Edge Functions Overview

Vercel's Edge Functions enable you to deliver dynamic, personalized content with the lightweight Edge Runtime. Learn more about Edge Functions here.

Table of Contents

Edge Functions are available on all plans

Vercel's Edge Functions enable you to deliver dynamic, personalized content with the lightweight Edge Runtime.

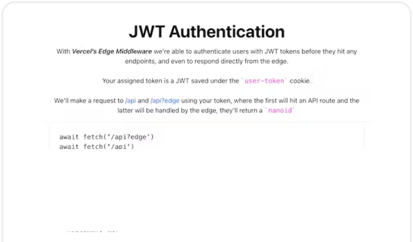
Our Edge Runtime is more performant and cost-effective than Serverless Functions on average. Edge Functions are deployed globally on our Edge Network, and can automatically execute in the region nearest to means they don't need extra time to start up before executing your code.

Edge Functions are useful when you need to interact with data over the network as fast as possible, such as executing OAuth callbacks, responding to webhook requests, or interacting with an API that fails if a request is not completed within a short time limit.

See the Quickstart to learn how to create your first Edge Function.

Get started in minutes

Deploy an Edge Function Template


JWT Authentication

With ***Vercel's Edge Middleware*** we're able to authenticate users with JWT tokens before they hit any endpoints, and even to respond directly from the edge.

Your assigned token is a JWT saved under the `user-token` cookie.

We'll make a request to /api and /api?edge using your token, where the first will hit an API route and the latter will be handled by the edge, they'll return a `nanoid`

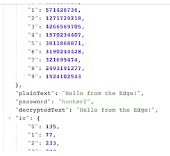
```
await fetch('/api?edge')
await fetch('/api')
```

JWT Authentication

Learn how to do JWT authentication at the edge.

CORS in Edge Functions

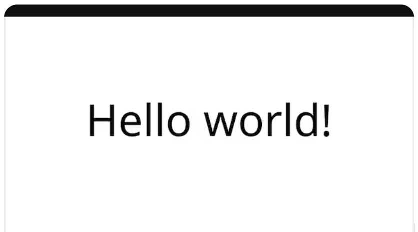
Handle CORS at the edge.



```
      "1": 571426736,
      "2": 1271729218,
      "3": 4266569705,
      "4": 1570234407,
      "5": 3811868871,
      "6": 3190244428,
      "7": 321699474,
      "8": 2493191277,
      "9": 1524102543
  },
  "plainText": "Hello from the Edge!",
  "password": "hunter2",
  "decryptedText": "Hello from the Edge!",
  "iv": {
      "0": 135,
      "1": 77,
      "2": 233,
```

Using Crypto in Edge Middlewa...

Learn to utilize the crypto Web APIs at the edge.



Open Graph Image Generation

Compute and generate dynamic social card images with React components.

SvelteKit Deployment Configuration

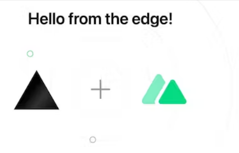
SvelteKit apps can now control their deployment configuration on a per-route basis. This means we can render some routes with Edge Functions, some with Serverless Functions, and others with Incremental Static Regeneration. If no configuration is specified, it will use the project defaults.

To specify the deployment configuration of this endpoint to be edge, add the following code to your `+page.js` file:

```
export const config = {
   runtime: 'edge'
};
```

SvelteKit Route Config

This template shows how to configure Edge Functions, Serverless Functions, and ISR in...



Nuxt on the Edge

Vue based SSR on the edge, powered by Nuxt 3, Nitro, and Vercel Edge Functions.

View All Templates

How Edge Functions work

Edge Functions use Vercel's Edge Runtime, which is built on the same high-performance V8 JavaScript and WebAssembly engine used by the Chrome browser.

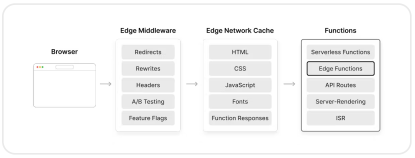
Using the V8 engines allows Edge Functions to run in isolated execution environments that don't require a container or virtual machine. This constrains the Edge Runtime in many ways, but also keeps it lightweight. Starting up an Edge Function process requires fewer resources than a Serverless Function, effectively eliminating cold boot times.

The universal benefit of Edge Functions is the Edge Runtime. It's cost-effective and resource-minimal. However, you can also use Edge Functions to deliver dynamic content at low latency in certain contexts.

Delivering dynamic content at low latency is possible because all Edge Functions are deployed

Network. They can be invoked in whichever data center is closest to the user, or in a region near your databases. You can also serve them personalized content based on the accessing region at high speeds.

Responses from Edge Functions can be *cached* and *streamed* in real time.



Edge Functions location within Vercel infrastructure.

Why use Edge Functions

Edge Functions offer faster responses than Serverless Functions on average for globally distributed userbases. The following are some specific benefits that Edge Functions provide:

Reduced network usage costs

Because Edge Functions can execute geographically near your users, or near your databases, their network requests can travel shorter distances and accrue lower costs than Serverless Functions.

Serverless Functions are usually

region, so they sometimes travel far to reach your database before they can deliver a response to the user.

See our docs on Edge Network regions to configure your Edge Functions to execute closest to their dependencies.

Reduced latency

Edge Functions that don't rely on databases or geographically distant dependencies can be configured to execute near your users. This can lead to responses being delivered to users at **much** lower latency than Serverless Functions.

For example, with Edge Config you could enable or disable feature flags in the frontend of your application based on the region an Edge Function executes in. Using an Edge Function to do this would cause all app routes that check for the feature flag to respond hundreds of milliseconds faster, since they would not suffer from the slow cold boot times and cross-region invocation delay that come with Serverless Functions.

Personalized dynamic content

You can use custom code in your Edge Functions to deliver content based on which data center on the Edge Network a user is invoking a Function in.

Support for other languages

You can compile libraries and tools written in languages like C or Rust and use them in Edge Functions. See our WebAssembly docs to learn more.

Regional Edge Function invocation

To learn more about configuring regions, see Regions.

Using a database with Edge Functions

If your Edge Function API routes depend on databases, overall response times could be slow in some scenarios.

For example, if a visitor triggers an Edge Function in Japan, but it depends on a database in San Francisco, the Function will have to send requests to San Francisco for each call to the DB.

To avoid these long roundtrips, you could limit your Edge Functions to regions near your database dependencies, or you could use a globally-distributed database.

We offer several global data

Blob. See our storage docs to learn which option is best for you.

Limitations

Edge Functions may not be suitable for all use cases and there are some limitations you should be aware of:

- The maximum initial response time for an Edge Function is **25 seconds**
- The maximum post-compression size for an Edge Function is **4 MB**, including all the code that is bundled in the function
- The distance between your data source and where the Edge Function runs could add unwanted latency to the request. Read our guide on setting edge functions regions manually to potentially mitigate this issue
- **Most native Node.js APIs are not supported**. See our list of supported Node.js APIs here

See the comparison with Serverless Functions and Edge Functions limitations page for more information.

Key takeaways

- Edge Functions run *after* the Edge Network cache, and unlike Edge Middleware, *can* therefore cache responses

Edge Functions use the Edge Runtime. This allows for faster cold boots, but means that you **cannot** use Node.js APIs in Edge Functions

- Their signature matches Edge Middleware
  - Note that when using Edge Functions with Next.js (as API Routes configured to run at the edge), you **can** import and use the next/server helpers (`NextRequest`, `NextResponse`). However, this is not a requirement
- Edge Functions *can* return responses
- When implementing Edge Functions, you can add the `@vercel/edge` package and import helper functions such as `geolocation` and `ipAddress`
- For information on comparing Edge Functions and Serverless Functions, see the comparison table.

Logging

Edge Functions have full support for the `console` API, including `time`, `debug`, `timeEnd`, etc. You can view runtime logs for your Edge Functions by using the **Logs** tab. You can filter to see Edge Functions by selecting **Edge Functions** under **Type**. You can also use the Functions section to search by and view the logs for specific functions.

Last updated on February 24, 2023

Previous
Ruby

Next
Quickstart

Was this helpful?