Version: 4.x

On this page

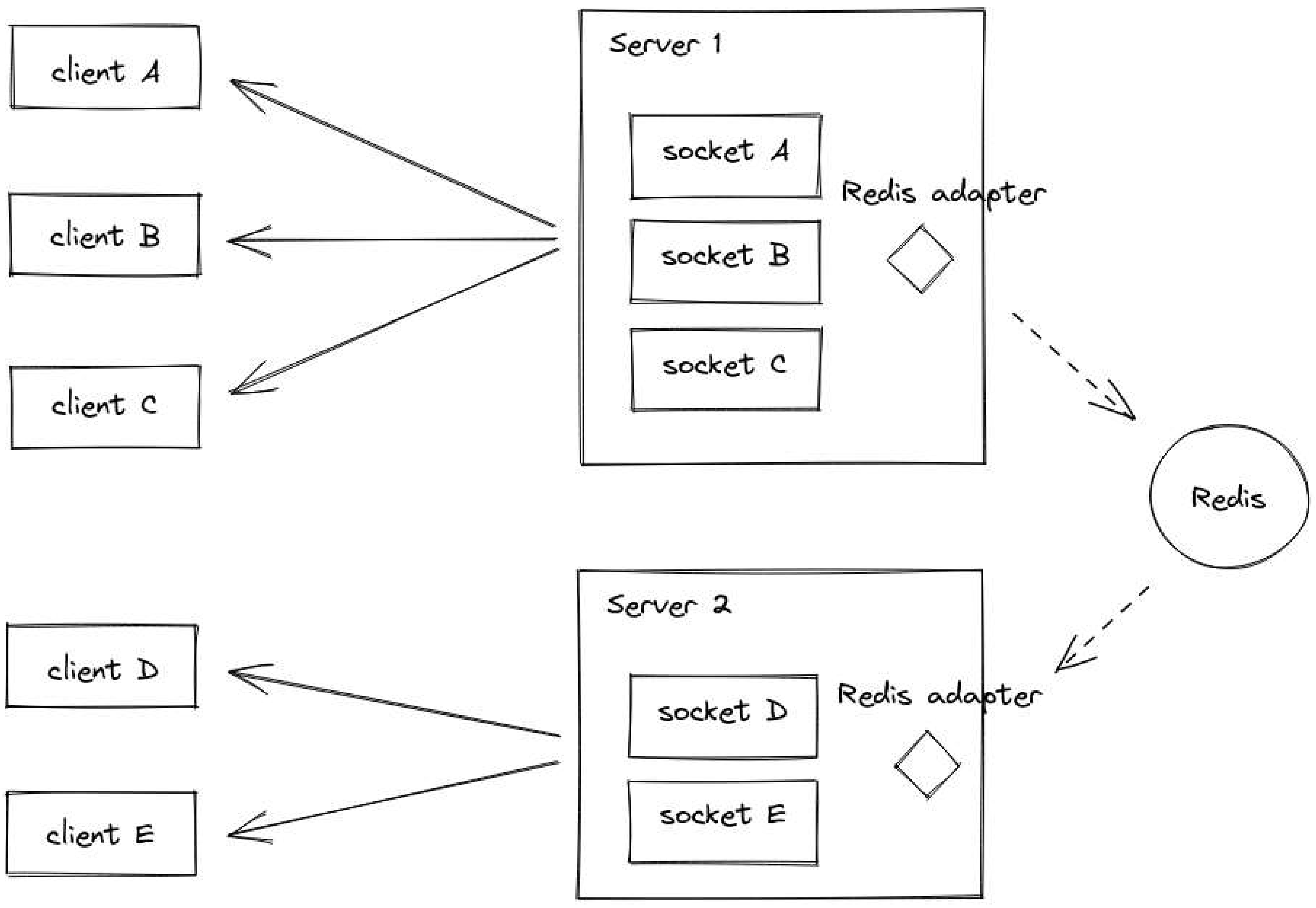
Redis adapter

How it works

The Redis adapter relies on the Redis Pub/Sub mechanism.

Every packet that is sent to multiple clients (e.g. `io.to("room1").emit()` or `socket.broadcast.emit()`) is:

- sent to all matching clients connected to the current server
- published in a Redis channel, and received by the other Socket.IO servers of the cluster



The source code of this adapter can be found here.

Installation

```
npm install @socket.io/redis-adapter redis
```

For TypeScript users, you will also need to install `@types/redis` if you are using `redis@3`.

Usage

```
import { Server } from "socket.io";
import { createAdapter } from "@socket.io/redis-adapter";
import { createClient } from "redis";

const io = new Server();

const pubClient = createClient({ url: "redis://localhost:6379" });
const subClient = pubClient.duplicate();

Promise.all([pubClient.connect(), subClient.connect()]).then(() => {
  io.adapter(createAdapter(pubClient, subClient));
  io.listen(3000);
});
```

Note: with `redis@3`, calling `connect()` on the Redis clients is not needed:

```
import { Server } from "socket.io";
import { createAdapter } from "@socket.io/redis-adapter";
import { createClient } from "redis";

const io = new Server();

const pubClient = createClient({ url: "redis://localhost:6379" });
const subClient = pubClient.duplicate();

io.adapter(createAdapter(pubClient, subClient));
io.listen(3000);
```

Or with `ioredis`:

```
import { Server } from "socket.io";
import { createAdapter } from "@socket.io/redis-adapter";
import { Cluster } from "ioredis";
```

```
const io = new Server();

const pubClient = new Cluster([
  {
    host: "localhost",
    port: 6380,
  },
  {
    host: "localhost",
    port: 6381,
  },
]);

const subClient = pubClient.duplicate();

io.adapter(createAdapter(pubClient, subClient));
io.listen(3000);
```

Options

Name	Description	Default value
`key`	the prefix for the name of the Pub/Sub channel	`socket.io`
`requestsTimeout`	the timeout for inter-server requests such as `fetchSockets()` or `serverSideEmit()` with ack	`5000`

Common questions

- Do I still need to enable sticky sessions when using the Redis adapter?

Yes. Failing to do so will result in HTTP 400 responses (you are reaching a server that is not aware of the Socket.IO session).

More information can be found here.

- What happens when the Redis server is down?

In case the connection to the Redis server is severed, the packets will only be sent to the clients that are connected to the current server.

Migrating from `socket.io-redis`

The package was renamed from `socket.io-redis` to `@socket.io/redis-adapter` in v7, in order to match the name of the Redis emitter (`@socket.io/redis-emitter`).

To migrate to the new package, you'll need to make sure to provide your own Redis clients, as the package will no longer create Redis clients on behalf of the user.

Before:

```
const redisAdapter = require("socket.io-redis");

io.adapter(redisAdapter({ host: "localhost", port: 6379 }));
```

After:

```
const { createClient } = require("redis");
const { createAdapter } = require("@socket.io/redis-adapter");

const pubClient = createClient({ url: "redis://localhost:6379" });
const subClient = pubClient.duplicate();

io.adapter(createAdapter(pubClient, subClient));
```

TIP

The communication protocol between the Socket.IO servers has not been updated, so you can have some servers with `socket.io-redis` and some others with `@socket.io/redis-adapter` at the same time.

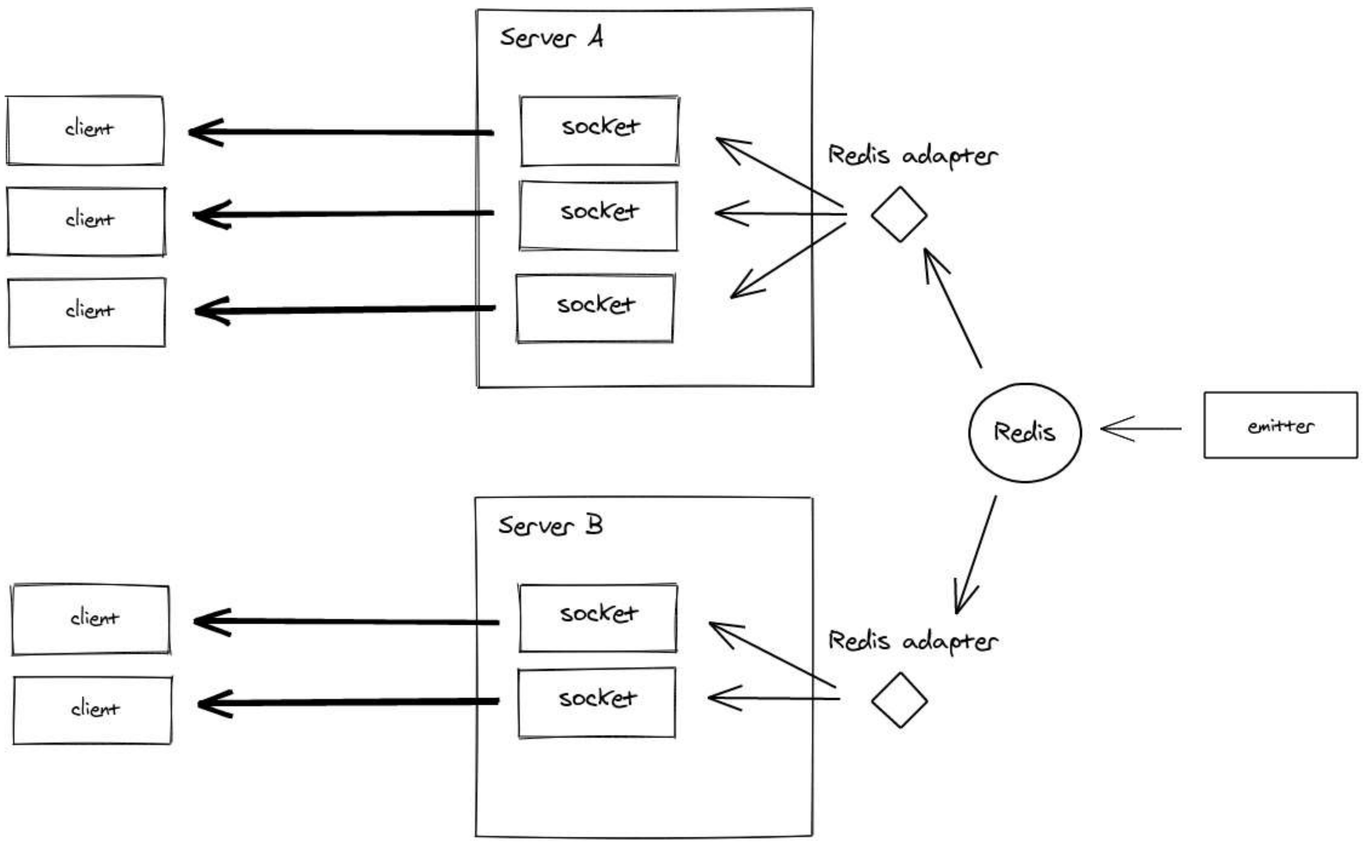
Latest releases

- 8.0.0 (Dec 2022)
- 7.2.0 (May 2022)

- 7.1.0 (Nov 2021)
- 7.0.1 (Nov 2021)
- 7.0.0 (May 2021)
- 6.1.0 (Mar 2021)
- 6.0.1 (Nov 2020)

Emitter

The Redis emitter allows sending packets to the connected clients from another Node.js process:



This emitter is also available in several languages:

- Javascript: https://github.com/socketio/socket.io-redis-emitter
- Java: https://github.com/sunsus/socket.io-java-emitter
- Python: https://pypi.org/project/socket.io-emitter/
- PHP: https://github.com/rase-/socket.io-php-emitter
- Golang: https://github.com/yosuke-furukawa/socket.io-go-emitter
- Perl: https://metacpan.org/pod/SocketIO::Emitter
- Rust: https://github.com/epli2/socketio-rust-emitter

Installation

```
npm install @socket.io/redis-emitter redis
```

Usage

```
import { Emitter } from "@socket.io/redis-emitter";
import { createClient } from "redis";

const redisClient = createClient({ url: "redis://localhost:6379" });

redisClient.connect().then(() => {
  const emitter = new Emitter(redisClient);

  setInterval(() => {
    emitter.emit("time", new Date);
  }, 5000);
});
```

Note: with `redis@3`, calling `connect()` on the Redis client is not needed:

```
import { Emitter } from "@socket.io/redis-emitter";
import { createClient } from "redis";

const redisClient = createClient({ url: "redis://localhost:6379" });
const emitter = new Emitter(redisClient);

setInterval(() => {
  emitter.emit("time", new Date);
}, 5000);
```

Please refer to the cheatsheet here.

Migrating from `socket.io-emitter`

The package was renamed from `socket.io-emitter` to `@socket.io/redis-emitter` in v4, in order to better reflect the relationship with Redis.

To migrate to the new package, you'll need to make sure to provide your own Redis clients, as the package will no longer create Redis clients on behalf of the user.

Before:

```
const io = require("socket.io-emitter")({ host: "127.0.0.1", port: 6379 });
```

After:

```
const { Emitter } = require("@socket.io/redis-emitter");
const { createClient } = require("redis");

const redisClient = createClient();
const io = new Emitter(redisClient);
```

Latest releases

- 5.0.0 (Sep 2022)
- 4.1.1 (Jan 2022)
- 4.1.0 (May 2021)
- 4.0.0 (Mar 2021)
- 3.2.0 (Dec 2020)
- 3.1.1 (Oct 2017)

Edit this page

*Last updated on **12/19/2022***