



Red Hat build of Keycloak 26.0

Getting Started Guide

Legal Notice

Copyright © 2025 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This guide helps you practice using Red Hat build of Keycloak 26.0 to evaluate it before you use it in a production environment. It includes instructions for installing and running the Red Hat build of Keycloak server in development mode, creating realms and accounts for managing users and applications, and securing a Single Page Application (SPA)

Table of Contents

CHAPTER 1. GETTING STARTED	3
1.1. BEFORE YOU START	3
1.2. DOWNLOAD RED HAT BUILD OF KEYCLOAK	3
1.3. START RED HAT BUILD OF KEYCLOAK	3
1.4. CREATE AN ADMIN USER	3
1.5. LOG IN TO THE ADMIN CONSOLE	3
1.6. CREATE A REALM	3
1.7. CREATE A USER	4
1.8. LOG IN TO THE ACCOUNT CONSOLE	6
1.9. SECURE THE FIRST APPLICATION	6
1.10. TAKING THE NEXT STEP	8
CHAPTER 2. SCALING	9
2.1. VERTICAL SCALING	9
2.1.1. Common Tuning Options	9
2.1.2. Vertical Autoscaling	9
2.2. HORIZONTAL SCALING	9
2.2.1. Horizontal Autoscaling	10

CHAPTER 1. GETTING STARTED

1.1. BEFORE YOU START

Make sure your machine or container platform can provide sufficient memory and CPU for your desired usage of Red Hat build of Keycloak. See [Concepts for sizing CPU and memory resources](#) for more on how to get started with production sizing.

Make sure you have [OpenJDK 21](#) installed.

1.2. DOWNLOAD RED HAT BUILD OF KEYCLOAK

Download Red Hat build of Keycloak from the [Red Hat website](#) and extract it.

After extracting this file, you should have a directory that is named **rhbk-26.0.15**.

1.3. START RED HAT BUILD OF KEYCLOAK

1. From a terminal, open the **rhbk-26.0.15** directory.
2. Enter the following command:

- On Linux, run:

```
bin/kc.sh start-dev
```

- On Windows, run:

```
bin\kc.bat start-dev
```

Using the **start-dev** option, you are starting Red Hat build of Keycloak in development mode. In this mode, you can try out Red Hat build of Keycloak for the first time to get it up and running quickly. This mode offers convenient defaults for developers, such as for developing a new Red Hat build of Keycloak theme.

1.4. CREATE AN ADMIN USER

Red Hat build of Keycloak has no default admin user. You need to create an admin user before you can start Keycloak.

1. Open <http://localhost:8080/>.
2. Fill in the form with your preferred username and password.

1.5. LOG IN TO THE ADMIN CONSOLE

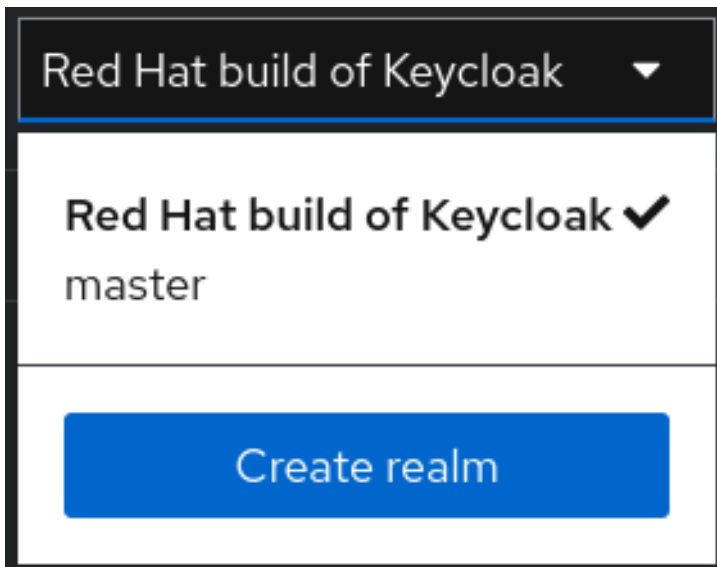
1. Go to the [Red Hat build of Keycloak Admin Console](#).
2. Log in with the username and password you created earlier.

1.6. CREATE A REALM

A realm in Red Hat build of Keycloak is equivalent to a tenant. Each realm allows an administrator to create isolated groups of applications and users. Initially, Red Hat build of Keycloak includes a single realm, called **master**. Use this realm only for managing Red Hat build of Keycloak and not for managing any applications.

Use these steps to create the first realm.

1. Open the [Red Hat build of Keycloak Admin Console](#) .
2. Click **Red Hat build of Keycloak** next to **master realm**, then click **Create Realm**.
3. Enter **myrealm** in the **Realm name** field.
4. Click **Create**.



1.7. CREATE A USER

Initially, the realm has no users. Use these steps to create a user:

1. Verify that you are still in the **myrealm** realm, which is shown above the word **Manage**.
2. Click **Users** in the left-hand menu.
3. Click **Create new user**.
4. Fill in the form with the following values:
 - **Username:** **myuser**
 - **First name:** any first name
 - **Last name:** any last name
5. Click **Create**.

Red Hat | Red Hat build of Keycloak

admin

myrealm

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Identity providers

User federation

Users > Create user

Create user

Enabled

Action

Username *

myuser

Email

Email verified ?

Off

First name

Foo

Last name

Bar

Required user actions

Select action

Groups ?

Join Groups

Create

Cancel

This user needs a password to log in. To set the initial password:

1. Click **Credentials** at the top of the page.
2. Fill in the **Set password** form with a password.
3. Toggle **Temporary** to **Off** so that the user does not need to update this password at the first login.

Details Attributes Credentials Role mapping Groups

Set password for myuser

Password *

.....

Password confirmation *

.....

Temporary ?

Off

Save

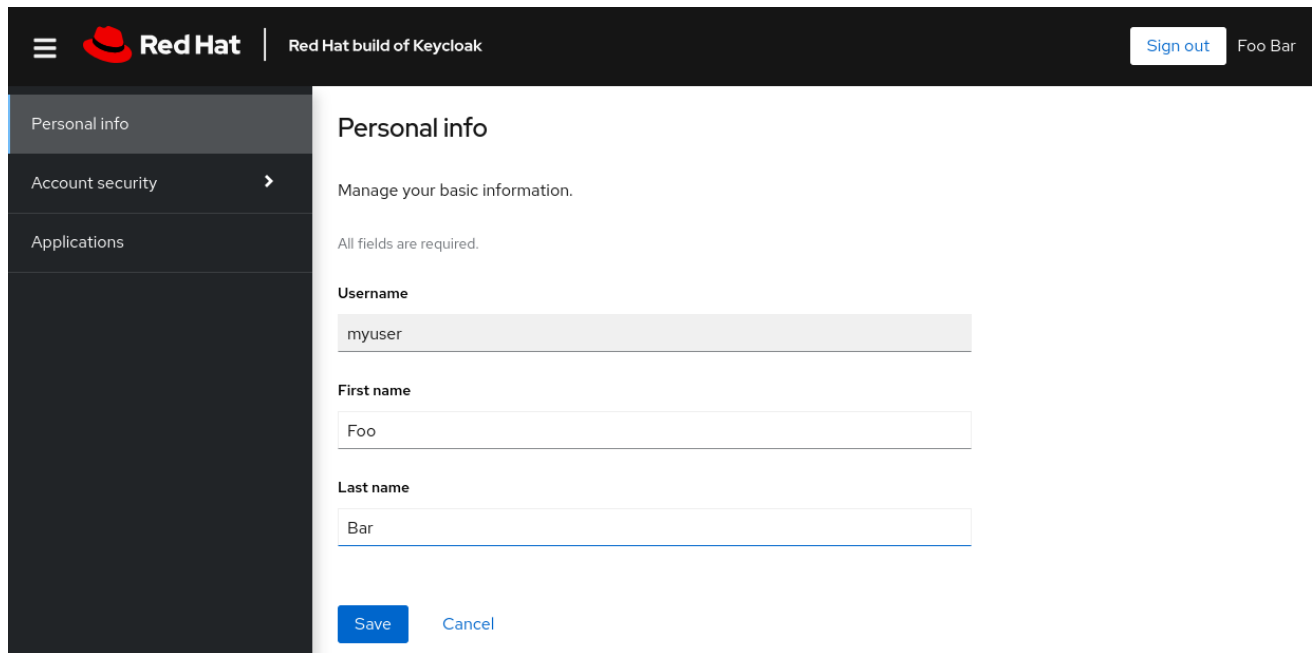
Cancel

1.8. LOG IN TO THE ACCOUNT CONSOLE

You can now log in to the Account Console to verify this user is configured correctly.

1. Open the [Red Hat build of Keycloak Account Console](#) .
2. Log in with **myuser** and the password you created earlier.

As a user in the Account Console, you can manage your account including modifying your profile, adding two-factor authentication, and including identity provider accounts.



The screenshot shows the 'Personal info' page of the Red Hat build of Keycloak Account Console. The page has a dark header with the Red Hat logo and 'Red Hat build of Keycloak' text. A 'Sign out' button and the user name 'Foo Bar' are in the top right. A left sidebar contains links for 'Personal info', 'Account security', and 'Applications'. The main content area is titled 'Personal info' and includes the instruction 'Manage your basic information.' and a note 'All fields are required.' Below this are three text input fields: 'Username' (containing 'myuser'), 'First name' (containing 'Foo'), and 'Last name' (containing 'Bar'). At the bottom are 'Save' and 'Cancel' buttons.

1.9. SECURE THE FIRST APPLICATION

To secure the first application, you start by registering the application with your Red Hat build of Keycloak instance:

1. Open the [Red Hat build of Keycloak Admin Console](#) .
2. Click the word **master** in the top-left corner, then click **myrealm**.
3. Click **Clients**.
4. Click **Create client**
5. Fill in the form with the following values:
 - **Client type:** **OpenID Connect**
 - **Client ID:** **myclient**

Red Hat | Red Hat build of Keycloak

myrealm

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Identity providers

User federation

Clients > Create client

Create client

Clients are applications and services that can request authentication of a user.

1 General Settings

Client type ⓘ OpenID Connect

Client ID * ⓘ myclient

Name ⓘ

Description ⓘ

Always display in console ⓘ ☐ Off

Next Back Cancel

6. Click **Next**
7. Confirm that **Standard flow** is enabled.
8. Click **Next**.
9. Make these changes under **Login settings**.
 - Set **Valid redirect URIs** to **https://www.keycloak.org/app/***
 - Set **Web origins** to **https://www.keycloak.org**
10. Click **Save**.

myrealm

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Identity providers

User federation

Clients > Create client

Create client

Clients are applications and services that can request authentication of a user.

- 1 General Settings
- 2 Capability config
- 3 Login settings

Root URL

Home URL

Valid redirect URIs [+ Add valid redirect URIs](#)

Valid post logout redirect URIs [+ Add valid post logout redirect URIs](#)

Web origins [+ Add web origins](#)

Save Back Cancel

To confirm the client was created successfully, you can use the SPA testing application on the [Keycloak website](https://www.keycloak.org/app/).

1. Open <https://www.keycloak.org/app/>.
2. Click **Save** to use the default configuration.
3. Click **Sign in** to authenticate to this application using the Red Hat build of Keycloak server you started earlier.

1.10. TAKING THE NEXT STEP

Before you run Red Hat build of Keycloak in production, consider the following actions:

- Switch to a production ready database such as PostgreSQL.
- Configure SSL with your own certificates.
- Switch the admin password to a more secure password.

For more information, see the [Server Configuration Guide](#).

CHAPTER 2. SCALING

After starting Red Hat build of Keycloak, consider adapting your instance to the required load using these scaling and tuning guidelines:

- minimize resource utilization
- achieve target response times
- minimize database pool contention
- resolve out of memory errors, or excessive garbage collection overhead
- provide higher availability via horizontal scaling

2.1. VERTICAL SCALING

As you monitor your Red Hat build of Keycloak workload, check to see if the CPU or memory is under or over utilized. Consult [Concepts for sizing CPU and memory resources](#) to better tune the resources available to the Java Virtual Machine (JVM).

Before increasing the amount of memory available to the JVM, in particular when experiencing an out of memory error, it is best to determine what is contributing to the increased footprint using a heap dump. Excessive response times may also indicate the HTTP work queue is too large and tuning for load shedding would be better than simply providing more memory. See the following section.

2.1.1. Common Tuning Options

Red Hat build of Keycloak automatically adjusts the number of used threads based upon how many cores you make available. Manually changing the thread count can improve overall throughput. For more details, see [Concepts for configuring thread pools](#). However, changing the thread count must be done in conjunction with other JVM resources, such as database connections; otherwise, you may be moving a bottleneck somewhere else. For more details, see [Concepts for database connection pools](#).

To limit memory utilization of queued work and to provide for load shedding, see [Concepts for configuring thread pools](#).

If you are experiencing timeouts in obtaining database connections, you should consider increasing the number of connections available. For more details, see [Concepts for database connection pools](#).

2.1.2. Vertical Autoscaling

Some platforms, such as Kubernetes, provide mechanisms to vertically autoscale. Vertical autoscaling is not recommended for Red Hat build of Keycloak if it requires restarting the server instance, which is currently the case for Java on Kubernetes. You can consider instead providing higher CPU and/or memory limits to allow your JVM to adapt within those limits as needed.

2.2. HORIZONTAL SCALING

A single Red Hat build of Keycloak instance is susceptible to availability issues. If the instance goes down, you experience a full outage until another instance comes up. By running two or more cluster members on different machines, you greatly increase the availability of Red Hat build of Keycloak.

A single JVM has a limit on how many concurrent requests it can handle. Additional server instances can provide roughly linear scaling of throughput until associated resources, such as the database or distributed caching, limit that scaling.

In general, consider allowing the Red Hat build of Keycloak Operator to handle horizontal scaling concerns. When using the Operator, set the Keycloak custom resource **spec.instances** as desired to horizontally scale. For more details, see [Deploy Red Hat build of Keycloak for HA with the Red Hat build of Keycloak Operator](#).

If you are not using the Operator, please review the following:

- Higher availability is possible if your instances are on separate machines. On Kubernetes, use Pod anti-affinity to enforce this.
- Use distributed caching; for multi-site clusters, use external caching for cluster members to share the same state. For details on the relevant configuration, see [Configuring distributed caches](#). The embedded Infinispan cache has horizontal scaling considerations including:
 - Your instances need a way to discover each other. For more information, see discovery in [Configuring distributed caches](#).
 - This cache is not optimal for clusters that span multiple availability zones, which are also called stretch clusters. For embedded Infinispan cache, work to have all instances in one availability zone. The goal is to avoid unnecessary round-trips in the communication that would amplify in the response times. On Kubernetes, use Pod affinity to enforce this grouping of Pods.
 - This cache does not gracefully handle multiple members joining or leaving concurrently. In particular, members leaving at the same time can lead to data loss. On Kubernetes, you can use a StatefulSet with the default serial handling to ensure Pods are started and stopped sequentially.

To avoid losing service availability when a whole site is unavailable, see the high availability guide for more information on a multi-site deployment. See [Multi-site deployments](#).

2.2.1. Horizontal Autoscaling

Horizontal autoscaling allows for adding or removing Red Hat build of Keycloak instances on demand. Keep in mind that startup times will not be instantaneous and that optimized images should be used to minimize the start time.

When using the embedded Infinispan cache cluster, dynamically adding or removing cluster members requires Infinispan to perform a rebalancing of the Infinispan caches, which can get expensive if many entries exist in those caches. To minimize this time we limit number of entries in session related caches to 10000 by default. Note, this optimization is possible only if **persistent-user-sessions** feature is not explicitly disabled in your configuration.

On Kubernetes, the Keycloak custom resource is scalable meaning that it can be targeted by the [built-in autoscaler](#).