Real-Time Notifications in Laravel with WebSockets and Pusher

Here's a comprehensive guide to implementing real-time notifications in your Laravel application using either Laravel WebSockets (self-hosted) or Pusher (cloud-based).

Option 1: Using Laravel WebSockets (Self-Hosted)

1. Install Required Packages

```
composer require beyondcode/laravel-websockets
composer require pusher/pusher-php-server
```

2. Publish Configurations

```
php artisan vendor:publish --
provider="BeyondCode\LaravelWebSockets\WebSocketsServiceProvider" --
tag="config"
php artisan vendor:publish --
provider="BeyondCode\LaravelWebSockets\WebSocketsServiceProvider" --
tag="migrations"
```

3. Configure Environment Variables (.env)

```
BROADCAST_DRIVER=pusher

PUSHER_APP_ID=12345
PUSHER_APP_KEY=your-app-key
PUSHER_APP_SECRET=your-app-secret
PUSHER_APP_CLUSTER=mt1

# For Laravel WebSockets
LARAVEL_WEBSOCKETS_PORT=6001
```

PROF

4. Configure config/broadcasting.php

```
'connections' => [
    'pusher' => [
        'driver' => 'pusher',
        'key' => env('PUSHER_APP_KEY'),
        'secret' => env('PUSHER_APP_SECRET'),
```

```
        'app_id' => env('PUSHER_APP_ID'),
        'options' => [
            'cluster' => env('PUSHER_APP_CLUSTER'),
            'encrypted' => true,
            'host' => '127.0.0.1',
            'port' => env('LARAVEL_WEBSOCKETS_PORT', 6001),
            'scheme' => 'http',
            'useTLS' => false,
        ],
    ],
],
```

5. Create Notification Event

```
php artisan make:event NotificationEvent
```

Update the event:

```
<?php

namespace App\Events;

use Illuminate\Broadcasting\Channel;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Contracts\Broadcasting\ShouldBroadcast;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Queue\SerializesModels;

class NotificationEvent implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    public $message;
    public $userId;

    public function __construct($message, $userId)
    {
        $this->message = $message;
        $this->userId = $userId;
    }

    public function broadcastOn()
    {
        return new Channel('user.'.$this->userId);
    }

    public function broadcastAs()
    {
```

```
        return 'notification.event';
    }
}
```

6. Create Notification Component (Frontend)

Install Laravel Echo and Pusher JS:

```
npm install --save laravel-echo pusher-js
```

Add to `resources/js/bootstrap.js`:

```
import Echo from 'laravel-echo';

window.Pusher = require('pusher-js');

window.Echo = new Echo({
    broadcaster: 'pusher',
    key: process.env.MIX_PUSHER_APP_KEY,
    wsHost: window.location.hostname,
    wsPort: 6001,
    forceTLS: false,
    disableStats: true,
    enabledTransports: ['ws', 'wss'],
});
```

7. Listen for Events in JavaScript

PROF

```
// Listen for notifications for the authenticated user
const userId = document.querySelector("meta[name='user-id']").content;

window.Echo.private(`user.${userId}`)
    .listen('.notification.event', (data) => {
        console.log('New notification:', data.message);
        // Update UI with the new notification
        showNotification(data.message);
    });

function showNotification(message) {
    // Implement your notification UI logic here
    alert(message); // Simple example
}
```

8. Trigger Notifications from Your Code

```
// Example controller method
public function sendNotification()
{
    $message = "This is a real-time notification!";
    $userId = auth()->id(); // Or any user ID you want to notify

    event(new NotificationEvent($message, $userId));

    return response()->json(['status' => 'Notification sent!']);
}
```

9. Run WebSockets Server

```
php artisan websockets:serve
```

10. Update Docker Setup (if using Docker)

Add this to your `docker-compose.yml`:

```
websocket:
    build:
      context: .
      dockerfile: docker/php/Dockerfile
    container_name: laravel_websocket
    command: php artisan websockets:serve
    volumes:
      - ./src:/var/www
    depends_on:
      - mysql
```

PROF

Option 2: Using Pusher (Cloud Service)

1. Install Pusher Package

```
composer require pusher/pusher-php-server
```

2. Configure Environment Variables (.env)

```
BROADCAST_DRIVER=pusher

PUSHER_APP_ID=your-app-id
PUSHER_APP_KEY=your-app-key
```

```
PUSHER_APP_SECRET=your-app-secret
PUSHER_APP_CLUSTER=your-cluster
```

3. Update config/broadcasting.php

```
'options' => [
    'cluster' => env('PUSHER_APP_CLUSTER'),
    'encrypted' => true,
    'useTLS' => true,
],
```

4. Frontend Setup

Update `resources/js/bootstrap.js`:

```
window.Echo = new Echo({
    broadcaster: 'pusher',
    key: process.env.MIX_PUSHER_APP_KEY,
    cluster: process.env.MIX_PUSHER_APP_CLUSTER,
    forceTLS: true
});
```

5. No Need to Run WebSocket Server

Pusher handles the WebSocket server for you in the cloud.

Testing Your Implementation

1. Start your Laravel application and WebSocket server (if using Laravel WebSockets)
2. Open two browser windows/tabs logged in as the same user
3. Trigger a notification from one window (via your controller method)
4. The other window should receive the notification in real-time

PROF

Additional Features

Database Notifications

Combine with Laravel's database notifications:

```
php artisan notifications:table
php artisan migrate
```

Create a notification:

```
php artisan make:notification RealTimeNotification
```

Update the notification:

```
public function toArray($notifiable)
{
    return [
        'message' => $this->message,
    ];
}

public function toBroadcast($notifiable)
{
    return new NotificationEvent($this->message, $notifiable->id);
}
```

Presence Channels (for online users)

```
public function broadcastOn()
{
    return new PresenceChannel('notifications.'.$this->userId);
}
```

Frontend:

```
window.Echo.join(`notifications.${userId}`)
    .here((users) => {
        console.log('Online users:', users);
    })
    .joining((user) => {
        console.log('User joined:', user);
    })
    .leaving((user) => {
        console.log('User left:', user);
    });
```

PROF

This implementation provides a complete real-time notification system that you can further customize based on your application's requirements.