Red Hat build of Keycloak 26.2

服务器配置指南

Last Updated: 2025-10-25

Legal Notice

Abstract

本指南由管理员配置 Keycloak 26.2 的红帽构建信息组成。

Table of Contents

第 1 章 配置红帽构建的 KEYCLOAK

配置并启动红帽构建的 Keycloak。

本章介绍了红帽构建的 Keycloak 的配置方法，以及如何启动和应用首选配置。它包括优化红帽构建的 Keycloak 的配置指南，以便更快地启动和低内存占用。

1.1. 为红帽构建的 KEYCLOAK 配置源

红帽构建的 Keycloak 从四个源加载配置，按应用程序顺序列出。

1. 命令行参数
2. 环境变量
3. 在 **conf/keycloak.conf** 文件中定义的选项，或者在用户创建的配置文件中定义。
4. 在用户创建的 Java KeyStore 文件中定义的敏感选项。

在多个源中设置选项时，列表中的第一个选项决定了该选项的值。例如，命令行参数设置的选项的值的优先级高于同一选项的环境变量。

1.1.1. 示例：配置 db-url-host 参数

以下示例演示了如何在四个配置源中设置 **db-url** 值：

Source	格式
命令行参数	**--db-url=cliValue**
环境变量	**KC_DB_URL=envVarValue**
配置文件	**db-url=confFileValue**
Java KeyStore file	**kc.db-url=keystoreValue**

根据应用的优先级，启动时使用的值是 **cliValue**，因为命令行是最高优先级。

如果没有使用 **--db-url=cliValue**，应用的值将为 **KC_DB_URL=envVarValue**。如果值不是由命令行或环境变量应用，则使用 **db-url=confFileValue**。如果没有应用前面的值，则使用值 **kc.db-url=keystoreValue**，因为可用配置源中的优先级最低。

1.2. 配置格式

配置使用统一的 *每源* 格式，它简化了从一个配置源到另一个配置源的键/值对转换。请注意，这些格式也适用于 spi 选项。

命令行参数格式

命令行的值使用-- < ***key-with-dashes> = <value>*** 格式。对于某些值，a **- <*abbreviation*> = <*value>***, 简写也存在。

环境变量格式

环境变量的值使用大写的 **KC_ *<key_with_underscores> = <value>*** 格式。

配置文件格式

进入配置文件的值使用 < ***key-with-dashes> = <value>*** 格式。

密钥存储配置文件格式

进入 KeyStore 配置文件的值使用 **kc. *<key-with-dashes>*** 格式。 ***<value*** > 然后是存储在 KeyStore 中的密码。

在每个配置章节的末尾，查找 Relevant *选项*标题，该标题定义适用的配置格式。有关所有配置选项，请查看 所有配置。选择适用于您的用例的配置源和格式。

1.2.1. 示例 - 基于配置源的替代格式

以下示例显示了三个配置源的 **db-url-host** 的配置格式：

命令行参数

```
bin/kc.[sh|bat] start --db-url-host=mykeycloakdb
```

环**境**变**量**

```
export KC_DB_URL_HOST=mykeycloakdb
```

conf/keycloak.conf

```
db-url-host=mykeycloakdb
```

1.2.2. 命令行参数的格式

Red Hat build of Keycloak 是与很多命令行参数一起打包的进行配置。要查看可用的配置格式，请输入以下命令：

```
bin/kc.[sh|bat] start --help
```

或者，请参阅 所有服务器选项的所有配置。

1.2.3. 环境变量的格式

您可以使用 **${ENV_VAR}** 语法使用占位符从 **keycloak.conf** 文件中的环境变量解析特定于环境的值：

```
db-url-host=${MY_DB_HOST}
```

如果无法解析环境变量，您可以指定一个回退值。使用 **:** (colon)，如 **mydb** 之前所示：

```
db-url-host=${MY_DB_HOST:mydb}
```

1.2.4. 包含特定配置文件的格式

默认情况下，服务器始终从 **conf/keycloak.conf** 文件中获取配置选项。对于新安装，此文件仅包含注释设置，作为您在生产中运行时要设置的内容。

您还可以通过输入以下命令来使用 **[-cf|--config-file]** 选项指定显式配置文件位置：

```
bin/kc.[sh|bat] --config-file=/path/to/myconfig.conf start
```

设置该选项可让红帽构建来自指定文件中的 Keycloak 的读取配置，而不是 **conf/keycloak.conf**。

1.2.5. 使用 Java KeyStore 文件设置敏感选项

由于 Keystore Configuration Source，您可以使用 **[--config-keystore]** 和 **[--config-keystore-password]** 选项直接从 Java KeyStore 加载属性。另外，您可以使用 **[--config-keystore-type]** 选项指定 KeyStore 类型。默认情况下，KeyStore 类型是 **PKCS12**。

KeyStore 中的 secret 需要使用 **PBE** （基于密码的加密）密钥算法存储，其中密钥是从 KeyStore 密码衍生而来。您可以使用以下 **keytool** 命令生成这样的 KeyStore：

```
keytool -importpass -alias kc.db-password -keystore keystore.p12 -storepass keystorepass -
storetype PKCS12 -v
```

执行该命令后，系统将提示您输入要 **存储的密码，这代表** 上面的 **kc.db-password** 属性的值。

创建 KeyStore 时，您可以使用以下参数启动服务器：

```
bin/kc.[sh|bat] start --config-keystore=/path/to/keystore.p12 --config-keystore-password=keystorepass
--config-keystore-type=PKCS12
```

1.2.6. 原始 Quarkus 属性的格式

在大多数情况下，可用的配置选项应该足以配置服务器。但是，对于红帽构建的 Keycloak 配置中缺少的特定行为或功能，您可以使用底层 Quarkus 框架中的属性。

如果可能，请避免直接使用 Quarkus 中的属性，因为红帽构建的 Keycloak 不支持它们。如果您需要至关重要，请考虑先打开一个 增强请求。这种方法有助于我们改进红帽构建的 Keycloak 配置以满足您的需要。

如果无法增强请求，您可以使用原始 Quarkus 属性配置服务器：

1. 在 **conf** 目录中创建 **quarkus.properties** 文件。
2. 在该文件中定义必要属性。
   您只能使用 Quarkus 文档 中定义的 Quarkus 扩展 的子集。另外，请注意 Quarkus 属性的不同：
   - Quarkus 文档中的 Quarkus 属性的锁定图标表示构建时间属性。您可以运行 **build** 命令来应用此属性。有关 build 命令的详情，请查看以下部分来优化红帽构建的 Keycloak。
   - Quarkus 指南中的属性没有锁定图标表示 Quarkus 和 Red Hat build of Keycloak 的运行时属性。

您还可以在 Java KeyStore 中存储 Quarkus 属性。

请注意，一些 Quarkus 属性已在红帽构建的 Keycloak 配置中映射，如 **quarkus.http.port** 和类似基本属性。如果红帽构建的 Keycloak 使用属性，在 **quarkus.properties** 中定义该属性键无效。Red Hat build of Keycloak 配置值优先于 Quarkus 属性值。

1.2.7. 在值中使用特殊字符

红帽构建的 Keycloak 依赖于 Quarkus 和 MicroProfile 来处理配置值。请注意，支持值表达式。例如，**${some_key}** 评估为 **some_key** 的值。

要禁用表达式评估，\ 字符作为转义字符。特别是，当它们似乎定义表达式或重复时，必须使用它来转义 **$** 字符的使用。例如，如果您希望配置值 **my$password**，请改为使用 **my\$\$password**。请注意，在使用大多数 unix shell 时，\ 字符需要额外的转义或引用，或者在属性文件中出现时。例如，bash 单引号保留单引号 **--db-password='my\$\$password'**。此外，使用 bash 双引号时，您需要额外的 backslash **--db-password="my\\$\\$password"**。在属性文件中类似，反斜杠字符也必须被转义：kc **.db-password=my\\$\\$password**

特定于 Windows 的注意事项

在配置值中指定 Windows 文件路径时，反斜杠也必须被转义。例如，如果要指定路径 **C:\path\to\file**，您应该将其写为 **C:\\path\\to\\file**。或者，您可以使用不需要转义的正斜杠： **C:/path/to/file**。

当使用 PowerShell 和值包含像逗号一样的特殊字符时，请使用双引号：

```
.\kc.bat start --log-level="'INFO,org.hibernate:debug'"
```

PowerShell 以不同的方式处理引号。它将解释引号的字符串，然后再将其传给 **kc.bat** 脚本，从而删除外部引用字符。因此，需要额外的引号层来保留值结构。否则，逗号将解释为分隔符。在 Windows CMD 中，您可以直接使用双引号。

1.3. 启动 RED HAT BUILD OF KEYCLOAK

您可以在 **开发** 模式中启动红帽构建的 Keycloak 或 **生产环境模式**。每个模式为预期的环境提供不同的默认值。

1.3.1. 在开发模式下启动 Red Hat build of Keycloak

使用开发模式来首次尝试红帽构建的 Keycloak，以便快速启动并运行。这个模式为开发人员提供方便的默认值，比如开发一个新的 Red Hat build of Keycloak 主题。

要在开发模式中启动，请输入以下命令：

```
bin/kc.[sh|bat] start-dev
```

默认值

开发模式设置以下默认配置：

- 启用 HTTP
- 禁用严格的主机名解析
- 缓存设置为 local （没有用于高可用性的分布式缓存机制）
- 禁用主题缓存和模板缓存

1.3.2. 在生产环境模式下启动 Red Hat build of Keycloak

使用 production 模式在生产环境中部署红帽构建的 Keycloak。这个模式 *默认遵循一个安全* 原则。

要在生产环境模式下启动，请输入以下命令：

```
bin/kc.[sh|bat] start
```

如果没有进一步配置，这个命令将不会启动红帽构建的 Keycloak，并会显示错误。这个响应是根据目的完成的，因为红帽构建的 Keycloak 遵循 *一个安全* 原则。生产环境模式需要设置主机名，并在启动后使用 HTTPS/TLS 设置。

默认值

production 模式设置以下默认值：

- HTTP 已作为传输层安全(HTTPS)被禁用。
- 主机名配置是预期的
- HTTPS/TLS 配置是预期的

在生产环境中部署 Red Hat build of Keycloak 前，请确保按照 为生产环境配置 红帽构建的 Keycloak 中介绍的步骤操作。

默认情况下，production 模式的配置选项示例会在默认的 **conf/keycloak.conf** 文件中被注释掉。这些选项可让您了解在生产环境中运行红帽构建的 Keycloak 时需要考虑的主要配置。

1.4. 创建初始管理员用户

您可以使用 web 前端创建初始 admin 用户，您可以使用本地连接(localhost)进行访问。您可以使用环境变量创建此用户。为初始 admin ***用户名设置 KC_BOOTSTRAP_ADMIN_USERNAME= &*** lt;username>，为初始 admin ***密码设置 KC_BOOTSTRAP_ADMIN_PASSWORD= &*** lt;password>。

Red Hat build of Keycloak 在第一次启动时解析这些值，以创建具有管理权限的初始用户。当第一个具有管理权限的用户存在后，您可以使用管理控制台或命令行工具 **kcadm.[sh|bat]** 来创建其他用户。

如果初始管理员已存在并且环境变量启动时仍然存在，则日志中会显示一条错误消息，指出初始管理员创建失败。Red Hat build of Keycloak 忽略了值并正确启动。

1.5. 优化 RED HAT BUILD OF KEYCLOAK 启动

我们建议优化红帽构建的 Keycloak，以便在生产环境中部署红帽构建的 Keycloak 前更快地提供启动和更好的内存消耗。本节论述了如何应用红帽构建的 Keycloak 优化，以获得最佳性能和运行时行为。

1.5.1. 创建优化的红帽构建的 Keycloak 构建

默认情况下，当您使用 **start** 或 **start-dev** 命令时，红帽构建的 Keycloak 会根据覆盖运行 **build** 命令。

此 **构建** 命令为启动和运行时行为执行一组优化。构建过程可能需要几秒钟。特别是在容器化环境中运行红帽构建的 Keycloak （如 Kubernetes 或 OpenShift）时，启动时间非常重要。为避免丢失这一时间，请在启动前明确 **运行构建**，如 CI/CD 管道中的独立步骤。

1.5.1.1. 第一步：明确运行构建

要运行 构建，请输入以下命令：

```
bin/kc.[sh|bat] build <build-options>
```

此命令显示 您输入的构建选项。Red Hat build of Keycloak 可区分运行 build 命令时可以使用的构建选项，这些选项 在启动服务器时可用。

对于红帽构建的 Keycloak 的非优化启动，这种区别没有影响。但是，如果您在启动前运行构建，则 build 命令只能使用一组选项。限制的原因是构建选项被保留至优化的红帽构建的 Keycloak 镜像。例如，出于安全原因，必须保留 db-password （即配置选项）等凭据的配置。

警告

所有构建选项都以纯文本形式保留。不要将任何敏感数据存储为构建选项。这适用于所有可用的配置源，包括 KeyStore 配置源。因此，我们不推荐将任何构建选项存储在 Java 密钥存储中。另外，当涉及配置选项时，我们建议使用 KeyStore 配置源来存储敏感数据。对于非敏感数据，您可以使用剩余的配置源。

构建选项会在 All configuration with a tool icon 中标记。要查找可用的构建选项，请输入以下命令：

```
bin/kc.[sh|bat] build --help
```

示例： 运行构建 以在启动前将数据库设置为 PostgreSQL

```
bin/kc.[sh|bat] build --db=postgres
```

1.5.1.2. 第二步：使用 优化启动红帽构建的 Keycloak

成功构建后，您可以启动红帽构建的 Keycloak，并输入以下命令关闭默认启动行为：

```
bin/kc.[sh|bat] start --optimized <configuration-options>
```

- optimized 参数告知红帽构建 Keycloak，以假定使用预建的、已优化的红帽构建的 Keycloak 镜像。因此，红帽构建的 Keycloak 可以避免在启动时直接检查并运行构建，这可以节省时间。

您可以在启动时输入所有配置选项；这些选项是所有配置中 没有通过 工具图标标记的选项。???

- 如果在启动时找到构建选项，其值等于输入 构建 时使用的值，则在使用 - optimized 参数时，该选项会 静默忽略。
- 如果该选项的值与输入构建时使用的值不同，日志中会出现警告，并使用之前构建的值。要使这个值生效，请在启动前运行 新构建。

创建优化的构建

以下示例显示了创建优化的构建，并在启动红帽构建的 Keycloak 时使用 - optimized 参数。

1. 使用 build 命令为 PostgreSQL 数据库厂商设置构建选项
   -

```
bin/kc.[sh|bat] build --db=postgres
```

2. 在 conf/keycloak.conf 文件中为 postgres 设置运行时配置选项。

```
db-url-host=keycloak-postgres
db-username=keycloak
db-password=change_me
hostname=mykeycloak.acme.com
https-certificate-file
```

3. 使用优化参数启动服务器

```
bin/kc.[sh|bat] start --optimized
```

您可以使用 build 命令实现大多数启动和运行时行为的优化。另外，通过将 keycloak.conf 文件用作配置源，您可以避免一些启动时需要命令行参数的步骤，如初始化 CLI 本身。因此，服务器启动速度更快。

1.6. 在域配置中使用系统变量

管理员可通过一些域功能在配置域及其组件时引用系统变量，如环境变量和系统属性。

默认情况下，Red Hat build of Keycloak 不允许使用系统变量，但只有通过 spi-admin-allowed-system-variables 配置选项明确指定的变量。此选项允许您指定一个以逗号分隔的键列表，这些键最终会使用相同的键从系统变量解析为值。

1. 启动服务器并将一组系统变量公开给服务器运行时

```
bin/kc.[sh|bat] start --spi-admin-allowed-system-variables=FOO,BAR
```

在以后的发行版本中，这个功能将被删除，而是防止在域配置中使用系统变量。

1.7. 底层概念

本节概述红帽构建的 Keycloak 使用的底层概念，特别是当它涉及优化启动时。

Red Hat build of Keycloak 使用 Quarkus 框架和覆盖下的 re-augment/mutable-jar 方法。此过程将在运行 build 命令时启动。

以下是 build 命令执行的一些优化：

- 创建了有关已安装供应商的新封闭假设，这意味着不需要重新创建 registry，并在每次红帽构建的 Keycloak 启动时初始化因素。
- 配置文件被预先解析，以便在启动服务器时减少 I/O。
- 配置数据库特定资源，并准备好针对特定的数据库供应商运行。
- 通过将构建选项持久化到服务器镜像中，服务器不会执行任何其他步骤来解释配置选项和（重新）配置其自身。

您可以在特定的 Quarkus 指南中了解更多信息

第 2 章 为生产环境配置 RED HAT BUILD OF KEYCLOAK

准备红帽构建的 Keycloak，以在生产环境中使用。

红帽构建的 Keycloak 生产环境为从内部部署部署提供安全身份验证和授权，这些部署支持几千名用户的部署。

本章论述了生产就绪的红帽构建 Keycloak 环境所需的一般配置区域。这些信息着重介绍常规概念，而不是实际实施，这取决于您的环境。本章涵盖的关键方面适用于所有环境，无论是容器化、内部、GitOps 或 Ansible。

2.1. TLS 用于安全通信

红帽构建的 Keycloak 持续交换敏感数据，这意味着所有与红帽构建的 Keycloak 的通信都需要安全通信频道。要防止几个攻击向量，您可以通过 TLS 或 HTTPS 为该频道启用 HTTP。

要为红帽构建的 Keycloak 配置安全通信频道，请参阅配置 TLS 和配置传出 HTTP 请求。

要保护红帽构建的 Keycloak 的缓存通信，请参阅配置分布式缓存。

2.2. 红帽构建的 KEYCLOAK 的主机名

在生产环境中，红帽构建的 Keycloak 实例通常在专用网络中运行，但红帽构建的 Keycloak 需要公开某些面向公共的端点，以便与应用程序进行通信。

有关端点类别以及如何为其配置公共主机名的说明，请参阅 配置主机名(v2)。

2.2.1. 在不同的主机名上公开红帽 Keycloak 管理 API 和 UI

最佳实践是公开红帽构建 Keycloak 管理 REST API，并在不同的主机名或上下文路径上公开红帽构建的 Keycloak 管理 REST API 和控制台，而不是通过登录流程使用的公共前端 URL。这种分离可确保管理界面不会公开给公共互联网，从而减少攻击面。

警告

如果不希望公开代理，则需要在反向代理级别上阻止对 REST API 的访问。

详情请参阅 配置主机名(v2)。

2.3. 分布式环境中的反向代理

除了 配置主机名(v2)外，生产环境通常包括反向代理/负载均衡器组件。它分离并统一访问您公司或组织所使用的网络。对于红帽构建的 Keycloak 生产环境，建议此组件。

有关在红帽构建的 Keycloak 中配置代理通信模式的详情，请参考配置反向代理。本章还建议在公共访问中隐藏哪些路径，哪些路径应该公开，以便红帽构建的 Keycloak 可以保护您的应用程序。

2.4. 限制排队请求数

生产环境应该保护自身免受过载情况，因此它会尽可能地响应尽可能多的有效请求，并在情况返回正常后继续常规操作。执行此操作的一种方法是在达到特定阈值后拒绝其他请求。

负载她应该在所有级别上实施，包括您环境中的负载均衡器。此外，红帽构建的 Keycloak 中有一个功能来限制无法立即处理的请求数量，并且需要排队。默认情况下，没有设置限制。设置选项 **http-max-queued-requests**，将排队请求数限制为与您的环境匹配的给定阈值。超过这个限制的任何请求都会返回，并显示即时 **503 Server not Available** 响应。

2.5. 生产环境等级数据库

红帽构建的 Keycloak 数据库对于红帽构建的 Keycloak 的整体性能、可用性、可靠性和完整性至关重要。有关如何配置支持的数据库的详情，请参阅配置数据库。

2.6. 在集群中运行 RED HAT BUILD OF KEYCLOAK

为确保用户在红帽构建的 Keycloak 实例停机时继续登录，典型的生产环境包含两个或更多红帽构建的 Keycloak 实例。

红帽 Keycloak 的构建在 JGroups 和 Infinispan 上运行，为集群场景提供可靠的高可用性堆栈。在默认设置中，节点之间的通信是使用 TLS 加密。

要了解更多有关使用多个节点的信息，不同的缓存以及适合您的环境的堆栈，请参阅 配置分布式缓存。

2.6.1. 配置防火墙端口

必须打开一组网络端口，以便红帽构建的 Keycloak 服务器之间有健康的网络通信。请参阅配置分布式缓存。它描述了需要打开哪些端口及其用法。

2.7. 使用 IPV4 或 IPV6 配置红帽构建的 KEYCLOAK 服务器

系统属性 **java.net.preferIPv4Stack** 和 **java.net.preferIPv6Addresses** 用于配置 JVM 以用于 IPv4 或 IPv6 地址。

默认情况下，红帽构建的 Keycloak 可通过 IPv4 和 IPv6 地址同时访问。要只使用 IPv4 地址运行，您需要指定属性 **java.net.preferIPv4Stack=true**。后者可确保任何主机名进行 IP 地址转换始终返回 IPv4 地址变体。

这些系统属性可通过 **JAVA_OPTS_APPEND** 环境变量方便地设置。例如，要将 IP 堆栈首选项更改为 IPv4，请按如下所示设置环境变量：

```
export JAVA_OPTS_APPEND="-Djava.net.preferIPv4Stack=true"
```

要仅为 IPv6 设置服务器，请按照如下所示设置环境变量来组成集群：

```
export JAVA_OPTS_APPEND="-Djava.net.preferIPv4Stack=false -
Djava.net.preferIPv6Addresses=true"
```

如需了解更多详细信息，请参阅配置分布式缓存。

第 3 章 引导和恢复管理员帐户

通过创建一个临时 admin 帐户，引导红帽构建的 Keycloak 并恢复访问。

3.1. 临时管理员帐户

使用下面描述的方法之一创建用户或服务管理员帐户 是临时的。这意味着，帐户应只在执行必要的操作所需的持续时间时才存在，以获得永久和更安全的 admin 访问权限。之后，需要手动删除帐户。各种 UI/UX 元素（如管理控制台警告横幅、标签和日志消息）将表明红帽构建的 Keycloak 管理员是临时的。

3.2. 在红帽构建的 KEYCLOAK 启动时引导临时 ADMIN 帐户

Red Hat build of Keycloak start 和 start-dev 命令支持用于引导临时管理员用户和 admin 服务帐户的选项。这些选项是标准配置选项，因此可以在任何配置 源 中指定，如环境变量或 CLI 参数。例如，以下示例演示了如何使用带有 CLI 参数的 start 和 start-dev 命令来分别引导临时 admin 用户和 admin 服务帐户：

```
bin/kc.[sh|bat] start --bootstrap-admin-username tmpadm --bootstrap-admin-password pass
```

```
bin/kc.[sh|bat] start-dev --bootstrap-admin-client-id tmpadm --bootstrap-admin-client-secret secret
```

可以省略用户名或客户端 ID 值 ; 如需更多信息，请参阅下面的 第 3.5 节 “默认值” 部分。

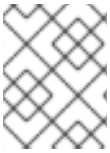
这些选项的目的仅适用于 bootstrap 临时管理帐户。只有当 master 域不存在时，才会在 Red Hat build of Keycloak 服务器的初始启动时创建这些帐户。帐户始终在 master 域中创建。要恢复丢失的 admin 访问权限，请使用以下部分中描述的专用命令。

3.3. 使用专用命令引导 ADMIN 用户或服务帐户

即使首次启动红帽构建的 Keycloak 之前，也可以执行 bootstrap-admin 命令。请记住，在使用此命令之前，需要停止所有使用红帽的 Keycloak 节点构建。其执行将触发初始 master 域的创建，因此当服务器首次启动时，将忽略 bootstrap admin 用户和服务帐户的启动选项。

另外，强烈建议使用带有红帽构建的 Keycloak 服务器相同的选项（如 db 选项）的专用命令。

如果您使用 build 命令构建红帽构建的 Keycloak 版本，如 配置红帽Keycloak 构建 中所述，请使用命令行选项 优化功能，让红帽构建 Keycloak 跳过一个更快的启动时间。执行此操作时，请从命令行删除构建时间选项，仅保留运行时选项。



注意

如果您没有使用 优化 性，则 bootstrap-admin 命令会隐式为您创建或更新优化镜像 - 如果您从与服务器实例相同的机器中运行该命令，这可能会影响服务器下次启动。

3.3.1. 创建管理员用户

要创建临时 admin 用户，请执行以下命令：

```
bin/kc.[sh|bat] bootstrap-admin user
```

如果没有指定其他参数，且/或没有设置相应的环境变量，则会提示用户输入所需的信息。可以省略用户名值以使用默认值。如需更多信息，请参阅以下 第 3.5 节 “默认值” 和 第 3.7 节 “环境变量” 部分。

或者，可以在命令中直接指定参数：

```
bin/kc.[sh|bat] bootstrap-admin user --username tmpadm --password:env PASS_VAR
```

此命令创建一个临时 admin 用户，用户名为 tmpadm，以及从环境变量检索的密码。

3.3.2. 创建服务帐户

在自动化场景中，临时 admin 服务帐户可以是更合适的临时 admin 用户替代方案。

要创建临时 admin 服务帐户，请执行以下命令：

```
bin/kc.[sh|bat] bootstrap-admin service
```

同样，如果没有设置对应的环境变量或设置其他参数，则会提示用户输入所需信息。可以省略客户端 ID 值，以使用默认值。如需更多信息，请参阅以下 第 3.5 节 “默认值” 和 第 3.7 节 “环境变量” 部分。

或者，可以在命令中直接指定参数：

```
bin/kc.[sh|bat] bootstrap-admin service --client-id tmpclient --client-secret:env=SECRET_VAR
```

此命令使用客户端 ID tmpclient 和从环境变量检索的机密创建临时 admin 服务帐户。

3.4. 通过提高安全性重新获得对域的访问权限

对于丢失了 admin 访问权限的域，可以强制使用免密码、OTP 或其他高级身份验证方法。在这种情况下，需要创建 admin 服务帐户来恢复对域的 admin 访问权限。创建服务帐户后，需要针对红帽构建的 Keycloak 实例进行身份验证来执行所有必要的操作：

```
bin/kcadm.[sh|bat] config credentials --server http://localhost:8080 --realm master --client
<service_account_client_name> --secret <service_account_secret>
```

接下来，检索 credentialId。在本例中，OTP 凭证是相关的凭证。使用以下命令获取 CredentialRepresentation 对象的数组，并查找类型为 otp 的数组：

```
bin/kcadm.[sh|bat] get users/{userId}/credentials -r {realm-name}
```

最后，检索的 ID 可用于删除高级验证方法（在我们的 case, OTP 中）：

```
bin/kcadm.[sh|bat] delete users/{userId}/credentials/{credentialId} -r {realm-name}
```

3.5. 默认值

对于启动和专用命令场景，用户名和客户端 ID 分别是可选的，默认为 temp-admin 用于 user 和 service 帐户。

3.6. 禁用参数提示

要禁用参数提示，可以使用 --no-prompt 参数。例如：

```
bin/kc.[sh|bat] bootstrap-admin user --username tmpadm --no-prompt
```

如果没有设置对应的环境变量，命令将失败，并显示出错信息，表示缺少所需的 password 参数。

如果应省略用户名或客户端 ID，--no-prompt 参数很有用。例如：

```
bin/kc.[sh|bat] bootstrap-admin user --password:env PASS_VAR --no-prompt
```

这会创建一个带有默认用户名的临时 admin 用户，而不提示确认。如需更多信息，请参阅上面的 第 3.5 节 “默认值” 部分。

3.7. 环境变量

对于 bootstrap-admin user 命令，可以选择性地将用户名和密码设置为环境变量：

```
bin/kc.[sh|bat] bootstrap-admin user --username:env <YourUsernameEnv> --password:env
<YourPassEnv>
```

对于 bootstrap-admin service 命令，客户端 ID 是可选的，默认为temp-admin，而客户端 secret 需要设置为环境变量：

```
bin/kc.[sh|bat] bootstrap-admin service --client-id:env <YourClientIdEnv> --client-secret:env
<YourSecretEnv>
```

第 4 章 目录结构

了解安装 root 下目录的用途。

4.1. 安装位置

如果您要从 zip 文件安装，默认情况下，将有一个 rhbk-26.2.10 的安装根目录，您可以在文件系统中选择的任何地方创建该目录。

/opt/keycloak 是服务器在红帽构建的 Keycloak 中显示的所有容器化用法的根安装位置。

注意

在其它文档中，可以理解相对路径相对于安装 root - 例如 conf/file.xml 表示 <install root>/conf/file.xml

4.2. 目录结构

在 Red Hat build of Keycloak install root 下有一个文件夹：

- bin/ - 包含服务器的所有 shell 脚本，包括kc.sh|bat、kcadm.sh|bat、kcreg.sh|bat
  - 客户端/ 内部使用
- 用于配置文件的 conf/ - 目录，包括 keycloak.conf - 请参阅配置红帽构建的 Keycloak。指定配置文件的许多选项预期路径相对于这个目录。
  - truststores/ - truststore-paths 选项使用的默认路径 - 请参阅配置可信证书
- 服务器用来存储运行时信息 的数据/ 目录，如事务日志
  - 文件日志记录的 logs/ - 默认目录 - 请参阅配置日志记录
- lib/ -- 内部使用
- 用户提供的依赖项的 providers/ - 目录 - 请参阅配置供应商 以扩展服务器以及为添加 JDBC 驱动程序的示例配置 数据库。
- 用于自定义管理控制台的 themes/ - 目录 - 请参阅开发主题

第 5 章 在容器中运行红帽构建的 KEYCLOAK

从容器镜像运行红帽构建的 Keycloak。

本章论述了如何优化并运行红帽构建的 Keycloak 容器镜像，以提供运行容器的最佳体验。

警告

本章只适用于构建您在 OpenShift 环境中运行的镜像。此镜像只支持 OpenShift 环境。如果您在其他 Kubernetes 发行版本中运行它，则不支持它。

5.1. 创建自定义和优化的容器镜像

默认的红帽 Keycloak 容器镜像构建已准备好配置和优化。

为了获得红帽构建的 Keycloak 容器的最佳启动，请在容器构建过程中运行 构建步骤 构建镜像。此步骤将在容器镜像的后续开始阶段节省时间。

5.1.1. 编写优化的红帽构建的 Keycloak Containerfile

以下 **Containerfile** 创建一个预先配置的红帽 Keycloak 镜像构建，它启用了健康和指标端点，启用令牌交换功能，并使用 PostgreSQL 数据库。

Containerfile ：

```
FROM registry.redhat.io/rhbk/keycloak-rhel9:26.2 AS builder

# Enable health and metrics support
ENV KC_HEALTH_ENABLED=true
ENV KC_METRICS_ENABLED=true

# Configure a database vendor
ENV KC_DB=postgres

WORKDIR /opt/keycloak
# for demonstration purposes only, please make sure to use proper certificates in production instead
RUN keytool -genkeypair -storepass password -storetype PKCS12 -keyalg RSA -keysize 2048 -dname "CN=server" -alias server -ext "SAN:c=DNS:localhost,IP:127.0.0.1" -keystore conf/server.keystore
RUN /opt/keycloak/bin/kc.sh build

FROM registry.redhat.io/rhbk/keycloak-rhel9:26.2
COPY --from=builder /opt/keycloak/ /opt/keycloak/

# change these values to point to a running postgres instance
ENV KC_DB=postgres
ENV KC_DB_URL=<DBURL>
ENV KC_DB_USERNAME=<DBUSERNAME>
```

```
ENV KC_DB_PASSWORD=<DBPASSWORD>
ENV KC_HOSTNAME=localhost
ENTRYPOINT ["/opt/keycloak/bin/kc.sh"]
```

构建过程包含多个阶段：

- 运行 build 命令，以设置服务器构建选项以创建优化的镜像。
- 由 构建阶段 生成的文件复制到新镜像中。
- 在最终镜像中，设置主机名和数据库的额外配置选项，以便在运行容器时不需要再次设置它们。
- 在入口点中，kc.sh 可让您访问所有 distribution 子命令。

要安装自定义提供程序，您只需要定义一个步骤，以将 JAR 文件包含在 /opt/keycloak/providers 目录中。此步骤必须放在 RUNs build 命令的行前，如下所示：

```
# A example build step that downloads a JAR file from a URL and adds it to the providers directory
FROM registry.redhat.io/rhbk/keycloak-rhel9:26.2 as builder

...

# Add the provider JAR file to the providers directory
ADD --chown=keycloak:keycloak --chmod=644 <MY_PROVIDER_JAR_URL> /opt/keycloak/providers/myprovider.jar

...

# Context: RUN the build command
RUN /opt/keycloak/bin/kc.sh build
```

5.1.2. 安装其他 RPM 软件包

如果您尝试在 FROM registry.redhat.io/rhbk/keycloak-rhel9 中安装新软件，您会注意到 microdnf、dnf 甚至 rpm 也不会被安装。另外，很少的软件包可用，只可用于 bash shell，并运行红帽 Keycloak 本身构建。这是因为安全强化措施减少了红帽构建的 Keycloak 容器的攻击面。

首先，请考虑您的用例是否可以以不同的方式实现，因此请避免将新 RPM 安装到最终容器中：

- Containerfile 中的 RUN curl 指令可以替换为 ADD，因为该指令原生支持远程 URL。
- 一些常见的 CLI 工具可以通过识别 Linux 文件系统来替代。例如，ip addr show tap0 变成 cat /sys/class/net/tap0/address
- 需要 RPM 的任务可以移到镜像构建的前阶段，以及复制的结果。

下面是一个示例。在以前的构建阶段运行 update-ca-trust，然后复制结果转发：

```
FROM registry.access.redhat.com/ubi9 AS ubi-micro-build
COPY mycertificate.crt /etc/pki/ca-trust/source/anchors/mycertificate.crt
RUN update-ca-trust

FROM registry.redhat.io/rhbk/keycloak-rhel9
COPY --from=ubi-micro-build /etc/pki /etc/pki
```

-

如果绝对需要，可以安装新的 RPM，遵循由 ubi-micro 建立的双阶段模式：

```
FROM registry.access.redhat.com/ubi9 AS ubi-micro-build
RUN mkdir -p /mnt/rootfs
RUN dnf install --installroot /mnt/rootfs <package names go here> --releasever 9 --setopt
install_weak_deps=false --nodocs -y && \
    dnf --installroot /mnt/rootfs clean all && \
    rpm --root /mnt/rootfs -e --nodeps setup

FROM registry.redhat.io/rhbk/keycloak-rhel9
COPY --from=ubi-micro-build /mnt/rootfs /
```

这种方法使用 chroot /mnt/rootfs，因此仅安装您指定的软件包及其依赖项，因此无需猜测即可轻松复制到第二个阶段。

警告

有些软件包具有大量依赖项。通过安装新的 RPM，您可能意外地增加容器的受攻击面。仔细检查安装的软件包列表。

5.1.3. 自定义 ENTRYPOINT shell 脚本

如果您使用自定义入口点脚本，请使用 **exec** 启动 Red Hat build of Keycloak，以便它接收对安全关闭至关重要的终止信号。

ENTRYPOINT shell 脚本的正确方法

```
#!/bin/bash

# (add your custom logic here)

# Run the 'exec' command as the last step of the script.
# As it replaces the current shell process, no additional shell commands will run after the
'exec' command.
exec /opt/keycloak/bin/kc.sh start "$@"
```

警告

如果没有 **exec**，shell 脚本会在容器中保留 PID 1，并阻止 **SIGTERM** 等信号访问红帽构建的 Keycloak。这可防止安全关闭，并可能导致缓存不一致或数据丢失。

5.1.4. 构建容器镜像

要构建实际的容器镜像，请从包含 Containerfile 的目录运行以下命令：

```
podman build . -t mykeycloak
```

注意

Podman 只能用于创建或自定义镜像。在生产环境中运行红帽构建的 Keycloak 不支持 podman。

5.1.5. 启动优化的红帽构建的 Keycloak 容器镜像

要启动镜像，请运行：

```
podman run --name mykeycloak -p 8443:8443 -p 9000:9000 \
      -e KC_BOOTSTRAP_ADMIN_USERNAME=admin -e
KC_BOOTSTRAP_ADMIN_PASSWORD=change_me \
      mykeycloak \
      start --optimized --hostname=localhost
```

红帽构建的 Keycloak 以生产模式启动，仅使用安全 HTTPS 通信，并可在 https://localhost:8443 中提供。

健康检查端点位于 https://localhost:9000/health、https://localhost:9000/health/ready 和 https://localhost:9000/health/live。

打开 https://localhost:9000/metrics 会导致一个页面，其中包含您的监控解决方案可以使用的操作指标。

5.1.6. Docker 的已知问题

- 如果 RUN dnf install 命令似乎需要过多的时间，则您的 Docker systemd 服务可能会错误地配置了文件限制设置 LimitNOFILE。更新服务配置以使用更好的值，如 1024000，或者在 RUN 命令中使用 ulimit ：

```
...
RUN ulimit -n 1024000 && dnf install --installroot ...
...
```

- 如果您包含提供商 JAR，并且容器无法使用供应商 JAR 已更改的通知 进行启动 --optimized，这是由于 Docker 截断，或者修改 以构建 命令记录到运行时所看到的内容的文件修改时间戳。在这种情况下，您需要在运行构建前强制镜像使用带有 touch 命令选择的已知时间戳：

```
...
# ADD or copy one or more provider jars
ADD --chown=keycloak:keycloak --chmod=644 some-jar.jar /opt/keycloak/providers/
...
RUN touch -m --date=@1743465600 /opt/keycloak/providers/*
RUN /opt/keycloak/bin/kc.sh build
...
```

5.2. 将容器公开给不同的端口

默认情况下，服务器分别使用端口 8080 和 8443 侦听 http 和 https 请求。

如果要使用其他端口公开容器，则需要相应地 设置主机名 ：

1. 使用默认端口以外的端口公开容器

```
podman run --name mykeycloak -p 3000:8443 \
        -e KC_BOOTSTRAP_ADMIN_USERNAME=admin -e
KC_BOOTSTRAP_ADMIN_PASSWORD=change_me \
        mykeycloak \
        start --optimized --hostname=https://localhost:3000
```

通过将 hostname 选项设置为完整的 url，您现在可以通过 https://localhost:3000 访问服务器。

5.3. 在开发模式中尝试红帽构建的 KEYCLOAK

从容器中尝试红帽构建的 Keycloak 最简单的方法是使用开发或测试目的。您可以使用 start-dev 命令：

```
podman run --name mykeycloak -p 8080:8080 \
        -e KC_BOOTSTRAP_ADMIN_USERNAME=admin -e
KC_BOOTSTRAP_ADMIN_PASSWORD=change_me \
        registry.redhat.io/rhbk/keycloak-rhel9:26.2 \
        start-dev
```

调用此命令将在开发模式下启动红帽构建的 Keycloak 服务器。

在生产环境中应该严格避免此模式，因为它具有不安全的默认值。有关在生产环境中运行红帽构建的 Keycloak 的更多信息，请参阅 为生产环境配置红帽构建的 Keycloak。

5.4. 运行 KEYCLOAK 容器的标准红帽构建

为了保持不可变基础架构等概念，容器需要定期重新置备。在这些环境中，您需要启动快速的容器，因此您需要创建一个优化镜像，如上一节中所述。但是，如果您的环境有不同的要求，可以通过运行 start

命令运行 Keycloak 镜像的标准红帽构建。例如：

```
podman run --name mykeycloak -p 8080:8080 \
      -e KC_BOOTSTRAP_ADMIN_USERNAME=admin -e
KC_BOOTSTRAP_ADMIN_PASSWORD=change_me \
      registry.redhat.io/rhbk/keycloak-rhel9:26.2 \
      start \
      --db=postgres --features=token-exchange \
      --db-url=<JDBC-URL> --db-username=<DB-USER> --db-password=<DB-PASSWORD> \
      --https-key-store-file=<file> --https-key-store-password=<password>
```

运行这个命令会启动红帽构建的 Keycloak 服务器，该服务器会首先检测并应用构建选项。在示例中，line-- db=postgres --features=token-exchange 将数据库供应商设置为 PostgreSQL，并启用令牌交换功能。

然后，Red Hat build of Keycloak 启动并应用特定环境的配置。这种方法显著增加启动时间，并创建可可变的镜像，这不是最佳实践。

5.5. 在容器中运行时提供初始 ADMIN 凭据

Red Hat build of Keycloak 只允许从本地网络连接创建初始 admin 用户。在容器中运行时并非如此，因此您必须在运行镜像时提供以下环境变量：

```
# setting the admin username
-e KC_BOOTSTRAP_ADMIN_USERNAME=<admin-user-name>

# setting the initial password
-e KC_BOOTSTRAP_ADMIN_PASSWORD=change_me
```

5.6. 导入域启动

红帽构建的 Keycloak 容器有一个目录 /opt/keycloak/data/import。如果您通过卷挂载或其他方法将一个或多个导入文件放到该目录中，并添加启动参数 --import-realm，则红帽构建的 Keycloak 容器将在启动时导入这些数据！这只适用于在 Dev 模式中。

```
podman run --name keycloak_unoptimized -p 8080:8080 \
      -e KC_BOOTSTRAP_ADMIN_USERNAME=admin -e
KC_BOOTSTRAP_ADMIN_PASSWORD=change_me \
      -v /path/to/realm/data:/opt/keycloak/data/import \
      registry.redhat.io/rhbk/keycloak-rhel9:26.2 \
      start-dev --import-realm
```

随意加入开放 GitHub 讨论，讨论管理 bootstrap 过程的改进。

5.7. 指定不同的内存设置

红帽构建的 Keycloak 容器，而不是为初始和最大堆大小指定硬编码值，而是对容器的总内存使用相对值。这个行为可通过 JVM options -XX:MaxRAMPercentage=70、和 -XX:InitialRAMPercentage=50 来实现。

-XX:MaxRAMPercentage 选项表示容器内存总量的最大堆大小为 70%。-XX:InitialRAMPercentage 选项表示容器内存总量的初始堆大小为 50%。根据红帽构建的 Keycloak 内存管理深度分析来选择这些值。

由于堆大小是根据总容器内存动态计算的，您应该始终为容器设置内存限制。在以前的版本中，最大堆大小被设置为 512 MB，为了采用类似值，您应该将内存限制设置为至少 750 MB。对于较小的生产环境就绪部署，推荐的内存限值为 2 GB。

通过设置环境变量 JAVA_OPTS_KC_HEAP，可以覆盖与堆相关的 JVM 选项。您可以在 kc.sh、或 kc.bat 脚本的源代码中找到 JAVA_OPTS_KC_HEAP 的默认值。

例如，您可以指定环境变量和内存限值，如下所示：

```
podman run --name mykeycloak -p 8080:8080 -m 1g \
      -e KC_BOOTSTRAP_ADMIN_USERNAME=admin -e
KC_BOOTSTRAP_ADMIN_PASSWORD=change_me \
      -e JAVA_OPTS_KC_HEAP="-XX:MaxHeapFreeRatio=30 -XX:MaxRAMPercentage=65" \
      registry.redhat.io/rhbk/keycloak-rhel9:26.2 \
      start-dev
```

警告

如果没有设置内存限制，则内存消耗会快速增加，因为堆大小可增长到总容器内存的 70%。JVM 分配内存后，会使用当前的 Keycloak GC 设置构建将其返回到操作系统。

5.8. 相关选项

	value
db 数据库供应商。 在 production 模式中，**dev-file** 的默认值已弃用，您应该明确指定 db。 CLI: -- **db** Env: **KC_DB**	**dev-file** (default), **dev-mem**,**mariadb**,**mssql**, **mysql**,**oracle**,**postgres**
db-password 数据库用户的密码。 CLI: **--db-password** Env: **KC_DB_PASSWORD**	
db-url 完整的数据库 JDBC URL。 如果没有提供，则会根据所选数据库供应商设置默认 URL。例如，如果使用 **postgres**，默认的 JDBC URL 为 **jdbc:postgresql://localhost/keycloak**。 CLI: **--db-url** Env: **KC_DB_URL**	
db-username 数据库用户的用户名。 CLI: **--db-username** Env: **KC_DB_USERNAME**	

	value
功能 启用一个或多个功能的集合。 CLI: -- **features** Env: **KC_FEATURES**	**account-api[:v1]**, **account[:v3]**, **admin-api[:v1]**, **admin-fine-grained-authz[:v1,v2]**, **admin[:v2]**, **authorization[:v1]**, **cache-embedded-remote-store[:v1]**, **ciba[:v1]**, **client-policies[:v** 1] **, client-secret-rotation[:v** 1] **, client-types[:v** 1] **, clusterless[:v** 1] **, declarative-ui[:v** 1] **, device-flow[:v** 1] **, docker[:v** 1] **, dpop[:v** 1] **, dynamic-scopes[:v** 1] **, fips[:** v1] **, hostname[:** v2] **, impersonation[:** v1] **, ipa-tuura-federation[:** v1] **, kerberos[:** v1] **, login[:v2,** v1] **, multi-site[:** v1] **, oid4vc-vci[:** v1] , **Opentelemetry[:** **v1], organization[:v 1] , par[:v 1] , passkeys[:v 1] , persistent-user-sessions[:v 1],preview , quick-theme[:v 1] , recovery-codes[:v 1] , rolling-updates[:v 1] , scripts[:v 1] , step-up-authentication** [: **v1] , token-exchange-standard** [: **v2] , token-exchange** [:v **1] , ephemeral-users** [: **v1] , update-email** [: **v1] , user-event-metrics** [: **v1] , web-authn**[:v1]

	value
hostname 服务器所公开的地址。 可以是完整的 URL，也可以是主机名。当只提供主机名时，会从请求解析端口和上下文路径。 CLI: -- **hostname** Env: **KC_HOSTNAME** 仅在启用 hostname:v2 功能时才可用	
https-key-store-file 保存证书信息的密钥存储，而不是指定单独的文件。 CLI: **--https-key-store-file** Env: **KC_HTTPS_KEY_STORE_FILE**	
https-key-store-password 密钥存储文件的密码。 CLI: **--https-key-store-password** Env: **KC_HTTPS_KEY_STORE_PASSWORD**	**密码** （默认）
health-enabled ▮ 如果服务器应该公开健康检查端点。 如果启用，健康检查位于 /health、 **/health /ready** 和 **/health/live** 端点上。 CLI: **--health-enabled** Env: **KC_HEALTH_ENABLED**	**true**,**false** （默认）
metrics-enabled ▮ 如果服务器应该公开指标。 如果启用，则指标位于 **/metrics** 端点。 CLI: **--metrics-enabled** Env: **KC_METRICS_ENABLED**	**true**,**false** （默认）

第 6 章 配置 TLS

为正在进行和传出请求配置 Keycloak 的 https 证书的红帽构建。

传输层安全性（短：TLS）对于通过安全通道交换数据至关重要。对于生产环境，您不应该通过 HTTP 公开红帽构建的 Keycloak 端点，因为敏感数据是红帽与其他应用程序构建 Keycloak 交换的核心。在本章中，您将了解如何将红帽构建的 Keycloak 配置为使用 HTTPS/TLS。

红帽构建的 Keycloak 可以配置为使用 PEM 格式或 Java 密钥存储中的文件来加载所需的证书基础架构。配置两个替代方法时，PEM 文件优先于 Java 密钥存储。

6.1. 以 PEM 格式提供证书

当您使用 PEM 格式的匹配证书和私钥文件时，您可以通过运行以下命令来将红帽构建的 Keycloak 配置为使用它们：

```
bin/kc.[sh|bat] start --https-certificate-file=/path/to/certfile.pem --https-certificate-key-file=/path/to/keyfile.pem
```

红帽构建的 Keycloak 在内存中创建了这些文件的密钥存储，并在之后使用此密钥存储。

6.2. 提供密钥存储

如果没有显式配置密钥存储文件，但 http-enabled 被设置为 false，红帽构建的 Keycloak 会查找 conf/server.keystore 文件。

另外，您可以通过运行以下命令来使用现有的密钥存储：

```
bin/kc.[sh|bat] start --https-key-store-file=/path/to/existing-keystore-file
```

可识别密钥存储的文件扩展：

- .p12、.pkcs12 和 .pfx 用于 pkcs12 文件

- .jks，以及 jks 文件的 .keystore
- .key、.crt、和 .pem 用于 pem 文件

如果您的密钥存储没有与其文件类型匹配的扩展名，您还需要设置 https-key-store-type 选项。

6.2.1. 设置密钥存储密码

您可以使用 https-key-store-password 选项为您的密钥存储设置安全密码：

```
bin/kc.[sh|bat] start --https-key-store-password=<value>
```

如果没有设置密码，则使用默认 密码。

6.2.1.1. 保护凭证

使用 CLI 避免在纯文本中设置密码，或将其添加到 conf/keycloak.conf 文件中。取而代之，如使用 vault / mounted secret。如需了解更多详细信息，请参阅 为生产环境使用 vault 和 配置红帽构建的 Keycloak。

6.3. 配置 TLS 协议

默认情况下，Red Hat build of Keycloak 不会启用弃用的 TLS 协议。如果您的客户端只支持已弃用的协议，请考虑升级客户端。但是，作为临时工作方法，您可以通过运行以下命令来启用已弃用的协议：

```
bin/kc.[sh|bat] start --https-protocols=<protocol>[,<protocol>]
```

例如，只启用 TLSv1.3，请使用如下命令：kc .sh start --https-protocols=TLSv1.3。

6.4. 切换 HTTPS 端口

红帽构建的 Keycloak 侦听端口 8443 上的 HTTPS 流量。要更改这个端口，请使用以下命令：

```
bin/kc.[sh|bat] start --https-port=<port>
```

6.5. 证书和密钥重新加载

默认情况下，Red Hat build of Keycloak 将每小时重新载入在 https fluentd 选项中指定的证书、密钥和密钥存储。对于您的服务器密钥可能需要频繁轮转的环境，这允许在不重启服务器的情况下发生。您可以通过 https-certificates-reload-period 选项覆盖默认设置。重新载入密钥存储、信任存储和证书文件的间隔，由 https channel 选项引用。该值可以是 java.time.Duration 值、整数数或整数，后跟一个时间单位之一 [ms,h,m,s s ,d]。必须大于 30 秒。使用 -1 禁用。

6.6. 相关选项

	value
http-enabled 启用 HTTP 侦听器。 在开发模式中默认启用。除非服务器前面是 TLS 终止代理，否则通常不会在生产环境中启用。 CLI: **--http-enabled** Env: **KC_HTTP_ENABLED**	**true**,**false** （默认）
https-certificate-file PEM 格式的服务器证书或证书链的文件路径。 CLI: **--https-certificate-file** Env: **KC_HTTPS_CERTIFICATE_FILE**	
https-certificate-key-file PEM 格式的私钥的文件路径。 CLI: **--https-certificate-key-file** Env: **KC_HTTPS_CERTIFICATE_KEY_FILE**	
https-certificates-reload-period 重新载入密钥存储、信任存储和证书文件的间隔，由 https channel 选项引用。 可以是 java.time.Duration 值、整数数或整数，后跟 [ms, h, m, s, d] 之一。必须大于 30 秒。使用 -1 禁用。 CLI: **--https-certificates-reload-period** Env: **KC_HTTPS_CERTIFICATES_RELOAD_PERIOD**	**1h** （默认）

	value
https-cipher-suites 要使用的密码套件。 如果未指定，则会选择合理的默认值。 CLI: **--https-cipher-suites** Env: **KC_HTTPS_CIPHER_SUITES**	
https-key-store-file 保存证书信息的密钥存储，而不是指定单独的文件。 CLI: **--https-key-store-file** Env: **KC_HTTPS_KEY_STORE_FILE**	
https-key-store-password 密钥存储文件的密码。 CLI: **--https-key-store-password** Env: **KC_HTTPS_KEY_STORE_PASSWORD**	**密码**（默认）
https-key-store-type 密钥存储文件的类型。 如果未指定，则根据文件扩展名自动检测到类型。如果将 **fips-mode** 设为 **strict**，且没有设置值，则默认为 **BCFKS**。 CLI: **--https-key-store-type** Env: **KC_HTTPS_KEY_STORE_TYPE**	
https-port 使用的 HTTPS 端口。 CLI: **--https-port** Env: **KC_HTTPS_PORT**	**8443**（默认）
https-protocols 要显式启用的协议列表。 如果 JRE / 安全配置不支持值，它将被静默忽略。 CLI: **--https-protocols** Env: **KC_HTTPS_PROTOCOLS**	**TLSv1.3**、**TLSv1.2** 或任何

6.6.1. 管理服务器

	value
https-management-certificate-file 管理服务器的 PEM 格式的服务器证书或证书链的文件路径。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-certificate-file** Env: **KC_HTTPS_MANAGEMENT_CERTIFICATE_FILE**	
https-management-certificate-key-file 管理服务器的 PEM 格式的私钥的文件路径。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-certificate-key-file** Env: **KC_HTTPS_MANAGEMENT_CERTIFICATE_KEY_FILE**	
https-management-certificates-reload-period 重新载入管理服务器的 https-management ldapmodify 选项引用的密钥存储、信任存储和证书文件的时间间隔。 可以是 java.time.Duration 值、整数数或整数，后跟 [ms, h, m, s, d] 之一。必须大于 30 秒。使用 -1 禁用。如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-certificates-reload-period** Env: **KC_HTTPS_MANAGEMENT_CERTIFICATES_RELOAD_PERIOD**	**1h** （默认）
https-management-key-store-file 保存证书信息的密钥存储，而不是为管理服务器指定单独的文件。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-key-store-file** Env: **KC_HTTPS_MANAGEMENT_KEY_STORE_FILE**	
https-management-key-store-password 管理服务器的密钥存储文件的密码。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-key-store-password** Env: **KC_HTTPS_MANAGEMENT_KEY_STORE_PASSWORD**	**密码** （默认）

第 7 章 配置主机名(V2)

配置由红帽构建的 Keycloak 公开的 frontend 和 backchannel 端点。

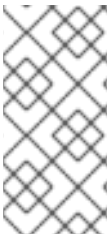
7.1. 设置 HOSTNAME 选项的重要性

默认情况下，Red Hat build of Keycloak 会强制配置 hostname 选项，且不会动态解析 URL。这是一种安全措施。

红帽构建的 Keycloak 可自由披露其自身 URL，例如通过 OIDC Discovery 端点，或作为电子邮件中的密码重置链接的一部分。如果主机名从主机名头中动态解释，则可能会提供一个潜在的攻击者来回操作电子邮件中的 URL，将用户重定向到攻击者的假期域，以及窃取敏感数据，如操作令牌、密码等。

通过显式设置 hostname 选项，我们避免了由欺诈签发者签发令牌的情况。使用以下命令可使用显式主机名启动服务器：

```
bin/kc.[sh|bat] start --hostname my.keycloak.org
```



注意

示例在生产环境模式下启动红帽构建的 Keycloak 实例，这需要一个公钥和私钥才能保护通信。如需更多信息，请参阅 为生产环境配置红帽构建的 Keycloak。

7.2. 定义 HOSTNAME 选项的特定部分

如上例中所示，不明确要求方案和端口。在这种情况下，红帽构建的 Keycloak 会自动处理这些方面。例如，该服务器可以通过给定示例中的 https://my.keycloak.org:8443 访问。但是，反向代理通常会在默认端口（如 443 ）上公开红帽构建的 Keycloak。在这种情况下，需要在 hostname 选项中指定完整的 URL，而不是保持 URL 动态部分。然后，可以使用以下内容启动服务器：

```
bin/kc.[sh|bat] start --hostname https://my.keycloak.org
```

同样，您的反向代理可能会以不同的上下文路径公开红帽构建的 Keycloak。可以配置红帽构建的 Keycloak，以反映通过主机名和 hostname -admin 选项。请参见以下示例：

```
bin/kc.[sh|bat] start --hostname https://my.keycloak.org:123/auth
```

7.3. 使用内部 URL 进行客户端之间的通信

Red Hat build of Keycloak 能够为后端频道请求提供单独的 URL，启用内部通信，同时维护对前端请求的使用公共 URL。此外，根据传入的标头动态解析回溯通道。考虑以下示例：

```
bin/kc.[sh|bat] start --hostname https://my.keycloak.org --hostname-backchannel-dynamic true
```

这样，您的应用程序称为客户端，可以通过本地网络与红帽构建的 Keycloak 连接，同时服务器仍然可在 https://my.keycloak.org 中公开访问。

7.4. 使用边缘 TLS 终止

在观察到时，HTTPS 协议是默认的选择，遵循红帽构建的 Keycloak 对安全最佳实践的承诺。但是，红帽构建的 Keycloak 也为用户提供了按需选择 HTTP 的灵活性。这可以通过指定 HTTP 侦听器来实现，详情请参阅 配置 TLS。使用边缘 TLS-termination 代理，您可以按如下方式启动服务器：

```
bin/kc.[sh|bat] start --hostname https://my.keycloak.org --http-enabled true
```

此配置的结果是，您可以继续通过 HTTPS 访问 https://my.keycloak.org 的红帽构建 Keycloak，而代理则使用 HTTP 和端口 8080 与实例交互。

7.5. 使用反向代理

当代理转发 http 或重新加密 TLS 请求时，应设置 proxy-headers 选项。根据主机名设置（部分或全部 URL）可能会动态确定。

警告

如果选择了 转发 或 xforwarded，请确保您的反向代理正确设置并分别覆盖 Forwarded 或 X-Forwarded EgressIP 标头。要设置这些标头，请查阅您的反向代理文档。错误配置会使红帽构建 Keycloak 暴露于安全漏洞。

7.5.1. 完全动态 URL。

例如，如果您的反向代理正确设置了 Forwarded 标头，而您不想硬编码主机名，则红帽构建的 Keycloak 可以容纳这一点。您只需要启动服务器，如下所示：

```
bin/kc.[sh|bat] start --hostname-strict false --proxy-headers forwarded
```

使用这个配置，服务器会尊重 Forwarded 标头设置的值。这也意味着所有端点都是动态解析的。

7.5.2. 部分动态 URL

当 hostname 选项没有指定为完整 URL 时，也可以使用 proxy-headers 选项来动态解析 URL。例如：

```
bin/kc.[sh|bat] start --hostname my.keycloak.org --proxy-headers xforwarded
```

在这种情况下，方案和端口从 X-ForwardedEUS 标头动态解析，而 hostname 则静态定义为 my.keycloak.org。

7.5.3. 修复了 URL

即使 主机名设置为 完整的 URL， proxy-headers 仍然相关，因为标头用于决定请求的来源。例如：

```
bin/kc.[sh|bat] start --hostname https://my.keycloak.org --proxy-headers xforwarded
```

在这种情况下，虽然没有从 X-Forwarded github 标头动态解析，则使用 X-Forwarded fluentd 标头来确定请求的正确来源。

7.6. 在单独的主机名上公开管理控制台

如果要在其他主机上公开管理控制台，您可以使用以下命令完成此操作：

```
bin/kc.[sh|bat] start --hostname https://my.keycloak.org --hostname-admin
https://admin.my.keycloak.org:8443
```

这可让您在 https://my.keycloak.org 和位于 https://admin.my.keycloak.org:8443 的管理控制台访问红帽构建的 Keycloak，而后端继续使用 https://my.keycloak.org。

注意

请记住，主机名和代理选项不会更改服务器侦听的端口。相反，它只更改静态资源的端口，如 JavaScript 和 CSS 链接、OIDC 已知的端点、重定向 URI 等。这将在代理前面使用。您需要使用 HTTP 配置选项来更改服务器正在侦听的实际端口。详情请参考 All configuration。

警告

使用 hostname-admin 选项不会阻止通过 hostname 选项指定的前端 URL 访问管理 REST API 端点。如果要限制对管理 REST API 的访问，您需要在反向代理级别上进行。管理控制台使用 hostname-admin 选项指定的 URL 隐式访问 API。

7.7. 背景 - 服务器端点

红帽构建的 Keycloak 会公开多个端点，每个端点都有不同的目的。它们通常用于应用之间的通信或管理服务器。我们识别 3 个主要端点组：

- frontend
- 后端
- 管理

如果要使用这些端点，则需要设置基本 URL。基本 URL 由几个部分组成：

- 一个方案（如 https 协议）
- 主机名（如 example.keycloak.org）

- 端口（如 8443）
- 一个路径（如 /auth）

每个组的基本 URL 对签发和验证令牌有重要影响，关于如何为需要用户重定向到红帽构建的 Keycloak （例如，通过电子邮件链接重置密码时，在获取 OpenID Connect Discovery Document .well -name}/.well-known/openid-configuration 时，如何发现这些端点）。

7.7.1. frontend

用户和应用程序使用前端 URL 通过前端频道访问 Keycloak 的红帽构建。前端频道是一个公开访问的通信频道。例如，基于浏览器的流（访问登录页面，单击链接以重置密码或绑定令牌）可被视为前端请求。

为了让红帽构建 Keycloak 可通过前端 URL 访问，您需要设置 hostname 选项：

```
bin/kc.[sh|bat] start --hostname my.keycloak.org
```

7.7.2. 后端

后端端点可以通过公共域或通过专用网络访问。它们与红帽构建的 Keycloak 和客户端之间的后端通信直接通信（由红帽构建的 Keycloak 保护的应用程序）。此类通信可能在本地网络中，避免反向代理。属于此组的端点示例包括授权端点、令牌和令牌内省端点、userinfo 端点、JWKS URI 端点等。

hostname-backchannel-dynamic 选项的默认值为 false，这意味着回溯通道 URL 与 frontchannel URL 相同。可以通过设置以下选项来启用来自传入请求标头的后端通道 URL 的动态解析：

```
bin/kc.[sh|bat] start --hostname https://my.keycloak.org --hostname-backchannel-dynamic
true
```

请注意，hostname 选项必须设置为 URL。如需更多信息，请参阅下面的 第 7.9 节 “验证” 部分。

7.7.3. 管理

与基本前端 URL 类似，您还可以为管理控制台的资源和端点设置基本 URL。服务器使用特定 URL 公

开管理控制台和静态资源。此 URL 用于重定向 URL、加载资源(CSS、JS)、管理 REST API 等。它可以通过设置 hostname-admin 选项来完成：

```
bin/kc.[sh|bat] start --hostname https://my.keycloak.org --hostname-admin
https://admin.my.keycloak.org:8443
```

同样，hostname 选项必须设置为 URL。如需更多信息，请参阅下面的 第 7.9 节 “验证” 部分。

7.8. 解析 URL 的源

如上一节中所示，可以使用以下几种方法解析 URL：可以动态生成、硬编码或两者的组合：

- 从传入请求动态：
  - 主机标头、方案、服务器端口、上下文路径
  - proxy-set 标头： Forwarded 和 X-Forwarded fluentd
- 硬编码：
  - 服务器范围内的配置（如主机名、 hostname -admin 等等）
  - 前端 URL 的域配置

7.9. 验证

- 验证 主机名 URL 和 hostname-admin URL 被验证是否使用完整 URL、incl. scheme 和 hostname。只有存在时端口才会验证端口，否则会假定给定协议的默认端口(80 或 443)。
- 在生产配置集(kc.sh|bat start)中，必须明确配置 either --hostname or--hostname-strict false。

- 这不适用于 dev 配置集(kc.sh|bat start-dev)，其中 --hostname-strict false 是默认值。

- if --hostname 没有配置：
  - hostname-backchannel-dynamic 必须设置为 false。
  - hostname-strict 必须设置为 false。
- 如果配置了 hostname-admin，则必须将 hostname 设置为 URL （不仅仅是 hostname）。否则，Red Hat build of Keycloak 不知道在访问管理控制台时正确的前端 URL (incl.port 等)。
- 如果 hostname-backchannel-dynamic 设为 true，则 主机名 必须设置为 URL （不仅仅是 hostname）。否则，Red Hat build of Keycloak 不知道当通过动态解析的 bachchannel 访问时，正确的前端 URL (incl.port 等)。

另外，如果配置了主机名，则忽略 hostname-strict。

7.10. 故障排除

要排除主机名配置的问题，您可以使用专用的 debug 工具，该工具可启用：

红帽构建的 Keycloak 配置：

```
bin/kc.[sh|bat] start --hostname=mykeycloak --hostname-debug=true
```

在 Red Hat build of Keycloak 正确启动后，打开浏览器并访问：
http://mykeycloak:8080/realms/<your-realm>/hostname-debug

7.11. 相关选项

表 7.1. 默认情况下，此端点被禁用(--hostname-debug=false)

	value
hostname 服务器所公开的地址。 可以是完整的 URL，也可以是主机名。当只提供主机名时，会从请求解析端口和上下文路径。 CLI: -- **hostname** Env: **KC_HOSTNAME** 仅在启用 hostname:v2 功能时才可用	
hostname-admin 用于访问管理控制台的地址。 如果您使用反向代理在与 **hostname** 选项中指定的不同地址上公开管理控制台，则使用这个选项。 CLI: **--hostname-admin** Env: **KC_HOSTNAME_ADMIN** 仅在启用 hostname:v2 功能时才可用	
hostname-backchannel-dynamic 启用动态解析回溯通道 URL，包括主机名、方案、端口和上下文路径。 如果您的应用程序通过私有网络访问 Keycloak，则设置为 true。如果设置为 true，则需要将 **hostname** 选项指定为完整的 URL。 CLI: **--hostname-backchannel-dynamic** Env: **KC_HOSTNAME_BACKCHANNEL_DYNAMIC** 仅在启用 hostname:v2 功能时才可用	**true**,**false** （默认）
hostname-debug 切换可在 /realms/master/hostname-debug 访问的主机名调试页面。 CLI: **--hostname-debug** Env: **KC_HOSTNAME_DEBUG** 仅在启用 hostname:v2 功能时才可用	**true**,**false** （默认）

	value
hostname-strict 禁用从请求标头动态解析主机名。 在生产环境中，应始终设为 true，除非反向代理会覆盖 Host 标头。如果启用，则需要指定 **hostname** 选项。 CLI: **--hostname-strict** Env: **KC_HOSTNAME_STRICT** 仅在启用 hostname:v2 功能时才可用	**true**（默认），**false**

第 8 章 配置一个反向代理

使用反向代理、API 网关或负载均衡器配置红帽构建的 Keycloak。

分布式环境通常需要使用反向代理。红帽构建的 Keycloak 提供多个选项来安全地与此类环境集成。

8.1. 要代理的端口

默认情况下，Red Hat build of Keycloak 在以下端口上运行：

- 8443 (当您显式启用 HTTP 时，8443)
- 9000

端口 8443 （或启用了 HTTP）用于 Admin UI、Account Console、SAML 和 OIDC 端点以及 Admin REST API，如 配置主机名(v2) 章节中所述。

端口 9000 用于管理，其中包括健康检查和指标的端点，如 配置 管理界面一章中所述。

您只需要代理端口 8443 （或 8080），即使您为 frontend/backend 和管理使用不同的主机名，如 为生产环境配置红帽构建的 Keycloak 所述。您不应该代理端口 9000，因为健康检查和指标直接使用这些端口，而您不想将此信息公开给外部调用者。

8.2. 配置反向代理标头

红帽构建的 Keycloak 将根据 proxy-headers 选项解析反向代理标头，该选项接受几个值：

- 默认情况下，如果没有指定 选项，则不会解析反向代理标头。当没有代理正在使用或带有 https 透传时，这应该使用它。

- 转发 启用根据 RFC7239 解析 Forwarded 标头。

- X forwarded 启用解析非标准 X-Forwarded输入 标头，如 X-Forwarded-For、X-Forwarded-Proto、X-Forwarded-Host 和 X-Forwarded-Port。

注意

如果您在 https passthrough 之外使用反向代理且没有设置 proxy-headers 选项，则默认情况下，您将看到通过执行原始检查的代理对请求的 403 Forbidden 响应。

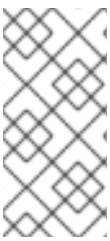
例如：

```
bin/kc.[sh|bat] start --proxy-headers forwarded
```

警告

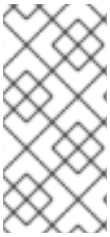
如果选择了 转发 或 xforwarded，请确保您的反向代理正确设置并分别覆盖 Forwarded 或 X-Forwarded EgressIP 标头。要设置这些标头，请查阅您的反向代理文档。不要使用 forward 或 x forwarded with https passthrough。错误配置会使红帽构建 Keycloak 暴露于安全漏洞。

采取额外的预防措施，确保客户端地址由您的反向代理通过 Forwarded 或 X-Forwarded-For 标头正确设置。如果正确配置了此标头，则恶意客户端可以设置此标头并欺骗红帽构建的 Keycloak，以认为客户端从与实际地址不同的 IP 地址连接。如果您执行任何拒绝或允许 IP 地址列表，则此预防措施可能更为重要。



注意

使用 xforwarded 设置时，X-Forwarded-Port 优先于 X-Forwarded-Host 中包含的任何端口。



注意

如果在反向代理（边缘终止）终止 TLS 连接，则需要通过 http-enabled 设置启用 HTTP。

8.3. 反向代理上的上下文路径

Red Hat build of Keycloak 假设它通过与为红帽构建的 Keycloak 配置相同的上下文路径下的反向代理公开。默认情况下，Red Hat build of Keycloak 通过 root (/)公开，这意味着它还需要通过 / 上的反向代理公开。在这些情况下，您可以使用 hostname 选项的完整 URL，例如 using--hostname=https://my.keycloak.org/auth 如果红帽构建的 Keycloak 通过 /auth 上的反向代理公开。

有关在不同的主机名或 context-path incl. 管理 REST API 和控制台上公开红帽构建的 Keycloak 的详情，请参阅 配置主机名(v2)。

另外，您还可以更改红帽构建的 Keycloak 本身的上下文路径，以使用 http-relative-path 选项与反向代理的上下文路径匹配，该选项将使用 http-relative-path 选项更改红帽构建的 Keycloak 本身的上下文路径，以匹配反向代理使用的上下文路径。

8.4. 启用粘性会话

典型的集群部署由负载均衡器（反向代理）和 2 个或更多红帽在专用网络上构建 Keycloak 服务器组成。出于性能的目的，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红帽构建的 Keycloak 后端节点，这可能很有用。

原因在于，红帽构建的 Keycloak 在 covers 下使用 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 分布式缓存配置有有限所有者数。这意味着，与会话相关的数据存储在一些集群节点中，其他节点需要远程查找数据（如果它们想要访问数据）。

例如，如果带有 ID 123 的身份验证会话保存在 node1 上的 Infinispan 缓存中，然后 node2 需要通过网络向 node1 发送请求，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，这非常有用，这可通过粘性会话的帮助来完成。带有公共前端负载均衡器和 Keycloak 节点的两个后端红帽构建的工作流可能类似如下：

- 用户发送初始请求以查看红帽构建的 Keycloak 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，节点不需要随机设置，但可以根据其他标准（客户端 IP 地址等）进行选择。它都取决于底层负载均衡器（反向代理）的实施和配置。

- 红帽构建的 Keycloak 创建具有随机 ID （如 123）的身份验证会话，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。有关此问题的更多详细信息，请参阅 Infinispan 文档。假设 Infinispan 分配的 node2 是此会话的所有者。

- Red Hat build of Keycloak 创建 Cookie AUTH_SESSION_ID，其格式为 <session-id>.<owner-node-id>。在我们的示例中，它将是 123.node2。

- 使用红帽构建的 Keycloak 登录屏幕和浏览器中的 AUTH_SESSION_ID cookie 将响应返回给用户

从这一刻，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 的身份验证会话的所有者，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 上，因为它具有相同的 ID 123。

集群设置中不强制使用粘性会话，但出于上述原因，这对性能很好。您需要配置 loadbalancer 以保留 AUTH_SESSION_ID cookie。进行此更改的适当流程取决于您的负载均衡器。

如果您的代理支持在没有处理来自后端节点的 Cookie 的情况下处理会话关联性，您应该将 spi-sticky-session-encoder-infinispan-should-attach-route 选项设置为 false，以避免将节点附加到 Cookie，并只依赖反向代理功能。

```
bin/kc.[sh|bat] start --spi-sticky-session-encoder-infinispan-should-attach-route=false
```

默认情况下，spi-sticky-session-encoder-infinispan-should-attach-route 选项值为 true，以便节点名称附加到 Cookie 以指明后续请求应发送到的节点。

8.5. 公开路径建议

在使用反向代理时，红帽构建的 Keycloak 只需要公开某些路径。下表显示了要公开的推荐路径。

红帽构建的 Keycloak 路径	反向代理路径	公开	原因
/	-	否	在公开所有路径时，管理员路径会不必要地公开。
/admin/	-	否	公开的管理路径会导致不必要的攻击向量。
/realms/	/realms/	是	这个路径需要正常工作，例如 OIDC 端点。
/resources/	/resources/	是	需要此路径才能正确服务资产。它可以从 CDN 提供，而不是红帽构建的 Keycloak 路径。
/metrics	-	否	公开的指标会导致不必要的攻击向量。
/health	-	否	公开的健康检查会导致不必要的攻击向量。

我们假设您在反向代理 / gateway 公共 API 上的根路径 / 上运行红帽构建的 Keycloak。如果没有，请为所需路径加上前缀。

8.6. 可信代理

为确保仅从您信任的代理中使用代理标头，请将 proxy-trusted-addresses 选项设置为以逗号分隔的 IP 地址列表(IPv4 或 IPv6)或无类别域间路由(CIDR)表示法。

例如：

```
bin/kc.[sh|bat] start --proxy-headers forwarded --proxy-trusted-addresses=192.168.0.32,127.0.0.0/8
```

8.7. PROXY 协议

proxy-protocol-enabled 选项控制服务器在提供代理后面的请求时是否应该使用 HA PROXY 协议。当设置为 true 时，返回的远程地址将是实际连接客户端中的地址。使用 proxy-headers 选项时，该值不能为 true。

这在在兼容 https 透传代理后面运行时很有用，因为无法操作请求标头。

例如：

```
bin/kc.[sh|bat] start --proxy-protocol-enabled true
```

8.8. 启用客户端证书查找

当代理配置为 TLS 终止代理时，客户端证书信息可以通过特定的 HTTP 请求标头转发到服务器，然后用来验证客户端。您可以配置服务器如何检索客户端证书信息，具体取决于您使用的代理。

警告

通过 X.509 身份验证的代理标头进行客户端证书查找被视为安全敏感。如果配置错误，则使用伪客户端证书标头进行身份验证。需要采取额外的预防措施，以确保在通过代理标头传递时可以信任客户端证书信息。

- 仔细检查您的用例需要重新加密或边缘 TLS 终止，这意味着使用代理标头进行客户端证书查找。当需要 X.509 身份验证时，建议使用 TLS passthrough 作为更安全的选项，因为它不需要通过代理标头传递证书。代理标头中的客户端证书查找仅适用于重新加密和边缘 TLS 终止。
- 如果 passthrough 不是一个选项，请实现以下安全措施：
  - 配置网络，以便红帽构建的 Keycloak 被隔离，只能接受来自代理的连接。
  - 确保代理覆盖 spi-x509cert-lookup-<provider>-ssl-client-cert 选项中配置的标头。
  - 额外注意 spi-x509cert-lookup-<provider>-trust-proxy-verification 设置。请确保仅在信任代理以验证客户端证书时启用它。在没有 代理验证客户端证书链的情况下，设置 spi-x509cert-lookup-<provider>-trust-proxy-verification=true 将红帽构建的 Keycloak 公开给安全漏洞。

服务器支持一些最常见的 TLS 终止代理，例如：

Proxy	供应商
Apache HTTP 服务器	Apache
HAProxy	hapoxy

Proxy	供应商
NGINX	nginx

要配置如何从您需要的请求检索客户端证书：

启用对应的代理供应商

```
bin/kc.[sh|bat] build --spi-x509cert-lookup-provider=<provider>
```

配置 HTTP 标头

```
bin/kc.[sh|bat] start --spi-x509cert-lookup-<provider>-ssl-client-cert=SSL_CLIENT_CERT --spi-x509cert-lookup-<provider>-ssl-cert-chain-prefix=CERT_CHAIN --spi-x509cert-lookup-<provider>-certificate-chain-length=10
```

在配置 HTTP 标头时，您需要确保使用的值与代理转发的标头名称对应。

配置供应商的可用选项包括：

选项	描述
ssl-client-cert	保存客户端证书的标头名称
ssl-cert-chain-prefix	标头包含链中额外证书的前缀，并用于在链的长度中相应地检索单个证书。例如，值 **CERT_CHAIN** 将告知服务器将标头 **CERT_CHAIN_0** 到 **CERT_CHAIN_9**（如果 **certificate-chain-length** 设为 **10**）加载额外的证书。

选项	描述
certificate-chain-length	证书链的最大长度。
trust-proxy-verification	启用信任 NGINX 代理证书验证，而不是将证书转发到红帽构建的 Keycloak，并在红帽构建的 Keycloak 中进行验证。
cert-is-url-encoded	转发的证书是 url 编码的。在 NGINX 中，这与 **$ssl_client_cert** 和 **$ssl_client_escaped_cert** 变量对应。这也可用于 Traefik PassTlsClientCert 中间件，因为它发送客户端证书未编码。

8.8.1. 配置 NGINX 供应商

NGINX SSL/TLS 模块不会公开客户端证书链。红帽构建的 Keycloak 的 NGINX 证书查找供应商使用红帽构建的 Keycloak 信任存储重建它。

如果您使用这个供应商，请参阅 配置可信证书 以了解如何配置红帽构建的 Keycloak Truststore。

8.9. 相关选项

	value
hostname 服务器所公开的地址。 可以是完整的 URL，也可以是主机名。当只提供主机名时，会从请求解析端口和上下文路径。 CLI: -- **hostname** Env: **KC_HOSTNAME** 仅在启用 hostname:v2 功能时才可用	
hostname-admin 用于访问管理控制台的地址。 如果您使用反向代理在与 **hostname** 选项中指定的不同地址上公开管理控制台，则使用这个选项。 CLI: **--hostname-admin** Env: **KC_HOSTNAME_ADMIN** 仅在启用 hostname:v2 功能时才可用	

	value
http-relative-path ▮ 设置用于服务资源的路径相对于 / 的路径。 该路径必须以 / 开头。 CLI: **--http-relative-path** Env: **KC_HTTP_RELATIVE_PATH**	/ （默认）
proxy-headers 服务器应接受的代理标头。 错误配置可能会使服务器暴露于安全漏洞。优先于已弃用的代理选项。 CLI: **--proxy-headers** Env: **KC_PROXY_HEADERS**	**转发**,**xforwarded**
proxy-protocol-enabled 服务器是否应该在代理后提供请求时使用 HA PROXY 协议。 当设置为 true 时，返回的远程地址将是实际连接客户端中的地址。使用**proxy-headers** 时无法启用。 CLI: **--proxy-protocol-enabled** Env: **KC_PROXY_PROTOCOL_ENABLED**	**true**,**false** （默认）
proxy-trusted-addresses 以逗号分隔的可信代理地址列表。 如果设置，则忽略来自其他地址的代理标头。默认情况下，所有地址都是可信的。可信代理地址被指定为 IP 地址(IPv4 或 IPv6)或无类别域间路由(CIDR)标记。仅在设置 proxy-headers 时可用。 CLI: **--proxy-trusted-addresses** Env: **KC_PROXY_TRUSTED_ADDRESSES**	

第 9 章 配置数据库

为红帽构建的 Keycloak 配置关系数据库，以存储用户、客户端和域数据。

本章介绍了如何配置红帽构建的 Keycloak 服务器，以将数据存储在关系数据库中。

9.1. 支持的数据库

服务器对不同的数据库有内置支持。您可以通过查看 db 配置选项的预期值来查询可用的数据库。下表列出了支持的数据库及其经过测试的版本。

数据库	选项值	测试的版本
MariaDB Server	**mariadb**	11.4
Microsoft SQL Server	**mssql**	2022
MySQL	**mysql**	8.4
Oracle 数据库	**oracle**	23.5
PostgreSQL	**postgres**	17
Amazon Aurora PostgreSQL	**postgres**	16.1

默认情况下，服务器使用 dev-file 数据库。这是服务器用来持久保留数据的默认数据库，仅适用于开发用例。dev-file 数据库不适用于生产环境的用例，必须在部署到生产环境前替换。

9.2. 安装数据库驱动程序

数据库驱动程序作为红帽构建的 Keycloak 的一部分提供，但 Oracle 数据库和 Microsoft SQL Server 驱动程序除外。

如果要连接到其中一个数据库，或者要连接到已包含数据库驱动程序的不同数据库，请手动安装缺少的驱动程序。

9.2.1. 安装 Oracle 数据库驱动程序

要为红帽构建的 Keycloak 安装 Oracle 数据库驱动程序：

1. 从以下源之一下载 ojdbc17 和 orai18n JAR 文件：
   a. 从 Oracle 驱动程序下载页面 zipped JDBC 驱动程序和 Companion Jars 版本 23.6.0.24.10。
   b. 通过 ojdbc17 和 orai18n 的 Maven Central。
   c. 数据库厂商推荐的安装介质适用于使用的特定数据库。
2. 在运行 unzipped 分发时：使用红帽构建的 Keycloak 提供程序 文件夹中的 ojdbc17 和 orai18n JAR 文件
3. 在运行容器时：构建自定义红帽构建的 Keycloak 镜像，并在 provider 文件夹中添加 JAR。在为 Operator 构建自定义镜像时，这些镜像需要使用红帽构建的 Keycloak 集合的所有构建时选项来优化镜像。

构建可用于红帽构建的 Keycloak Operator 的镜像的最小 Containerfile，包括从 Maven Central 下载的 Oracle Database JDBC 驱动程序，如下所示：

```
FROM registry.redhat.io/rhbk/keycloak-rhel9:26.2
ADD --chown=keycloak:keycloak --chmod=644
https://repo1.maven.org/maven2/com/oracle/database/jdbc/ojdbc17/23.6.0.24.10/ojdbc1
7-23.6.0.24.10.jar /opt/keycloak/providers/ojdbc17.jar
ADD --chown=keycloak:keycloak --chmod=644
https://repo1.maven.org/maven2/com/oracle/database/nls/orai18n/23.6.0.24.10/orai18n-
23.6.0.24.10.jar /opt/keycloak/providers/orai18n.jar
# Setting the build parameter for the database:
ENV KC_DB=oracle
# Add all other build parameters needed, for example enable health and metrics:
ENV KC_HEALTH_ENABLED=true
ENV KC_METRICS_ENABLED=true
# To be able to use the image with the Red Hat build of Keycloak Operator, it needs to
be optimized, which requires Red Hat build of Keycloak's build step:
RUN /opt/keycloak/bin/kc.sh build
```

如需有关如何 构建优化镜像的详细信息，请参阅在容器中运行 Keycloak 的 Keycloak 部分。

然后，继续配置数据库，如下一节中所述。

9.2.2. 安装 Microsoft SQL Server 驱动程序

要为红帽构建的 Keycloak 安装 Microsoft SQL Server 驱动程序：

1. 从以下源之一下载 mssql-jdbc JAR 文件：
   a. 从 用于 SQL Server 页的 Microsoft JDBC 驱动程序 下载版本。
   b. 通过 mssql-jdbc Maven Central。
   c. 数据库厂商推荐的安装介质适用于使用的特定数据库。
2. 在运行 unzipped 分发时：防止 Red Hat build of Keycloak 的 provider folder 中的 mssql-jdbc
3. 在运行容器时：构建自定义红帽构建的 Keycloak 镜像，并在 provider 文件夹中添加 JAR。当为红帽构建的 Keycloak Operator 构建自定义镜像时，这些镜像需要使用红帽构建的 Keycloak 集合的所有构建时选项优化镜像。

构建可用于红帽构建的 Keycloak Operator 的镜像的最小 Containerfile，包括从 Maven Central 下载的 Microsoft SQL Server JDBC 驱动程序，如下所示：

```
FROM registry.redhat.io/rhbk/keycloak-rhel9:26.2
ADD --chown=keycloak:keycloak --chmod=644
https://repo1.maven.org/maven2/com/microsoft/sqlserver/mssql-
jdbc/12.8.1.jre11/mssql-jdbc-12.8.1.jre11.jar /opt/keycloak/providers/mssql-jdbc.jar
# Setting the build parameter for the database:
ENV KC_DB=mssql
# Add all other build parameters needed, for example enable health and metrics:
ENV KC_HEALTH_ENABLED=true
```

```
ENV KC_METRICS_ENABLED=true
# To be able to use the image with the Red Hat build of Keycloak Operator, it needs to be optimized, which requires Red Hat build of Keycloak's build step:
RUN /opt/keycloak/bin/kc.sh build
```

如需有关如何 构建优化镜像的详细信息，请参阅在容器中运行 Keycloak 的 Keycloak 部分。

然后，继续配置数据库，如下一节中所述。

9.3. 配置数据库

对于每个支持的数据库，服务器提供了一些建议的默认值来简化数据库配置。您可以通过提供一些关键设置（如数据库主机和凭证）来完成配置。

可以在 构建 命令或 start 命令中设置配置：

1. 使用 构建 命令，后跟一个优化的 start 命令（推荐）

   首先，在 conf/keycloak.conf 中指定连接到数据库所需的最小设置：

   ```
   # The database vendor.
   db=postgres

   # The username of the database user.
   db-username=keycloak

   # The password of the database user.
   db-password=change_me

   # Sets the hostname of the default JDBC URL of the chosen vendor
   db-url-host=keycloak-postgres
   ```

   然后，以下命令会根据配置选项创建一个新的和优化的服务器镜像，并启动服务器。

   ```
   bin/kc.[sh|bat] build
   bin/kc.[sh|bat] start --optimized
   ```

2. 仅使用 start 命令（没有 优化的）

```
bin/kc.[sh|bat] start --db postgres --db-url-host keycloak-postgres --db-username keycloak --db-password change_me
```

警告

上面的示例包括连接到数据库所需的最小设置，但它会公开数据库密码，我们不建议这样做。使用如上所示的 conf/keycloak.conf、环境变量或密钥存储（至少为密码）。

默认模式是 keycloak，但您可以使用 db-schema 配置选项更改它。

在 导入和导出域或 Bootstrapping 并在恢复管理员帐户 时，也可以配置数据库：

```
bin/kc.[sh|bat] import --help
bin/kc.[sh|bat] export --help
bin/kc.[sh|bat] bootstrap-admin --help
```

如需更多信息，请参阅配置红帽构建的 Keycloak。

9.4. 覆盖默认连接设置

服务器使用 JDBC 作为底层技术与数据库通信。如果默认连接设置不足，您可以使用 db-url 配置选项指定 JDBC URL。

以下是 PostgreSQL 数据库的示例命令。

```
bin/kc.[sh|bat] start --db postgres --db-url jdbc:postgresql://mypostgres/mydatabase
```

请注意，在调用包含特殊 shell 字符（如 ）的命令时，您需要转义字符 ；因此，您可能希望在配置文件中设置它。

9.5. 覆盖默认 JDBC 驱动程序

服务器根据您选择的数据库相应地使用默认的 JDBC 驱动程序。

要设置不同的驱动程序，您可以使用 JDBC 驱动程序的完全限定域名设置 db-driver ：

```
bin/kc.[sh|bat] start --db postgres --db-driver=my.Driver
```

无论您设置的任何驱动程序，默认驱动程序始终在运行时可用。

只有在您真正需要时才设置此属性。例如，当将 JDBC 驱动程序 Wrapper 的功能用于特定云数据库服务时。

9.6. 为数据库配置 UNICODE 支持

Unicode 对所有字段的支持取决于数据库是否允许 VARCHAR 和 CHAR 字段使用 Unicode 字符集。

- 如果可以设置这些字段，则 Unicode 可能可以正常工作，通常以字段长度为代价。
- 如果数据库只支持 NVARCHAR 和 NCHAR 字段中的 Unicode，则 Unicode 对所有文本字段的支持不太可能工作，因为服务器架构广泛使用 VARCHAR 和 CHAR 字段。

数据库模式只为以下特殊字段提供对 Unicode 字符串的支持：

- realm ：显示名称、HTML 显示名称、本地化文本（键和值）
- 联邦 供应商：显示名称

- Users: username, given name, last name, attribute name and values
- groups ：名称、属性名称和值
- roles: name
- 对象描述

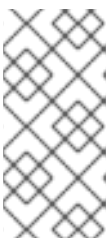
否则，字符仅限于数据库编码中包含的字符，通常为 8 位。但是，对于某些数据库系统，您可以启用 Unicode 字符的 UTF-8 编码，并在所有文本字段中使用完整的 Unicode 字符。对于给定的数据库，这个选择可能会导致最大字符串长度比 8 位编码支持的最大字符串长度要短。

9.6.1. 为 Oracle 数据库配置 Unicode 支持

如果在 VARCHAR 和 CHAR 字段中使用 Unicode 支持创建数据库，则 Oracle 数据库中支持 Unicode 字符。例如，您配置了 AL32UTF8 作为数据库字符集。在这种情况下，jdbc 驱动程序不需要特殊设置。

如果没有使用 Unicode 支持创建数据库，您需要配置 JDBC 驱动程序来支持特殊字段中的 Unicode 字符。您可以配置两个属性。请注意，您可以将这些属性配置为系统属性或连接属性。

1. 将 oracle.jdbc.defaultNChar 设置为 true。
2. （可选）将 oracle.jdbc.convertNcharLiterals 设置为 true。



注意

有关这些属性和任何性能影响的详情，请查看 Oracle JDBC 驱动程序配置文档。

9.6.2. 对 Microsoft SQL Server 数据库的 Unicode 支持

Unicode 字符只支持 Microsoft SQL Server 数据库的特殊字段。数据库不需要特殊设置。

JDBC 驱动程序的 sendStringParametersAsUnicode 属性应设为 false，以显著提高性能。如果没有此参数，Microsoft SQL Server 可能无法使用索引。

9.6.3. 为 MySQL 数据库配置 Unicode 支持

如果使用 CREATE DATABASE 命令，如果在 VARCHAR 和 CHAR 字段中使用 Unicode 支持创建数据库，则 MySQL 数据库中支持 Unicode 字符。

请注意，由于 utf8 字符集的不同存储要求，不支持 utf8mb4 字符集。详情请查看 MySQL 文档。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字符数，而不是字节。如果数据库默认字符集不允许 Unicode 存储，则只有特殊字段可以存储 Unicode 值。

1. 启动 MySQL 服务器。
2. 在 JDBC 驱动程序设置下，找到 JDBC 连接设置。
3. 添加此连接属性： characterEncoding=UTF-8

9.6.4. 为 PostgreSQL 数据库配置 Unicode 支持

当数据库字符设置为 UTF8 时，PostgreSQL 数据库支持 Unicode。对于非特殊字段，任何字段都可使用 Unicode 字符，且不能减少字段长度。JDBC 驱动程序不需要特殊设置。字符集是在创建 PostgreSQL 数据库时决定的。

1. 输入以下 SQL 命令，检查 PostgreSQL 集群的默认字符集。

```
show server_encoding;
```

2. 如果默认字符集不是 UTF 8，请使用命令创建使用 UTF8 作为默认字符集的数据库，例如：

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

9.7. 准备 AMAZON AURORA POSTGRESQL

使用 Amazon Aurora PostgreSQL 时， Amazon Web Services JDBC 驱动程序提供额外的功能，如在 Multi-AZ 设置中的写器实例更改时传输数据库连接。这个驱动程序不是发行版的一部分，需要在使用前安装它。

要安装这个驱动程序，请执行以下步骤：

1. 在运行 unzipped 分发时：从 Amazon Web Services JDBC 驱动程序发行版本页面 下载 JAR 文件，并将其放在红帽构建的 Keycloak 的供应商 文件夹中。

2. 在运行容器时：构建自定义红帽构建的 Keycloak 镜像，并在 provider 文件夹中添加 JAR。

   构建可用于 Red Hat build of Keycloak Operator 的镜像的最小 Containerfile 类似如下：

```
FROM registry.redhat.io/rhbk/keycloak-rhel9:26.2
ADD --chmod=0666 https://github.com/awslabs/aws-advanced-jdbc-wrapper/releases/download/2.3.1/aws-advanced-jdbc-wrapper-2.3.1.jar /opt/keycloak/providers/aws-advanced-jdbc-wrapper.jar
```

   如需了解如何 构建优化的镜像，请参阅使用自定义红帽构建的 Keycloak 镜像部分运行红帽 Keycloak 镜像的 红帽构建的 Keycloak 镜像 部分，了解如何使用 Keycloak Operator 的优化和非优化的镜像。

3. 配置红帽构建的 Keycloak 以使用以下参数运行：

   db-url

   将 aws-wrapper 插入到常规 PostgreSQL JDBC URL 中，生成类似 jdbc:aws-wrapper:postgresql:// 的 URL。

   db-driver

   设置为 software.amazon.jdbc.Driver，以使用 AWS JDBC 包装程序。

9.8. 准备 MYSQL 服务器

从 MySQL 8.0.30 开始，MySQL 支持为在没有显式主密钥的情况下创建的 InnoDB 表生成不可见的主密钥(这里提供了更多信息)。如果启用了此功能，数据库架构初始化并且迁移将失败，并显示出错信息 Multiple primary key defined (1068)。然后，您需要在安装或升级 Red Hat build of Keycloak 前，在 MySQL 服务器配置中将 sql_generate_in visible_primary_key 设置为 OFF 来禁用它。

9.9. 在集群配置中更改数据库锁定超时

由于集群节点可同时引导，因此对数据库操作需要额外的时间。例如，引导服务器实例可以执行一些数据库迁移、导入或首次初始化。数据库锁定可防止在集群节点同时引导时启动操作相互冲突。

此锁定的最大超时时间为 900 秒。如果节点在此锁定中等待超过超时时间，引导会失败。需要更改默认值不太可能，但您可以输入以下命令来更改它：

```
bin/kc.[sh|bat] start --spi-dblock-jpa-lock-wait-timeout 900
```

9.10. 使用带有 XA 事务支持的数据库供应商

默认情况下，Red Hat build of Keycloak 使用非 XA 事务和适当的数据库驱动程序。

如果要使用驱动程序提供的 XA 事务支持，请输入以下命令：

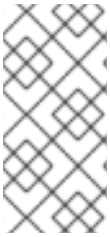
```
bin/kc.[sh|bat] build --db=<vendor> --transaction-xa-enabled=true
```

红帽构建的 Keycloak 会自动为您的供应商选择适当的 JDBC 驱动程序。

注意

某些供应商（如 Azure SQL 和 MariaDB Galera）不支持或依赖 XA 事务机制。

XA 恢复默认为启用，并使用文件系统位置 KEYCLOAK_HOME/data/transaction-logs 存储事务日志。



注意

在容器化环境中启用 XA 事务并不完全支持 XA 恢复，除非该路径上提供了稳定的存储。

9.11. 为 MIGRATIONSTRATEGY 设置 JPA PROVIDER CONFIGURATION 选项

要设置 JPA migrationStrategy (manual/update/validate)，您应该设置 JPA 供应商，如下所示：

为 connection- jpa SPI 的 quarkus 供应商设置 migration- strategy

```
bin/kc.[sh|bat] start --spi-connections-jpa-quarkus-migration-strategy=manual
```

如果要获取 DB 初始化的 SQL 文件，还必须添加这个额外的 SPI initializeEmpty (true/false)：

为 connection- jpa SPI 的 quarkus 供应商设置 initialize- empty

```
bin/kc.[sh|bat] start --spi-connections-jpa-quarkus-initialize-empty=false
```

与 migrationExport 指向特定文件和位置的方法相同：

为 connection- jpa SPI 的 quarkus 供应商设置 migration- export

```
bin/kc.[sh|bat] start --spi-connections-jpa-quarkus-migration-export=<path>/<file.sql>
```

如需更多信息，请参阅 迁移数据库 文档。

9.12. 相关选项

	value
db ▮ 数据库供应商。 在 production 模式中，**dev-file** 的默认值已弃用，您应该明确指定 db。 CLI: -- **db** Env: **KC_DB**	**dev-file** (default), **dev-mem**,**mariadb**,**mssql**, **mysql**,**oracle**,**postgres**
db-driver ▮ JDBC 驱动程序的完全限定类名称。 如果没有设置，则会将默认驱动程序相应地设置为所选数据库。 CLI: **--db-driver** Env: **KC_DB_DRIVER**	
db-password 数据库用户的密码。 CLI: **--db-password** Env: **KC_DB_PASSWORD**	
db-pool-initial-size 连接池的初始大小。 CLI: **--db-pool-initial-size** Env: **KC_DB_POOL_INITIAL_SIZE**	
db-pool-max-size 连接池的最大大小。 CLI: **--db-pool-max-size** Env: **KC_DB_POOL_MAX_SIZE**	**100** （默认）

	value
db-pool-min-size 连接池的最小大小。 CLI: **--db-pool-min-size** Env: **KC_DB_POOL_MIN_SIZE**	
db-schema 要使用的数据库模式。 CLI: **--db-schema** Env: **KC_DB_SCHEMA**	
db-url 完整的数据库 JDBC URL。 如果没有提供，则会根据所选数据库供应商设置默认 URL。例如，如果使用 **postgres**，默认的 JDBC URL 为 **jdbc:postgresql://localhost/keycloak**。 CLI: **--db-url** Env: **KC_DB_URL**	
db-url-database 设置所选供应商的默认 JDBC URL 的数据库名称。 如果设置了 **db-url** 选项，则忽略这个选项。 CLI: **--db-url-database** Env: **KC_DB_URL_DATABASE**	
db-url-host 设置所选供应商的默认 JDBC URL 的主机名。 如果设置了 **db-url** 选项，则忽略这个选项。 CLI: **--db-url-host** Env: **KC_DB_URL_HOST**	
db-url-port 设置所选供应商的默认 JDBC URL 端口。 如果设置了 **db-url** 选项，则忽略这个选项。 CLI: **--db-url-port** Env: **KC_DB_URL_PORT**	

	value
db-url-properties 设置所选供应商的默认 JDBC URL 的属性。 确保将属性相应地设置为数据库供应商期望的格式，并在此属性值的开头附加正确的字符。如果设置了 **db-url** 选项，则忽略这个选项。 CLI: **--db-url-properties** Env: **KC_DB_URL_PROPERTIES**	
db-username 数据库用户的用户名。 CLI: **--db-username** Env: **KC_DB_USERNAME**	
Transaction-xa-enabled wagon 如果设置为 true，则使用 XA 数据源。 CLI: **--transaction-xa-enabled** Env: **KC_TRANSACTION_XA_ENABLED**	**true**,**false** （默认）

第 10 章 配置分布式缓存

将缓存层配置为多个红帽构建的 Keycloak 实例，并提高性能。

Red Hat build of Keycloak 专为高可用性和多节点集群设置而设计。当前分布式缓存实施基于 Infinispan （一个高性能、可分布式内存数据网格）之上。

10.1. 启用分布式缓存

当您在生产环境模式下启动 Red Hat build of Keycloak 时，使用 start 命令启用缓存，并发现网络中的所有 Red Hat build of Keycloak 节点。

默认情况下，缓存使用 jdbc-ping 堆栈，该堆栈基于 TCP 传输，并使用配置的数据库来跟踪加入集群的节点。红帽构建的 Keycloak 允许您从一组预定义的默认传输堆栈中选择，或者定义您自己的自定义堆栈，如本章后文所述。

要显式启用分布式 infinispan 缓存，请输入以下命令：

```
bin/kc.[sh|bat] start --cache=ispn
```

当您在开发模式下启动红帽构建的 Keycloak 时，通过使用 start-dev 命令，红帽构建的 Keycloak 仅使用本地缓存，通过隐式设置-- cache=local 选项完全禁用分布式缓存。本地 缓存模式仅用于开发和测试目的。

10.2. 配置缓存

Red Hat build of Keycloak 提供了一个缓存配置文件，该文件位于 conf/cache-ispn.xml。

缓存配置是常规的 {infinispan_configuring_docs}[Infinispan 配置文件]。

下表提供了红帽构建的 Keycloak 使用的特定缓存概述。您可以在 conf/cache-ispn.xml 中配置这些缓存：

缓存名称	缓存类型	描述
realms	Local	缓存持久的域数据
users	Local	缓存持久的用户数据
授权	Local	缓存持久的授权数据
keys	Local	缓存外部公钥
crl	Local	X.509 authenticator CRL 的缓存
work	复制	在节点间传播无效消息
authenticationSessions	分布式	缓存身份验证会话，在身份验证过程中创建/销毁/过期
会话	分布式	缓存持久的用户会话数据
clientSessions	分布式	缓存持久的客户端会话数据
offlineSessions	分布式	缓存保留离线用户会话数据
offlineClientSessions	分布式	缓存保留离线客户端会话数据
loginFailures	分布式	跟踪失败的登录、欺诈检测
actionTokens	分布式	缓存操作令牌

10.2.1. 缓存类型和默认值

本地缓存

红帽 Keycloak 的构建会在本地缓存持久数据，以避免对数据库进行不必要的往返。

使用本地缓存将以下数据保存在集群中的每个节点中：

- **域和 相关数据，如客户端、角色和组。**
- **用户和 相关数据，如授权角色和组成员身份。**

- 授权和 相关数据，如资源、权限和策略。
- keys

域、用户和授权的本地缓存配置为默认容纳最多 10,000 个条目。本地密钥缓存每默认值最多可容纳 1,000 个条目，默认为每小时过期。因此，密钥会被强制从外部客户端或身份提供程序定期下载。

为了实现最佳运行时并避免对数据库的额外往返，您应该考虑查看每个缓存的配置，以确保最大条目数与数据库的大小一致。您可以缓存更多条目，通常服务器需要从数据库获取数据。您应该评估内存利用率和性能之间的利弊。

本地缓存无效

本地缓存提高了性能，但在多节点设置中添加了一个挑战。

当一个红帽构建的 Keycloak 节点更新共享数据库中数据时，所有其他节点都需要了解它，因此它们会导致数据从其缓存中无效。

工作 缓存是一种复制缓存，用于发送这些失效消息。这个缓存中的条目/消息非常短，您不应该预期这个缓存的大小会随时间增长。

身份验证会话

每当用户尝试进行身份验证时，都会创建身份验证会话。身份验证过程完成后或达到其过期时间后会自动销毁。

authenticationSessions 分布式缓存用于存储身份验证会话，以及在身份验证过程中与其关联的任何其他数据。

通过依赖可分布式缓存，身份验证会话可供集群中的任何节点使用，以便用户可以重定向到任何节点，而不会丢失其身份验证状态。但是，生产环境就绪的部署应始终考虑会话关联，而是将用户重定向到初始创建会话的节点。通过这样做，您要避免节点间不必要的状态传输，并改进 CPU、内存和网络利用率。

用户会话

经过身份验证后，将创建一个用户会话。用户会话跟踪您的活跃用户及其状态，以便他们可以无缝地向任何应用程序进行身份验证，而无需再次要求其凭证。对于每个应用程序，用户使用客户端会话进行身份验证，以便服务器可以根据应用程序跟踪用户通过 进行身份验证的应用程序及其状态。

当用户执行注销时，用户和客户端会话都会自动销毁，客户端会执行令牌撤销，或者因为达到它们的过期时间而被销毁。

会话数据默认存储在数据库中，并按需载入到以下缓存：

- 会话
- clientSessions

通过依赖可分布式缓存，缓存的用户和客户端会话可供集群中的任何节点使用，以便用户可以重定向到任何节点，而无需从数据库加载会话数据。但是，生产环境就绪的部署应始终考虑会话关联，而是将用户重定向到初始创建会话的节点。通过这样做，您要避免节点间不必要的状态传输，并改进 CPU、内存和网络利用率。

对于用户会话和客户端会话的内存缓存，默认为每个节点，10000 个条目会减少红帽构建的 Keycloak 的总内存用量。内部缓存将只为每个缓存条目使用一个所有者运行。

离线用户会话

作为 OpenID Connect Provider，服务器能够对用户进行身份验证并发布离线令牌。在身份验证成功后发出离线令牌时，服务器会创建一个离线用户会话和离线客户端会话。

以下缓存用于存储离线会话：

- offlineSessions

- offlineClientSessions

与用户和客户端会话缓存一样，离线用户和客户端会话缓存默认限制为每个节点 10000 条目。根据需要，从内存驱除的项目将从数据库根据需要加载。

密码过期强制检测

loginFailures 分布式缓存用于跟踪失败登录尝试的数据。对于在多节点红帽构建的 Keycloak 设置中工作，Brute Force Protection 功能需要这个缓存。

操作令牌

当用户需要异步确认操作（例如，忘记密码流发送的电子邮件中）时，可以使用操作令牌。actionTokens 分布式缓存用于跟踪有关操作令牌的元数据。

10.2.2. volatile 用户会话

默认情况下，常规用户会话存储在数据库中，并按需加载到缓存中。可以配置红帽构建的 Keycloak，以仅将常规用户会话存储在缓存中，并最小化对数据库的调用。

由于此设置中的所有会话都存储在内存中，因此与此相关的两个副作用：

- 当所有红帽构建的 Keycloak 节点重启时，会丢失会话。
- 增加了内存消耗。

在使用易失性用户会话时，缓存是用户和客户端会话的真实来源。Red Hat build of Keycloak 会自动调整可在内存中存储的条目数量，并增加副本数以防止数据丢失。

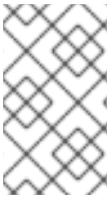
警告

由于可能较高的内存用量，不建议在大量使用离线会话时使用易失性用户会话。对于易失性会话，可以在内存中缓存时间离线会话，使用 SPI 选项 spi-user-sessions-infinispan-offline-client-session-cache-entry-lifespan-override 和 spi-user-sessions-infinispan-offline-session-cache-entry-lifespan-override。

按照以下步骤启用此设置：

1. 使用以下命令禁用 persistent-user-sessions 功能：

```
bin/kc.sh start --features-disabled=persistent-user-sessions ...
```



注意

启用 多站点功能时，无法禁用 persistent- user-sessions。

10.2.3. 配置缓存最大大小

为了减少内存用量，可以将上限放在存储在给定缓存中的条目数。要在缓存上指定 的上限，您必须提供以下命令行参数 --cache-embedded-${CACHE_NAME}-max-count=，用 ${CACHE_NAME} 替换要应用上限的缓存名称。例如，要将上限 1000 应用到 offlineSessions 缓存，您需要 configure --cache-embedded-offline-sessions-max-count=1000。在以下缓存上不能定义上限：actionToken,authenticationSessions,loginFailures,work。

在启用易失性会话时，不支持为 会话、clientSessions、offlineSessions 和 offlineClientSessions 设置最大缓存大小。

10.2.4. 指定您自己的缓存配置文件

要指定您自己的缓存配置文件，请输入以下命令：

```
bin/kc.[sh|bat] start --cache-config-file=my-cache-file.xml
```

配置文件相对于 conf/ 目录。

10.2.5. 远程服务器的 CLI 选项

为了配置红帽构建的 Keycloak 服务器配置，以实现高可用性和多节点集群设置，包括以下 CLI 选项 cache-remote-host、cache-remote-port、cache-remote-username 和 cache-remote-password 简化 XML 文件中的配置。存在任何声明的 CLI 参数后，XML 文件中应当没有与远程存储相关的配置。

10.2.5.1. 连接到不安全的 Infinispan 服务器

警告

不建议在生产环境中禁用安全性！

在开发或测试环境中，启动不安全的 Infinispan 服务器更容易。对于这些用例，CLI 选项 cache-remote-tls-enabled 会禁用红帽构建的 Keycloak 和 Data Grid 之间的加密(TLS)。如果 Data Grid 服务器被配置为只接受加密的连接，则 Red Hat build of Keycloak 无法启动。

CLI 选项 cache-remote-username 和 cache-remote-password 是可选的，如果未设置，红帽构建的 Keycloak 将连接到 Data Grid 服务器，而无需提供任何凭证。如果 Data Grid 服务器启用了身份验证，红帽构建的 Keycloak 将无法启动。

10.3. 传输堆栈

传输堆栈确保红帽以可靠的方式在集群中构建 Keycloak 节点。Red Hat build of Keycloak 支持各种传输堆栈：

- jdbc-ping
- kubernetes

- **jdbc-ping-udp （已弃用）**
- **tcp （已弃用）**
- **UDP （已弃用）**
- **ec2 （已弃用）**
- **Azure （已弃用）**
- **Google （已弃用）**

要应用特定的缓存堆栈，请输入以下命令：

```
bin/kc.[sh|bat] start --cache-stack=<stack>
```

启用分布式缓存时，默认堆栈设置为 jdbc-ping，该缓存与 Red Hat build of Keycloak 的 26.x 发行流中的默认值向后兼容。

10.3.1. 可用的传输堆栈

下表显示了在没有进一步配置的情况下可用的传输堆栈，它们比使用 --cache-stack 运行时选项不同：

堆栈名称	传输协议	Discovery（发现）
jdbc-ping	TCP	使用 JGroups **JDBC_PING2** 协议的数据库注册表.
jdbc-ping-udp （已弃用）	UDP	使用 JGroups **JDBC_PING2** 协议的数据库注册表.

下表显示了使用- cache-stack 运行时选项和最低配置可用的传输堆栈：

堆栈名称	传输协议	Discovery（发现）
kubernetes	TCP	使用 JGroups **DNS_PING** 协议进行 DNS 解析。它需要将 **jgroups.dns.query** 设置为无头服务 FQDN。
tcp（已弃用）	TCP	使用 JGroups **MPING** 协议进行 IP 多播。请参阅以下有关如何为每个集群配置唯一的 **jgroups.mcast_addr** 或 **jgroups.mcast_port**。
UDP（已弃用）	UDP	使用 JGroups **PING** 协议进行 IP 多播。请参阅以下有关如何为每个集群配置唯一的 **jgroups.mcast_addr** 或 **jgroups.mcast_port**。

使用 tcp、udp 或 jdbc-ping-udp 堆栈时，每个集群都必须使用不同的多播地址和/或端口，以便其节点组成不同的集群。默认情况下，红帽构建的 Keycloak 使用 239.6.7.8 作为 jgroups.mcast_addr 和 46655 的多播地址，用于多播端口 jgroups.mcast_port。

注意

use -D<property>=<value > 通过 JAVA_OPTS_APPEND 环境变量或 CLI 命令传递属性。

其他堆栈

建议您使用上面提供的其中一个堆栈。其他堆栈由 Infinispan 提供，但它超出了本指南的配置范围。如需了解更多文档，请参阅设置 Infinispan 集群传输 和自定义 JGroups 堆栈。

10.4. 保护传输堆栈

对于基于 TCP 的传输堆栈，默认启用使用 TLS 的加密，这也是默认配置。只要使用基于 TCP 的传输堆栈，就不需要额外的 CLI 选项或修改缓存 XML。

注意

如果您使用基于 UDP 或 TCP_NIO2 的传输堆栈，请按如下所示配置传输堆栈的加密：

1. 将选项 cache-embedded-mtls-enabled 设置为 false。
2. 按照 JGroups 加密文档中的文档和 加密集群传输。

启用 TLS 后，红帽构建的 Keycloak 会自动生成自签名 RSA 2048 位证书来保护连接，并使用 TLS 1.3 来保护通信。密钥和证书存储在数据库中，以便它们可供所有节点使用。默认情况下，证书在 60 天内有效，并在运行时每 30 天进行轮转。使用选项 cache-embedded-mtls-rotation-interval-days 来更改它。

10.4.1. 在服务网格内运行

当使用类似 Istio 的服务网格时，您可能需要允许红帽构建的 Keycloak Pod 之间的直接 mTLS 通信，以允许 mutual 身份验证正常工作。否则，您可能会看到 JGRP000006: failed accept connection from peer SSLSocket （表示显示错误的证书），集群将无法正确表单。

然后，您可以选择允许红帽构建的 Keycloak Pod 之间的 mTLS 通信，或者依赖服务网格传输安全性来加密通信并验证对等点。

在使用 Istio 时，为红帽构建的 Keycloak 允许直接 mTLS 通信：

- 应用以下配置以允许直接连接。

```
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: infinispan-allow-nomtls
spec:
  selector:
    matchLabels:
      app: keycloak (1)
```

```
portLevelMtls:
  "7800": 2
    mode: PERMISSIVE
```

1 更新标签以匹配红帽构建的 Keycloak 部署。

2 端口 7800 是默认值。如果您更改了数据传输端口，请调整它。

另外，要禁用 mTLS 通信，并依赖服务网格来加密流量：

- 将选项 cache-embedded-mtls-enabled 设置为 false。
- 将您的服务网格配置为仅授权来自其他红帽构建的 Keycloak Pod 的流量进行数据传输端口（默认为 7800）。

10.4.2. 提供您自己的密钥和证书

虽然不建议标准设置，但在特定设置中至关重要，您可以使用传输堆栈的证书手动配置密钥存储。cache-embedded-mtls-key-store-file 设置密钥存储的路径，而 cache-embedded-mtls-key-store-password 会设置密码来解密它。truststore 包含用来接受连接的有效证书，它可以使用 cache-embedded-mtls-trust-store-file （信任存储的路径）和 cache-embedded-mtls-trust-store-password （解密它的密码）。要限制未授权的访问，请始终为每个红帽构建的 Keycloak 部署使用自签名证书。

10.5. 网络端口

为确保红帽构建的 Keycloak 集群健康，需要打开一些网络端口。下表显示了需要为 jdbc-ping 堆栈打开的 TCP 端口，以及流量通过它的描述。

port	属性	描述
7800	jgroups.bind.port	单播数据传输.

port	属性	描述
57800	jgroups.fd.port-offset	协议 FD_SOCK2 的故障检测.它侦听一个套接字的关闭，以怀疑红帽构建的 Keycloak 服务器故障。jgroups.fd.port-offset 属性定义 jgroups.bind.port 的偏移量。

注意

use -D<property>=<value > 修改 JAVA_OPTS_APPEND 环境变量或 CLI 命令中上面的端口。

10.6. 网络绑定地址

为确保红帽健康的 Keycloak 集群构建，必须在集群所有其他节点访问的接口绑定该网络端口。

默认情况下，它选择站点本地（不可路由）IP 地址，例如，从 192.168.0.0/16 或 10.0.0.0/8 地址范围中。

若要覆盖地址，请设置 jgroups.bind.address 属性。

注意

use -Djgroups.bind.address=<IP > 修改 JAVA_OPTS_APPEND 环境变量或 CLI 命令中的绑定地址。

要只为 IPv6 设置并有红帽构建的 Keycloak 自动选择绑定地址，请使用以下设置：

```
export JAVA_OPTS_APPEND="-Djava.net.preferIPv4Stack=false -
Djava.net.preferIPv6Addresses=true"
```

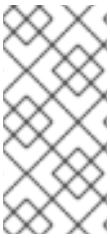
10.7. 在不同网络中运行实例

如果您在不同的网络中（如防火墙或容器中）运行红帽构建的 Keycloak 实例，则不同的实例将无法通

过其本地 IP 地址相互访问。在这种情况下，将端口转发规则（有时称为"虚拟服务器"）设置为其本地 IP 地址。

使用端口转发时，请使用以下属性，以便每个节点可以正确地向其他节点公告其外部地址：

属性	描述
jgroups.external_port	红帽构建的 Keycloak 集群中的其他实例应该用来联系这个节点的端口。
jgroups.external_addr	红帽构建的 Keycloak 中的其他实例的 IP 地址应该用来联系这个节点。



注意

use -D<property>=<value > 在 JAVA_OPTS_APPEND 环境变量或 CLI 命令中设定它。

10.8. 从缓存公开指标

在启用指标时，来自缓存的指标会自动公开。

要为缓存指标启用直方图，请将 cache-metrics-histograms-enabled 设置为 true。虽然这些指标提供了对延迟分布的深入了解，但收集它们可能会产生性能影响，因此您应该最好在已饱和的系统中激活它们。

```
bin/kc.[sh|bat] start --metrics-enabled=true --cache-metrics-histograms-enabled=true
```

有关如何启用指标的更多详细信息，请参阅使用指标 获取见解。

10.9. 相关选项

	value
cache 定义高可用性的缓存机制。 默认情况下，在生产环境模式中，使用 **ispn** 缓存在多个服务器节点之间创建集群。默认情况下，在开发模式中，**本地缓存** 会禁用集群，用于开发和测试目的。 CLI: -- **cache** Env: **KC_CACHE**	**ISPN** （默认），**local**
cache-config-file 定义应从中加载缓存配置的文件。 配置文件相对于 **conf/** 目录。 CLI: **--cache-config-file** Env: **KC_CACHE_CONFIG_FILE**	
cache-metrics-histograms-enabled 为嵌入式缓存的指标启用直方图。 CLI: **--cache-metrics-histograms-enabled** Env: **KC_CACHE_METRICS_HISTOGRAMS_ENABLED** 仅在启用指标时可用	**true**,**false** （默认）
cache-stack 定义用于集群通信和节点发现的默认堆栈。 如果没有设置，则默认为 **jdbc-ping**。 CLI: **--cache-stack** Env: **KC_CACHE_STACK** 仅在 'cache' type 设为 'ispn' 时才可用 使用 'jdbc-ping' 相反，方法是取消设置 Deprecated 值：**azure**,**ec2**,**google**,**tcp**,**udp**,**jdbc-ping-udp**	**jdbc-ping**,**kubernetes**,**jdbc-ping-udp** （已弃用）、**tcp** （已弃用）、**udp** （已弃用）、**ec2** （已弃用）、**azure** （已弃用）、**google** （已弃用）或任何

10.9.1. 嵌入式缓存

	value
cache-embedded-authorization-max-count 授权缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-authorization-max-count** Env: **KC_CACHE_EMBEDDED_AUTHORIZATION_MAX_COUNT**	
cache-embedded-client-sessions-max-count 客户端Sessions 缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-client-sessions-max-count** Env: **KC_CACHE_EMBEDDED_CLIENT_SESSIONS_MAX_COUNT** 仅在配置了嵌入式 Infinispan 集群时可用	
cache-embedded-crl-max-count crl 缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-crl-max-count** Env: **KC_CACHE_EMBEDDED_CRL_MAX_COUNT**	
cache-embedded-keys-max-count 密钥缓存可在内存中存储的最大条目数。 CLI: **--cache-embedded-keys-max-count** Env: **KC_CACHE_EMBEDDED_KEYS_MAX_COUNT**	
cache-embedded-mtls-enabled 加密 Keycloak 服务器之间的网络通信。 如果没有提供有关密钥存储和信任存储的额外参数，则会自动创建和轮转临时密钥对，这是标准设置的建议。 CLI: **--cache-embedded-mtls-enabled** Env: **KC_CACHE_EMBEDDED_MTLS_ENABLED** 仅在使用基于 TCP 的 cache-stack 时可用	**true**（默认），**false**

	value
cache-embedded-mtls-key-store-file Keystore 文件路径。 Keystore 必须包含 TLS 协议使用的证书。默认情况下，它会在 conf/ 目录下查找 **cache-mtls-keystore.p12**。 CLI: **--cache-embedded-mtls-key-store-file** Env: **KC_CACHE_EMBEDDED_MTLS_KEY_STORE_FILE** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	
cache-embedded-mtls-key-store-password 用于访问密钥存储的密码。 CLI: **--cache-embedded-mtls-key-store-password** Env: **KC_CACHE_EMBEDDED_MTLS_KEY_STORE_PASSWORD** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	
cache-embedded-mtls-rotation-interval-days 自动 JGroups MTLS 证书轮转周期（以天为单位）。 CLI: **--cache-embedded-mtls-rotation-interval-days** Env: **KC_CACHE_EMBEDDED_MTLS_ROTATION_INTERVAL_DAYS** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	**30** （默认）
cache-embedded-mtls-trust-store-file Truststore 文件路径。 它应包含可信证书或签发证书的证书颁发机构。默认情况下，它会在 conf/ 目录下查找 **cache-mtls-truststore.p12**。 CLI: **--cache-embedded-mtls-trust-store-file** Env: **KC_CACHE_EMBEDDED_MTLS_TRUST_STORE_FILE** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	
cache-embedded-mtls-trust-store-password 访问 Truststore 的密码。 CLI: **--cache-embedded-mtls-trust-store-password** Env: **KC_CACHE_EMBEDDED_MTLS_TRUST_STORE_PASSWORD** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	

	value
cache-embedded-offline-client-sessions-max-count offlineClientSessions 缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-offline-client-sessions-max-count** Env: **KC_CACHE_EMBEDDED_OFFLINE_CLIENT_SESSIONS_MAX_COUNT** 仅在配置了嵌入式 Infinispan 集群时可用	
cache-embedded-offline-sessions-max-count offlineSessions 缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-offline-sessions-max-count** Env: **KC_CACHE_EMBEDDED_OFFLINE_SESSIONS_MAX_COUNT** 仅在配置了嵌入式 Infinispan 集群时可用	
cache-embedded-realms-max-count 域缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-realms-max-count** Env: **KC_CACHE_EMBEDDED_REALMS_MAX_COUNT**	
cache-embedded-sessions-max-count 会话缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-sessions-max-count** Env: **KC_CACHE_EMBEDDED_SESSIONS_MAX_COUNT** 仅在配置了嵌入式 Infinispan 集群时可用	
cache-embedded-users-max-count 用户缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-users-max-count** Env: **KC_CACHE_EMBEDDED_USERS_MAX_COUNT**	

10.9.2. 远程缓存

	value
cache-remote-host 外部 Infinispan 集群的主机名。 仅在设置了功能 **多站点**、**无集群或 cache-embedded-remote-store** 时才可用。 CLI: **--cache-remote-host** Env: **KC_CACHE_REMOTE_HOST**	
cache-remote-password 对外部 Infinispan 集群进行身份验证的密码。 如果连接到不安全的外部 Infinispan 集群，则它是可选的。如果指定了这个选项，则需要 **cache-remote-username**。 CLI: **--cache-remote-password** Env: **KC_CACHE_REMOTE_PASSWORD** 仅在设置远程主机时可用	
cache-remote-port 外部 Infinispan 集群的端口。 CLI: **--cache-remote-port** Env: **KC_CACHE_REMOTE_PORT** 仅在设置远程主机时可用	**11222** （默认）
cache-remote-tls-enabled 启用 TLS 支持与安全远程 Infinispan 服务器通信。 建议在生产环境中启用。 CLI: **--cache-remote-tls-enabled** Env: **KC_CACHE_REMOTE_TLS_ENABLED** 仅在设置远程主机时可用	**true** （默认），**false**
cache-remote-username 外部 Infinispan 集群身份验证的用户名。 如果连接到不安全的外部 Infinispan 集群，则它是可选的。如果指定了 选项，则需要 **cache-remote-password**。 CLI: **--cache-remote-username** Env: **KC_CACHE_REMOTE_USERNAME** 仅在设置远程主机时可用	

第 11 章 配置传出 HTTP 请求

配置用于传出 HTTP 请求的客户端。

红帽构建的 Keycloak 通常需要向它保护的应用程序和服务发出请求。红帽构建的 Keycloak 使用 HTTP 客户端管理这些传出连接。本章介绍了如何配置客户端、连接池、代理设置、超时等。

11.1. 为 TLS 连接配置可信证书

如需了解如何配置红帽构建的 Keycloak Truststore，以便红帽构建的 Keycloak 能够使用 TLS 执行传出请求，请参阅配置可信证书。

11.2. 客户端配置命令

红帽构建的 Keycloak 用于传出通信的 HTTP 客户端是高度可配置的。要配置红帽构建的 Keycloak 传出 HTTP 客户端，请输入以下命令：

```
bin/kc.[sh|bat] start --spi-connections-http-client-default-<configurationoption>=<value>
```

以下是命令选项：

establish-connection-timeout-millis

建立连接超时前的最大时间（毫秒）。默认： 未设置。

socket-timeout-millis

两个数据数据包之间的最大不活跃时间，直到套接字连接超时（以毫秒为单位）。默认： 5000ms

connection-pool-size

用于传出连接的连接池的大小。默认：128。

max-pooled-per-route

每个主机可以池多少连接。默认：64。

connection-ttl-millis

以毫秒为单位的最大连接时间。默认： 未设置。

max-connection-idle-time-millis

闲置连接保留在连接池中的最大时间（以毫秒为单位）。闲置连接将通过后台清理线程从池中移除。将这个选项设置为 -1 以禁用此检查。默认： 900000。

disable-cookies

启用或禁用 Cookie 缓存。默认： true。

client-keystore

Java 密钥存储文件的文件路径。此密钥存储包含 mTLS 的客户端证书。

client-keystore-password

客户端密钥存储的密码。REQUIRED，当设置 client-keystore 时。

client-key-password

客户端的私钥密码。REQUIRED，当设置 client-keystore 时。

proxy-mappings

为传出 HTTP 请求指定代理配置。如需了解更多详细信息，请参阅 第 11.3 节 “传出 HTTP 请求的代理映射”。

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 development 和 never 在生产环境中使用，因为它将禁用 SSL 证书的验证。默认：false。

11.3. 传出 HTTP 请求的代理映射

要将传出请求配置为使用代理，您可以使用以下标准代理环境变量来配置代理映射：HTTP_PROXY、HTTPS_PROXY 和 NO_PROXY。

- HTTP_PROXY 和 HTTPS_PROXY 变量代表用于传出 HTTP 请求的代理服务器。红帽构建的 Keycloak 不会区分这两个变量。如果您定义了这两个变量，HTTPS_PROXY 优先于代理服务器使用的实际方案。

- NO_PROXY 变量定义不应使用代理的主机名列表。对于您指定的每个主机名，其所有子域也不包括在使用代理中。

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

代理映射和环境变量示例

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

在本例中，会出现以下结果：

- 所有传出请求都使用代理 https://www-proxy.acme.com:8080，除了对 google.com 或 google.com 的任何子域（如 auth.google.com ）的请求。
- login.facebook.com 及其所有子域不使用定义的代理，但 groups.facebook.com 使用代理，因为它不是 login.facebook.com 的子域。

11.4. 使用正则表达式进行代理映射

将环境变量用于代理映射的替代方法是为红帽构建 Keycloak 发送的传出请求配置以逗号分隔的 proxy-mapping 列表。proxy-mapping 由基于 regex 的主机名模式和 proxy-uri 组成，格式为 hostname-pattern;proxy-uri。

例如，请考虑以下正则表达式：

```
.*\.(google|googleapis)\.com
```

您可以输入以下命令应用基于 regex 的主机名模式：

```
bin/kc.[sh|bat] start --spi-connections-http-client-default-proxy-mappings='.*\\.
(google|googleapis)\\.com;http://www-proxy.acme.com:8080'
```

反斜杠字符 \ 再次转义，因为微配置集配置用于解析映射数组。

要确定传出 HTTP 请求的代理，会出现以下情况：

- 目标主机名与所有配置的主机名模式匹配。
- 使用第一个匹配模式的 proxy-uri。
- 如果没有配置的对象与主机名匹配，则不使用代理。

当代理服务器需要身份验证时，请使用用户名 username:password@ 包括代理用户的凭证。例如：

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

proxy-mapping 的正则表达式示例：

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

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

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

在本例中，会出现以下情况：

- proxy-uri 的特殊值 NO_PROXY 被使用，这意味着没有代理用于与相关主机名模式匹配的主

机。

- catch-all 模式结束 proxy-mappings，为所有传出请求提供默认代理。

11.5. 相关选项

	value
truststore-paths pkcs12 列表(p12、pfx 或 pkcs12 文件扩展)、PEM 文件或目录，其中包含要用作系统信任存储的这些文件。 CLI: **--truststore-paths** Env: **KC_TRUSTSTORE_PATHS**	

第 12 章 配置可信证书

配置红帽构建的 Keycloak Truststore，以通过 TLS 进行通信。

当红帽构建的 Keycloak 与外部服务通信或通过 TLS 有传入连接时，必须验证远程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。

这些客户端或服务器的证书或签署这些证书的 CA 必须放在信任存储中。然后，红帽构建的 Keycloak 将使用此信任存储。

12.1. 配置系统信任存储

现有的 Java 默认信任存储证书将始终被信任。如果您需要额外的证书，如果您有未经 JRE 可识别的自签名或内部证书颁发机构，则这些证书将包含在 conf/truststores 目录或子目录中。证书可能位于 PEM 文件中，或 PKCS12 文件，扩展为 .p12、.pfx 或 .pkcs12。在 PKCS12 中，证书必须是未加密的 - 表示不需要密码。

如果您需要替代路径，请使用 --truststore-paths 选项指定 PEM 或 PKCS12 文件所在的其他文件或目录。路径相对于您启动红帽构建的 Keycloak 的位置，因此推荐使用绝对路径。如果指定了目录，则会对信任存储文件进行递归扫描。

包含了所有适用的证书后，信任存储将通过 javax.net.ssl 属性用作系统默认信任存储，并作为红帽构建的 Keycloak 内部使用的默认值。

例如：

```
bin/kc.[sh|bat] start --truststore-paths=/opt/truststore/myTrustStore.pfx,/opt/other-truststore/myOtherTrustStore.pem
```

仍可直接设置您自己的 javax.net.ssl truststore 系统属性，但建议使用 -truststore-path。

12.2. 主机名验证策略

您可以使用 tls-hostname-verifier 属性重新定义 TLS 连接验证主机名的方式。

- DEFAULT （默认）允许子域名称中的通配符（如以前foo.com）匹配具有相同级别（如 a.foo.com，但不是 a.b.foo.com）的名称 - 基于 https://publicsuffix.org/list/的公共后缀的规则和排除
- ANY 表示主机名没有被验证 - 此模式不应在生产环境中使用。
- WILDCARD （已弃用）允许子域名称中的通配符（如foo.com）匹配任何内容，包括多个级别（如 a.b.foo.com）。改为使用 DEFAULT。
- STRICT （已弃用）允许子域名称中的通配符（例如，3.foo.com）匹配具有相同级别（如 a.foo.com，而不是 a.b.foo.com）的名称，有些有限排除。改为使用 DEFAULT。

请注意，此设置不适用于需要严格的主机名检查的 LDAP 安全连接。

12.3. 相关选项

	value
tls-hostname-verifier 用于传出 HTTPS 和 SMTP 请求的 TLS 主机名验证策略。 ANY 不应在生产环境中使用。 CLI: **--tls-hostname-verifier** Env: **KC_TLS_HOSTNAME_VERIFIER** STRICT 和 WILDCARD 已被弃用，改为使用 DEFAULT。**已弃用的值：STRICT,WILDCARD**	**ANY**,**WILDCARD** （已弃用）、**STRICT** （已弃用）、**DEFAULT** （默认）
truststore-paths pkcs12 列表(p12、pfx 或 pkcs12 文件扩展)、PEM 文件或目录，其中包含要用作系统信任存储的这些文件。 CLI: **--truststore-paths** Env: **KC_TRUSTSTORE_PATHS**	

第 13 章 为 MTLS 配置可信证书

配置 Mutual TLS，以验证连接到红帽构建的 Keycloak 的客户端。

为了正确验证客户端证书并启用某些身份验证方法，如双向 TLS 或 mTLS，您可以使用服务器应信任的所有证书（和证书链）设置信任存储。有很多功能依赖此信任存储来使用 Mutual TLS 和 X.509 身份验证等证书正确验证客户端。

13.1. 启用 MTLS

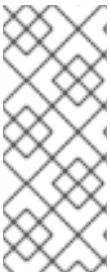
默认禁用使用 mTLS 进行身份验证。当红帽构建的 Keycloak 是服务器时，要启用 mTLS 证书处理，需要验证来自红帽构建的 Keycloak 端点的请求的证书，请将适当的证书放在信任存储中，并使用以下命令启用 mTLS：

```
bin/kc.[sh|bat] start --https-client-auth=<none|request|required>
```

使用 所需的值 设置红帽构建的 Keycloak 始终要求证书，并在请求中没有提供证书时失败。通过将值设置为 请求，红帽构建的 Keycloak 也接受没有证书的请求，并只验证证书的正确性（如果存在）。

警告

mTLS 配置和信任存储由所有 Realms 共享。无法为不同的 Realms 配置不同的信任存储。



注意

管理接口属性继承自主 HTTP 服务器，包括 mTLS 设置。这意味着，当设置 mTLS 时，也会为管理界面启用它。要覆盖行为，请使用 https-management-client-auth 属性。

13.2. 为 MTLS 使用专用信任存储

默认情况下，红帽构建的 Keycloak 使用 System Truststore 来验证证书。详情请参阅 配置可信证

书。

如果需要为 mTLS 使用专用信任存储，您可以通过运行以下命令来配置此信任存储的位置：

```
bin/kc.[sh|bat] start --https-trust-store-file=/path/to/file --https-trust-store-password=<value>
```

可识别的信任存储文件扩展：

- .p12、.pkcs12 和 .pfx 用于 pkcs12 文件
- .jks，以及 jks 文件的 .truststore
- .CA、.crt、和 .pem 文件

如果您的信任存储没有与其文件类型匹配的扩展名，您还需要设置 https-key-store-type 选项。

13.3. 其他资源

13.3.1. 将 mTLS 用于传出 HTTP 请求

请注意，这是红帽构建的 Keycloak 充当服务器的 mTLS 用例的基本证书配置。当红帽构建的 Keycloak 充当客户端时，例如红帽构建的 Keycloak 会尝试从经过 mTLS 保护的代理身份提供程序的令牌端点获取令牌，您需要设置 HttpClient，以便在密钥存储中为传出请求提供正确的证书。要在这些情况下配置 mTLS，请参阅配置传出的 HTTP 请求。

13.3.2. 配置 X.509 身份验证

有关如何配置 X.509 身份验证的更多信息，请参阅 X.509 客户端证书用户身份验证 部分。

13.4. 相关选项

	value
https-client-auth ▮ 将服务器配置为 require/request 客户端身份验证。 CLI: **--https-client-auth** Env: **KC_HTTPS_CLIENT_AUTH**	**none**（默认），请求，**required**
https-trust-store-file 保存要信任的证书的证书信息的信任存储。 CLI: **--https-trust-store-file** Env: **KC_HTTPS_TRUST_STORE_FILE**	
https-trust-store-password 信任存储文件的密码。 CLI: **--https-trust-store-password** Env: **KC_HTTPS_TRUST_STORE_PASSWORD**	
https-trust-store-type 信任存储文件的类型。 如果未指定，则根据文件扩展名自动检测到类型。如果将 **fips-mode** 设为 **strict**，且没有设置值，则默认为 **BCFKS**。 CLI: **--https-trust-store-type** Env: **KC_HTTPS_TRUST_STORE_TYPE**	
https-management-client-auth ▮ 将管理接口配置为 require/request 客户端身份验证。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-client-auth** Env: **KC_HTTPS_MANAGEMENT_CLIENT_AUTH**	**none**（默认），请求，**required**

第 14 章 启用和禁用功能

配置红帽构建的 Keycloak 以使用可选功能。

Red Hat build of Keycloak 包括了一些功能，包括一些禁用的功能，如技术预览和已弃用的功能。其他功能是默认启用的，但如果它们不适用于您使用 Keycloak 的 Red Hat build，则可以禁用它们。

14.1. 启用功能

一些支持的功能以及所有预览功能都默认禁用。要启用功能，请输入以下命令：

```
bin/kc.[sh|bat] build --features="<name>[,<name>]"
```

例如，要启用 docker 和 token-exchange，请输入以下命令：

```
bin/kc.[sh|bat] build --features="docker,token-exchange"
```

要启用所有预览功能，请输入以下命令：

```
bin/kc.[sh|bat] build --features="preview"
```

启用的功能可能版本化或未指定版本。如果您使用版本化功能名称，如 feature:v1，只要在运行时仍然存在，将启用确切的功能版本。如果您改为使用未指定版本的名称，如功能，选择特定的支持功能版本可能会根据以下优先级从发行版本改为发行版本：

1. 最高默认支持版本
2. 最高非默认支持版本
3. 最高弃用的版本
4. 最高预览版本

5. 最高实验性版本

14.2. 禁用功能

要禁用默认启用的功能，请输入以下命令：

```
bin/kc.[sh|bat] build --features-disabled="<name>[,<name>]"
```

例如，要禁用 模拟，请输入以下命令：

```
bin/kc.[sh|bat] build --features-disabled="impersonation"
```

不允许在 features-disabled 列表中和 features 列表中都有一个功能。

当禁用该功能时，该功能都会被禁用。

14.3. 支持的功能

以下列表包含默认启用的功能，如果不需要，可以禁用。

account-api

帐户管理 REST API

account-v3

帐户控制台版本 3

admin-api

Admin API

admin-fine-grained-authz-v2

精细的 Admin 权限版本 2

admin-v2

新的管理控制台

授权

授权服务

ciba

OpenID Connect Client Initiated Backchannel Authentication (CIBA)

client-policies

客户端配置策略

device-flow

OAuth 2.0 设备授权

hostname-v2

主机名选项 V2

模拟

管理员模拟用户的能力

Kerberos

Kerberos

login-v2

新的登录主题

opentelemetry

OpenTelemetry 追踪

机构

领域中的机构支持

par

OAuth 2.0 推送授权请求(PAR)

persistent-user-sessions

重启和升级后持久的在线用户会话

rolling-updates-v1

滚动更新

step-up-authentication

步骤身份验证

token-exchange-standard-v2

标准令牌交换版本 2

user-event-metrics

根据用户事件收集指标

web-authn

W3C Web 身份验证(WebAuthn)

14.3.1. 默认禁用

以下列表包含默认禁用的支持功能，并在需要时可以启用。

docker

Docker Registry 协议

fips

FIPS 140-2 模式

多站点

多站点支持

14.4. 技术预览功能

技术预览功能默认为禁用，不建议在生产环境中使用。这些功能可能会在以后的版本中更改或删除。

admin-fine-grained-authz

细粒度的 Admin 权限

client-secret-rotation

客户端 Secret 轮转

dpop

应用程序层上的 OAuth 2.0 演示概念验证-Possession

passkeys

Passkeys

recovery-codes

恢复代码

脚本

使用 JavaScript 编写自定义验证器

token-exchange

令牌交换服务

update-email

更新电子邮件操作

14.5. 弃用的功能

以下列表包含将在以后的版本中删除的已弃用的功能。这些功能默认为禁用。

login-v1

传统登录主题

14.6. 相关选项

	value

	value
功能 启用一个或多个功能的集合。 CLI: -- **features** Env: **KC_FEATURES**	**account-api[:v1]**, **account[:v3]**, **admin-api[:v1]**, **admin-fine-grained-authz[:v1,v2]**, **admin[:v2]**, **authorization[:v1]**, **cache-embedded-remote-store[:v1]**, **ciba[:v1]**, **client-policies[:v** 1] **, client-secret-rotation[:v** 1] **, client-types[:v** 1] **, clusterless[:v** 1] **, declarative-ui[:v** 1] **, device-flow[:v** 1] **, docker[:v** 1] **, dpop[:v** 1] **, dynamic-scopes[:v** 1] **, fips[:** v1] **, hostname[:** v2] **, impersonation[:** v1] **, ipa-tuura-federation[:** v1] **, kerberos[:** v1] **, login[:v2,** v1] **, multi-site[:** v1] **, oid4vc-vci[:** v1] , **Opentelemetry[:** **v1], organization[:v 1] , par[:v 1] , passkeys[:v 1] , persistent-user-sessions[:v 1],preview , quick-theme[:v 1] , recovery-codes[:v 1] , rolling-updates[:v 1] , scripts[:v 1] , step-up-authentication** [: **v1] , token-exchange-standard** [: **v2] , token-exchange** [:v **1] , ephemeral-users** [: **v1] , update-email** [: **v1] , user-event-metrics** [: **v1] , web-authn**[:v1]

	value
features-disabled swig 禁用一个或多个功能的集合。 CLI: **--features-disabled** Env: **KC_FEATURES_DISABLED**	**帐户**,**account-api**,**admin**,**admin-api**,**admin-fine-grained-authz**,**authorization**,**cache-embedded-remote-store**,**ciba**,**client-policies**,**client-secret-rotation**,**client-types**,**clusterless**, clusterless**-ui**,**device-flow**,**docker**,**d** pop**, dynamic-** scopes**,** fips**,** impersonation**, ipa-tuura-** federation**,** kerberos**,** login ,**oid4vc-v** ci , **opentelemetry, organization, par, pass keys, persistent-user-sessions, preview , preview, quick - theme, recovery - codes, rolling - updates, scripts, step-up - authentication, token -exchange, token-exchange -standard, transient -users, update -email, user-event -metrics, web-authn**

第 15 章 配置供应商

为红帽构建的 Keycloak 配置供应商。

服务器在构建时具有可扩展性，因为它提供了多个服务提供商接口或 SPI，各自负责为服务器提供特定功能。在本章中，您将了解关于 SPI 配置及其相应提供程序的核心概念。

阅读本章后，您应能够使用概念和步骤来说明安装、卸载、启用、禁用和配置任何提供程序，包括您实施的那些用来扩展服务器功能以更好地满足您的要求。

15.1. 配置选项格式

可以使用特定的配置格式配置提供程序。格式由以下组成：

```
spi-<spi-id>-<provider-id>-<property>=<value>
```

& lt;spi-id > 是您要配置的 SPI 的名称。

& lt;provider-id > 是您要配置的供应商的 id。这是将 id 设置为对应的供应商工厂实现。

<property > 是您要为给定供应商设置的属性的实际名称。

所有这些名称（用于 spi、provider 和 属性）都应为小写，如果名称位于 camel-case 中，如 myKeycloakProvider，它应包含大写字母前的短划线(-)，如下所示： my-keycloak-provider。

将 HttpClientSpi SPI 作为示例，SPI 的名称是 connectionsHttpClient，另一个可用的提供程序实施名为 default。要设置 connectionPoolSize 属性，您可以使用配置选项，如下所示：

```
spi-connections-http-client-default-connection-pool-size=10
```

15.2. 设置供应商配置选项

启动服务器时会提供提供程序配置选项。有关配置 红帽构建的 Keycloak 中的选项，请查看所有支持配置源和格式。例如，通过命令行选项：

为 connections-http-client SPI 的默认 供应商设置 connection-pool-size

```
bin/kc.[sh|bat] start --spi-connections-http-client-default-connection-pool-size=10
```

15.3. 为 SPI 配置单个供应商

根据 SPI，多个提供商实施可以共存，但一次只能使用其中一个。对于这些 SPI，特定提供程序是激活并在运行时使用的主要实施。

要将供应商配置为单一供应商，您应该运行 build 命令，如下所示：

将 mycustomprovider 提供程序标记为 email-template SPI 的单个提供程序

```
bin/kc.[sh|bat] build --spi-email-template-provider=mycustomprovider
```

15.4. 为 SPI 配置默认供应商

根据 SPI，多个提供商实施可以共存，默认情况下会使用一个。对于这些 SPI，特定提供程序是要选择的默认实施，除非请求特定的提供程序。

以下逻辑用于决定默认供应商：

1. 明确配置的默认供应商

2. 具有最高顺序的供应商（带有顺序为 0 的供应商将被忽略）

3. id 设置为 default的供应商

要将供应商配置为默认供应商，您应该运行 build 命令，如下所示：

将 mycustomhash 供应商标记为 密码哈希 SPI 的默认 提供程序

```
bin/kc.[sh|bat] build --spi-password-hashing-provider-default=mycustomprovider
```

15.5. 启用和禁用供应商

要启用或禁用供应商，您应该运行 build 命令，如下所示：

启用供应商

```
bin/kc.[sh|bat] build --spi-email-template-mycustomprovider-enabled=true
```

要禁用提供程序，请使用相同的命令并将 enabled 属性设置为 false。

15.6. 安装和卸载供应商

自定义提供程序应打包在 Java 存档(JAR)文件中，并复制到发行版 的供应商 目录中。之后，您必须运行 build 命令，才能使用 JAR 文件中的实现来更新服务器的供应商注册表。

需要使用这个步骤来优化服务器运行时，以便提前知道所有供应商，而不是仅在启动服务器或运行时发现。

要卸载提供程序，您应该从 providers 目录中删除 JAR 文件，然后再次运行 build 命令。

15.7. 使用第三方依赖项

在实施提供程序时，您可能需要使用从服务器分发中不可用的一些第三方依赖项。

在这种情况下，您应该将任何其他依赖项复制到 供应商 目录中，并运行 build 命令。一旦这样做，服务器将在运行时为依赖于它们的任何供应商提供这些额外的依赖项。

15.8. 参考

- 配置红帽构建的 Keycloak
- 服务器开发人员文档

第 16 章 配置日志记录

为红帽构建的 Keycloak 配置日志记录。

红帽构建的 Keycloak 使用 JBoss Logging 框架。以下是可用日志处理程序的高级概述，它带有通用父日志处理程序 root ：

- 控制台
- file
- syslog

16.1. 日志记录配置

日志记录在红帽构建的 Keycloak 中按类别完成。您可以为根日志级别或更具体的类别（如 org.hibernate 或 org.keycloak ）配置日志记录。也可以为每个特定的日志处理程序定制日志级别。

本章论述了如何配置日志记录。

16.1.1. 日志级别

下表定义可用的日志级别。

级别	描述
FATAL	具有完整无法服务的任何请求的关键故障。
ERROR	导致无法处理请求的重大错误或问题。
WARN	可能需要立即更正的非关键错误或问题。
INFO	红帽构建的 Keycloak 生命周期事件或重要信息。低频率。

级别	描述
DEBUG	用于调试目的（如数据库日志）的更多详细信息。频率更高。
TRACE	最详细的调试信息。非常高的频率。
ALL	所有日志消息的特殊级别。
OFF	完全关闭日志记录的特殊级别（不推荐）。

16.1.2. 配置 root 日志级别

当没有针对更具体的类别日志记录器的日志级别配置时，会使用封闭类别。如果没有封闭类别，则使用根日志记录器级别。

要设置 root 日志级别，请输入以下命令：

```
bin/kc.[sh|bat] start --log-level=<root-level>
```

对这个命令使用以下指南：

- 对于 *<root-level>*，请提供上表中定义的级别。
- 日志级别不区分大小写。例如，您可以使用 DEBUG 或 debug。
- 如果您要意外设置日志级别两次，则列表中最后一次出现的内容将变为日志级别。例如，如果您包含 syntax -log-level="info,...,DEBUG,..."，则根日志记录器将为 DEBUG。

16.1.3. 配置特定于类别的日志级别

您可以为红帽构建的 Keycloak 中的特定区域设置不同的日志级别。使用这个命令提供您需要不同日志级别的类别列表：

```
bin/kc.[sh|bat] start --log-level="<root-level>,<org.category1>:<org.category1-level>"
```

应用到类别的配置也适用于其子类别，除非您包含更为具体的匹配子类别。

Example

```
bin/kc.[sh|bat] start --log-level="INFO,org.hibernate:debug,org.hibernate.hql.internal.ast:info"
```

这个示例设置以下日志级别：

- 所有日志记录器的根日志级别都设置为 INFO。
- 通常，hibernate 日志级别被设置为 debug。
- 要从创建详细日志输出中保留 SQL 抽象语法树，特定的子类别 org.hibernate.hql.internal.ast 设置为 info。因此，SQL 抽象语法树会被忽略，而不是出现在 debug 级别中。

16.1.3.1. 将级别配置为单个选项

在配置特定于类别的日志级别时，您还可以将日志级别设置为单独的 log-level-<category> 选项，而不是使用该级别的 log-level 选项。这在您要为所选类别设置日志级别时很有用，而不覆盖之前设置的 日志级别 选项。

Example

如果您以以下方式启动服务器：

```
bin/kc.[sh|bat] start --log-level="INFO,org.hibernate:debug"
```

然后，您可以设置一个环境变量 KC_LOG_LEVEL_ORG_KEYCLOAK=trace，以更改 org.keycloak 类别的日志级别。

log-level -<category > 选项优先于日志级别。这可让您覆盖 log-level 选项中设置的内容。例如，如果您为上述 CLI 示例设置了 KC_LOG_LEVEL_ORG_HIBERNATE=trace，则 org.hibernate 类别将使用 trace 级别而不是 debug。

请记住，在使用环境变量时，类别名称必须采用大写，并且点必须替换为下划线。使用其他配置源时，必须指定类别名称 "as is"，例如：

```
bin/kc.[sh|bat] start --log-level="INFO,org.hibernate:debug" --log-level-org.keycloak=trace
```

16.2. 启用日志处理程序

要启用日志处理程序，请输入以下命令：

```
bin/kc.[sh|bat] start --log="<handler1>,<handler2>"
```

可用的处理程序有：

- 控制台
- file
- syslog

以下提到的更具体的处理程序配置仅在将处理程序添加到此逗号分隔列表中时生效。

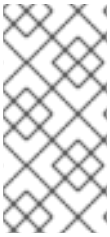
16.2.1. 为每个处理器指定日志级别

log-level 属性指定所选类别的全局根日志级别和级别。但是，需要更精细的日志级别来满足现代应用的要求。

要为特定处理程序设置日志级别，格式为 log-<handler>-level （其中 & lt;handler > 是可用的日志处理程序）。

这意味着日志级别设置的属性如下：

- log-console-level - Console log handler
- log-file-level - File log handler
- log-syslog-level - Syslog 日志处理程序



注意

log-<handler>-level 属性仅在启用了特定日志处理程序时才可用。以下日志处理程序设置中的更多信息。

只有 第 16.1.1 节 “日志级别” 部分指定的日志级别才会被接受，且必须为小写。尚不支持为日志处理程序指定特定的类别。

16.2.1.1. 常规原则

需要了解，为每个特定处理程序设置日志级别 不会覆盖 log-level 属性中指定的根级别。日志处理程序遵循 root 日志级别，这代表了整个日志记录系统的最大详细程度。这意味着单个日志处理程序可以配置为小于根日志记录器，但不能配置更多。

具体来说，当为处理程序定义了任意日志级别时，并不表示输出中会显示带有日志级别的日志记录。在这种情况下，还必须评估 root 日志级别。日志级别为 root 日志级别 提供限制，日志处理程序的默认日志级别 都是 无限制的。

16.2.1.2. 例子

示例：对文件处理程序进行 debug，但 console 处理程序的信息：

```
bin/kc.[sh|bat] start --log=console,file --log-level=debug --log-console-level=info
```

root 日志级别设置为 debug，因此每个日志处理程序都会继承值 - 因此，文件日志处理程序会继承文件日志处理程序。要 在控制台中隐藏调试 记录，我们需要将 console 处理程序的最小（最低严重）级别设置为 info。

示例： 警告 所有处理程序，但对文件处理程序进行 debug ：

```
bin/kc.[sh|bat] start --log=console,file,syslog --log-level=debug --log-console-level=warn --log-syslog-level=warn
```

root 级别必须设置为最详细的必要级别（本例中为debug ），并且必须相应地修改其他日志处理程序。

示例：所有处理程序的信息，但对 Syslog 处理程序进行 debug+org.keycloak.events:trace ：

```
bin/kc.[sh|bat] start --log=console,file,syslog --log-level=debug,org.keycloak.events:trace, --log-syslog-level=trace --log-console-level=info --log-file-level=info
```

要查看 org.keycloak.events:trace，必须为 Syslog 处理程序设置 trace 级别。

16.2.2. 为日志处理程序使用不同的 JSON 格式

每个日志处理程序都提供了以 JSON 格式具有结构化日志输出的功能。它可以被属性启用，格式为 log-<handler>-output=json （其中 & lt;handler& gt; 是一个日志处理程序）。

如果您需要生成的 JSON 的不同格式，您可以使用以下 JSON 输出格式：

- 默认 （默认）
- ECS

ecs 值指的是 ECS (Elastic Common Schema)。

ECS 是一种开源、社区驱动的规范，用于定义用于 Elastic 解决方案的通用字段集合。ECS 规格与 OpenTelemetry Semantic Conventions 合并，目的是创建由 OpenTelemetry 维护的一个标准。

要更改 JSON 输出格式，引入了格式 log-<handler>-json-format （其中 &lt ;handler& gt; 是一个日志处理器）的属性：

- log-console-json-format - Console log handler
- log-file-json-format - File log handler
- log-syslog-json-format - Syslog 日志处理程序

16.2.2.1. Example

如果要以 ECS (Elastic Common Schema)格式具有 JSON 日志，您可以输入以下命令：

```
bin/kc.[sh|bat] start --log-console-output=json --log-console-json-format=ecs
```

日志消息示例


```
{"@timestamp":"2025-02-03T14:53:22.539484211+01:00","event.sequence":9608,"log.logger":"io.quarkus","log.level":"INFO","message":"Keycloak 999.0.0-SNAPSHOT on JVM (powered by Quarkus 3.17.8) started in 4.615s. Listening on: http://0.0.0.0:8080","process.thread.name":"main","process.thread.id":1,"mdc":
```

{},"ndc":"","host.hostname":"host-name","process.name":"/usr/lib/jvm/jdk-21.0.3+9/bin/java","process.pid":77561,"data_stream.type":"logs","ecs.version":"1.12.2","service.environment":"prod","service.name":"Keycloak","service.version":"999.0.0-SNAPSHOT"}

16.3. 控制台日志处理程序

控制台日志处理程序默认启用，为控制台提供无结构日志消息。

16.3.1. 配置控制台日志格式

红帽 Keycloak 的构建使用基于模式的日志格式器，默认生成人类可读的文本日志。

这些行的日志记录格式模板可以在根级别上应用。默认格式模板为：

- %d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n

格式字符串支持下表中的符号：

符号	Summary	描述
%%	%	呈现简单的%字符。
%c	类别	呈现日志类别名称。
%d{xxx}	Date	使用 **java.text.SimpleDateFormat** 定义的给定日期格式 string.String 语法呈现日期
%e	例外	呈现抛出异常。
%H	主机名	呈现简单的主机名。
%H	限定主机名	呈现完全限定主机名，该主机名可能与简单的主机名相同，具体取决于操作系统配置。

符号	Summary	描述
%i	进程 ID	呈现当前进程 PID。
%m	完整消息	如果抛出，则呈现日志消息和异常。
%n	换行符	呈现特定于平台的行分隔符字符串。
%N	进程名称	呈现当前进程的名称。
%p	级别	呈现消息的日志级别。
%R	相对时间	从应用程序日志开始后，呈现时间（毫秒）。
%s	简单消息	仅在没有异常追踪的情况下呈现日志消息。
%t	线程名称	呈现线程名称。
%t{id}	线程 ID	呈现线程 ID。
%z{<zone name>}	timezone	将日志输出的时区设置为 <zone name>。
%L	行号	呈现日志消息的行号。

16.3.2. 设置日志记录格式

要为日志记录行设置日志记录格式，请执行以下步骤：

1. 使用上表构建所需的格式模板。
2. 输入以下命令：

```
bin/kc.[sh|bat] start --log-console-format="'<format>'"
```

请注意，在使用 CLI 中调用包含特殊 shell 字符（如 ）的命令时，您需要转义字符。 因此，请考虑在配置文件中设置它。

示例：缩写完全限定类别名称

```
bin/kc.[sh|bat] start --log-console-format="'%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c{3.}] (%t) %s%e%n'"
```

本例通过将类别名称缩写为三个字符，方法是在模板中设置 [%c{3.}]，而不是默认的 [%c]。

16.3.3. 配置 JSON 或普通控制台日志记录

默认情况下，控制台日志处理程序会将普通非结构化数据记录到控制台。要使用结构化 JSON 日志输出，请输入以下命令：

```
bin/kc.[sh|bat] start --log-console-output=json
```

日志消息示例


```
{"timestamp":"2025-02-03T14:52:20.290353085+01:00","sequence":9605,"loggerClassName":"org.jboss.logging.Logger","loggerName":"io.quarkus","level":"INFO","message":"Keycloak 999.0.0-SNAPSHOT on JVM (powered by Quarkus 3.17.8) started in 4.440s. Listening on: http://0.0.0.0:8080","threadName":"main","threadId":1,"mdc":{},"ndc":"","hostName":"host-name","processName":"/usr/lib/jvm/jdk-21.0.3+9/bin/java","processId":76944}
```


使用 JSON 输出时，颜色被禁用，不应用由 --log-console-format 设置设置。

要使用非结构的日志，请输入以下命令：

```
bin/kc.[sh|bat] start --log-console-output=default
```

日志消息示例

```
2025-02-03 14:53:56,653 INFO  [io.quarkus] (main) Keycloak 999.0.0-SNAPSHOT on JVM (powered by Quarkus 3.17.8) started in 4.795s. Listening on: http://0.0.0.0:8080
```

16.3.4. colors

默认禁用无结构日志的带颜色的控制台日志输出。颜色可能会提高可读性，但在向外部日志聚合系统发送日志时可能会造成问题。要启用或禁用颜色编码的控制台日志输出，请输入以下命令：

```
bin/kc.[sh|bat] start --log-console-color=<false|true>
```

16.3.5. 配置控制台日志级别

控制台日志处理器的日志级别可由 **--log-console-level** 属性指定，如下所示：

```
bin/kc.[sh|bat] start --log-console-level=warn
```

如需更多信息，请参阅上面的 第 16.2.1 节 “为每个处理器指定日志级别” 部分。

16.4. 文件日志记录

作为控制台日志记录的替代选择，您可以使用非结构化日志记录到文件。

16.4.1. 启用文件日志记录

默认禁用登录到文件。要启用它，请输入以下命令：

```
bin/kc.[sh|bat] start --log="console,file"
```

在红帽构建的 Keycloak 安装的 data/log 目录中创建名为 keycloak.log 的日志文件。

16.4.2. 配置日志文件的位置和名称

要更改日志文件的创建和文件名，请执行以下步骤：

1. 创建可写入目录来存储日志文件。

   如果该目录不可写入，红帽 Keycloak 的构建将正确启动，但它会发生错误，且不会创建日志文件。

2. 输入这个命令：

```
bin/kc.[sh|bat] start --log="console,file" --log-file=<path-to>/<your-file.log>
```

16.4.3. 配置文件处理程序格式

要为文件日志处理器配置不同的日志记录格式，请输入以下命令：

```
bin/kc.[sh|bat] start --log-file-format="<pattern>"
```

有关可用模式配置的更多信息和表，请参阅 第 16.3.1 节 “配置控制台日志格式”。

16.4.4. 配置文件日志级别

文件日志处理器的日志级别可以通过 -log-file-level 属性指定，如下所示：

```
bin/kc.[sh|bat] start --log-file-level=warn
```

如需更多信息，请参阅上面的 第 16.2.1 节 “为每个处理器指定日志级别” 部分。

16.5. 使用 SYSLOG 进行集中式日志记录

Red Hat build of Keycloak 提供了将日志发送到远程 Syslog 服务器的功能。它使用 RFC 5424 中定义的协议。

16.5.1. 启用 Syslog 处理程序

要使用 Syslog 启用日志记录，请将其添加到激活的日志处理程序列表中，如下所示：

```
bin/kc.[sh|bat] start --log="console,syslog"
```

16.5.2. 配置 Syslog 应用程序名称

要设置不同的应用程序名称，请添加 the --log-syslog-app-name 选项，如下所示：

```
bin/kc.[sh|bat] start --log="console,syslog" --log-syslog-app-name=kc-p-itadmins
```

如果没有设置，应用程序名称默认为 keycloak。

16.5.3. 配置 Syslog 端点

要配置集中式日志记录系统的端点(*host:port*)，请输入以下命令并将值替换为您的特定值：

```
bin/kc.[sh|bat] start --log="console,syslog" --log-syslog-endpoint=myhost:12345
```

启用 Syslog 处理程序后，主机将使用 localhost 作为主机值。默认端口为 514。

16.5.4. 配置 Syslog 日志级别

Syslog 日志处理器的日志级别可由 -log-syslog-level 属性指定，如下所示：

```
bin/kc.[sh|bat] start --log-syslog-level=warn
```

如需更多信息，请参阅上面的 第 16.2.1 节 “为每个处理器指定日志级别” 部分。

16.5.5. 配置 Syslog 协议

syslog 使用 TCP 作为通信的默认协议。要使用 UDP 而不是 TCP，请添加 --log-syslog-protocol 选项，如下所示：

```
bin/kc.[sh|bat] start --log="console,syslog" --log-syslog-protocol=udp
```

可用的协议有： tpc、udp 和 ssl-tcp。

16.5.6. 配置 Syslog 日志格式

要为日志记录行设置日志记录格式，请执行以下步骤：

1. 使用上表构建所需的格式模板。
2. 输入以下命令：

```
bin/kc.[sh|bat] start --log-syslog-format="'<format>'"
```

请注意，在使用 CLI 中调用包含特殊 shell 字符（如 ）的命令时，您需要转义字符。 因此，请考虑在配置文件中设置它。

示例：缩写完全限定类别名称

```
bin/kc.[sh|bat] start --log-syslog-format="'%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c{3.}] (%t) %s%e%n'"
```

本例通过将类别名称缩写为三个字符，方法是在模板中设置 [%c{3.}]，而不是默认的 [%c]。

16.5.7. 配置 Syslog 类型

syslog 根据特定的 RFC 规格使用不同的消息格式。要使用不同的消息格式更改 Syslog 类型，请使用 --log-syslog-type 选项，如下所示：

```
bin/kc.[sh|bat] start --log-syslog-type=rfc3164
```

the -log-syslog-type 选项的可能值有：

- RFC5424 （默认）
- rfc3164

首选 Syslog 类型是 RFC 5424，它过时的 RFC 3164，称为 BSD Syslog 协议。

16.5.8. 配置 Syslog 最大消息长度

要设置允许发送的消息的最大长度（以字节为单位），请使用 --log-syslog-max-length 选项，如下所示：

```
bin/kc.[sh|bat] start --log-syslog-max-length=1536
```

可使用合适的后缀（如 1k 或 1K ）以内存大小格式指定长度。长度包括标头和消息。

如果没有显式设置长度，则会根据 --log-syslog-type 选项设置默认值，如下所示：

- 2048B - for RFC 5424
- 1024B - for RFC 3164

16.5.9. 配置 Syslog 结构化输出

默认情况下，Syslog 日志处理程序将普通非结构化数据发送到 Syslog 服务器。要使用结构化 JSON 日志输出，请输入以下命令：

```
bin/kc.[sh|bat] start --log-syslog-output=json
```

日志消息示例

```
2024-04-05T12:32:20.616+02:00 host keycloak 2788276 io.quarkus - {"timestamp":"2024-04-05T12:32:20.616208533+02:00","sequence":9948,"loggerClassName":"org.jboss.logging.Logger","loggerName":"io.quarkus","level":"INFO","message":"Profile prod activated.
","threadName":"main","threadId":1,"mdc":
{},"ndc":"","hostName":"host","processName":"QuarkusEntryPoint","processId":2788276}
```

使用 JSON 输出时，颜色会被禁用，且不会应用 by --log-syslog-format 设置。

要使用非结构的日志，请输入以下命令：

```
bin/kc.[sh|bat] start --log-syslog-output=default
```

日志消息示例

```
2024-04-05T12:31:38.473+02:00 host keycloak 2787568 io.quarkus - 2024-04-05 12:31:38,473 INFO  [io.quarkus] (main) Profile prod activated.
```

正如您所见，时间戳存在两次，因此您可以通过 --log-syslog-format 属性进行相应的处理。

16.6. 相关选项

	value
log 在以逗号分隔的列表中启用一个或多个日志处理程序。 CLI: -- **log** Env: **KC_LOG**	**控制台**,**文件**,**syslog**

	value
log-level 根类别的日志级别或以逗号分隔的类别列表及其级别。 对于 root 类别，您不需要指定一个类别。 CLI: **--log-level** Env: **KC_LOG_LEVEL**	**[info]**（默认）

16.6.1. 控制台（Console）

	value
log-console-color 登录到控制台时启用或禁用颜色。 CLI: **--log-console-color** Env: **KC_LOG_CONSOLE_COLOR** 仅在激活 Console 日志处理程序时可用	**true**,**false**（默认）
log-console-format 无结构控制台日志条目的格式。 如果格式有空格，请使用 "<format>" 转义值。 CLI: **--log-console-format** Env: **KC_LOG_CONSOLE_FORMAT** 仅在激活 Console 日志处理程序时可用	**%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c](%t)%s%e%n** (default)
log-console-include-trace 在控制台日志中包括追踪信息。 如果指定了 **log-console-format** 选项，这个选项无效。 CLI: **--log-console-include-trace** Env: **KC_LOG_CONSOLE_INCLUDE_TRACE** 仅在激活 Console 日志处理程序和 Tracing 时可用	**true**（默认），**false**

	value
log-console-json-format 设置生成的 JSON 格式。 CLI: **--log-console-json-format** Env: **KC_LOG_CONSOLE_JSON_FORMAT** 仅在激活 Console 日志处理程序并且输出设置为 'json' 时才可用	**默认** （默认），**ecs**
log-console-level 设置控制台处理程序的日志级别。 它指定输出中显示的日志的最详细日志级别。它遵循 **log-level** 选项指定的级别，它代表整个日志记录系统的最大详细程度。如需更多信息，请参阅 Logging 指南。 CLI: **--log-console-level** Env: **KC_LOG_CONSOLE_LEVEL** 仅在激活 Console 日志处理程序时可用	**off**,**fatal**,**error**,**warn**,**info**,**debug**,**trace**,**all** （默认）
log-console-output 将日志输出设置为 JSON 或默认(plain)非结构化日志记录。 CLI: **--log-console-output** Env: **KC_LOG_CONSOLE_OUTPUT** 仅在激活 Console 日志处理程序时可用	**默认** （默认），**json**

16.6.2. File

	value
log-file 设置日志文件路径和文件名。 CLI: **--log-file** Env: **KC_LOG_FILE** 仅在文件日志处理程序激活时才可用	**data/log/keycloak.log** （默认）

	value
log-file-format 设置特定于文件日志条目的格式。 CLI: **--log-file-format** Env: **KC_LOG_FILE_FORMAT** 仅在文件日志处理程序激活时才可用	**%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n** (default)
log-file-include-trace 在文件日志中包含追踪信息。 如果指定了 **log-file-format** 选项，这个选项无效。 CLI: **--log-file-include-trace** Env: **KC_LOG_FILE_INCLUDE_TRACE** 仅在激活文件日志处理程序和跟踪时才可用	**true** （默认），**false**
log-file-json-format 设置生成的 JSON 格式。 CLI: **--log-file-json-format** Env: **KC_LOG_FILE_JSON_FORMAT** 仅在文件日志处理程序激活并且输出设置为 'json' 时才可用	**默认** （默认），**ecs**
log-file-level 设置文件处理程序的日志级别。 它指定输出中显示的日志的最详细日志级别。它遵循 **log-level** 选项指定的级别，它代表整个日志记录系统的最大详细程度。如需更多信息，请参阅 Logging 指南。 CLI: **--log-file-level** Env: **KC_LOG_FILE_LEVEL** 仅在文件日志处理程序激活时才可用	**off**,**fatal**,**error**,**warn**,**info**,**debug**,**trace**,**all** （默认）
log-file-output 将日志输出设置为 JSON 或默认(plain)非结构化日志记录。 CLI: **--log-file-output** Env: **KC_LOG_FILE_OUTPUT** 仅在文件日志处理程序激活时才可用	**默认** （默认），**json**

16.6.3. Syslog

	value
log-syslog-app-name 设置使用 RFC5424 格式格式化消息时使用的应用程序名称。 CLI: **--log-syslog-app-name** Env: **KC_LOG_SYSLOG_APP_NAME** 仅在 Syslog 激活时才可用	**Keycloak**（默认）
log-syslog-endpoint 设置 Syslog 服务器的 IP 地址和端口。 CLI: **--log-syslog-endpoint** Env: **KC_LOG_SYSLOG_ENDPOINT** 仅在 Syslog 激活时才可用	**localhost:514**（默认）
log-syslog-format 设置特定于 Syslog 条目的格式。 CLI: **--log-syslog-format** Env: **KC_LOG_SYSLOG_FORMAT** 仅在 Syslog 激活时才可用	**%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n** (default)
log-syslog-include-trace 在 Syslog 中包含追踪信息。 如果指定了 **log-syslog-format** 选项，这个选项无效。 CLI: **--log-syslog-include-trace** Env: **KC_LOG_SYSLOG_INCLUDE_TRACE** 仅在激活 Syslog 处理程序和 Tracing 时可用	**true**（默认），**false**
log-syslog-json-format 设置生成的 JSON 格式。 CLI: **--log-syslog-json-format** Env: **KC_LOG_SYSLOG_JSON_FORMAT** 仅在 Syslog 激活且输出设置为 'json' 时才可用	**默认**（默认），**ecs**

	value
log-syslog-level 设置 Syslog 处理程序的日志级别。 它指定输出中显示的日志的最详细日志级别。它遵循 **log-level** 选项指定的级别，它代表整个日志记录系统的最大详细程度。如需更多信息，请参阅 Logging 指南。 CLI: **--log-syslog-level** Env: **KC_LOG_SYSLOG_LEVEL** 仅在 Syslog 激活时才可用	**off**,**fatal**,**error**,**warn**,**info**,**debug**,**trace**,**all** （默认）
log-syslog-max-length 设置允许发送的消息的最大长度（以字节为单位）。 长度包括标头和消息。如果没有设置，当 **log-syslog-type** 为 rfc5424 （默认）和 1024 时，当 **log-syslog-type** 为 rfc3164 时，默认值为 2048。 CLI: **--log-syslog-max-length** Env: **KC_LOG_SYSLOG_MAX_LENGTH** 仅在 Syslog 激活时才可用	
log-syslog-output 将 Syslog 输出设置为 JSON 或默认(plain)非结构化日志记录。 CLI: **--log-syslog-output** Env: **KC_LOG_SYSLOG_OUTPUT** 仅在 Syslog 激活时才可用	**默认** （默认），**json**
log-syslog-protocol 设置用于连接 Syslog 服务器的协议。 CLI: **--log-syslog-protocol** Env: **KC_LOG_SYSLOG_PROTOCOL** 仅在 Syslog 激活时才可用	**TCP** （默认）、**udp**、**ssl-tcp**
log-syslog-type 设置用于格式化发送消息的 Syslog 类型。 CLI: **--log-syslog-type** Env: **KC_LOG_SYSLOG_TYPE** 仅在 Syslog 激活时才可用	**RFC5424** （默认），**rfc3164**

第 17 章 FIPS 140-2 支持

为 FIPS 合规性配置红帽构建的 Keycloak 服务器。

Federal Information Processing Standard Publication 140-2,(FIPS 140-2)，是用于批准加密模块的美国政府计算机安全标准。红帽构建的 Keycloak 支持以 FIPS 140-2 兼容模式运行。在这种情况下，红帽构建的 Keycloak 将只使用 FIPS 批准的加密算法来实现其功能。

要在 FIPS 140-2 中运行，红帽构建的 Keycloak 应该运行在启用了 FIPS 140-2 的系统中。这个要求通常会假设 RHEL 或 Fedora 在安装过程中启用了 FIPS。详情请查看 RHEL 文档。当系统处于 FIPS 模式时，它会确保底层 OpenJDK 处于 FIPS 模式，并且只使用 启用了 FIPS 的安全供应商。

要检查系统是否处于 FIPS 模式，您可以从命令行使用以下命令检查它：

```
fips-mode-setup --check
```

如果系统不处于 FIPS 模式，您可以使用以下命令启用它，但建议系统处于 FIPS 模式，因为安装而不是随后启用它：

```
fips-mode-setup --enable
```

17.1. BOUNCYCASTLE 库

红帽 Keycloak 内部构建对许多加密工具使用 BouncyCastle 库。请注意，红帽构建的 Keycloak 附带的 BouncyCastle 库的默认版本不兼容 FIPS，但 BouncyCastle 也提供其库的 FIPS 验证版本。红帽构建的 Keycloak 不提供 FIPS 验证的 BouncyCastle 库，因为红帽构建的 Keycloak 无法提供官方支持。因此，为了以 FIPS 兼容模式运行，您需要下载 BouncyCastle-FIPS 位，并将它们添加到红帽构建的 Keycloak 分发中。当红帽构建的 Keycloak 以 fips 模式执行时，它将使用 BCFIPS 位而不是默认的 BouncyCastle 位，这将实现 FIPS 合规性。

17.1.1. BouncyCastle FIPS 位

BouncyCastle FIPS 可以从 BouncyCastle 官方页面 下载。然后，您可以将它们添加到发行版本的目录 KEYCLOAK_HOME/providers 中。确保使用正确的版本与 BouncyCastle Red Hat build of Keycloak 依赖项兼容。所需的 BCFIPS 位有：

- bc-fips 版本 2.0.0.
- bctls-fips 版本 2.0.19。
- bcpkix-fips 版本 2.0.7。
- bcutil-fips 版本 2.0.3。

17.2. 生成密钥存储

您可以创建 pkcs12 或 bcfks 密钥存储，以用于红帽构建的 Keycloak 服务器 SSL。

17.2.1. PKCS12 keystore

p12 （或 pkcs12）密钥存储（以及/或信任存储）在 BCFIPS 非批准模式下可以正常工作。

PKCS12 密钥存储可以在 RHEL 9 上使用 OpenJDK 21 Java 生成。例如，以下命令可用于生成密钥存储：

```
keytool -genkeypair -sigalg SHA512withRSA -keyalg RSA -storepass passwordpassword \
  -keystore $KEYCLOAK_HOME/conf/server.keystore \
  -alias localhost \
  -dname CN=localhost -keypass passwordpassword
```

FIPS 模式中的 pkcs12 密钥存储 不管理 secret (symmetric)密钥。这个限制由 BCFIPS 供应商实施，该提供程序不允许在 pkcs12 密钥存储类型内这种类型的密钥。

当系统处于 FIPS 模式时，默认的 java.security 文件将更改为使用启用了 FIPS 的安全供应商，因此不需要额外的配置。另外，在 PKCS12 密钥存储中，您只需使用 keytool 命令存储 PBE （基于密码的加密）密钥，这使其成为使用红帽构建的 Keycloak KeyStore Vault 和/或将配置属性存储在 KeyStore Config Source 中的配置属性的理想选择。如需了解更多详细信息，请参阅配置红帽构建的 Keycloak 和使用密码库。

17.2.2. BCFKS 密钥存储

BCFKS 密钥存储生成需要使用 BouncyCastle FIPS 库和自定义安全文件。

您可以创建一个帮助程序文件，如 /tmp/kc.keystore-create.java.security。文件的内容只需要具有以下属性：

```
securerandom.strongAlgorithms=PKCS11:SunPKCS11-NSS-FIPS
```

接下来，输入以下命令来生成密钥存储：

```
keytool -keystore $KEYCLOAK_HOME/conf/server.keystore \
  -storetype bcfks \
  -providername BCFIPS \
  -providerclass org.bouncycastle.jcajce.provider.BouncyCastleFipsProvider \
  -provider org.bouncycastle.jcajce.provider.BouncyCastleFipsProvider \
  -providerpath $KEYCLOAK_HOME/providers/bc-fips-*.jar \
  -alias localhost \
  -genkeypair -sigalg SHA512withRSA -keyalg RSA -storepass passwordpassword \
  -dname CN=localhost -keypass passwordpassword \
  -J-Djava.security.properties=/tmp/kc.keystore-create.java.security
```

警告

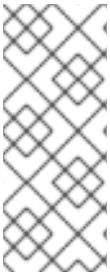
使用自签名证书仅用于演示目的，因此当您迁移到生产环境时，将这些证书替换为正确的证书。

当您使用 bcfks 类型的 keystore/truststore 进行任何其他操作时，需要类似的选项。

17.3. 运行 服务器。

要在非批准模式下使用 BCFIPS 运行服务器，请输入以下命令

```
bin/kc.[sh|bat] start --features=fips --hostname=localhost --https-key-store-
password=passwordpassword --log-
level=INFO,org.keycloak.common.crypto:TRACE,org.keycloak.crypto:TRACE
```



注意

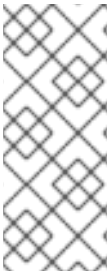
在非批准模式中，默认密钥存储类型（以及默认信任存储类型）是 PKCS12。因此，如果您生成了上述 BCFKS 密钥存储，也需要使用 command --https-key-store-type=bcfks。如果您希望使用 truststore，也可能需要一个类似的命令。

注意

如果一切按预期工作，您可以禁用生产中的日志。

17.4. STRICT 模式

有 fips-mode 选项，当启用 fips 功能时，该选项会自动设置为 non-strict。这意味着在 "non-approved mode" 中运行 BCFIPS。更安全的替代方案是 use-- features=fips --fips-mode=strict，在这种情况下，BouncyCastle FIPS 将使用 "approved 模式"。使用该选项会对加密和安全算法造成更严格的安全要求。



注意

在严格模式中，默认密钥存储类型（以及默认信任存储类型）是 BCFKS。如果要使用不同的密钥存储类型，则需要使用带有适当类型的 option -https-key-store-type。如果您希望使用 truststore，也可能需要一个类似的命令。

启动服务器时，您可以在启动命令中包含 TRACE 级别。例如：

```
--log-level=INFO,org.keycloak.common.crypto.CryptoIntegration:TRACE
```

通过使用 TRACE 级别，您可以检查启动日志是否包含 KC 供应商，其中包含有关 Approved Mode 的备注，如下所示：

```
KC(BCFIPS version 2.0 Approved Mode, FIPS-JVM: enabled) version 1.0 - class
org.keycloak.crypto.fips.KeycloakFipsSecurityProvider,
```

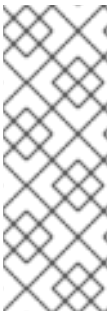
17.4.1. 严格模式中的加密限制

- 如上一节中所述，严格的模式可能无法用于 pkcs12 密钥存储。如前文所述，需要使用另一个密钥存储（如 bcfks）。在使用 strict 模式时，红帽构建的 Keycloak 不支持 jks 和 pkcs12 密

钥存储。有些示例是在管理控制台中导入或生成 OIDC 或 SAML 客户端的密钥存储，或用于 realm 密钥中的 java-keystore 供应商。

- 用户密码必须为 14 个字符或更长时间。默认情况下，Red Hat build of Keycloak 使用基于 PBKDF2 的密码编码。BCFIPS 批准模式需要至少 112 位（有效的 14 个字符）使用 PBKDF2 算法。如果要允许较短的密码，请将供应商 pbkdf2-sha512 of SPI password-hashing 的属性 max-padding-length 设置为 14，以便在验证此算法创建的哈希时提供额外的 padding。此设置也与之前存储的密码向后兼容。例如，如果用户的数据库位于非FIPS 环境中，且您有较短的密码，并且您想要在批准模式中使用 BCFIPS 在红帽构建的 Keycloak 中验证它们，则密码应该可以正常工作。因此，您可以在启动服务器时使用如下选项：

```
--spi-password-hashing-pbkdf2-sha512-max-padding-length=14
```



注意

使用以上选项不会中断 FIPS 合规性。但请注意，较长的密码都是很好的做法。例如，现代浏览器自动生成的密码符合此要求，因为它们超过 14 个字符。如果要省略 max-padding-length 的 选项，您可以将密码策略设置为您的域，使其至少有 14 个字符的密码。

注意

当您从早于 24 的红帽构建的 Keycloak 迁移时，或者如果您明确设置了密码策略来覆盖默认的哈希算法，您的某些用户可能会使用像 pbkdf2-sha256 这样的旧算法。在这种情况下，请考虑添加 --spi-password-hashing-pbkdf2-sha256-max-padding-length=14 选项，以确保其密码使用旧的 pbkdf2-sha256 可以登录，因为其密码可能比 14 个字符要短。

- RSA 密钥为 1024 位不能工作（最小是2048）。这适用于由红帽构建 Keycloak 域本身使用的键（来自管理控制台中的 Keys 选项卡的Realm 密钥），以及客户端密钥和 IDP 密钥

- HMAC SHA-XXX 密钥必须至少为 112 位（或 14 个字符）。例如，如果您使用带有客户端身份验证的 OIDC 客户端 Signed Jwt with Client Secret （或者在 OIDC 表示法中的 client-secret-jwt ），则您的客户端 secret 应该至少 14 个字符。请注意，为了获得良好的安全性，建议使用红帽构建的 Keycloak 服务器生成的客户端 secret，这始终可以满足这个要求。

- bc-fips 版本 1.0.2.4 处理 PKCS 1.5 RSA 加密的过渡周期结束。因此，默认情况下，在严格模式下不允许带有算法 RSA1_5 的 JSON Web 加密(JWE) (BC 提供系统属性 -

Dorg.bouncycastle.rsa.allow_pkcs15_enc=true 作为现在的向后兼容性选项)。RSA-OAEP 和 RSA-OAEP-256 仍如以前提供。

17.5. 其他限制

要使 SAML 正常工作，请确保您的安全供应商提供了 XMLDSig 安全供应商。要使 Kerberos 正常工作，请确保 SunJGSS 安全供应商可用。在 OpenJDK 21 中启用了 FIPS 的 RHEL 9，默认情况下，在 java.security 中可能会启用 XMLDSig 安全供应商，与最新的 OpenJDK 17 应用相同。但是，对于旧的 OpenJDK 17，它可能无法默认启用，这意味着 SAML 有效地可以正常工作。

要使 SAML 正常工作，您可以手动将供应商添加到 JAVA_HOME/conf/security/java.security.security.security 中。例如，在 FIPS 安全供应商中没有行时添加如下行：

```
fips.provider.7=XMLDSig
```

添加此安全提供程序应该可以正常工作。实际上，它兼容 FIPS，在 OpenJDK 21 及更新版本的 OpenJDK 17 中已添加。详情包括在 bugzilla 中。

注意

建议查看 JAVA_HOME/conf/security/java.security，并在这里检查所有配置的供应商，并确保数字匹配。换句话说，fips.provider.7 假设已经有 6 个供应商配置了前缀，如 fips.provider.N。

如果您不希望在 java 本身内编辑 java.security 文件，您可以创建一个自定义的 java 安全文件（例如 kc.java.security），仅添加以上用于添加 XMLDSig 提供程序的单个属性。然后，使用附加此属性文件启动红帽构建的 Keycloak 服务器：

```
-Djava.security.properties=/location/to/your/file/kc.java.security
```

对于 Kerberos/SPNEGO，安全供应商 SunJGSS 尚未完全兼容 FIPS。因此，如果您希望符合 FIPS，则不建议将其添加到安全供应商列表中。当在 FIPS 平台以及安全供应商不可用时，红帽构建的 Keycloak 中默认禁用 KERBEROS 功能。详情包括在 bugzilla 中。

算法 EdDSA 无法用于 FIPS 模式。虽然当前的 BCFIPS 供应商支持 Ed25519 和 Ed448 curves，但生成的密钥不实施标准的 JDK 接口来管理它们(EdECKey、

EdECPublicKey、EdECPublicKey、EdECPrivateKey、...)和红帽构建的 Keycloak 不能将它们用于签名。

17.6. 在 FIPS 主机上运行 CLI

如果要运行客户端注册 CLI (kcreg.sh|bat 脚本)或 Admin CLI (kcadm.sh|bat 脚本)，CLI 还必须使用 BouncyCastle FIPS 依赖项，而不是普通的 BouncyCastle 依赖项。要达到此目的，您可以将 jar 复制到 CLI 库文件夹，这足够了。当 CLI 工具检测到对应的 BCFIPS jar 是否存在（请参阅上面的版本），则 CLI 工具将自动使用 BCFIPS 依赖项而不是纯 BCFIPS jar。例如，在运行 CLI 前使用以下命令：

```
cp $KEYCLOAK_HOME/providers/bc-fips-*.jar $KEYCLOAK_HOME/bin/client/lib/
cp $KEYCLOAK_HOME/providers/bctls-fips-*.jar $KEYCLOAK_HOME/bin/client/lib/
cp $KEYCLOAK_HOME/providers/bcutil-fips-*.jar $KEYCLOAK_HOME/bin/client/lib/
```

注意

当试图将 BCFKS truststore/keystore 与 CLI 搭配使用时，您可能会遇到问题，因为此信任存储不是默认的 java 密钥存储类型。在 java 安全属性中，可以将它指定为默认值。例如，在使用 kcadm|kcreg 客户端执行任何操作前，在基于 unix 的系统中运行这个命令：

```
echo "keystore.type=bcfks
fips.keystore.type=bcfks" > /tmp/kcadm.java.security
export KC_OPTS="-Djava.security.properties=/tmp/kcadm.java.security"
```

17.7. 红帽在容器中以 FIPS 模式构建 KEYCLOAK 服务器

当您希望红帽以 FIPS 模式构建 Keycloak 时，您的"主机"也必须使用 FIPS 模式。然后，容器将从父主机"inherit" FIPS 模式。详情请查看 RHEL 文档中的这个部分。https://access.redhat.com/documentation/zh-cn/red_hat_enterprise_linux/9/html/security_hardening/using-the-system-wide-cryptographic-policies_security-hardening#enabling-fips-mode-in-a-container_using-the-system-wide-cryptographic-policies

当从 FIPS 模式下从主机执行时，Red Hat build of Keycloak 容器镜像将自动处于 fips 模式。但是，请确保红帽构建的 Keycloak 容器也使用 BCFIPS jars （而不是 BC jar）和启动时正确的选项。

有关此问题，最好构建自己的容器镜像，如 在容器中运行红帽构建的 Keycloak 所述，并调整 它以使用 BCFIPS 等。

例如在当前目录中，您可以创建子目录 文件 并添加：

- 如上所述， BC FIPS jar 文件
- 自定义密钥存储文件 - 例如 keycloak-fips.keystore.bcfks
- 安全文件 kc.java.security，为 SAML 添加了供应商(OpenJDK 21 或更新版本 OpenJDK 17 不需要)

然后，在当前目录中创建 Containerfile，如下所示：

Containerfile ：

```
FROM registry.redhat.io/rhbk/keycloak-rhel9:26.2 as builder

ADD files /tmp/files/

WORKDIR /opt/keycloak
RUN cp /tmp/files/*.jar /opt/keycloak/providers/
RUN cp /tmp/files/keycloak-fips.keystore.* /opt/keycloak/conf/server.keystore
RUN cp /tmp/files/kc.java.security /opt/keycloak/conf/

RUN /opt/keycloak/bin/kc.sh build --features=fips --fips-mode=strict

FROM registry.redhat.io/rhbk/keycloak-rhel9:26.2
COPY --from=builder /opt/keycloak/ /opt/keycloak/

ENTRYPOINT ["/opt/keycloak/bin/kc.sh"]
```

然后，将 FIPS 作为优化的 Docker 镜像构建并启动它，如 容器中运行红帽构建的 Keycloak 所述。这些步骤要求您使用启动镜像时的参数。

17.8. 从非 FIPS 环境迁移

如果您之前在非fips 环境中使用了 Red Hat build of Keycloak，则可以将其迁移到 FIPS 环境中，包括其数据。但是，在上一节中所述，存在限制和注意事项，即：

- 从红帽构建的 Keycloak 25 开始，密码散列的默认算法是 argon2。但是，FIPS 140-2 不支持这个算法。这意味着，如果您的用户使用 argon2 对密码进行哈希处理，该用户将无法在切换到 FIPS 环境后登录。如果您计划迁移到 FIPS 环境，请考虑从头（创建任何用户之前）设置域的密码策略，并覆盖 example 为 pbkdf2-sha512 （符合 FIPS）的默认算法。此策略有助于把系统迁移到 FIPS 环境变得平稳。否则，如果您的用户已在 argon2 密码上，只需要求用户在迁移到 FIPS 环境后重置密码。例如，要求用户使用"Forget 密码"或向所有用户发送电子邮件。

- 确保所有红帽构建的 Keycloak 功能依赖于密钥存储，它只使用受支持的密钥存储类型。这与是否使用严格的模式或非限制模式而有所不同。

- Kerberos 身份验证可能无法正常工作。如果您的身份验证流使用 Kerberos 验证器，则当迁移到 FIPS 环境时，这个验证器会自动切换到 DISABLED。建议您从您的域中删除任何 Kerberos 用户存储供应商，并在切换到 FIPS 环境前禁用 LDAP 供应商中的 Kerberos 相关功能。

除了前面的要求外，请务必在切换到 FIPS 严格模式前再次检查它：

- 确保所有红帽构建的 Keycloak 功能依赖于密钥（如 realm 或客户端密钥）至少使用 2048 位的 RSA 密钥

- 确保依赖使用客户端 Secret 签名 JWT 的客户端至少使用 14 个字符长 secret （最好生成 secret）

- 如前所述，密码长度限制。如果您的用户有较短的密码，请确保启动将 max padding length 设置为 14 of PBKDF2 供应商的服务器。如果您希望避免这个选项，您可以要求所有用户在新环境中进行第一个身份验证期间重置其密码(例如，Forgot 密码 链接)。

17.9. 在非FIPS 系统中构建 KEYCLOAK FIPS 模式

在启用了 FIPS 的 RHEL 8 系统和 ubi8 镜像中，支持并测试 Red Hat build of Keycloak。RHEL 9（和 ubi9 镜像）也支持它。在非 RHEL 兼容平台或非FIPS 启用的平台上运行，FIPS 合规性无法严格保证，且无法被正式支持。

如果您仍然仅限于在这样的系统上运行红帽构建的 Keycloak，您至少可以更新 java.security 文件中配置的安全供应商。这个版本没有 FIPS 合规性，但至少设置会更接近它。它可以通过提供自定义安全文件且仅提供覆盖的安全供应商列表来完成，如前面所述。有关推荐供应商列表，请参阅 OpenJDK 21 文档。

您可以在启动时检查红帽构建的 Keycloak 服务器日志，以查看是否使用了正确的安全供应商。应该为与 Keycloak 软件包相关的红帽构建启用 TRACE 日志记录，如之前的 Keycloak start 命令中所述。

第 18 章 配置管理界面

为端点配置红帽构建的 Keycloak 管理界面，如指标和健康检查。

管理接口允许通过与主 HTTP 服务器不同的 HTTP 服务器访问管理端点。它提供隐藏外部世界中的 /metrics 或 /health 等端点，从而强化安全性。Kubernetes 环境中可能会看到最重要的优势，因为特定的管理端口可能无法公开。

18.1. 管理界面配置

当其上公开内容时，将开启管理界面。启用指标和健康时，在默认的管理端口 9000 上公开管理端点，如 /metrics 和 /health。管理界面提供了一组选项，并且是完全可配置的。

注意

如果没有显式设置管理接口属性，则它们的值会自动从默认的 HTTP 服务器继承。

18.1.1. port

要更改管理界面的端口，您可以使用红帽 build of Keycloak 选项 http-management-port。

18.1.2. 相对路径

您可以更改管理界面的相对路径，因为管理端点的前缀路径可能会有所不同。您可以通过红帽构建的 Keycloak 选项 http-management-relative-path 实现它。

例如，如果您设置了 CLI 选项 --http-management-relative-path=/management，则指标和健康端点将在 /management/ metrics 和 /management /health 路径上访问。

在指定相对路径时，用户会 自动重定向到 托管 Red Hat build of Keycloak 的路径。这意味着，当相对路径设置为 /management，并且用户访问 localhost:9000/ 时，该页面会被重定向到 localhost:9000/management。

注意

如果没有显式设置它的值，则使用 http-relative-path 属性中的值。例如，如果您设置了 CLI 选项 --http-relative-path=/auth，这些端点可以在 /auth/metrics 和 /auth/health 路径上访问。

18.1.3. TLS 支持

当为默认红帽构建的 Keycloak 服务器设置了 TLS 时，管理界面也可以通过 HTTPS 访问。管理接口只能在 HTTP 或 HTTPS 上运行，不能同时在主服务器上运行。

为管理 HTTP 服务器设置不同的 TLS 参数提供了前缀 https-management expectations 的特定红帽构建的 Keycloak 管理界面选项。它们的功能与主 HTTP 服务器的对应部分类似，请参阅 配置 TLS。如果没有显式设置这些选项，TLS 参数将继承到默认的 HTTP 服务器。

18.1.4. 禁用管理界面

当其上没有公开时，管理界面会自动关闭。目前，无论如何，都只会在管理界面上公开健康检查和指标。如果要在管理界面中禁用公开它们，请将红帽 Keycloak 属性的 build of legacy-observability-interface 设置为 true。

警告

出于安全原因，不建议在默认服务器上公开健康和指标端点，您应该始终使用管理界面。请注意，legacy-observability-interface 选项已弃用，并将在以后的发行版本中删除。它只允许您为迁移留出更多时间。

18.2. 相关选项

	value
http-management-port 管理接口的端口。 仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--http-management-port** Env: **KC_HTTP_MANAGEMENT_PORT**	**9000** （默认）
http-management-relative-path 设置相对于 / 的路径，以便从管理界面提供资源。 该路径必须以 / 开头。如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--http-management-relative-path** Env: **KC_HTTP_MANAGEMENT_RELATIVE_PATH**	/ （默认）
https-management-certificate-file 管理服务器的 PEM 格式的服务器证书或证书链的文件路径。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-certificate-file** Env: **KC_HTTPS_MANAGEMENT_CERTIFICATE_FILE**	
https-management-certificate-key-file 管理服务器的 PEM 格式的私钥的文件路径。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-certificate-key-file** Env: **KC_HTTPS_MANAGEMENT_CERTIFICATE_KEY_FILE**	
https-management-certificates-reload-period 重新载入管理服务器的 https-management ldapmodify 选项引用的密钥存储、信任存储和证书文件的时间间隔。 可以是 java.time.Duration 值、整数数或整数，后跟 [ms, h, m, s, d] 之一。必须大于 30 秒。使用 -1 禁用。如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-certificates-reload-period** Env: **KC_HTTPS_MANAGEMENT_CERTIFICATES_RELOAD_PERIOD**	**1h** （默认）

	value
https-management-client-auth ▮ 将管理接口配置为 require/request 客户端身份验证。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-client-auth** Env: **KC_HTTPS_MANAGEMENT_CLIENT_AUTH**	**none**（默认），请求，**required**
https-management-key-store-file 保存证书信息的密钥存储，而不是为管理服务器指定单独的文件。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-key-store-file** Env: **KC_HTTPS_MANAGEMENT_KEY_STORE_FILE**	
https-management-key-store-password 管理服务器的密钥存储文件的密码。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-key-store-password** Env: **KC_HTTPS_MANAGEMENT_KEY_STORE_PASSWORD**	**密码**（默认）
legacy-observability-interface wagon 如果应在主 HTTP 服务器上公开 metrics/health 端点（不推荐）。 如果设置为 true，则禁用管理界面。 CLI: **--legacy-observability-interface** Env: **KC_LEGACY_OBSERVABILITY_INTERFACE** **已弃用。**	**true**,**false**（默认）

第 19 章 导入和导出域

导入和导出域作为 JSON 文件。

在本章中，您将了解使用 JSON 文件导入和导出域的不同方法。

注意

导出并导入到单个文件可生成大量文件，因此如果您的数据库包含 500 多个用户，请将导出到一个目录，而不是单个文件。使用目录执行更好，因为目录提供程序对每个"页面"（用户的一个文件）使用单独的事务。每个文件和每个事务的用户的默认计数为 fifty。增大到更大的数字会导致执行时间指数级增长。

所有红帽构建的 Keycloak 节点都需要在使用 kc.[sh|bat] import | export 命令前停止。这样可确保生成的操作不会与并发请求产生一致性问题。它还确保从与服务器实例相同的计算机上运行导入或导出命令不会产生端口或其他冲突。

19.1. 为数据库连接参数提供选项

当使用 导出和导入命令时，红帽构建的 Keycloak 需要了解如何连接到存储域、客户端、用户和其他实体信息的数据库。如 配置红帽构建的 Keycloak 所述，这些信息可作为命令行参数、环境变量或配置文件提供。对每个命令使用- help 命令行选项查看可用选项。

有些配置选项是构建时间配置选项。默认情况下，如果红帽构建的 Keycloak 检测到构建时间参数的变化，则会自动为 导出和导入 命令重新构建。

如果您使用 build 命令构建红帽构建的 Keycloak 版本，如 配置红帽 Keycloak 构建 中所述，请使用命令行选项 优化功能，让红帽构建 Keycloak 跳过一个更快的启动时间。执行此操作时，请从命令行删除构建时间选项，仅保留运行时选项。

注意

如果您没有使用优化功能，则 导入 或导出 命令会隐式为您创建或更新优化镜像 - 如果您从与服务器实例相同的机器运行该命令，则这可能会影响服务器下次启动。

19.2. 将 REALM 导出到目录

要导出域，您可以使用 export 命令。在调用此命令时，不能启动 Red Hat build of Keycloak 服务器实例。

```
bin/kc.[sh|bat] export --help
```

要将域导出到一个目录，您可以使用--dir &lt ;dir> 选项。

```
bin/kc.[sh|bat] export --dir <dir>
```

将域导出到目录时，服务器将为每个要导出的域创建单独的文件。

19.2.1. 配置如何导出用户

您还可以通过设置-users < strategy> 选项来配置如何导出用户。这个选项的值有：

different_files

用户根据每个文件设置的最大用户数，导出到不同的 json 文件。这是默认值。

skip

跳过导出用户。

realm_file

用户将导出到与 realm 设置相同的文件。对于名为 "foo" 的域，这代表 "foo-realm.json" 带有 realm data 和 users。

same_file

所有用户都导出到一个显式文件。因此，您将为域获取两个 json 文件，一个用于 realm 数据，另一个用于用户。

如果您要使用 different_files 策略导出用户，您可以通过设置 --users-per-file 选项来设置您想要的每个文件的用户数量。默认值为 50。

```
bin/kc.[sh|bat] export --dir <dir> --users different_files --users-per-file 100
```

19.3. 将 REALM 导出到文件

要将域导出到文件，您可以使用--file &lt ;file> 选项。

```
bin/kc.[sh|bat] export --file <file>
```

将域导出到文件时，服务器将使用相同的文件来存储要导出的所有域的配置。

19.4. 导出特定域

如果您没有指定要导出的特定域，则会导出所有域。要导出单个域，您可以使用 the-- realm 选项，如下所示：

```
bin/kc.[sh|bat] export [--dir|--file] <path> --realm my-realm
```

19.5. 导入文件命名约定

当您使用域特定的文件名惯例时，还必须在启动时从目录导入或导入。要导入的域文件必须命名为 <realm name>-realm.json。与域关联的常规和联邦用户文件必须命名为 <realm-name>-users-<file number>.json 和 <realm-name>-federated-users-<file number>.json。如果无法使用此惯例，会导致错误或用户文件没有被导入。

19.6. 从目录导入 REALM

要导入域，您可以使用 import 命令。在调用此命令时，不能启动 Red Hat build of Keycloak 服务器实例。

```
bin/kc.[sh|bat] import --help
```

将域导出到一个目录后，您可以使用-- dir <dir& gt; 选项将 realm 导入到服务器，如下所示：

```
bin/kc.[sh|bat] import --dir <dir>
```

在使用 import 命令导入域时，如果应该跳过现有域，或者应该使用新配置覆盖现有域。为此，您可以

设置-- override 选项，如下所示：

```
bin/kc.[sh|bat] import --dir <dir> --override false
```

默认情况下，-- override 选项设置为 true，以便域始终被新配置覆盖。

19.7. 从文件导入 REALM

要导入之前在单个文件中导出的域，您可以使用-file < file> 选项，如下所示：

```
bin/kc.[sh|bat] import --file <file>
```

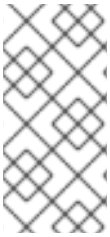
19.8. 在 REALM 配置文件中使用环境变量

您可以使用占位符从任何域配置的环境变量中解析值。

使用占位符的域配置

```
{
  "realm": "${MY_REALM_NAME}",
  "enabled": true,
  ...
}
```

在上例中，将值设为 MY_REALM_NAME 环境变量将用于设置 realm 属性。



注意

目前，可以引用哪些环境变量没有限制。当环境变量用于提供敏感信息时，请谨慎确保占位符引用不会不恰当公开敏感环境变量值。

19.9. 在启动过程中导入 REALM

您还可以使用-- import-realm 选项启动服务器时导入域。

```
bin/kc.[sh|bat] start --import-realm
```

设置 --import-realm 选项时，服务器将尝试从 data/import 目录中导入任何域配置文件。只有使用 .json 扩展名的常规文件才能从此目录读取，子目录将被忽略。

注意

对于红帽构建的 Keycloak 容器，导入目录为 /opt/keycloak/data/import

如果服务器中已存在域，则会跳过导入操作。此行为的主要原因是避免在服务器重新启动之间重新创建域并可能丢失状态。

要重新创建域，您应该在启动服务器前显式运行 import 命令。

19.10. 使用管理控制台导入和导出

您还可以使用 Admin Console 导入和导出域。此功能与前面部分中描述的其他 CLI 选项不同，因为管理控制台仅提供 *部分* 导出域的能力。在这种情况下，可以导出当前域设置以及某些资源，如客户端、角色和组。*无法* 使用此方法导出该域的用户。

注意

使用 Admin Console 导出时，realm 和所选资源始终导出到名为 realm-export.json 的文件。另外，密码和客户端 secret 等所有敏感值都使用 * 符号进行屏蔽。

要使用管理控制台导出域，请执行以下步骤：

1. 选择一个 realm。

2. 单击菜单中的 **Realm settings**。

3. 指向 realm 设置屏幕右上角的 **Action** 菜单，然后选择 **Partial export**。

   资源列表与域配置一起显示。

4. 选择您要导出的资源。

5. 单击 **Export**。

注意

从管理控制台导出的域不适合服务器之间的备份或数据传输。只有 CLI 导出适用于服务器之间的备份或数据传输。

警告

如果 realm 包含很多组、角色和客户端，则操作可能会导致服务器对用户请求没有响应。请谨慎使用此功能，特别是在生产环境中。

类似地，您可以导入之前导出的域。执行这些步骤：

1. 单击菜单中的 **Realm settings**。

2. 指向 realm 设置屏幕右上角的 **Action** 菜单，然后选择 **Partial import**。

   此时会出现一个提示，您可以在其中选择要导入的文件。根据这个文件，您会看到可以导入的资源以及 realm 设置。

3. 点 Import。

如果导入的资源已存在，您还可以控制红帽构建的 Keycloak 应该做什么。这些选项存在：

导入失败

中止导入。

skip

在不中止进程的情况下跳过重复资源

覆盖

将现有的资源替换为正在导入的资源。

注意

Admin Console 部分导入也可以导入由 CLI export 命令创建的文件。换句话说，可以使用管理控制台导入 CLI 创建的完整导出。如果文件包含用户，则这些用户还将可用于导入到当前域中。

第 20 章 使用密码库

在红帽构建的 Keycloak 中配置和使用 vault。

红帽 Keycloak 的构建提供了 Vault SPI 的两个开箱即用的实现：基于纯文本文件的 vault 和基于 Java KeyStore 的密码库。

基于文件的 vault 实施对 Kubernetes/OpenShift secret 特别有用。您可以将 Kubernetes secret 挂载到红帽构建的 Keycloak Container 中，数据字段将在带有无格式文件结构的挂载文件夹中可用。

基于 Java KeyStore 的 vault 实现对于将 secret 存储在裸机安装中非常有用。您可以使用 KeyStore vault，该密码库使用密码加密。

20.1. 可用的集成

存储在密码库中的 secret 可以在管理控制台的以下位置使用：

- 获取 SMTP 邮件服务器密码
- 在使用基于 LDAP 的用户联邦时获取 LDAP 绑定凭证
- 在集成外部身份提供程序时获取 OIDC 身份提供程序客户端 Secret

20.2. 启用密码库

要启用基于文件的库，首先需要使用以下构建选项构建红帽 Keycloak 的构建：

```
bin/kc.[sh|bat] build --vault=file
```

对于基于 Java KeyStore 的，您需要指定以下构建选项：

```
bin/kc.[sh|bat] build --vault=keystore
```

20.3. 配置基于文件的密码库

20.3.1. 将基础目录设置为查找 secret

Kubernetes/OpenShift secret 基本上是挂载的文件。要配置应挂载这些文件的目录，请输入以下命令：

```
bin/kc.[sh|bat] start --vault-dir=/my/path
```

20.3.2. 特定于域的 secret 文件

Kubernetes/OpenShift Secret 在 Red Hat build of Keycloak 中基于每个域使用，这需要文件命名规则：

```
${vault.<realmname>_<secretname>}
```

20.4. 配置基于 JAVA KEYSTORE 的密码库

要使用基于 Java KeyStore 的密码库，您需要首先创建一个 KeyStore 文件。您可以使用以下命令完成此操作：

```
keytool -importpass -alias <realm-name>_<alias> -keystore keystore.p12 -storepass
keystorepassword
```

然后，输入您要存储在密码库中的值。请注意，-alias 参数的格式取决于使用的密钥解析器。默认密钥解析器为 REALM_UNDERSCORE_KEY。

默认情况下，这会导致在 SecretKeyEntry 中以通用 PBEKey （基于密码的加密）的形式存储值。

然后，您可以使用以下运行时选项启动红帽构建的 Keycloak：

```
bin/kc.[sh|bat] start  --vault-file=/path/to/keystore.p12 --vault-pass=<value> --vault-type=
<value>
```

请注意，--vault-type 参数是可选的，默认为 PKCS12。

然后，存储在密码库中的 secret 可以通过以下占位符访问域（假设使用 REALM_UNDERSCORE_KEY 密钥解析器）：${vault.realm-name_alias}。

20.5. 在 SECRET 名称中使用下划线

要正确处理 secret，您可以在 <secretname> 中加倍所有下划线。当使用 REALM_UNDERSCORE_KEY 键解析器时，<realmname> 中的下划线也会加倍，<secretname> 和 <realmname> 由一个下划线分开。

Example

- realm Name: sso_realm
- 所需名称： ldap_credential
- 生成的文件名：

```
sso__realm_ldap__credential
```

请注意 *sso* 和 *realm* 之间的双下划线，以及 *ldap* 和 *credential* 之间的双引号。

要了解有关关键解析器的更多信息，请参阅 服务器管理指南中的密钥解析器部分。

20.6. 示例：在管理门户中使用 LDAP 绑定凭证 SECRET

设置示例

- 名为 secrettest的域
- 绑定凭证所需的名称 ldapBc

- 生成的文件名： **secrettest_ldapBc**

Admin Console 中的使用

然后，您可以在配置 LDAP 用户联邦时，从 Admin 控制台使用此 secret，将 ${vault.ldapBc} 用作 Bind Credential 的值。

20.7. 相关选项

	value
Vault wagon 启用 vault 供应商。 CLI: -- **vault** Env: **KC_VAULT**	**文件**,**keystore**
vault-dir 如果设置，可以通过读取给定目录中文件的内容来获取 secret。 CLI: **--vault-dir** Env: **KC_VAULT_DIR**	
vault-file 密钥存储文件的路径。 CLI: **--vault-file** Env: **KC_VAULT_FILE**	
vault-pass vault 密钥存储的密码。 CLI: **--vault-pass** Env: **KC_VAULT_PASS**	
vault-type 指定密钥存储文件的类型。 CLI: **--vault-type** Env: **KC_VAULT_TYPE**	**PKCS12**（默认）

第 21 章 所有配置

查看红帽构建的 Keycloak 的构建选项和配置。

21.1. CACHE

	value
cache 定义高可用性的缓存机制。 默认情况下，在生产环境模式中，使用 **ispn** 缓存在多个服务器节点之间创建集群。默认情况下，在开发模式中，**本地缓存** 会禁用集群，用于开发和测试目的。 CLI: -- **cache** Env: **KC_CACHE**	**ISPN** （默认），**local**
cache-config-file 定义应从中加载缓存配置的文件。 配置文件相对于 **conf/** 目录。 CLI: **--cache-config-file** Env: **KC_CACHE_CONFIG_FILE**	
cache-embedded-authorization-max-count 授权缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-authorization-max-count** Env: **KC_CACHE_EMBEDDED_AUTHORIZATION_MAX_COUNT**	
cache-embedded-client-sessions-max-count 客户端Sessions 缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-client-sessions-max-count** Env: **KC_CACHE_EMBEDDED_CLIENT_SESSIONS_MAX_COUNT** 仅在配置了嵌入式 Infinispan 集群时可用	
cache-embedded-crl-max-count crl 缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-crl-max-count** Env: **KC_CACHE_EMBEDDED_CRL_MAX_COUNT**	

	value
cache-embedded-keys-max-count 密钥缓存可在内存中存储的最大条目数。 CLI: **--cache-embedded-keys-max-count** Env: **KC_CACHE_EMBEDDED_KEYS_MAX_COUNT**	
cache-embedded-mtls-enabled 加密 Keycloak 服务器之间的网络通信。 如果没有提供有关密钥存储和信任存储的额外参数，则会自动创建和轮转临时密钥对，这是标准设置的建议。 CLI: **--cache-embedded-mtls-enabled** Env: **KC_CACHE_EMBEDDED_MTLS_ENABLED** 仅在使用基于 TCP 的 cache-stack 时可用	**true** （默认），**false**
cache-embedded-mtls-key-store-file Keystore 文件路径。 Keystore 必须包含 TLS 协议使用的证书。默认情况下，它会在 conf/ 目录下查找 **cache-mtls-keystore.p12**。 CLI: **--cache-embedded-mtls-key-store-file** Env: **KC_CACHE_EMBEDDED_MTLS_KEY_STORE_FILE** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	
cache-embedded-mtls-key-store-password 用于访问密钥存储的密码。 CLI: **--cache-embedded-mtls-key-store-password** Env: **KC_CACHE_EMBEDDED_MTLS_KEY_STORE_PASSWORD** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	
cache-embedded-mtls-rotation-interval-days 自动 JGroups MTLS 证书轮转周期（以天为单位）。 CLI: **--cache-embedded-mtls-rotation-interval-days** Env: **KC_CACHE_EMBEDDED_MTLS_ROTATION_INTERVAL_DAYS** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	**30** （默认）

	value
cache-embedded-mtls-trust-store-file Truststore 文件路径。 它应包含可信证书或签发证书的证书颁发机构。默认情况下，它会在 conf/ 目录下查找 **cache-mtls-truststore.p12**。 CLI: **--cache-embedded-mtls-trust-store-file** Env: **KC_CACHE_EMBEDDED_MTLS_TRUST_STORE_FILE** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	
cache-embedded-mtls-trust-store-password 访问 Truststore 的密码。 CLI: **--cache-embedded-mtls-trust-store-password** Env: **KC_CACHE_EMBEDDED_MTLS_TRUST_STORE_PASSWORD** 仅在启用了属性 'cache-embedded-mtls-enabled' 时可用	
cache-embedded-offline-client-sessions-max-count offlineClientSessions 缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-offline-client-sessions-max-count** Env: **KC_CACHE_EMBEDDED_OFFLINE_CLIENT_SESSIONS_MAX_COUNT** 仅在配置了嵌入式 Infinispan 集群时可用	
cache-embedded-offline-sessions-max-count offlineSessions 缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-offline-sessions-max-count** Env: **KC_CACHE_EMBEDDED_OFFLINE_SESSIONS_MAX_COUNT** 仅在配置了嵌入式 Infinispan 集群时可用	
cache-embedded-realms-max-count 域缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-realms-max-count** Env: **KC_CACHE_EMBEDDED_REALMS_MAX_COUNT**	

	value
cache-embedded-sessions-max-count 会话缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-sessions-max-count** Env: **KC_CACHE_EMBEDDED_SESSIONS_MAX_COUNT** 仅在配置了嵌入式 Infinispan 集群时可用	
cache-embedded-users-max-count 用户缓存可以存储在内存中的最大条目数。 CLI: **--cache-embedded-users-max-count** Env: **KC_CACHE_EMBEDDED_USERS_MAX_COUNT**	
cache-metrics-histograms-enabled 为嵌入式缓存的指标启用直方图。 CLI: **--cache-metrics-histograms-enabled** Env: **KC_CACHE_METRICS_HISTOGRAMS_ENABLED** 仅在启用指标时可用	**true**,**false** （默认）
cache-remote-host 外部 Infinispan 集群的主机名。 仅在设置了功能 **多站点**、**无集群或 cache-embedded-remote-store** 时才可用。 CLI: **--cache-remote-host** Env: **KC_CACHE_REMOTE_HOST**	
cache-remote-password 对外部 Infinispan 集群进行身份验证的密码。 如果连接到不安全的外部 Infinispan 集群，则它是可选的。如果指定了这个选项，则需要 **cache-remote-username**。 CLI: **--cache-remote-password** Env: **KC_CACHE_REMOTE_PASSWORD** 仅在设置远程主机时可用	

	value
cache-remote-port 外部 Infinispan 集群的端口。 CLI: **--cache-remote-port** Env: **KC_CACHE_REMOTE_PORT** 仅在设置远程主机时可用	**11222** （默认）
cache-remote-tls-enabled 启用 TLS 支持与安全远程 Infinispan 服务器通信。 建议在生产环境中启用。 CLI: **--cache-remote-tls-enabled** Env: **KC_CACHE_REMOTE_TLS_ENABLED** 仅在设置远程主机时可用	**true** （默认），**false**
cache-remote-username 外部 Infinispan 集群身份验证的用户名。 如果连接到不安全的外部 Infinispan 集群，则它是可选的。如果指定了 选项，则需要 **cache-remote-password**。 CLI: **--cache-remote-username** Env: **KC_CACHE_REMOTE_USERNAME** 仅在设置远程主机时可用	
cache-stack 定义用于集群通信和节点发现的默认堆栈。 如果没有设置，则默认为 **jdbc-ping**。 CLI: **--cache-stack** Env: **KC_CACHE_STACK** 仅在 'cache' type 设为 'ispn' 时才可用 使用 'jdbc-ping' 相反，方法是取消设置 Deprecated 值：**azure**,**ec2**,**google**,**tcp**,**udp**,**jdbc-ping-udp**	**jdbc-ping**,**kubernetes**,**jdbc-ping-udp** （已弃用）、**tcp** （已弃用）、**udp** （已弃用）、**ec2** （已弃用）、**azure** （已弃用）、**google** （已弃用）或任何

21.2. CONFIG

	value
config-keystore 指定 KeyStore 配置源的路径。 CLI: **--config-keystore** Env: **KC_CONFIG_KEYSTORE**	
config-keystore-password 指定 KeyStore 配置源的密码。 CLI: **--config-keystore-password** Env: **KC_CONFIG_KEYSTORE_PASSWORD**	
config-keystore-type 指定 KeyStore 配置源的类型。 CLI: **--config-keystore-type** Env: **KC_CONFIG_KEYSTORE_TYPE**	**PKCS12** （默认）

21.3. 数据库

	value
db ▮ 数据库供应商。 在 production 模式中，**dev-file** 的默认值已弃用，您应该明确指定 db。 CLI: -- **db** Env: **KC_DB**	**dev-file** (default), **dev-mem**,**mariadb**,**mssql**, **mysql**,**oracle**,**postgres**
db-driver ▮ JDBC 驱动程序的完全限定类名称。 如果没有设置，则会将默认驱动程序相应地设置为所选数据库。 CLI: **--db-driver** Env: **KC_DB_DRIVER**	
db-password 数据库用户的密码。 CLI: **--db-password** Env: **KC_DB_PASSWORD**	

	value
db-pool-initial-size 连接池的初始大小。 CLI: **--db-pool-initial-size** Env: **KC_DB_POOL_INITIAL_SIZE**	
db-pool-max-size 连接池的最大大小。 CLI: **--db-pool-max-size** Env: **KC_DB_POOL_MAX_SIZE**	**100**（默认）
db-pool-min-size 连接池的最小大小。 CLI: **--db-pool-min-size** Env: **KC_DB_POOL_MIN_SIZE**	
db-schema 要使用的数据库模式。 CLI: **--db-schema** Env: **KC_DB_SCHEMA**	
db-url 完整的数据库 JDBC URL。 如果没有提供，则会根据所选数据库供应商设置默认 URL。例如，如果使用 **postgres**，默认的 JDBC URL 为 **jdbc:postgresql://localhost/keycloak**。 CLI: **--db-url** Env: **KC_DB_URL**	
db-url-database 设置所选供应商的默认 JDBC URL 的数据库名称。 如果设置了 **db-url** 选项，则忽略这个选项。 CLI: **--db-url-database** Env: **KC_DB_URL_DATABASE**	

	value
db-url-host 设置所选供应商的默认 JDBC URL 的主机名。 如果设置了 **db-url** 选项，则忽略这个选项。 CLI: **--db-url-host** Env: **KC_DB_URL_HOST**	
db-url-port 设置所选供应商的默认 JDBC URL 端口。 如果设置了 **db-url** 选项，则忽略这个选项。 CLI: **--db-url-port** Env: **KC_DB_URL_PORT**	
db-url-properties 设置所选供应商的默认 JDBC URL 的属性。 确保将属性相应地设置为数据库供应商期望的格式，并在此属性值的开头附加正确的字符。如果设置了 **db-url** 选项，则忽略这个选项。 CLI: **--db-url-properties** Env: **KC_DB_URL_PROPERTIES**	
db-username 数据库用户的用户名。 CLI: **--db-username** Env: **KC_DB_USERNAME**	

21.4. TRANSACTION

	value
Transaction-xa-enabled wagon 如果设置为 true，则使用 XA 数据源。 CLI: **--transaction-xa-enabled** Env: **KC_TRANSACTION_XA_ENABLED**	**true**,**false** （默认）

21.5. 功能

	value
功能 启用一个或多个功能的集合。 CLI: -- **features** Env: **KC_FEATURES**	**account-api[:v1]**, **account[:v3]**, **admin-api[:v1]**, **admin-fine-grained-authz[:v1,v2]**, **admin[:v2]**, **authorization[:v1]**, **cache-embedded-remote-store[:v1]**, **ciba[:v1]**, **client-policies[:v** 1] **, client-secret-rotation[:v** 1] **, client-types[:v** 1] **, clusterless[:v** 1] **, declarative-ui[:v** 1] **, device-flow[:v** 1] **, docker[:v** 1] **, dpop[:v** 1] **, dynamic-scopes[:v** 1] **, fips[:** v1] **, hostname[:** v2] **, impersonation[:** v1] **, ipa-tuura-federation[:** v1] **, kerberos[:** v1] **, login[:v2,** v1] **, multi-site[:** v1] **, oid4vc-vci[:** v1] , **Opentelemetry[:** **v1], organization[:v 1] , par[:v 1] , passkeys[:v 1] , persistent-user-sessions[:v 1],preview , quick-theme[:v 1] , recovery-codes[:v 1] , rolling-updates[:v 1] , scripts[:v 1] , step-up-authentication** [: **v1] , token-exchange-standard** [: **v2] , token-exchange** [:v **1] , ephemeral-users** [: **v1] , update-email** [: **v1] , user-event-metrics** [: **v1] , web-authn**[:v1]

	value
features-disabled swig 禁用一个或多个功能的集合。 CLI: **--features-disabled** Env: **KC_FEATURES_DISABLED**	帐户,**account-api**,**admin**,**admin-api**,**admin-fine-grained-authz**,**authorization**,**cache-embedded-remote-store**,**ciba**,**client-policies**,**client-secret-rotation**,**client-types**,**clusterless**, clusterless**-ui**,**device-flow**,**docker**,**d** pop**, dynamic-** scopes**,** fips**,** impersonation**, ipa-tuura-** federation**,** kerberos**,** login ,**oid4vc-v** ci , **opentelemetry, organization, par, pass keys, persistent-user-sessions, preview , preview, quick -theme, recovery -codes, rolling -updates, scripts, step-up -authentication, token -exchange, token-exchange -standard, transient -users, update -email, user-event -metrics, web-authn**

21.6. 主机名 V2

	value
hostname 服务器所公开的地址。 可以是完整的 URL，也可以是主机名。当只提供主机名时，会从请求解析端口和上下文路径。 CLI: -- **hostname** Env: **KC_HOSTNAME** 仅在启用 hostname:v2 功能时才可用	
hostname-admin 用于访问管理控制台的地址。 如果您使用反向代理在与 **hostname** 选项中指定的不同地址上公开管理控制台，则使用这个选项。 CLI: **--hostname-admin** Env: **KC_HOSTNAME_ADMIN** 仅在启用 hostname:v2 功能时才可用	
hostname-backchannel-dynamic 启用动态解析回溯通道 URL，包括主机名、方案、端口和上下文路径。 如果您的应用程序通过私有网络访问 Keycloak，则设置为 true。如果设置为 true，则需要将 **hostname** 选项指定为完整的 URL。 CLI: **--hostname-backchannel-dynamic** Env: **KC_HOSTNAME_BACKCHANNEL_DYNAMIC** 仅在启用 hostname:v2 功能时才可用	**true**,**false** （默认）
hostname-debug 切换可在 /realms/master/hostname-debug 访问的主机名调试页面。 CLI: **--hostname-debug** Env: **KC_HOSTNAME_DEBUG** 仅在启用 hostname:v2 功能时才可用	**true**,**false** （默认）

	value
hostname-strict 禁用从请求标头动态解析主机名。 在生产环境中，应始终设为 true，除非反向代理会覆盖 Host 标头。如果启用，则需要指定 **hostname** 选项。 CLI: **--hostname-strict** Env: **KC_HOSTNAME_STRICT** 仅在启用 hostname:v2 功能时才可用	**true** （默认），**false**

21.7. HTTP(S)

	value
http-enabled 启用 HTTP 侦听器。 在开发模式中默认启用。除非服务器前面是 TLS 终止代理，否则通常不会在生产环境中启用。 CLI: **--http-enabled** Env: **KC_HTTP_ENABLED**	**true**,**false** （默认）
http-host HTTP 主机. CLI: **--http-host** Env: **KC_HTTP_HOST**	**0.0.0.0** （默认）
http-max-queued-requests 排队的 HTTP 请求数上限。 在过载的情况下，使用它来更正负载。过量请求将返回 "503 Server not Available" 响应。 CLI: **--http-max-queued-requests** Env: **KC_HTTP_MAX_QUEUED_REQUESTS**	

	value
http-metrics-histograms-enabled 在 HTTP 服务器请求期间启用带有默认存储桶的直方图。 CLI: **--http-metrics-histograms-enabled** Env: **KC_METRICS_HISTOGRAMS_ENABLED** 仅在启用指标时可用	**true**,**false** （默认）
http-metrics-slos HTTP 服务器请求的服务级别目标。 使用这个选项而不是默认的直方图，或者结合使用它来添加额外的存储桶。指定以逗号分开的值列表（以毫秒为单位）。bucket 从 5ms 到 10s: 5,10,25,50,250,500,1000,2500,5000,10000 CLI: **--http-metrics-slos** Env: **KC_HTTP_METRICS_SLOS** 仅在启用指标时可用	
http-pool-max-threads 最大线程数。 如果没有指定，则它将自动调整为大于 4 * 可用处理器数量和 50。例如，如果最多线程为 50 个处理器，则最大线程数为 50。如果有 48 个处理器，它将是 192 个。 CLI: **--http-pool-max-threads** Env: **KC_HTTP_POOL_MAX_THREADS**	
http-port 使用的 HTTP 端口。 CLI: **--http-port** Env: **KC_HTTP_PORT**	**8080** （默认）
http-relative-path ▮ 设置用于服务资源的路径相对于 / 的路径。 该路径必须以 / 开头。 CLI: **--http-relative-path** Env: **KC_HTTP_RELATIVE_PATH**	/ （默认）

	value
https-certificate-file PEM 格式的服务器证书或证书链的文件路径。 CLI: **--https-certificate-file** Env: **KC_HTTPS_CERTIFICATE_FILE**	
https-certificate-key-file PEM 格式的私钥的文件路径。 CLI: **--https-certificate-key-file** Env: **KC_HTTPS_CERTIFICATE_KEY_FILE**	
https-certificates-reload-period 重新载入密钥存储、信任存储和证书文件的间隔，由 https channel 选项引用。 可以是 java.time.Duration 值、整数数或整数，后跟 [ms, h, m, s, d] 之一。必须大于 30 秒。使用 -1 禁用。 CLI: **--https-certificates-reload-period** Env: **KC_HTTPS_CERTIFICATES_RELOAD_PERIOD**	**1h**（默认）
https-cipher-suites 要使用的密码套件。 如果未指定，则会选择合理的默认值。 CLI: **--https-cipher-suites** Env: **KC_HTTPS_CIPHER_SUITES**	
https-client-auth 将服务器配置为 require/request 客户端身份验证。 CLI: **--https-client-auth** Env: **KC_HTTPS_CLIENT_AUTH**	**none**（默认），请求，**required**
https-key-store-file 保存证书信息的密钥存储，而不是指定单独的文件。 CLI: **--https-key-store-file** Env: **KC_HTTPS_KEY_STORE_FILE**	

	value
https-key-store-password 密钥存储文件的密码。 CLI: **--https-key-store-password** Env: **KC_HTTPS_KEY_STORE_PASSWORD**	**密码** （默认）
https-key-store-type 密钥存储文件的类型。 如果未指定，则根据文件扩展名自动检测到类型。如果将 **fips-mode** 设为 **strict**，且没有设置值，则默认为 **BCFKS**。 CLI: **--https-key-store-type** Env: **KC_HTTPS_KEY_STORE_TYPE**	
https-port 使用的 HTTPS 端口。 CLI: **--https-port** Env: **KC_HTTPS_PORT**	**8443** （默认）
https-protocols 要显式启用的协议列表。 如果 JRE / 安全配置不支持值，它将被静默忽略。 CLI: **--https-protocols** Env: **KC_HTTPS_PROTOCOLS**	**TLSv1.3**、**TLSv1.2** 或任何
https-trust-store-file 保存要信任的证书的证书信息的信任存储。 CLI: **--https-trust-store-file** Env: **KC_HTTPS_TRUST_STORE_FILE**	
https-trust-store-password 信任存储文件的密码。 CLI: **--https-trust-store-password** Env: **KC_HTTPS_TRUST_STORE_PASSWORD**	

	value
https-trust-store-type 信任存储文件的类型。 如果未指定，则根据文件扩展名自动检测到类型。如果将 **fips-mode** 设为 **strict**，且没有设置值，则默认为 **BCFKS**。 CLI: **--https-trust-store-type** Env: **KC_HTTPS_TRUST_STORE_TYPE**	

21.8. HEALTH

	value
启用 健康状态 如果服务器应该公开健康检查端点。 如果启用，健康检查位于 /health、 **/health /ready** 和 **/health/live** 端点上。 CLI: **--health-enabled** Env: **KC_HEALTH_ENABLED**	**true**,**false** （默认）

21.9. 管理

	value
http-management-port 管理接口的端口。 仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--http-management-port** Env: **KC_HTTP_MANAGEMENT_PORT**	**9000** （默认）
http-management-relative-path 设置相对于 / 的路径，以便从管理界面提供资源。 该路径必须以 / 开头。如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--http-management-relative-path** Env: **KC_HTTP_MANAGEMENT_RELATIVE_PATH**	/ （默认）

	value
https-management-certificate-file 管理服务器的 PEM 格式的服务器证书或证书链的文件路径。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-certificate-file** Env: **KC_HTTPS_MANAGEMENT_CERTIFICATE_FILE**	
https-management-certificate-key-file 管理服务器的 PEM 格式的私钥的文件路径。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-certificate-key-file** Env: **KC_HTTPS_MANAGEMENT_CERTIFICATE_KEY_FILE**	
https-management-certificates-reload-period 重新载入管理服务器的 https-management ldapmodify 选项引用的密钥存储、信任存储和证书文件的时间间隔。 可以是 java.time.Duration 值、整数数或整数，后跟 [ms, h, m, s, d] 之一。必须大于 30 秒。使用 -1 禁用。如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-certificates-reload-period** Env: **KC_HTTPS_MANAGEMENT_CERTIFICATES_RELOAD_PERIOD**	**1h** （默认）
https-management-client-auth ▮ 将管理接口配置为 require/request 客户端身份验证。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-client-auth** Env: **KC_HTTPS_MANAGEMENT_CLIENT_AUTH**	**none** （默认），请求，**required**
https-management-key-store-file 保存证书信息的密钥存储，而不是为管理服务器指定单独的文件。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-key-store-file** Env: **KC_HTTPS_MANAGEMENT_KEY_STORE_FILE**	

	value
https-management-key-store-password 管理服务器的密钥存储文件的密码。 如果未指定，则该值从 HTTP 选项继承。仅在管理界面上公开内容时相关 - 详情请参阅指南。 CLI: **--https-management-key-store-password** Env: **KC_HTTPS_MANAGEMENT_KEY_STORE_PASSWORD**	**密码** （默认）
legacy-observability-interface wagon 如果应在主 HTTP 服务器上公开 metrics/health 端点（不推荐）。 如果设置为 true，则禁用管理界面。 CLI: **--legacy-observability-interface** Env: **KC_LEGACY_OBSERVABILITY_INTERFACE** **已弃用。**	**true**,**false** （默认）

21.10. 指标

	value
启用 指标 如果服务器应该公开指标。 如果启用，则指标位于 **/metrics** 端点。 CLI: **--metrics-enabled** Env: **KC_METRICS_ENABLED**	**true**,**false** （默认）

21.11. PROXY

	value
proxy-headers 服务器应接受的代理标头。 错误配置可能会使服务器暴露于安全漏洞。优先于已弃用的代理选项。 CLI: **--proxy-headers** Env: **KC_PROXY_HEADERS**	**转发**,**xforwarded**

	value
proxy-protocol-enabled 服务器是否应该在代理后提供请求时使用 HA PROXY 协议。 当设置为 true 时，返回的远程地址将是实际连接客户端中的地址。使用 **proxy-headers** 时无法启用。 CLI: **--proxy-protocol-enabled** Env: **KC_PROXY_PROTOCOL_ENABLED**	**true**,**false** （默认）
proxy-trusted-addresses 以逗号分隔的可信代理地址列表。 如果设置，则忽略来自其他地址的代理标头。默认情况下，所有地址都是可信的。可信代理地址被指定为 IP 地址(IPv4 或 IPv6)或无类别域间路由(CIDR)标记。仅在设置 proxy-headers 时可用。 CLI: **--proxy-trusted-addresses** Env: **KC_PROXY_TRUSTED_ADDRESSES**	

21.12. VAULT

	value
vault ▮ 启用 vault 供应商。 CLI: -- **vault** Env: **KC_VAULT**	**文件**,**keystore**
vault-dir 如果设置，可以通过读取给定目录中文件的内容来获取 secret。 CLI: **--vault-dir** Env: **KC_VAULT_DIR**	
vault-file 密钥存储文件的路径。 CLI: **--vault-file** Env: **KC_VAULT_FILE**	

	value
vault-pass vault 密钥存储的密码。 CLI: **--vault-pass** Env: **KC_VAULT_PASS**	
vault-type 指定密钥存储文件的类型。 CLI: **--vault-type** Env: **KC_VAULT_TYPE**	**PKCS12** （默认）

21.13. 日志记录

	value
log 在以逗号分隔的列表中启用一个或多个日志处理程序。 CLI: -- **log** Env: **KC_LOG**	**控制台**,**文件**,**syslog**
log-console-color 登录到控制台时启用或禁用颜色。 CLI: **--log-console-color** Env: **KC_LOG_CONSOLE_COLOR** 仅在激活 Console 日志处理程序时可用	**true**,**false** （默认）
log-console-format 无结构控制台日志条目的格式。 如果格式有空格，请使用 "<format>" 转义值。 CLI: **--log-console-format** Env: **KC_LOG_CONSOLE_FORMAT** 仅在激活 Console 日志处理程序时可用	**%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n** (default)

	value
log-console-include-trace 在控制台日志中包括追踪信息。 如果指定了 **log-console-format** 选项，这个选项无效。 CLI: **--log-console-include-trace** Env: **KC_LOG_CONSOLE_INCLUDE_TRACE** 仅在激活 Console 日志处理程序和 Tracing 时可用	**true** （默认），**false**
log-console-json-format 设置生成的 JSON 格式。 CLI: **--log-console-json-format** Env: **KC_LOG_CONSOLE_JSON_FORMAT** 仅在激活 Console 日志处理程序并且输出设置为 'json' 时才可用	**默认** （默认），**ecs**
log-console-level 设置控制台处理程序的日志级别。 它指定输出中显示的日志的最详细日志级别。它遵循 **log-level** 选项指定的级别，它代表整个日志记录系统的最大详细程度。如需更多信息，请参阅 Logging 指南。 CLI: **--log-console-level** Env: **KC_LOG_CONSOLE_LEVEL** 仅在激活 Console 日志处理程序时可用	**off**,**fatal**,**error**,**warn**,**info**,**debug**,**trace**,**all** （默认）
log-console-output 将日志输出设置为 JSON 或默认(plain)非结构化日志记录。 CLI: **--log-console-output** Env: **KC_LOG_CONSOLE_OUTPUT** 仅在激活 Console 日志处理程序时可用	**默认** （默认），**json**
log-file 设置日志文件路径和文件名。 CLI: **--log-file** Env: **KC_LOG_FILE** 仅在文件日志处理程序激活时才可用	**data/log/keycloak.log** （默认）

	value
log-file-format 设置特定于文件日志条目的格式。 CLI: **--log-file-format** Env: **KC_LOG_FILE_FORMAT** 仅在文件日志处理程序激活时才可用	**%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n** (default)
log-file-include-trace 在文件日志中包含追踪信息。 如果指定了 **log-file-format** 选项，这个选项无效。 CLI: **--log-file-include-trace** Env: **KC_LOG_FILE_INCLUDE_TRACE** 仅在激活文件日志处理程序和跟踪时才可用	**true** （默认），**false**
log-file-json-format 设置生成的 JSON 格式。 CLI: **--log-file-json-format** Env: **KC_LOG_FILE_JSON_FORMAT** 仅在文件日志处理程序激活并且输出设置为 'json' 时才可用	**默认** （默认），**ecs**
log-file-level 设置文件处理程序的日志级别。 它指定输出中显示的日志的最详细日志级别。它遵循 **log-level** 选项指定的级别，它代表整个日志记录系统的最大详细程度。如需更多信息，请参阅 Logging 指南。 CLI: **--log-file-level** Env: **KC_LOG_FILE_LEVEL** 仅在文件日志处理程序激活时才可用	**off**,**fatal**,**error**,**warn**,**info**,**debug**,**trace**,**all** （默认）
log-file-output 将日志输出设置为 JSON 或默认(plain)非结构化日志记录。 CLI: **--log-file-output** Env: **KC_LOG_FILE_OUTPUT** 仅在文件日志处理程序激活时才可用	**默认** （默认），**json**

	value
log-level 根类别的日志级别或以逗号分隔的类别列表及其级别。 对于 root 类别，您不需要指定一个类别。 CLI: **--log-level** Env: **KC_LOG_LEVEL**	**[info]**（默认）
log-syslog-app-name 设置使用 RFC5424 格式格式化消息时使用的应用程序名称。 CLI: **--log-syslog-app-name** Env: **KC_LOG_SYSLOG_APP_NAME** 仅在 Syslog 激活时才可用	**Keycloak**（默认）
log-syslog-endpoint 设置 Syslog 服务器的 IP 地址和端口。 CLI: **--log-syslog-endpoint** Env: **KC_LOG_SYSLOG_ENDPOINT** 仅在 Syslog 激活时才可用	**localhost:514**（默认）
log-syslog-format 设置特定于 Syslog 条目的格式。 CLI: **--log-syslog-format** Env: **KC_LOG_SYSLOG_FORMAT** 仅在 Syslog 激活时才可用	**%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n** (default)
log-syslog-include-trace 在 Syslog 中包含追踪信息。 如果指定了 **log-syslog-format** 选项，这个选项无效。 CLI: **--log-syslog-include-trace** Env: **KC_LOG_SYSLOG_INCLUDE_TRACE** 仅在激活 Syslog 处理程序和 Tracing 时可用	**true**（默认），**false**

	value
log-syslog-json-format 设置生成的 JSON 格式。 CLI: **--log-syslog-json-format** Env: **KC_LOG_SYSLOG_JSON_FORMAT** 仅在 Syslog 激活且输出设置为 'json' 时才可用	**默认**（默认），**ecs**
log-syslog-level 设置 Syslog 处理程序的日志级别。 它指定输出中显示的日志的最详细日志级别。它遵循 **log-level** 选项指定的级别，它代表整个日志记录系统的最大详细程度。如需更多信息，请参阅 Logging 指南。 CLI: **--log-syslog-level** Env: **KC_LOG_SYSLOG_LEVEL** 仅在 Syslog 激活时才可用	**off**,**fatal**,**error**,**warn**,**info**,**debug**,**trace**,**all**（默认）
log-syslog-max-length 设置允许发送的消息的最大长度（以字节为单位）。 长度包括标头和消息。如果没有设置，当 **log-syslog-type** 为 rfc5424 （默认）和 1024 时，当 **log-syslog-type** 为 rfc3164 时，默认值为 2048。 CLI: **--log-syslog-max-length** Env: **KC_LOG_SYSLOG_MAX_LENGTH** 仅在 Syslog 激活时才可用	
log-syslog-output 将 Syslog 输出设置为 JSON 或默认(plain)非结构化日志记录。 CLI: **--log-syslog-output** Env: **KC_LOG_SYSLOG_OUTPUT** 仅在 Syslog 激活时才可用	**默认**（默认），**json**
log-syslog-protocol 设置用于连接 Syslog 服务器的协议。 CLI: **--log-syslog-protocol** Env: **KC_LOG_SYSLOG_PROTOCOL** 仅在 Syslog 激活时才可用	**TCP**（默认）、**udp**、**ssl-tcp**

	value
log-syslog-type 设置用于格式化发送消息的 Syslog 类型。 CLI: **--log-syslog-type** Env: **KC_LOG_SYSLOG_TYPE** 仅在 Syslog 激活时才可用	**RFC5424**（默认），**rfc3164**

21.14. TRACING

	value
tracing-compression OpenTelemetry 压缩方法用于压缩有效负载。 如果未设置，则禁用压缩。 CLI: **--tracing-compression** Env: **KC_TRACING_COMPRESSION** 仅在启用 Tracing 时可用	**gzip**，**none**（默认）
启用了追踪的 wagon 启用 OpenTelemetry 追踪。 CLI: **--tracing-enabled** Env: **KC_TRACING_ENABLED** 仅在启用 'opentelemetry' 功能时才可用	**true**,**false**（默认）
tracing-endpoint 要连接的 OpenTelemetry 端点。 CLI: **--tracing-endpoint** Env: **KC_TRACING_ENDPOINT** 仅在启用 Tracing 时可用	**http://localhost:4317** (default)
tracing-jdbc-enabled 启用 OpenTelemetry JDBC 追踪。 CLI: **--tracing-jdbc-enabled** Env: **KC_TRACING_JDBC_ENABLED** 仅在启用 Tracing 时可用	**true**（默认），**false**

	value
tracing-protocol OpenTelemetry 协议用于遥测数据。 CLI: **--tracing-protocol** Env: **KC_TRACING_PROTOCOL** 仅在启用 Tracing 时可用	**grpc** (default), **http/protobuf**
tracing-resource-attributes OpenTelemetry 资源属性存在于导出的 trace 中，以特征遥测制作者。 格式为 **key1=val1,key2=val2** 的值。如需更多信息，请参阅跟踪指南。 CLI: **--tracing-resource-attributes** Env: **KC_TRACING_RESOURCE_ATTRIBUTES** 仅在启用 Tracing 时可用	
tracing-sampler-ratio OpenTelemetry sampler 比率。 跨度的概率将被抽样。预期的双值（间隔为 [0,1]）。 CLI: **--tracing-sampler-ratio** Env: **KC_TRACING_SAMPLER_RATIO** 仅在启用 Tracing 时可用	**1.0** （默认）
tracing-sampler-type ▌ OpenTelemetry sampler 用于追踪。 CLI: **--tracing-sampler-type** Env: **KC_TRACING_SAMPLER_TYPE** 仅在启用 Tracing 时可用	**always_on**,**always_off**,**traceidratio** (default), **parentbased_always_on**,**parentbased_always_off**,**parentbased_traceidratio**
tracing-service-name OpenTelemetry 服务名称。 优先于 **tracing-resource-attributes** 属性中定义的 **service.name**。 CLI: **--tracing-service-name** Env: **KC_TRACING_SERVICE_NAME** 仅在启用 Tracing 时可用	**Keycloak** （默认）

21.15. 事件

	value
event-metrics-user-enabled 基于用户事件创建指标。 CLI: **--event-metrics-user-enabled** Env: **KC_EVENT_METRICS_USER_ENABLED** 仅在启用指标并启用了 user-event-metrics 功能时才可用	**true**,**false** （默认）
event-metrics-user-events 为用户事件指标收集的、以逗号分隔的事件列表。 此选项可用于减少默认为所有用户事件创建的指标数量。 CLI: **--event-metrics-user-events** Env: **KC_EVENT_METRICS_USER_EVENTS** 仅在启用用户事件指标时可用 使用 **remove_credential** 而不是 **remove_totp**，并使用 **update_credential** 而不是 **update_totp** 和 **update_password**。**弃用的值：remove_totp**,**update_totp**,**update_password**	**authreqid_to_token**,**client_delete**,**client_info**,**client_initiated_account_linking**,**client_login**,**client_register**,**client_update**,**code_to_token**,**custom_required_action**,**delete_account**,**execute_action_token**,**execute_actions**,**federated_identity_** link, **federated_identity_override_** link, **grant_con** sent, **identity_provider_first_** login, **identity_provider_link_** account, **identity_provider_** login, **identity_provider_post_** login, **identity_provider_** response, **identity_provider_retrieve_** token, impersonate ,**introspection** _token, **invalid** _signature, **invite** _org, **login**, **logout**, **oauth2_device_auth**, **oauth2_device_code_to _token**, **oauth2_device_verify_user _code**, **oauth2_extension**

	value
	_grant, permission _token, pushed_authorizatio n _request, refresh _token , register ,register_node,remo ve_credential,remov e_federated_identity, remove_totp (deprecated), reset_password , restart_authenticatio n , revoke_grant ,send_identity_provi der_link,send_reset_ password,send_verif y_email,token_excha nge,unregister_node ,update_consent,upd ate_credential,updat e_email,update_pass word （已弃用）、update_profile 、update_totp （已弃用）、user_disabled _by_permanent_lock out,user_disabled_b y_temporary_lockou t,user_info_request, verify_email,verify_p rofile
event-metrics-user-tags 为用户事件指标收集的以逗号分隔的标签列表。 默认情况下，只启用 **realm** 来避免高指标卡。 CLI: **--event-metrics-user-tags** Env: **KC_EVENT_METRICS_USER_TAGS** 仅在启用用户事件指标时可用	**realm,idp,clientId**

21.16. TRUSTSTORE

	value
tls-hostname-verifier 用于传出 HTTPS 和 SMTP 请求的 TLS 主机名验证策略。 ANY 不应在生产环境中使用。 CLI: **--tls-hostname-verifier** Env: **KC_TLS_HOSTNAME_VERIFIER** STRICT 和 WILDCARD 已被弃用，改为使用 DEFAULT。**已弃用的值：STRICT,WILDCARD**	**ANY**,**WILDCARD**（已弃用）、**STRICT**（已弃用）、**DEFAULT**（默认）
truststore-paths pkcs12 列表(p12、pfx 或 pkcs12 文件扩展)、PEM 文件或目录，其中包含要用作系统信任存储的这些文件。 CLI: **--truststore-paths** Env: **KC_TRUSTSTORE_PATHS**	

21.17. 安全性

	value
fips-mode ▮ 设置 FIPS 模式。 如果设置了 **非限制**，则在非批准模式下启用 FIPS。对于完整的 FIPS 合规性，将 **strict** 设置为在批准的模式下运行。**当禁用 fips** 功能时，这个选项默认为禁用。当 **fips** 功能被启用时，这个选项默认为 **non-strict**。 CLI: **--fips-mode** Env: **KC_FIPS_MODE**	**非限制**,**strict**

21.18. EXPORT

	value
dir 设置使用导出的数据创建文件的目录的路径。 CLI: -- **dir** Env: **KC_DIR**	

	value
file 设置将使用导出的数据创建的文件的路径。 要导出超过 500 个用户，请导出到具有不同文件的目录。 CLI: -- **file** Env: **KC_FILE**	
realm 设置要导出的域的名称。 如果没有设置，则会导出所有域。 CLI: -- **realm** Env: **KC_REALM**	
用户 设置应如何导出用户。 CLI: -- **users** Env: **KC_USERS**	**跳过**,**realm_file**,**same_file**,**different_files**（默认）
users-per-file 设置每个文件的用户数量。 只有在 **用户** 设置为 **different_files** 时，才会使用它。增加这个数字会导致导出时间指数增加。 CLI: **--users-per-file** Env: **KC_USERS_PER_FILE**	**50**（默认）

21.19. IMPORT

	value
dir 设置要从其中读取文件的目录的路径。 CLI: -- **dir** Env: **KC_DIR**	

	value
file 设置将要读取的文件的路径。 CLI: -- **file** Env: **KC_FILE**	
override 设置现有数据是否应被覆盖。 如果设置为 false，则忽略数据。 CLI: -- **override** Env: **KC_OVERRIDE**	**true**（默认），**false**

21.20. BOOTSTRAP ADMIN

	value
bootstrap-admin-client-id 临时 bootstrap admin 服务帐户的客户端 ID。 仅在创建 master 域时使用。仅在设置 bootstrap admin 客户端 secret 时可用。 CLI: **--bootstrap-admin-client-id** Env: **KC_BOOTSTRAP_ADMIN_CLIENT_ID**	**temp-admin** (default)
bootstrap-admin-client-secret 临时 bootstrap admin 服务帐户的客户端 secret。 仅在创建 master 域时使用。如果可能，请使用非CLI 配置选项。 CLI: **--bootstrap-admin-client-secret** Env: **KC_BOOTSTRAP_ADMIN_CLIENT_SECRET**	
bootstrap-admin-password 临时 bootstrap 管理密码。 仅在创建 master 域时使用。如果可能，请使用非CLI 配置选项。 CLI: **--bootstrap-admin-password** Env: **KC_BOOTSTRAP_ADMIN_PASSWORD**	

	value
bootstrap-admin-username 临时 bootstrap 管理用户名。 仅在创建 master 域时使用。仅在设置 bootstrap admin 密码时才可用。 CLI: **--bootstrap-admin-username** Env: **KC_BOOTSTRAP_ADMIN_USERNAME**	**temp-admin** (default)

第 22 章 所有供应商配置

查看供应商配置选项。

22.1. AUTHENTICATION-SESSIONS

22.1.1. infinispan

	value
spi-authentication-sessions-infinispan-auth-sessions-limit 每个 RootAuthenticationSession 的最大并发身份验证会话数。 CLI: **--spi-authentication-sessions-infinispan-auth-sessions-limit** Env: **KC_SPI_AUTHENTICATION_SESSIONS_INFINISPAN_AUTH_SESSIONS_LIMIT**	**300** （默认）或任何 **int**

22.1.2. remote

	value
spi-authentication-sessions-remote-auth-sessions-limit 每个 RootAuthenticationSession 的最大并发身份验证会话数。 CLI: **--spi-authentication-sessions-remote-auth-sessions-limit** Env: **KC_SPI_AUTHENTICATION_SESSIONS_REMOTE_AUTH_SESSIONS_LIMIT**	**300** （默认）或任何 **int**
spi-authentication-sessions-remote-max-retries 发生错误时的最大重试次数。 值为零或更少禁用任何重试。 CLI: **--spi-authentication-sessions-remote-max-retries** Env: **KC_SPI_AUTHENTICATION_SESSIONS_REMOTE_MAX_RETRIES**	**10** （默认）或任何 **int**

	value
spi-authentication-sessions-remote-retry-base-time 基础后端时间（以毫秒为单位）。 CLI: **--spi-authentication-sessions-remote-retry-base-time** Env: **KC_SPI_AUTHENTICATION_SESSIONS_REMOTE_RETRY_BASE_TIME**	**10**（默认）或任何 **int**

22.2. BRUTE-FORCE-PROTECTOR

22.2.1. default-brute-force-detector

	value
spi-brute-force-protector-default-brute-force-detector-allow-concurrent-requests 如果括号强制保护允许并发登录。 CLI: **--spi-brute-force-protector-default-brute-force-detector-allow-concurrent-requests** Env: **KC_SPI_BRUTE_FORCE_PROTECTOR_DEFAULT_BRUTE_FORCE_DETECTOR_ALLOW_CONCURRENT_REQUESTS**	**true**,**false**（默认）

22.3. CIBA-AUTH-CHANNEL

22.3.1. ciba-http-auth-channel

	value
spi-ciba-auth-channel-ciba-http-auth-channel-http-authentication-channel-uri 身份验证频道的 HTTP (S) URI。 CLI: **--spi-ciba-auth-channel-ciba-http-auth-channel-http-authentication-channel-uri** Env: **KC_SPI_CIBA_AUTH_CHANNEL_CIBA_HTTP_AUTH_CHANNEL_AUTHENTICATION_CHANNEL_URI**	**任何字符串**

22.4. CONNECTIONS-HTTP-CLIENT

22.4.1. default

	value
spi-connections-http-client-default-client-key-password 密钥密码。 CLI: **--spi-connections-http-client-default-client-key-password** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_CLIENT_KEY_PASSWORD**	**-1**（默认）或 **任何字符串**
spi-connections-http-client-default-client-keystore 从中读取关键资料到设置 TLS 连接的密钥存储的文件路径。 CLI: **--spi-connections-http-client-default-client-keystore** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_CLIENT_KEYSTORE**	**任何字符串**
spi-connections-http-client-default-client-keystore-password 密钥存储密码。 CLI: **--spi-connections-http-client-default-client-keystore-password** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_CLIENT_KEYSTORE_PASSWORD**	**任何字符串**
spi-connections-http-client-default-connection-pool-size 分配最大连接值。 CLI: **--spi-connections-http-client-default-connection-pool-size** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_CONNECTION_POOL_SIZE**	any **int**
spi-connections-http-client-default-connection-ttl-millis 将持久连接的最大时间（以毫秒为单位）设置为 live。 CLI: **--spi-connections-http-client-default-connection-ttl-millis** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_CONNECTION_TTL_MILLIS**	**-1**（默认）或任何 长

	value
spi-connections-http-client-default-disable-cookies 禁用状态(cookie)管理。 CLI: **--spi-connections-http-client-default-disable-cookies** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_DISABLE_COOKIES**	**true** （默认），**false**
spi-connections-http-client-default-disable-trust-manager 禁用信任管理和主机名验证。 请注意，这是安全漏洞，因此仅当您无法或者不想验证您要与之通信的主机身份时，才设置这个选项。 CLI: **--spi-connections-http-client-default-disable-trust-manager** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_DISABLE_TRUST_MANAGER**	**true**,**false** （默认）
spi-connections-http-client-default-establish-connection-timeout-millis 当尝试进行初始套接字连接时，超时是什么？ CLI: **--spi-connections-http-client-default- establish-connection-timeout-millis** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_ESTABLISH_CONNECTION_TIMEOUT_MILLIS**	**-1** （默认）或任何长
spi-connections-http-client-default-max-connection-idle-time-millis 设置从池中驱除闲置连接的时间（以毫秒为单位）。 CLI: **--spi-connections-http-client-default-max-connection-idle-time-millis** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_MAX_CONNECTION_IDLE_TIME_MILLIS**	**900000** （默认）或任何长

	value
spi-connections-http-client-default-max-consumed-response-size 客户端消耗的响应的最大大小（以防止拒绝服务） CLI: **--spi-connections-http-client-default-max-consumed-response-size** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_MAX_CONSUMED_RESPONSE_SIZE**	**10000000**（默认）或任何 长
spi-connections-http-client-default-max-pooled-per-route 为每个路由值分配最大连接。 CLI: **--spi-connections-http-client-default-max-pooled-per-route** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_MAX_POOLED_PER_ROUTE**	**64**（默认）或任何 **int**
spi-connections-http-client-default-proxy-mappings 以 hostnamePattern;proxyUri 的形式表示基于 regex 的主机名模式和 proxy-uri 的组合。 CLI: **--spi-connections-http-client-default-proxy-mappings** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_PROXY_MAPPINGS**	**任何字符串**
spi-connections-http-client-default-reuse-connections 如果连接应该被重复使用。 CLI: **--spi-connections-http-client-default-reuse-connections** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_REUSE_CONNECTIONS**	**true**（默认），**false**
spi-connections-http-client-default-socket-timeout-millis 套接字不活跃超时。 CLI: **--spi-connections-http-client-default-socket-timeout-timeout-millis** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_DEFAULT_SOCKET_TIMEOUT_MILLIS**	**5000**（默认）或任何 长

22.4.2. OpenTelemetry

	value
spi-connections-http-client-opentelemetry-client-key-password 密钥密码。 CLI: **--spi-connections-http-client-opentelemetry-client-key-password** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_CLIENT_KEY_PASSWORD**	**-1**（默认）或 **任何字符串**
spi-connections-http-client-opentelemetry-client-keystore 从中读取关键资料到设置 TLS 连接的密钥存储的文件路径。 CLI: **--spi-connections-http-client-opentelemetry-client-keystore** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_CLIENT_KEYSTORE**	**任何字符串**
spi-connections-http-client-opentelemetry-client-keystore-password 密钥存储密码。 CLI: **--spi-connections-http-client-opentelemetry-client-keystore-password** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_CLIENT_KEYSTORE_PASSWORD**	**任何字符串**
spi-connections-http-client-opentelemetry-connection-pool-size 分配最大连接值。 CLI: **--spi-connections-http-client-opentelemetry-connection-pool-size** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_CONNECTION_POOL_SIZE**	any **int**
spi-connections-http-client-opentelemetry-connection-ttl-millis 将持久连接的最大时间（以毫秒为单位）设置为 live。 CLI: **--spi-connections-http-client-opentelemetry-connection-ttl-millis** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_CONNECTION_TTL_MILLIS**	**-1**（默认）或任何 长

	value
spi-connections-http-client-opentelemetry-disable-cookies 禁用状态(cookie)管理。 CLI: **--spi-connections-http-client-opentelemetry-disable-cookies** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_DISABLE_COOKIES**	**true** （默认），**false**
spi-connections-http-client-opentelemetry-disable-trust-manager 禁用信任管理和主机名验证。 请注意，这是安全漏洞，因此仅当您无法或者不想验证您要与之通信的主机身份时，才设置这个选项。 CLI: **--spi-connections-http-client-opentelemetry-disable-trust-manager** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_DISABLE_TRUST_MANAGER**	**true**,**false** （默认）
spi-connections-http-client-opentelemetry-establish-connection-timeout-millis 当尝试进行初始套接字连接时，超时是什么？ CLI: **--spi-connections-http-client-opentelemetry- establish-connection-timeout-millis** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_ESTABLISH_CONNECTION_TIMEOUT_MILLIS**	**-1** （默认）或任何长
spi-connections-http-client-opentelemetry-max-connection-idle-time-millis 设置从池中驱除闲置连接的时间（以毫秒为单位）。 CLI: **--spi-connections-http-client-opentelemetry-max-connection-idle-time-millis** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_MAX_CONNECTION_IDLE_TIME_MILLIS**	**900000** （默认）或任何长

	value
spi-connections-http-client-opentelemetry-max-consumed-response-size 客户端消耗的响应的最大大小（以防止拒绝服务） CLI: **--spi-connections-http-client-opentelemetry-max-consumed-response-size** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_MAX_CONSUMED_RESPONSE_SIZE**	**10000000**（默认）或任何 长
spi-connections-http-client-opentelemetry-max-pooled-per-route 为每个路由值分配最大连接。 CLI: **--spi-connections-http-client-opentelemetry-max-pooled-per-route** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_MAX_POOLED_PER_ROUTE**	**64**（默认）或任何 **int**
spi-connections-http-client-opentelemetry-proxy-mappings 以 hostnamePattern;proxyUri 的形式表示基于 regex 的主机名模式和 proxy-uri 的组合。 CLI: **--spi-connections-http-client-opentelemetry-proxy-mappings** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_PROXY_MAPPINGS**	**任何字符串**
spi-connections-http-client-opentelemetry-reuse-connections 如果连接应该被重复使用。 CLI: **--spi-connections-http-client-opentelemetry-reuse-connections** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_REUSE_CONNECTIONS**	**true**（默认），**false**
spi-connections-http-client-opentelemetry-socket-timeout-millis 套接字不活跃超时。 CLI: **--spi-connections-http-client-opentelemetry-socket-timeout-millis** Env: **KC_SPI_CONNECTIONS_HTTP_CLIENT_OPENTELEMETRY_SOCKET_TIMEOUT_MILLIS**	**5000**（默认）或任何 长

22.5. CONNECTIONS-INFINISPAN

22.5.1. Quarkus

	value
spi-connections-infinispan-quarkus-site-name 多站点部署的站点名称 CLI: **--spi-connections-infinispan-quarkus-site-name** Env: **KC_SPI_CONNECTIONS_INFINISPAN_QUARKUS_SITE_NAME**	**任何字符串**

22.6. CONNECTIONS-JPA

22.6.1. Quarkus

	value
spi-connections-jpa-quarkus-initialize-empty 如果为空，初始化数据库。 如果设置为 false，则必须手动初始化数据库。如果要将数据库设置 migrationStrategy 手动初始化，这将使用 SQL 命令创建一个文件来初始化数据库。 CLI: **--spi-connections-jpa-quarkus-initialize-empty** Env: **KC_SPI_CONNECTIONS_CONNECTION_QUARKUS_INITIALIZE_EMPTY**	**true** （默认），**false**
spi-connections-jpa-quarkus-migration-export 编写手动数据库初始化/迁移文件的路径。 CLI: **--spi-connections-jpa-quarkus-migration-export** Env: **KC_SPI_CONNECTIONS_CONNECTION_QUARKUS_MIGRATION_EXPORT**	**任何字符串**
spi-connections-jpa-quarkus-migration-strategy 用于迁移数据库的策略。 有效值为 update、manual 和 validate。更新 将自动迁移数据库架构。手动将使用 SQL 命令手动将所需的更改导出到文件，从而对数据库进行手动执行。验证将仅检查数据库是否是最新的。 CLI: **--spi-connections-jpa-quarkus-migration-strategy** Env: **KC_SPI_CONNECTIONS_CONNECTION_QUARKUS_MIGRATION_STRATEGY**	**更新** （默认），**手动验证**

22.7. CREDENTIAL

22.7.1. keycloak-password

	value
spi-credential-keycloak-password-validations-counter-tags 发布密码验证计数器指标时使用的标签的逗号分隔列表。 CLI: **--spi-credential-keycloak-password-validations-counter-tags** Env: **KC_SPI_CREDENTIAL_KEYCLOAK_PASSWORD_VALIDATIONS_COUNTER_TAGS**	**realm,algorithm,hash_strength,结果**

22.8. CRL-STORAGE

22.8.1. Infinispan

	value
spi-crl-storage-infinispan-cache-time 缓存 CRL 的时间间隔（以秒为单位）。 如果存在，CRL 的下一次更新时间始终为最小值。零或负值意味着 CRL 被缓存，直到 CRL 中指定的下一次更新时间（如果 CRL 不包含下一次更新，则为无限）。 CLI: **--spi-crl-storage-infinispan-cache-time** Env: **KC_SPI_CRL_STORAGE_INFINISPAN_CACHE_TIME**	**-1**（默认）或任何 **int**
spi-crl-storage-infinispan-min-time-between-requests 两个请求之间用于检索 CRL 的最小间隔（以秒为单位）。 在从之前的刷新开始通过这个最小时间之前，才会再次从 URL 更新 CRL。如果在下一次更新时正确刷新 CRL，则不会使用此选项。间隔应该是正数。默认 10 秒。 CLI: **--spi-crl-storage-infinispan-min-time-between-requests** Env: **KC_SPI_CRL_STORAGE_INFINISPAN_MIN_TIME_BETWEEN_REQUESTS**	**10**（默认）或任何 **int**

22.9. DATASTORE

22.9.1. 传统

	value
spi-datastore-legacy-allow-migrate-existing-database-to-snapshot 默认情况下，不允许对数据库运行快照/开发服务器，该服务器之前迁移到了一些官方发布的服务器版本。 尝试执行此操作时，这表示您正在尝试针对生产数据库运行开发服务器，这可能导致数据丢失或损坏数据，也不允许升级。如果确实如此，您可以使用此选项，在明确切换为 true 时，该选项允许针对生产数据库使用每天/开发服务器。这个选项只建议在开发环境中使用，不应在生产环境中使用。 CLI: **--spi-datastore-legacy-allow-migrate-existing-database-to-snapshot** Env: **KC_SPI_DATASTORE_LEGACY_ALLOW_MIGRATE_EXISTING_DATABASE_TO_SNAPSHOT**	**true**,**false** （默认）

22.10. DBLOCK

22.10.1. jpa

	value
spi-dblock-jpa-lock-wait-timeout 等待释放数据库锁定时等待的最长时间。 CLI: **--spi-dblock-jpa-lock-wait-timeout** Env: **KC_SPI_DBLOCK_LOCK_WAIT_TIMEOUT**	any **int**

22.11. EVENTS-LISTENER

22.11.1. email

	value
spi-events-listener-email-exclude-events 以逗号分隔的事件列表，这些事件不应通过电子邮件发送到用户帐户。 CLI: **--spi-events-listener-email-exclude-events** Env: **KC_SPI_EVENTS_LISTENER_EMAIL_EXCLUDE_EVENTS**	**authreqid_to_token**,**authreqid_to_token_error**,**client_delete**,**client_delete_error**,**client_info**,**client_info_error**,**client_initiated_account_linking**,**client_initiated_account_linking_error**, client_login ,**client_login** _error ,**client_login_error**

	value
	,**client_ register** , **client_register_ error** , **client_** update, **client_update_** error, **code_to_** token, **code_to_token_** error, **custom_required_** action, **custom_required_action_** error, **delete_** account, **delete_account_** error, **execute_action_** token, **execute_action_token_** error, **execute** _actions, **execute_actions** _error, **federated_identity** _link, **federated_identity_link** _error ,**federated_identity_override_link**,**federated_identity_override_link_error**, grant_consent, **grant_consent_error** , **identity_provider_first_login**,**identity_provider_first_login_error**,**identity_provider_link_account**,**identity_provider_link_account_error**,**identity_provider_login**,**identity_provider_login_error**,**identity_provider_post_login**,**identity_provider_post_login_error**,**identity_provider_response**,**identity_provider_response_error**,**identity_provider_retrieve_token**,**identity_provider_retrieve_token_error** , impersonate ,**impersonate_error**, introspection_token,

	introspection_token value **_error**, invalid_**signature ,invalid _signature** _error , invite_**org ,invite _org** _error**,** login , login_error**, logout**,**logout_error**,**oauth2_device_auth**, oauth2_device_auth**oauth2_device_auth_error**, oauth2_device_code_**to_token ,oauth2_device_code_to_token_error** ,**oauth2_device_verify_user_code**,**oauth2_device_verify_user_code_error**, oauth2_extension_grant ,**oauth2_extension_grant** _error , oauth2_**device_verify_user_code_error ,** oauth2_extension_grant , oauth2_device_auth_code_error**permission_token**,**permission_token_error**,**pushed_authorization_request**,**pushed_authorization_request_error**,**refresh_token**,**refresh_token_error**, register_error ,**register_node**,**register_node_error**,**remove_credential**,**remove_credential_error**,**remove_federated_identity**, remove_federated_identity ,**remove_federated_identity_error**,**remove_totp**,**remove_totp_error**,**reset_password_error ,** **restart_password** _error ,**restart_authentication** _error

	value
	,**revoke_grant**,**revoke_grant_error**,**send_identity_provider_link**, **send_identity_provider_link_error** ,**send_reset_password**, send_**reset_password_error**,**send_verify_email** _email ,**send_verify_email_error** ,**token_exchange**,**token_exchange_error**,**unregister_node** _error ,**update_consent**,**update_consent_error**,**update_credential**, update_credential ,**update_credential_error**,**update_email**,**update_email_error**,**update_password**,**update_password_error**,**update_profile**,**update_profile_error**,**update_totp**,**update_totp_error**,**user_disabled_by_permanent_lockout**, **user_disabled_by_permanent_lockout_error**,**user_disabled_by_temporary_lockout**,**user_disabled_by_temporary_lockout_error**,**user_info_request** ,**user_info_request_error**, validate_access_token,**validate_access_token_error**, verify_**email** , **verify_email_error**, verify_profile ,**verify_profile_error**
spi-events-listener-email-include-events 以逗号分隔的事件列表，应该通过电子邮件发送给用户帐户。 CLI: **--spi-events-listener-email-include-events** Env: **KC_SPI_EVENTS_LISTENER_EMAIL_INCLUDE_EVENTS**	**authreqid_to_token**,**authreqid_to_token_error**,**client_delete**,**client_delete_error**,**client_info**,**client_info_error**,**client_initiated_ac**

	value
	count_linking,**client_initiated_account_linking_error**, client_login,**client_login** _error,**client_login_error**,**client_ register**, **client_register_ error**, **client_** update, **client_update_** error, **code_to_** token, **code_to_token_** error, **custom_required_** action, **custom_required_action_** error, **delete_** account, **delete_account_** error, **execute_action_** token, **execute_action_token_** error, **execute_actions**, **execute_actions** _error, **federated_identity** _link, **federated_identity_link** _error,**federated_identity_override_link**,**federated_identity_override_link_error**, grant_consent, **grant_consent_error** , **identity_provider_first_login**,**identity_provider_first_login_error**,**identity_provider_link_account**,**identity_provider_link_account_error**,**identity_provider_login**,**identity_provider_login_error**,**identity_provider_post_login**,**identity_provider_post_login_error**,**identity_provider_response**,**identity_provider_response_error**,**identity_provider_retrieve_token**,**ide**

	value
	ntity_provider_retrieve_token_error , impersonate ,**impersonate_error**, introspection_token**, introspection_token _error**, invalid_**signature ,invalid _signature** _error , invite_**org ,invite _org** _error**,** login , login_error**, logout**,**logout_error**,**oauth2_device_auth**, oauth2_device_auth**oauth2_device_auth_error**, oauth2_device_code_**to _token ,oauth2_device_ code_to_token_error** ,**oauth2_device_verify_user_code**,**oauth2_device_verify_user_code_error**, oauth2_extension_grant ,**oauth2_extension_grant** _error , oauth2_**device_verify _user_code_error ,** oauth2_extension_grant , oauth2_device_auth_code_error**permission_token**,**permission_token_error**,**pushed_authorization_request**,**pushed_authorization _request_error**,**refresh_token**,**refresh_token_error**, register_error ,**register_node**,**register_node_error**,**remove_credential**,**remove _credential_error**,**remove_federated_identity**, remove_federated_identity ,**remove_federated_identity_error**,**remove_totp**,**remove_totp_error**,**reset_password**

	value
	_error , **restart_password** _error ,**restart_authentication** _error ,**revoke_grant**,**revoke_grant_error**,**send_identity_provider_link**, **send_identity_provider_link_error** ,**send_reset_password**, send_**reset_password_error**,**send_verify_email** _email ,**send_verify_email_error** **,token_exchange**,**token_exchange_error**,**unregister_node** _error ,**update_consent**,**update_consent_error**,**update_credential**, update_credential ,**update_credential_error**,**update_email**,**update_email_error**,**update_password**,**update_password_error**,**update_profile**,**update_profile_error**,**update_totp**,**update_totp_error**,**user_disabled_by_permanent_lockout**, **user_disabled_by_permanent_lockout_error**,**user_disabled_by_temporary_lockout**,**user_disabled_by_temporary_lockout_error**,**user_info_request ,user_info_request_error**, validate_access_token**,validate_access_token_error**, verify_**email , verify_email_error**, verify_profile ,**verify_profile_error**

22.11.2. jboss-logging

	value
spi-events-listener-jboss-logging-error-level 错误消息的日志级别。 CLI: **--spi-events-listener-jboss-logging-error-level** Env: **KC_SPI_EVENTS_LISTENER_JBOSS_LOGGING_ERROR_LEVEL**	**debug**,**error**,**fatal**,**info**,**trace**,**warn** （默认）
spi-events-listener-jboss-logging-include-representation 当 "true" 带有 JSON admin 对象的 "representation" 字段也会添加到消息中。 该域还应配置为包含 admin 事件的表示。 CLI: **--spi-events-listener-jboss-logging-include-representation** Env: **KC_SPI_EVENTS_LISTENER_JBOSS_LOGGING_INCLUDE_REPRESENTATION**	**true**,**false** （默认）
spi-events-listener-jboss-logging-quotes 对值使用的引号，它应该是 " 或 ' 中的一个字符。 如果不需要引号，请使用 "none"。 CLI: **--spi-events-listener-jboss-logging- quotes** Env: **KC_SPI_EVENTS_LISTENER_JBOSS_LOGGING_QUOTES**	**"** （默认）或 **任何字符串**
spi-events-listener-jboss-logging-sanitize 如果为 true，则会清理日志消息以避免出现换行符。 如果为 false 信息，则不会被清理。 CLI: **--spi-events-listener-jboss-logging-sanitize** Env: **KC_SPI_EVENTS_LISTENER_JBOSS_LOGGING_SANITIZE**	**true** （默认），**false**
spi-events-listener-jboss-logging-success-level 成功消息的日志级别。 CLI: **--spi-events-listener-jboss-logging-success-level** Env: **KC_SPI_EVENTS_LISTENER_JBOSS_LOGGING_SUCCESS_LEVEL**	**debug** （默认）、**error**,**fatal**,**info**,**trace**,**warn**

22.12. EXPORT

22.12.1. dir

	value
spi-export-dir-dir 要导出到的目录 CLI: **--spi-export-dir-dir** Env: **KC_SPI_EXPORT_DIR_DIR**	**任何字符串**
spi-export-dir-realm-name 要导出的 realm CLI: **--spi-export-dir-realm-name** Env: **KC_SPI_EXPORT_DIR_REALM_NAME**	**任何字符串**
spi-export-dir-users-export-strategy 用户导出策略 CLI: **--spi-export-dir-users-export-strategy** Env: **KC_SPI_EXPORT_DIR_USERS_EXPORT_STRATEGY**	**DIFFERENT_FILES**（默认）或 **任何字符串**
spi-export-dir-users-per-file 每个导出的文件的用户 CLI: **--spi-export-dir-users-per-file** Env: **KC_SPI_EXPORT_DIR_USERS_PER_FILE**	**50**（默认）或任何 **int**

22.12.2. single-file

	value
spi-export-single-file-file 要导出到的文件 CLI: **--spi-export-single-file-file** Env: **KC_SPI_EXPORT_SINGLE_FILE_FILE**	**任何字符串**
spi-export-single-file-realm-name 要导出的 realm CLI: **--spi-export-single-file-realm-name** Env: **KC_SPI_EXPORT_SINGLE_FILE_REALM_NAME**	**任何字符串**

22.13. GROUP

22.13.1. jpa

	value
spi-group-jpa-escape-slashes-in-group-path 如果为 true 斜杠/，则当转换为路径时，使用字符 ~ 进行转义。 CLI: **--spi-group-jpa-escape-slashes-in-group-path** Env: **KC_SPI_GROUP_ESCAPE_SLASHES_IN_GROUP_PATH**	**true**,**false** （默认）
spi-group-jpa-searchable-attributes 客户端属性搜索中允许以逗号分开的属性列表。 CLI: **--spi-group-jpa-searchable-attributes** Env: **KC_SPI_GROUP_VolGroup_SEARCHABLE_ATTRIBUTES**	**任何字符串**

22.14. IMPORT

22.14.1. dir

	value
spi-import-dir-dir 要从中导入的目录 CLI: **--spi-import-dir** Env: **KC_SPI_IMPORT_DIR_DIR**	**任何字符串**
spi-import-dir-realm-name 要导出的 realm CLI: **--spi-import-dir-realm-name** Env: **KC_SPI_IMPORT_DIR_REALM_NAME**	**任何字符串**
spi-import-dir-strategy 导入策略：IGNORE_EXISTING, OVERWRITE_EXISTING CLI: **--spi-import-dir-strategy** Env: **KC_SPI_IMPORT_DIR_STRATEGY**	**任何字符串**

22.14.2. single-file

	value
spi-import-single-file-file 要从中导入的文件 CLI: **--spi-import-single-file-file** Env: **KC_SPI_IMPORT_SINGLE_FILE_FILE**	**任何字符串**
spi-import-single-file-realm-name 要导出的 realm CLI: **--spi-import-single-file-realm-name** Env: **KC_SPI_IMPORT_SINGLE_FILE_REALM_NAME**	**任何字符串**
spi-import-single-file-strategy 导入策略：IGNORE_EXISTING, OVERWRITE_EXISTING CLI: **--spi-import-single-file-strategy** Env: **KC_SPI_IMPORT_SINGLE_FILE_STRATEGY**	**任何字符串**

22.15. LOAD-BALANCER-CHECK

22.15.1. remote

	value
spi-load-balancer-check-remote-poll-interval 远程缓存轮询间隔（以毫秒为单位），以实现连接可用性 CLI: **--spi-load-balancer-check-remote-poll-interval** Env: **KC_SPI_LOAD_BALANCER_CHECK_REMOTE_POLL_INTERVAL**	**5000**（默认）或任何 **int**

22.16. LOGIN-PROTOCOL

22.16.1. openid-connect

	value
spi-login-protocol-openid-connect-add-req-params-fail-fast 如果发送到 OIDC 身份验证请求的参数满足一些标准 OIDC 参数的限制，则是否应强制执行 fail-fast 策略。 如果为 false，则所有未满足配置的额外请求参数都会被静默忽略。如果为 true，则会引发异常，并且不允许 OIDC 身份验证请求。 CLI: **--spi-login-protocol-openid-connect-add-req-params-fail-fast** Env: **KC_SPI_LOGIN_PROTOCOL_OPENID_CONNECT_ADD_REQ_PARAMS_FAIL_FAST**	**true**,**false** （默认）
spi-login-protocol-openid-connect-add-req-params-max-number 发送到 OIDC 身份验证请求的最大额外请求参数数。 因为 'additional request parameter' 意味着一些自定义参数没有直接被视为标准的 OIDC/OAuth2 协议参数。其他参数可能很有用，例如将自定义声明添加到 OIDC 令牌（如果也配置了特定协议映射程序）。 CLI: **--spi-login-protocol-openid-connect-add-req-params-max-number** Env: **KC_SPI_LOGIN_PROTOCOL_OPENID_CONNECT_ADD_REQ_PARAMS_MAX_NUMBER**	**5** （默认）或任何 **int**
spi-login-protocol-openid-connect-add-req-params-max-overall-size 所有其他请求参数值的最大大小。 如需了解更多请求参数的详情，请参阅 **add-req-params-max-number** CLI: **--spi-login-protocol-openid-connect-add-req-params-max-overall-size** Env: **KC_SPI_LOGIN_PROTOCOL_OPENID_CONNECT_ADD_REQ_PARAMS_MAX_OVERALL_SIZE**	**2147483647** （默认）或任何 **int**
spi-login-protocol-openid-connect-add-req-params-max-size 有关附加请求参数的详情，请参阅 **add-req-params-max-number** 的最大大小 CLI: **--spi-login-protocol-openid-connect-add-req-params-max-size** Env: **KC_SPI_LOGIN_PROTOCOL_OPENID_CONNECT_ADD_REQ_PARAMS_MAX_SIZE**	**2000** （默认）或任何 **int**

	value
spi-login-protocol-openid-connect-req-params-default-max-size 发送到 OIDC 身份验证请求的标准 OIDC 参数的最大默认长度。 这适用于大多数标准参数，如 **状态**、**非** 等。例外是 **login_hint** 参数，其长度最大为 255 个字符。 CLI: **--spi-login-protocol-openid-connect-req-params-default-max-size** Env: **KC_SPI_LOGIN_PROTOCOL_OPENID_CONNECT_REQ_PARAMS_DEFAULT_MAX_SIZE**	**4000**（默认）或任何 **int**
spi-login-protocol-openid-connect-req-params-max-size—login_hint 指定参数的标准 OIDC 身份验证请求参数的最大长度。 如果某些标准 OIDC 参数的限制与 **req-params-default-max-size** 不同，则很有用。需要在此前缀后添加参数的名称到配置中。在本例中，使用了 **login_hint** 参数，但这种格式支持任何已知的标准 OIDC/OAuth2 参数。 CLI: **--spi-login-protocol-openid-connect-req-params-max-size-login_hint** Env: **KC_SPI_LOGIN_PROTOCOL_OPENID_CONNECT_REQ_PARAMS_MAX_SIZE__LOGIN_HINT**	any **int**

22.17. LOGIN-FAILURE

22.17.1. remote

	value
spi-login-failure-remote-max-retries 发生错误时的最大重试次数。 值为零或更少禁用任何重试。 CLI: **--spi-login-failure-remote-max-retries** Env: **KC_SPI_LOGIN_FAILURE_REMOTE_MAX_RETRIES**	**10**（默认）或任何 **int**
spi-login-failure-remote-retry-base-time 基础后端时间（以毫秒为单位）。 CLI: **--spi-login-failure-remote-retry-base-time** Env: **KC_SPI_LOGIN_FAILURE_REMOTE_RETRY_BASE_TIME**	**10**（默认）或任何 **int**

22.18. PASSWORD-HASHING

22.18.1. argon2

	value
spi-password-hashing-argon2-cpu-cores 用于哈希的最大并行 CPU 内核 CLI: **--spi-password-hashing-argon2-cpu-cores** Env: **KC_SPI_PASSWORD_HASHING_ARGON2_CPU_CORES**	any **int**
spi-password-hashing-argon2-hash-length 哈希长度 CLI: **--spi-password-hashing-argon2-hash-length** Env: **KC_SPI_PASSWORD_HASHING_ARGON2_HASH_LENGTH**	**32** （默认）或任何 **int**
spi-password-hashing-argon2-iterations iterations CLI: **--spi-password-hashing-argon2-iterations** Env: **KC_SPI_PASSWORD_HASHING_ARGON2_ITERATIONS**	**5** （默认）或任何 **int**
spi-password-hashing-argon2-memory 内存大小(KB) CLI: **--spi-password-hashing-argon2-memory** Env: **KC_SPI_PASSWORD_HASHING_ARGON2_MEMORY**	**7168** （默认）或任何 **int**
spi-password-hashing-argon2-parallelism parallelism CLI: **--spi-password-hashing-argon2-parallelism** Env: **KC_SPI_PASSWORD_HASHING_ARGON2_PARALLELISM**	**1 (** 默认)或任何 **int**
spi-password-hashing-argon2-type 类型 CLI: **--spi-password-hashing-argon2-type** Env: **KC_SPI_PASSWORD_HASHING_ARGON2_TYPE**	**id** （默认）、**d**、**i**

	value
spi-password-hashing-argon2-version Version CLI: **--spi-password-hashing-argon2-version** Env: **KC_SPI_PASSWORD_HASHING_ARGON2_VERSION**	**1.3**（默认）、**1.0**

22.19. PUBLIC-KEY-STORAGE

22.19.1. Infinispan

	value
spi-public-key-storage-infinispan-max-cache-time 当密钥通过所有密钥方法检索时，密钥的最大间隔（以秒为单位）。 当检索条目的所有密钥时，无法检测是否缺少密钥（例如，使用 ID 检索密钥时不同）。在这种情况下，这个选项会强制刷新时间。默认 24 小时。 CLI: **--spi-public-key-storage-infinispan-max-cache-time** Env: **KC_SPI_PUBLIC_KEY_STORAGE_INFINISPAN_MAX_CACHE_TIME**	**86400**（默认）或任何 **int**
spi-public-key-storage-infinispan-min-time-between-requests 两个请求之间用于检索新公钥的最小间隔（以秒为单位）。 请求一个密钥且没有找到时，服务器将始终尝试下载新公钥。但是，如果之前的刷新时间少于 10 秒，它将避免下载（默认）。这个行为用于避免对外部密钥端点的 DoS 攻击。 CLI: **--spi-public-key-storage-infinispan-min-time-between-requests** Env: **KC_SPI_PUBLIC_KEY_STORAGE_INFINISPAN_MIN_TIME_BETWEEN_REQUESTS**	**10**（默认）或任何 **int**

22.20. REQUIRED-ACTION

22.20.1. UPDATE_PASSWORD

	value
spi-required-action-UPDATE_PASSWORD-max_auth_age 在需要用户重新验证前，在最后一次验证后配置此操作的时间（以秒为单位）。 当请求中有 kc_action 参数时，此参数仅在 AIA 上下文中使用，当用户自己在帐户控制台中更新其密码时，该参数适用于实例。当在域中使用 'Maximum Authentication Age' 密码策略时，它的值优先于此处配置的值。 CLI: **--spi-required-action-UPDATE_PASSWORD-max_auth_age** Env: **KC_SPI_REQUIRED_ACTION_UPDATE_PASSWORD_MAX_AUTH_AGE**	**300**（默认）或**任何字符串**

22.21. RESOURCE-ENCODING

22.21.1. gzip

	value
spi-resource-encoding-gzip-excluded-content-types 从编码中排除的内容类型列表。 CLI: **--spi-resource-encoding-gzip-excluded-content-types** Env: **KC_SPI_RESOURCE_ENCODING_GZIP_EXCLUDED_CONTENT_TYPES**	**image/png** **image/jpeg**（默认）或**任何字符串**

22.22. SECURITY-PROFILE

22.22.1. default

	value
spi-security-profile-default-name 要使用的安全配置文件的名称。 文件 **名称**.json 在 classapth 和**conf** 安装文件夹中搜索。 CLI: **--spi-security-profile-default-name** Env: **KC_SPI_SECURITY_PROFILE_DEFAULT_NAME**	**任何字符串**

22.23. SINGLE-USE-OBJECT

22.23.1. Infinispan

	value
spi-single-use-object-infinispan-persist-revoked-tokens 如果在重启后存储已撤销的令牌 CLI: **--spi-single-use-object-infinispan-persist-revoked-tokens** Env: **KC_SPI_SINGLE_USE_OBJECT_INFINISPAN_PERSIST_REVOKED_TOKENS**	**true**（默认），**false**

22.23.2. remote

	value
spi-single-use-object-remote-persist-revoked-tokens 如果在重启后存储已撤销的令牌 CLI: **--spi-single-use-object-remote-persist-revoked-tokens** Env: **KC_SPI_SINGLE_USE_OBJECT_REMOTE_PERSIST_REVOKED_TOKENS**	**true**（默认），**false**

22.24. STICKY-SESSION-ENCODER

22.24.1. Infinispan

	value
spi-sticky-session-encoder-infinispan-should-attach-route 如果路由应该附加到 Cookie，以反映拥有特定会话的节点。 CLI: **--spi-sticky-session-encoder-infinispan-should-attach-route** Env: **KC_SPI_STICKY_SESSION_ENCODER_INFINISPAN_SHOULD_ATTACH_ROUTE**	**true**（默认），**false**

22.24.2. remote

	value
spi-sticky-session-encoder-remote-should-attach-route 如果路由应该附加到 Cookie，以反映拥有特定会话的节点。 CLI: **--spi-sticky-session-encoder-remote-should-attach-route** Env: **KC_SPI_STICKY_SESSION_ENCODER_REMOTE_SHOULD_ATTACH_ROUTE**	**true**（默认），**false**

22.25. TRUSTSTORE

22.25.1. file

	value
spi-truststore-file-file DEPRECATED ：从中读取证书以验证 TLS 连接的信任存储的文件路径。 CLI: **--spi-truststore-file-file** Env: **KC_SPI_TRUSTSTORE_FILE_FILE**	**任何字符串**
spi-truststore-file-hostname-verification-policy DEPRECATED ：主机名验证策略。 CLI: **--spi-truststore-file-hostname-verification-policy** Env: **KC_SPI_TRUSTSTORE_FILE_HOSTNAME_VERIFICATION_POLICY**	**ANY**,**WILDCARD**,**STRICT**,**DEFAULT**（默认）
spi-truststore-file-password DEPRECATED ：信任存储密码。 CLI: **--spi-truststore-file-password** Env: **KC_SPI_TRUSTSTORE_FILE_PASSWORD**	**任何字符串**
spi-truststore-file-type DEPRECATED ：信任存储的类型。 如果没有提供，则会根据信任存储文件扩展或平台默认类型检测到类型。 CLI: **--spi-truststore-file-type** Env: **KC_SPI_TRUSTSTORE_FILE_TYPE**	**任何字符串**

22.26. USER-PROFILE

22.26.1. declarative-user-profile

	value
spi-user-profile-declarative-user-profile-admin-read-only-attributes 正则表达式数组来识别应被视为只读的字段，以便管理员无法更改它们。 CLI: **--spi-user-profile-declarative-user-profile-admin-read-only-attributes** Env: **KC_SPI_USER_PROFILE_DECLARATIVE_USER_PROFILE_ADMIN_READ_ONLY_ATTRIBUTES**	任何 **MultivaluedString**
spi-user-profile-declarative-user-profile-max-email-local-part-length 要设置用户配置文件最大电子邮件本地部分长度 CLI: **--spi-user-profile-declarative-user-profile-max-email-local-part-length** Env: **KC_SPI_USER_PROFILE_DECLARATIVE_USER_PROFILE_MAX_EMAIL_LOCAL_PART_LENGTH**	**任何字符串**
spi-user-profile-declarative-user-profile-read-only-attributes 正则表达式数组来识别应被视为只读的字段，以便用户无法更改它们。 CLI: **--spi-user-profile-declarative-user-profile-read-only-attributes** Env: **KC_SPI_USER_PROFILE_DECLARATIVE_USER_PROFILE_READ_ONLY_ATTRIBUTES**	任何 **MultivaluedString**

22.27. USER-SESSIONS

22.27.1. Infinispan

	value
spi-user-sessions-infinispan-max-batch-size 批处理大小的最大大小（仅适用于持久会话） CLI: **--spi-user-sessions-infinispan-max-batch-size** Env: **KC_SPI_USER_SESSIONS_INFINISPAN_MAX_BATCH_SIZE**	**4 (** 默认)或任何 **int**

	value
spi-user-sessions-infinispan-offline-client-session-cache-entry-lifespan-override 覆盖离线客户端会话应保存在内存中的时间（以秒为单位） CLI: **--spi-user-sessions-infinispan-offline-client-session-cache-entry-lifespan-override** Env: **KC_SPI_USER_SESSIONS_INFINISPAN_OFFLINE_CLIENT_SESSION_CACHE_ENTRY_LIFESPAN_OVERRIDE**	any **int**
spi-user-sessions-infinispan-offline-session-cache-entry-lifespan-override 覆盖离线用户会话应保存在内存中的时间（以秒为单位） CLI: **--spi-user-sessions-infinispan-offline-session-cache-entry-lifespan-override** Env: **KC_SPI_USER_SESSIONS_INFINISPAN_OFFLINE_SESSION_CACHE_ENTRY_LIFESPAN_OVERRIDE**	any **int**
spi-user-sessions-infinispan-use-caches 启用或禁用缓存。 默认启用，除非只使用外部远程缓存 CLI: **--spi-user-sessions-infinispan-use-caches** Env: **KC_SPI_USER_SESSIONS_INFINISPAN_USE_CACHES**	**true**, **false**

22.27.2. remote

	value
spi-user-sessions-remote-batch-size 从远程缓存流传输会话时的批处理大小 CLI: **--spi-user-sessions-remote-batch-size** Env: **KC_SPI_USER_SESSIONS_REMOTE_BATCH_SIZE**	**1024**（默认）或任何 **int**

	value
spi-user-sessions-remote-max-retries 发生错误时的最大重试次数。 值为零或更少禁用任何重试。 CLI: **--spi-user-sessions-remote-max-retries** Env: **KC_SPI_USER_SESSIONS_REMOTE_MAX_RETRIES**	**10**（默认）或任何 **int**
spi-user-sessions-remote-retry-base-time 基础后端时间（以毫秒为单位）。 CLI: **--spi-user-sessions-remote-retry-base-time** Env: **KC_SPI_USER_SESSIONS_REMOTE_RETRY_BASE_TIME**	**10**（默认）或任何 **int**

22.28. 已知的

22.28.1. oauth-authorization-server

	value
spi-well-known-oauth-authorization-server-include-client-scopes 如果应使用客户端范围来计算支持的范围列表。 CLI: **--spi-well-known-oauth-authorization-server-include-client-scopes** Env: **KC_SPI_WELL_KNOWN_OAUTH_AUTHORIZATION_SERVER_INCLUDE_CLIENT_SCOPES**	**true**（默认），**false**
spi-well-known-oauth-authorization-server-openid-configuration-override 应该从中加载元数据的文件路径。 您可以使用绝对路径；或者，如果文件位于服务器类路径中，请使用 **classpath:** 前缀来加载 classpathpath 中的文件。 CLI: **--spi-well-known-oauth-authorization-server-openid-configuration-override** Env: **KC_SPI_WELL_KNOWN_OAUTH_AUTHORIZATION_SERVER_OPENID_CONFIGURATION_OVERRIDE**	**任何字符串**

22.28.2. openid-configuration

	value
spi-well-known-openid-configuration-include-client-scopes 如果应使用客户端范围来计算支持的范围列表。 CLI: **--spi-well-known-openid-configuration-include-client-scopes** Env: **KC_SPI_WELL_KNOWN_OPENID_CONFIGURATION_INCLUDE_CLIENT_SCOPES**	**true**（默认），**false**
spi-well-known-openid-configuration-openid-configuration-override 应该从中加载元数据的文件路径。 您可以使用绝对路径；或者，如果文件位于服务器类路径中，请使用 **classpath:** 前缀来加载 classpathpath 中的文件。 CLI: **--spi-well-known-openid-configuration-openid-configuration-override** Env: **KC_SPI_WELL_KNOWN_OPENID_CONFIGURATION_OPENID_CONFIGURATION_OVERRIDE**	**任何字符串**

第 23 章 检查是否可以滚动更新

执行 update compatibility 命令，以检查红帽构建的 Keycloak 支持对部署的更改进行滚动更新。

使用 update compatibility 命令，在启用或禁用功能或更改红帽构建的 Keycloak 版本、配置或供应商及其时，使用滚动更新策略来更新部署。结果显示是否可以进行滚动更新，还是需要重新创建的更新。

在当前版本中，显示红帽构建的 Keycloak 版本和新版本可能会进行滚动更新。红帽构建的 Keycloak 的未来版本可能会改变该行为，以使用配置中的附加信息、镜像和版本，以确定是否可以进行滚动更新。

这可以完全脚本，因此您的更新过程可以使用该信息根据执行的更改执行滚动或重新创建策略。它还对 GitOps 友好，因为它允许将之前配置的元数据存储在文件中。在 CI/CD 管道中将此文件与新配置一起使用，以确定是否可以进行滚动更新或是否需要重新创建更新。

如果您使用红帽构建的 Keycloak Operator，继续使用滚动更新章节和 Auto 策略来获得更详细的停机时间。

23.1. 支持的更新策略

滚动更新

在本指南中，滚动更新是一个更新，您的部署可在不停机的情况下执行，由至少两个节点组成。逐一更新您的红帽 Keycloak 构建；关闭其中一个旧部署节点并启动新的部署节点。等待新节点的启动探测返回成功，然后继续下一个红帽构建的 Keycloak 节点。如需有关如何启用和使用启动探测的详细信息，请参阅使用健康检查跟踪实例状态 的章节。

重新创建更新

重新创建更新与零停机时间不兼容，需要应用停机时间。在使用新版本启动节点前，关闭运行旧版本集群的所有节点。

23.2. 为更新的配置确定更新策略

要确定是否可以滚动更新，请运行 update 兼容性命令：

1. 使用旧配置生成所需的元数据。

2. 使用新配置检查元数据以确定更新策略。

警告

此命令目前仅提供有限的功能。目前，只考虑红帽构建的 Keycloak 版本和嵌入式 Infinispan，以确定是否可以进行滚动更新。如果这些不动，则报告有可能进行滚动更新。

当前版本还没有验证配置更改，并假定所有配置更改都有资格进行滚动更新。这同样适用于自定义扩展及其更改。

使用此选项时的良好用例是，当您希望在更改红帽构建的 Keycloak 主题或自定义扩展时进行滚动更新，且仅在红帽构建的 Keycloak 版本更改时进行重新创建，它还不允许滚动更新。

虽然这些命令的用户应该知道目前存在的限制，但它们不应依赖于元数据文件的内部行为或结构。相反，它们应该只依赖 check 命令的退出代码，以便以后对内部逻辑的改进中受益，以确定何时能够进行滚动更新。

23.2.1. 生成元数据

要生成元数据，请使用相同的红帽构建的 Keycloak 版本和配置选项执行以下命令：

从当前部署生成并保存元数据。

```
bin/kc.[sh|bat] update-compatibility metadata --file=/path/to/file.json
```

此命令接受 start 命令使用的所有选项。命令在控制台中以 JSON 格式显示元数据，用于调试目的。--

file 参数允许您将元数据保存到文件中。将此文件与后续的 check 命令一起使用。

警告

在运行上述命令时，请确保包括所有配置选项（无论是通过环境变量或 CLI 参数设置）。

省略任何配置选项会导致元数据不完整，并可能导致下一步报告的结果。

23.2.2. 检查元数据

这个命令检查上一命令生成的元数据，并将其与当前配置和红帽构建的 Keycloak 版本进行比较。如果您要升级到新的红帽构建的 Keycloak 版本，则必须使用新版本执行这个命令。

检查之前部署的元数据。

```
bin/kc.[sh|bat] update-compatibility check --file=/path/to/file.json
```

警告

- 在运行此命令时，请确保包含通过环境变量或 CLI 参数设置的所有配置选项。
- 验证是否使用正确的红帽构建的 Keycloak 版本。

无法满足这些要求会导致结果不正确。

命令将结果输出到控制台。例如，如果可能滚动更新，它会显示：

滚动更新可能消息

```
[OK] Rolling Update is available.
```

如果没有进行滚动更新，命令会提供有关不兼容的详细信息：

滚动更新不可能的消息

```
[keycloak] Rolling Update is not available. 'keycloak.version' is incompatible: 26.2.0 -> 26.2.1
1
```

1

在本例中，Keycloak 版本 26.2.0 与版本 26.2.1 不兼容，且无法进行滚动更新。

命令退出代码

使用命令的退出代码决定自动化管道中的更新类型：

退出代码	描述
0	可以滚动更新。
1	发生意外错误（如元数据文件缺失或损坏）。
2	无效的 CLI 选项。
3	无法进行滚动更新。在应用新配置前，必须关闭部署。
4	无法进行滚动更新。禁用 **功能 滚动更新**。

23.3. 进一步阅读

Red Hat build of Keycloak Operator 使用上述功能来决定是否可以进行滚动更新。如需更多信息，请参阅 滚动更新 章节和 Auto 策略 的停机时间。