



**FREE eBook**

# LEARNING Shopify

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

**#shopify**

# Table of Contents

About.....	1
<b>Chapter 1: Getting started with Shopify</b> .....	<b>2</b>
Remarks.....	2
<b>No Versions for Shopify</b> .....	<b>2</b>
Examples.....	2
Installation or Setup.....	2
Local theme setup.....	2
Ruby installer.....	3
Development-Kit.....	3
SSL Certificate.....	3
Shopify Theme Gem.....	3
<b>Create a Private App</b> .....	<b>4</b>
<b>Create a config file</b> .....	<b>4</b>
Shopify local development with Theme Kit.....	5
Install Shopify Theme Kit.....	5
Windows Installation.....	5
Get API Access.....	6
<b>Chapter 2: Implementing Sections</b> .....	<b>10</b>
Introduction.....	10
Remarks.....	10
<b>Things to remember:</b> .....	<b>10</b>
<b>More Resources:</b> .....	<b>10</b>
Examples.....	10
Including a Section on a Page.....	10
In File Schema.....	11
Using Blocks for Recurring Elements.....	12
Using JavaScript and Stylesheets in Sections.....	12
<b>Credits</b> .....	<b>14</b>

---

# About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [shopify](#)

It is an unofficial and free Shopify ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Shopify.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

---

# Chapter 1: Getting started with Shopify

## Remarks

This section provides an overview of what shopify is, and why a developer might want to use it.

It should also mention any large subjects within shopify, and link out to the related topics. Since the Documentation for shopify is new, you may need to create initial versions of those related topics.

---

---

## No Versions for Shopify

Shopify does not release version numbers nor change logs, due to security concerns. To keep up with the changes in Shopify, the best places to check would be:

- [Shopify Blog](#)
- [Shopify Forums](#)
- [Shopify Open Source Projects](#)

## Examples

### Installation or Setup

Detailed instructions on getting shopify set up or installed.

### Local theme setup

There are a few ways to setup a local copy of your Shopify theme:

- Node.js ( gulp/grunt )
- Ruby
- Theme App ( Mac only )

There might be a few other ways, but I found the ruby way the most easy to work with.

In order to work locally on the Shopify theme files you will need a few things:

- Ruby installer ( <http://rubyinstaller.org/downloads/> )
  - Development-Kit ( <http://rubyinstaller.org/downloads/> )
  - SSL Certificate - ( <https://curl.haxx.se/ca/cacert.pem> )
  - Shopify Theme Gem - ( [https://rubygems.org/gems/shopify\\_theme/versions/0.0.25](https://rubygems.org/gems/shopify_theme/versions/0.0.25) )
-

# Ruby installer

The first thing is to install the Ruby installer, use the latest version.

The installation process is straight forward, but **it will ask you to add Ruby to the Windows Environment Global Path and you should click the checkbox and add it there**, so that you don't have to manually do it so afterwards.

---

## Development-Kit

You should choose the correct version of the DevKit that goes with the Ruby installer you added.

The installation process is easy as well.

1. You unzip the file in C:\DevKit for example
2. You cmd in that folder and write `ruby dk.rb init` which should create a `config.yml` in the folder
3. If you open the `config.yml` file you should see at the bottom - `C:/Ruby23-x64` ( depending where you installed your Ruby and what version is ) if it's not there you must add it manually
4. At the same folder afterwards you run the following `ruby dk.rb install`
5. If everything is OK you should be able to run the following command - `gem install json --platform=ruby` that will confirm that your Ruby installation is working

More detailed instructions can be seen here as well:

<https://github.com/oneclick/rubyinstaller/wiki/Development-Kit>

---

## SSL Certificate

You will need to add the **cacert.pem** file to your window global variables.

1. You have to download the following file: <https://curl.haxx.se/ca/cacert.pem> ( save it as `.pem` and not as `.pem.txt` )
  2. You have to enter your Environment Variables screen and under System variables click New
  3. For Variable name you enter **SSL\_CERT\_FILE** and for variable value you enter the path to the file, for example **C:\RailsInstaller\cacert.pem**.
- 

## Shopify Theme Gem

Open your cmd and enter the following code: `gem install shopify_theme`

This should install the the Shopify theme gem that we will need.

---

If you went through all of the above steps once, you won't be doing them any more.

---

Now comes that steps that you will have to do every time you start a new Shopify Project:

- Create a Private App ( from the Shopify Store )
- Create a config file using the Private App information

---

## Create a Private App

You must login to your Shopify store admin panel and go to the App screen -> Private app:

<https://YOUR-STORE-NAME.myshopify.com/admin/apps/private>

And click the button **Create private app**.

It will ask you for a title. ( enter what ever you like )

**The important part is to change all of the permissions to Read and write**

Once you change the permissions and save the App it will redirect you to a page from which you will get the API Key and Password for this App. ( you need those for the config file we will create below)

---

## Create a config file

Go to a local folder where you like the theme files to be in and cmd inside.

Write the following command:

```
theme configure API-KEY PASSWORD DOMAIN ID
```

Where:

- API key - is taken from them Private app we created
- Password - is taken from them Private app we created
- Domain - this is your domain **YOUR-STORE-NAME.myshopify.com**, without https:// before it
- ID - this is the theme ID, if you are going to work on the live theme, you can skip this, but if you are working on a cloned them then you will have to enter the Theme ID here

Once you run the above code it should create a file called **config.yml** where all the provided information will be present.

Now you should cmd ( in case you closed the console ) in the folder where the **config.yml** is and you write the following code: `theme download` which should start downloading the files one by one.

Once you complete the download process you write `theme watch` and every time you modify the downloaded files it will update the files in the Shopify store after 1-2 sec as well. This is true if you

add files as well, but not when you delete files.

More detailed information for the Gem can be found here:

[https://github.com/Shopify/shopify\\_theme](https://github.com/Shopify/shopify_theme)

## Shopify local development with Theme Kit

Shopify themes can also be developed locally. Follow the steps below to setup local development environment:

- Install Shopify Theme Kit
- Get API Access
- Use a new theme or Configure an existing theme with theme kit

Explanation of the above steps are as follows:

## Install Shopify Theme Kit

### Automatic Installation

If you are on Mac or Linux you can use the following installation script to automatically download and install Theme Kit for you.

```
curl -s https://raw.githubusercontent.com/Shopify/themekit/master/scripts/install | sudo  
python
```

### Homebrew

If you have homebrew installed you can install Theme Kit by running the following commands.

```
brew tap shopify/shopify  
brew install themekit
```

## Windows Installation

### Automatic Powershell Install

Run the following commands in Powershell as Administrator.

```
(New-Object  
System.Net.WebClient).DownloadString("https://raw.githubusercontent.com/Shopify/themekit/master/scripts/
```

## Manual

- Create a folder inside C:\Program Files\ called Theme Kit
- Download themekit (below) and copy the extracted program into C:\Program Files\Theme Kit
- Navigate to Control Panel > System and Security > System. Another way to get there is to Right-Click on My Computer and choose the properties item
- Look for the button or link called Environment Variables
- In the second panel look for the item called Path and double-click on it. This should open a window with a text field that is overflowing with content.
- Move your cursor all the way to the end and add the following: ;C:\Program Files\Theme Kit\
- Click OK until all the windows are gone.
- To verify that Theme Kit has been installed, open cmd.exe and type in theme.

## Manual Installation

Download the latest release executable.

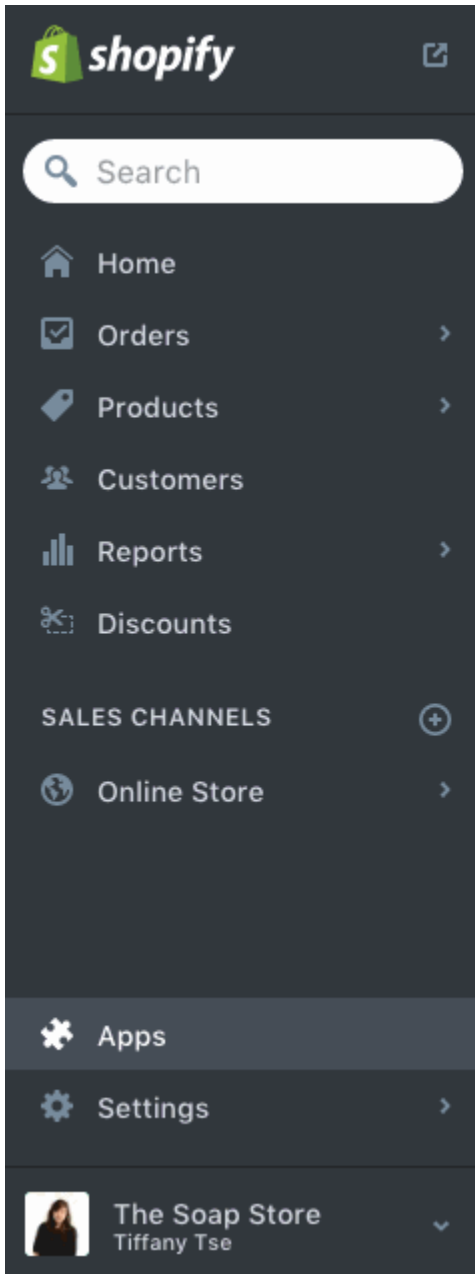
OS	Architecture	Link
macOS	64-bit	<a href="#">Download</a>
Windows	64-bit	<a href="#">Download</a>
Windows	32-bit	<a href="#">Download</a>
Linux	64-bit	<a href="#">Download</a>
Linux	32-bit	<a href="#">Download</a>

## Get API Access

You will need to set up an API key to add to our configuration and create a connection between your store and Theme Kit. The API key allows Theme Kit to talk to and access your store, as well as its theme files.

To do so, log into the Shopify store, and create a private app. In the Shopify Admin, go to Apps and click on View private apps. From there, click Generate API credentials to create your private app. Make sure to set the permissions of Theme templates and theme assets to have Read and write access in order to generate the appropriate API credentials, then click Save.





Apps

## Use apps to en

Apps help you manage your s



Get started with

Fill out the information at the top and set the permissions of Theme templates and theme assets to read and write access. Press Save and you will be presented with the next screen. In it you will see your access credentials. Please make note of the password. You will need it later.

## Authentication

With these credentials you can use your [private app](#) to access your store through the [Shopify API](#).

API key  
28c59b68796ff0bad3b53e042de8d557

Password  
96d9631b4090589db4202c8a5f24cb5c

Shared secret  
880c7b2c2d36aaa989f65650a0a69ce4

URL format  
`https://apikey:password@hostname/admin/resource`

Example URL  
`https://28c59b68796ff0bad3b53e042de8d557:96d9631b4090589db4202c8a5f24cb5c@can-i-buy-a-feeling.myshopify.com/admin/orders`

## Use a new theme.

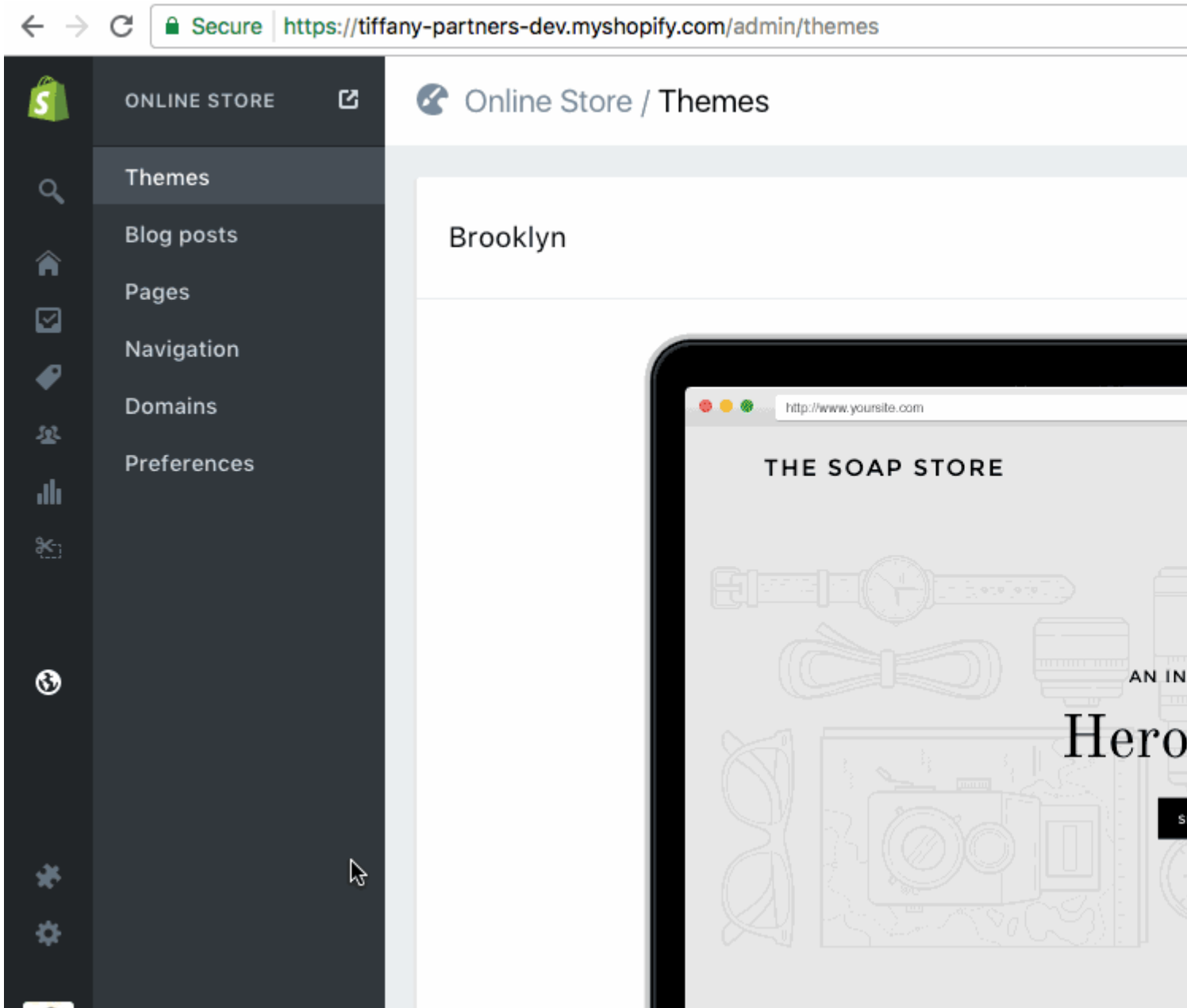
If you are starting from scratch and want to get a quick start, run the following:

```
theme bootstrap --password=[your-password] --store=[your-store.myshopify.com]
```

This will create a new theme for your online store from the [Timber](#) template. Then it will download all those assets from Shopify and automatically create a `config.yml` file for you.

## Configure an existing theme.

To connect an existing theme, you need the theme's ID number. The easiest way to get your theme's ID number is to go to the Theme Editor click on Edit HTML/CSS and copy the theme ID number from the URL — it will be last several digits after `mystore.myshopify.com/admin/themes/`.



Then once you have noted your theme ID, run the following commands:

```
# create configuration
theme configure --password=[your-password] --store=[you-store.myshopify.com] --themeid=[your-theme-id]
# download and setup project in the current directory
theme download
```

Read [Getting started with Shopify online](https://riptutorial.com/shopify/topic/5772/getting-started-with-shopify): <https://riptutorial.com/shopify/topic/5772/getting-started-with-shopify>

---

# Chapter 2: Implementing Sections

## Introduction

Sections are a feature that allow store owners to add, edit, remove and easily reorder content on a page. There are 2 types of sections: dynamic and fixed. Dynamic sections are able to be reordered in their entirety with other sections on the page (homepage). Fixed sections cannot be reordered, but the content within them can still be added to, edited, removed and reordered.

## Remarks

---

### Things to remember:

- The schema of a section is scoped to that particular section. This means you can give your variables relatively simple names without worrying about them clashing with variable names from other sections on the same page.

---

### More Resources:

- <https://help.shopify.com/themes/development/theme-editor/sections>
- <https://help.shopify.com/themes/development/theme-editor/settings-schema>

## Examples

### Including a Section on a Page

Including a Static Section on a page is similar to including a snippet. The only difference is instead of using the word `include` you use the word `section`.

```
{% section 'section-file-name' %}
```

Using this tag will "hardcode" the section into your theme at the place where you put the code and cannot be moved or deleted from the page.

To use Dynamic Sections, you need to use the `preset` control at the bottom of the schema settings for that section.

```
{% schema %}
{
  "presets": [
    {
      "category": "Custom Content",
      "name": "Text",
```

```

    "settings": {
      "heading": "Hello World"
    },
    "blocks": [
      {
        "type": "text",
        "settings": {
          "content": "Once upon a time..."
        }
      }
    ]
  }
}
{% endschema %}

```

By using presets, this will enable you to move and add your section around your page as you require. It also allows you to fill in the section with dummy content so that you can view exactly how the section will look on your page without actually having to add it to the page.

## In File Schema

For sections, instead of utilizing one big JSON file (like `settings_schema.json`, they instead keep their schema within the file of the section that is using it. To do this `{% schema %}` `{% endschema %}` tags are used and JSON is placed between them. From there, the format is similar to

`settings_schema.json`.

```

{% schema %}
{
  "name": "Header Banner",
  "settings": [
    {
      "type": "checkbox",
      "id": "banner_enable",
      "label": "Enable Banner",
      "default": false
    },
    {
      "type": "color",
      "id": "banner_color",
      "label": "Banner Background Color",
      "default": "#000000"
    },
    {
      "type": "color",
      "id": "text_color",
      "label": "Banner Text Color",
      "default": "#ffffff"
    },
    {
      "type": "text",
      "id": "banner_text",
      "label": "Banner Text",
      "default": "Welcome to my Section"
    }
  ]
}

```

```
{% endschema %}
```

To use this in your HTML/Liquid code you would pull from `section.settings`. Something like:

```
{% if section.settings.banner_enable %}
  <div class="banner">
    <p>{{ section.settings.banner_text }}</p>
  </div>
{% endif %}
```

## Using Blocks for Recurring Elements

One of the most useful things that came along with sections is **blocks**. Blocks are basically a blueprint for something that can be created an unlimited amount of times. One of the best examples is the slides of a slider. A block is a top level item in the schema, meaning it is alongside things like `name` and `settings`, but not within them. Within blocks are basically mini schemas.

```
{% schema %}
{
  "name": "Slider",
  "max_blocks": 6,
  "blocks": [
    {
      "type": "slide",
      "name": "Slide",
      "limit": 4,
      "settings": [
        {
          "id": "image",
          "type": "image_picker",
          "label": "Image"
        }
      ]
    }
  ]
}
{% endschema %}
```

Notice the Slide also has a type of `slide`. You can give your blocks their own type that is not one of the Shopify provided types.

To use this you would just loop over each block in your HTML/Liquid code.

```
<div class="slider">
  {% for block in section.blocks %}
    
  {% endfor %}
</div>
```

## Using JavaScript and Stylesheets in Sections

Each section can contain JavaScript and Stylesheets, these two languages are used within liquid tags: `{% javascript %}{% endjavascript %}` & `{% stylesheet %}{% endstylesheet %}`.

When placing code inside of these two tags, Shopify compiles each piece into `shopify_compiled.js` & `shopify_compiled.css`. This allows for greater readability within each section's code as you don't have to search through a long document to find each piece.

If you don't want to use CSS, you can alternatively use SCSS by using `{% stylesheet 'scss' %}` instead.

There are downfalls to using these, though. Liquid cannot be used in any of them, just as it can't be used in normal `*.js`, `*.css`, `*.scss` files without the `.liquid` extension.

Also, when using the SCSS stylesheet tag, this isn't a global stylesheet, so any variables defined within your theme's `.scss` document won't be accessible and will need to be redefined.

## Usage Examples

### JavaScript

```
{% javascript %}
  $( "p" ).click(function() {
    $( this ).slideUp();
  });
{% endjavascript %}
```

### CSS

```
{% stylesheet %}
  .container {
    width: 100%;
  }
  .container p {
    color: #ff0000;
  }
{% endstylesheet %}
```

### SASS

```
{% stylesheet 'scss' %}
  $red: #ff0000;
  .container {
    width: 100%;
    p {
      color: $red;
    }
  }
{% endstylesheet %}
```

Read [Implementing Sections](https://riptutorial.com/shopify/topic/9198/implementing-sections) online: <https://riptutorial.com/shopify/topic/9198/implementing-sections>

---

# Credits

S. No	Chapters	Contributors
1	Getting started with Shopify	<a href="#">Community</a> , <a href="#">drip</a> , <a href="#">Inzamam Tahir</a> , <a href="#">Sam</a>
2	Implementing Sections	<a href="#">BeesonBison</a> , <a href="#">ConduciveMammal</a> , <a href="#">Sam</a>