<https://laravel.com/docs/12.x/broadcasting#pusher-channels>

composer require pusher/pusher-php-server

npm install --save laravel-echo pusher-js

php artisan install:broadcasting

.env

BROADCAST\_DRIVER=pusher

PUSHER\_APP\_ID=""

PUSHER\_APP\_KEY=""

PUSHER\_APP\_SECRET=""

PUSHER\_HOST=

PUSHER\_PORT=443

PUSHER\_SCHEME=https

PUSHER\_APP\_CLUSTER="ap1"

Make any event

<?php

namespace App\Events;

use Illuminate\Broadcasting\Channel;

use Illuminate\Broadcasting\InteractsWithSockets;

use Illuminate\Broadcasting\PresenceChannel;

use Illuminate\Broadcasting\PrivateChannel;

use Illuminate\Contracts\Broadcasting\ShouldBroadcast;

use Illuminate\Foundation\Events\Dispatchable;

use Illuminate\Queue\SerializesModels;

class NewLiveMessageEvent implements ShouldBroadcast

{

    use Dispatchable, InteractsWithSockets, SerializesModels;

    public $messages;

    public function \_\_construct( $messages)

    {

          $this->messages = $messages;

    }

    public function broadcastOn()

    {

        return [

            new Channel('live\_message'),

        ];

    }

   public function broadcastAs()

    {

        return 'live\_message.new';

}

  public function broadcastWith()

    {

        return [

            'data' => $this->messages,

        ];

    }

}

In controller

  public function guestSend(Request $request)

    {

        event(new NewLiveMessageEvent($message));

    }

lib/echo.ts or use useEcho

import Echo from 'laravel-echo';

import Pusher from 'pusher-js';

// Important: register pusher globally for Echo

window.Pusher = Pusher;

declare global {

    interface Window {

        Pusher: any;

    }

}

export const echo = new Echo({

    broadcaster: '',

    key: '',  // your pusher key

    cluster: 'ap1',              // your cluster

    forceTLS: true,

});

Usage in frontend react

import { echo } from '@/lib/echo';

useEffect(() => {

        // Subscribe to private channel

        const channel = echo.channel(`live\_message`);

        channel.listen('.live\_message.new', (e: any) => {

            fetchMessages(guestToken);

        });

 return ()=>{

               echo.leave('live\_message.new');// prevent double result

   channel.stopListening('live\_message.new'); // ✅ cleanup

        }

    }, [guestToken]);