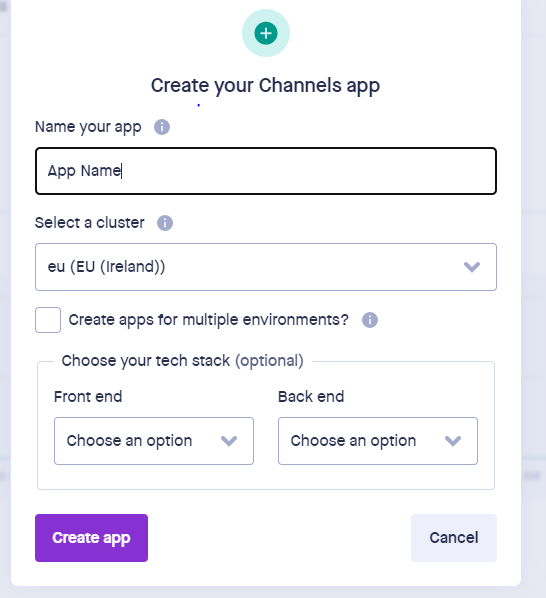
To Install Pusher:

1. Go to pusher site [pusher](https://pusher.com/) and click to Sign up to create account or Sign in to login if you have already account.
2. Create new app and fill the form data



* Your app name.
* Your cluster area
* Your frontend language or framework.
* Your backend language or framework.

1. Install pusher server && laravel echo

composer require pusher/pusher-php-server

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

1. Update your env file:

* Change **BROADCAST\_DRIVER** value to **pusher**.
* Add your

**PUSHER\_APP\_ID**

**PUSHER\_APP\_KEY**

**PUSHER\_APP\_SECRET**

**PUSHER\_APP\_CLUSTER**

**Ex =>**

PUSHER\_APP\_ID=1403548

PUSHER\_APP\_KEY=8a54692d8cd3a078d328

PUSHER\_APP\_SECRET=20a9db67b966ea5ba2ff

PUSHER\_HOST=

PUSHER\_PORT=443

PUSHER\_SCHEME=https

PUSHER\_APP\_CLUSTER=eu

VITE\_PUSHER\_APP\_KEY="${PUSHER\_APP\_KEY}"

VITE\_PUSHER\_HOST="${PUSHER\_HOST}"

VITE\_PUSHER\_PORT="${PUSHER\_PORT}"

VITE\_PUSHER\_SCHEME="${PUSHER\_SCHEME}"

VITE\_PUSHER\_APP\_CLUSTER="${PUSHER\_APP\_CLUSTER}"

* And all this values can get it from **App Keys** page in your pusher profile.

1. In config /app.php , uncomment this line

App\Providers\BroadcastServiceProvider::class

1. In **boot** method in **BroadcatServiceProvider** class append this lines:

        Broadcast::routes();

*require* base\_path('routes/channels.php');

1. In config/broadcasting.php change:

'options' => [

'cluster' = **PUSHER\_APP\_CLUSTER**,

'useTLS' => true

],

1. In ***resources/js/ bootstrap.js*** file uncomment for this code

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

*import* Pusher *from* 'pusher-js';

window.Pusher = Pusher;

window.Echo = new Echo({

    broadcaster: 'pusher',

    key: *import*.meta.env.VITE\_PUSHER\_APP\_KEY,

    cluster: *import*.meta.env.VITE\_PUSHER\_APP\_CLUSTER,

    wsHost: *import*.meta.env.VITE\_PUSHER\_HOST ?? `ws-${*import*.meta.env.VITE\_PUSHER\_APP\_CLUSTER}.pusher.com`,

    wsPort: *import*.meta.env.VITE\_PUSHER\_PORT ?? 80,

    wssPort: *import*.meta.env.VITE\_PUSHER\_PORT ?? 443,

    forceTLS: (*import*.meta.env.VITE\_PUSHER\_SCHEME ?? 'https') === 'https',

    enabledTransports: ['ws', 'wss'],

});

1. Then you can create event using this command

**php artisan make:event {MyEven}**

1. Channels type

- Channel class to create public channel => (my-channel)

- PrivateChannel class to create private channel => (my-channel.{UserID})

1. In master blade must include public/js/app.js file to can use window.Echo
2. Run this command **npm run watch** to make compile for app.js
3. To make listen using this code

 window.Echo.channel(`private-my-channel.${userID}`)

     .listen('MyEvent', (data) => {

alert(data);

     });

1. For private channel must make channel to return auth details in routes/channel.php

Broadcast::channel('my-channel.{id}', function ($user, $id) {

*return* (int) $user->id == (int) $id ? $user : null;

});

Must return user details not true or false

1. For public channel not need to create channel route because broadcast for all users.
2. This example for create **MessageCreated** Event that have private channel new-message with append user id

<?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 MessageCreated implements ShouldBroadcast

{

    use Dispatchable, InteractsWithSockets, SerializesModels;

*/\*\**

*\* Create a new event instance.*

*\**

*\* @return void*

*\*/*

    public function \_\_construct(public Message $message, public int $user\_id)

    {

*//*

    }

*/\*\**

*\* Get the channels the event should broadcast on.*

*\**

*\* @return \Illuminate\Broadcasting\Channel|array*

*\*/*

    public function broadcastOn()

    {

*return* new PrivateChannel("new-message.{*$this*->user\_id}");

    }

}

1. Example for fire event MessageCreated

                broadcast(new MessageCreated($Message, $recevier\_id));

1. Example for event listen in blade

        <script>

            $(function() {

const AUTH\_USER\_ID = {{ auth()->id() }};

                Echo.channel(`private-new-message.${AUTH\_USER\_ID}`)

                .listen('MessageCreated', (data) => {

                    alert(data);

                });

            });

        </script>

1. How to detect user is online or Ofline

window.Echo.join(`chat`)

           .joining((user) => { *// This user is join to chat page*

               console.log(user);

           })

           .leaving((user) => { *// This user is leaving to chat page*

               console.log(user);

           })

1. And add this channel to get user details

Broadcast::channel('chat', function ($user) {

*return* $user;

});

1. To use typing

- First must make enable for pusher client events from **App Setting** and make enable to **Enable client events.**

- For char channel in last step add new listener on **whisper typing event**

    let chatChannel = window.Echo.join(`chat`)

                            .listenForWhisper('typing', (e) => {

                                If (e.typing)

Do something when typing

Else

Do something after stop

                            });

And save the result in object to use it in another place.

1. Make js event on input and use the last object to make event **whisper typing**

    let time = false;

    $('body').on('keydown', '[name="message"]', function(){

        chatChannel.whisper('typing', {

            typing: true,

        });

*if* (time) clearTimeout(time);

        time = setTimeout( () => {

            chatChannel.whisper('typing', {

                typing: false,

            });

        }, 600);

    });

**Client Events**

1. In your pusher dashboard => App Settings => Enable client events [make check] to cant listen to client events
2. To create client event with listen

* Create global variable to have channel with the event listen.

Ex => create typing listen

let clientEvents = window.Echo.join(`chat`)

                   .listenForWhisper('typing'(e) => {

                      alert(e); *// the object e have all data that coming from event*

                   })

* To make fire for this event do this code when user make key down in input.

            data = {}; *// any data you want to send it to listener*

            clientEvents.whisper('typing', data);

* joining && leaving is events in pusher to get the data of user when make leave the browser or when join

let clientEvents = window.Echo.join(`chat`)

           .joining((user) => { *// This user is join to chat page*

*// you can display the online icon for this user*

           })

           .leaving((user) => { *// This user is leaving to chat page*

*// you can hidden the online icon for this user because he is offline now*

           })

* Events to check if user is read the message or receive without read.

Let clientEvents = window.Echo.join(`chat`)

.listenForWhisper('seen-message', (e) => {

alert(e);

            })

            .listenForWhisper('receive-message', (e) => {

alert(e);

            });

* To change message status from sent to receive or read must create 2 events with listeners.

1. Event to make receive message

data = {}; *// any data you want to send it to listener*

clientEvents.whisper('receive-message', data);

1. Event to read message

data = {}; *// any data you want to send it to listener*

clientEvents.whisper('read-message', data);

And this event will call in multi places

* When user make joining to page then message will update from sent to receive.

let clientEvents = window.Echo.join(`chat`)

           .joining((user) => { *// This user is join to chat page*

data = {}; *// any data you want to send it to listener*

         clientEvents.whisper('receive-message', data);

           });

* When user already is online, here have 2 cases on event when receive data from pusher

Echo.channel(`private-new-message.${AUTH\_USER\_ID}`)

               .listen('MessageCreated', (data) => {

// first case : if not open chat window, update to receive

data = {}; *// any data you want to send it to listener*

         clientEvents.whisper('receive-message', data);

// second case: if open chat window, update to read message

                    data = {}; *// any data you want to send it to listener*

          clientEvents.whisper('read-message', data);

               });