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:

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def index():
    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:

```
PS C:\azure-flask> python -m venv venv
PS C:\azure-flask> .\venv\Scripts\Activate.ps1
(venv) PS C:\azure-flask> pip install flask
(venv) PS C:\azure-flask> $env:FLASK_APP = "hello.py"
(venv) PS C:\azure-flask> git init
Initialized empty Git repository in C:/azure-flask/.git/
(venv) PS C:\azure-flask> flask run
 * Serving Flask app "hello.py"
 * 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:

```
venv/
__pycache__/
.vscode/
```

hello-gitignore hosted with ♡ by GitHub view raw

Step 2 — configure Azure service

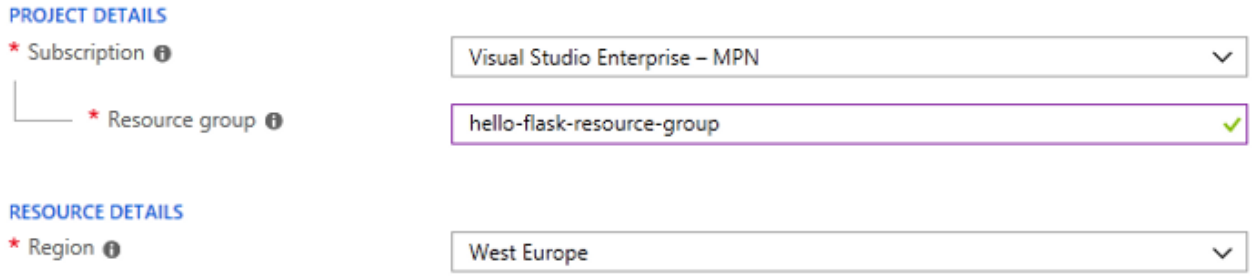
(if you don't have an Azure subscription, create a free account before you begin)

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:

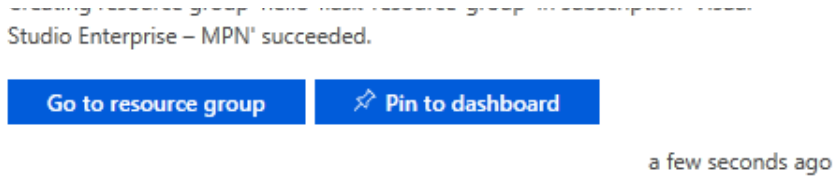
Create a resource group

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.

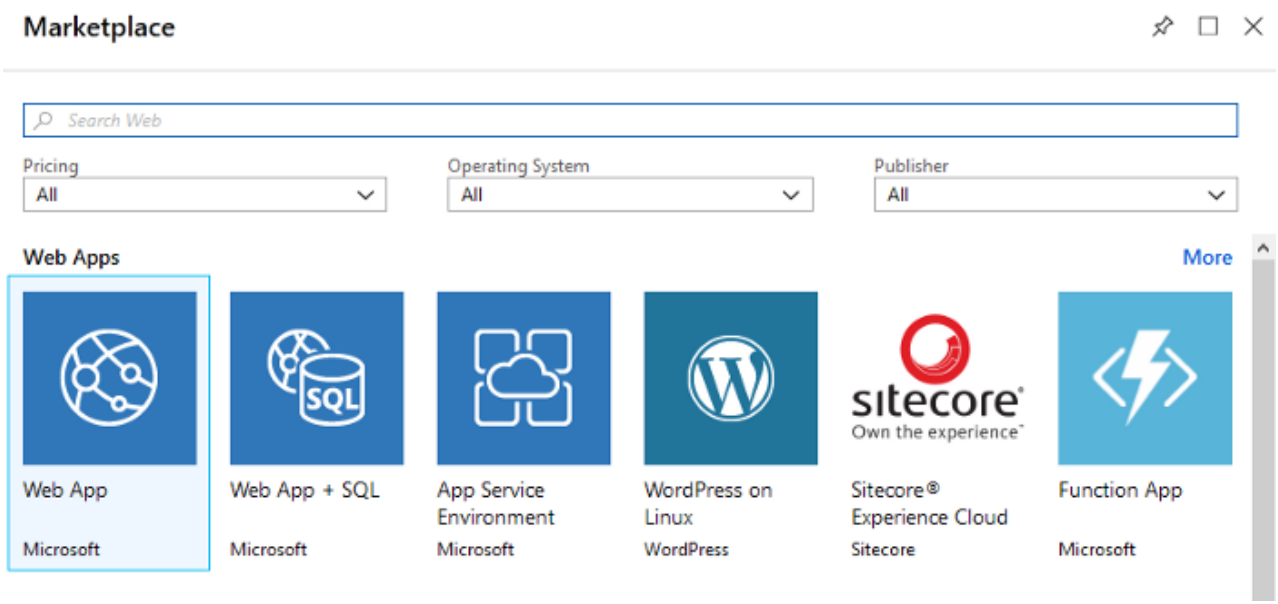


Click "Create and Review" button and wait for notification:

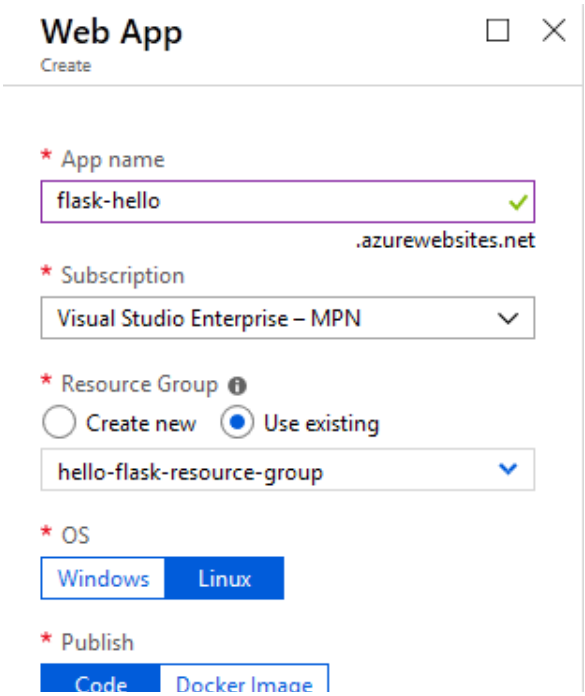


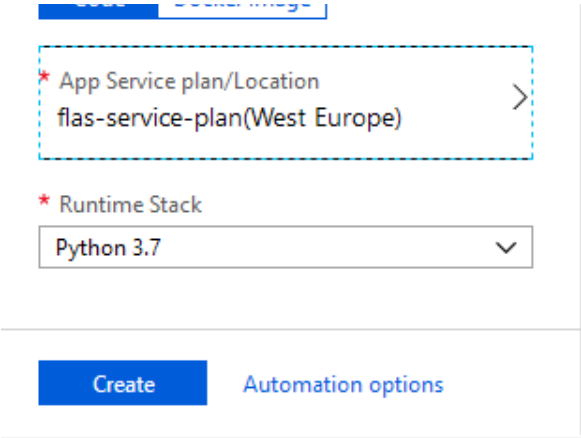
Create a web app

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”:

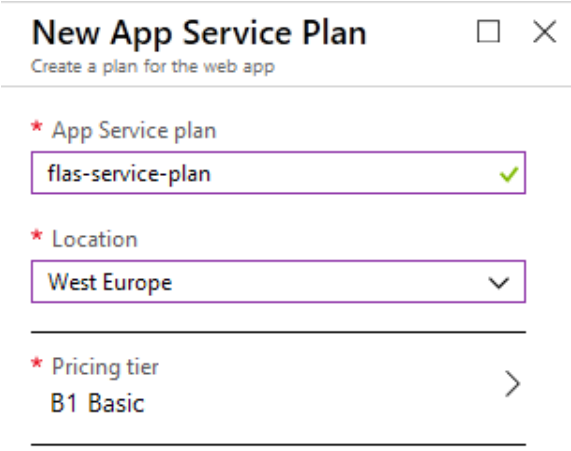


In Create Web App form, enter app name, select your subscription and existing resource group, Linux as OS, Code as publishing mode and Python 3.x as runtime stack:



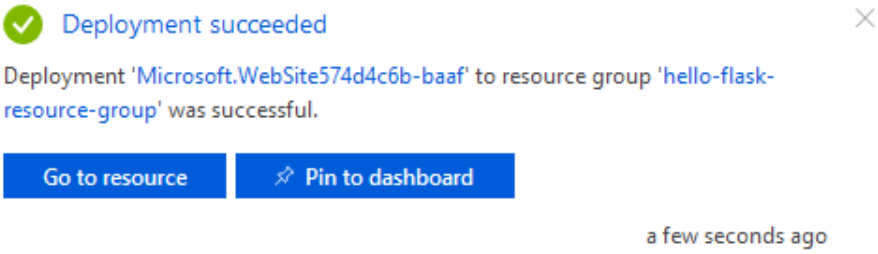


Also, you have to create an App Service Plan for your app. An **App Service Plan** defines a set of compute resources for a web app to run. These compute resources are analogous to the server farm in conventional web hosting. One or more apps can be configured to run on the same computing resources (or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



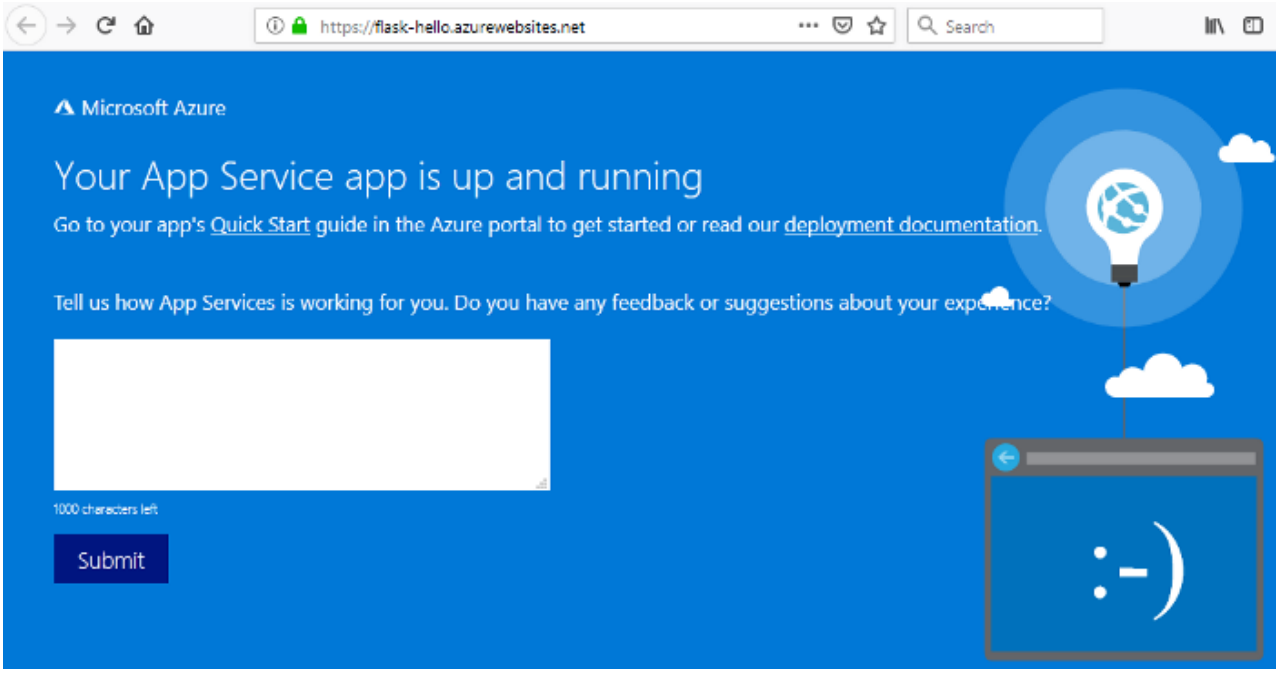
Select appropriate pricing tier, but for demo purposes, Basic tier is the most suitable option.

Click “Create app” and wait for notification message:



Go to app service resource and examine your app info:

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



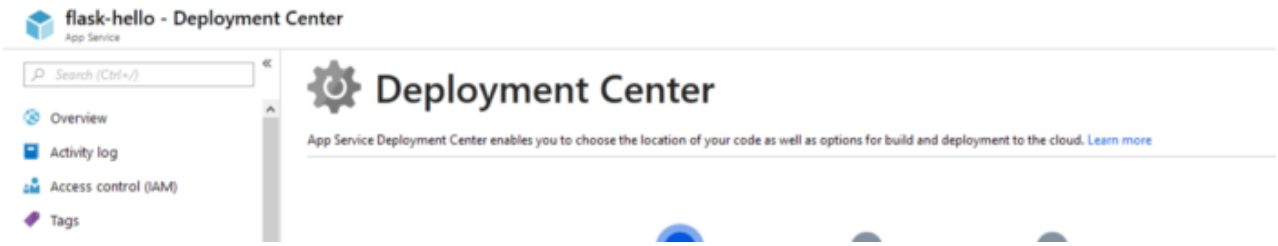
Remember that URL of your app is:

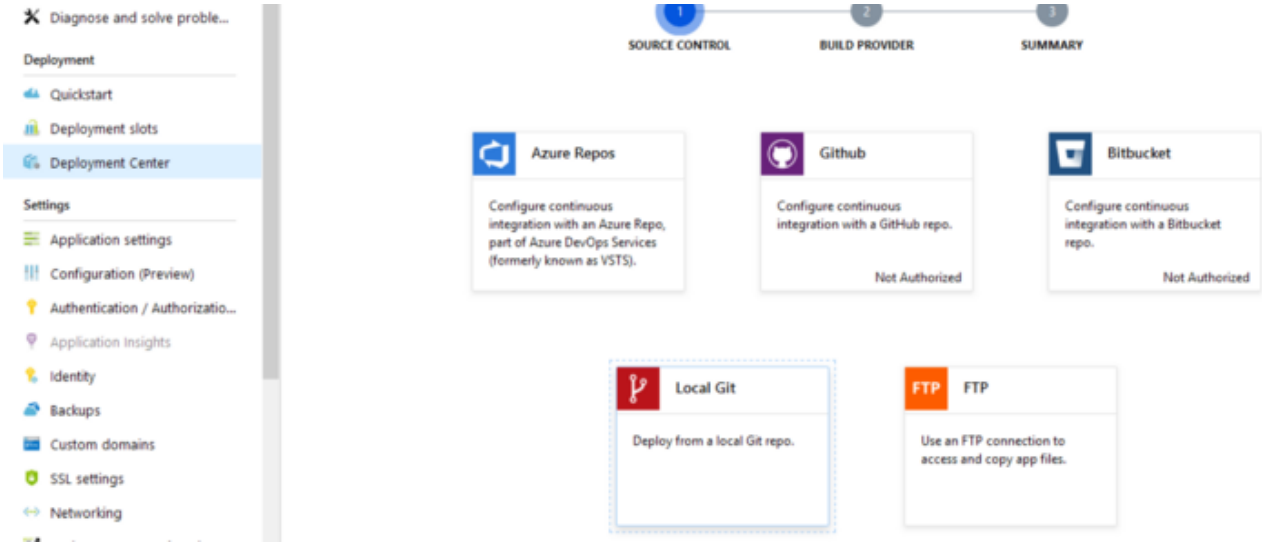
> *http://<app_name>.azurewebsites.net*

Step 3 — Deployment

Configure deployment options

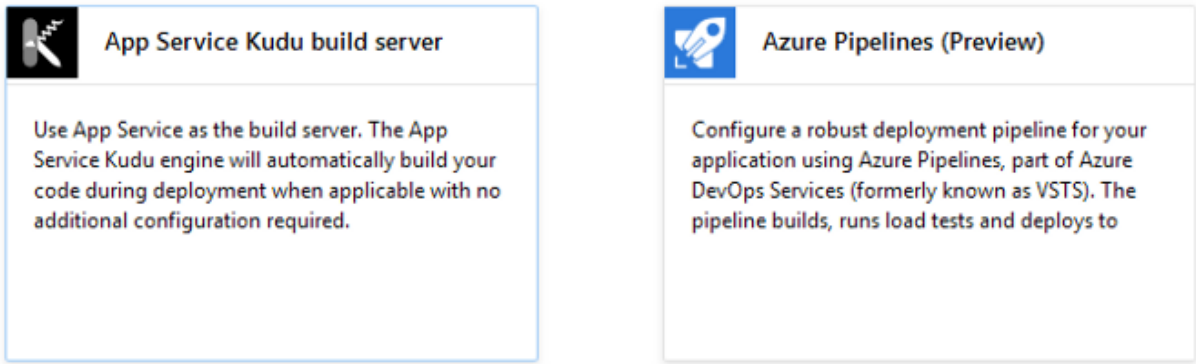
Our goal is to deploy our app from local git to Azure App Service that we created. To enable that, we first have to configure some deployment settings. Click “Deployment center”, and select “Local Git”:





Azure allows a lot of options for build and deployment

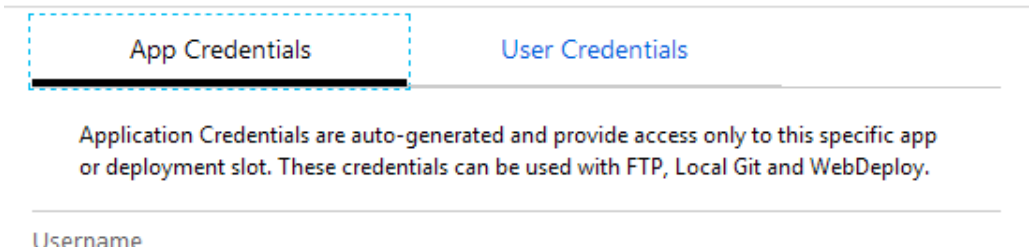
On the next step, select “Kudu” as the build server:

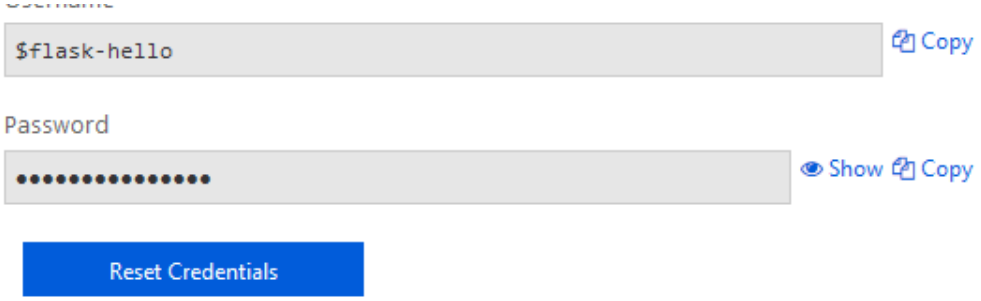


Click “Finish”, wait for notification and you will get *Git Clone Uri* like this:

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

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





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.

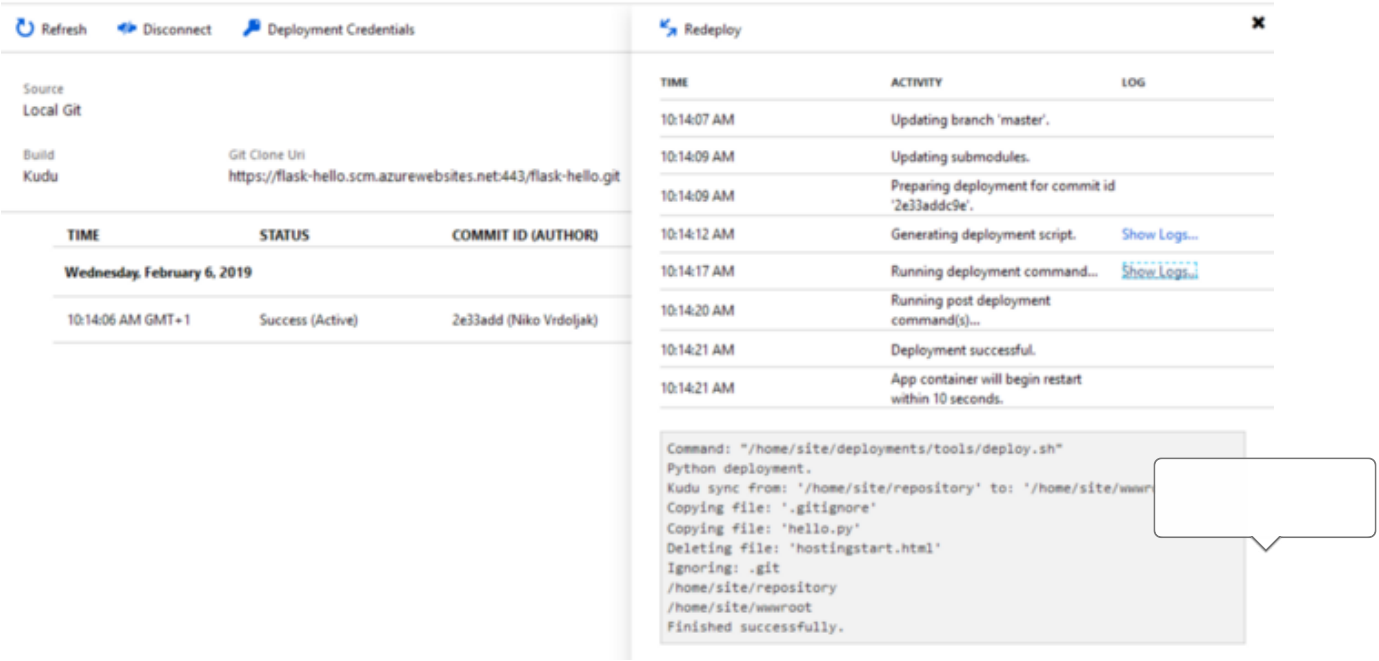
This command may take a few minutes to run. While running, it displays something like:

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

```
remote: Running deployment command...
remote: Python deployment.
remote: Kudu sync from: '/home/site/repository' to: '/home/site/wwwroot'
remote: Copying file: '.gitignore'
remote: Copying file: 'hello.py'
remote: Deleting file: 'hostingstart.html'
remote: Ignoring: .git
remote: /home/site/repository
remote: /home/site/wwwroot
remote: Finished successfully.
remote: Running post deployment command(s)...
remote: Deployment successful.
remote: App container will begin restart within 10 seconds.
To https://flask-hello.scm.azurewebsites.net:443/flask-hello.git
 * [new branch]      master -> master
PS C:\azure-flask>
```

git push azure-hello master hosted with ♡ by GitHub view raw

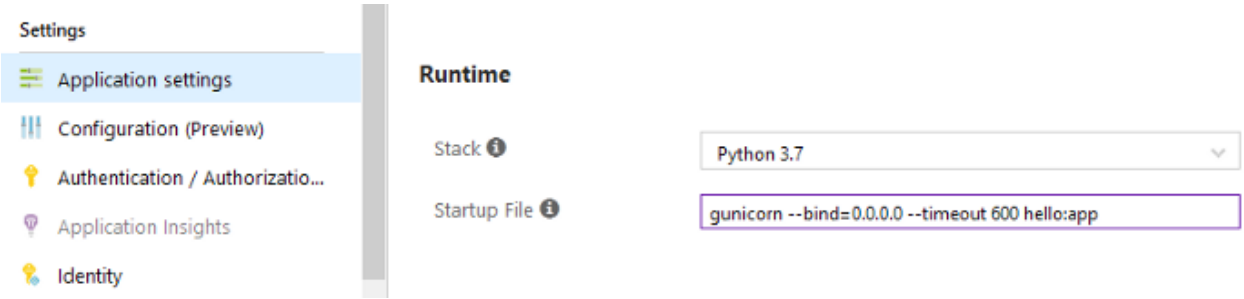
You can see similar log information in the Deployment Center:



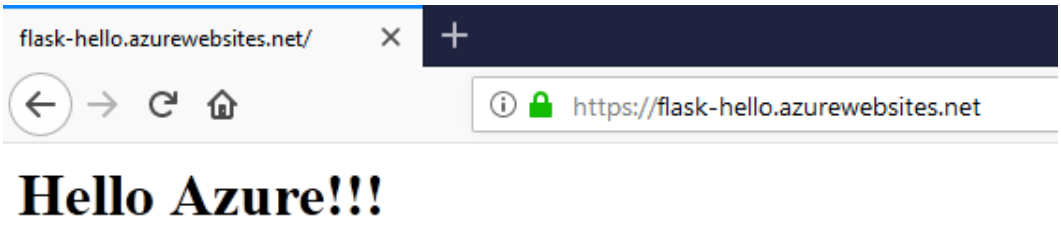
If you refresh your app, you will see that nothing changed yet. The reason is App Service uses Gunicorn WSGI HTTP Server to run an app, which looks for a file named *application.py* or *app.py.* Since our main module is in *hello.py file,* we have to customize startup command. Go to “Application settings”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

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

gunicorn 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, out 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, it’s 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: babb1fdcd78d3e047e08c5c6d45c466a0f35f297737d421bab1adc276c54b99e
- 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