

Building a website or web application can be confusing without prior experience. There are so many tasks that must be done aside from simply coding out your project. While YouTube tutorials are undeniably helpful, they can sometimes overwhelm us with too much information (demonstrating complicated or avant-garde setups), leaving us more confused on how to even begin. What is the purpose of a web server and how do we set it up? What's the difference between front-end development and back-end development? What is GitHub? Yes, ChatGPT has made it so anyone can code, but not everyone can build and maintain a usable web app. Writing this while building a digital visitor log web app for Imperial Distributors Canada Inc. has allowed me to understand a fraction of a fraction of software development and IT. Furthermore, one of the best ways of learning, aside from learning by doing, is being able to verbally explain concepts in a simple and understandable way.

### **Code Editor**

Odds are, if you have taken a programming course, you have installed a code editor or integrated development environment (IDE). This is a program that is used to write and edit computer code, basically a specialized version of a text editor like Notepad with a bunch of useful features. You can technically use Notepad for coding, if you save the files with the right extension (and then view the rendered version by right clicking the file and opening it using whatever browser you use), but for very complicated projects it is slow and frustrating. The code editor, in my case VS Code, edits files directly. So, when you open a *C:\Users\david.zhu\OneDrive - Imperial Distributors Canada Inc\Documents\docs\html files\GuestInfo.html* file in VS Code, it loads it from your local file system. Editing such files through the editor leads to those changes written directly to the corresponding files on your C: Drive in real time. Within the C: Drive, I tend to save my projects in documents (or sometimes 'desktop'), which you can control when setting up VS Code. I use VS Code as it's a free, open-source code editor (but has many features of an IDE) and can be used for web development and general programming. IntelliJ IDEA is a good IDE for Java dev, and PyCharm is a specifically designed IDE for python dev.

### **Backend vs Frontend**

There are several components to building a web app. The front-end refers to the user interface (sometimes called client side), which is the component of the app that users directly interact with. For example: visual elements, user flow, user interactions, etc. The focus here is on design and user experience, and commonly front-end developers use languages such as HTML, CSS, JavaScript and libraries such as React. Meanwhile, the back-end involves server-side logic, database, responding to client (browser) requests on server, and other processes. Back-end code handles requests from the front-end and

interacts with databases to send or retrieve data. Example: A user fills out a form on their browser, which is sent from the front-end to the server via an HTTP request, which the back-end updates into the database (server processes request and executes logic) - assuming you wanted to store form data. Common programming languages for the back-end include Python, PHP, C#, etc.

A programming stack is the combination of technologies that are used to build and run an app. An example would be the LAMP web server stack: Linux (OS), Apache (Web Server), MySQL (database), PHP/Python (server side code). For my project, instead of using Apache I used Windows as my OS and of course IIS as my web server – the rest remain the same.

### **IIS (Internet Information Services): Internet Information Services**

- A web server specific to the Microsoft .Net platform (also referred to as Windows Web Server)
  - o A server is a computer (hardware server) or program (software server) that provides services such as storing files, hosting websites, or running apps accessible to and used by other computers (a.k.a clients) over a network (system of interconnected devices that can communicate with each other through physical cables - ethernet - or wireless signals - Wi-Fi)
  - o Types of servers: Web servers for hosting websites and serving web pages to users (IIS or Apache), Database servers for managing and storing data for applications (MySQL), file servers for providing access to shared files and folders (Google Drive)
  - o Client-server model: Concept in computer networking where clients and servers communicate to perform tasks. Clients send a request, through a network, for data or services to servers, which sends back a response to the client. For example, web browsers (Google Chrome, MS Edge, Safari) are common client applications that request data from a web server to display web pages.
- IIS processes incoming app/web requests from default TCP ports (ex. default IIS server port for [https](#) traffic is 443 and default port for [http](#) traffic is 80 – used globally)
  - o A TCP port is a 16-bit number (1-65535) used by the Transmission Control Protocol (TCP) to distinguish multiple services (e.g. SMTP, DNS, etc.) running on a single device (host) by associating a specific service or process on a networked device with a logical communication endpoint (port number)
  - o IIS listens for incoming client requests on specific IIS TCP server ports (usually 443 and 80)

- The client device (e.g., your browser or app) uses a randomly chosen ephemeral port, typically in the range of 49152–65535, and opens the connection from that port to the server port

Device	IP Address	Port Used	Role
Client	192.168.1.10	51234	Source port
Server	203.0.113.5 (IIS)	80	Destination port

- Site Binding: rule in IIS servers that tells the server how to route incoming requests to a specific website – so that the user visits their intended website. Each binding consists of the Server IP address (or “all unassigned”), TCP destination port, Host name (for name-based routing - DNS), and Protocol (http or https) - and MAPS these to a website. Allows server to host and differentiate between multiple websites.
- When a client sends a request to this IIS server (by visiting a website), IIS first checks the incoming TCP connection by looking at the **destination IP address (IP address of the IIS server)** and destination port and host header (ex. `http://visit.idci.local`). It then checks if any binding matches the incoming request, and if there is a match, it will route the request to the corresponding site and the site's application pool. Two sites may have the same destination (server) IP address, in which case IIS uses the host name to figure out which site to serve.
- When a client (like a web browser) sends a request to a server, the destination IP address is the IP address of the IIS server
- Each website in IIS is assigned to an application pool, which IIS uses to run the site's code (e.g. PHP, ASP.NET, etc)

Website	IP Address	Port	Host Name	App Pool
Site A	All	80	<code>visit.idci.local</code>	<code>AppPool_A</code>
Site B	All	80	<code>api.idci.local</code>	<code>AppPool_B</code>

- From the above example of binding, a user visits <http://visit.idci.local> on their browser (client making a request to IIS server). The browser resolves `visit.idci.local` to an **IP address known as the destination IP address** (because computers can't understand domain names) via DNS. The browser then establishes the TCP connection to destination IP 10.0.0.5 on Port 80.

You can host multiple sites on the **same IP and port on the IIS server by setting different host headers**. If all binding criteria match, IIS routes request to site A and routes the request to Site A's file path and AppPool\_A. If no site matches, IIS returns 404 (site not found) error.

Simplified HTTPS request in IIS:

- A user inputs the website URL.
- The request hits the IIS web server, which uses HTTP.SYS to detect incoming client requests.
- IIS (using site bindings) matches the request to a site and routes it to the correct application pool
- The application pool is running in a worker process (w3wp.exe)
- That process executes your web app's code (HTML, PHP, .NET, etc.)
- The response is sent back to the client

IIS cannot run on Linux and Mac OS itself, and a Windows IIS implementation is the most dependable and stable. There are two ways a web server handles incoming requests from clients: Single-threaded model and multi-threaded model. The latter is used by IIS. A thread is the smallest unit of execution within a process – the smallest task that the OS can schedule to run on a CPU core. **Important: A thread is small in system design terms, not in terms of the amount of work that is being done.** In IIS and the context of hosting a web app, a process refers to w3wp.exe, the IIS worker process, which is the Windows program that handles http requests, manages memory, and is assigned to each Application Pool. An application pool is a container in IIS that isolates web applications from each other by running them in separate worker processes. Each worker process will use multiple threads to handle multiple web requests concurrently, with each thread processing one HTTP request (receiving a request, processing a request, and sending back a response) from start to finish.

Each thread runs a specific set of instructions, such as handling an http request or connecting to a database. A thread cannot be divided into “sub threads” that the OS runs separately. So, when a thread is assigned to handle a request, it runs **all** the instructions for that request sequentially (one by one) from start to end. In a more technical example, an IIS process is running and gets three incoming HTTP requests. A single thread handles all parts of a single HTTP request. It does not share the request with other threads. They are all part of one IIS process, and they are running in parallel. A good real-world analogy would be imagining each thread as a waiter at a restaurant. The waiter handles one customer (request) each - taking the order, bringing food, and handling payment for the

customer – until the request has been satisfied. Multiple waiters can help multiple customers at the same time.

In a single-threaded model, all incoming requests are handled one at a time, using only one thread. If one request takes a long time, other subsequent requests will have to wait, which makes it not scalable. In a multi-threaded model, each incoming request is handled on a separate thread, where each thread is pulled from a thread pool instead of being created to save time. This allows multiple users to connect simultaneously to the server. Once finished, the thread is returned to the pool and reused for the next request.

IIS runs using two layers: Kernel Mode and User Mode.

Kernel Mode (core of the computer): Code can execute any command and has full access to everything: Hardware, memory, etc. Runs very fast, but if something breaks, it can crash the entire computer. In IIS, HTTP.SYS runs in kernel mode (it listens for web requests from internet on ports 80 and 443 – incoming traffic – and routes requests to the correct IIS site based on bindings) and this process is initiated when the client inputs the website URL into the browser bar.

User Mode (where your website \*usually\* lives): More limited and subsequently safer. Executed code cannot access hardware or reference memory, meaning if a mistake is made, the consequences won't be as devastating as in kernel mode. In IIS, Admin tools, websites, worker process (w3wp.exe), and application pools all run in user mode.

IIS Web Server Features:

- Popular (use is widespread)
- Typically used for hosting ASP.NET (Active Server Pages .NET) status websites and web applications
  - o ASP.NET is a server-side web app framework developed by MS to allow devs to build dynamic websites/apps using languages like C#
- It can also be used as an FTP server (File Transfer Protocol), which is used to transfer files between a client and server on a computer network. IIS handles file uploads and downloads in this instance.
- Application pools with zero to many IIS worker processes running
- Authentication
- Security (managing TLS certificates, binding so HTTPS can be enabled)

## **Setting Up an IIS Server:**

FYI: **staging.example.com** is a hostname, **staging** is the subdomain, **example.com** is the domain, **.com** is the top-level domain (TLD)

As IIS is a Windows feature, setting it up is only a matter of enabling it rather than installing it. On Windows 10/11 PCs, you need to access the **control panel** (go to Windows Search in the taskbar and type Control Panel -> Programs -> Programs and Features -> Turn Windows features on and off), where you'll have the option of clicking on a "Turn Windows features on and off" link. This will open a "Windows Features" box, which will give you the opportunity to turn more Windows features on and off. For the purposes of a simple web app, enable "Internet Information Services" and make sure "Web Management Tools" and "World Wide Web Services" are checked. Also make sure that .NET, Application, Initialization, and CGI is checked (World Wide Web Services -> Application Development Features" -> .NET, Application, Initialization, and CGI). You can expand each feature to access further app dev features but note that less is better, and you can always install features on a as needed basis. This has more limited features compared to using Windows Server and is more for testing or small home projects.

Alternatively, if you are using a Windows Server, you can go to Server Manager and click "Add roles and features", select "Role-based or feature-based installation", select "Select a server from the server pool", in Server Roles select "Web Server (IIS)". Don't need to add any Features or Role Services than what has already been done. Then click Install. If it is done right, you should be able to see IIS on the Server Manager dashboard. On Server Manager -> Local Server, you can disable the IE Enhanced Security Configuration for TESTING to remove pop up messages but in an actual production environment it must be turned ON.

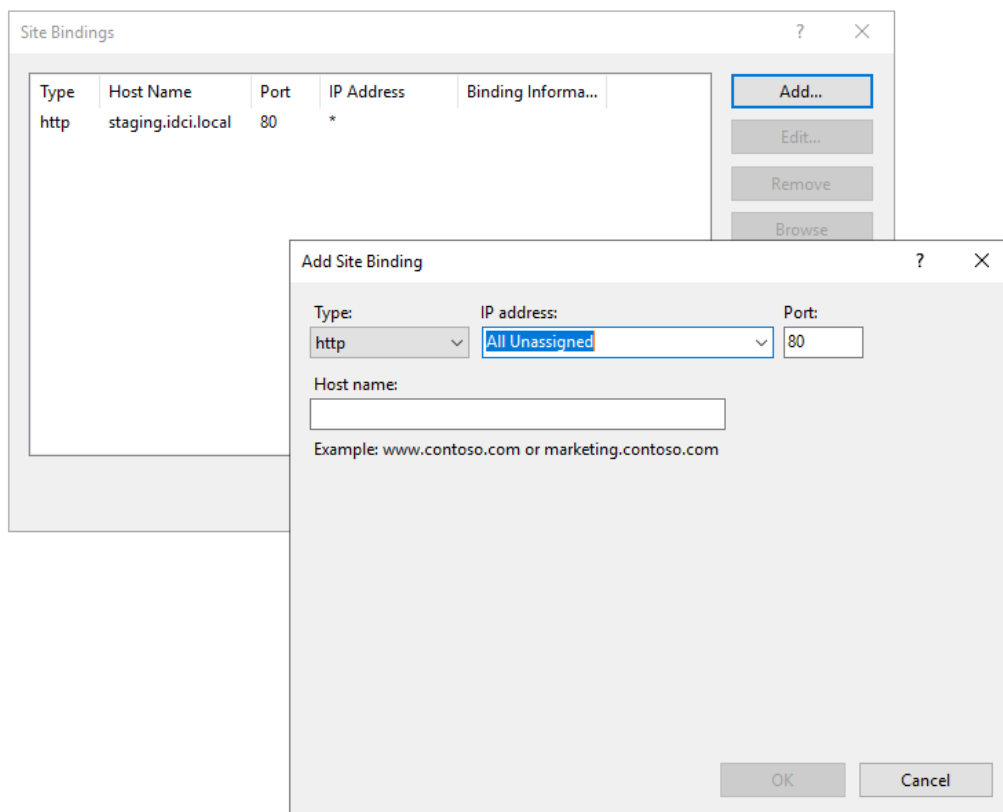
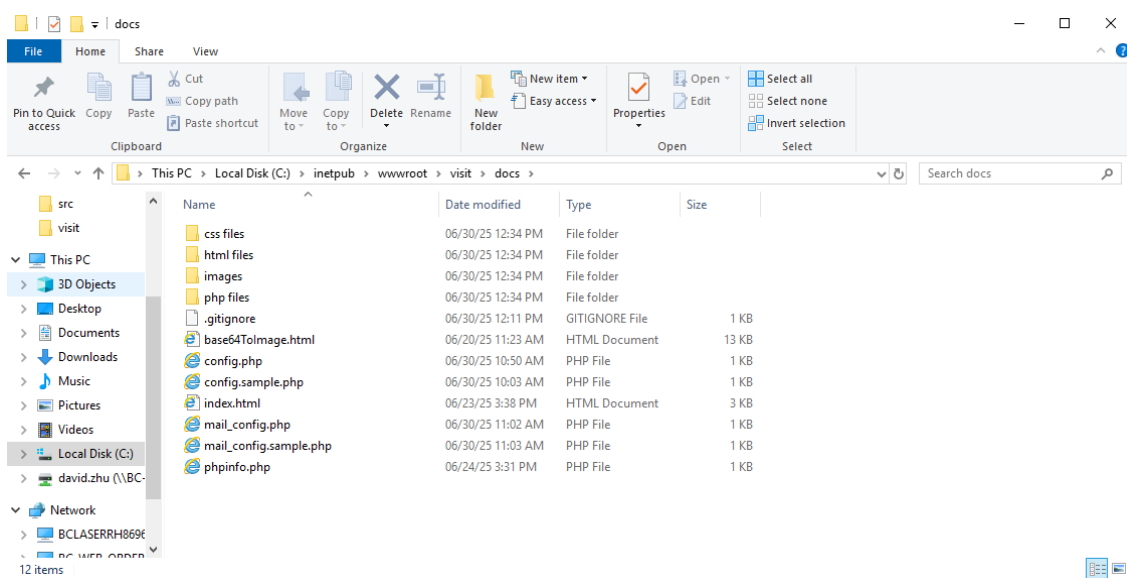
In both methods (for setting up IIS in different environments – if you are doing this all on your personal laptop you likely won't have Windows Server (most likely Windows 10/11)), if you access <http://localhost> on your browser, you should be able to access the default web page, which indicates you have successfully set up a web server.

It is recommended that you install the Web Platform Installer Tool. It acts like an app store for IIS-related components, so you can quickly install IIS extensions, PHP, MySQL, etc. Open your web browser and search for Web Platform Installer. click on "Free Download" to download it and click on Save File when prompted. Launch the file (open the file in downloads), run it, and once it's installed you can install components and models without having to manually configure it – everything will be done by the Web Platform Installer tool.

**IMPORTANT NOTE: RECENTLY MS HAS DEPRECATED WEB PLATFORM INSTALLER** so it may actually be better now to manually install and configure PHP and MySQL – you'll learn more anyways.

To create a custom domain entry (so a user can visit your web page by typing a URL in browser), go to IIS Manager and go to Sites. There will be a Default Web Site under sites, which is what you see when you access <http://localhost>. Right click on Sites and select “Add Website”. Set “Site Name”, which is the name that will appear on the IIS Manager, as your domain name for simplicity – note that “Site Name” does NOT affect the URL that users type into their browsers. The “Physical path” is where the website files – source code – are located. The default web root directory for IIS is the wwwroot folder in the inetpub folder in the C Drive (C:\inetpub\wwwroot). All HTML, PHP, and other web files should be placed here in order to be accessible via a web browser. In the case of my project, all code files are stored in docs folder which is in the visit folder (which is in the wwwroot folder). I have created a visit folder specifically for this project since I am hosting multiple web apps on IIS. You can see <http://visit.idci.local/> is pointing to "C:\inetpub\wwwroot\visit\docs\" in IIS setup and simply imagine it as "http://visit.idci.local" = "C:\inetpub\wwwroot\visit\docs\".

The screenshot shows the 'Add Website' dialog box in IIS Manager. The 'Site name' field is set to 'visit-staging'. The 'Application pool' is also set to 'visit-staging'. The 'Content Directory' section shows the 'Physical path' as 'C:\inetpub\wwwroot\visit-staging\docs'. The 'Binding' section shows 'Type' as 'http', 'IP address' as 'All Unassigned', and 'Port' as '80'. The 'Host name' field is empty. At the bottom, the 'Start Website immediately' checkbox is checked. The 'OK' and 'Cancel' buttons are visible at the bottom right.

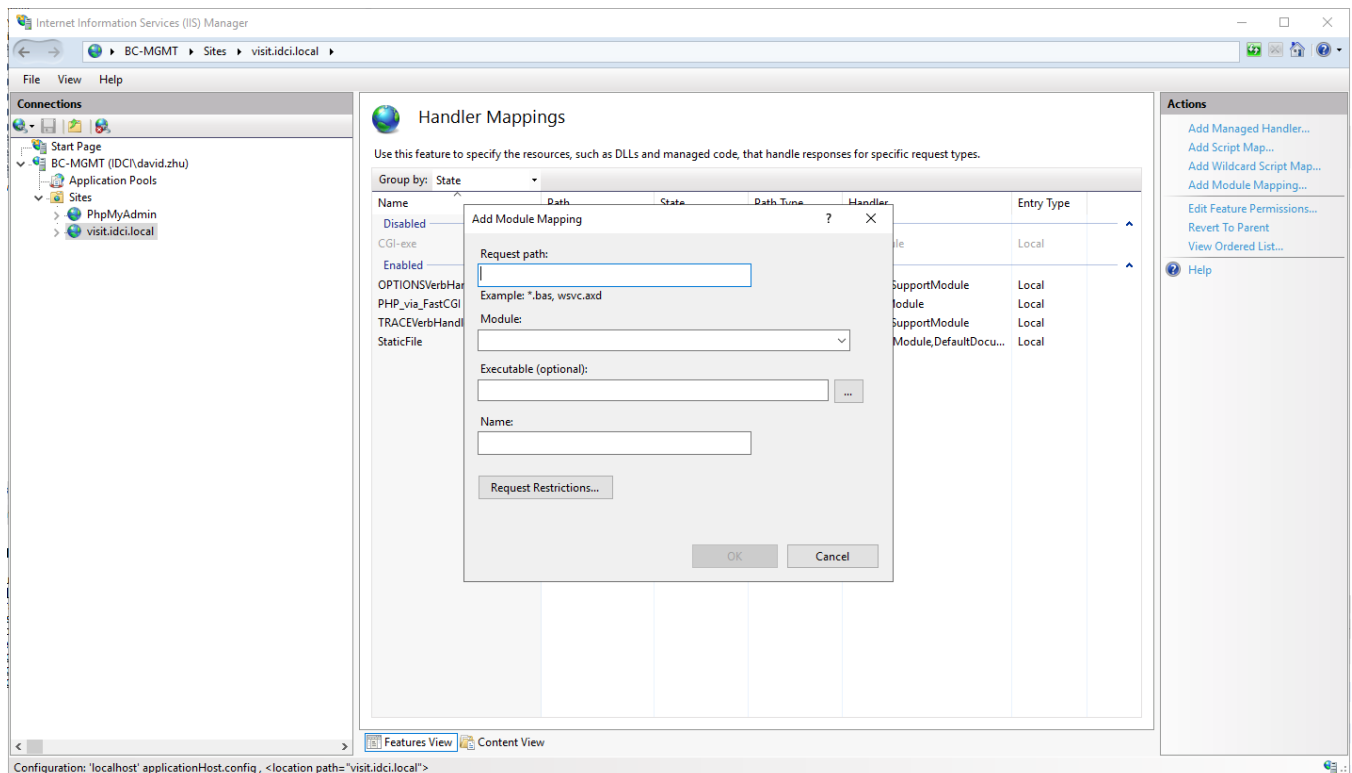


Note that the IP address is the IP address of the IIS server.

**Handler Mappings:** In IIS Manager, handler mappings control how IIS processes different specific file types or URL paths requested from the website or webapp



(.php, .html, .css, .js). Since I used PHP as my back-end scripting language, I need a handler for .php otherwise IIS won't serve .php files. StaticFile and TraceVerbHandler are both automatically installed and configured by default when you enable IIS and the "Static Content" feature in Windows. StaticFile handles requests for static files (usually front-end code) like .html, .css, .js, .jpg while TraceVerbHandler handles HTTP requests with the TRACE verb (used for debugging). The latter is not often used. To install and register FastCGI module, download and install PHP manually. Once you have done that, open IIS Manager, click on your site in the left panel, open Handler Mappings, Click "Add Module Mapping..." on the Actions Panel and fill in the details.



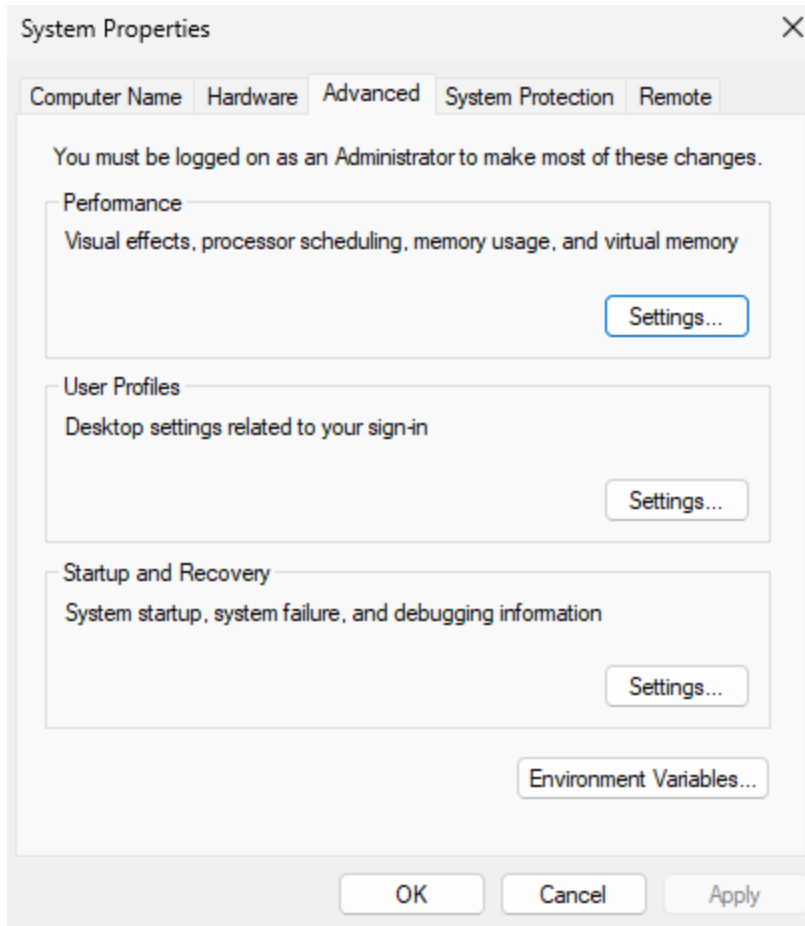
Request Path: \*.php

Module: FastCgiModule

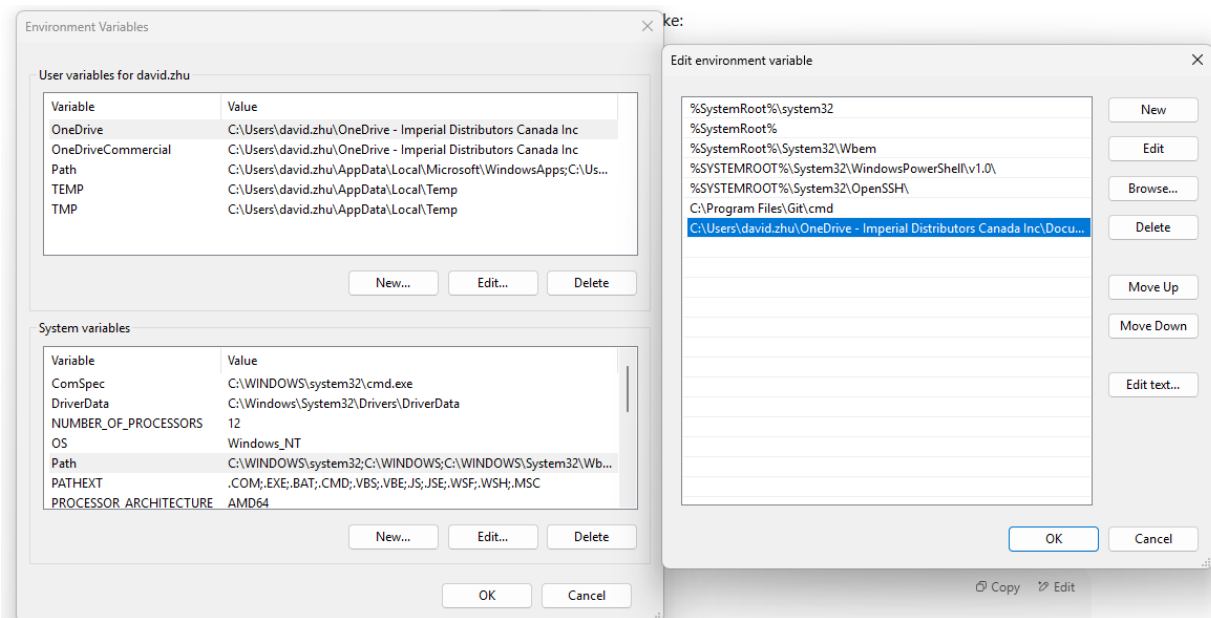
Executable : C:\Path\To\php-cgi.exe (file path of actual PHP install)

Name: PHP\_via\_FastCGI

For PHP Download, make sure to download the x64 Non-Thread Safe Version of PHP. [PHP For Windows: Binaries and sources Releases](#) -> download the ZIP file for x64 for the latest version -> extract zip to C: drive (technically you can extract it anywhere – documents folder, downloads, desktop, etc...). Add PHP to System Path, by searching for "Edit the System Environment Variables" in your taskbar.



Go to Environment Variables -> System variables -> Path -> Edit -> New. Copy and paste the path to the PHP folder (ex. C:\php). To test, open vs code terminal or command prompt and type *php -v*, which should return the version info.



In my case my path was C:\Users\david.zhu\OneDrive - Imperial Distributors Canada Inc\Documents\php. Setting the system variable path tells Windows where to look for php.exe when you run a command in terminal or command prompt. When you type a command like php, Windows looks through all the folders listed in the Path variable and tries to find a file named php.exe. If your PHP folder (which must contain php.exe) isn't in your Path, you'll get an error. Basically, the system variable Path allows me to store my PHP folder anywhere I want, and as long as that folder is added to the Path, I can run commands like accessing the PHP version info by simply typing *php -v* in the terminal or command prompt without having to type in the full folder path every time.

To start the built-in PHP server, type *php -S localhost:8000*, and go to <http://localhost:8000> in your browser. This allows you to test your PHP code without needing a full fledged server like Apache, IIS, or XAMPP. You can also connect to a database if you want but note that this is only for local testing.

Copy and rename php.ini-development to php.ini and place it in the same folder as php-cgi.exe. \*\*Make sure it is the right config file; there are two config files, development and production. Open php.ini and enable (by uncommenting) the following common extensions:

`extension=mysqli` allows PHP to connect to MySQL database w/ improved functions  
`extension=pdo_mysql` enables PDO support for MySQL, more flexible than above  
`extension=mbstring` handles multibyte strings

`extension=openssl` Enables encryption (SSL/TLS support) needed for secure connections and tools like composer

And edit the following lines:

`cgi.force_redirect = 0` required for running PHP via IIS or CGI properly

`fastcgi.impersonate = 1`

`fastcgi.logging = 0`

`display_errors = On` Shows error messages in browser

`error_reporting = E_ALL` Enables error reporting

`extension_dir = "ext"`

All crucial for PHP to function correctly with IIS or other servers. Then set default document. If it's index.php you need to manually add it, but since my default document is index.html you can just leave it as is. To test PHP (and make sure you configured correctly), create a file called phpinfo.php in your site root (wwwroot) and have the file contain the following:

```
<?php phpinfo(); ?>
```

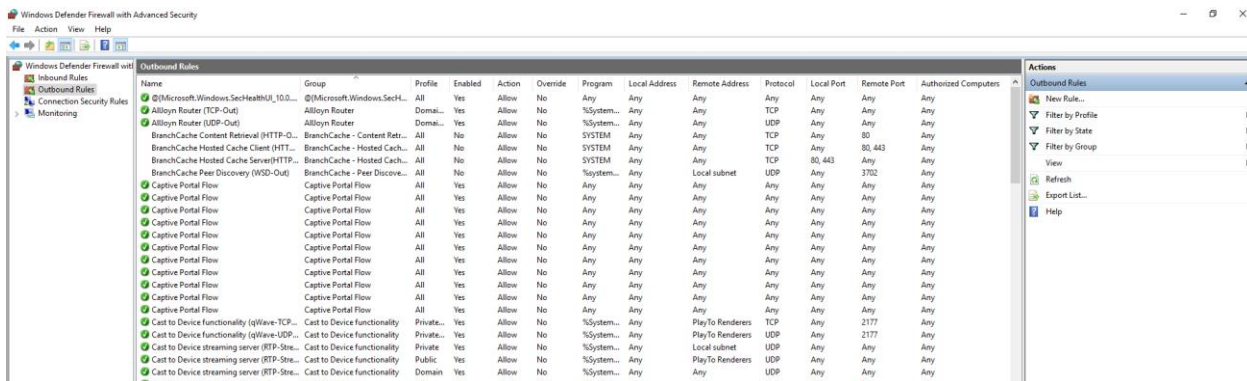
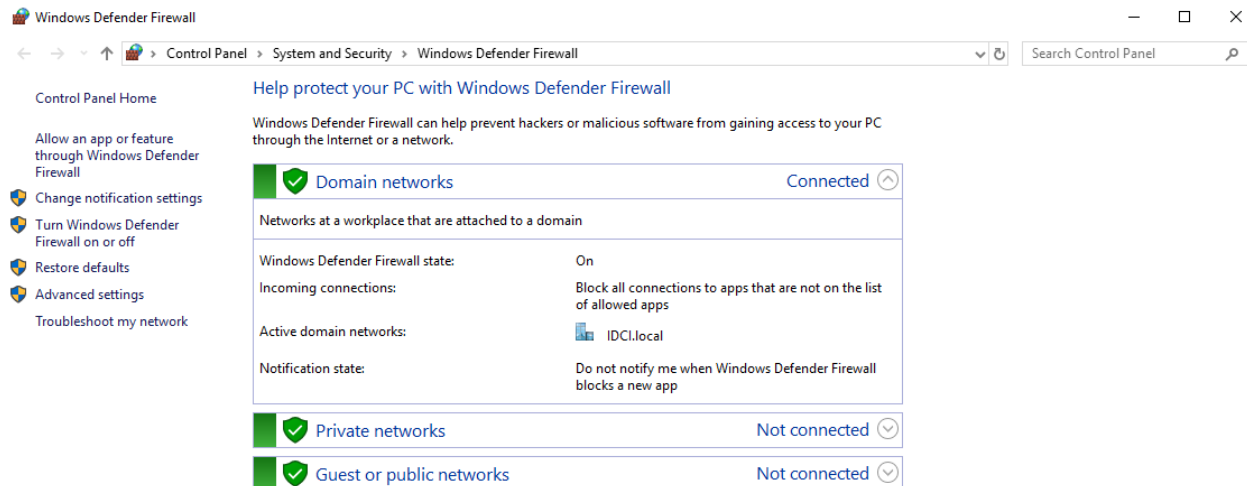
Then, on your browser, visit <http://domainName/phpinfo.php>. You should see the PHP info page.

You can also see your error log, which records the error type, severity level, file path/line number, timestamp, and more when PHP detects. To locate file path of the error log, first make sure its error logging is enabled by checking the PHP Info page. If it's not, enable it. If it is, go to php.ini file, Control+F to find error\_log, which will show the file path (most likely C:/Error/php-error.log).

ignore_user_abort	Off	Off
implicit_flush	Off	Off
include_path	.;C:\php\pear	.;C:\php\pear
input_encoding	no value	no value
internal_encoding	no value	no value
log_errors	On	On
mail.add_x_header	Off	Off
mail.force_extra_parameters	no value	no value
mail.log	no value	no value
mail.mixed_if_and_crlf	Off	Off
max_execution_time	30	30
max_file_uploads	20	20
max_input_nesting_level	64	64
max_input_time	60	60
max_input_vars	1000	1000

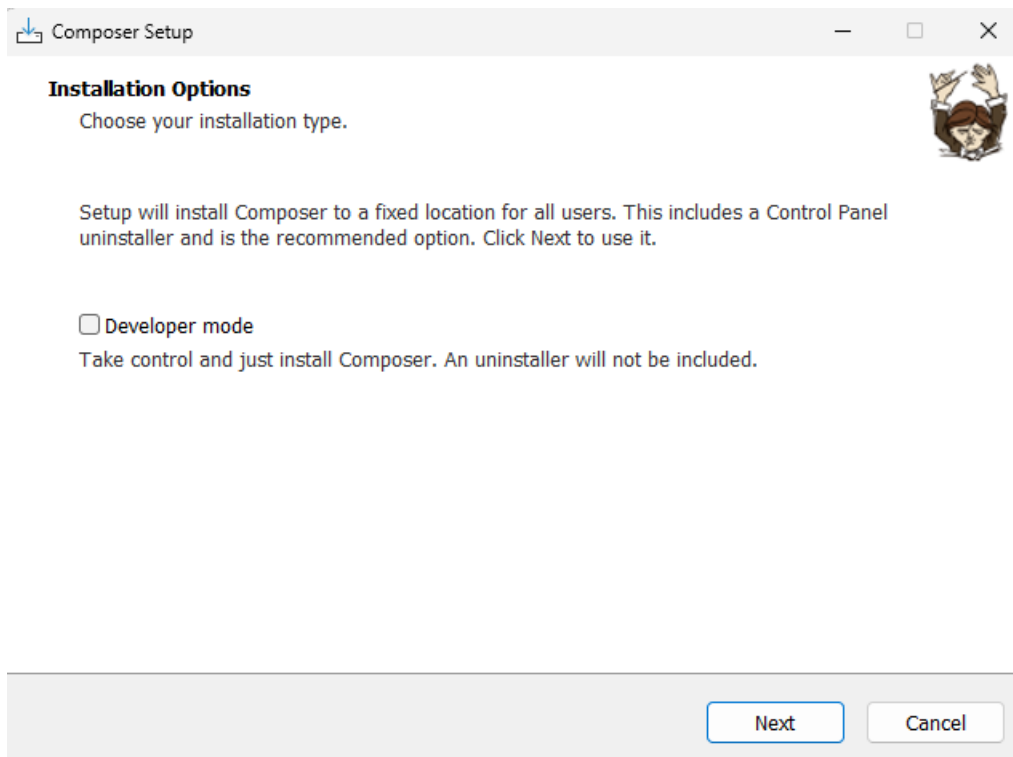
```
php-error.log - Notepad
File Edit Format View Help
[09-Jul-2025 12:09:46 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 12:25:39 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 12:27:23 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 12:29:08 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 12:37:53 America/Vancouver] PHP Warning: Undefined array key "contact" in C:\inetpub\wwwroot\visit\docs\php files\send-mail.php on line 61
[09-Jul-2025 12:37:55 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 12:45:51 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 15:22:15 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 15:34:32 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 15:52:31 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 15:52:38 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 15:54:33 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 15:54:40 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 16:07:15 America/Vancouver] PHP Warning: PHP Request Startup: Unable to create temporary file, Check permissions in temporary files directory. in Unknown on line
[09-Jul-2025 16:07:15 America/Vancouver] PHP Warning: PHP Request Startup: POST data can't be buffered; all data discarded in Unknown on line 0
[09-Jul-2025 16:07:15 America/Vancouver] PHP Warning: session_start(): Session cannot be started after headers have already been sent in C:\inetpub\wwwroot\visit\docs\php fil
[09-Jul-2025 16:07:15 America/Vancouver] PHP Warning: Cannot modify header information - headers already sent in C:\inetpub\wwwroot\visit\docs\php files\process_policy.php on
[09-Jul-2025 16:19:46 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 17:37:23 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 17:43:23 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 18:04:23 America/Vancouver] PHP Warning: Undefined array key "company" in C:\inetpub\wwwroot\visit\docs\php files\process_policy.php on line 71
[09-Jul-2025 18:04:23 America/Vancouver] PHP Warning: Undefined array key "company" in C:\inetpub\wwwroot\visit\docs\php files\process_policy.php on line 72
[09-Jul-2025 18:05:59 America/Vancouver] PHP Fatal error: Maximum execution time of 30 seconds exceeded in C:\inetpub\wwwroot\visit\docs\php files\process_policy.php on line
[09-Jul-2025 18:05:59 America/Vancouver] PHP Fatal error: Maximum execution time of 30 seconds exceeded in C:\inetpub\wwwroot\visit\docs\php files\process_interview.php on li
[09-Jul-2025 18:06:18 America/Vancouver] PHP Warning: Undefined array key "company" in C:\inetpub\wwwroot\visit\docs\php files\process_policy.php on line 71
[09-Jul-2025 18:06:18 America/Vancouver] PHP Warning: Undefined array key "company" in C:\inetpub\wwwroot\visit\docs\php files\process_policy.php on line 72
[09-Jul-2025 18:06:21 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 18:07:11 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 18:12:48 America/Vancouver] Email has been sent to david.zhu@idci.ca
[09-Jul-2025 18:17:01 America/Vancouver] Email has been sent to david.zhu@idci.ca
[10-Jul-2025 08:55:04 America/Vancouver] Email has been sent to david.zhu@idci.ca
[10-Jul-2025 08:58:35 America/Vancouver] Email has been sent to david.zhu@idci.ca
[10-Jul-2025 08:59:55 America/Vancouver] Email has been sent to david.zhu@idci.ca
[10-Jul-2025 09:04:23 America/Vancouver] Email has been sent to david.zhu@idci.ca
[10-Jul-2025 09:06:19 America/Vancouver] Email has been sent to david.zhu@idci.ca
```

Also make sure that you have configured DNS if you want multiple devices across a network to access your IIS site by name or if you want public internet access via a real domain. Go to Control Panel > System and Security > Windows Defender Firewall > Advanced Settings > Inbound rules. This ensures incoming traffic on these ports can reach your IIS server from other devices. In this case I have set the Local Port for my visitor staging site to 8080 and the Local Port for my production site to 90.

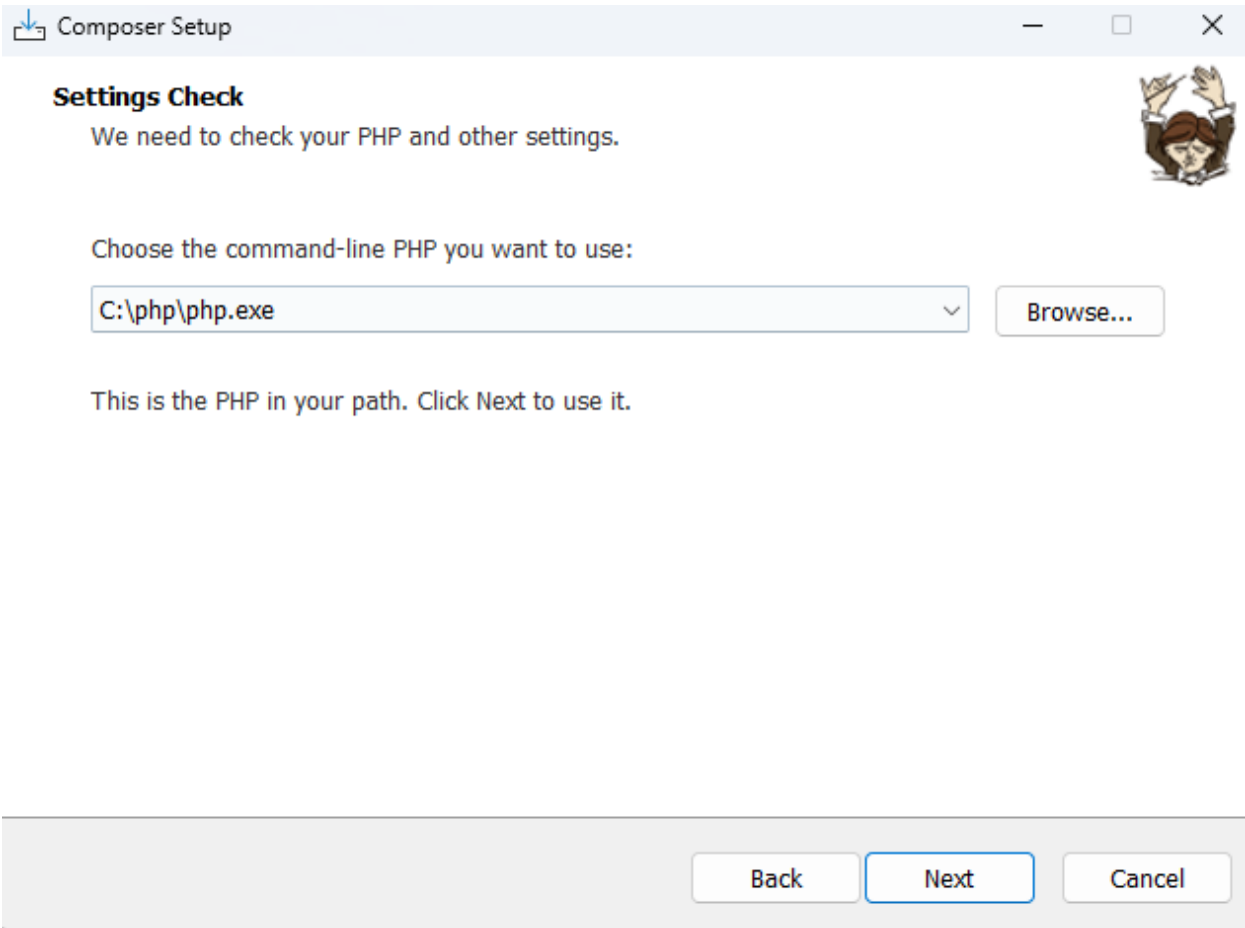


## Installing Composer:

To install Composer, download and run [Composer-Setup.exe](#). Make sure PHP is already installed. Clicking on the downloaded file will open the prompt:



Click Next.

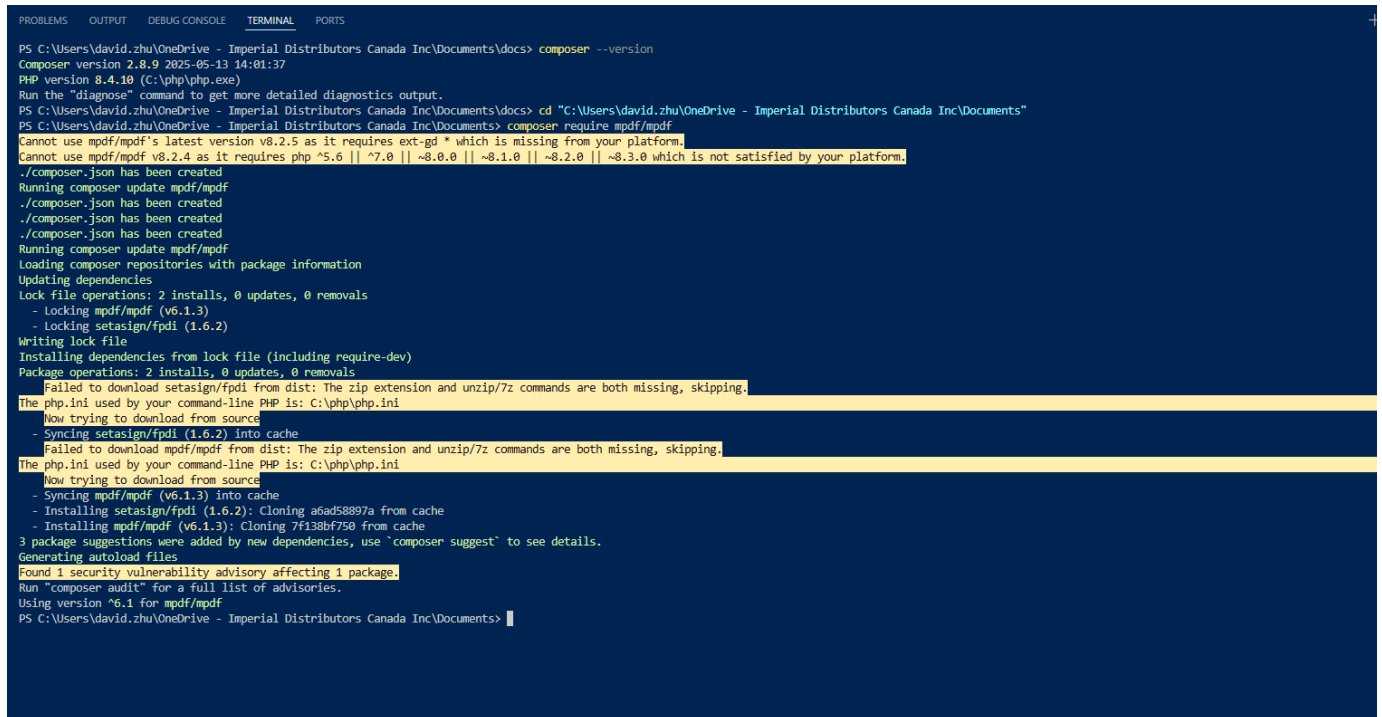


Note that the path is now in my C: drive since **C:\Users\david.zhu\OneDrive - Imperial Distributors Canada Inc\Documents\php** (OneDrive Documents Folder) is too long and may cause path length issues.





can then easily and quickly install add-ons like mPDF to your code. See below image (snapshot of VS Code terminal) - zoom in to see more details.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\david.zhu\OneDrive - Imperial Distributors Canada Inc\Documents\docs> composer --version
Composer version 2.8.9 2025-05-13 14:01:37
PHP version 8.4.10 (C:\php\php.exe)
Run the "diagnose" command to get more detailed diagnostics output.
PS C:\Users\david.zhu\OneDrive - Imperial Distributors Canada Inc\Documents\docs> cd "C:\Users\david.zhu\OneDrive - Imperial Distributors Canada Inc\Documents"
PS C:\Users\david.zhu\OneDrive - Imperial Distributors Canada Inc\Documents> composer require mpdf/mpdf
Cannot use mpdf/mpdf's latest version v8.2.5 as it requires ext-gd * which is missing from your platform.
Cannot use mpdf/mpdf v8.2.4 as it requires php ^5.6 || ^7.0 || ~8.0.0 || ~8.1.0 || ~8.2.0 || ~8.3.0 which is not satisfied by your platform.
./composer.json has been created
Running composer update mpdf/mpdf
./composer.json has been created
./composer.json has been created
./composer.json has been created
Running composer update mpdf/mpdf
Loading composer repositories with package information
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
- Locking mpdf/mpdf (v6.1.3)
- Locking setasign/fpdf (1.6.2)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
Failed to download setasign/fpdf from dist: The zip extension and unzip/7z commands are both missing, skipping.
The php.ini used by your command-line PHP is: C:\php\php.ini
Now trying to download from source
- Syncing setasign/fpdf (1.6.2) into cache
Failed to download mpdf/mpdf from dist: The zip extension and unzip/7z commands are both missing, skipping.
The php.ini used by your command-line PHP is: C:\php\php.ini
Now trying to download from source
- Syncing mpdf/mpdf (v6.1.3) into cache
- Installing setasign/fpdf (1.6.2): Cloning a6ad58897a from cache
- Installing mpdf/mpdf (v6.1.3): Cloning 7f138bf750 from cache
3 package suggestions were added by new dependencies, use 'composer suggest' to see details.
Generating autoload files
Found 1 security vulnerability advisory affecting 1 package.
Run "composer audit" for a full list of advisories.
Using version ^6.1 for mpdf/mpdf
PS C:\Users\david.zhu\OneDrive - Imperial Distributors Canada Inc\Documents>
```

Make sure to also enable the following extensions in php.ini file:

extension=gd

extension=zip

I also am using the latest version of PHP, which mPDF doesn't yet support. I would need to downgrade PHP to use the latest version of mPDF. I decided to just use an older version of mPDF.

Once you have set up configured and set up everything:

1. Download the source code.
2. Create a docs subfolder in the wwwroot folder in the C:drive.
3. Copy and paste the vendor folder from the mPDF library to the same level in the file structure as the docs folder.
4. Create a pdfrecord folder to store your PDF records.

It should look something like this:

This PC > Local Disk (C:) > inetpub > wwwroot > visit				
Name	Date modified	Type	Size	
docs	07/11/25 10:50 AM	File folder		
pdfrecord	07/17/25 3:46 PM	File folder		
vendor	07/08/25 1:28 PM	File folder		
composer.json	07/08/25 11:44 AM	JSON File	1 KB	
composer.lock	07/08/25 11:43 AM	LOCK File	18 KB	
web.config	06/23/25 11:25 AM	CONFIG File	1 KB	

This PC > Local Disk (C:) > inetpub > wwwroot > visit > docs >				
Name	Date modified	Type	Size	
css files	07/11/25 10:50 AM	File folder		
html files	07/17/25 3:42 PM	File folder		
images	07/09/25 11:12 AM	File folder		
php files	07/17/25 10:49 AM	File folder		
.gitignore	06/30/25 12:11 PM	GITIGNORE File	1 KB	
base64ToImage.html	06/20/25 11:23 AM	HTML Document	13 KB	
config.php	07/07/25 3:46 PM	PHP File	1 KB	
config.sample.php	06/30/25 10:03 AM	PHP File	1 KB	
index.html	07/07/25 3:46 PM	HTML Document	3 KB	
mail_config.php	06/30/25 11:02 AM	PHP File	1 KB	
mail_config.sample.php	06/30/25 11:03 AM	PHP File	1 KB	
phpinfo.php	06/24/25 3:31 PM	PHP File	1 KB	

**Note:** You must have a properly configured MySQL database with the required tables already created, as well as a valid email account set up for sending emails through the application.

## **GitHub:**

Your dev environment/code editor, in my case VS Code, edits files directly. So, when you open a project in VS Code, it loads files from your local file system. Editing such files through the editor leads to those changes written directly to the corresponding files on your disk in real time. A Git repository is a project folder that has been initialized with Git using the command `git init`. This repo tracks changes to files in the folder that contains your source code. When you edit a file in VS Code, (assuming auto save) Git detects the edits (changes) and marks the modified files as uncommitted files. You can tell a file has uncommitted changes on the Explorer Section of VS Code since they appear in a different color. There are three types of file states: Untracked files, Modified files, and Staged files. Untracked are files that Git isn't tracking (files that have been created but haven't been added to Git using the command `Git add .`).

Modified files are tracked files that have been edited since the last commit but haven't been staged (staging is the process of **SELECTING** files to be committed). Staged Files are modified files that you have marked for commit (using `git add .`)

Now let's distinguish Github from git. Git is a distributed version control system (allows each dev to have a local copy of entire project history), designed to track changes in source code during dev. Git is a command-line tool maintained by Linux community BUT itself does NOT PROVIDE hosting or COLLABORATION feature. Git operates locally on computer. Github is a web-placed platform BUILT ON TOP OF Git. It hosts Git repos (repository is a folder with version control) and adds features like issue tracking and pull requests. Most importantly, it acts as a centralized place to store, share, and collaborate on projects with more than one contributor.

GitHub utilizes Git, a version control system, to manage code repositories. The default created and named "main branch", is the primary branch where all changes are initially made. It is the central branch for pull requests and code commits. You can change the name of the default branch, or even which branch is designated as the default branch. Imagine there are several developers working on different parts of the same project. Each developer would have a separate branch from the default branch that they would commit changes to, taking care not to commit to the main branch. The idea is that if someone's code has a bug, their bug will not affect the other developers' code until it has been resolved.

GitHub repository consists of two components: Local Repository and Remote Repository. The local repository, which lives on your computer, is updated automatically when changes are made and committed using Git commands in the local dev environment. The Remote Repository, which lives on Github's servers, is a centralized repository where code is shared – think a Google Docs of sorts.

#### Pull Request:

A pull request (PR) is a mechanism in github that allows developers to propose, review, and integrate changes from one branch into another, usually into the default main branch. Basically, developers do not always write perfect code, and so there may be errors/bugs in the code. Sometimes the code is also improperly written, messy, or has logical errors that the developer fails to catch. To prevent faulty code from affecting the rest of the project, developers will create a new branch in remote repository (using Github web interface) and commit their changes to the new branch. You can also create a new branch directly on VSCode using its integrated git support. They then open a pull request to merge their changes into the main branch. The rest of the team review this pull request and may suggest improvements. After receiving approval, the pull request is merged with the main branch. Below are some terminal commands that you may commonly use.

**TERMINAL: (before you do all of this, make sure you have created the Git repository by typing in git init)**

1. Check what changed (modified, new, deleted files):
  - a. `git status`
2. Stage all Changed files:
  - a. `git add .`
3. Commit changes:
  - a. `git commit -m "Describe what has been changed"`
4. Pull the latest changes from remote (Github) to avoid conflicts:
  - a. `git pull origin main`
5. Push changes to Github
  - a. `git push origin main`

\*Try to refrain from making changes to your code directly on Github web interface code section (you will need to pull the code to your local repository to see the most up-to-date version, although this becomes a little more complicated if there are conflicts between the two versions)

VS Code Git GUI (Source Control Panel)

1. Open Source Control Panel
2. Go to *Changes* file (to stage all changes at once, click “+” next to *Changes* header)
3. Write commit message in textbox labeled *Message*
4. Commit changes by clicking checkmark button above message box

Pull latest changes from Github

- a. Click three dot menu ... from top-right of Source Control Panel and choose *Pull* -> select branch (usually *main*)
5. Push your commits
    - a. Click three dot menu ... again and choose *Push* to upload commits to Github

Testing:

Now you've completed development of my visitor web application, which is currently running on a virtual server with a configured MySQL database, PHP, Windows IIS server, and all source code files in place. At the moment, whenever I make changes to the source code in my local VS Code environment, I have to manually copy and paste the updated files into the live application folder on the remote server. This workflow poses significant risks: if the code contains any errors, it can cause the live site to crash, and any testing I perform is stored in the same database as real visitor data, which for obvious reasons is bad. The solution? **Staging Environments.**

A staging environment is a safe copy of your live website, database, and environment where you can test code changes and work with test data only. Once tested, you can deploy your changes to the live environment.

You can use the same IIS server for both live and staging environments. Essentially, you add a new website to your IIS Manager that uses different hostnames (but can use the same port). Very similar to the abovementioned process, refer to it if you are unfamiliar. Also duplicate the MySQL database and ensure that the staging site points towards the staging database. Remember that config files such as DB or SMTP credentials may be different, so do not overwrite these files.