

Docs > Integrations > Subscription channels

Subscription Foundation

Subscription Foundation delivers an [open source framework](#) for building BigCommerce apps that empower shoppers to subscribe to products and services. It provides a template for custom subscription billing and invoicing solutions for your business or client. Subscription Foundation includes a default integration with Stripe Billing, which can significantly accelerate your development time.

Subscription Foundation uses the [Channels toolkit](#) to display the custom subscription channel in the **Channel Manager** of a store's control panel alongside other sales channels.

Software requirements

- [Node.js](#) 18.15.0
- The [npm](#) package manager
- A [supported SQL database engine](#), either Postgres or another database server of your choice

Configure accounts



Account configuration requirements

- Because there is a [store email address constraint](#) on draft and private apps, you need to make sure that both your sandbox store and your Dev Portal account use the same email address.
- We recommend that you use a sandbox store that has the same multi-storefront status as the production store. For example, if you're developing for a multi-storefront enabled merchant store, use a multi-storefront sandbox. For information on configuring multi-storefront, see [Multi-Storefront](#).

To configure your accounts, complete the following steps:

1. To develop and test apps, you need a BigCommerce sandbox store. If you don't have one, [Create a Sandbox Store](#).
2. To register apps and create app API accounts, you need a BigCommerce Dev Portal account. If you don't have one, create a [Dev Portal account](#).

Fork and install the source repository

To fork the repository, complete the following steps:

1. Fork the [subscription-foundation repository \(GitHub\)](#) to your GitHub account.
2. Clone your fork to your local development environment.
3. Navigate to the root directory of your cloned repository and install the default packages for your app by running the following command:

Install packages

```
1 npm install
```




Create an HTTPS tunnel

After you successfully install the packages, prepare to expose the app to the internet so you can install it on your sandbox store.

Because you must serve apps over fully-qualified publicly accessible URLs, this guide uses a tool called [ngrok](#) to open HTTP tunnels that securely expose `localhost` ports to the internet.

To get started with ngrok, follow the [Create an HTTPS Tunnel](#) section of the sample app tutorial. If your app does not run on port 3000, replace `3000` with the port of your app server.



Make note of your ngrok ID

After you successfully start ngrok, make sure to note your ngrok ID. You will use it to create an app profile. By default, ngrok generates a new ID every time you start it up. Make sure to update the callback URLs in your [app profile](#) each time the ngrok id changes.

Create an app profile

As a next step, you need to create a draft app profile to generate app API account credentials. To develop and test the app, you need to install it on your [sandbox store](#).

To install your app on your sandbox store, complete the following steps:

1.

Sign in to the [Dev Portal](#). Then follow our directions to [create an app](#). Enter the following values on the **Technical** tab.
2.

In the **Callback URLs** section, supply the following URLs. Replace `{ngrok_url}` with the ngrok ID you noted in [Create an HTTPS tunnel](#). Edit the app profile with new callback URLs each time the ngrok id changes. To learn more about developing with ngrok, see [How to Test App Authentication Locally with ngrok](#).

Callback	URL value
Auth Callback	<code>https://{ngrok_url}/api/auth</code>
Load Callback	<code>https://{ngrok_url}/api/load</code>
Uninstall Callback	<code>https://{ngrok_url}/api/uninstall</code>

3.

In the **OAuth scopes** section, add the following scopes:

UI Name	Permission	Parameter
Orders	modify	<code>store_v2_orders</code>
Order Transactions	modify	<code>store_v2_transactions</code>
Products	modify	<code>store_v2_products</code>
Customers	modify	<code>store_v2_customers</code>
Content	modify	<code>store_v2_content</code>
Carts	modify	<code>store_cart</code>

UI Name	Permission	Parameter
Channel Listings	modify	store_channel_listings
Channel Settings	modify	store_channel_settings
Information & Settings	modify	store_v2_information
Sites & Routes	read-only	store_sites_read-only

4. On the **App Supported Features** tab, select **MULTI STOREFRONT** to specify this is a multi-storefront enabled app.
5. Click **Update & Close**, then click **Confirm Update** to acknowledge the OAuth scopes you configured.

Make sure to remain signed in to the Dev Portal.


Configure Stripe

Subscription Foundation uses Stripe Connect so that your app can use one single connection to submit payments to the previously configured Stripe accounts of multiple BigCommerce merchants. See [Stripe Billing](#) for merchant configuration instructions.

When the app initiates subscription charges, it will use the **merchant's dedicated public key** with the **app's dedicated secret key**. This ensures that a subscriber's entire purchase history is available in the merchant's Stripe Dashboard and store control panel.

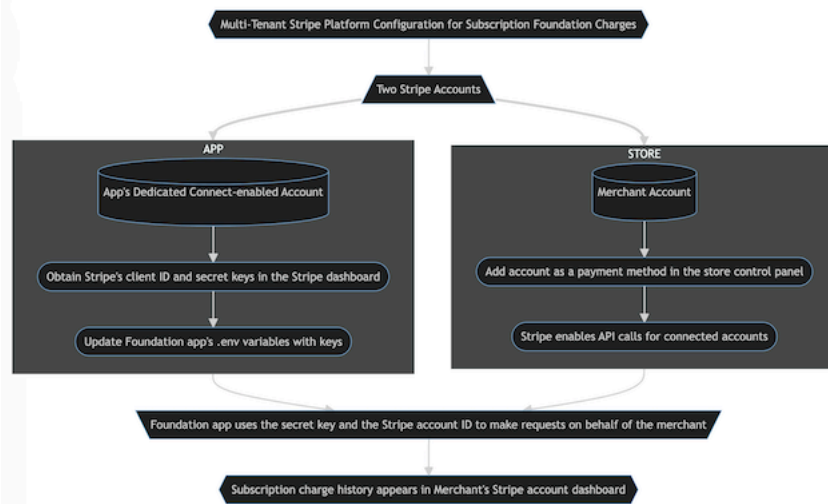
During development, you'll work with two Stripe accounts. You'll create them both in the next section. The following table lists them in the order of creation:

Order of creation	Description
Merchant account	This account simulates a merchant's pre-existing account
App account	You'll configure this account as the app's dedicated Stripe Connect-enabled account



Multi-tenant setup

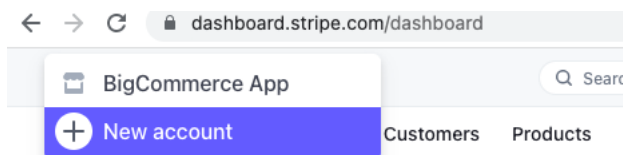
Prior to installing the app, add the merchant account to the sandbox store as a payment method. When the app is installed, the UI prompts the store owner or authorized user to give the app permission to make charges on behalf of the store. Upon consent, the app initiates an OAuth code grant flow to link the merchant's Stripe account with the app's Stripe Connect-enabled account. The diagram below shows the configuration and the integration process.



Create Stripe accounts

To create a Stripe account, complete the following steps:

1. To create the Stripe account that simulates your merchant's pre-existing Stripe account, [sign up with Stripe](#). Your app uses this account's client ID to process subscriber payments and manage subscriptions. Make sure to give it a merchant-specific name.
2. Sign in to the [Stripe Dashboard](#).
3. On the [Stripe Dashboard](#), create and configure a second app-specific Stripe account. The app uses this account's client secret to process subscription payments.
 - a. In the top left corner of your Stripe Dashboard, click the name of your merchant-specific account. Select **+ New Account** from the drop-down list.
 - b. Give the new account a name that clearly identifies it as app-specific to differentiate it from the merchant-specific payment and subscription management account. This guide uses **BigCommerce App** as an example.



- c. Under **Get started with Stripe**, copy the **Secret key**.



Make note of the secret key

Make note of the **secret key** and keep it in a safe location. In a later step, you will use the secret key to update the `STRIPE_SECRET_KEY` environment variable in the `.env` file.

Enable Stripe Connect for Platforms

As you complete the following steps, make sure that you are making changes to the app-specific account. To verify, check the top left corner of your Stripe Dashboard for the app account name you chose in step 3b of the preceding section.



Test mode

You can simulate transactions in test mode to confirm your integration works correctly.

To configure test mode, complete the following steps:

1. On your [Stripe Dashboard](#), click **Connect** to enable [Stripe Connect for Platforms](#).
2. Click **Get started**.
3. Select **Platform or Marketplace**, then click **Continue**.
4. To configure the app-specific account's Stripe Connect settings, go to [Settings > Connect settings](#) on your Stripe Dashboard.
5. Under **Integration** do the following:
 - a. Copy the **Test mode client ID**.



Make note of the client ID

Make note of the **test mode client ID** and keep it in a safe location. In a later step, you will use the client ID to update the `NEXT_PUBLIC_STRIPE_CLIENT_ID` environment variable in the `.env` file.

- b. Set **OAuth settings** to **OAuth for Standard accounts**.
- c. Click **Redirects**, then click **+ Add URI**. Add the following callback URI for your app:

Your app's Stripe callback URI

```
1 https://{ngrok_url}/stripe/callback
```

Configure the BigCommerce store

After you successfully configure test mode, configure your BigCommerce sandbox store in the store control panel.

To configure the store to make subscription charges, complete the following steps:

1. In the control panel of your BigCommerce store, navigate to **Store Setup > Payments > [Stripe settings](#)** and make sure that **Test Mode** is set to **Yes**.


[Connecting with Stripe](#)
[Stripe Dashboard](#)

Display Name ⓘ

Credit Card

Transaction Type ⓘ

Authorize & Capture

Test Mode ⓘ

Yes

2. To start a subscription, shoppers must check out using a stored payment instrument. To enable this functionality, go to [Stripe settings](#), locate the **Stored Credit Cards** section, and toggle on **Enable stored credit cards with Stripe**.

Stored Credit Cards

Allow your registered customers to safely and securely store their credit card details so that they are able to complete future purchases faster.

The credit card details will be stored securely with Stripe and associated with the billing address stored with the customer record on your store.

Enable stored credit cards with Stripe



3. If you don't have a stored card to charge, go to your BigCommerce storefront and add some products to your cart. Begin the checkout sequence and either sign in as a shopper or create a new account. During the payments step, select **Save this card for later**.

Declare environment variables

After you configure the store, you have all the information you need to add environment variables to your project.



Note on naming conventions

In some places, this guide and app template code refer to a Stripe API account's public key as a **client ID**, and its secret key as a **client secret**.

To declare environment variables, complete the following steps:

1. Create a `.env` file in the root directory of your project.
2. Copy the contents of `.env.sample` to `.env` with the following command:

Copy `.env.sample` contents

```
1 cp .env.sample .env
```

Open the `.env` file you just created and update the following environment variables:

Environment variable	Description	Reference location
<code>NEXT_PUBLIC_APP_URL</code>	A public-facing URL that can receive webhooks.	On this page, see Create an HTTPS tunnel ; see also the HTTPS tunnel section of the Sample App Tutorial
<code>DATABASE_URL</code>	A URL that connects to a database	On this page, see Run migration and start the server
<code>NEXT_PUBLIC_APP_ID</code>	The app's ID	Find an App's ID
<code>BC_APP_CLIENT_ID</code>	The app API account's client ID	View App Credentials
<code>BC_APP_SECRET</code>	The app API account's client secret	View App Credentials
<code>NEXT_PUBLIC_STRIPE_CLIENT_ID</code>	The app-specific Stripe Connect API account client ID	see Enable Stripe Connect for Platforms
<code>STRIPE_SECRET_KEY</code>	The app-specific Stripe Connect API account client secret	see Create Stripe Accounts

Run migration and start the server



Database note

This example uses Postgres as a data store. We recommend using [Supabase](#) for a free database. For information on switching databases, see the [Replacing Postgres](#) section.

To run the migration and start the server, complete the following steps:

1. If you're using Postgres, which is the default data store for Subscription Foundation, skip to the next step. Otherwise, follow the instructions in [Replacing Postgres](#).
2. Obtain the database URL in [Supabase](#) by doing the following:
 - a. In the Supabase dashboard, click **New project** and select the appropriate organization. Enter a name and database password in the **Create a new project** dialog, and click **Create new project**.
 - b. In the Supabase dashboard, click the Settings (gear) icon, and in the left navigation menu, click **Database**. Scroll down the page to the **Connection string** section, click **URI**, and copy the connection string. Use this string to update the `DATABASE_URL` environment variable in the `.env` file. The string resembles the following example:

Example Postgres Cloud connection string

```
1 postgresql://postgres:[YOUR-PASSWORD]@db.uqchmyniufaqqijttavq.supabase.co:5432/postgres
```

3. Run the pre-configured Prisma migration script to create the database tables and initial client as defined in `/prisma/migrations/*` by issuing the following command:

Run Prisma migration

```
1 npx prisma migrate dev
```

```
Environment variables loaded from .env
[Prisma schema loaded from prisma/schema.prisma
Datasource "db": PostgreSQL database "postgres", schema "public" at "db.uqchmyniufaqk
ijttavq.supabase.co:5432"

The following migration(s) have been applied:

migrations/
├─ 20220822143907_init/
└─ migration.sql

Your database is now in sync with your schema.

✓ Generated Prisma Client (2.30.3) to ./node_modules/@prisma/client in 741ms
```

3. Start the app server with the following npm script:

Start the app server

```
1 npm run dev
```

After the app server and ngrok are running, you can install the draft app on your sandbox store. For more information about installing and troubleshooting apps in development, read our App Tutorial section on [Installing and launching an app](#).

Replacing Postgres

To use an alternate SQL database (e.g., SQLite), complete the following steps:

1. Update the `/prisma/schema.prisma` file with a `provider` other than `Postgresql`. For a list of options, read [Prisma's reference docs](#).

```
database db {
  provider = "sqlite"
  url      = env("DATABASE_URL")
}
```

2. In `/prisma/.env`, update the value of the `DATABASE_URL` variable to match the connection string of your new database. Use the same string to update the `DATABASE_URL` environment variable in the `.env` file.

3. Run the prisma migration script with the following command:

Run Prisma migration

```
1 npx prisma migrate dev
```

⚠ If you miss the preceding step, the database provider will not be successfully switched. For a list of Prisma migrate limitations, see [Prisma Migrate limitations and known issues](#).

4. To generate a fresh app client that uses the new database provider, run the following script:

Run Prisma generate

```
1 npx prisma generate
```

5. To access this database locally, run the following command, then use a visual editor to verify that the database tables have been created.

Run Prisma studio

```
1 npx prisma studio
```

6. Start the app server by running the following command:

Start the app server

```
1 npm run dev
```

Managing subscription products

To add new subscription rules and edit existing ones, complete the following steps:

1. In the store control panel, navigate to the **Channel Manager** menu and click **Stripe Subscriptions**.
2. Click the product you want to modify. For more information, see [Foundations for Stripe Billing](#).



Subscription sales channel

The app will create its own dedicated sales channel upon installation.

If you plan to use the API to add products to the subscription sales channel, see [product channel assignments](#) for more information.

Deploying your app with Vercel

The BigCommerce Subscription Foundation framework enables you to deploy your application with Vercel directly from the GitHub repo.

To deploy your app with Vercel:

1. Login to GitHub, and then navigate to the [BigCommerce Subscription Foundation README.md](#) file.
2. Scroll down the page to the **Deploy with Vercel** section and click **Deploy**.

Deploy with Vercel



3. In the **Get started** section, select a Git provider, and then authorize Vercel to connect to your provider.
4. In the **Create Git Repository**, select a Git scope, enter a name for the repository, and then click **Create**. Do not enable **Create private Git Repository** because a different OAuth is needed and will cause an error.

Create Git Repository

To ensure you can easily update your project after deploying it, a Git repository must be created. Every push to that Git repository will be deployed automatically.

GIT SCOPE

REPOSITORY NAME

☐ Create private Git Repository

Create

5. (Optional) If you are deploying to Vercel from a GitHub organization, you need to create a team by following the steps in the **Create a team** section.

Create a Team

Because you will be deploying from a GitHub organization, creating a Vercel Team is required for collaborating with others. You've already used your free trial, so you will need to add a credit card to create a team.

TEAM NAME

TEAM SLUG

Create

6. In the **Configure Project** section, in the **DATABASE_URL** field enter the connection string for the Supabase database you created in the previous step. See [Run migration and start the server](#) for information on creating the database and obtaining the **DATABASE_URL**.
7. Enter '1234' for the remaining environment variables and then click **Deploy**. The deployment process will take a few minutes.

Configure Project

Please provide values for the required Environment Variables.

▼ Required Environment Variables

NAME	VALUE (WILL BE ENCRYPTED)
NEXT_PUBLIC_APP_URL	1234
DATABASE_URL	postgresql://postgres:fb9uaUFbDPRCNOF@db.s...
NEXT_PUBLIC_BC_APP_ID	1234
BC_APP_CLIENT_ID	1234
BC_APP_SECRET	1234
NEXT_PUBLIC_STRIPE_CLIENT_ID	1234
STRIPE_SECRET_KEY	1234

View the Subscription Foundation docs for information on each ENV variable. [Learn More](#)

Deploy

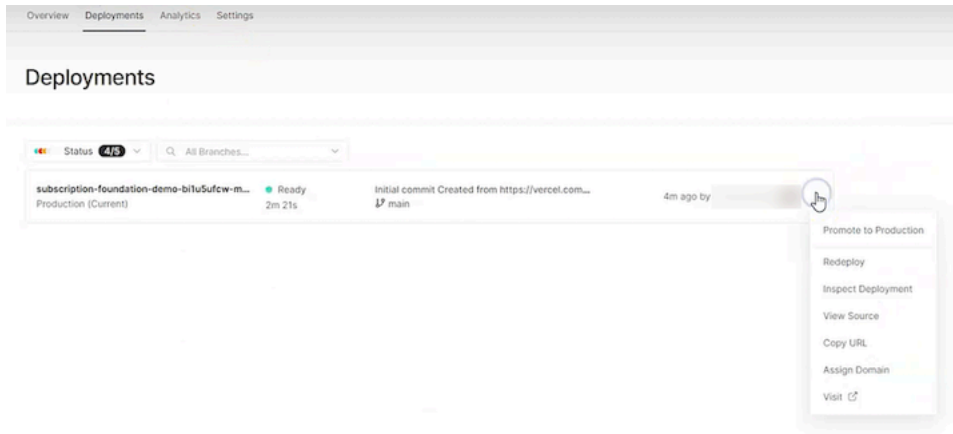
8. After you have successfully deployed to Vercel, you must update your environment variables. For more information, see [Declare environment variables](#).



Updating `NEXT_PUBLIC_APP_URL`

Vercel generates the `NEXT_PUBLIC_APP_URL` after the first deployment. You can update the `APP_URL` with this value.

9. Redeploy your application to Vercel for the environment variable changes to take place. Click on the **Deployments** tab and select **Redeploy** from the three vertical dots on the right.



Troubleshooting

Issue: Environment variable not found when running the database migration

This error can occur when the `/prisma/schema.prisma` file contains the incorrect provider, or the `.env` file contains an incorrect `DATABASE_URL`. Verify that the information in both files is correct.

Join our community of thousands of developers

Join #BigCommerceDevs Slack

Join thousands of other #BigCommerceDevs that are building on BigCommerce. We will also keep you updated with our monthly BigCommerce Developers newsletter. You may opt out at any time.

Need help? [Join our BigCommerceDevs Slack](#)

Find endpoints quickly in the [API reference](#)

Stay up to date with [Release notes](#)

Visit [BigCommerce home](#)



Everything you need for growth, all from one modern platform.