






React Router Quickstart

Before you start

-  [Set up a Clerk application](#)
-  [Create a React Router application](#)

Example repository

-  [React Router Quickstart Repo](#)

React Router can be used as a framework or as a standalone library. This tutorial explains how to use React Router as a framework. To use React Router as a library instead, see the [library mode tutorial](#).

This tutorial assumes that you're using React Router **v7.1.2 or later** in framework mode.

1 Install `@clerk/react-router`

The [Clerk React Router SDK](#) provides prebuilt components, hooks, and helpers to make it easy to integrate authentication and user management in your React Router app.

Run the following command to install the SDK:

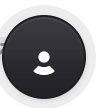
```
npm yarn pnpm bun
```

```
terminal
```

```
1 npm install @clerk/react-router
```

2 Set your Clerk API keys

Add the following keys to your `.env` file. These keys can always be retrieved from the [API keys](#) page in the Clerk Dashboard.



You can control which content signed-in and signed-out users can see with Clerk's prebuilt control components.

The following example adds `<ClerkProvider>` and creates a header using the following Clerk components:

- `<SignedIn>`: Children of this component can only be seen while **signed in**.
- `<SignedOut>`: Children of this component can only be seen while **signed out**.
- `<UserButton />`: Shows the signed-in user's avatar. Selecting it opens a dropdown menu with account management options.
- `<SignInButton />`: An unstyled component that links to the sign-in page. In this example, since no props or environment variables are set for the sign-in URL, this component links to the Account Portal sign-in page.

app/root.tsx

```
1 // Other imports
2
3 import { ClerkProvider, SignedIn, SignedOut, UserButton, SignInButton } from '
4
5 export default function App({ loaderData }: Route.ComponentProps) {
6   return (
7     <ClerkProvider
8       loaderData={loaderData}
9       signUpFallbackRedirectUrl="/"
10      signInFallbackRedirectUrl="/"
11     >
12       <header className="flex items-center justify-center py-8 px-4">
13         <SignedOut>
14           <SignInButton />
15         </SignedOut>
16         <SignedIn>
17           <UserButton />
18         </SignedIn>
19       </header>
20       <main>
```

```
24   )  
25 }  
26  
27 // Rest of the root.tsx code
```

5 Create your first user

Run your project with the following command:

```
npm yarn pnpm bun
```

```
terminal
```

```
1 npm run dev
```

Visit your app's homepage at <http://localhost:5173>. Sign up to create your first user.

Next steps

Create a custom sign-in-or-up page

Learn how add custom sign-in-or-up page with Clerk components.


Read session and user data

Learn how to use Clerk's hooks and helpers to access the active session and user data in your React Router app.

Library mode

Learn how to use Clerk with React Router in library mode to add authentication to your application.



 [Edit this page on GitHub](#)



Last updated on Jun 16, 2025



Product

[Components](#)

[Pricing](#)

[Dashboard](#)

[Feature requests](#)

[React authentication](#)

Developers

[Documentation](#)

[Discord server](#)

[Support](#)

[Glossary](#)

[Terms of Engagement](#)

[Changelog](#)

Company

[About](#)

[Careers](#)

[Blog](#)

[Contact](#)

[Brand assets](#)

Legal

[Terms and Conditions](#)

[Privacy Policy](#)

[Data Processing Agreement](#)

[Do Not Sell/Share My Info](#)

[Cookie manager](#)



Clerk React Router SDK

The Clerk React Router SDK gives you access to prebuilt components, React hooks, and helpers to make user authentication easier. Refer to the [quickstart guide](#) to get started.

Client-side helpers

Because the React Router SDK is built on top of the [React SDK](#), you can use the hooks that the React SDK provides. These hooks include access to the `Clerk` object, `User` object, `Organization` object, and a set of useful helper methods for signing in and signing up. Learn more in the [React SDK reference](#).

- `useUser()`
- `useClerk()`
- `useAuth()`
- `useSignIn()`
- `useSignUp()`
- `useSession()`
- `useSessionList()`
- `useOrganization()`
- `useOrganizationList()`

Server-side helpers

The following references show how to integrate Clerk features into applications using React Router server functions and API routes.

- `getAuth()`



React Router implementations

React Router can be integrated with Clerk in two ways:

- As a framework (recommended): Configure your app using [Clerk's React Router SDK](#)
- As a library: Manually integrate React Router into your React + Vite app using [library mode](#)

What did you think of this content?



It was helpful



It was not helpful



I have feedback



[Edit this page on GitHub](#)



Last updated on Jun 16, 2025



Product

[Components](#)

[Pricing](#)

[Dashboard](#)

[Feature requests](#)

[React authentication](#)

Developers

[Documentation](#)

[Discord server](#)

[Support](#)

[Glossary](#)

[Terms of Engagement](#)

[Changelog](#)

Company

[About](#)

[Careers](#)

[Blog](#)

[Contact](#)

[Brand assets](#)

Legal

[Terms and Conditions](#)

[Privacy Policy](#)

[Data Processing Agreement](#)

[Do Not Sell/Share My Info](#)

[Cookie manager](#)



rootAuthLoader()

The `rootAuthLoader()` function configures Clerk to handle authentication state for React Router routes, allowing easy access to user session information in your app.

Usage

You can use the `rootAuthLoader()` in two different ways:

- Without a callback, which will just return the auth state
- With a callback function to handle custom data loading while having access to auth state

You can also pass configuration options to `rootAuthLoader()` no matter which method you use.

Without a callback

In your `root.tsx` file, add `rootAuthLoader()` to the `loader()` function. If your app doesn't have a `loader()` function yet, you'll need to add it manually.

app/root.tsx



```
1 import { rootAuthLoader } from '@clerk/react-router/ssr.server'
2 import type { Route } from './+types/root'
3
4 export async function loader(args: Route.LoaderArgs) {
5   return rootAuthLoader(args)
6 }
```



route data loading with auth state.

app/root.tsx

```
1 import { rootAuthLoader } from '@clerk/react-router/ssr.server'
2 import type { Route } from './+types/root'
3
4 export async function loader(args: Route.LoaderArgs) {
5   return rootAuthLoader(args, ({ request, context, params }) => {
6     const { sessionId, userId, getToken } = request.auth
7     // Add logic to fetch data
8     return { yourData: 'here' }
9   })
10 }
```

Pass configuration options

To pass configuration options to `rootAuthLoader()`, you can pass an optional argument to the `rootAuthLoader()` function.

app/root.tsx

```
1 import { rootAuthLoader } from '@clerk/react-router/ssr.server'
2 import type { Route } from './+types/root'
3
4 export async function loader(args: Route.LoaderArgs) {
5   return rootAuthLoader(
6     args,
7     ({ request, context, params }) => {
8       const { sessionId, userId, getToken } = request.auth
9       // Add logic to fetch data
10      return { yourData: 'here' }
11    },
12    { signInUrl: '/sign-in' }, // Options
13  )
14 }
```

`audience?` string | string[]

A string or list of [audiences](#) 🔗. If passed, it is checked against the `aud` claim in the token.

`authorizedParties?` string[]

An allowlist of origins to verify against, to protect your application from the subdomain cookie leaking attack. For example: `['http://localhost:3000', 'https://example.com']`

`domain?` string

The domain used for satellites to inform Clerk where this application is deployed.

`isSatellite?` boolean

When using Clerk's satellite feature, this should be set to `true` for secondary domains.

`jwtKey` string

Used to verify the session token in a networkless manner. Supply the **JWKS Public Key** from the [API keys](#) 🔗 page in the Clerk Dashboard. **It's recommended to use the environment variable instead.** For more information, refer to [Manual JWT verification](#).

`proxyUrl?` string

Specify the URL of the proxy, if using a proxy.

`publishableKey` string

The Clerk Publishable Key for your instance. This can be found in the [API keys](#) 🔗 page -> **Show Publishable Key** section in the Clerk Dashboard. It's recommended to use the environment variable instead.

`secretKey?` string

`signInUrl` string

This URL will be used for any redirects that might happen and needs to point to your primary application on the client-side. This option is optional for production instances. **It is required to be set for a satellite application in a development instance.** It's recommended to use [the environment variable](#) instead.

`signUpUrl` string

This URL will be used for any redirects that might happen and needs to point to your primary application on the client-side. This option is optional for production instances but **must be set for a satellite application in a development instance.** It's recommended to use [the environment variable](#) instead.

`signInFallbackRedirectUrl?` string

The fallback URL to redirect to after the user signs in, if there's no `redirect_url` in the path already. Defaults to `/`. It's recommended to use [the environment variable](#) instead.

`signUpFallbackRedirectUrl?` string


The fallback URL to redirect to after the user signs up, if there's no `redirect_url` in the path already. Defaults to `/`. It's recommended to use [the environment variable](#) instead.

`signInForceRedirectUrl?` string

If provided, this URL will always be redirected to after the user signs in. It's recommended to use [the environment variable](#) instead.

`signUpForceRedirectUrl?` string

If provided, this URL will always be redirected to after the user signs up. It's recommended to use [the environment variable](#) instead.

 Edit this page on GitHub



Last updated on Jun 16, 2025



Product

- Components
- Pricing
- Dashboard
- Feature requests
- React authentication

Developers

- Documentation
- Discord server
- Support
- Glossary
- Terms of Engagement
- Changelog

Company

- About
- Careers
- Blog
- Contact
- Brand assets

Legal

- Terms and Conditions
- Privacy Policy
- Data Processing Agreement
- Do Not Sell/Share My Info
- Cookie manager



References / `getAuth()`

getAuth()

The `getAuth()` helper retrieves the current user's authentication state from the request object.

Parameters

`request`

The request object.

`opts?`

An optional object that can be used to configure the behavior of the `getAuth()` function. It accepts the following properties:

- `secretKey?`: A string that represents the secret key used to sign the session token. If not provided, the secret key is retrieved from the environment variable `CLERK_SECRET_KEY`.

`acceptsToken?` `TokenType`

The type of authentication token(s) to accept. Valid values are:

- `'session_token'`
- `'oauth_token'`
- `'machine_token'`
- `'machine_key'`

Can be set to:

- A single token type.
- An array of token types.



Returns

`getAuth()` returns the `Auth` object. This JavaScript object contains important information like the current user's session ID, user ID, and organization ID. Learn more about the `Auth` object [↗](#).

Usage

See the [dedicated guide](#) for example usage.

What did you think of this content?



It was helpful



It was not helpful



I have feedback



Edit this page on GitHub



Last updated on Jun 16, 2025

