Laravel Queues Integration with Cloudflare Queues

Documentation by Yogesh

📓 Overview

This document explains how to connect **Laravel's queue system** with **Cloudflare Queues**.
Laravel normally supports queue backends like Redis, SQS, and Database.
Since Cloudflare Queues is not natively supported, we will create a **custom Laravel queue driver** that communicates with Cloudflare via its **HTTP API**.

🧠 What You Will Achieve

By the end of this guide:

- Laravel will be able to **send jobs** to Cloudflare Queues.
- You'll understand how to configure Cloudflare Queues.
- You'll be ready to extend Laravel workers to **consume jobs** from Cloudflare.

⚙️ Prerequisites

✅ Installed and working Laravel project (e.g. Laravel 10 or higher)
✅ PHP 8.1+
✅ Composer installed
✅ Cloudflare account with "Workers & Queues" access

□ Step 1: Verify Laravel Queues Work Locally

Before connecting to Cloudflare, test Laravel's built-in queue system.

1. Create the queue table:

```
2. php artisan queue:table
3. php artisan migrate
```

4. Set queue connection in `.env`:

```
5. QUEUE_CONNECTION=database
```

6. Create a test job:

```
php artisan make:job TestJob
```

8. Edit `app/Jobs/TestJob.php` → inside `handle()` method:

```
public function handle()
{
    \Log::info('✅ Test Job Executed!');
}
```

13. Dispatch a test job:

```
php artisan tinker
>>> dispatch(new App\Jobs\TestJob);
```

16. Start the queue worker:

```
php artisan queue:work
```

✅ Check `storage/logs/laravel.log` — you should see "Test Job Executed!".

□ Step 2: Create a Cloudflare Queue

1. Go to your **Cloudflare Dashboard** → https://dash.cloudflare.com
2. In the left sidebar, click **Workers & Queues → Queues**
3. Click **Create Queue** and give it a name:

```
laravel-test-queue
```

5. Copy these details:
   - **Account ID**
   - **Queue Name**
   - **API Token** (create one with `Workers & Queues:Edit` permission)

You'll use these in Laravel next.

□ Step 3: Understand Cloudflare Queue APIs

Cloudflare Queues uses HTTPS endpoints.

📤 Publish (Send message)

```
POST
https://api.cloudflare.com/client/v4/accounts/<ACCOUNT_ID>/queues/<QUEUE_NAME>/messages

Headers:

  Authorization: Bearer <API_TOKEN>
```

```
  Content-Type: application/json
Body:
  {"body": "Hello from Laravel!"}
```

📥 Pull (Receive message)

```
POST https://api.cloudflare.com/client/v4/accounts/<ACCOUNT_ID>/queues/<QUEUE_NAME>/pull
Headers:
  Authorization: Bearer <API_TOKEN>
```

□ Step 4: Create a Custom Laravel Queue Driver (Cloudflare)

We'll create a small custom driver to let Laravel communicate with Cloudflare Queues.

Step 4.1 Create Folder

In your Laravel project:

```
app/Queue/Connectors/
```

Step 4.2 Create `CloudflareConnector.php`

```php
<?php

namespace App\Queue\Connectors;

use Illuminate\Queue\Connectors\ConnectorInterface;
use App\Queue\CloudflareQueue;

class CloudflareConnector implements ConnectorInterface
{
    public function connect(array $config)
    {
        return new CloudflareQueue(
            $config['account_id'],
            $config['queue_name'],
            $config['api_token']
        );
    }
}
```

Step 4.3 Create `CloudflareQueue.php`

```php
<?php

namespace App\Queue;

use Illuminate\Contracts\Queue\Queue as QueueContract;
use Illuminate\Queue\Queue;
use Illuminate\Support\Facades\Http;

class CloudflareQueue extends Queue implements QueueContract
{
    protected $accountId;
    protected $queueName;
    protected $apiToken;

    public function __construct($accountId, $queueName, $apiToken)
    {
        $this->accountId = $accountId;
        $this->queueName = $queueName;
        $this->apiToken = $apiToken;
    }

    public function push($job, $data = '', $queue = null)
    {
        $payload = $this->createPayload($job, $this->getQueue($queue), $data);

        $url = "https://api.cloudflare.com/client/v4/accounts/{$this->accountId}/queues/{$this->queueName}/messages";

        return Http::withToken($this->apiToken)
            ->post($url, [
                'body' => $payload,
            ])->json();
    }

    public function pop($queue = null)
    {
        $url = "https://api.cloudflare.com/client/v4/accounts/{$this->accountId}/queues/{$this->queueName}/pull";
```

```
        $response = Http::withToken($this->apiToken)->post($url)->json();

        return $response;
    }
}
```

□ Step 5: Register the Cloudflare Driver in Laravel

Open:

`app/Providers/AppServiceProvider.php`

Add:

```
use Illuminate\Support\Facades\Queue;
use App\Queue\Connectors\CloudflareConnector;

public function boot()
{
    Queue::extend('cloudflare', function ($app) {
        return new CloudflareConnector();
    });
}
```

□ Step 6: Add Configuration in `.env`

```
QUEUE_CONNECTION=cloudflare
CLOUDFLARE_ACCOUNT_ID=xxxxxxxxxxxxxxxx
CLOUDFLARE_QUEUE_NAME=laravel-test-queue
CLOUDFLARE_API_TOKEN=xxxxxxxxxxxxxxxx
```

□ Step 7: Update `config/queue.php`

Inside the `'connections'` array, add:

```
'cloudflare' => [
    'driver' => 'cloudflare',
    'account_id' => env('CLOUDFLARE_ACCOUNT_ID'),
    'queue_name' => env('CLOUDFLARE_QUEUE_NAME'),
    'api_token' => env('CLOUDFLARE_API_TOKEN'),
```

```
],
```

□ Step 8: Test Sending Jobs to Cloudflare

Run in Tinker:

```
php artisan tinker
>>> dispatch(new App\Jobs\TestJob);
```

Then check your Cloudflare Dashboard → Queues → you should see a message appear in the queue.

✅ Congratulations — Laravel is now pushing jobs into Cloudflare Queues!

□ Step 9: (Optional) Consume Messages

Currently, Laravel sends jobs to Cloudflare.
To **process** them:

- You can use a **Cloudflare Worker** to consume and call a Laravel API endpoint.
- Or write a Laravel command that **pulls messages** via Cloudflare's `/pull` API and runs them.

This can be added later as a second phase.

⚠ Limitations

Limitation	Description
No official Laravel driver	Custom code required
HTTP-based polling	Slightly higher latency than Redis/SQS
Order not guaranteed	Cloudflare Queues does not ensure strict ordering
Retries/manual delete	Must be handled in custom logic

□ Summary

Step	Description
1	Verified Laravel queue works locally
2	Created Cloudflare queue
3	Learned Cloudflare API endpoints

Step	Description
4	Built custom Laravel driver
5	Registered driver in AppServiceProvider
6	Added `.env` and config setup
7	Tested queue dispatch
8	(Optional) Consume messages

🔑 Conclusion

✅ **Yes, Laravel can work with Cloudflare Queues**, but not natively.
You can integrate it by building a small custom driver that sends jobs via the Cloudflare API.
This gives flexibility to use Cloudflare's global network for scalable, cost-efficient message queuing.
