Quickstart: Create a Python app in Azure App Service on Linux

04/03/2020 • 6 minutes to read • +12

In this quickstart, you deploy a Python web app to App Service on Linux, Azure's highly scalable, self-patching web hosting service. You use the local Azure command-line interface (CLI) on a Mac, Linux, or Windows computer. The web app you configure uses a free App Service tier, so you incur no costs in the course of this article.

If you prefer to deploy apps through an IDE, see Deploy Python apps to App Service from Visual Studio Code.

Prerequisites

- Azure subscription - create one for free
- Python 3.7 (Python 3.6 is also supported)
- Git
- Azure CLI 2.0.80 or higher. Run `az --version` to check your version.

Download the sample

In a terminal window, run the following command to clone the sample application to your local computer.

```terminal
git clone https://github.com/Azure-Samples/python-docs-hello-world
```

Then go into that folder:

```terminal
cd python-docs-hello-world
```

The repository contains an *application.py* file, which tells App Service that the code contains a Flask app. For more information, see Container startup process and customizations.

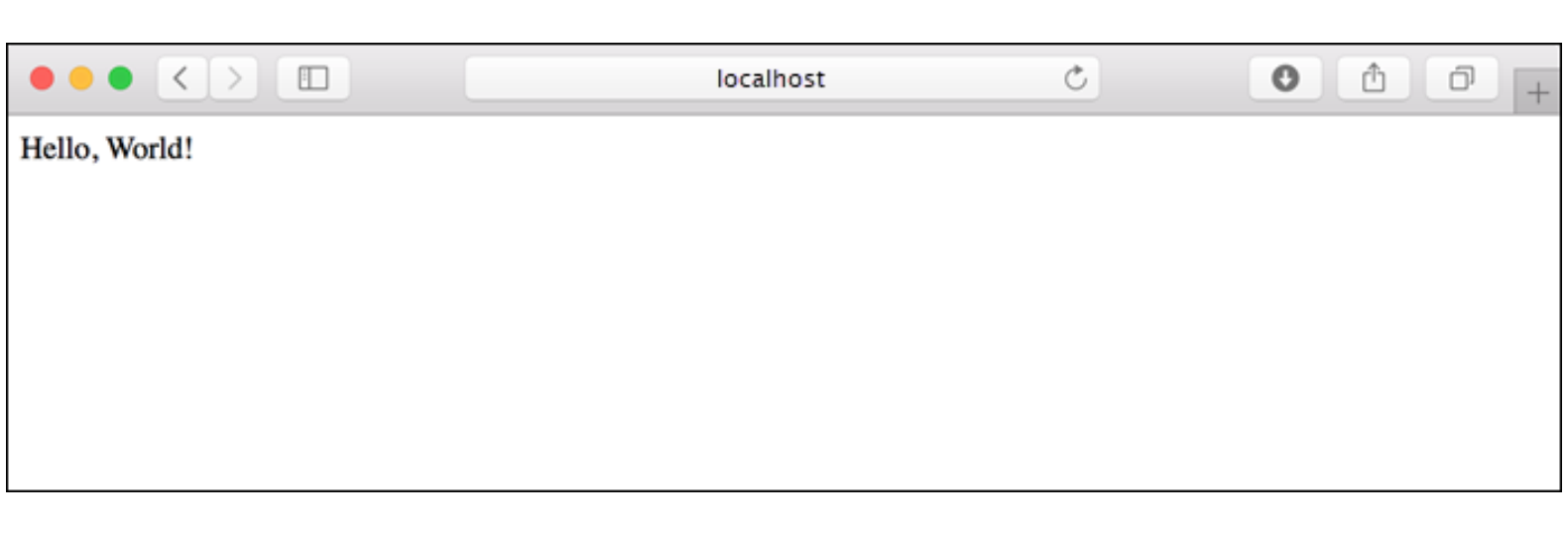
Run the sample

In a terminal window, use the commands below (as appropriate for your operating system) to install the required dependencies and launch the built-in development server.

Bash | PowerShell | Cmd

```bash
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
export FLASK_APP=application.py
flask run
```

Open a web browser, and go to the sample app at `http://localhost:5000/`. The app displays the message **Hello World!**.



In your terminal window, press **Ctrl**+**C** to exit the web server.

Sign in to Azure

The Azure CLI provides you with many convenient commands that you use from a local terminal to provision and manage Azure resources from the command line. You can use commands to complete the same tasks that you would through the Azure portal in a browser. You can also use CLI commands in scripts to automate management processes.

To run Azure commands in the Azure CLI, you must first sign in using the `az login` command. This command opens a browser to gather your credentials.

```Azure CLI
az login
```

Deploy the sample

The az webapp up command creates the web app on App Service and deploys your code.

In the *python-docs-hello-world* folder that contains the sample code, run the following `az webapp up` command. Replace `<app-name>` with a globally unique app name (*valid characters are* `a-z`, `0-9`, *and* `-`).

```Azure CLI
az webapp up --sku F1 -n <app-name>
```

> **Caution**
>
> If you are using **Azure-CLI version 2.5.0** there is a regression in `az webapp up` where certain scenarios will fail if the `-l <location-name>` parameter is not included. This issue being tracked here.
>
> You can check what version of the Azure-CLI you are using with the `az --version` command.

The `--sku F1` argument creates the web app on the Free pricing tier. You can omit this argument to use a premium tier instead, which incurs an hourly cost.

You can optionally include the argument `-l <location-name>` where `<location_name>` is an Azure region such as **centralus**, **eastasia**, **westeurope**, **koreasouth**, **brazilsouth**, **centralindia**, and so on. You can retrieve a list of allowable regions for your Azure account by running the az account list-locations command.

The `az webapp up` command may take a few minutes to completely run. While running, it displays information similar to the following example, where `<app-name>` will be the name you provided earlier:

```
Creating Resource group 'appsvc_rg_Linux_centralus' ...
Resource group creation complete
Creating App service plan 'appsvc_asp_Linux_centralus' ...
App service plan creation complete
Creating app '<app-name>' ....
Configuring default logging for the app, if not already enabled
Creating zip with contents of dir D:\Examples\python-docs-hello-worl
Getting scm site credentials for zip deployment
Starting zip deployment. This operation can take a while to complete
Deployment endpoint responded with status code 202
You can launch the app at http://<app-name>.azurewebsites.net
{
  "URL": "http://<app-name>.net",
  "appserviceplan": "appsvc_asp_Linux_centralus",
  "location": "eastus",
  "name": "<app-name>",
  "os": "Linux",
  "resourcegroup": "appsvc_rg_Linux_centralus",
  "runtime_version": "python|3.7",
  "runtime_version_detected": "-",
  "sku": "FREE",
  "src_path": "D:\\Examples\\python-docs-hello-world"
}
```

> **Note**
>
> The `az webapp up` command does the following actions:
>
> - Create a default resource group.
> - Create a default app service plan.
> - Create an app with the specified name.
> - Zip deploy files from the current working directory to the app.

Browse to the app

Browse to the deployed application in your web browser at the URL `http://<app-name>.azurewebsites.net`.

The Python sample code is running a Linux container in App Service using a built-in image.

Congratulations! You've deployed your Python app to App Service on Linux.

Redeploy updates

In your favorite code editor, open *application.py* and update the `hello` function as follows. This change adds a `print` statement to generate logging output that you work with in the next section.

```python
def hello():
    print("Handling request to home page.")
    return "Hello Azure!"
```

Save your changes and exit the editor.

Redeploy the app using the `az webapp up` command again:

```Azure CLI
az webapp up
```

This command uses values that are cached in the *.azure/config* file, including the app name, resource group, and App Service plan.

Once deployment has completed, switch back to the browser window open to `http://<app-name>.azurewebsites.net` and refresh the page, which should display the modified message:

> **Tip**
>
> Visual Studio Code provides powerful extensions for Python and Azure App Service, which simplify the process of deploying Python web apps to App Service. For more information, see Deploy Python apps to App Service from Visual Studio Code.

Stream logs

You can access the console logs generated from inside the app and the container in which it runs. Logs include any output generated using `print` statements.

To stream logs, run the following command:

```Azure CLI
az webapp log tail
```

Refresh the app in the browser to generate console logs, which should include lines similar to the following text. If you don't see output immediately, try again in 30 seconds.

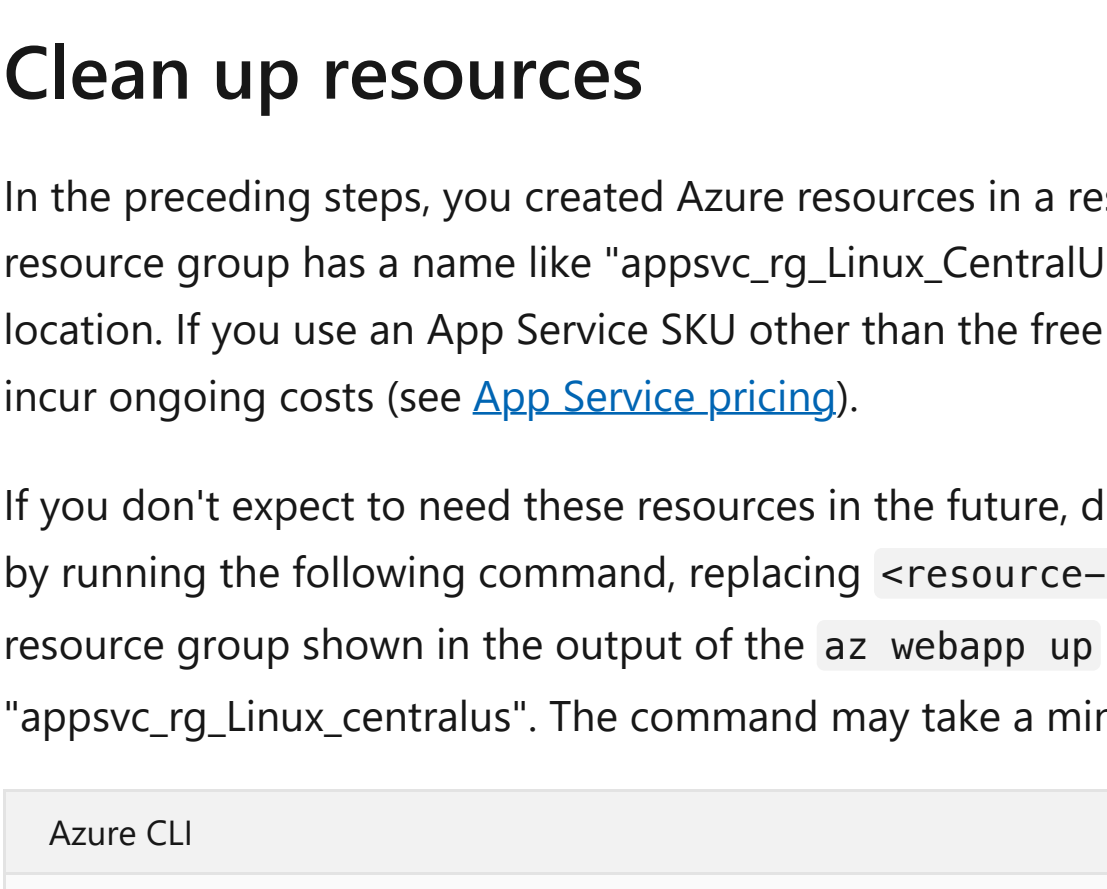
```
2020-04-03T22:54:04.236405938Z Handling request to home page.
2020-04-03T22:54:04.236497641Z 172.16.0.1 - - [03/Apr/2020:22:54:04
```

You can also inspect the log files from the browser at `https://<app-name>.scm.azurewebsites.net/api/logs/docker`.

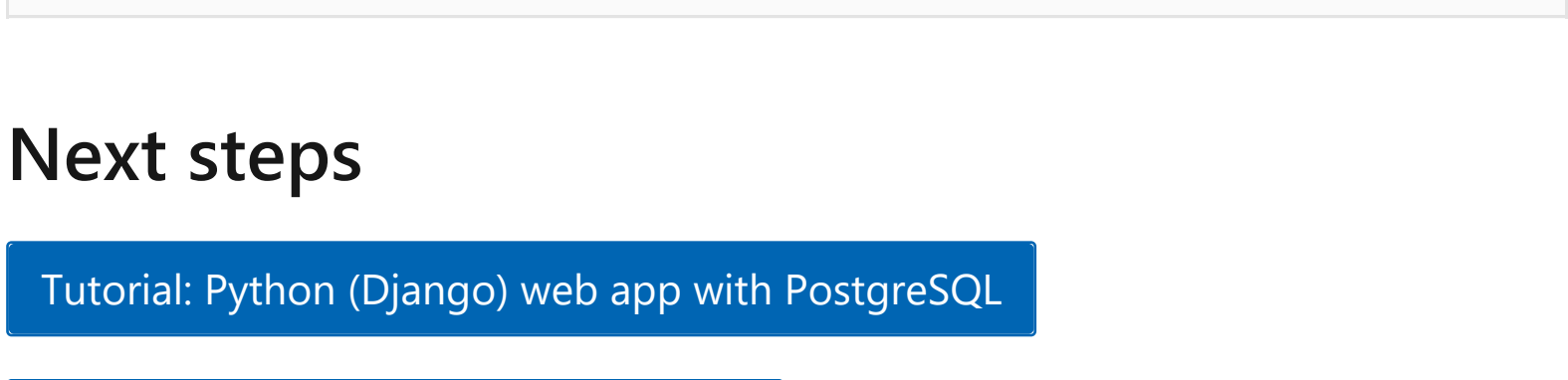
To stop log streaming at any time, type `Ctrl`+`C`.

Manage the Azure app

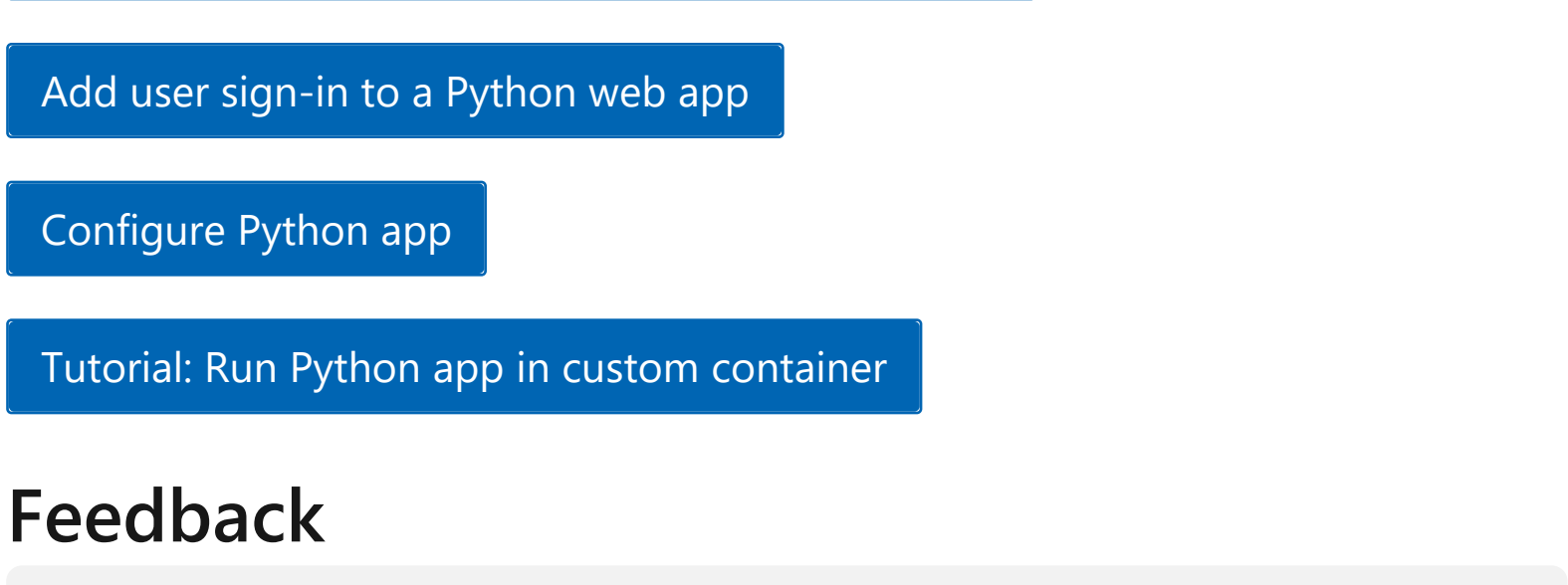
Go to the Azure portal to manage the app you created. Search for and select **App Services**.



Select the name of your Azure app.



You see your app's Overview page. Here, you can perform basic management tasks like browse, stop, start, restart, and delete.



The App Service menu provides different pages for configuring your app.

Clean up resources

In the preceding steps, you created Azure resources in a resource group. The resource group has a name like "appsvc_rg_Linux_CentralUS" depending on your location. If you use an App Service SKU other than the free F1 tier, these resources incur ongoing costs (see App Service pricing).

If you don't expect to need these resources in the future, delete the resource group by running the following command, replacing `<resource-group-name>` with the resource group shown in the output of the `az webapp up` command, such as "appsvc_rg_Linux_centralus". The command may take a minute to complete.

```Azure CLI
az group delete -n <resource-group-name>
```

Next steps

Tutorial: Python (Django) web app with PostgreSQL

Add user sign-in to a Python web app

Configure Python app

Tutorial: Run Python app in custom container

Feedback

Submit and view feedback for

This product | This page

View all page feedback