

# Stripe with React & NodeJS

**Last Edited:** 16/10/2021

**Author:** Tham Kei Lok, Team 1 Project INC

keiloktql - Overview

Software Engineer Intern @ GovTech Singapore | Info Tech @ SP - keiloktql

<https://github.com/keiloktql>



## Table of Contents

Table of Contents

[1 Introduction](#)

[2 Setting up](#)

[Step 1: Setup Stripe account](#)

[Step 1.1: Create Stripe Account](#)

[Step 1.2: Verify Stripe Account](#)

[Step 2: Setup Server-side](#)

[Step 2.1 Install the Stripe Node library](#)

[Step 2.2 Copy Test Secret API key from Stripe dashboard](#)

[Step 2.3 Paste Test Secret API key in '.env' file](#)

[Step 3: Setup Client-side](#)

[Step 3.1 Add Stripe to your React App](#)

[Step 3.2 Copy Test Publishable API key from Stripe dashboard](#)

[Step 3.3 Paste Test Publishable API key in '.env' file](#)

[3 Account Registration](#)

[4 Collecting card information on the client-side](#)

[4.1 Stripe Checkout and Stripe Elements](#)

[4.2 How to set up Stripe Elements with React](#)

[5 How will testing be done?](#)

[6 Accept One time Credit Card Payment \(Custom flow\)](#)

[6.1 Developer's Point of View](#)

[6.2 How to get started?](#)

[6.3 View payments](#)

[7 Setup Credit/Debit Card for Future Usage](#)

[7.1 Save Card during Payment](#)

[7.2 Manual Setup for Future Payments](#)

[7.3 Potential Issue: Duplicate Cards for each Customer](#)

[7.3.1 Workaround to prevent duplicate cards](#)

## 8 Subscription Plans

### 8.1 How to get started

[Step 1: Following step by step tutorial](#)

[Step 2: View Sample Integration](#)

[Step 3: Decide which Pricing model best suits your needs](#)

### 8.2 Sending emails to inform customer successful and failed payments

### 8.3 View and Manage Subscriptions

### 8.4 Handling failed recurring payments

[8.4.1 Requires payment method](#)

[8.4.2 Requires Action](#)

### 8.5 Upgrade and downgrade subscriptions

## 9 Webhooks

### 9.1 What are Webhooks?

### 9.2 How to Listen to Webhook Events?

[9.2.1 Download ngrok](#)

[9.2.2 Configure system settings to allow ngrok to run anywhere](#)

[9.2.3 Login to ngrok on your computer](#)

[9.2.4 Start a ngrok connection](#)

[9.2.5 Set up webhook in Stripe Dashboard](#)

[9.2.6 Monitor your webhook events](#)

## 10 Rate limits

---

# 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!

GitHub - keilokimnida/tutorial-stripe-payments: Repository for 'Stripe with React & NodeJS' Tutorial


Repository for 'Stripe with React & NodeJS' Tutorial - GitHub - keilokimnida/tutorial-stripe-payments: Repository for 'Stripe with React & NodeJS' Tutorial

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

keiloktql/**tutorial-stripe-payments**

Repository for 'Stripe with React & NodeJS' Tutorial

2 Contributors 0 Issues 0 Stars 0 Forks



## 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'**

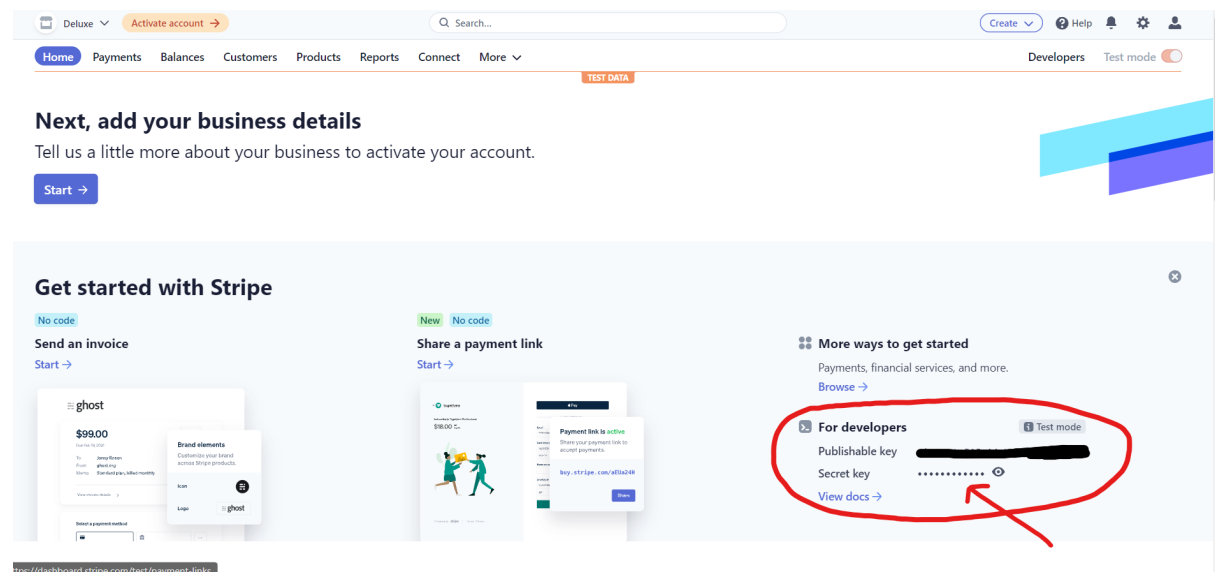
## Step 2: Setup Server-side

## 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

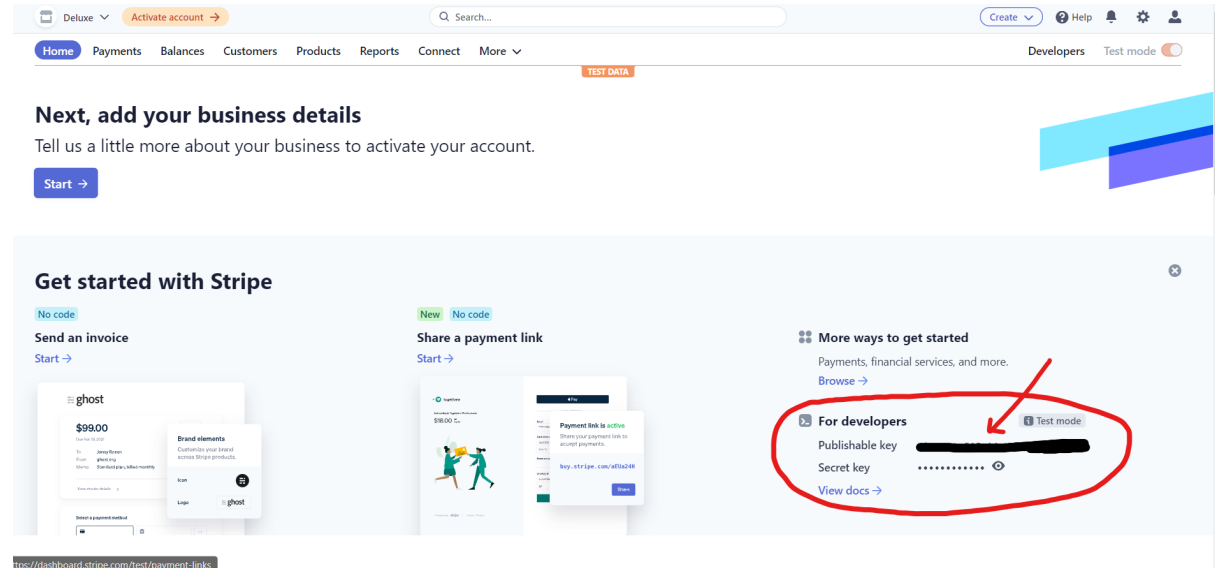
STRIPE\_TEST\_SK=sk\_test\_somethingsecretejworjwoierjwoeijroiwejriwjeorwejrjwejroiweriowjer

### 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



The screenshot shows the Stripe dashboard interface. At the top, there's a navigation bar with 'Deluxe' and 'Activate account' buttons, a search bar, and a 'Create' dropdown. Below the navigation bar, there's a 'TEST DATA' button. The main content area has a heading 'Next, add your business details' with a 'Start' button. Below this, there's a 'Get started with Stripe' section with three cards: 'Send an invoice', 'Share a payment link', and 'More ways to get started'. The 'More ways to get started' card is highlighted with a red circle and contains the 'For developers' section, which shows the 'Publishable key' and 'Secret key'. A red arrow points to the 'Test mode' toggle, which is turned on.

<https://dashboard.stripe.com/test/payment-links>

### 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_randomabcdefghijklmnopqrstuvwxyz0123456789
```

All set!

## 3 Account Registration

## Sign Up

**Email**

**Username**

**Password**

**Sign Up**

Already have an account? [Login](#)

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

Deluxe

Activate account

Q Search...

Create

Help

Home

Payments

Balances

Customers

Products

Reports

Connect

More

TEST DATA

Developers

Test mode

Customers

Filter

Export

Add customer

All

Top 100

<input type="checkbox"/>	EMAIL	DESCRIPTION	PAYMENT METHOD	CREATED
<input type="checkbox"/>	yup@yup.com	yup	—	Oct 15, 11:23 PM

- 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.

<p><b>Stripe Checkout</b></p> <p>The quickest way to build conversion-optimized payment forms, hosted on Stripe.</p> <p> <a href="https://stripe.com/docs/payments/checkout">https://stripe.com/docs/payments/checkout</a></p>	<p><b>Stripe Checkout</b></p> <p>Use Stripe Checkout as a low-code integration to build a customized payment page, hosted on Stripe.</p>	<p><b>Stripe.js and Stripe Elements</b></p> <p>Use Stripe.js and Elements to build custom payment forms on the web.</p> <p> <a href="https://stripe.com/docs/stripe-js">https://stripe.com/docs/stripe-js</a></p>	<p><b>Stripe.js and Stripe Elements</b></p> <p>Use Stripe.js and Elements to build custom payment forms on the web.</p>
---	--	--	---

## 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

<p><b>React Stripe.js reference</b></p> <p>React Stripe.js is a thin wrapper around Stripe Elements. It allows you to add Elements to any React app. The Stripe.js reference covers complete Elements customization details. You can use Elements with any Stripe product to collect online payments. To find the right integration path for your business, explore our</p> <p> <a href="https://stripe.com/docs/stripe-js/react">https://stripe.com/docs/stripe-js/react</a></p>	<p><b>React Stripe.js reference</b></p> <p>Learn about React components for Stripe.js and Stripe Elements.</p>
--	--

## 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

#### Test your integration

Hire a Stripe certified expert or use a prebuilt solution created by one of our verified partners (no code required). This page includes test card numbers and other information to make sure your integration works as planned. Use it to trigger different flows in your integration and ensure they are handled accordingly.

 <https://stripe.com/docs/testing>

Stripe Docs

#### Testing

Simulate payments to test your integration.

## 6 Accept One time Credit Card Payment (Custom flow)

### 6.1 Developer's Point of View

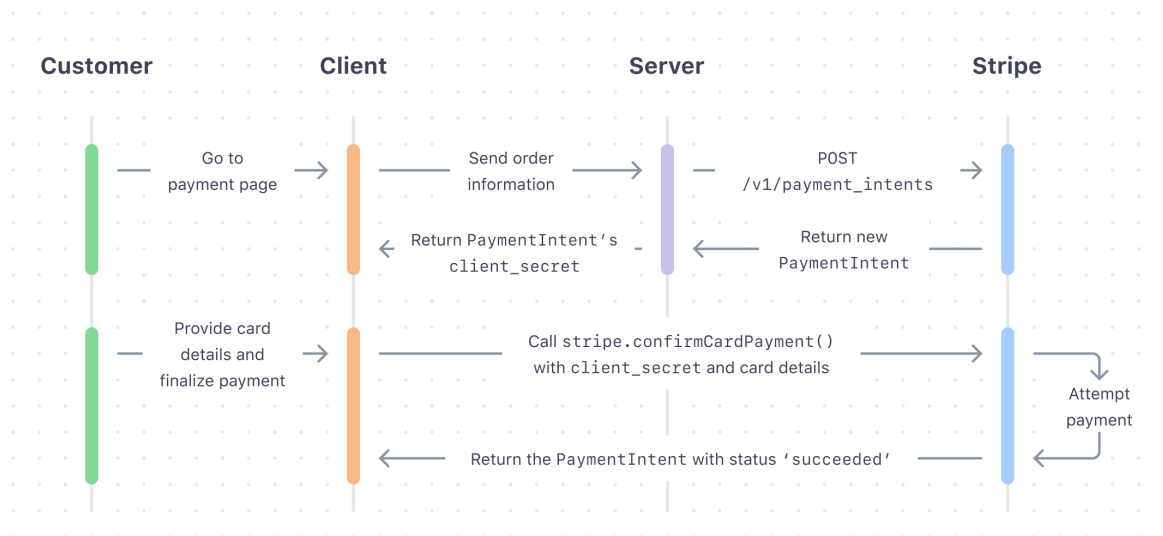


Diagram taken from Stripe's website

### 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

Platform: [Web](#) [iOS](#) [Android](#) Frontend: [HTML](#) [React](#) Backend: [Node](#) [Ruby](#) [Python](#) [PHP](#) [.NET](#) [Java](#) [Go](#)

#### Custom payment flow

[Prebuilt Checkout page](#) [Custom payment flow](#)

Learn how to embed a custom Stripe payment form in your website or application. The client- and server-side code builds a checkout form to complete a card payment.

[Download full app](#) Don't code? Get help from [our partners](#).

#### Set up the server

**Install the Stripe Node library**

Install the package and import it in your code. Alternatively, if you're starting from scratch and need a `package.json` file, download the project files using the link in the code editor.

```

1 const express = require("express");
2 const app = express();
3 // This is your real test secret API key.
4 const stripe = require("stripe")("sk_test_51lg4dH073201buP401AS2bdcXp13Hq3Jb1KX0Mw1D6");
5
6 app.use(express.static("public"));
7 app.use(express.json());
8
9 const calculateAndSavePayment = (req, res) => {
  
```

Stripe's Sample Integration

Please visit the link here to visit the page

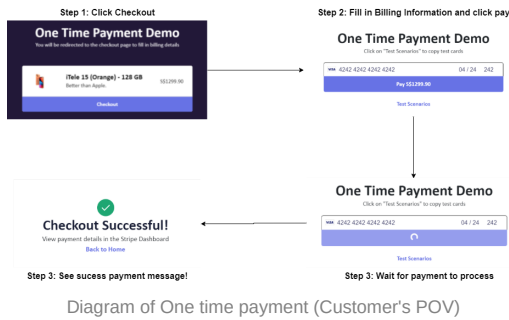
**Custom payment flow**

Learn how to embed a custom Stripe payment form in your website or

<https://stripe.com/docs/payments/integration-builder>

- View sample integration on GitHub Repository

Please visit the link here to download the repository



GitHub - keilokimnida/tutorial-stripe-payments: Repository for 'Stripe with React & NodeJS' Tutorial - GitHub for 'Stripe with React & NodeJS' Tutorial  
<https://github.com/keilokimnida/tutorial-stripe-payment>

## 6.3 View payments

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

The screenshot shows the Stripe dashboard with the 'Payments' tab selected. The left sidebar lists various sections like 'All payments', 'Reviews', 'Disputes', 'All transactions', 'Invoices', 'Subscriptions', 'Quotes', and 'Payment links'. The main area displays a table of payments.

AMOUNT	CURRENCY	STATUS	DESCRIPTION	CUSTOMER	DATE
\$10.00	USD	Incomplete	Created by stripe.com/docs demo		Oct 16, 4:30 PM
\$1,299.90	SGD	Succeeded	[Redacted]	yup@yup.com	Oct 16, 4:08 PM
\$1,299.90	SGD	Succeeded	[Redacted]	yup@yup.com	Oct 16, 4:00 PM
\$6.00	SGD	Succeeded	Subscription update	yup@yup.com	Oct 15, 11:25 PM
\$9.90	SGD	Succeeded	Subscription creation	yup@yup.com	Oct 15, 11:24 PM
\$15.90	SGD	Succeeded	Subscription creation	tyt@tyt.com	Oct 15, 5:04 PM

## 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

The screenshot shows the Stripe checkout page. There is a checkbox labeled 'Save card for future payments' with an arrow pointing to it. Below the checkbox, the text 'Save card during payment' is displayed.

The screenshot shows the 'Add Payment Method' form. It includes a 'Card number' field, an 'MM / YY CVC' field, and a 'Save' button. Below the 'Save' button is a 'Cancel' link.

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>

stripe docs

#### Save payment details during payment

Learn how to save payment details during a 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>

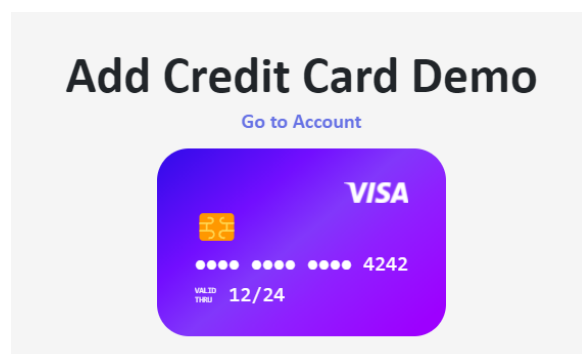
stripe docs

#### Set up future payments


Learn how to save card details and charge your customers later.

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

Please visit the link here to download the repository



GitHub - keilokimnida/tutorial-stripe-payments: Repository for 'Stripe with React & NodeJS' Tutorial - GitHub - keilokimnida/tutorial-stripe-payments: Repository for 'Stripe with React & NodeJS' Tutorial

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

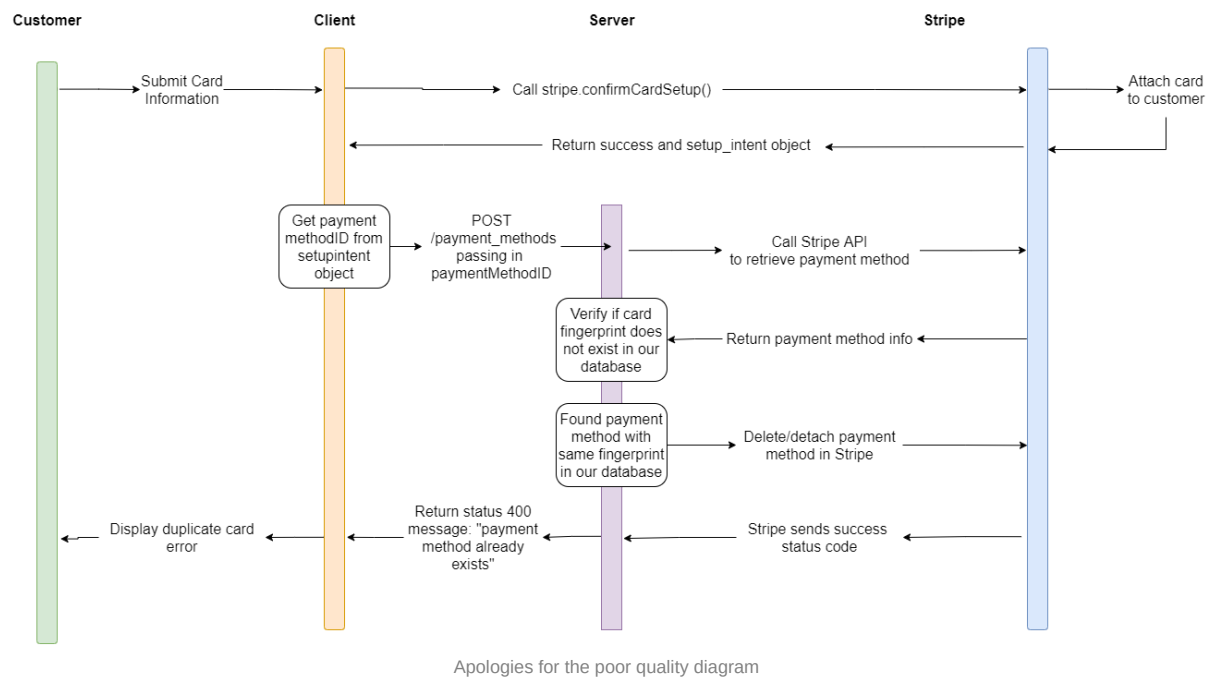
## 7.3 Potential Issue: Duplicate Cards for each Customer

- 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**



## 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

### Subscription

Current Billing Cycle: October 15, 2021 - November 15, 2021  
Credits are given when downgrading plans. If you have credit/debit in your account, it will be applied in your next invoice.

Current Plan

**Premium**

\$15.90 per month

Active

Account Balance

**\$0.10**

Credit

Cancel Subscription

Change Plan

Please visit the link here to download the repository

GitHub - keilokimnida/tutorial-stripe-payments: Repository for 'Stripe with React & NodeJS' Tutorial - GitHub - keilokimnida/tutorial-stripe-payments: Repository 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

Designing an integration

How do you want your customers to modify their subscriptions after setting them up for the first time? If you're

<https://stripe.com/docs/billing/subscriptions/designing-integration>

Billing examples

This page explores some common billing use cases and demonstrates how to model them with products and

<https://stripe.com/docs/billing/subscriptions/examples>

## 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

Settings > Emails

Customer emails

Email customers about...

☒ Successful payments ⓘ

☐ Refunds

To manage emails about invoices, failed payments, and more, visit [Billing settings →](#)

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

Deluxe
Activate account
Search...

Home
Payments
Balances
Customers
Products
Reports
Connect
More

### Manage failed payments for subscriptions

Configure the steps you'd like to take when charging a customer's payment method fails.

**Retry schedule**

☐ Use Smart Retries for subscriptions
☒ Use custom retry schedule for subscriptions

Retry 1 day after the previous attempt

Retry 1 day after the previous attempt

Retry 1 day after the previous attempt

**Customer emails**

☒ Send emails when card payments fail

**Subscription status**

If all retries for a payment fail,
cancel the subscription

**Invoice status**

If all retries for a payment fail,
leave the invoice as-is

## 8.3 View and Manage Subscriptions

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

Deluxe
Activate account
Search...
Create
Help
Settings
User

Home
Payments
Balances
Customers
Products
Reports
Connect
More
TEST DATA
Developers
Test mode

**Payments**
All payments
Invoices
Subscriptions
Quotes
Payment links

**Subscriptions**

Current
Scheduled
Canceled

CUSTOMER	STATUS	BILLING	PRODUCT	CREATED
yup@yup.com	Active	Auto	Premium Plan	Oct 15, 11:24 PM
tyt@tyt.com	Active	Auto	Standard Plan	Oct 15, 5:03 PM
plp@plp.com	Active	Auto	Premium Plan	Oct 15, 4:11 PM
iu@iu.com	Cancels Nov 15	Auto	Standard Plan	Oct 15, 4:04 PM
qwe@qwe.com	Active	Auto	Standard Plan	Oct 15, 3:54 PM
oop@oop.com	Cancels Nov 15	Auto	Standard Plan	Oct 15, 1:08 AM
a@a.com	Active	Auto	Standard Plan	Oct 14, 11:33 PM

## 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>

stripe docs

### How subscriptions work

Learn how subscriptions work within Stripe.

There are two types of error for recurring charge error

- Requires payment method
- Requires action

### 8.4.1 Requires payment method

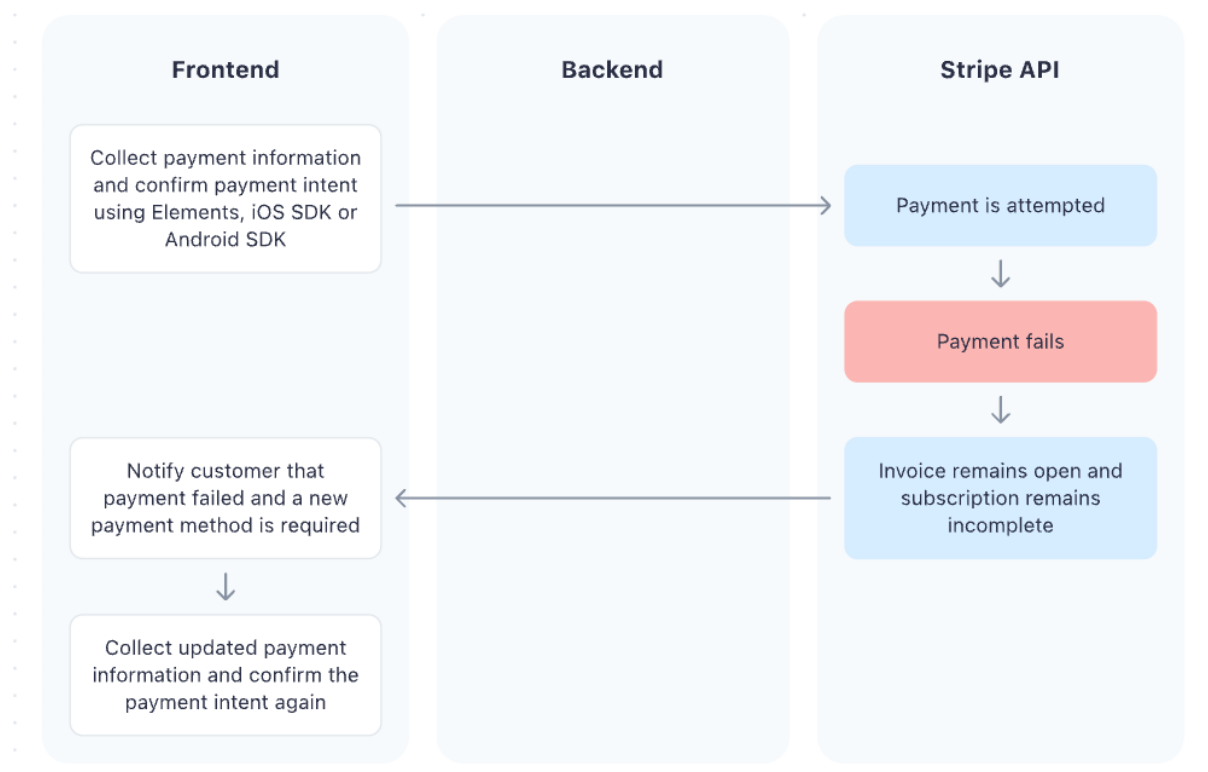
If payment fails because of a card error such as a decline, the status of the PaymentIntent is `requires_payment_method` and the subscription is incomplete.

RESPONSE	SUBSCRIPTION	PAYMENTINTENT
<pre>1 { 2   "id": "sub_1ELI8bClCIK1jWvsvK36TX1C", 3   "object": "subscription", 4   "status": "incomplete", 5   ... 6   "latest_invoice": { 7     "id": "in_EmGqfJMYy3Nt9M", 8     "status": "open", 9     ... 10    "payment_intent": { 11      "status": "requires_payment_method", 12      ... 13    } 14  } 15 }</pre>	incomplete	requires_payment_method

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**.

RESPONSE	SUBSCRIPTION	PAYMENTINTENT
<pre>1  { 2    "id": "sub_1ELI8bClCIKljWvsvK36TX1C", 3    "object": "subscription", 4    "status": "incomplete", 5    ... 6    "latest_invoice": { 7      "id": "in_EmGqfJMYy3Nt9M", 8      "status": "open", 9      ... 10   "payment_intent": { 11     "status": "requires_action", 12     "client_secret": "pi_91_secret_W9", 13     "next_action": { 14       "type": "use_stripe_sdk", 15       ... 16     }, 17   }, 18   ... 19 } 20 }</pre>	incomplete	requires_action

Screenshot taken from Stripe's website


To handle these scenarios:

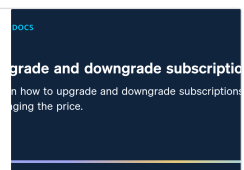
- Notify the customer that they must authenticate.
- Complete authentication using **stripe.ConfirmCardPayment**.

## 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>

Stripe docs

### Use incoming webhooks to get real-time updates


Listen for events on your Stripe account so your integration can automatically trigger reactions.

### 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/question/s/30188582/ngrok-command-not-found>



Solution for Mac devices

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

One of the great things about Windows is that you can get many of your tasks done from the Command Prompt on your machine. You just need to enter cmd.exe and 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-external-command/>

Solution for Windows devices

#### 9.2.3 Login to ngrok on your computer

- You will not need to redo this step to relogin again upon successful login
- Login to ngrok dashboard and copy this code under "Setup and Installation"

## 2. Connect your account

Running this command will add your authtoken to the default `ngrok.yml` configuration file. This will grant you access to more features and longer session times. Running tunnels will be listed on the [status page](#) of the dashboard.

```
$ ./ngrok authtoken [REDACTED]
```

- Open up command prompt/terminal and type and paste the code

```
ngrok authtoken {{TOKEN}}
```

### 9.2.4 Start a ngrok connection

- Open up command prompt/terminal and type and enter

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

- You should see something like this

```
Session Status      online
Account             [REDACTED] Plan: Free)
Version             2.3.40
Region              United States (us)
Web Interface       [REDACTED]
Forwarding           [REDACTED]
Forwarding           [REDACTED]
Connections         [REDACTED]
HTTP Requests       [REDACTED]
```

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

### 9.2.5 Set up webhook in Stripe Dashboard

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

The screenshot shows the Stripe Dashboard interface. The top navigation bar includes 'Deluxe', 'Activate account', a search bar, and buttons for 'Create', 'Help', 'Settings', and 'User'. The main navigation bar has 'Home', 'Payments', 'Balances', 'Customers', 'Products', 'Reports', 'Connect', and 'More'. The 'Developers' section is active, showing 'Webhooks' as the selected tab. The 'Webhooks' section has a sidebar with 'Overview', 'API keys', 'Webhooks', 'Events', 'Logs', and 'Extensions'. The main content area shows 'Hosted endpoints' with a table containing one entry. The table has columns for 'URL', 'TYPE', 'LAST 7 DAYS', 'ERROR RATE', and 'STATUS'. The entry has a URL ending in '.ngrok.io', a 'Direct' type, a graph for 'LAST 7 DAYS', a '13%' error rate, and an 'Active' status. A black arrow points to the '+ Add endpoint' button. Below the table is a 'Local listeners' section with an '+ Add local listener' button and a note: 'Listen to live Stripe events and forward them to your local device using the Stripe CLI.'

- 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

### Listen to Stripe events

[Add an endpoint](#) [Test in a local environment](#)

Set up your webhook endpoint to receive live events from Stripe or [learn more about Webhooks](#).

Endpoint URL

https://

Description

An optional description of what this webhook endpoint is used for...

☐ Listen to events on Connected accounts ⓘ

Select events to listen to

[+ Select events](#)

[Add endpoint](#)

[Cancel](#)

## 9.2.6 Monitor your webhook events

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

Status	Listening for	API version	Signing secret	Configuration
Enabled	8 events	2020-08-27 ⓘ	<a href="#">Reveal</a>	<a href="#">View logs</a>

[All](#)
[Succeeded](#)
[Failed](#)

TODAY	
✓ payment_intent.succeeded	4:08:30 PM
✓ payment_intent.succeeded	4:00:10 PM
✓ invoice.paid	12:48:01 PM
✓ customer.subscription.updated	12:47:11 PM

**payment\_intent.succeeded** [Resend](#) [...](#)

**Response**

✓ 200 OK

## 10 Rate limits

### 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. We have several limiters in the API, including: A rate limiter that limits the number of

<https://stripe.com/docs/rate-limits>

Stripe Docs

### Rate limits

Learn about API rate limits and how to work with them.

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! ^^