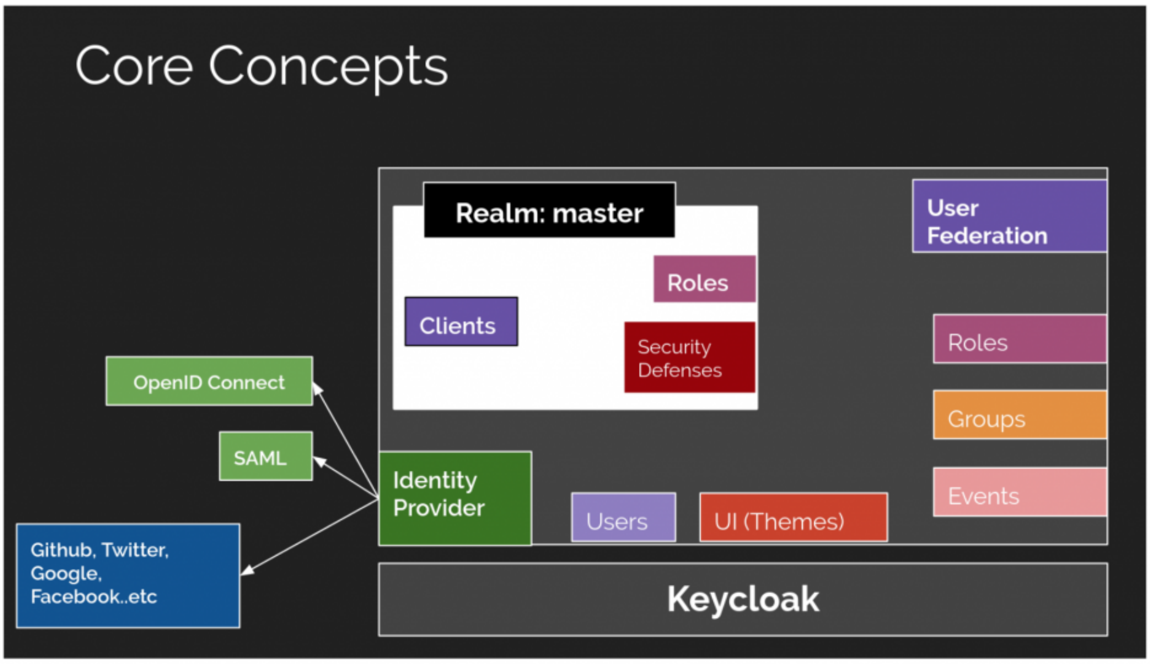
Article

Keycloak: Core concepts of open source identity and access management

December 11, 2019 Security

Abhishek Koserwal
Senior Software Engineer



Keycloak provides the flexibility to export and import configurations easily, using a single view to manage everything. Together, these technologies let you integrate front-end, mobile, and monolithic applications into a microservice architecture. In this article, we discuss the core concepts and features of Keycloak and its application integration mechanisms. You will find links to implementation details near the end.

Core concepts

Let's start with Keycloak's core concepts, as shown in Figure 1:

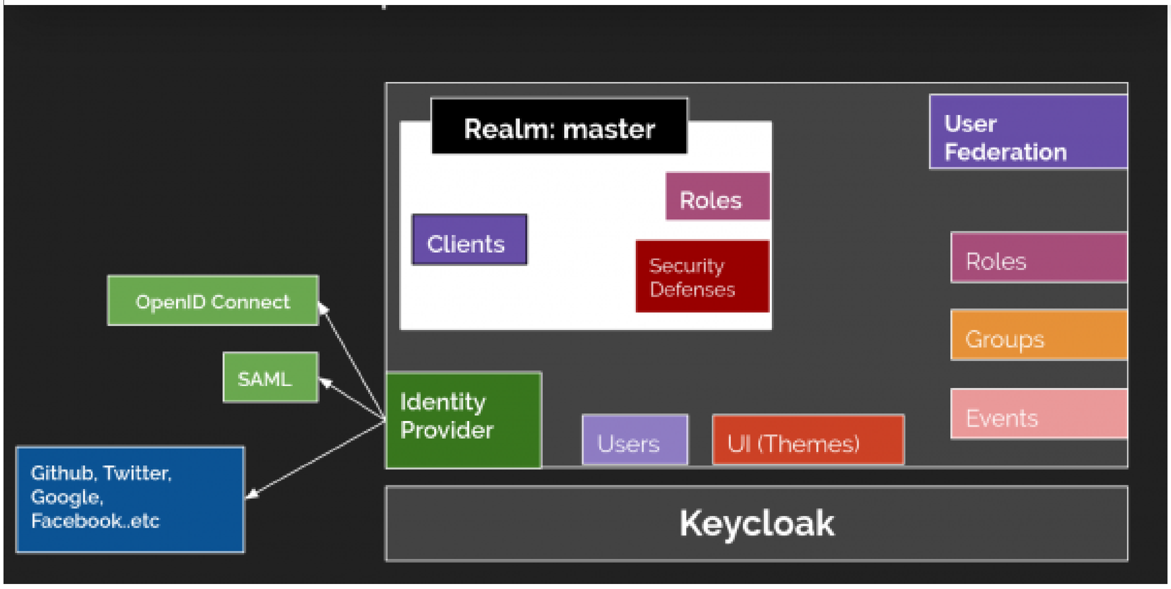


Figure 1: Keycloak's core concepts.">

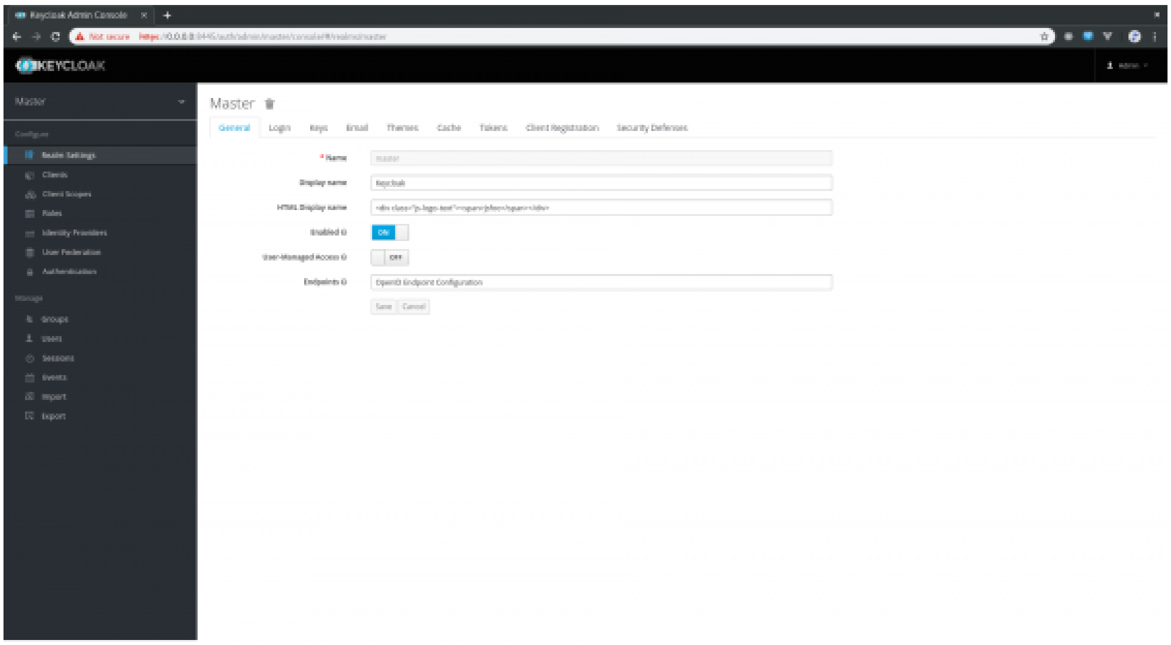
A Keycloak *realm* is like a namespace that allows you to manage all of your metadata and configurations. You can have multiple realms based on your requirements. Generally, it is recommended to avoid using the *master realm*, which is for administration purposes only.

In Figure 1, you can see the information that Keycloak lets you manage, namely:

- Clients (per application)
- Configuration management
- Custom themes (UI)
- Events
- Federation
- LDAP or Active Directory integration
- User management (users and groups)

Note: You can have one client that contains configuration information for a single application, such as the URL, protocol, and redirect URI.

Figure 2 shows how Keycloak gives you access to all of this information in a single view:



Master

* Name	master
Display name	Keycloak
HTML Display name	<div class="p-logo-text"><span>[illegible]</span></div>
Enabled	ON
User-Managed Access	OFF
Endpoints	OpenID Endpoint Configuration

Keycloak's UI offers access to many settings.

Figure 2: Keycloak's UI offers access to many settings.">

Why should you use Keycloak?

Let's take a look at why you might choose Keycloak, aside from the sheer amount of management you can accomplish within a single view.

Keycloak is reliable

Keycloak is a reliable solution, designed following standard security protocols to provide a dynamic single sign-on solution. Red Hat runs on Red Hat products, which includes single sign-on (SSO), and Red Hat trusts the upstream product Keycloak for their downstream product Red Hat SSO. Red Hat SSO handles Red Hat's entire authentication and authorization system. Additionally, Keycloak is licensed under Apache License Version 2.0 and has a strong and active open source community.

Keycloak supports standard protocols

Keycloak supports the following standard protocols:

- OAuth 2.0
- OpenID Connect
- SAML 2.0

This support means that any tool or application that supports integration with the above protocols can be plugged into with Keycloak (for example, enterprise applications like Red Hat Ansible Tower or SAP Business Intelligence Platform).

Keycloak is ready for production

As mentioned earlier, Keycloak is already being used in production. Before doing so yourself, make sure to go through the production-readiness documentation.

Launching Keycloak

To launch Keycloak with Docker, use:

```
$ docker pull jboss/keycloak
$ docker run -d -e KEYCLOAK_USER=<USERNAME> -e
KEYCLOAK_PASSWORD=<PASSWORD> -p 8081:8080 jboss/keycloak
```

However, your configuration information (like realm settings, clients, or certificates) will be temporary in this scenario. Therefore, export the configuration and re-import every time before you instantiate a new container. In other words, use a persistent volume for storing the state.

To do a standalone Keycloak launch (https://www.keycloak.org/downloads.html) with JBoss WildFly, use:

```
$ keycloak-x.x.x.Final/bin>./standalone.sh
```

Preparing to integrate with Keycloak

Once you're ready to integrate your apps, tools, and services with Keycloak, you have decisions to make (see Figure 3):

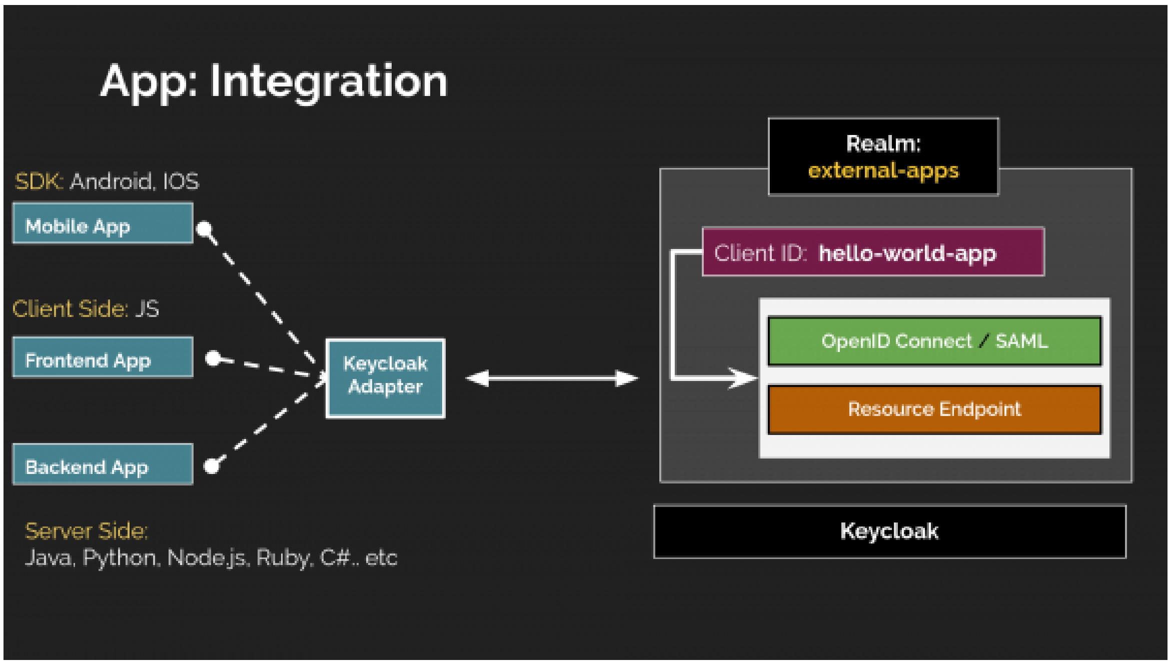


Figure 3: Keycloak integration map.">

First, you need to decide which protocol you intend to use, such as:

- OAuth2
- OpenID Connect
- Security Assertion Markup Language (SAML).

Are you looking for *authentication* or *authorization*?

```
OAuth 2 != Authentication, only Authorization

OpenID Connect = Identity + Authentication + Authorization
```

Now, regarding the application:

- Is it running on a container (stateless) or is it in a legacy clustered (shared state) environment?
- What does the architecture consist of, such as single-page applications (SPA), microservices, serverless, or MVC?
- Identify the resources and endpoints you want to secure. Is your integration between, for example, client and server, service-to-service, or API endpoints.
- Identify which adapter will be suited for your architecture.

Integrating with Keycloak

To integrate your apps with Keycloak:

1. Create a realm. You can use `master` for a dev environment or base it on your business domain (for example, `external-apps` or `internal-apps`).
2. Create a client for your application (for example, `hello-world-app`). Client configuration requires details like this:
   - **Protocol:** Which protocol, such as SAML or OpenID.
   - **Resource Endpoint:** The application hostname or REST endpoint.
   - **Redirect URI:** Where to redirect the user when authentication is granted.
3. Provide the client configuration to your application as input, such as:
   - The clientId (i.e., `hello-world-app`)
   - The realm (i.e., `external-apps`)
   - The Keycloak server's URL.

That’s all you need to do in order to configure your application with Keycloak.

Wrapping up

In conclusion, you can refer to the following integration patterns when you work with Keycloak yourself:

- Vue
- Angular
- React
- Spring-boot 2
- Quarkus and React

References

- Keycloak Documentation
- Securing Applications and Services Guide

Happy secure coding!

Last updated: December 23, 2021

Recent Articles

The Red Hat Cloud way: Event-driven, serverless, distributed cloud services to support modern apps

Fine-tune Kafka performance with the Kafka optimization theorem

Podman basics: Resources for beginners and experts

No-code and low-code integrations with Camel and Kaoto

Process Formula 1 telemetry with Quarkus and OpenShift Streams for Apache Kafka

Comments

What do you think of this content?

15 Responses

Upvote Funny Love Surprised Angry Sad

Red Hat Developer Comment Policy

Please keep your comments relevant and polite. Opinions shared in comments are not official Red Hat news or policy.
Please read our Comment Policy before commenting.

1 Comment **Red Hat Developer** **Disqus' Privacy Policy** 1 **Login**

Favorite Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Lu • 7 months ago

Hey. Where is that "production-readiness documentation"?

• Reply • Share ›

FEATURED TOPICS

Istio
Quarkus
CI/CD
Serverless
Enterprise
Java
Linux
Microservices

BUILD

Getting Started Center
Developer Tools
Interactive Tutorials
Container Catalog

QUICKLINKS

What's new
DevNation events
Upcoming Events
Books
Cheat Sheets
Videos
Products

COMMUNI

Site Status Dashboard
Report a wet issue
Report a seci problem
Helping durir COVID-19
About us

RED HAT DEVELOPER

Build here. Go anywhere.

We serve the builders. The problem solvers who create careers with code.