

CGI

If all other deployment methods do not work, CGI will work for sure. CGI is supported by all major servers but usually has a sub-optimal performance.

This is also the way you can use a Flask application on Google's [App Engine](#), where execution happens in a CGI-like environment.

Watch Out:

Please make sure in advance that any `app.run()` calls you might have in your application file are inside an `if __name__ == '__main__':` block or moved to a separate file. Just make sure it's not called because this will always start a local WSGI server which we do not want if we deploy that application to CGI / app engine.

With CGI, you will also have to make sure that your code does not contain any `print` statements, or that `sys.stdout` is overridden by something that doesn't write into the HTTP response.

Creating a `.cgi` file

First you need to create the CGI application file. Let's call it `yourapplication.cgi`:

```
#!/usr/bin/python
from wsgiref.handlers import CGIHandler
from yourapplication import app

CGIHandler().run(app)
```

Server Setup

Usually there are two ways to configure the server. Either just copy the `.cgi` into a `cgi-bin` (and use `mod_rewrite` or something similar to rewrite the URL) or let the server point to the file directly.

In Apache for example you can put something like this into the config:

```
ScriptAlias /app /path/to/the/application.cgi
```

 v: 1.1.x ▼

On shared webhosting, though, you might not have access to your Apache config. In this case, a file called `.htaccess`, sitting in the public directory you want your app to be available, works too but the `ScriptAlias` directive won't work in that case:

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f # Don't interfere with static files
RewriteRule ^(.*)$ /path/to/the/application.cgi/$1 [L]
```

For more information consult the documentation of your webserver.