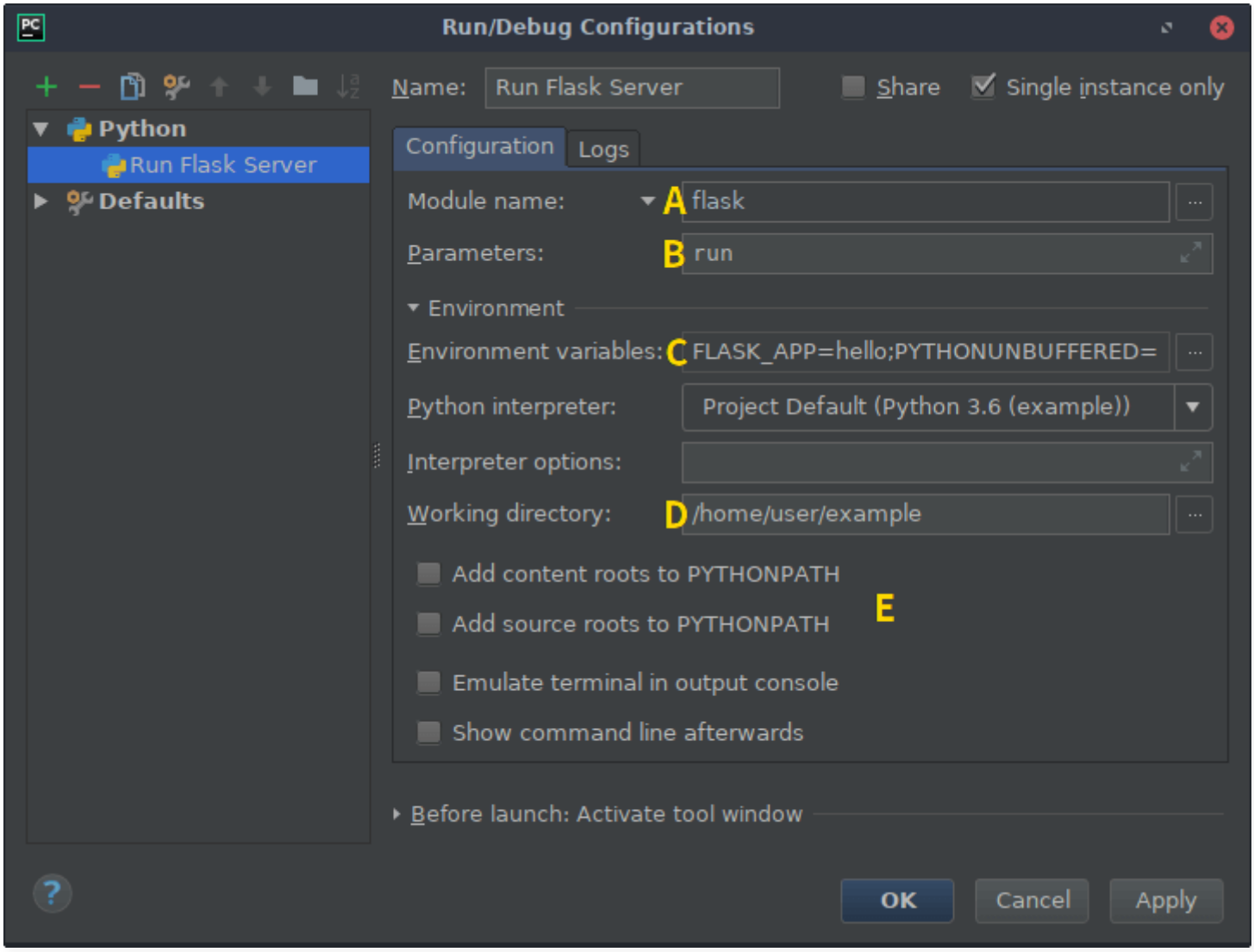
PyCharm Integration

Prior to PyCharm 2018.1, the Flask CLI features weren't yet fully integrated into PyCharm. We have to do a few tweaks to get them working smoothly. These instructions should be similar for any other IDE you might want to use.

In PyCharm, with your project open, click on *Run* from the menu bar and go to *Edit Configurations*. You'll be greeted by a screen similar to this:



There's quite a few options to change, but once we've done it for one command, we can easily copy the entire configuration and make a single tweak to give us access to other commands, including any custom ones you may implement yourself.

Click the + (*Add New Configuration*) button and select *Python*. Give the configuration a good descriptive name such as "Run Flask Server". For the `flask run` command, check "Single instance only" since you can't run the server more than once at the same time.

Select *Module name* from the dropdown (**A**) then input `flask`.

The *Parameters* field (**B**) is set to the CLI command to execute (with any arguments). In this example we use `run`, which will run the development server.

You can skip this next step if you're using Environment Variables From dotenv. We need to add an environment variable (**C**) to identify our application. Click on the browse button and add an entry with `FLASK_APP` on the left and the Python import or file on the right (`hello` for example).

Next we need to set the working directory (**D**) to be the folder where our application resides.

If you have installed your project as a package in your virtualenv, you may untick the *PYTHONPATH* options (**E**). This will more accurately match how you deploy the app later.

Click *Apply* to save the configuration, or *OK* to save and close the window. Select the configuration in the main PyCharm window and click the play button next to it to run the server.

Now that we have a configuration which runs `flask run` from within PyCharm, we can copy that configuration and alter the *Script* argument to run a different CLI command, e.g. `flask shell`.