

Deploy your Flask app on Azure in 3 easy steps



Niko Vrdoljak

[Follow](#)

Feb 6, 2019 · 5 min read

In this article, I will show you how to deploy and publish your Flask web app on Azure. We will use an Azure App Service on Linux, which provides a highly scalable web hosting service using the Linux operating system.

Step 1 — prepare Flask app

I will assume that you are familiar with Flask framework, but in any case, I will create minimal Flask application for demonstration purposes and emphasize some aspects of app environment needed for successful publishing on Azure (or any other hosting service).

First in “hello.py” file create “Hello world” app:

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route("/")
5  def index():
6      return "<h1>Hello Azure!</h1>"
```

hello.py hosted with ❤ by GitHub

[view raw](#)

Next, create and activate virtual environment, set startup file, initialize local git and start the app from your terminal:

```
1  PS C:\azure-flask> python -m venv venv
2  PS C:\azure-flask> .\venv\Scripts\Activate.ps1
3  (venv) PS C:\azure-flask> pip install flask
4  (venv) PS C:\azure-flask> $env:FLASK_APP = "hello.py"
5  (venv) PS C:\azure-flask> git init
6  Initialized empty Git repository in C:/azure-flask/.git/
7  (venv) PS C:\azure-flask> flask run
8  * Serving Flask app "hello.py"
9  * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

flask-cmd-init hosted with ❤ by GitHub

[view raw](#)

Navigate to `http://127.0.0.1:5000/` to check that your app is running.

We will use git later to push our app to Azure. Also, create a `.gitignore` file to specify files and folders you don't want Git to check in:

```
1  venv/  
2  __pycache__/  
3  .vscode/
```

hello-gitignore hosted with ❤ by GitHub

[view raw](#)

Step 2 — configure Azure service

(if you don't have an Azure subscription, create a free account before you begin)

We will configure service via Azure portal. There is also an alternative to do this via Azure Cloud Shell, but we will skip it this time. So, log in to Azure portal. We will configure several elements to complete this step:

Create a resource group

A **resource group** is a logical container into which Azure resources like web apps, databases, and storage accounts are deployed and managed.

On Azure portal left navigation bar click “Resource groups” and then “Add”. In displayed form, select your Azure subscription, location for your resources and type the name for your group.

PROJECT DETAILS

* Subscription ⓘ

Visual Studio Enterprise – MPN

* Resource group ⓘ

hello-flask-resource-group ✓

RESOURCE DETAILS

* Region ⓘ

West Europe

Click “Create and Review” button and wait for notification:

**Resource group created**

Creating resource group 'hello-flask-resource-group' in subscription 'Visual

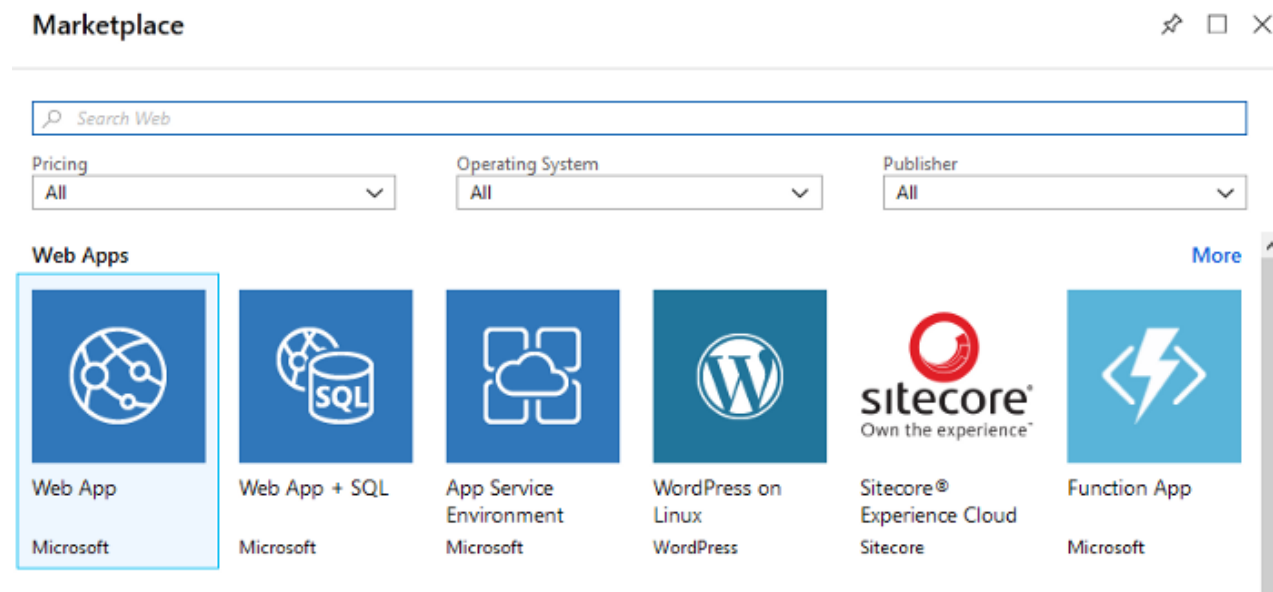
Studio Enterprise – MPN' succeeded.

[Go to resource group](#)[Pin to dashboard](#)

a few seconds ago

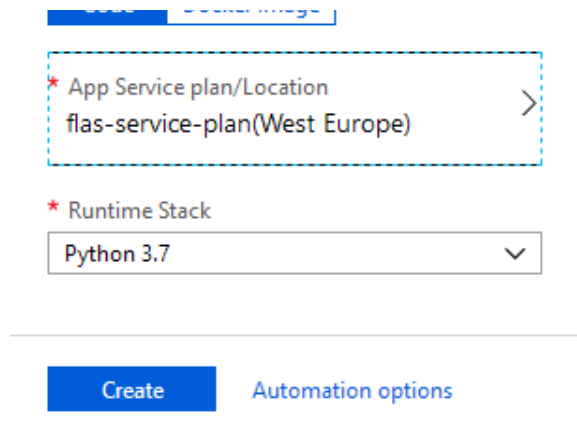
Create a web app

Now we can create web app or **App Service**. On portal left navigation bar click “App Services” and then “Add”. Select “Web App” and click “Create”:



In Create Web App form, enter app name, select your subscription and existing resource group, Linux as OS, Code as publishing mode and Python 3.x as runtime stack:

The image shows the 'Web App' creation form in the Azure portal. The form is titled 'Web App' and has a 'Create' button. It contains several fields: 'App name' with the value 'flask-hello' and a green checkmark; 'Subscription' with the value 'Visual Studio Enterprise – MPN'; 'Resource Group' with the value 'hello-flask-resource-group' and a dropdown arrow; 'OS' with 'Linux' selected; and 'Publish' with 'Code' selected. There are also radio buttons for 'Create new' and 'Use existing' resource groups.

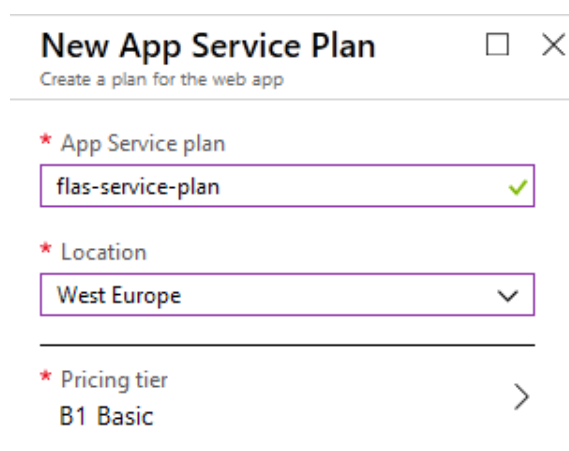


* App Service plan/Location
flas-service-plan(West Europe) >

* Runtime Stack
Python 3.7 v

Create Automation options

Also, you have to create an App Service Plan for your app. An **App Service Plan** defines a set of compute resources for a web app to run. These compute resources are analogous to the server farm in conventional web hosting. One or more apps can be configured to run on the same computing resources (or in the same App Service plan). So click the App Service plan button and then Create new and fill the form:



New App Service Plan □ ×
Create a plan for the web app

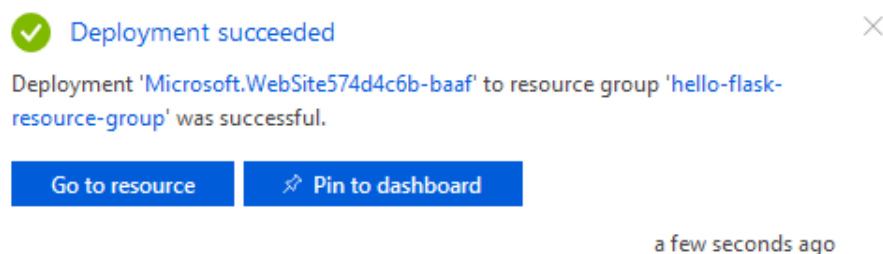
* App Service plan
flas-service-plan ✓

* Location
West Europe v

* Pricing tier
B1 Basic >

Select appropriate pricing tier, but for demo purposes, Basic tier is the most suitable option.

Click “Create app” and wait for notification message:



✓ Deployment succeeded ×

Deployment 'Microsoft.WebSite574d4c6b-baaf' to resource group 'hello-flask-resource-group' was successful.

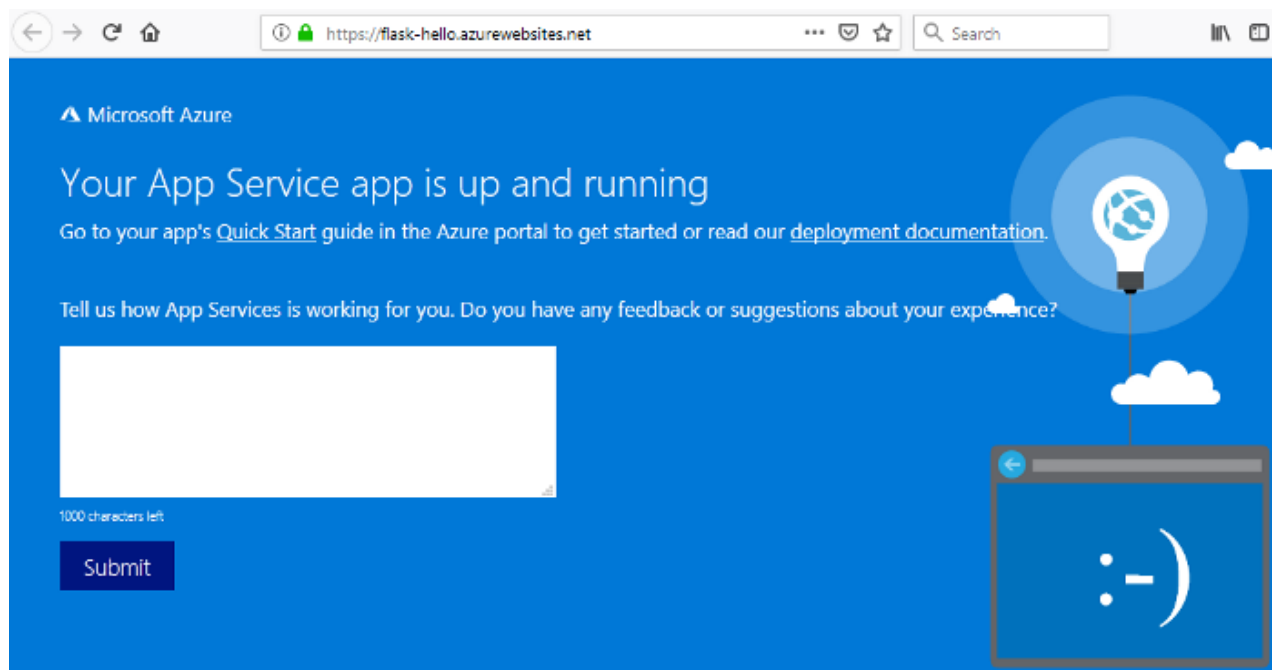
Go to resource Pin to dashboard

a few seconds ago

Go to app service resource and examine your app info:

Resource group (change)	: hello-flask-resource-group	URL	: https://flask-hello.azurewebsites.net
Status	: Running	App Service Plan	: flask-service-plan (Basic: 1 Small)
Location	: West Europe	FTP/deployment userna...	: flask-hello\nikov_flask_app

Notice URL for your app. If you click on it, it will open default page for your app which at this moment looks like:



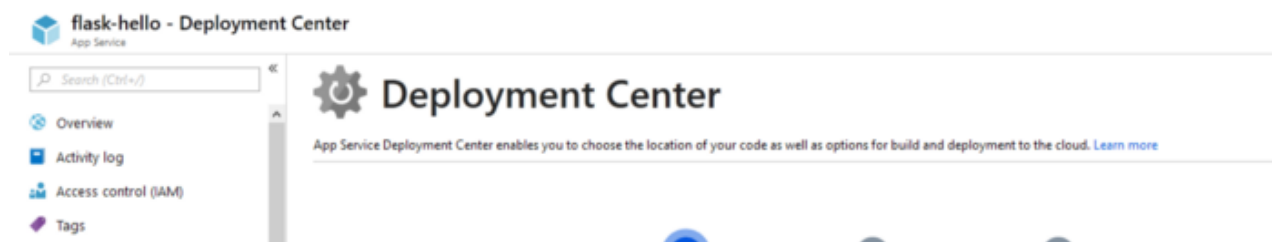
Remember that URL of your app is:

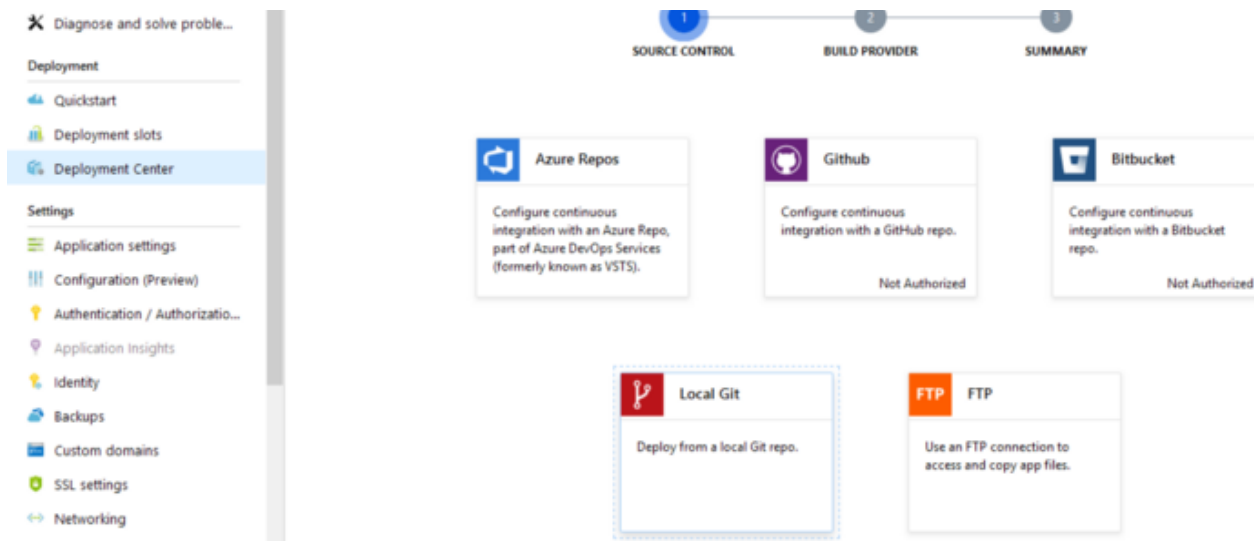
http://<app_name>.azurewebsites.net

Step 3 — Deployment

Configure deployment options

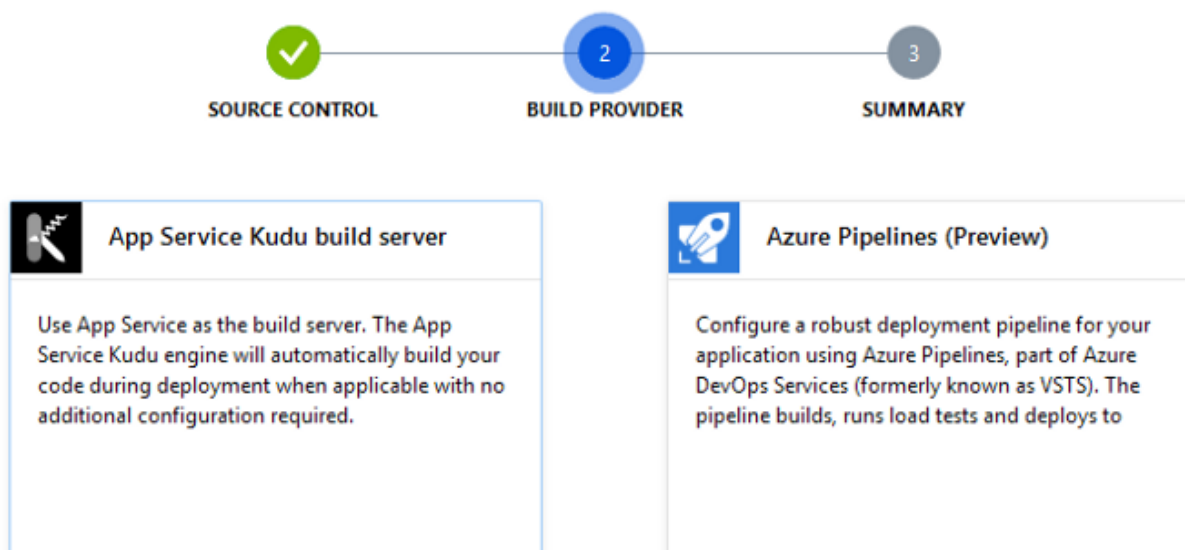
Our goal is to deploy our app from local git to Azure App Service that we created. To enable that, we first have to configure some deployment settings. Click “Deployment center”, and select “Local Git”:





Azure allows a lot of options for build and deployment

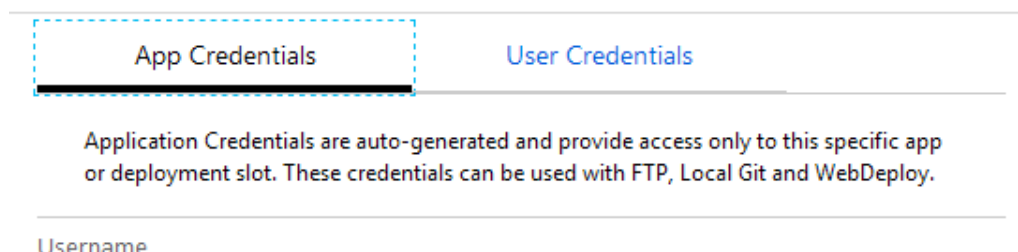
On the next step, select “Kudu” as the build server:

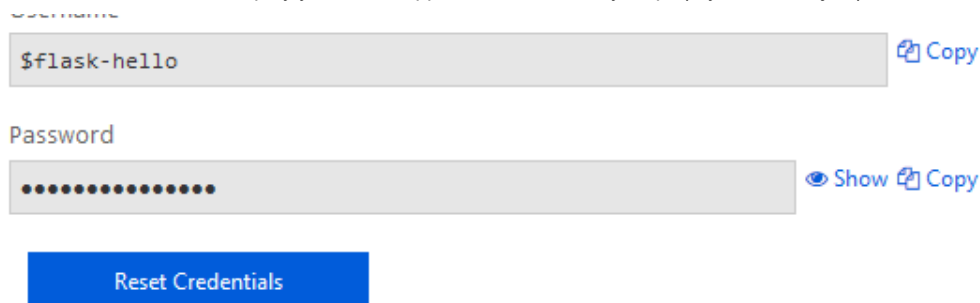


Click “Finish”, wait for notification and you will get *Git Clone Uri* like this:

```
https://flask-hello.scm.azurewebsites.net:443/flask-hello.git
```

Also, click “Deployment Credentials” to see your app credentials:





Username

\$flask-hello Copy

Password

..... Show Copy

Reset Credentials

Here you can see your app username and password, which you can change or create another user credentials for your deployment purposes.

Now we are ready for our deployment. Go to your local terminal and add Azure remote to your local Git repository. Replace `<deploymentLocalGitUrl-from-create-step>` with the URL of the Git remote that you get in the previous step:

```
git remote add azure-hello https://flask-hello.scm.azurewebsites.net:443/flask-hello.git
```

Now commit any changes to local git:

```
git commit -a -m "first commit"
```

And push to the "azure-hello" remote to deploy your app with the following command:

```
git push azure-hello master
```

When prompted for credentials by Git Credential Manager, make sure that you enter the credentials you created in Configure a deployment user, not the credentials you use to sign in to the Azure portal.

This command may take a few minutes to run. While running, it displays something like:

```
1 PS C:\azure-flask> git push azure-hello master
2 Counting objects: 7, done.
3 Delta compression using up to 4 threads.
4 Compressing objects: 100% (6/6), done.
5 Writing objects: 100% (7/7), 653 bytes | 93.00 KiB/s, done.
6 Total 7 (delta 1), reused 0 (delta 0)
7 remote: Updating branch 'master'.
8 remote: Updating submodules.
9 remote: Preparing deployment for commit id '2e33addc9e'.
10 remote: Generating deployment script.
11 remote: Generating deployment script for python Web Site
12 remote: Generated deployment script files
```

```

13 remote: Running deployment command...
14 remote: Python deployment.
15 remote: Kudu sync from: '/home/site/repository' to: '/home/site/wwwroot'
16 remote: Copying file: '.gitignore'
17 remote: Copying file: 'hello.py'
18 remote: Deleting file: 'hostingstart.html'
19 remote: Ignoring: .git
20 remote: /home/site/repository
21 remote: /home/site/wwwroot
22 remote: Finished successfully.
23 remote: Running post deployment command(s)...
24 remote: Deployment successful.
25 remote: App container will begin restart within 10 seconds.
26 To https://flask-hello.scm.azurewebsites.net:443/flask-hello.git
27 * [new branch]      master -> master
28 PS C:\azure-flask>

```

git push azure-hello master hosted with ❤ by GitHub

[view raw](#)

You can see similar log information in the Deployment Center:

[Refresh](#)
[Disconnect](#)
[Deployment Credentials](#)

Source
Local Git

Build
Kudu

Git Clone Uri
https://flask-hello.scm.azurewebsites.net:443/flask-hello.git

TIME	STATUS	COMMIT ID (AUTHOR)
Wednesday, February 6, 2019		
10:14:06 AM GMT+1	Success (Active)	2e33add (Niko Vrdoljak)

[Redeploy](#)

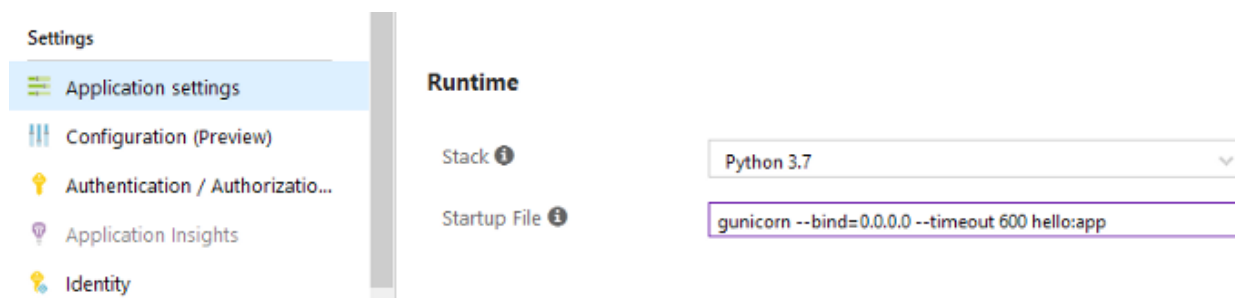
TIME	ACTIVITY	LOG
10:14:07 AM	Updating branch 'master'.	
10:14:09 AM	Updating submodules.	
10:14:09 AM	Preparing deployment for commit id '2e33addc9e'.	
10:14:12 AM	Generating deployment script.	Show Logs...
10:14:17 AM	Running deployment command...	Show Logs...
10:14:20 AM	Running post deployment command(s)...	
10:14:21 AM	Deployment successful.	
10:14:21 AM	App container will begin restart within 10 seconds.	

Command: "/home/site/deployments/tools/deploy.sh"
Python deployment.
Kudu sync from: '/home/site/repository' to: '/home/site/wwwroot'
Copying file: '.gitignore'
Copying file: 'hello.py'
Deleting file: 'hostingstart.html'
Ignoring: .git
/home/site/repository
/home/site/wwwroot
Finished successfully.

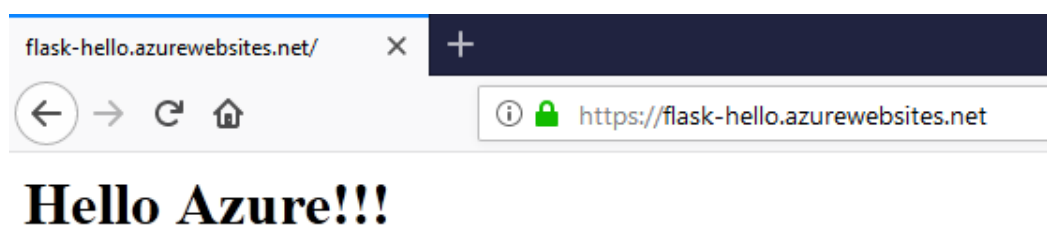
If you refresh your app, you will see that nothing changed yet. The reason is App Service uses Gunicorn WSGI HTTP Server to run an app, which looks for a file named *application.py* or *app.py*. Since our main module is in *hello.py* file, we have to customize startup command. Go to “Application settings” and enter the following line in “Startup File” field:

```
1 gunicorn --bind=0.0.0.0 --timeout 600 hello:app
```


unicorn startup hosted with ❤ by GitHub

[view raw](#)

Click “Save”, go to “Overview”, and click “Restart” to start the app with the new configuration. Now refresh your app. If everything is OK, our sample app should run in App Service on Linux:



Our app is ready!

That’s it. If you want to learn more on how you can customize the behavior of App Service with Python visit:



Configure Python apps - Azure App Service


This tutorial describes options for authoring and configuring Python apps for Azure App Service on Linux.

docs.microsoft.com

Also, if you want to get more information about your web site, its configuration, or access the diagnostic console from Bash or SSH, go to associated SCM service site:

https://<app_name>.scm.azurewebsites.net/

  https://flask-hello.scm.azurewebsites.net/Env.cshtml

 Azure App Service Environment SSH Bash

Index

- [System Info](#)
- [App Settings](#)
- [Connection Strings](#)
- [Environment variables](#)
- [PATH](#)
- [HTTP Headers](#)
- [Server variables](#)

System info

- System up time: 04:51:01.1940000
- OS version: Unix 4.4.0.128
- 64 bit system: True
- 64 bit process: True
- Processor count: 1
- Machine name: 3e3856bd2ba7
- Instance id: babb1fdc78d3e047e08c5c6d45c466a0f35f297737d421bab1adc276c54b99e
- Short instance id: babb1f
- CLR version: 4.0.30319.42000
- System directory:
- Current working directory: /etc/apache2/sites-available
- IIS command line: /usr/lib/mono/4.5/mod-mono-server4.exe --filename /tmp/mod_mono_server_default --applications

AppSettings

- SCM_GIT_USERNAME = windowsazure
- SCM_GIT_EMAIL = windowsazure
- webpages:Version = 3.0.0.0
- webpages.Enabled = true
- webactivator.assembliesToScan = Kudu.Services.Web

Azure Kudu CSM service site (notice "scm" in site address)

Python Flask Azure

About Help Legal

Get the Medium app

