Configure Python web apps for IIS

12/06/2018 • 4 minutes to read •

In this article

When using Internet Information Services (IIS) as a web server on a Windows computer (including Windows virtual machines on Azure, Python apps must include specific settings in their *web.config* files so that IIS can properly process Python code. The computer itself must also have Python installed along with any packages the web app requires.

Note

This article previously contained guidance for configuring Python on Azure App Service on Windows. The Python extensions and Windows hosts used in that scenario have been deprecated in favor of Azure App Service on Linux. For more information, see **Publishing Python Apps to Azure App Service (Linux)**. The previous article, however, is still available on **Managing App Service on Windows with the Python extensions**.

Install Python on Windows

To run a web app, first install your required version of Python directly on the Windows host machine as described on Install Python interpreters.

Record the location of the `python.exe` interpreter for later steps. For convenience, you can add that location to your PATH environment variable.

Install packages

When using a dedicated host, you can use the global Python environment to run your app rather than a virtual environment. Accordingly, you can install all of your app's requirements into the global environment simply by running `pip install -r requirements.txt` at a command prompt.

Set web.config to point to the Python interpreter

Your app's *web.config* file instructs the IIS (7+) web server running on Windows about how it should handle Python requests through either HttpPlatform (recommended) or FastCGI. Visual Studio versions 2015 and earlier make these modifications automatically. When using Visual Studio 2017 and later, you must modify *web.config* manually.

Configure the HttpPlatform handler

The HttpPlatform module passes socket connections directly to a standalone Python process. This pass-through allows you to run any web server you like, but requires a startup script that runs a local web server. You specify the script in the `<httpPlatform>` element of *web.config*, where the `processPath` attribute points to the site extension's Python interpreter and the `arguments` attribute points to your script and any arguments you want to provide:

XML Copy

```xml
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>
    <handlers>
      <add name="PythonHandler" path="*" verb="*"
modules="httpPlatformHandler" resourceType="Unspecified"/>
    </handlers>
    <httpPlatform processPath="c:\python36-32\python.exe"
                  arguments="c:\home\site\wwwroot\runserver.py
--port %HTTP_PLATFORM_PORT%"
                  stdoutLogEnabled="true"
                  stdoutLogFile="c:\home\LogFiles\python.log"
                  startupTimeLimit="60"
                  processesPerApplication="16">
      <environmentVariables>
        <environmentVariable name="SERVER_PORT" value="%HTTP_-
```

```
PLATFORM_PORT%" />
      </environmentVariables>
    </httpPlatform>
  </system.webServer>
</configuration>
```

The `HTTP_PLATFORM_PORT` environment variable shown here contains the port that your local server should listen on for connections from localhost. This example also shows how to create another environment variable, if desired, in this case `SERVER_PORT`.

Configure the FastCGI handler

FastCGI is an interface that works at the request level. IIS receives incoming connections and forwards each request to a WSGI app running in one or more persistent Python processes.

To use it, first install and configure the wfastcgi package as described on pypi.org/project/wfastcgi/ .

Next, modify your app's *web.config* file to include the full paths to *python.exe* and *wfastcgi.py* in the `PythonHandler` key. The steps below assume that Python is installed in *c:\python36-32* and that your app code is in *c:\home\site\wwwroot*; adjust for your paths accordingly:

1. Modify the `PythonHandler` entry in *web.config* so that the path matches the Python install location (see IIS Configuration Reference (iis.net) for exact details).

XML Copy

```xml
<system.webServer>
  <handlers>
    <add name="PythonHandler" path="*" verb="*"
modules="FastCgiModule"
        scriptProcessor="c:\python36-
32\python.exe|c:\python36-32\wfastcgi.py"
        resourceType="Unspecified"
requireAccess="Script"/>
  </handlers>
</system.webServer>
```

2. Within the `<appSettings>` section of *web.config*, add keys for `WSGI_HANDLER`, `WSGI_LOG` (optional), and `PYTHONPATH`:

XML Copy

```xml
<appSettings>
  <add key="PYTHONPATH" value="c:\home\site\wwwroot"/>
  <!-- The handler here is specific to Bottle; see the
next section. -->
  <add key="WSGI_HANDLER" value="app.wsgi_app()"/>
  <add key="WSGI_LOG" value="c:\home\LogFiles\wfastc-
gi.log"/>
</appSettings>
```

These `<appSettings>` values are available to your app as environment variables:

- The value for `PYTHONPATH` may be freely extended but must include the root of your app.
- `WSGI_HANDLER` must point to a WSGI app importable from your app.
- `WSGI_LOG` is optional but recommended for debugging your app.

3. Set the `WSGI_HANDLER` entry in *web.config* as appropriate for the framework you're using:

- **Bottle**: make sure that you have parentheses after `app.wsgi_app` as shown below. This is necessary because that object is a function (see *app.py*) rather than a variable:

XML Copy

```xml
<!-- Bottle apps only -->
<add key="WSGI_HANDLER" value="app.wsgi_app()"/>
```

- **Flask**: Change the `WSGI_HANDLER` value to `<project_name>.app` where `<project_name>` matches the name of your project. You can find the exact identifier by looking at the `from <project_name> import app` statement in the *runserver.py*. For example, if the project is named "FlaskAzurePublishExample", the entry would appear as follows:

XML Copy

```xml
<!-- Flask apps only: change the project name to
match your app -->
<add key="WSGI_HANDLER"
value="flask_iis_example.app"/>
```

- **Django**: Two changes are needed to *web.config* for Django projects. First, change the `WSGI_HANDLER` value to `django.core.wsgi.get_wsgi_application()` (the object is in the *wsgi.py* file):

  XML Copy

  ```xml
  <!-- Django apps only -->
  <add key="WSGI_HANDLER" value="django.core.ws-
  gi.get_wsgi_application()"/>
  ```

  Second, add the following entry below the one for `WSGI_HANDLER`, replacing `DjangoAzurePublishExample` with the name of your project:

  XML Copy

  ```xml
  <add key="DJANGO_SETTINGS_MODULE" value="django_i-
  is_example.settings" />
  ```

4. **Django apps only**: In the Django project's *settings.py* file, add your site URL domain or IP address to `ALLOWED_HOSTS` as shown below, replacing '1.2.3.4' with your URL or IP address, of course:

   Python Copy

   ```python
   # Change the URL or IP address to your specific site
   ALLOWED_HOSTS = ['1.2.3.4']
   ```

   Failure to add your URL to the array results in the error **DisallowedHost at / Invalid HTTP_HOST header: '<site URL>'. You may need to add '<site URL>' to ALLOWED_HOSTS.**

Note that when the array is empty, Django automatically allows 'localhost' and '127.0.0.1', but adding your production URL removes those capabilities. For this reason you might want to maintain separate development and production copies of *settings.py*, or use environment variables to control the run time values.

Deploy to IIS or a Windows VM

With the correct *web.config* file in your project, you can publish to the computer running IIS by using the **Publish** command on the project's context menu in **Solution Explorer**, and selecting the option, **IIS, FTP, etc.**. In this case, Visual Studio simply copies the project files to the server; you're responsible for all server-side configuration.

Is this page helpful?

Yes No

Recommended content

FastCGI Application &lt;application&gt;

Overview The FastCGI &lt;application&gt; element contains the configurations settings for a specific FastCGI process pool definition. When FastCGI is used, I...

FastCGI &lt;fastCgi&gt;

Overview The &lt;fastCgi&gt; element contains a collection of &lt;application&gt; elements, each of which creates a FastCGI application pool definition. Inte...

Install and Configure PHP

Introduction The fastest and easiest way to install PHP on Internet Information Services (IIS) is by using the Microsoft ® Web Platform Installer (Web PI). W...

Using FastCGI to Host PHP Applications on IIS 7

This article describes how to configure the FastCGI module and PHP to host PHP applications on IIS 7 and above. IMPORTANT : This article provides instruction...

Show more