。更新将自动迁移数据库架构。使用 SQL 命令手动将所需的更改导出到您可以手动对数据库执行的文件。验证将直接检查数据库是否最新。

migrationExport

编写手动数据库初始化/迁移文件的路径。

showSql

指定 Hibernate 是否应该在控制台中显示所有 SQL 命令（默认为false）。这是非常详细！

formatSql

指定 Hibernate 是否应该格式化 SQL 命令（默认为true）

globalStatsInterval

将从 Hibernate 记录有关执行 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）报告给服务器日志，并在每个报告后清除。

schema

指定要使用的数据库 schema

注意

JBoss EAP 开发指南 中介绍了这些配置切换等信息。

6.6. 数据库的 UNICODE 注意事项

Red Hat Single Sign-On 中的数据库 schema 在以下特殊字段中只有 Unicode 字符串的帐户：

- realms: 显示名称、HTML 显示名称、本地化文本（密钥和值）
- 联合供应商：显示名称
- 用户：用户名、指定名称、姓、属性名称和值
- groups: name、attribute name 和 values

- *roles: name*

- *对象描述*

否则，字符仅限于数据库编码中包含的字符，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，这与使用 8 位编码的情况相比，这个字符串的最大长度会比较短。

有些数据库需要对数据库和/或 JDBC 驱动程序进行特殊设置，以便能处理 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，它仍然可以正常工作，只要它在数据库和 JDBC 驱动程序的级别上正确处理 UTF-8 编码。

从技术上讲，所有字段的 Unicode 支持的关键条件是数据库是否允许为 VARCHAR 和 CHAR 字段设置 Unicode 字符集。如果是，则 Unicode 的几率会非常容易，通常以字段长度的代价为代价。如果它只支持 NVARCHAR 和 NCHAR 字段的 Unicode 支持，则所有文本字段的 Unicode 支持不太可能，因为 Keycloak 模式使用 VARCHAR 和 CHAR 字段。

6.6.1. Oracle 数据库

Unicode 字符被正确处理，为 VARCHAR 和 CHAR 字段（例如，使用 AL32UTF8 字符集作为数据库字符集）的 Unicode 支持创建数据库。JDBC 驱动程序不需要特殊设置。

如果数据库字符集不是 Unicode，那么要在特殊字段中使用 Unicode 字符，需要将 JDBC 驱动程序配置为 oracle.jdbc.defaultNChar 设为 true。这可能很明智的，但并非严格必需，但也会将 oracle.jdbc.convertNcharLiterals 连接属性设置为 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 字符只针对特殊字段正确处理。不需要 JDBC 驱动程序或数据库的特殊设置。

6.6.3. MySQL 数据库

Unicode 字符被正确处理，只要在 CREATE DATABASE 命令中的 VARCHAR 和 CHAR 字段中有 Unicode 支持来创建数据库（例如，使用 utf8 字符设置为 MySQL 5.5 中的默认数据库字符）。请注意，由于 utf8 字符集有不同的存储要求，utf8mb4 字符集无法正常工作 [1]).请注意，在这种情况下，对非特

殊字段的长度限制不会应用，因为创建了列以容纳给定字符数而不是字节数。如果数据库默认字符集不允许存储 Unicode，则只允许特殊字段存储 Unicode 值。

在 JDBC 驱动程序设置一侧，需要在 JDBC 连接设置中添加 连接属性Encoding=UTF-8。

6.6.4. PostgreSQL 数据库

当数据库字符集为 UTF8 时，支持 Unicode。在这种情况下，可以在任何字段中使用 Unicode 字符，非特殊字段不会减少字段长度。不需要对 JDBC 驱动程序的特殊设置。

PostgreSQL 数据库的字符集在创建后确定。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，那么您可以使用 UTF8 创建数据库作为其字符集：

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

---

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

第 7 章 使用公共主机名

Red Hat Single Sign-On 使用公共主机名来做多个事项。例如，在令牌签发者字段中，使用密码重置电子邮件发送的 URL。

主机名 SPI 提供为请求配置主机名的方法。默认供应商允许为 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，也可以开发您自己的供应商。

7.1. 默认供应商

默认主机名供应商使用配置的 frontendUrl 作为 frontend 请求的基本 URL（来自 user-agents 的 requests），并使用请求 URL 作为后端请求（从客户端重定向请求）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，而内部它的 URL 可能是 https://10.0.0.10:8080/auth。

这样，可以把 user-agents（浏览器）通过公共域名向 Red Hat Single Sign-On 发送请求，而内部客户端则可使用内部域名或 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。在这里，一个实例的公共客户端会通过公共端点联系 Keycloak，这会导致 authorization_endpoint 和 token_endpoint 的基础是相同的。

要为 Keycloak 设置 frontendUrl，您可以将 -Dkeycloak.frontendUrl=https://auth.example.org 传递给启动，或者在 standalone.xml 中配置。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=default:write-
attribute(name=properties.frontendUrl,value="https://auth.example.com")
```

如果您希望所有请求都通过公共域名，您可以强制后端请求使用前端 URL，也可以通过将 forceBackendUrlToFrontendUrl 设置为 true。

也可以覆盖单个域的默认前端 URL。这可以在管理控制台中完成。

如果您不想公开公共域中的 admin 端点和控制台，请使用属性 adminUrl 为 admin 控制台设置固定 URL，这与 frontendUrl 不同。还需要阻止外部对 /auth/admin 的访问，有关如何操作 "服务器管理指南" 的详细信息。

7.2. 自定义供应商

要开发自定义主机名提供程序，您需要实施 org.keycloak.urls.HostnameProviderProvidery 和 org.keycloak.urls.HostnameProvider。

请参阅《服务器开发人员指南》 中的"服务提供程序接口"一节中的说明，了解如何开发自定义供应商。

第 8 章 设置网络

Red Hat Single Sign-On 的默认安装可以在一些网络限制的情况下运行。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个本地机器上使用。对于基于 HTTP 的连接，不使用 80 和 443 等默认端口。HTTPS/SSL 未开箱即用且未配置，Red Hat Single Sign-On 有很多安全漏洞。最后，Red Hat Single Sign-On 通常可能需要向外部服务器提供安全 SSL 和 HTTPS 连接，因此需要设置信任存储，以便可以正确验证端点。本章讨论所有这些内容。

8.1. 绑定地址

默认情况下，红帽单点登录绑定到 localhost 回送地址 127.0.0.1。如果您需要网络中的身份验证服务器，这不是非常有用的默认值。通常，建议您在公共网络上部署反向代理或负载均衡器，并将流量路由到专用网络上的各个红帽单点登录服务器实例。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，可以在命令行中完成 Choosing a Operating Mode 章节中讨论的 standalone.sh 或 domain.sh 启动脚本。

```
$ 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，如果您不想在命令行中设置绑定地址，您可以编辑部署的配置集配置。根据您的 操作模式，打开配置文件(standalone.xml 或 domain.xml)，并查找 interfaces XML 块。

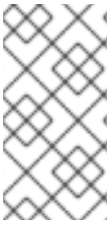
```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
    </interface>
    <interface name="public">
        <inet-address value="${jboss.bind.address:127.0.0.1}"/>
    </interface>
</interfaces>
```

公共接口 与创建可公开提供的套接字子系统相对应。这些子系统的一个示例是 Web 层，它是提供 Red Hat Single Sign-On 的身份验证端点的 Web 层。管理接口 对应于 JBoss EAP 的管理层所打开的套接字。具体来说，socket 允许您使用 jboss-cli.sh 命令行界面和 JBoss EAP web 控制台。

查看 公共接口 时，您会看到它具有特殊字符串 ${jboss.bind.address:127.0.0.1}。此字符串表示可以在命令行中通过设置 Java 系统属性覆盖的值 127.0.0.1，例如：

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

-b 只是此命令的简写表示法。因此，您可以直接在配置集中配置绑定地址值，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. SOCKET 端口绑定

为每个套接字打开的端口具有一个预定义的默认端口，可在命令行中或配置内被覆盖。要说明此配置，请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 定义服务器将打开的套接字连接。这些绑定指定了它们使用 的接口 （绑定地址）以及他们将打开的端口号。您最感兴趣的对象有：

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 用作负载平衡器时，Apache HTTPD 服务器以组合 mod-cluster 使用此协议。

management-http

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

在域模式下运行时，套接字配置会很欺骗，因为示例 domain.xml 文件定义了多个 socket-binding-groups。???如果滚动到 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. HTTPS/SSL

警告

默认情况下，Red Hat Single Sign-On 不会被设置来处理 SSL/HTTPS。强烈建议您在红帽单点登录服务器本身或 Red Hat Single Sign-On 服务器前面的反向代理上启用 SSL。

这个默认行为由每个 Red Hat Single Sign-On 域的 SSL/HTTPS 模式定义。这在《服务器管理指南》中更为详细地讨论，但让我们为一些上下文和这些模式的简单概述。

外部请求

Red Hat Single Sign-On 可在没有 SSL 的情况下运行，因此只要您升级到私有 IP 地址，如 localhost、127.0.0.1、10.x.x.x、192.x、192.x 和 172.16.x.x。如果您没有服务器上配置了 SSL/HTTPS，或者您尝试从非专用 IP 广告通过 HTTP 访问 Red Hat Single Sign-On，则会出现错误。

none

Red Hat Single Sign-On 不需要 SSL。这只在开发过程中使用。

所有请求

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

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

8.4. 为 RED HAT SINGLE SIGN-ON 服务器启用 HTTPS/SSL

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. 配置红帽单点登录服务器以使用此密钥对和证书。

8.4.1. 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证书，并将其导入到 Java 密钥存储中，然后才能在其中部署 Red Hat Single Sign-On Server 的 web 容器中启用 HTTPS。

8.4.1.1. 自签名证书

在开发中，您可能没有可用的第三方签名证书来测试 Red Hat Single Sign-On 部署，因此您将需要使用 Java JDK 附带的 密钥 工具生成自签名证书。

```
$ 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
```

当您看到 哪个问题是您的名字和姓氏？ 时，请提供您要在其中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 文件将产生与您在其中执行 keytool 命令相同的目录中。

如果您需要第三方签名证书，但没有证书，您可以在 cacert.org 处获取一个免费证书。但是，您首先需要使用以下步骤。

流程

1. 生成证书请求：

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

其中，您的域 是生成此证书的 DNS 名称。keytool 生成请求：

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIC2jCCAcICAQAwZTELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk1BMREwDwYDVQQHE
whXZXN0Zm9y
ZDEQMA4GA1UEChMHUmVkIEhhdDEQMA4GA1UECxMHUmVkIEhhdDESMBAGA1UEAxM
JbG9jYWxob3N0
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAr7kck2TaavlEOGbcpi9c0rncY4Hhd
zmY
Ax2nZfq1eZEaIPqI5aTxwQZzzLDK9qbeAd8Ji79HzSqnRDxNYaZu7mAYhFKHgixsolE3o5Yfzb
w1
29RvyeUVe+WZxv5oo9wolVVpdSINIMEL2LaFhtX/c1dqiqYVpfnvFshZQaIg2nL8juzZcBjj4as
H98gIS7khql/dkZKsw9NLvyxgJvp7PaXurX29fNf3ihG+oFrL22oFyV54BWWxXCKU/GPn61EGZ
Gw
Ft2qSIGLdctpMD1aJR2bcnlhEjZKDksjQZoQ5YMXaAGkcYkG6QkgrocDE2YXDbi7GIdf9MegV
J35
2DQMpwIDAQABoDAwLgYJKoZIhvcNAQkOMSEwHzAdBgNVHQ4EFgQUQwlZJBA+fjiDdiVz
aO9vrE/i
n2swDQYJKoZIhvcNAQELBQADggEBAC5FRvMkhal3q86tHPBYWBuTtmcSjs4qUm6V6f63frh
veWHf
PzRrI1xH272XUIeBk0gtzWo0nNZnf0mMCtUBbHhhDcG82xolikfqibZijoQZCiGiedVjHJFtniDQ
9bMDUOXEMQ7gHZg5q6mJfNG9MbMpQaUVEEFvfGEQQxbiFK7hRWU8S23/d80e8nExgQx
dJWJ6vd0X
MzzFK6j4Dj55bJVuM7GFmfdNC52pNOD5vYe47Aqh8oajHX9XTycVtPXl45rrWAH33ftbrS8SrZ
2S
vqIFQeuLL3BaHwpl3t7j2lMWcK1p80laAxEASib/fAwrRHpLHBXRcq6uALUOZl4Alt8=
-----END NEW CERTIFICATE REQUEST-----
```

2. ***将此 CA 请求发送到您的证书颁发机构(CA)。***

   CA 将发布您签名的证书并将其发送给您。

3. *获取和导入* ***CA 的 root*** *证书。*

   您可以从 CA 下载证书（换句话说：root.crt）并导入，如下所示：

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

4. ***将新 CA 生成的证书导入到密钥存储中：***

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

8.4.2. 配置红帽单点登录以使用密钥存储

现在，您已拥有有适当证书的 Java 密钥存储，您需要配置您的红帽单点登录安装才能使用它。使用适用于您的安装的配置步骤：

- *JBoss 安全传统*
- *Elytron TLS v1.2*
- *Elytron TLS v1.3*

8.4.2.1. JBoss 安全传统

流程

1. *编辑 standalone.xml、standalone-ha.xml 或 host.xml 文件，以使用密钥存储并启用 HTTPS。*
2. *将密钥存储文件移到 部署的 配置目录中，或者在您选择的位置中的文件，并提供绝对路径。*

   如果您使用绝对路径，请从配置中删除可选的 relative-to 参数（请参阅 操作模式）。
3. *在 JBoss EAP 的 bin 目录中，创建一个名为 sso_legacy.cli 的批处理文件。*
4. *在批处理文件中添加以下内容：*

```
# Start batching commands

batch

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

# Run the batch commands

run-batch
```

5. *启动 Red Hat Single Sign-On 服务器。*

6. 更改到 JBoss EAP 的 bin 目录。

7. 运行以下脚本：

```
$ sh jboss-cli.sh --connect --file=sso_legacy.cli

The batch executed successfully
process-state: reload-required
```

8. 重启 Red Hat Single Sign-On 服务器，使 sso_legacy.cli 更改生效。

8.4.2.2. Elytron TLS v1.2

流程

1. 在 JBoss EAP 的 bin 目录中，创建一个名为 sso.cli 的批处理文件。

2. 在批处理文件中添加以下内容：

```
# Start batching commands

batch

# Add the keystore, key manager and ssl context configuration in the elytron
subsystem

/subsystem=elytron/key-store=httpsKS:add(relative-
to=jboss.server.config.dir,path=keycloak.jks,credential-reference={clear-
text=secret},type=JKS)
/subsystem=elytron/key-manager=httpsKM:add(key-store=httpsKS,credential-
reference={clear-text=secret})
/subsystem=elytron/server-ssl-context=httpsSSC:add(key-
manager=httpsKM,protocols=["TLSv1.2"])

# Change the undertow subsystem configuration to use the ssl context defined in the
previous step for https

/subsystem=undertow/server=default-server/https-listener=https:undefine-
attribute(name=security-realm)
/subsystem=undertow/server=default-server/https-listener=https:write-
attribute(name=ssl-context, value=httpsSSC)

# Run the batch commands

run-batch
```

3. 启动 Red Hat Single Sign-On 服务器。

4. 更改到 JBoss EAP 的 bin 目录。

5. 运行以下脚本：

```
$ sh jboss-cli.sh --connect --file=sso.cli

The batch executed successfully
process-state: reload-required
```

6. 重启 Red Hat Single Sign-On 服务器，使 sso.cli 更改生效。

有关配置 TLS 的更多信息，请参阅 WildFly 文档。

8.4.2.3. Elytron TLS 1.3

流程

1. 在 JBoss EAP 的 bin 目录中，创建一个名为 sso.cli 的批处理文件。

2. 在批处理文件中添加以下内容：

```
batch

# Add the keystore, key manager and ssl context configuration in the elytron subsystem
/subsystem=elytron/key-store=httpsKS:add(relative-to=jboss.server.config.dir,path=keycloak.jks,credential-reference={clear-text=secret},type=JKS)
/subsystem=elytron/key-manager=httpsKM:add(key-store=httpsKS,credential-reference={clear-text=secret})


/subsystem=elytron/server-ssl-context=httpsSSC:add(key-manager=httpsKM,protocols=["TLSv1.3"])
/subsystem=elytron/server-ssl-context=httpsSSC:write-attribute(name=cipher-suite-names,value=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256)

# Change the undertow subsystem configuration to use the ssl context defined in the
```

```
previous step for https

/subsystem=undertow/server=default-server/https-listener=https:undefine-
attribute(name=security-realm)
/subsystem=undertow/server=default-server/https-listener=https:write-
attribute(name=ssl-context, value=httpsSSC)

# Run the batch commands

run-batch
```

3. 启动 Red Hat Single Sign-On 服务器。

4. 更改到 JBoss EAP 的 bin 目录。

5. 运行以下脚本：

```
$ sh jboss-cli.sh --connect --file=sso.cli

The batch executed successfully
process-state: reload-required
```

6. 重启 Red Hat Single Sign-On 服务器，使 sso.cli 更改生效。

有关配置 TLS 的更多信息，请参阅 WildFly 文档。

8.5. 传出 HTTP 请求

Red Hat Single Sign-On 服务器通常需要向它保护的应用程序和服务提供非浏览器 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。您需要在 standalone.xml、standalone-ha.xml 或 domain.xml 中配置。这个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

如果传出请求没有接收这个时间的数据，请超时连接。

connection-pool-size

池中可以有多少个连接（默认为128）。

max-pooled-per-route

每个主机可以池划分多少连接（默认为 64）。

connection-ttl-millis

以毫秒为单位到最长连接时间。默认不设置。

max-connection-idle-time-millis

连接在连接池中可以保持闲置的最大时间（默认为900 秒）。将启动 Apache HTTP 客户端的后台清理线程。设置为 -1 可禁用此检查和后台线程。

disable-cookies

默认为 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 代理映射的部分，以了解 HTTP 请求。

disable-trust-manager

如果传出请求需要 HTTPS，且该配置选项被设置为 true，则不需要指定信任存储。此设置应只在开发期间使用，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 选项。默认值为 false。

8.5.1. 传出 HTTP 请求的代理映射

Red Hat Single Sign-On 发送的传出 HTTP 请求可以选择性地根据以逗号分隔的代理映射列表使用代理服务器。proxy-mapping 以 hostnamePattern;proxyUri 的形式表示基于 regex 的主机名和 proxy-uri 的组合，例如：

```
.*\.(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解析模式匹配。第一个匹配模式决定要使用的代理部分。如果给定主机名没有配置的模式都不匹配，则无法使用代理。

如果代理服务器需要身份验证，请使用以下格式包括代理用户的凭据，即 username:password@。例如：

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

proxy-uri 的特殊值 NO_PROXY 用于表示不应将任何代理用于与相关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，一个需要用 " 对字符进行编码。

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

8.5.2. 使用标准环境变量

或者，也可以使用标准环境变量来配置代理映射，即 **HTTP_PROXY**、**HTTPS_PROXY** 和 **NO_PROXY** 变量。

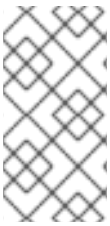
HTTP_PROXY 和 **HTTPS_PROXY** 变量代表应用于所有传出 HTTP 请求的代理服务器。**Red Hat Single Sign-On** 在两者之间没有不同。如果同时指定了，**HTTPS_PROXY** 会优先考虑代理服务器使用的实际方案。

NO_PROXY 变量用于定义不应使用代理的主机名的逗号分隔列表。如果指定了主机名，则使用代理将其所有前缀（子域）都排除在内。

使用以下示例：

```
HTTPS_PROXY=https://www-proxy.acme.com:8080
NO_PROXY=google.com,login.facebook.com
```

在这个示例中，除 login.google.com 的请求外，所有传出的 HTTP 请求都将使用 https://www-proxy.acme.com:8080 代理服务器，google.com、auth.login.facebook.com。但是，例如 groups.facebook.com 将通过代理路由。



注意

环境变量可以是小写或大写。小写具有优先权。例如，如果同时定义了 HTTP_PROXY 和 http_proxy，则将使用 http_proxy。

如果使用子系统配置来定义代理映射（如上所述），红帽单点登录不会考虑环境变量。在这种情况下，不应使用代理服务器，尽管定义了 HTTP_PROXY 环境变量。要做到这一点，您可以指定一个通用没有代理路由，如下所示：

```
<spi name="connectionsHttpClient">
    <provider name="default" enabled="true">
        <properties>
            <property name="proxy-mappings" value=".*;NO_PROXY"/>
        </properties>
    </provider>
</spi>
```

8.5.3. 传出的 HTTPS 请求信任存储

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。这是防止中间人攻击的必要性。这些远程服务器或者签署这些证书的 CA 的证书必须放在信任存储中。这个信任存储由红帽单点登录服务器管理。

信任器的配置始终由 Red Hat Single Sign-On truststore SPI 进行。如果密钥存储是由 JBoss 安全传统或 Elytron TLS 配置的，则本节中的说明适用。

在安全地连接到身份代理、LDAP 身份提供程序、发送电子邮件以及与客户端应用程序后端通信时使用信任存储。

警告

默认情况下，没有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 创建新信任存储文件，或将可信主机证书添加到现有主机证书中：

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

truststore 在分发的 standalone.xml、standalone-ha.xml 或 domain.xml 文件中配置。这个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"/>
        </properties>
    </provider>
</spi>
```

此设置的可能配置选项有：

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。这就是信任者的作用。该密钥存储包含一个或多个可信主机证书或证书颁发机构。这个信任存储文件应该只包含安全主机的公共证书。如果定义了这些属性，则这是 REQUIRED。

password

密钥存储的密码。如果定义了这些属性，则这是 REQUIRED。

hostname-verification-policy

WILDCARD 默认.对于 HTTPS 请求，这将验证服务器证书的主机名。ANY 表示无法验证主机名。WILDCARD Allows 子域名称中的通配符，例如：*.foo.com。STRICT CN CN 必须与主机名完全匹配。

第 9 章 配置红帽单点登录以在集群中运行

要配置 Red Hat Single Sign-On 在集群中运行，您需要执行以下操作：

- 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

本指南前面讨论过一个操作模式和配置共享数据库。本章论述了设置负载均衡器并提供专用网络以及在集群中引导主机。

注意

在没有 IP 多播的情况下可以集群 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Groups 章节。

9.1. 推荐的网络架构

推荐的网络架构用于部署红帽单点登录，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红帽单点登录服务器（位于专用网络上）。这会隔离所有集群连接，并提供保护服务器的 nice 方法。

注意

默认情况下，无法阻止未经授权的节点加入集群和广播多播消息。因此，集群节点应该位于私有网络中，而防火墙则防止它们被外部攻击。

9.2. 集群示例

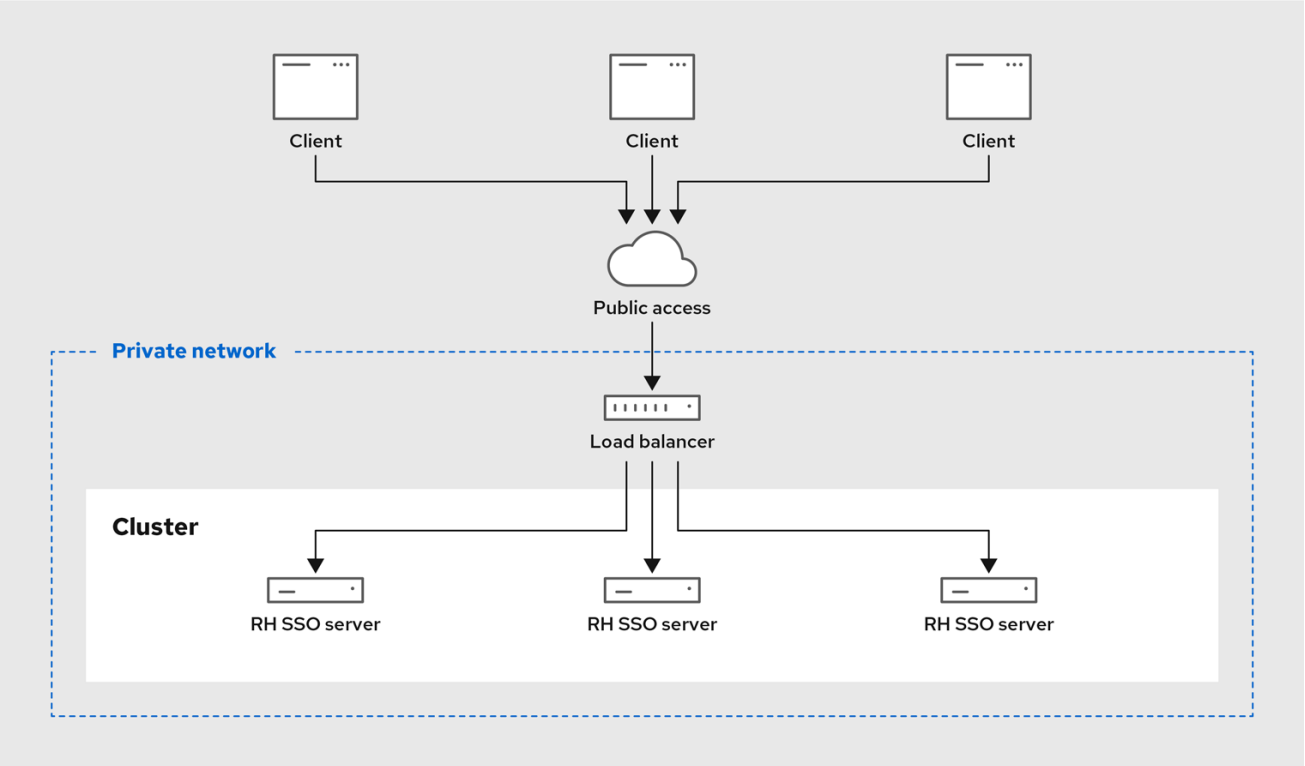
Red Hat Single Sign-On 带有利用域模式的开箱即用集群演示。如需了解更多详细信息，请参阅集群域示例 章节。

9.3. 设置负载均衡器或代理

本节讨论在集群的 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事项。它还涵盖了配置 集群域示例 的内置负载均衡器。

下图演示了负载平衡器的使用。在本例中，负载均衡器充当三个客户端和三个红帽单点登录服务器集群之间的反向代理。

Load Balancer 图表示例



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 - 失败的登录尝试使用错误的源 IP 地址记录

- SSL 必需 - 如果需要 SSL，则需要对所有外部请求设置为外部（默认值）
- 身份验证流 - 使用 IP 地址的自定义身份验证流，如仅为外部请求显示 OTP
- 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反向代理或负载均衡器时，这可能会有问题。通常的设置是，您有一个前端代理位于公共网络上，该网络可负载平衡并将请求转发给位于专用网络中的后端红帽单点登录服务器实例。在这种场景中，您必须进行一些额外的配置，以便红帽单点登录服务器实例将实际客户端 IP 地址转发到和处理。具体来说：

- 配置反向代理或负载均衡器以正确设置 X-Forwarded-For 和 X-Forwarded-Proto HTTP 标头。
- 配置反向代理或负载均衡器以保留原始的 'Host' HTTP 标头。
- 配置身份验证服务器，从 X-Forwarded-For 标头读取客户端的 IP 地址。

将您的代理配置为生成 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。采取额外的预防措施以确保代理设置了 X-Forwarded-For 标头。如果您的代理没有正确配置，那么 恶意 客户端可以自行设置这个标头，并欺骗 Red Hat Single Sign-On 认为客户端正在与实际 IP 地址不同的 IP 地址连接。如果您正在执行任何黑色或白名单 IP 地址，这将变得非常重要。

除代理本身外，您需要在红帽单点登录端配置一些事项。如果您的代理通过 HTTP 协议转发请求，那么您需要配置 Red Hat Single Sign-On，以从 X-Forwarded-For 标头而不是从网络数据包拉取客户端的 IP 地址。要做到这一点，请打开配置文件配置文件(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:12.0 XML 块。

x-Forwarded-For HTTP 配置

```
<subsystem xmlns="urn:jboss:domain:undertow:12.0">
  <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** 协议而非 **HTTP** 来转发请求（如 **Apache HTTPD + mod-cluster**），则必须配置其他不同。您不需要修改 **http-listener**，而是需要添加过滤器来从 **AJP** 数据包中提取此信息。

X-Forwarded-For AJP 配置

```
<subsystem xmlns="urn:jboss:domain:undertow:12.0">
    <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. 使用反向代理启用 HTTPS/SSL

假设您的反向代理不使用 **SSL** 端口 **8443**，您还需要配置 **HTTPS** 流量重定向的端口。

```
<subsystem xmlns="urn:jboss:domain:undertow:12.0">
    ...
```

```
    <http-listener name="default" socket-binding="http"
      proxy-address-forwarding="true" redirect-socket="proxy-https"/>
    ...
</subsystem>
```

流程

1. 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。

2. 在 **socket-binding - group** 元素中添加新的 **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

您可以验证反向代理或负载均衡器配置

流程

1. 通过反向代理打开路径 **/auth/realms/master/.well-known/openid-configuration**。

   例如，如果反向代理地址是 **https://acme.com/**，打开 URL **https://acme.com/auth/realms/master/.well-known/openid-configuration**。这将显示 **JSON** 文档，其中列出了红帽单点登录的多个端点。

2. 确保端点以反向代理或负载均衡器的地址（**scheme**、**domain** 和 **port**）开头。这样做后，请确保您的 **Red Hat Single Sign-On** 使用正确的端点。

3. 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。这应该在服务器日志中显示如下警告：

```
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

4. 检查 **ipAddress** 的值是您试图使用 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. 使用内置负载均衡器

本节介绍配置在 **集群域示例** 中讨论的内置负载均衡器。

集群域示例 仅用于在一个计算机上运行。要在另一台主机上启动从设备，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。您不需要 **standalone/** 目录。
3. 编辑 **host-slave.xml** 文件，以更改所用的绑定地址或在命令行中覆盖它们

流程

1. 打开 **domain.xml**，以便您可以使用负载均衡器配置注册新主机从设备。
2. 转到 **load-balancer** 配置集中的 **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 配置

```
<subsystem xmlns="urn:jboss:domain:undertow:12.0">
 ...
 <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是唯一的，因为此值由 Cookie 使用，以便在负载平衡时启用粘性会话。

3.

转到 **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>
```

9.3.4.1. Master 绑定地址

接下来，您需要做的是更改 master 主机的公共 和管理 绑定地址。按照 绑定地址 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
```

9.3.4.2. 主机从属绑定地址

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，或者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
```

与主机的从设备 IP 地址相关的 jboss.bind.address.address.management 的值。jboss.domain.master.address 的值需要是域控制器的 IP 地址，即 master 主机的管理地址。

其他资源

- 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 负载平衡 部分。

9.4. 粘性会话

典型的集群部署包括负载均衡器（逆向代理）以及 2 个或更多红帽在专用网络上的 Red Hat Single Sign-On 服务器。出于性能考虑，如果负载均衡器将所有与特定浏览器会话相关的请求转发到同一个红帽单点登录后端节点，这可能很有用。

其原因在于，Red Hat Single Sign-On 在涵盖保存与当前身份验证会话和用户会话相关的数据的情况下，使用 Infinispan 分布式缓存。默认情况下，Infinispan 分布式缓存配置为一个所有者。这意味着，特定会话只保存在一个集群节点中，另一个节点需要远程查找会话（如果它们想要访问它）。

例如，如果 ID 为 123 的验证会话保存在 node1 上的 Infinispan 缓存中，而 node2 需要查找此会话，则需要通过网络将请求发送到 node1，以返回特定会话实体。

如果特定会话实体始终在本地可用，则可使用粘性会话的帮助。带有公共前端负载均衡器的集群环境中的工作流和两个后端红帽单点登录节点可能如下所示：

- 用户发送初始请求以查看红帽单点登录登录屏幕

- 此请求由 frontend 负载均衡器提供，它将转发到一些随机节点（如 node1）。严格说，该节点不需要随机，但可以根据其他标准（客户端 IP 地址等）选择。它完全取决于底层负载均衡器的实施和配置（逆向代理）。

- Red Hat Single Sign-On 使用随机 ID（如 123）创建身份验证会话，并将其保存到 Infinispan 缓存中。

- Infinispan 分布式缓存根据会话 ID 的哈希分配会话的主所有者。有关此问题的详情，请参阅 Infinispan 文档。我们假设将 node2 分配为此会话的所有者。

- Red Hat Single Sign-On 使用 < session-id>.<owner-node-id> 格式创建 cookie AUTH_SESSION_ID。在我们的示例中，它是 123.node2。

- 使用 Red Hat Single Sign-On 登录屏幕和浏览器中的 AUTH_SESSION_ID cookie 返回响应

此时，如果负载均衡器将所有下一个请求转发到 node2，这是具有 ID 123 的身份验证会话的所有者，因此 Infinispan 可以在本地查询该会话时很有用。身份验证完成后，身份验证会话将转换为用户会话，该会话也将保存在 node2 上，因为它具有相同的 ID 123。

集群设置不强制使用粘性会话，但出于上述原因，性能很好。您需要通过 AUTH_SESSION_ID cookie 配置负载均衡器来粘性。具体操作方法取决于您的负载均衡器。

建议您在 Red Hat Single Sign-On side 中在启动时使用系统属性 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。以下是使用这个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，在生产环境下，路由名称应使用与后端主机相同的名称，但这不是必须的。您可以使用不同的路由名称。例如，如果您要将 Red Hat Single Sign-On 服务器的主机名隐藏到您的专用网络中。

9.4.1. 禁用添加路由

有些负载均衡器可以自行配置来添加路由信息，而不依赖于后端红帽单点登录节点。但如上述所述，推荐使用 Red Hat Single Sign-On 添加路由。这是因为，在以性能提高的情况下，红帽单点登录知道知道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。

如果您更喜欢在 Red Hat Single Sign-On 中添加以下内容到 RHSSO_HOME/standalone/configuration/standalone-ha.xml 文件中的 AUTH_SESSION_ID cookie，可以禁用在 Red Hat Single Sign-On 子系统配置中添加路由信息：

```
<subsystem xmlns="urn:jboss:domain:keycloak-server:1.2">
 ...
   <spi name="stickySessionEncoder">
      <provider name="infinispan" enabled="true">
         <properties>
            <property name="shouldAttachRoute" value="false"/>
         </properties>
      </provider>
   </spi>

</subsystem>
```

9.5. 设置多播网络

默认集群支持需要 IP 多播。多播是一个网络广播协议。这个协议在引导时用来发现和加入集群。它还用于广播有关复制的消息，以及红帽单点登录使用的分布式缓存无效。

红帽单点登录的集群子系统在 JGroups 堆栈上运行。在框中，集群的绑定地址绑定到私有网络接口，使用 127.0.0.1 作为默认 IP 地址。

流程

1. 编辑 绑定地址 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>
```

2. **配置 jboss.bind.address.private 和 jboss.default.multicast.address，以及集群堆栈中服务的端口。**

注意

在没有 IP 多播的情况下可以集群 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

当集群节点隔离在专用网络上时，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。只要您的专用网络安全，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向集群中发送非常敏感信息。

如果要启用集群通信的验证和加密，请参阅 JBoss EAP 配置指南中的 保护集群。https://access.redhat.com/documentation/zh-cn/red_hat_jboss_enterprise_application_platform/7.4/html-single/configuration_guide/configuring_high_availability#securing_cluster

9.7. 序列化集群启动

Red Hat Single Sign-On 集群节点可以同时引导。当 Red Hat Single Sign-On 服务器实例引导它时，可能需要进行一些数据库迁移、导入或首次初始化。DB 锁定用于在集群节点同时引导时防止启动操作相互冲突。

默认情况下，此锁定的最大超时时间为 900 秒。如果某个节点在这个锁定中等待超过超时，它将无法引导。通常，您通常不需要增加/减少默认值，但只需将它在 standalone. xml、standalone -ha.xml 或

domain.xml 文件中配置它时。这个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
```

您可能需要使用附加参数或系统属性。例如，绑定主机的参数 -b 或系统属性 jboss.node.name 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，您应该在这两个集群节点日志中看到类似于如下的消息：*

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

> node2/keycloak]

如果您只看到提到一个节点，则集群主机可能无法加入。

通常，最好在专用网络上让集群节点在专用网络上进行防火墙，以便它们间的通信。只能在公共访问点上启用防火墙，指向您的网络。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，其多播地址为 230.0.0.4。请注意，如果要为您的 JGroups 堆栈启用额外的功能，如诊断，您可能需要更多端口。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感兴趣（高可用性）、驱除、过期和缓存调整，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 有两个缓存类型。其中一种缓存位于数据库前面，以减少 DB 的负载，并通过将数据保留在内存中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。这个缓存是一个本地缓存。即使您位于带有更多 Red Hat Single Sign-On 服务器的集群中，本地缓存也不使用复制。相反，它们只会在本地保留副本，如果条目被更新为集群的其余部分，则会使条目被驱除。复制的缓存 工作 有单独的，该任务是将无效消息发送到整个集群，了解应从本地缓存驱除哪些条目。这可大大减少网络流量，提高了效率，避免通过线路传输敏感元数据。

第二个缓存处理管理用户会话、离线令牌以及跟踪登录故障，以便服务器能够检测密码中断和其他攻击。这些缓存中保存的数据仅在内存中临时，但可能在群集中复制。

本章讨论为集群和非集群部署的这些缓存的一些配置选项。

注意

这些缓存的更高级配置可以在 JBoss EAP 配置指南 的 Infinispan 部分中找到。

10.1. 驱除和过期

为红帽单点登录配置了多个不同缓存。有一个域缓存，它保存有关安全应用程序、常规安全数据和配置选项的信息。还有一个用户缓存，其中包含用户元数据。两个缓存都默认为 10000 个条目，并使用至少最近使用的驱除策略。它们也都与用于控制集群设置中驱除的对象修订缓存相关联。此缓存隐式创建，其配置的大小会两倍。同样适用于包含授权数据 的授权 缓存。密钥 缓存保存有关外部密钥的数据，不需要具有专用修订版本缓存。相反，它明确声明了 过期，因此这些密钥会定期过期，并强制定期从外部客户端或身份提供程序下载。

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，有 infinispan 子系统的部分，其类似于：

```
<subsystem xmlns="urn:jboss:domain:infinispan:12.0">
    <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>
```

要限制或扩展允许条目的数量，只需添加或编辑 对象 元素或特定缓存配置的 过期 元素。

此外，也有单独的缓存 会话、客户端会话、离线 会话、 offlineClientSessions、 loginFailures 和 actionTokens。这些缓存在集群环境中分布，默认大小不绑定。如果将它们绑定，则可能会丢失一些会话。 Red Hat Single Sign-On 本身在内部清除过期的会话，以避免在不限制的情况下增加这些缓存的大小。如果因为大量会话而导致内存问题，您可以尝试：

- 增加集群的大小（集群中更多节点意味着会话在节点间平均分配）
- 增加 Red Hat Single Sign-On 服务器进程的内存
- 减少所有者的数量，以确保缓存保存在一个位置。详情请查看 第 10.2 节 “复制和故障转移”
- 为分布式缓存禁用 l1-lifespan。如需了解更多信息，请参阅span 文档
- 减少会话超时，这可单独为 Red Hat Single Sign-On 管理控制台中的每个域进行。但是，这可能会对最终用户的可用性产生影响。如需了解更多详细信息， 请参阅超时。

有一个额外的复制缓存 work，它主要用于在集群节点之间发送消息；默认情况下也未绑定。但是，这个缓存不应该造成任何内存问题，因为这个缓存中的条目非常短。

10.2. 复制和故障转移

有如 会话、身份验证会话、离线会话、 login Failure 和一些其他缓存（更多详情 第 10.1 节 “驱除和过期”）等缓存，在使用集群时被配置为分布式缓存。条目不会复制到每一个节点，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，故障转移是什么，如果所有拥有数据的节点停机，则数据会永久丢失。默认情况下， Red Hat Single Sign-On 仅指定一个数据所有者。因此，如果一个节点停机，数据就会丢失。这通常意味着用户将被注销，并且必须再次登录。

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:12.0">
  <cache-container name="keycloak">
      <distributed-cache name="sessions" owners="2"/>
...
```

在这里，我们更改了它，因此至少两个节点都会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。如果您不小心，如果用户在节点停机时注销，则有一个所有者足够好，并且会避免复制。

提示

通常最好将您的环境配置为使用粘性会话的 loadbalancer。作为红帽单点登录服务器（提供特定请求）时，性能会很有用，通常是来自分布式缓存的数据所有者，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

您可以禁用 realm 或 user cache。

流程

1. 编辑发行版中的 standalone.xml、standalone-ha.xml 或 domain.xml 文件。

   这个文件的位置取决于您的 操作模式。下面是一个示例配置文件。

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

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

2. 将您要禁用的缓存的 **enabled** 属性设置为 **false**。

3. 重启您的服务器以使这个更改生效。

10.4. 在运行时清除缓存

您可以清除域缓存、用户缓存或外部公钥。

流程

1. 登录到 **Admin** 控制台。

2. 单击 **Realm Settings**。

3. 点 **Cache** 选项卡。

4. 清除域缓存、外部公钥的用户缓存或缓存。

注意

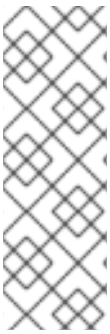
将为所有域清除缓存！

第 11 章 RED HAT SINGLE SIGN-ON OPERATOR

Red Hat Single Sign-On Operator 在 Openshift 中自动执行 Red Hat Single Sign-On 管理。您可以使用此 Operator 创建可自动化管理任务的自定义资源(CR)。例如，您可以在 Red Hat Single Sign-On admin 控制台中创建客户端或用户，而不必创建自定义资源来执行这些任务。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
- 安装扩展及主题



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这些资源。但是，您无法同时使用这两种方法，因为 Operator 为您修改的自定义资源执行一种同步方式。例如，如果您修改 realm 自定义资源，则 admin 控制台中显示的更改。但是，如果您使用 admin 控制台修改域，则这些更改不会影响自定义资源。

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

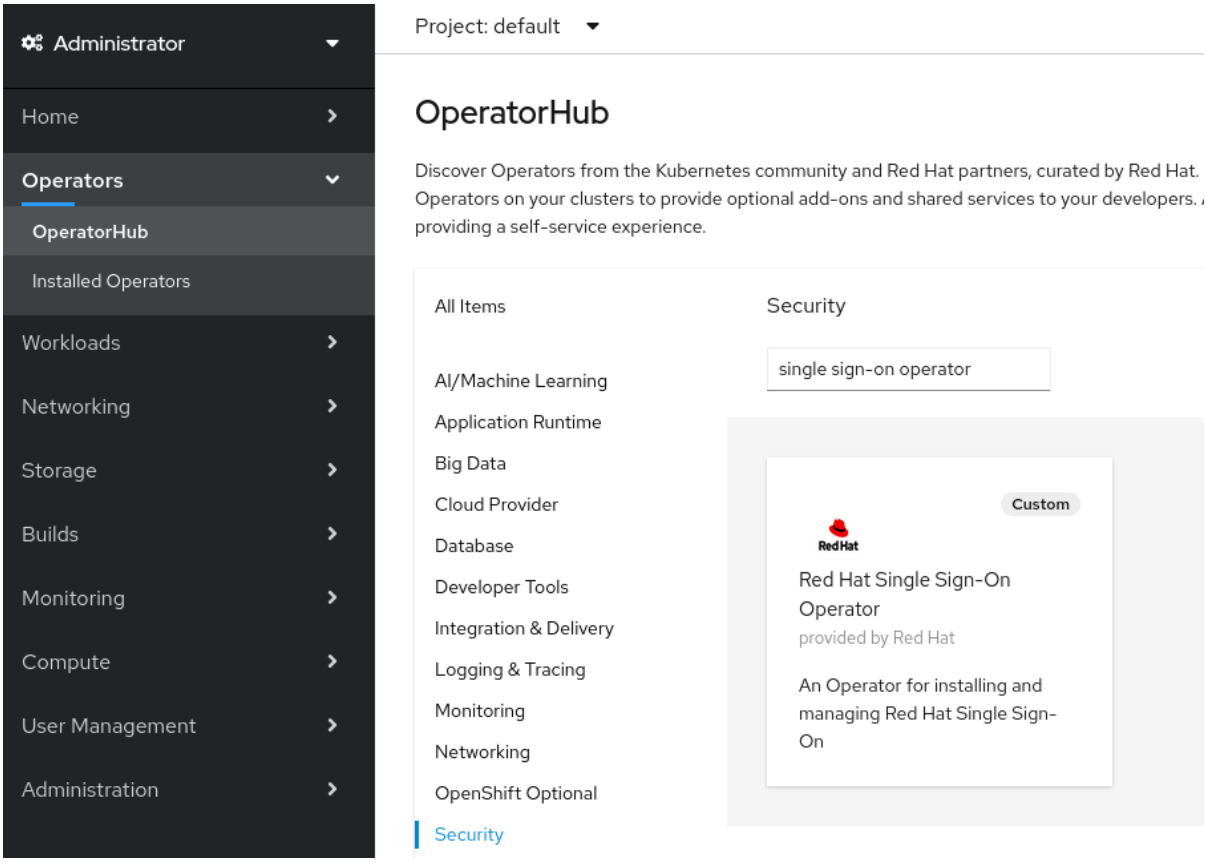
- 您有 cluster-admin 权限或管理员授予了同等权限的权限级别。

流程

在 OpenShift 集群上执行此步骤。

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

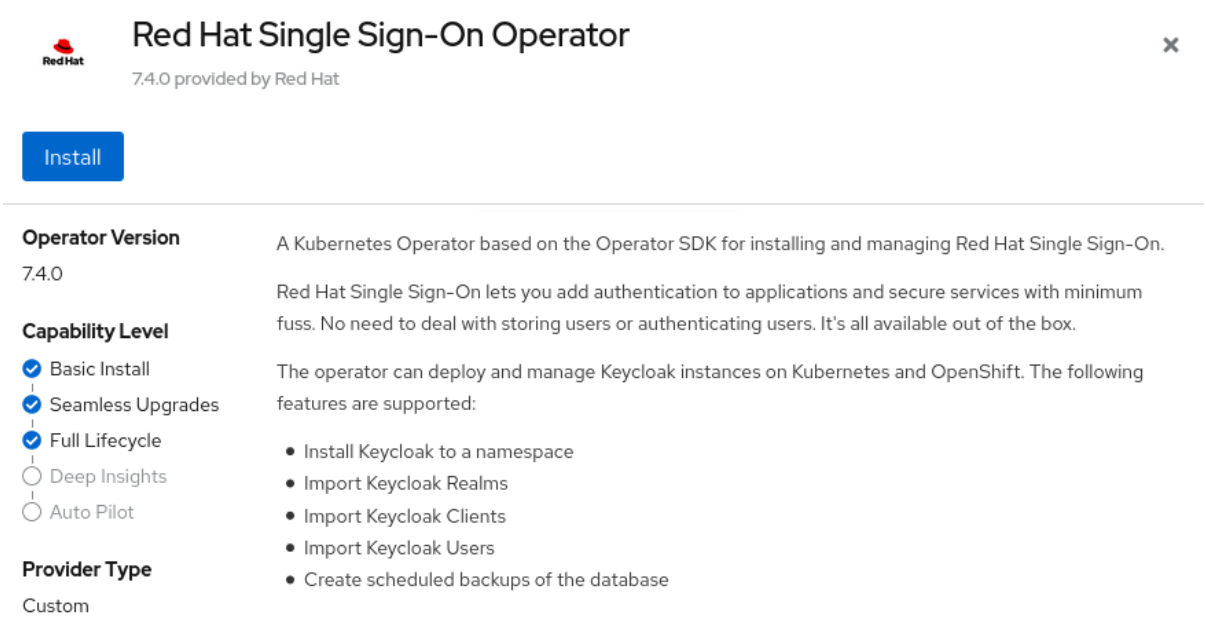
OpenShift 中的 OperatorHub 选项卡



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

   *此时会打开 **Install** 页面。*

Operator Install page on OpenShift

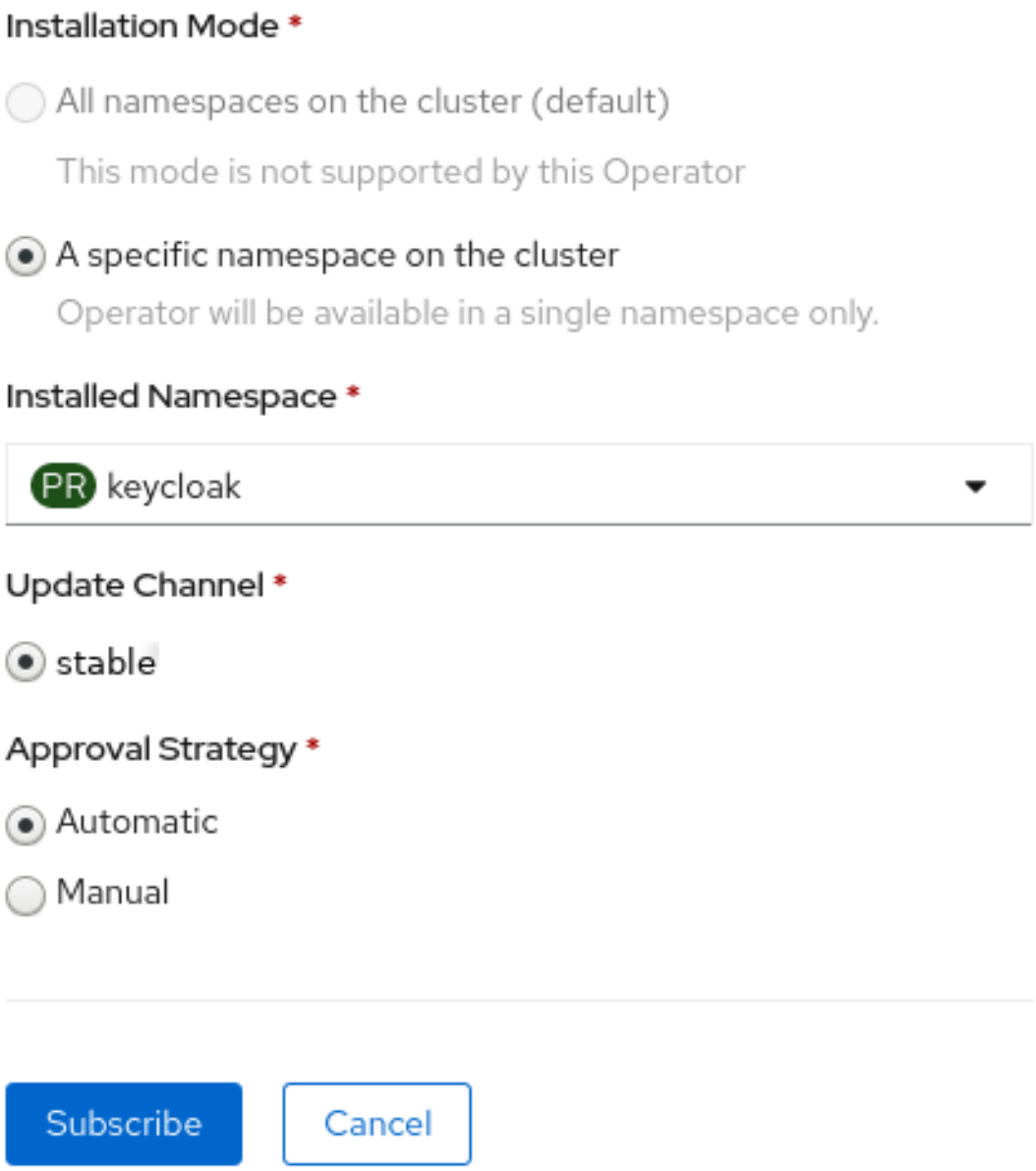


5. *点 Install。*

6. *选择一个命名空间并点 Subscribe。*

   确定您使用 stable 频道。

OpenShift 中的命名空间选择



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

- 您有 cluster-admin 权限或管理员授予了同等权限的权限级别。

流程

1. 创建一个项目。

```
$ oc new-project <namespace>
```

2. 创建名为 rhsso-operator-olm.yaml 的文件，以定义 YAML 组和订阅 operator。

将 targetNamespaces 更新为 RH-SSO 的命名空间。

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: rhsso-operator-group
spec:
  targetNamespaces:
  -  <namespace> # change this to the namespace you will use for RH-SSO
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: rhsso-operator
spec:
```

```
channel: stable
installPlanApproval: Manual
name: rhsso-operator
source: redhat-operators
sourceNamespace: openshift-marketplace
# Here you can specify a specific Operator version, otherwise it will use the latest
# startingCSV: rhsso-operator.7.6.0-opr-001
```

3. 安装 operator 组和订阅。

```
$ oc apply -f rhsso-operator-olm.yaml
```

4. 批准安装计划并填写适当的 命名空间。

```
oc patch installplan $(oc get ip -n <namespace>  -o=jsonpath='{.items[?(@.spec.approved==false)].metadata.name}') -n <namespace> --type merge --patch '{"spec":{"approved":true}}'
```

5. 验证 Operator 是否正在运行。

```
$ oc get pods
NAME                              READY   STATUS    RESTARTS   AGE
rhsso-operator-558876f75c-m25mt   1/1     Running   0          28s
```

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. 在生产环境中使用 RED HAT SINGLE SIGN-ON OPERATOR

- 在生产环境中不支持使用内嵌 DB。
- 备份 CRD 已被弃用，在生产环境中不被支持。
- Keycloak CR 中的 podDisruptionBudget 字段已弃用，并在 Kubernetes 版本 1.25 及更高版本上部署 Operator 时忽略。

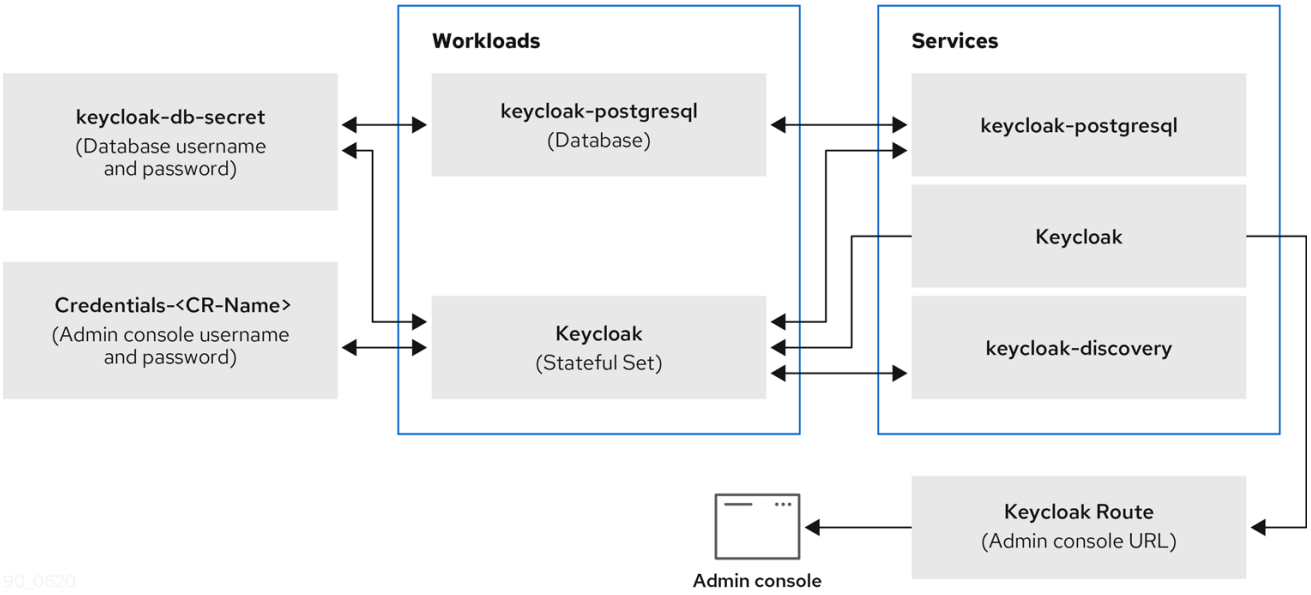
- 即使 v1alpha1 版本，我们完全支持在生产环境中使用其他 CRD。我们不计划在此 CRD 版本进行任何破坏更改。

11.3. 使用自定义资源安装 RED HAT SINGLE SIGN-ON

您可以通过创建 Keycloak 自定义资源，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，并在下图所示显示。

- Keycloak-db-secret - 存储属性，如数据库用户名、密码和外部地址（如果您连接到外部数据库）
- credentials-<CR-Name > - 用户名和密码登录到 Red Hat Single Sign-On admin 控制台（& lt;CR-Name > 基于 Keycloak 自定义资源名称）
- Keycloak - Keycloak 部署规格，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 - 连接到内部和外部数据库实例
- 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



11.3.1. Keycloak 自定义资源

Keycloak 自定义资源是一个 YAML 文件，它定义了用于安装的参数。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 则为 True，Operator 会为 Red Hat Single Sign-On 集群创建一个路由。您可以将 host 设置为覆盖自动选择的 Route 的主机名***
- ***externalDatabase - 以连接到外部托管数据库。本指南的 外部数据库 部分将涵盖该主题。将其设置为 false 时应用于测试目的，并将安装嵌入的 PostgreSQL 数据库。请注意，在生产环境中不支持 externalDatabase:false。***

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 控制台中出现的更改不会更新自定义资源。

11.3.2. 在 OpenShift 中创建 Keycloak 自定义资源

在 OpenShift 中，您可以使用自定义资源来创建路由，该路由是 admin 控制台的 URL，并找到包含 admin 控制台的用户名和密码。

先决条件

- 对此自定义资源有一个 YAML 文件。
- 您有 cluster-admin 权限或管理员授予了同等权限的权限级别。

流程

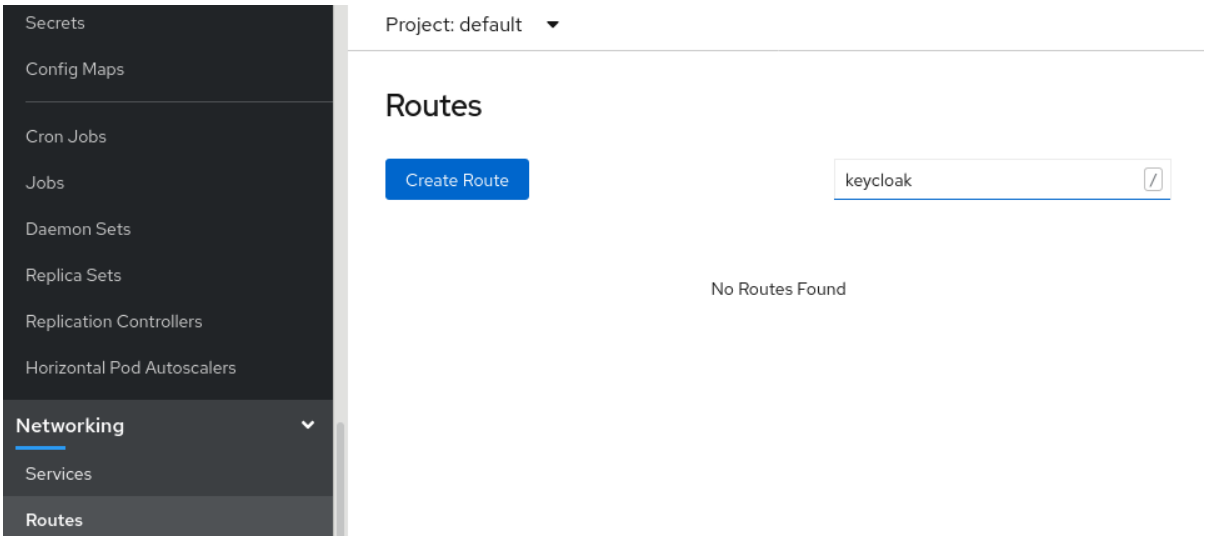
1. 使用您的 YAML 文件创建路由： oc create -f <filename>.yaml -n <namespace>.例如：

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

OpenShift 中创建了一个路由。

2. 登录到 OpenShift Web 控制台。
3. 选择 Networking,Routes 并搜索 Keycloak。

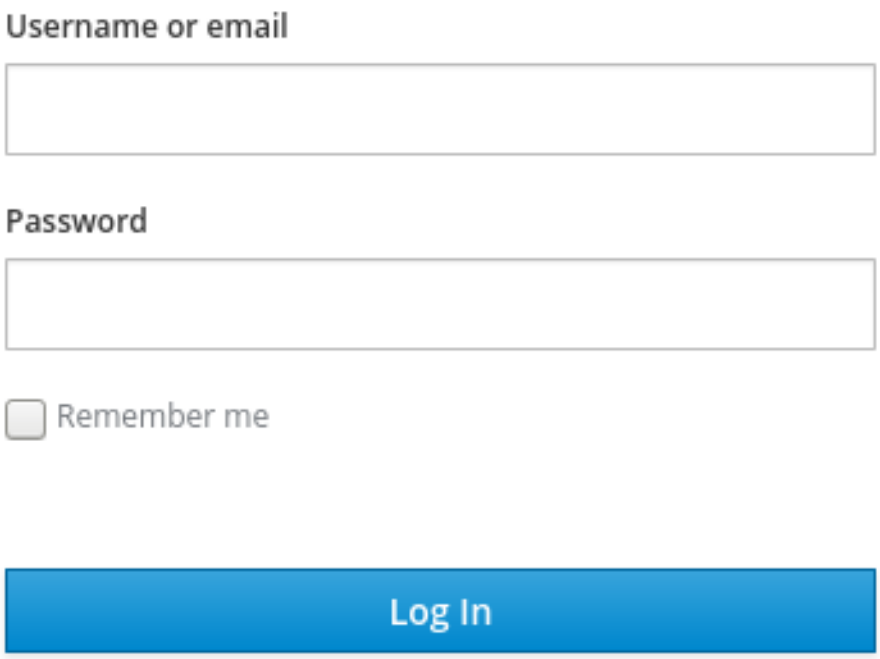
OpenShift Web 控制台中的路由屏幕



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

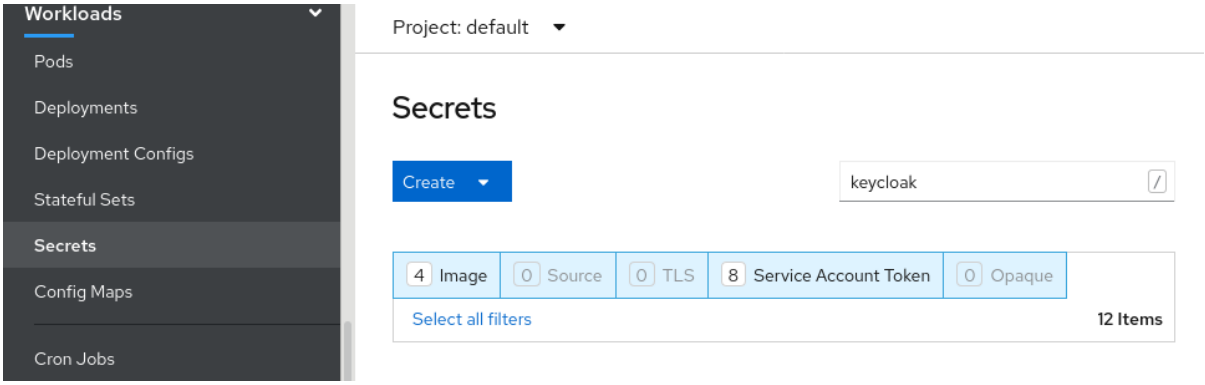
   此时会出现 Red Hat Single Sign-On admin 控制台登录屏幕。

管理控制台登录屏幕



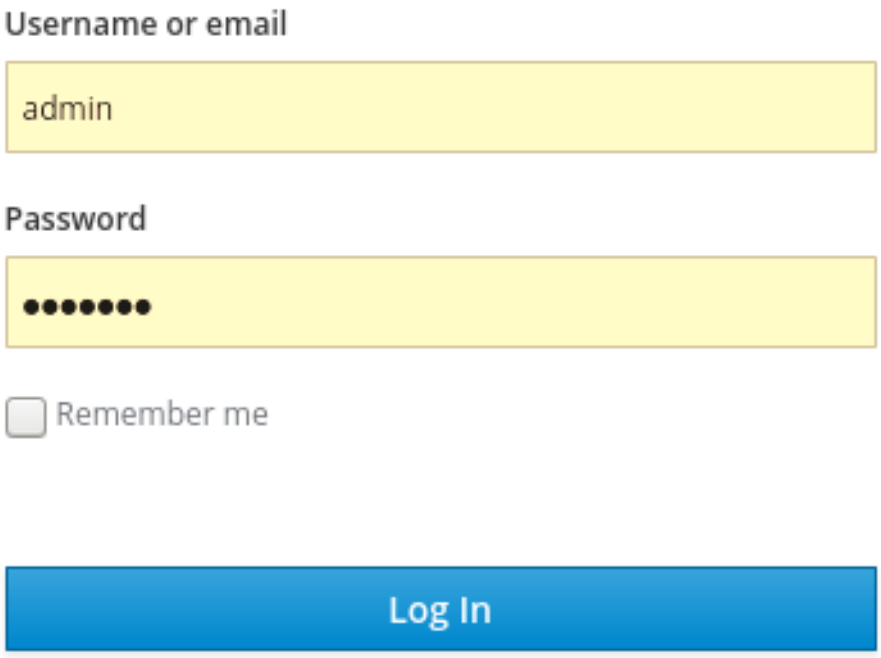
5. *在 OpenShift web 控制台中找到 admin 控制台的用户名和密码；在 Workloads 下，点 Secrets 并搜索 Keycloak。*

OpenShift Web 控制台中的机密页面



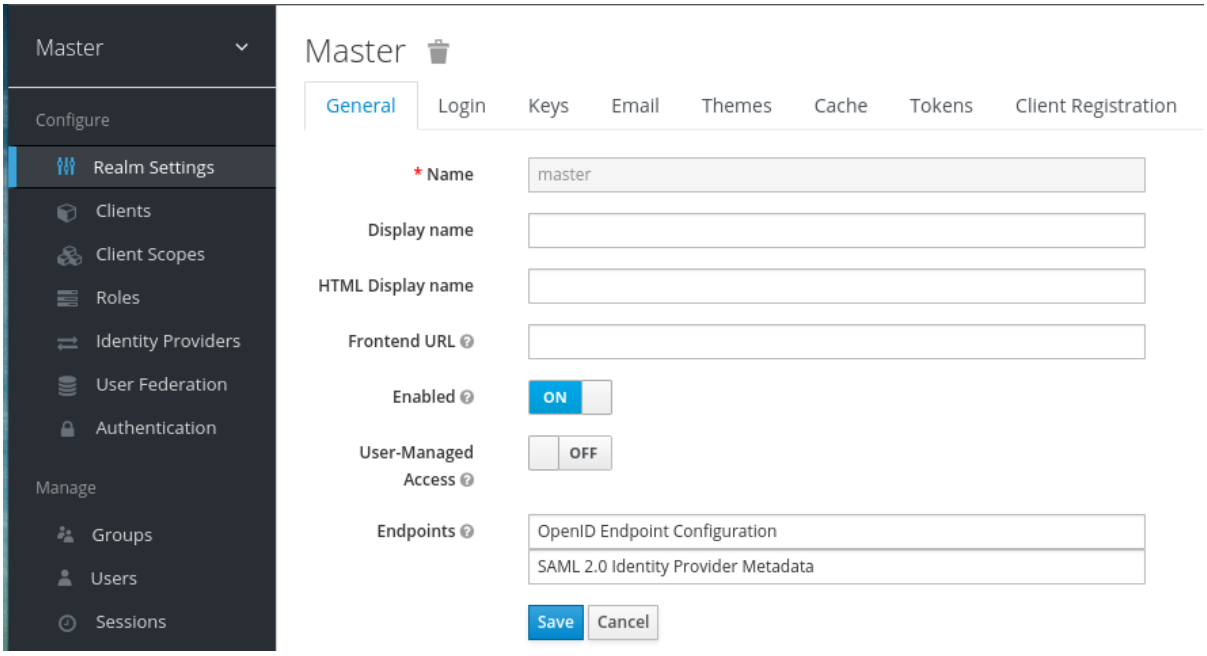
6. *在 admin 控制台登录屏幕中输入用户名和密码。*

管理控制台登录屏幕



您现在已登录到一个由 Keycloak 自定义资源安装的 Red Hat Single Sign-On 实例。您已准备好为 realm、client 和 users 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 创建一个域自定义资源。*
- *外部数据库是受支持的选项，需要在 Keycloak 自定义资源中启用。您只能只在测试时禁用这个选项，并在切换到生产环境时启用它。请参阅连接到外部数据库。*

11.4. 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

您只能通过创建或删除 YAML 文件来创建和删除域，且更改出现在 Red Hat Single Sign-On admin 控制台中。但是，对 admin 控制台的更改不会反映在域创建后 CR 的更改并不被支持。

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

- 对此自定义资源有一个 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 的正确实例中创建 realm。
- 您有 cluster-admin 权限或管理员授予了同等权限的权限级别。

流程

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

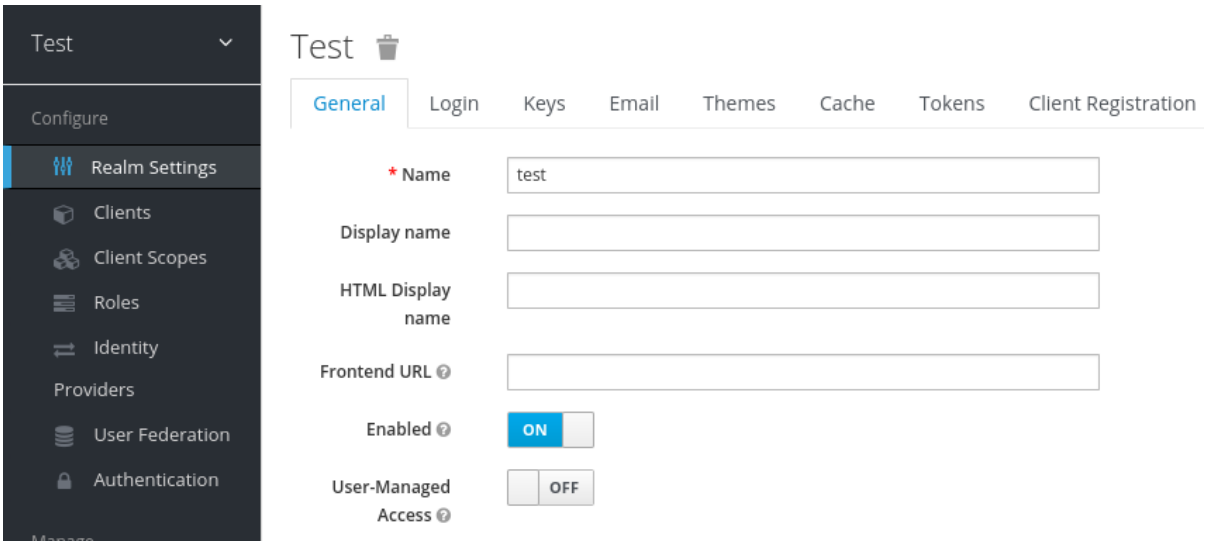
```
$ oc create -f initial_realm.yaml
keycloak.keycloak.org/test created
```

2. 登录到 Red Hat Single Sign-On 的相关实例的管理控制台。

3. *点 Select Realm，找到您创建的域。*

   新域将打开。

管理控制台 master 域



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。

11.5. 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 文件和更改出现在 Red Hat Single Sign-On admin 控制台中，但 admin 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

- *对此自定义资源有一个 YAML 文件。*
- *您有 cluster-admin 权限或管理员授予了同等权限的权限级别。*

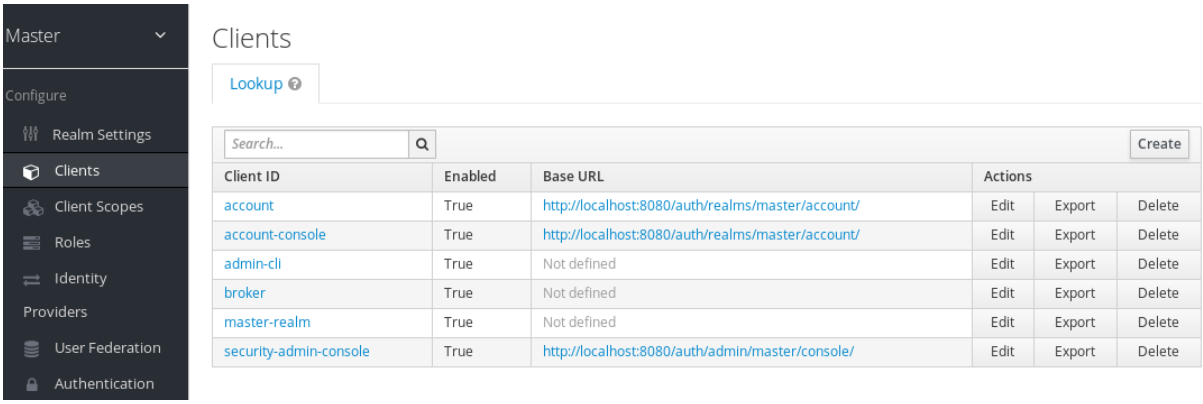
流程

1. *在您创建的 YAML 文件中使用此命令： oc create -f <client-name>.yaml.例如：*

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

2. *登录红帽单点登录相关实例的红帽单点登录管理控制台。*
3. *点 Clients。*

   新客户端会出现在客户端列表中。



结果

创建客户端后，Operator 会创建一个包含 Client ID 和客户端 secret 的 Secret，其命名模式：keycloak-client-secret-<custom 资源名称>。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
```

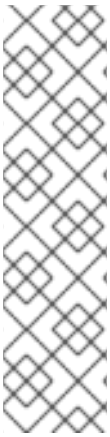
```
    Id:                  keycloak-client-secret
  Realm Selector:
    Match Labels:
      App:  sso
Status:
  Message:
  Phase:    reconciling
  Ready:    true
  Secondary Resources:
    Secret:
      keycloak-client-secret-client-secret
Events:  <none>
```

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。

11.6. 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 YAML 文件中的属性，更改会出现在 Red Hat Single Sign-On 管理控制台中，但对 admin 控制台的更改不会更新自定义资源。

请注意，同样适用于凭据。如果定义了 credentials 字段，则用户的凭证始终与 CR 中设置的值匹配。换句话说，如果用户通过帐户控制台更改密码，Operator 将重置它以匹配 CR 中设置的值。

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
  credentials:
    - type: "password"
      value: "12345"
  realmRoles:
    - "offline_access"
  clientRoles:
    account:
      - "manage-account"
    realm-management:
      - "manage-users"
realmSelector:
  matchLabels:
    app: sso
```

先决条件

- *对此自定义资源有一个 YAML 文件。*
- *realmSelector 与现有 realm 自定义资源的标签匹配。*
- *您有 cluster-admin 权限或管理员授予了同等权限的权限级别。*

流程

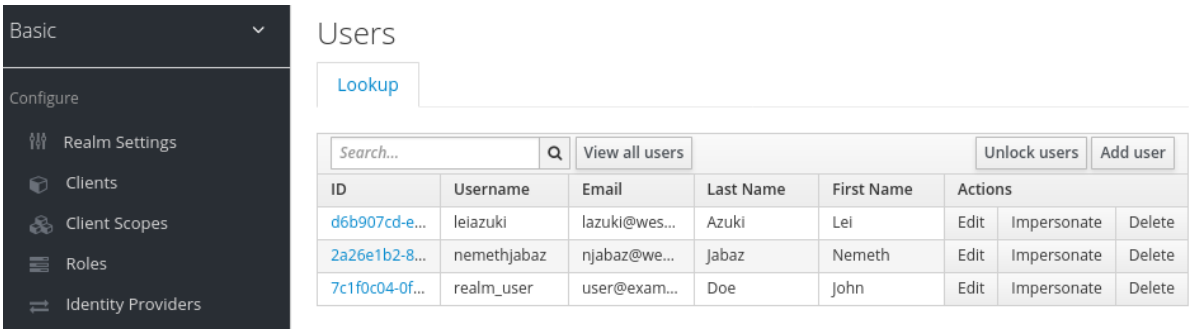
1. *在您创建的 YAML 文件中使用此命令： oc create -f <user_cr>.yaml.例如：*

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

2. *登录到 Red Hat Single Sign-On 的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。



结果

创建用户后，Operator 会使用以下命名模式创建一个 Secret：credential-<realm name>-<username>-<namespace >，其中包含用户名，并在 CR credentials 属性中指定了密码。

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。

11.7. 连接到外部数据库

您可以通过创建一个 keycloak-db-secret YAML 文件并将 Keycloak CR externalDatabase 属性设置为启用，来使用 Operator 连接到外部 PostgreSQL 数据库。请注意，数值采用 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 URL (resolvable by K8s)>
  POSTGRES_EXTERNAL_PORT: <External Database Port>
  POSTGRES_PASSWORD: <Database Password>
  # Required for AWS Backup functionality
  POSTGRES_SUPERUSER: "true"
  POSTGRES_USERNAME: <Database Username>
  SSLMODE: <TLS configuration for the Database connection>
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 - 外部数据库的主机名。如果您的数据库只有一个 IP 而不是主机名，则创建一个 服务和相应的 EndpointSlice 或 Endpoint 来提供主机名。
- POSTGRES_EXTERNAL_PORT - （可选）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。按照如下所示设置它们：

- POSTGRES_DATABASE - 要使用的数据库名称。
- 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。
- SSLMODE - 指示是否在连接到外部 PostgreSQL 数据库的连接上使用 TLS。检查 可能的值

启用 SSLMODE 后，Operator 会搜索名为 keycloak-db-ssl-cert-secret 的 secret，其中包含 PostgreSQL 数据库使用的 root.crt。创建 secret 是可选的，只有在您要验证数据库的证书时使用 secret（例如 SSLMODE: verify-ca）。下面是一个示例：

供 Operator 使用的 TLS Secret 示例。

```
apiVersion: v1
kind: Secret
metadata:
  name: keycloak-db-ssl-cert-secret
  namespace: keycloak
type: Opaque
data:
  root.crt: {root.crt base64}
```

Operator 将创建名为 keycloak-postgresql 的 Service。该服务由 Operator 配置，以根据 POSTGRES_EXTERNAL_ADDRESS 的内容公开外部数据库。Red Hat Single Sign-On 使用此服务连接到数据库，这意味着它不直接连接到数据库，而是通过这个服务。

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 Keycloak 自定义资源，将 externalDatabase 设置为 true。
- 您有 cluster-admin 权限或管理员授予了同等权限的权限级别。

流程

1. 为您的 PostgreSQL 数据库找到 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. 确认已解码的值与您的数据库的 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，以确认它们将 IP 地址用于您的数据库：

```
$ 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
```

11.8. 连接到外部 RED HAT SINGLE SIGN-ON

此运算符也可用于部分管理外部红帽单点登录实例。在当前状态中，它将只能创建客户端。

要做到这一点，您需要创建 Keycloak 和 KeycloakRealm CRD 的非受管版本以用于目标和配置。

external-keycloak的 YAML 文件示例

```
apiVersion: keycloak.org/v1alpha1
kind: Keycloak
metadata:
  name: external-ref
  labels:
    app: external-sso
spec:
  unmanaged: true
  external:
    enabled: true
    url: https://some.external.url
```

为了根据这个 keycloak 进行身份验证，Operator 通过带有 凭证的 CRD 名称来推断来自 CRD 的 secret 名称。

credentials -external-ref的 YAML 文件示例

```
apiVersion: v1
kind: Secret
metadata:
  name: credential-external-ref
type: Opaque
data:
  ADMIN_USERNAME: YWRtaW4=
  ADMIN_PASSWORD: cGFzcw==
```

external-realm的 YAML 文件示例

```
apiVersion: keycloak.org/v1alpha1
kind: KeycloakRealm
metadata:
  name: external-realm
  labels:
    app: external-sso
spec:
  unmanaged: true
  realm:
    id: "basic"
    realm: "basic"
  instanceSelector:
    matchLabels:
      app: external-sso
```

现在，您可以照常使用客户端中的域引用，它将在外部红帽单点登录实例上创建客户端。

11.9. 调度数据库备份

警告

备份 CR 已被弃用，并可能在以后的版本中被删除。

您可以使用 Operator 来调度由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。

备份自定义资源的 YAML 文件示例

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

先决条件

- *对此自定义资源有一个 YAML 文件。*
- *您有一个带有 claimRef 的 PersistentVolume，它只适用于 Red Hat Single Sign-On Operator 创建的 PersistentVolumeClaim。*

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-postresql-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. 查看完成的备份作业的日志：

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

其他资源

- 如需有关持久性卷的详情，请参阅了解持久性存储。

11.10. 安装扩展及其

您可以使用 Operator 安装扩展功能，并适用于您的公司或组织所需的扩展。扩展或主题可以是 Red Hat Single Sign-On 都可以使用的任何内容。例如，您可以添加指标扩展。您可以在 Keycloak 自定义资

源中添加扩展或主题。

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
```

您可以像任何其他扩展一样打包和部署它们。如需更多信息，请参阅 部署主题 手动条目。

先决条件

- 有一个用于 Keycloak 自定义资源的 YAML 文件。
- 您有 cluster-admin 权限或管理员授予了同等权限的权限级别。

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. 在 JAR 文件中为自定义扩展或主题添加 URL。
4. 保存该文件。

Operator 下载扩展名或主题并安装它。

11.11. 用于管理自定义资源的命令选项

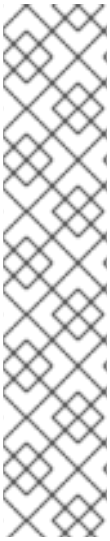
在创建自定义请求后，您可以使用 oc 命令编辑它或删除。

- 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 的域自定义请求，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 Keycloak CR YAML 文件，更改将应用到部署。

对其他资源的更新有限：

Keycloak Realm CR 只支持基本创建和删除，而无需同步选项。Keycloak 用户和客户端 CR 支持单向更新（切换到 CR）反映在 Keycloak 中，但没有在 CR 中更新 Keycloak 的更改。

11.12. 升级策略

您可以配置 Operator 如何执行红帽单点登录升级。您可以从以下升级策略中选择。

- recreate ：这是默认策略。Operator 删除所有 Red Hat Single Sign-On 副本，可选创建备份，然后根据更新的 Red Hat Single Sign-On 镜像创建副本。此策略适用于升级，因为单个红

帽单点登录版本正在访问底层数据库。不足之处在于在升级过程中需要关闭 Red Hat Single Sign-On。

- 滚动：Operator 一次移除一个副本，并根据更新的 Red Hat Single Sign-On 镜像再次创建副本。这样可保证零停机时间升级，但更适合于不需要数据库迁移的次版本，因为多个红帽单点登录版本同时访问数据库。此策略不支持自动备份。

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: 2
  migration:
    strategy: recreate
    backups:
      enabled: True
  externalAccess:
    enabled: True
```

注意

由于之前版本的 Operator 中存在一个错误，Red Hat Single Sign-On StatefulSet 上的 Selector 字段可能会根据您的配置被错误配置。如果 Operator 检测到这样的状态，且您使用 recreate 策略，它将 删除并重新创建带有正确的 Selector 字段的 StatefulSet。这是必要的，因为 Selector 字段是不可变的。

因为一个"删除"操作在非常罕见的情况下可能会具有潜在的危险性副作用，例如，当您将 Operator 增加了自定义功能未知时，您可以手动删除 StatefulSet 定义。