

[DOCS](#)[CONTACT SALES](#)[Web](#)[Choose a Page](#)[GETTING STARTED](#)[WEB INTEGRATIONS](#)

Getting started on Web Integrations

[SHAREPOINT](#)[ONLINE](#)[FILE HANDLER](#)

Integrate into SharePoint Online as a File Handler

This guide will walk you through the steps necessary to integrate PSPDFKit for Web into SharePoint Online as a file handler.

File handlers instruct SharePoint to redirect to your URL of choice whenever a file with a particular extension is clicked. By using a file handler, you'll override SharePoint's default behavior so that PDF documents are opened directly inside your application. Any edits to the opened files using your application can then be directly saved back to SharePoint.

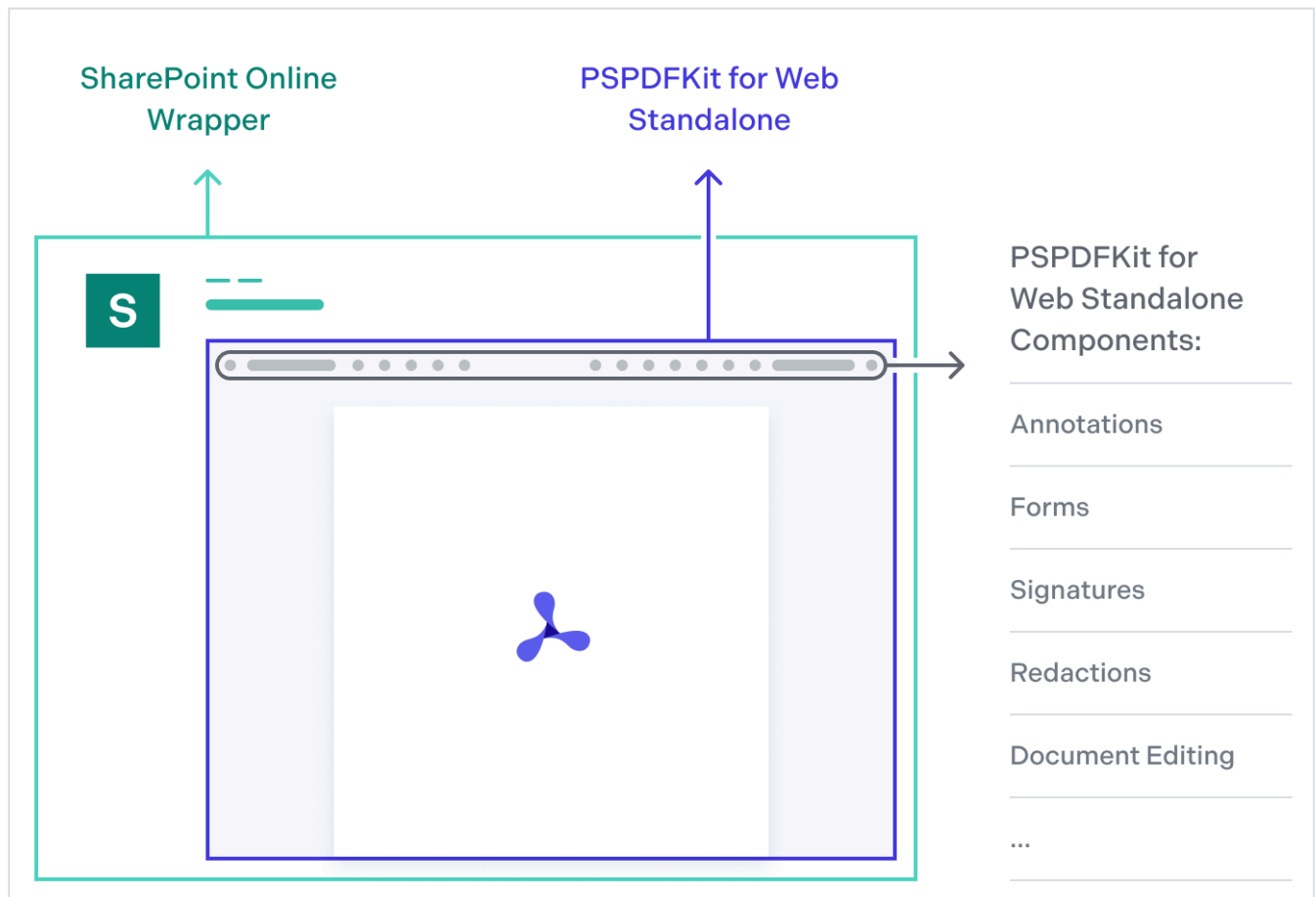
PSPDFKit for SharePoint is a wrapper on top of PSPDFKit for Web Standalone, which is a client-side JavaScript library for viewing and editing PDF documents directly in a web browser.

[ASK AI](#)

PSPDFKit for SharePoint shares the same APIs as PSPDFKit for Web Standalone, so use the Web documentation when customizing your SharePoint application.

TRY OUR NO-CODE APP

PSPDFKit for SharePoint



Requirements

A Microsoft 365 tenant (you can set up a Microsoft 365 tenant for free when you join the Microsoft 365 Developer Program)

The SharePoint Framework

A GitHub account (for cloning the PSPDFKit for SharePoint repository)

Node.js version 16.19.0

A Microsoft Azure Active Directory account

OpenSSL

Admin privileges for the SharePoint tenant

Microsoft Azure admin privileges



File handlers are registered centrally for the entire SharePoint tenant and cannot be selectively enabled or disabled. Uninstalling a file handler from the environment can be problematic due to aggressive caching and may take 24–48 hours to take effect.

1 Setup

- 1 Clone the `pspdfkit-sp-online-filehandler` repository from GitHub.

Navigate to the directory where you want to place the SharePoint integration, and type the following command in your command line/terminal:

```
git clone https://github.com/PSPDFKit/pspdfkit-sp-online-filehandler.git
```

Alternatively, you can download the project as a `.zip` file without cloning the project.

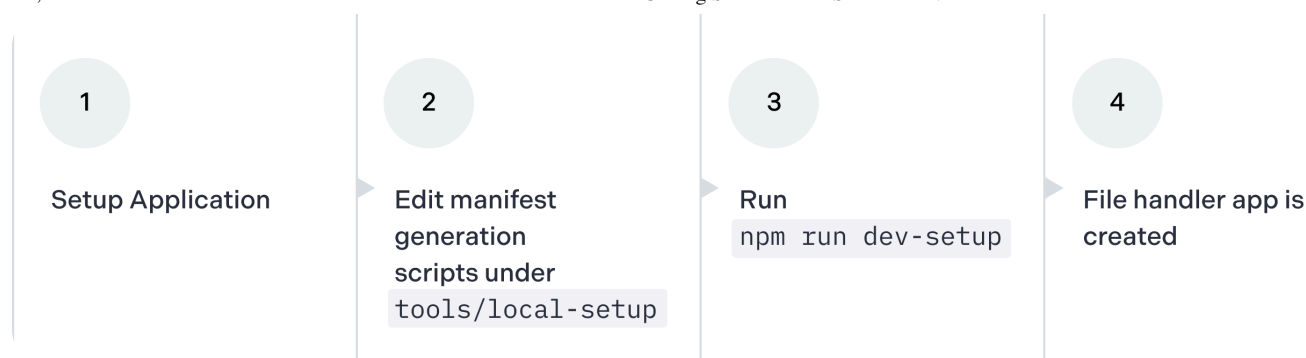
This project uses Next.js as a server-side framework; however, you can use any other server-side framework you like. The application serves as a native file handler of PDF documents for SharePoint sites.

Concretely, this server handles requests to open PDF documents from a SharePoint site and then proceeds to render an HTML page containing the PSPDFKit for Web Standalone viewer.

This project is based on the Markdown FileHandler example project.

2 Installation and Development Deployment

After cloning the repository, and before starting the development setup process, it's worth understanding how the file handler architecture works.



Setup Application is a special helper application that needs to be registered in the Azure App Directory. However, it doesn't represent the file handler app, rather it's a helper to automate the process of creating the file handler. This helper app can later safely be removed from the environment.

3 Customizing the Manifest Specification

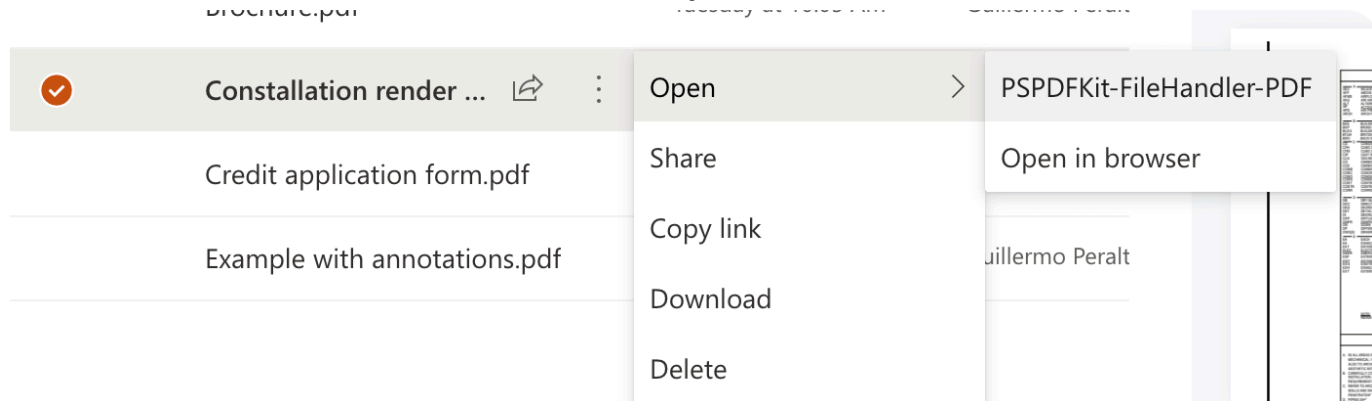
The file handler server is a regular server-side web application that handles requests initiated by SharePoint when users click files from the SharePoint Document Library.

The main important part around the file handler is the manifest file, which defines the URL that will handle the requests, how authentication will be handled, and what elements will be rendered in the SharePoint UI (e.g. the context menu item).

In this project, the final manifest is generated by a set of TypeScript scripts under the `./tools/local-setup` directory. The entry point is `./tools/local-setup/bin/localsetup.ts`. The result of this series of scripts is the automatic generation of the final file handler application registration in your Azure Active Directory.

There are certain key aspects that you can customize as part of the manifest (outlined below).

Display name of the menu — To change the string used when a file is selected, go to `./tools/local-setup/tasks/ensure-app.ts` and change the value for the `appDisplayName` variable.



Login URL — To change the URL SharePoint uses to handle logging in, go to `./tools/local-setup/tasks/ensure-app.ts` and change the value for `redirectUri`.

Logout URL — To change the URL SharePoint uses to handle logging out, go to `./tools/local-setup/tasks/ensure-app.ts` and change the value for `logoutUrls`.

URL to edit a selected file — To change the URL SharePoint uses to handle redirection after a file is opened, go to `tools/local-setup/tasks/inject-manifest.ts` and change the value for the `"url"` field as part of the `actions` array (line 9). Notice how the file extension your file handler will operate on — in addition to other metadata — is also defined here.

Once these aspects are defined, you can proceed with the development environment preparation for the file handler.

Notice that, by default, you're defining `localhost:3000`. This means SharePoint will, for the time being, always use that URL when selecting a file, which means you need to have your local development server running.

Later, there are instructions on how to change that server once a production deployment exists.

4 Development Setup

- 1 Change your directory into the `pspdfkit-sp-online-filehandler` directory:

```
cd pspdfkit-sp-online-filehandler
```

- 2 Open the project in your preferred code editor and check which version of Node.js you're using:

```
node --version
```



Make sure you have Node.js version 16.19.0 installed. If you're using a different version of Node, switch to Node.js version 16.19.0 via nvm (node version manager) or asdf.



Tip: If you're using `asdf`, you can switch the Node version locally for this project. First, install Node.js 16.19.0 using the following command:

```
asdf install nodejs 16.19.0
```

Now, run `asdf local nodejs 16.19.0` to switch to the Node.js version you just installed. This will create a `.tool-versions` file in the root of your project with Node version 16.19.0. If you encounter any issues, check out the asdf documentation.



Tip: If you're using `nvm`, you can switch to Node.js 16.19.0 by typing the following command:

```
nvm use 16.19.0
```

3 Install the dependencies:

```
npm install
```

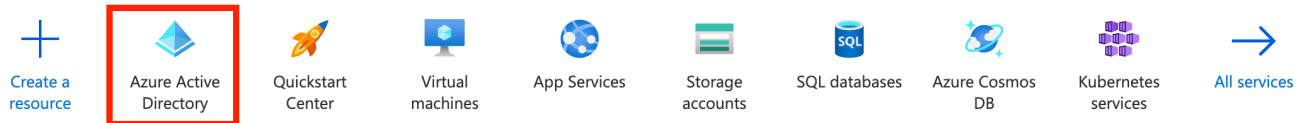
4 You need to set up a file handler app registration and load it with a manifest. To do this, you need to be an admin of a SharePoint tenant.



The app you're manually creating is used as a way of automating the registration of the actual file handler, but it isn't the file handler itself. At a later point, you'll no longer need this temporal app, and you can remove it from your Azure Active Directory.

- 5 Sign in to the Azure portal.
- 6 Switch to the tenant in which you want to register the application (if you have more than one).
- 7 Select Azure Active Directory.

Azure services



- 8 Under Manage, select App registrations > New registration.



Preview features



Diagnose and solve problems

Manage



Users



Groups



External Identities



Roles and administrators



Administrative units



Enterprise applications



Devices



App registrations



Identity Governance



Application proxy



Custom security attributes
(Preview)



Licenses



Azure AD Connect

0:00 / 0:07



9 Give it an easy-to-find name (e.g. “filehandler localdev setup”).

10 Add API permissions:

Microsoft Graph > Delegated permissions: `openid` , `Directory.AccessAsUser.All` .

Grant admin consent to the permissions.



0:00 / 0:24

11 Under the Authentication section:

Add a platform:

Add mobile and desktop applications.

Under Redirect URIs, select the MSAL only option.

Set Allow public client flows to Yes. This will treat the app as a public client. **Make sure this step isn't skipped, as otherwise, the whole process will later fail when running your dev-setup script.**

12 Copy the Application (client) ID and the Directory (tenant) ID, which you'll need in the next step.

0:00 / 0:13

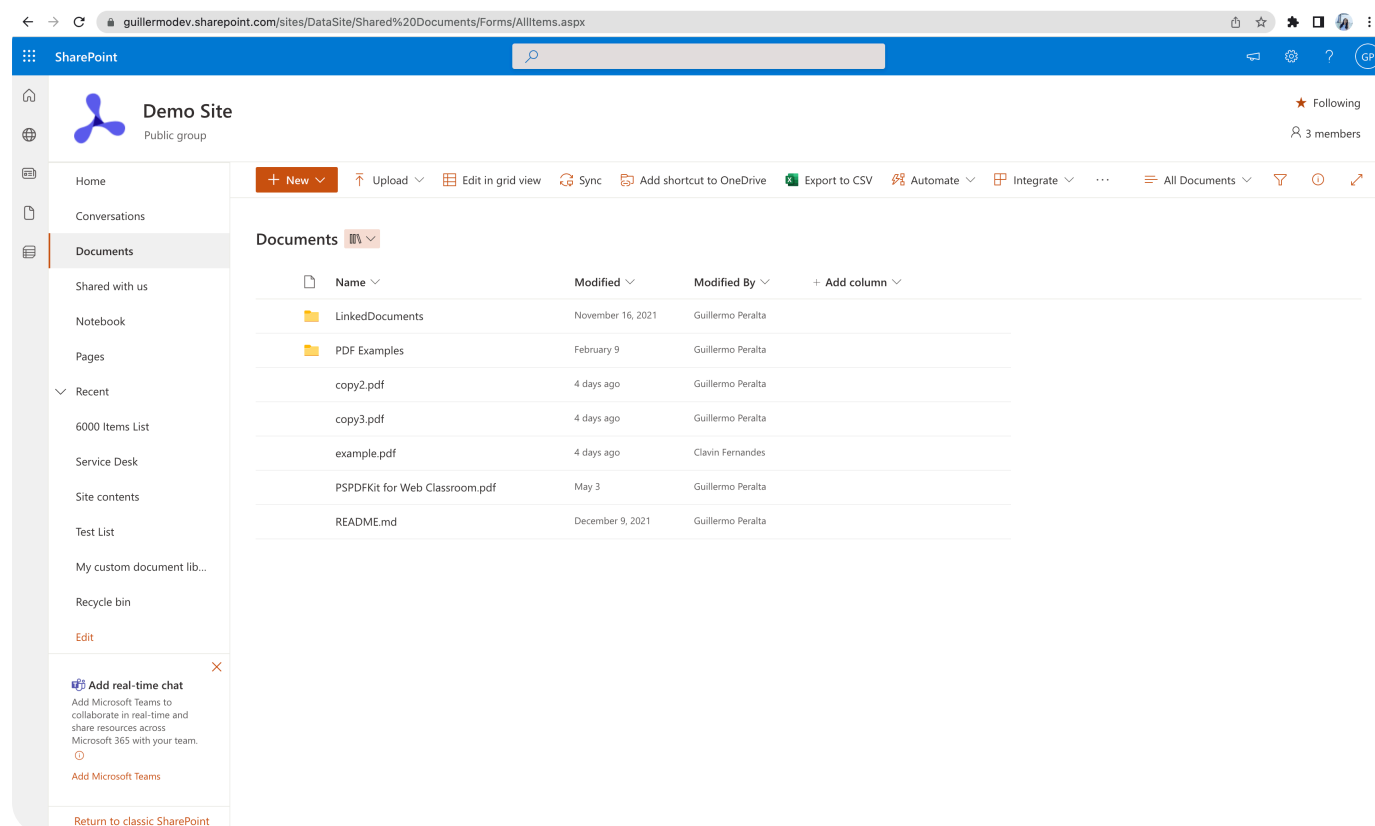
- 13 Make sure OpenSSL is installed on your system. Then, run `npm run dev-setup`, and supply the client and tenant IDs from the previous step when prompted. Follow the onscreen prompts to complete the device code authorization flow.



It'll take 24–48 hours for the manifest change to take effect. You can follow these steps for clearing the cache to speed this up.

- 14 Once setup is complete, you can optionally delete the registration helper application or leave it in place for future use.
- 15 Running the `dev-setup` command in step 4 generates an `.env.local` file in the root of this project. Review the values and ensure it was created correctly. It'll appear similar to what's shown below:


- 16 Run `npm run dev` and launch the file handler from within OneDrive or the SharePoint document library on a `.pdf` file.





5 Production Deployment


Follow these steps when you want to upload the file handler into a production server after development.

- 1 Run `npm run build` from your terminal.
- 2 Deploy the Next.js application somewhere. For this, you can use Vercel, Heroku, AWS Amplify, Digital Ocean, or another self-hosting option.
- 3 Sign in to the Azure portal.
- 4 Go to the Azure Active Directory. Then, follow the App registrations link in the sidebar.
- 5 Select your existing application (created from step 3 of the development setup).
- 6 Select Manifest from the sidebar, as shown in the screenshot below.




PSPDFKit-FileHandler-PDF





YES 


NO


<<

 Delete

 Endpoints


 Overview


 Quickstart

 Integration assistant

Contact us

Manage

 Branding & properties

 Got a second? We would love to hear from you.

^ Essentials

Display name

Application Client ID