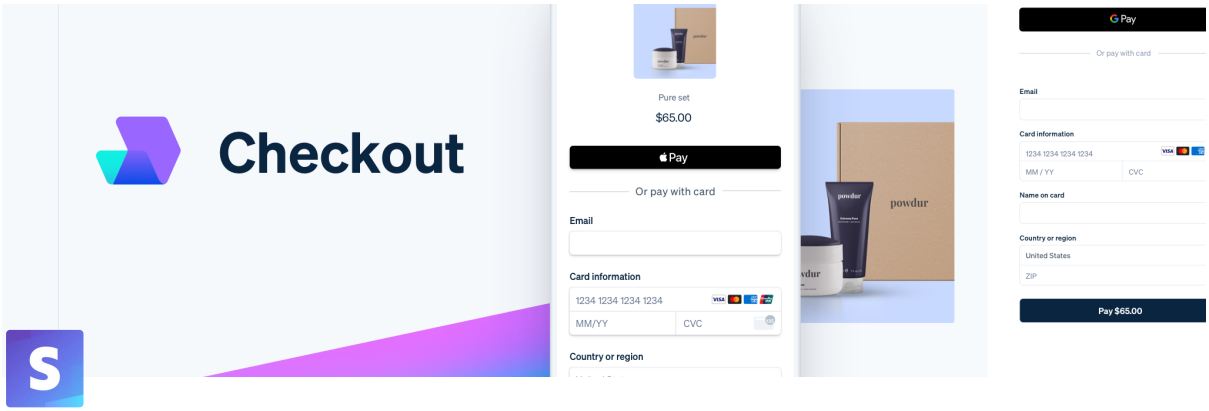


Stripe with React & NodeJS

Last Edited: 16/10/2021

Author: Tham Kei Lok

keilokimnida - Overview

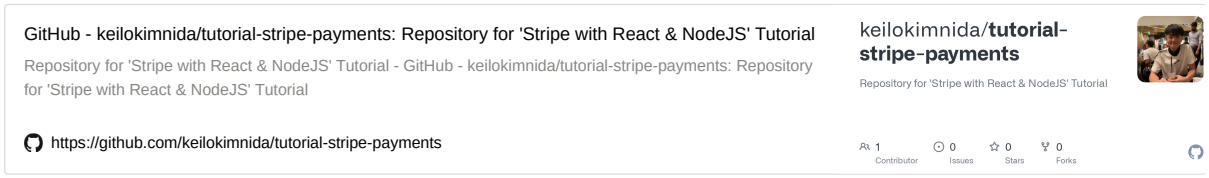
Info Tech @ Singapore Polytechnic. keilokimnida has 4 repositories available. Follow their code on GitHub.

https://github.com/keilokimnida

Table of Contents

1 Introduction

This document serves as a guide on integrating **Stripe with React & NodeJS** applications for **payment functionalities**. Stripe is an **online payment gateway** that lets people accept and send money over the internet. This guide will provide a top level view on **subscription plans** and **one-time payment** features! If you wish to try out/view the codes for the payment features shown in this guide, visit the embedded link below to download the repository!



Prerequisites

- Intermediate knowledge in NodeJS & React

Learning Objectives

- One time credit card payment
- Saving credit/debit cards for future payments
- Subscription Billing with saved credit card

Have fun! =)

2 Setting up

Here's an overview of tasks to start using Stripe

1. Setup Stripe Account and grab API keys
2. Install Stripe API for backend and paste test secret API key

3. Install Stripe packages for frontend and paste test publishable API key

Step 1: Setup Stripe account

Having a Stripe account will grant you access to Stripe Dashboard which is used to **monitor and test payment transactions**. It will also contain your secret and public API keys which will be used to authenticate your API requests.

Step 1.1: Create Stripe Account

- If you do not have a Stripe account set up yet, **sign up for an account** via this link.

Step 1.2: Verify Stripe Account

- Once an account has been created, Stripe will send a **verification email** to the email you have signed up with.
- Look for the verification mail and **click 'verify account'**

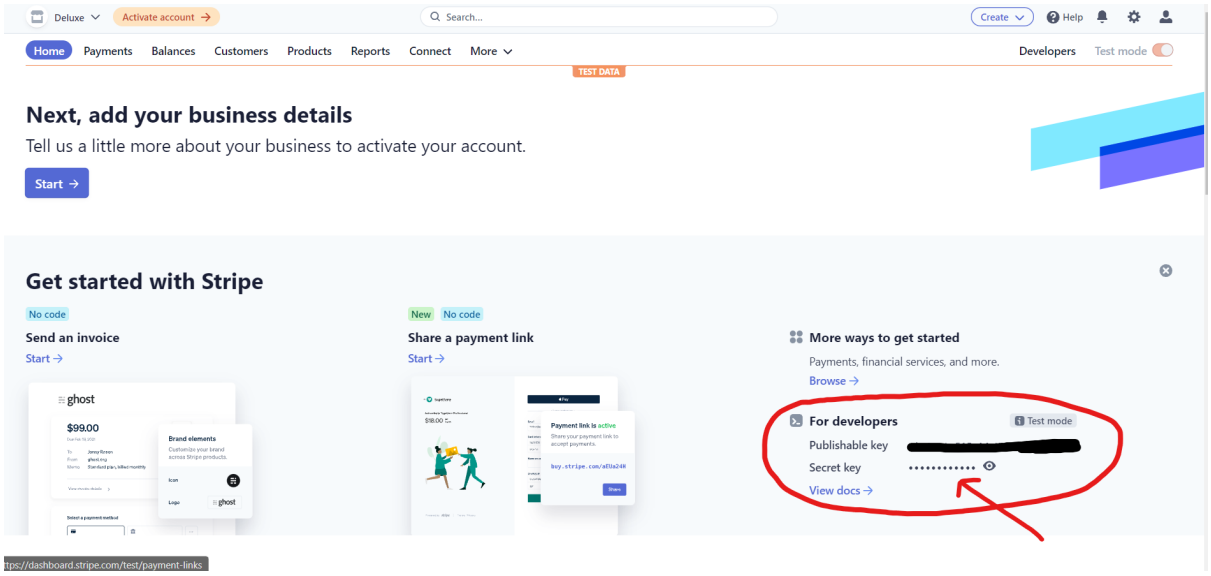
Step 2: Setup Server-side

Step 2.1 Install the Stripe Node library

```
npm install --save stripe
```

Step 2.2 Copy Test Secret API key from Stripe dashboard

- The test secret API key can be found on 'Home' page of your Stripe dashboard



Step 2.3 Paste Test Secret API key in '.env' file

- Stripe provides keys for accessing their services, but mentioning these keys in the code is a **security threat** to the entire application
- Therefore, a **'.env'** file is created which includes all the secret keys in it

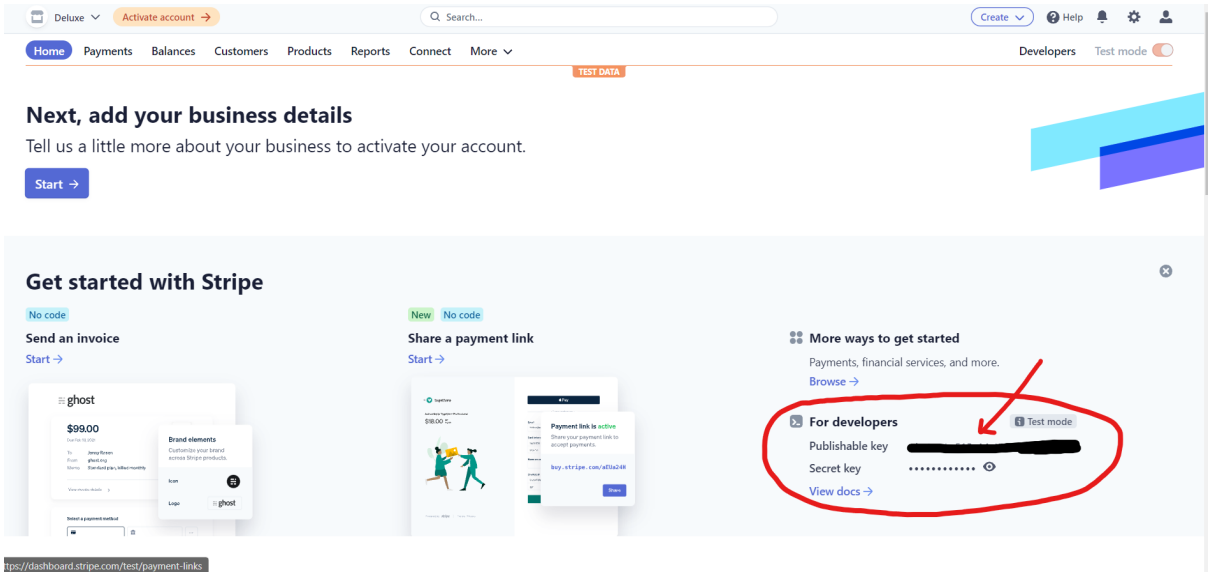
```
STRIPE_TEST_SK=sk_test_somethingsecretejworjwoierjwoeijroiwejriwjeorwejrjwejrojweriowjer
```

Step 3: Setup Client-side

Step 3.1 Add Stripe to your React App

```
npm install --save @stripe/react-stripe-js @stripe/stripe-js
```

Step 3.2 Copy Test Publishable API key from Stripe dashboard



Step 3.3 Paste Test Publishable API key in '.env' file

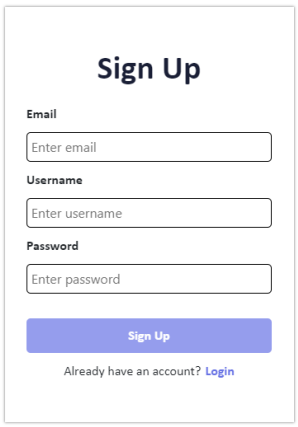
- Stripe provides keys for accessing their services, but mentioning these keys in the code is a **security threat** to the entire application
- Therefore, a **'.env'** file is created which includes all the secret keys in it

```
REACT_APP_BASEURL_STRIPE_PK_TEST=pk_test_randomabcdefghijklmnnopqrstuboerkpowkrpwekrpoewkrpweorkrpo
```

All set!

---

3 Account Registration



Sign Up Page

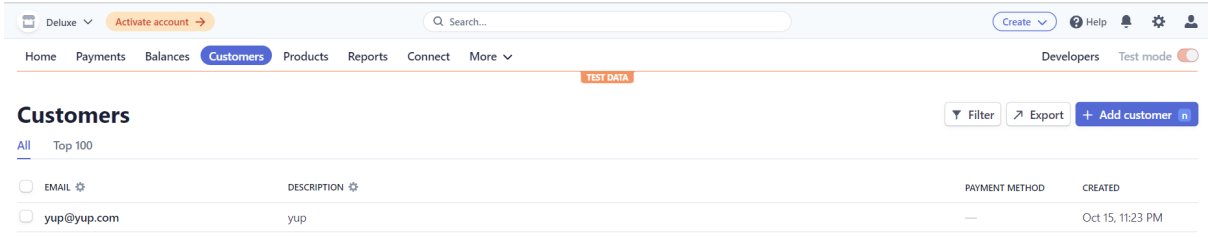
- When users register for an account in our system, besides inserting a record in our database, we should also create a **'Customer' Object** in Stripe's Database too.
- This is so that we can **associate payment information** (e.g. payment method, subscription) **to a user**.

Create a customer object in Stripe API

```
// Create customer object
const customer = await stripe.customers.create({
  email,
  name
});
```

Upon successful signup,

- You should be able to see customer object in Stripe's database



- and in your own database

account_id	username	email	trialed	stripe_customer_id	stripe_payment_intent_id	stripe_payment_intent_client_secret	created_at	updated_at
19	yup	yup@yup.com	1	cus_KPhfGX8MR7Ufpr	NULL	NULL	2021-10-15 15:23:28	2021-10-16 04:46:35

- You should also store the Stripe customer id in your account table as it will be used to identify who the user is

---

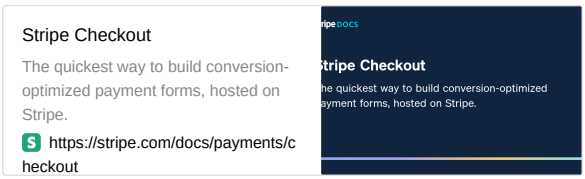
4 Collecting card information on the client-side

4.1 Stripe Checkout and Stripe Elements

Stripe offers 2 types of methods for the collection of card information

- **Stripe Elements** - **Custom payment form**, more flexibility, but harder to implement
- **Stripe Checkout** - **Hosted Checkout page by Stripe**, lesser flexibility, easier to implement

Please refer to the links below for more information.



4.2 How to set up Stripe Elements with React

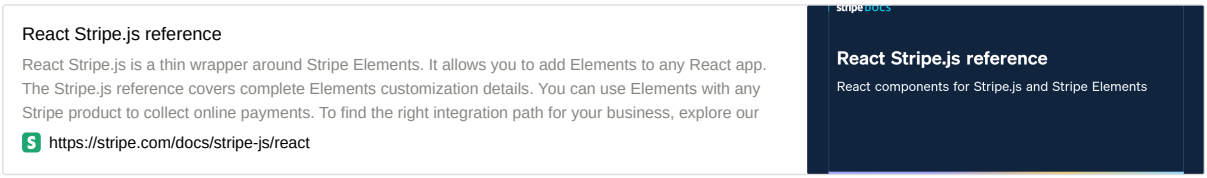
- The Elements provider allows you to use Element components and access the Stripe object in any nested component.
- Render an Elements provider at the **root of your React app** so that it is available everywhere you need it

- To use the Elements provider, call **loadStripe** from @stripe/stripe-js with your publishable key.
- Pass the resulting promise from **loadStripe** to the **Elements provider**.
- This allows the child components to access the Stripe service via the **Elements consumer**.

```
// index.js
import Routes from './Routes';
import { loadStripe } from "@stripe/stripe-js";
import { Elements } from "@stripe/react-stripe-js";

const stripePKTest = {{TEST_PRIMARY_KEY_TO_BE_INSERTED}};
const promise = loadStripe(stripePKTest);

ReactDOM.render(
    <Elements stripe={promise}>
      <Routes />
    </Elements>,
  document.getElementById('root')
);
```

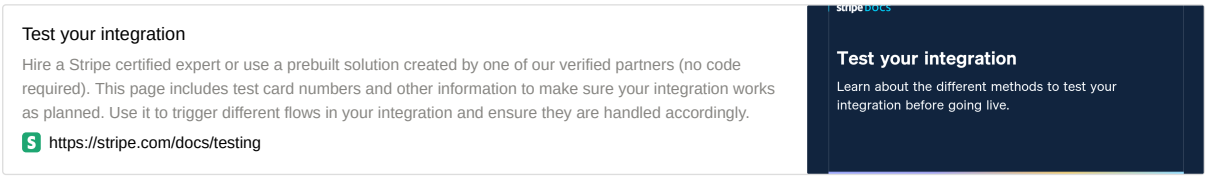
- With that, you have successfully set up Stripe Elements for your React application
- You can now use components from Stripe Elements

For more information, visit



5 How will testing be done?

- You will not be required to bring your system to production mode to test the system
- Ensure that your Stripe dashboard is in "Test Mode" when doing testing
- Live API keys will only be used in production, so ensure that you are using Test API keys
- The card number that you will be using most often is `4242 4242 4242 4242`, this is a test card that will always result in a payment success.
- Refer to the link below for a list of test cards



6 Accept One time Credit Card Payment (Custom flow)

6.1 Developer's Point of View

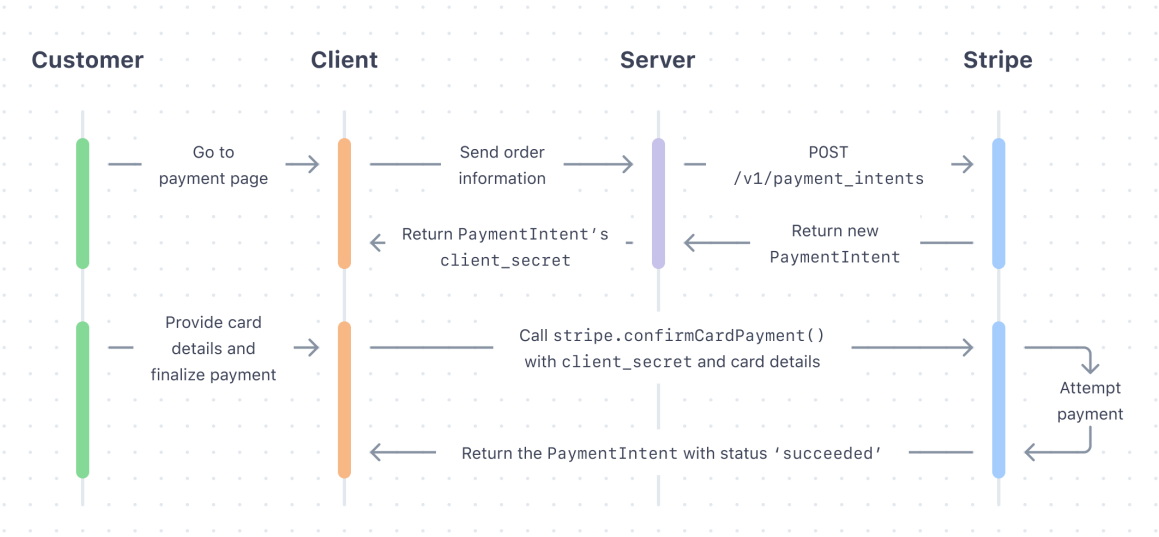
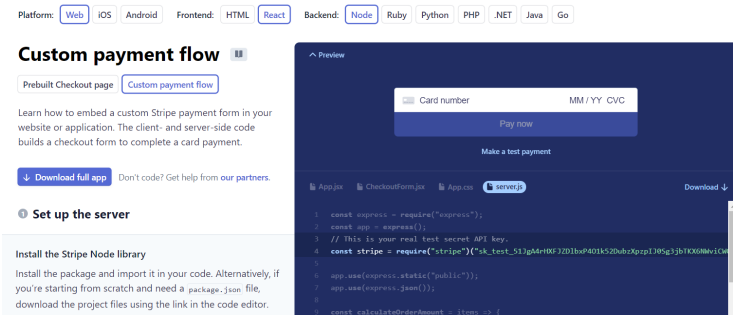


Diagram taken from Stripe's website

6.2 How to get started?

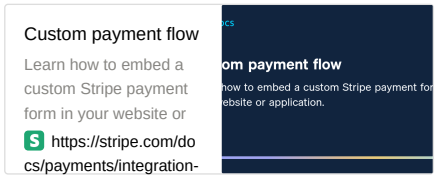
Here are some ways you can learn how to implement one time payment

- Follow Step by Step instructions on Stripe's website



Stripe's Sample Integration

Please visit the link here to visit the page



- View sample integration on GitHub Repository

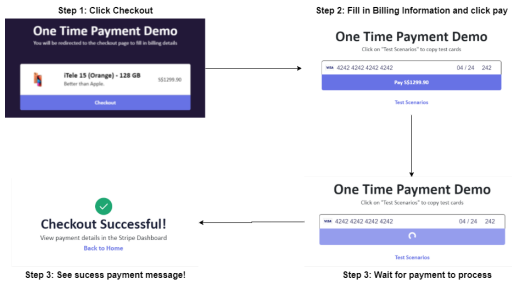
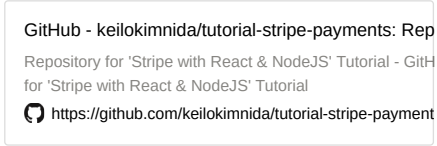


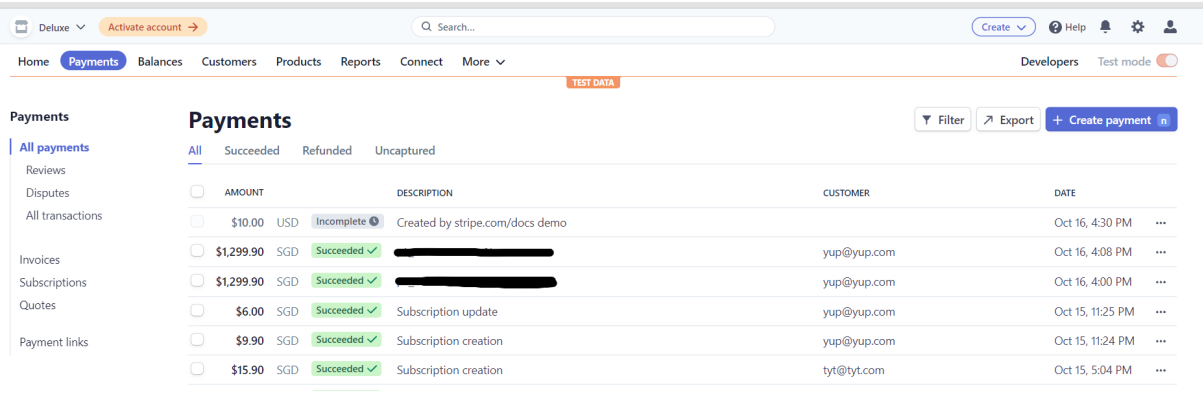
Diagram of One time payment (Customer's POV)

Please visit the link here to download the repository



6.3 View payments

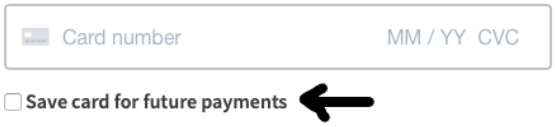
- You can view and manage payments in Stripe dashboard 'payments' tab



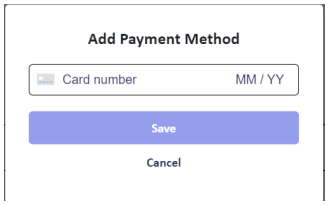
7 Setup Credit/Debit Card for Future Usage

There are **two ways** credit cards can be saved for **future usage**,

- When a user is **confirming card payment** for a purchase and/or,
- **Manually set up** a card for future payments



Save card during payment



Manual set up

7.1 Save Card during Payment

- This is an additional feature for one time payment
- Ensure that the customer is aware that the card will be saved for future usage upon payment
- Add `setup_future_usage` parameter when confirming card payment on the client side
- Possible values for `setup_future_usage` include
  - 'on_session'
    - Use '**on_session**' if you intend to only reuse the payment method when your customer is **present in your checkout flow**.
  - 'off_session'
    - Use '**off_session**' if your customer **may or may not be in your checkout flow**.

```
// Sample implementation

const Checkout = () => {

  ...

  const handleFormSubmit = async () => {

    ...

    const paymentIntent = await stripe.confirmCardPayment(clientSecret, {
      payment_method: {
        card: card,
      },
      setup_future_usage: 'off_session' // <-- add this line of code,
    });
```

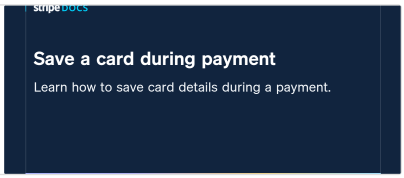
```
    ...

  };

  return (
    ...
  );
};

export default Checkout;
```

For more information, visit

Save a card during payment

After creating a PaymentIntent on the server and passing its client secret to the browser, you're ready to collect card information with Stripe Elements on your client. Elements is a set of prebuilt UI components for collecting and validating card number, ZIP code, and expiration date.

https://stripe.com/docs/payments/save-during-payment



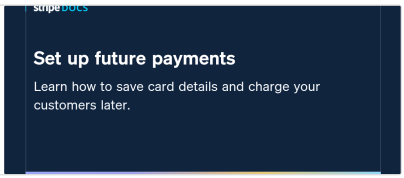
7.2 Manual Setup for Future Payments

- Please visit the link below to find out how to set up card manually

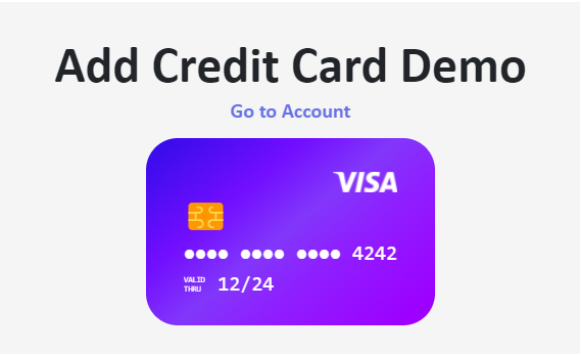
Set up future payments

After a customer successfully completes their Checkout Session, you need to retrieve the Session object. There are two ways to do this: Asynchronously: Handle checkout.session.completed , which contain a Session object. Learn more about setting up webhooks. Synchronously: Obtain the sessionId from the URL

https://stripe.com/docs/payments/save-and-reuse



- Alternatively, you can view a sample integration on GitHub Repository



Add Credit Card Demo on Github Repository

Please visit the link here to download the repository

GitHub - keilokimnida/tutorial-stripe-payments: Repository for 'Stripe w

Repository for 'Stripe with React & NodeJS' Tutorial - GitHub - keilokimnida/tuto for 'Stripe with React & NodeJS' Tutorial

https://github.com/keilokimnida/tutorial-stripe-payments

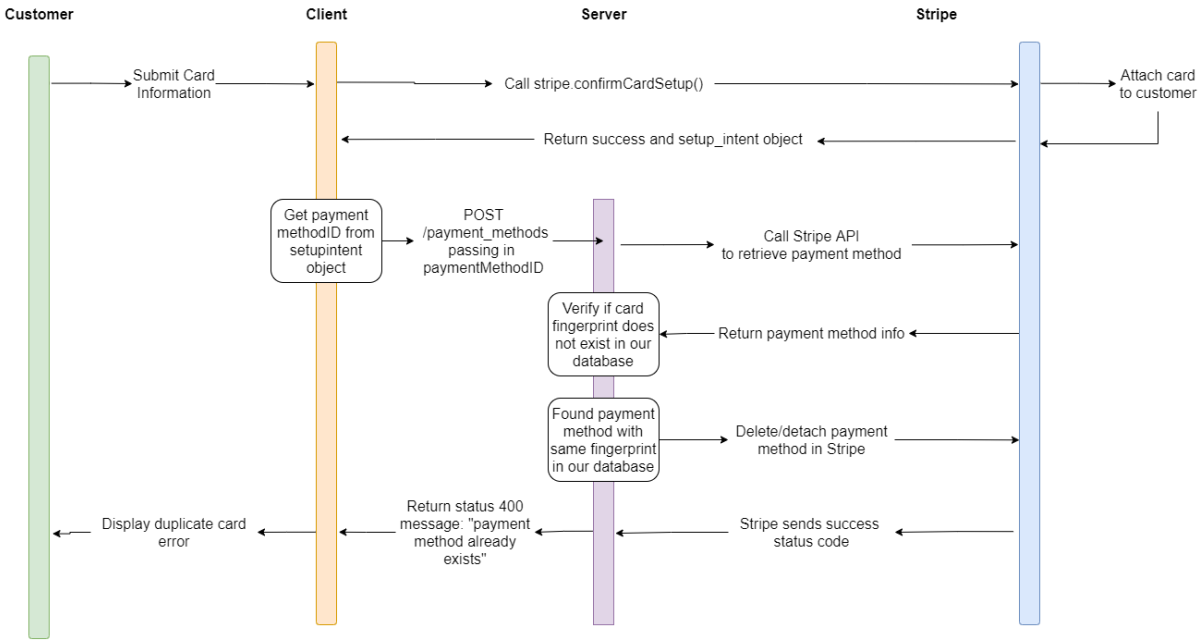
7.3 Potential Issue: Duplicate Cards for each Customer

- By default, when customer attempts to add a card, Stripe does not check whether it already exists as a payment method for the customer.
- This is a known issue to Stripe https://github.com/stripe/stripe-payments-demo/issues/45 and will probably not be fixed/patched.
- If this is not the behavior you wish to implement, you can consider the following workaround.

7.3.1 Workaround to prevent duplicate cards

- You can find the code for this implementation in the GitHub Repository here

Scenario: User is attempting to add an existing attached card as a payment method



Apologies for the poor quality diagram

8 Subscription Plans

8.1 How to get started

Recommended steps to learn how to implement subscription plans.

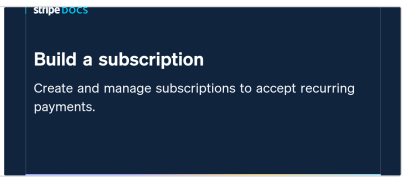
Step 1: Following step by step tutorial

- Build a subscription integration based on Stripe's step by step tutorial
- The tutorial will teach you how to implement a simple monthly subscription

Build a subscription

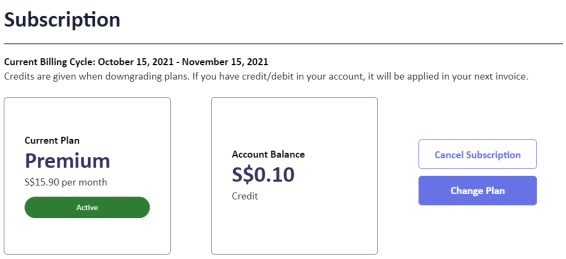
Checkout is used to collect payment method information and initiate Subscriptions. This guide shows you how to get started with a basic Subscription. After you complete this integration, you can extend Checkout to display taxes, apply discounts, and more. Install the Stripe client of your choice: Install the Stripe CLI

https://stripe.com/docs/billing/subscriptions/build-subscription



Step 2: View Sample Integration

- View sample integration in GitHub Repository



Please visit the link here to download the repository

GitHub - keilokimnida/tutorial-stripe-payments: Repository for 'Stripe w

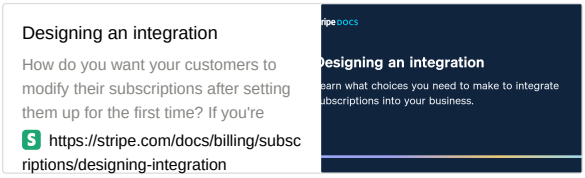
Repository for 'Stripe with React & NodeJS' Tutorial - GitHub - keilokimnida/tuto for 'Stripe with React & NodeJS' Tutorial

https://github.com/keilokimnida/tutorial-stripe-payments

- **Subscription model:** Freemium, monthly payment only
- **Free trial:** Yes, 7 days free trial of premium plan for first time subscribers
- **Types of plans:**
  - **Standard** - S$9.90
  - **Premium** - S$15.90
- **Refund policy:** No refunds
- **Upgrading plan:** Charged at a prorated amount immediately
- **Downgrade plan:** Given prorated credits immediately which will be used to discount the next invoice

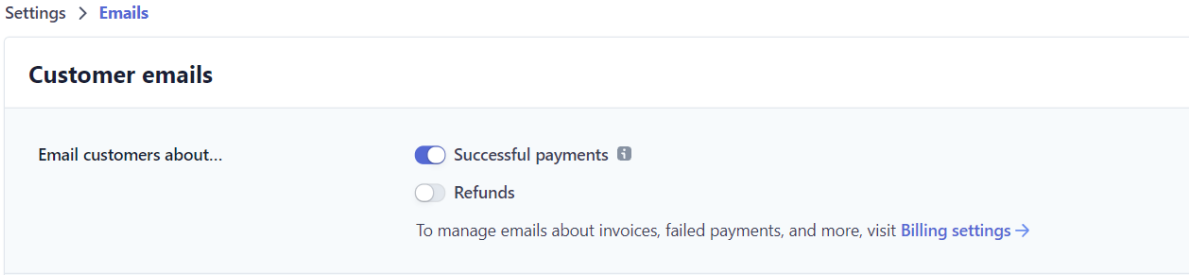
Step 3: Decide which Pricing model best suits your needs

Visit the links below to find out the different subscription and pricing models

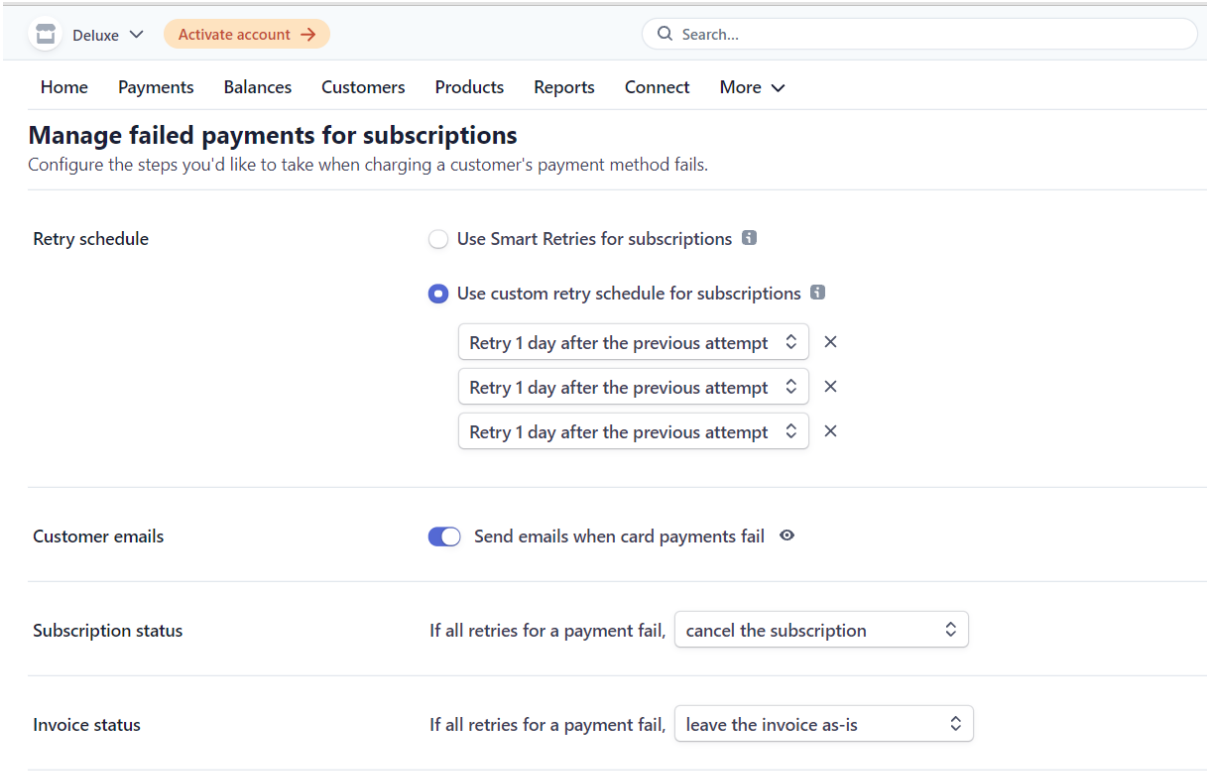


8.2 Sending emails to inform customer successful and failed payments

- Under https://dashboard.stripe.com/settings/emails, there is a option to email customer when there is a successful payment

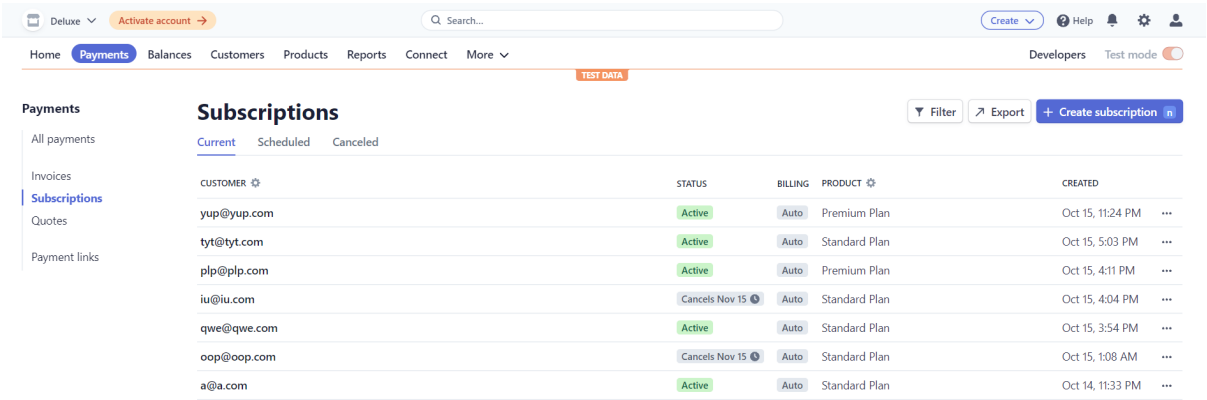


- Under https://dashboard.stripe.com/settings/billing/automatic, there are options to send email to customer when there are failed payments



8.3 View and Manage Subscriptions

Subscriptions can be viewed and managed under "payments/subscriptions" tab in Stripe Dashboard

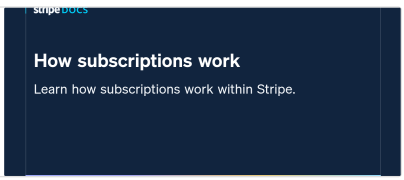


8.4 Handling failed recurring payments

How subscriptions work

To simplify handling failed payments pass payment_behavior=default_incomplete when creating a subscription. If the subscription requires payment it's created with status incomplete, otherwise the subscription immediately becomes active. Activate an incomplete subscription by paying the first invoice.

https://stripe.com/docs/billing/subscriptions/overview#how-payments-work-subscriptions

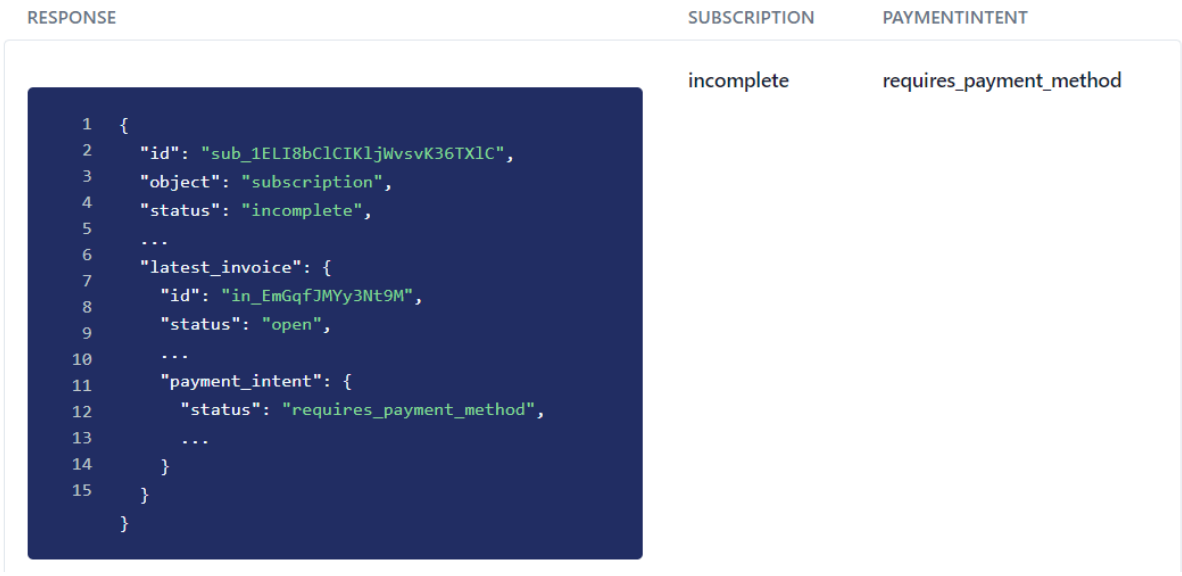


There are two types of error for recurring charge error

- Requires payment method
- Requires action

8.4.1 Requires payment method

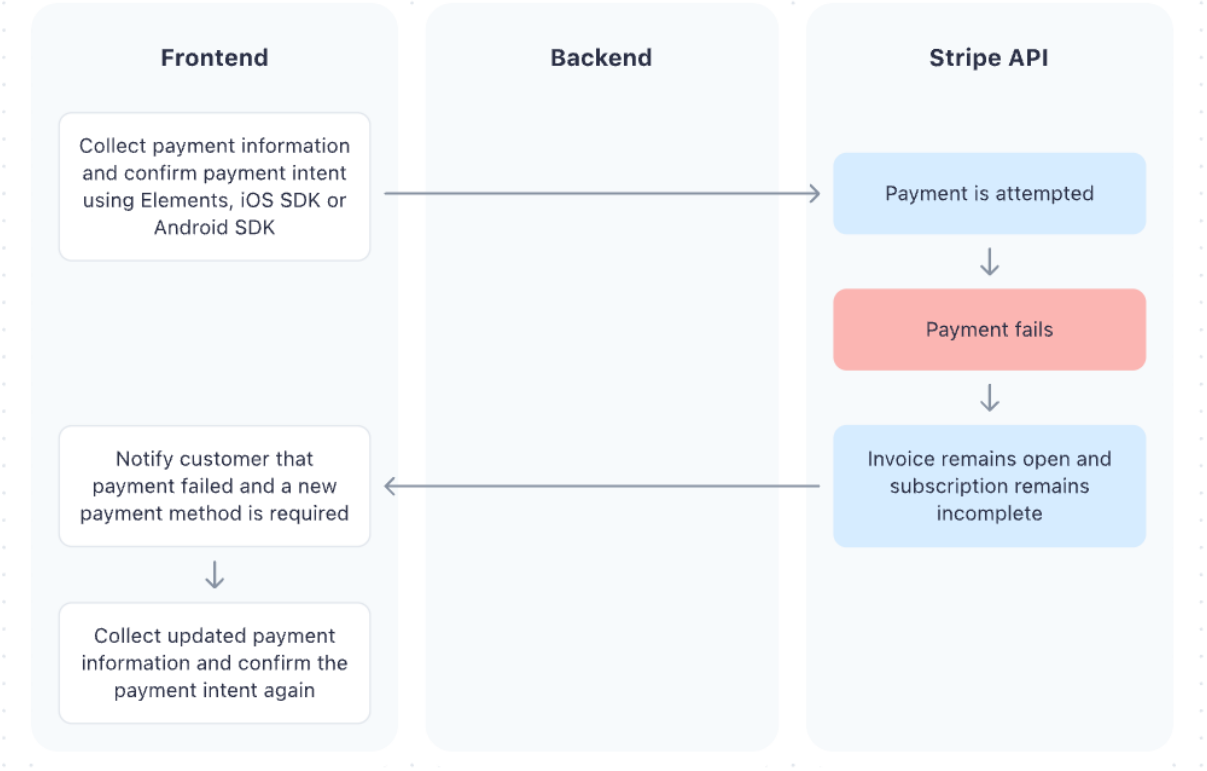
If payment fails because of a card error such as a decline, the status of the PaymentIntent is `requires_payment_method` and the subscription is incomplete.



Screenshot taken from Stripe's website

To resolve these scenarios:

- Notify the customer.
- Collect new payment information and **confirm** the payment intent.
- Update the **default payment method** on the subscription.



Screenshot taken from Stripe's website

8.4.2 Requires Action

Some **payment methods** require authentication to complete the payment process. When this happens, the status of the PaymentIntent is `requires_action` and 3D Secure completes the authentication process. Whether or not a payment method requires authentication is based on your **Radar rules** and the issuing bank for the card.

Regulations in Europe often require 3D Secure. See **Strong Customer Authentication** to determine whether handling this status is important for your business. If you have an existing billing integration and want to add support for this flow, also see the **Billing SCA Migration guide**.

Screenshot taken from Stripe's website

To handle these scenarios:

- Notify the customer that they must authenticate.
- Complete authentication using **stripe.ConfirmCardPayment**.

8.5 Upgrade and downgrade subscriptions

- Change a subscription often results in a proration, which is a feature that Stripe offers
- Read the links below to find out prorations in Stripe

https://stripe.com/docs/billing/subscriptions/prorations

https://stripe.com/docs/billing/subscriptions/prorations

Upgrade and downgrade subscriptions

This guide focuses on using the Subscriptions API to manage customer subscriptions. You can instead implement

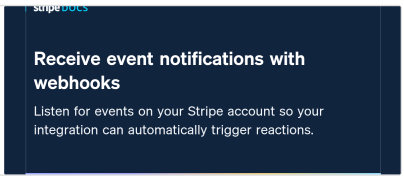
https://stripe.com/docs/billing/subscriptions/upgrade-downgrade

9 Webhooks

Receive event notifications with webhooks

Stripe uses to notify your application when an event happens in your account. Webhooks are particularly useful for asynchronous events like when a customer's bank a payment, a customer disputes a charge, or a recurring payment succeeds. Begin using webhooks with your Stripe integration in just three steps: Create a

https://stripe.com/docs/webhooks



9.1 What are Webhooks?

- Stripe uses Webhooks to notify your application when an **event** happens in your account.
- Webhooks are particularly useful for asynchronous **events** like when a customer's subscription change, a customer disputes a charge, or a recurring payment succeeds.
- Refer to the link below to see the types of events you can listen to,

Stripe API reference - Events - Node

Complete reference documentation for the Stripe API. Includes code snippets and examples for our Python, Java, PHP, Node.js, Go, Ruby, and .NET libraries.

https://stripe.com/docs/api/events?lang=node

9.2 How to Listen to Webhook Events?

9.2.1 Download ngrok

- To listen to webhook events, you will need to set up a ngrok connection to enable our local development server to be exposed to the internet
- Sign up for an account and download ngrok

ngrok - download

Running this command will add your authtoken to your ngrok.yml file. Connecting an account will list your open tunnels in the dashboard, give you longer tunnel timeouts, and more. Visit the dashboard to get your auth token.

https://ngrok.com/download

- Once you have downloaded ngrok, a zipped file will be installed
- Unzip the file

9.2.2 Configure system settings to allow ngrok to run anywhere

- Right now, your system will not recognize what ngrok.exe is, hence you need to tell your system what ngrok.exe is
- Refer to the links below on how to configure this

ngrok command not found

Asked Want to improve this question? Update the question so it's on-topic for Stack Overflow. Closed 6 months ago.

https://stackoverflow.com/questions/30188582/ngrok-command-not-found

Solution for Mac devices

How to Fix "not recognized as an internal or external command" in Wi

One of the great things about Windows is that you can get many of your tasks d from the Command Prompt on your machine. You just need to enter cmd.exe a Windows will run it for you. But occasionally, you might come across errors like

https://helpdeskgeek.com/how-to/fix-not-recognized-as-an-internal-or-extern mmand/

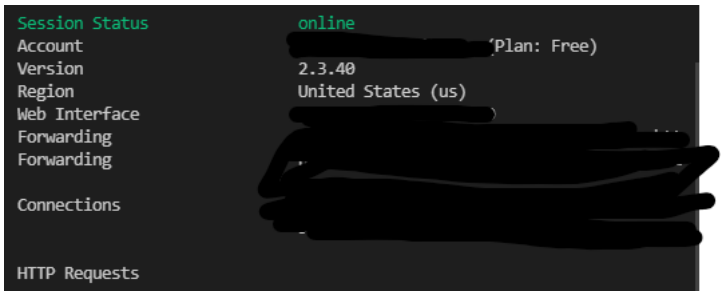
Solution for Windows devices

9.2.3 Start a ngrok connection

- Open up command prompt/terminal and type and enter

```
ngrok http {{SERVER_PORT_NUMBER}}
```

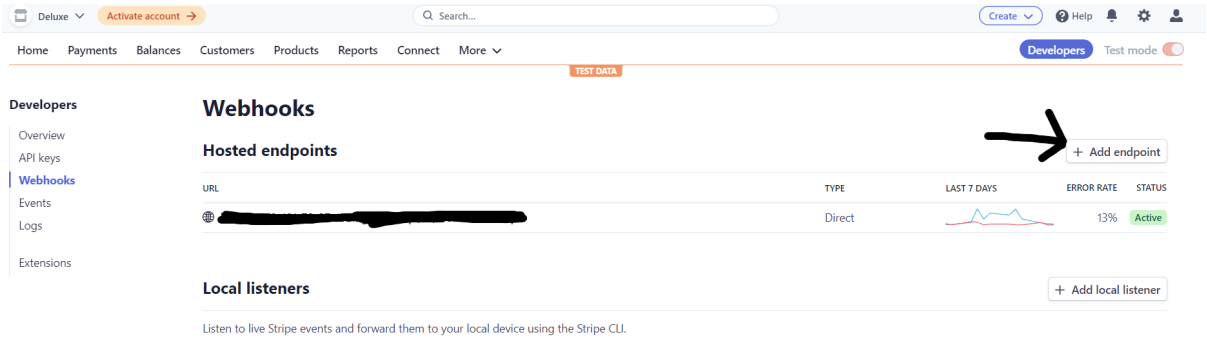
- You should see something like this



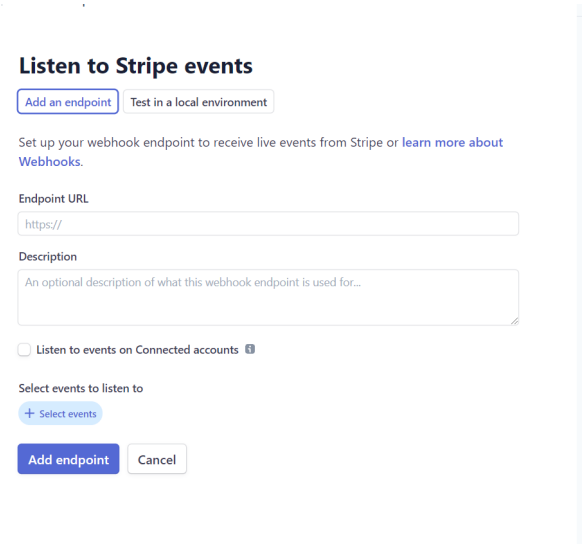
- Take note of the link that ends in **.ngrok.io** as it will be used later

9.2.4 Set up webhook in Stripe Dashboard

- Under 'Developers/Webhooks' tab in Stripe Dashboard, click on 'add endpoint'

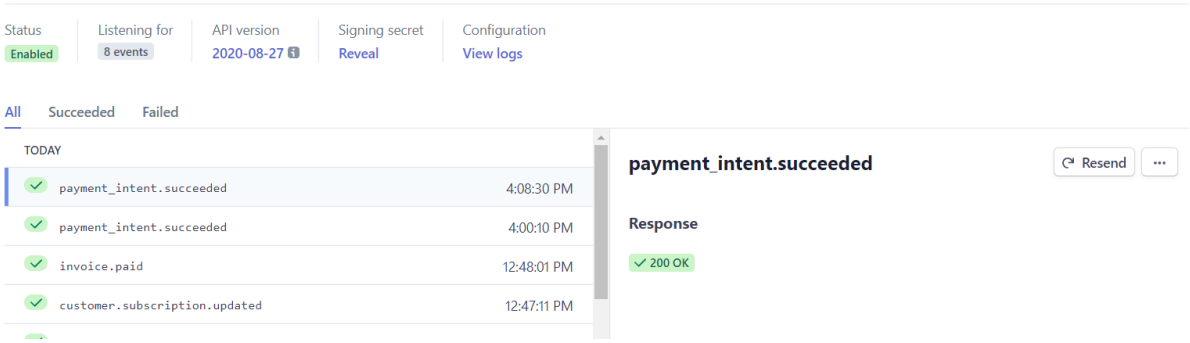


- Fill in the information
- Your Endpoint URL should be "http://something.ngrok.io/{{WEBHOOK_ENDPOINT}}
- Select the events you would like to listen to
- Click on add endpoint when done

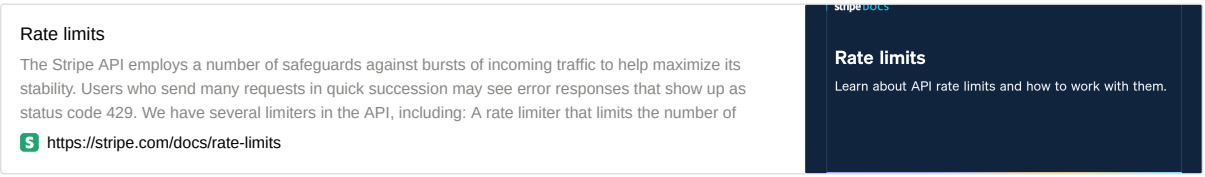


9.2.5 Monitor your webhook events

- With that, you're all set.
- Now you can listen to events via the Stripe Dashboard user interface



10 Rate limits



The Stripe API employs a number of safeguards against bursts of incoming traffic to help maximize its stability. Users who send many requests in quick succession may see error responses that show up as status code `429`. There are several limiters in the API, including:

- A **rate limiter** that limits the number of requests received by the API within any given second.

  For most APIs, Stripe allows up to 100 read operations per second and 100 write operations per second in live mode, and 25 operations per second for each in test mode.

  For the **Files API**, Stripe allows up to 20 read operations per second and 20 write operations per second. Live mode and test mode limits are separate and equal.

- A **concurrency limiter** that limits the number of requests that are active at any given time. Problems with this limiter are less common compared to the request rate limiter, but it's more likely to result in resource-intensive, long-lived requests.

Important: To prevent your system from hitting the rate limit, you should store information that are frequently accessed in your own database.

Thank you for reading! Hope you learnt something from this guide! ^^