

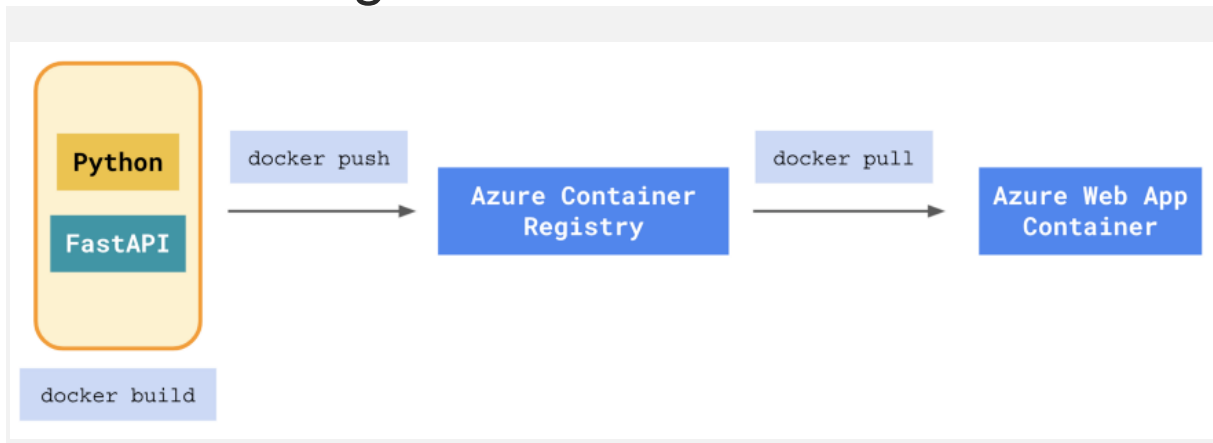
Github Actions

Github Actions is a new workflow automation tool for building, testing, and deploying your code directly from Github.

If you're already using Github for your code repository, there's no need to integrate a third-party tool to automate common developer workflows.

Simply place a Github Actions `.yaml` file in your repository, designed to run in response to specific events. (push, pull requests, etc)

Workflow diagram



We will build and push a container image to Azure Container Registry, and pull the image into an Azure Web App for deployment.

The full list of packages and technologies we'll be utilizing include:

- [VS Code](#) — as the IDE of choice.
- FastAPI— Python API development framework.
- [Docker Desktop](#)— build and run Docker container images on our local machine.
- Azure Container Registry — repository for storing our container image in Azure cloud.
- Azure App Service — PaaS service to host our FastAPI app server.
- Github Actions — automate continuous deployment workflow of FastAPI app.

Docker

Head over to [Docker Desktop](#) and install the program.

This allows you to run Docker commands on your local machine.

Build and Run Container Image

Instead of deploying our app as a package, let's build our app as a Docker image, so that the build process is outlined in a `Dockerfile`.

`Dockerfile` makes the build process seamless and reproducible.

Build your container and check that the image has been created with the `docker images` command.

```
docker build . -t fastapi-cd:1.0
docker images
```

Run your container with the image tag you specified earlier, on port 8000.

```
docker run -p 8000:8000 -t fastapi-cd:1.0
```

Check your containerized application by going to <http://localhost:8000/docs>.

You should see the same FastAPI application running.

Infrastructure setup

Azure Container Registry

Time to deploy our application in Azure. First off, create an Azure Container Registry.



Create container registry ...

Basics Networking Encryption Tags Review + create

Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments. Use Azure container registries with your existing container development and deployment pipelines. Use Azure Container Registry Tasks to build container images in Azure on-demand, or automate builds triggered by source code updates, updates to a container's base image, or timers. [Learn more](#)

Project details

Subscription *

Resource group * [Create new](#)

Instance details

Registry name * .azurecr.io

Location *

Availability zones ⓘ ☐ Enabled

i Availability zones are enabled on premium registries and in regions that support availability zones. [Learn more](#)

SKU * ⓘ

Enable the admin user option in the Access keys pane.

[Home](#) > [Microsoft.ContainerRegistry](#) > [fastapicd](#)



fastapicd | Access keys ...

Container registry

- Overview
- Activity log
- Access control (IAM)
- Tags
- Quick start
- Events
- Settings
 - Access keys

Registry name	<input type="text" value="fastapicd"/>		
Login server	<input type="text" value="fastapicd.azurecr.io"/>		
Admin user ⓘ	<input checked="" type="checkbox"/> Enabled		
Username	<input type="text" value=""/>		
Name	Password	Regenerate	
password	<input type="text" value=""/>	<input type="text" value=""/>	<input type="button" value="Regenerate"/>
password2	<input type="text" value=""/>	<input type="text" value=""/>	<input type="button" value="Regenerate"/>

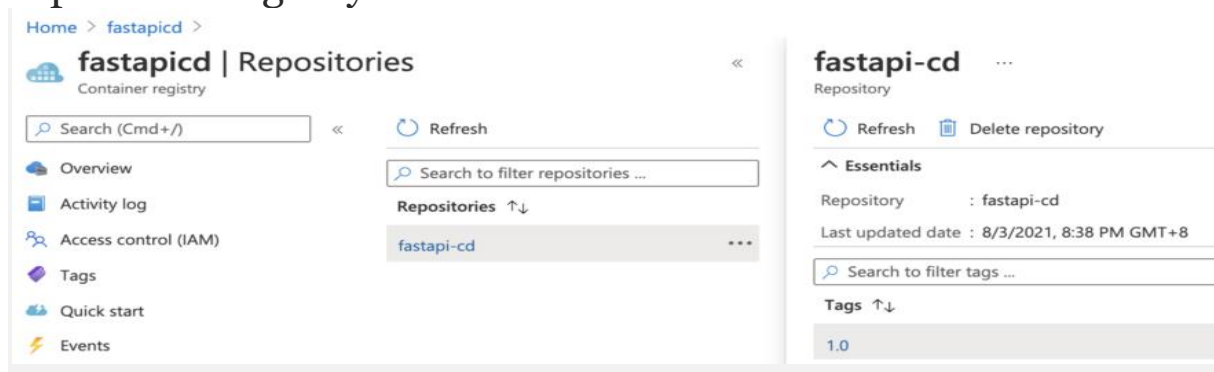
Going back to your VS Code terminal, log in to Azure with your account.

```
az login  
az acr login --name fastapicd
```

Build and push your Docker image to the registry server.

```
docker build . -t fastapicd.azurecr.io/fastapi-cd:1.0  
docker images  
docker push fastapicd.azurecr.io/fastapi-cd:1.0
```

Check that your Docker image has been successfully pushed to a repo in the registry.



Azure App Service

Next, create an App Service resource.

Create Web App ...

Basics Docker Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#) ↗

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Resource Group * ⓘ [Create new](#)

Instance Details

Name * ✓ .azurewebsites.net

Publish * ☐ Code ☒ Docker Container

Operating System * ☒ Linux ☐ Windows

Region * [Not finding your App Service Plan? Try a different region.](#)

App Service Plan

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#) ↗


Linux Plan (Southeast Asia) * ⓘ [Create new](#)


Sku and size * **Standard S1**
100 total ACU, 1.75 GB memory
[Change size](#)


For the App Service Plan, change the size to Standard S1 SKU under the Production Workloads tab.

Spec Picker

×


Dev / Test
 For less demanding workloads


Production
 For most production workloads


Isolated
 Advanced networking and scale

i The first Basic (B1) core for Linux is free for the first 30 days!

Recommended pricing tiers

P1V2 210 total ACU 3.5 GB memory Dv2-Series compute equivalent 83.95 USD/Month (Estimated)	P2V2 420 total ACU 7 GB memory Dv2-Series compute equivalent 168.63 USD/Month (Estimated)	P3V2 840 total ACU 14 GB memory Dv2-Series compute equivalent 337.26 USD/Month (Estimated)
P1V3 195 minimum ACU/vCPU 8 GB memory 2 vCPU 102.93 USD/Month (Estimated)	P2V3 195 minimum ACU/vCPU 16 GB memory 4 vCPU 205.86 USD/Month (Estimated)	P3V3 195 minimum ACU/vCPU 32 GB memory 8 vCPU 411.72 USD/Month (Estimated)


[^ See only recommended options](#)

Additional pricing tiers

S1 100 total ACU 1.75 GB memory A-Series compute equivalent 69.35 USD/Month (Estimated)	S2 200 total ACU 3.5 GB memory A-Series compute equivalent 138.70 USD/Month (Estimated)	S3 400 total ACU 7 GB memory A-Series compute equivalent 277.40 USD/Month (Estimated)
--	--	--


Included features

Every app hosted on this App Service plan will have access to these features:


Custom domains / SSL
 Configure and purchase custom domains with SNI and IP SSL bindings

Included hardware

Every instance of your App Service plan will include the following hardware configuration:


Azure Compute Units (ACU)
 Dedicated compute resources used to run applications deployed in the App Service Plan. [Learn more](#)

Apply

On the Docker configuration tab, make sure the container image you pushed earlier is selected.

Create Web App ...

Basics Docker Monitoring Tags Review + create

Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options	Single Container
Image Source	Azure Container Registry
Azure container registry options	
Registry *	fastapicd
Image *	fastapi-cd
Tag *	1.0
Startup Command ⓘ	

Once resource deployment is complete, check that your web app is running by going to <https://<your-webapp-name>.azurewebsites.net/docs>.

You should see the FastAPI swagger docs page render.

Github Actions

Now we're ready to create our Github Actions .yaml file that will automate deployment of our FastAPI application.

In VS Code, create a `.github/workflows` *directory* and

a `prod.workflow.yaml` file within.

```
mkdir .github
cd .github
mkdir workflows
cd ..
touch .github/workflows/prod.workflow.yaml
```

Build your `prod.workflow.yaml` file like below:


```

name: Build and deploy to production

on:
  push:
    branches:
      - main

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest

    steps:

      - name: Checkout GitHub Actions
        uses: actions/checkout@main

      - name: Login via Azure CLI
        uses: azure/login@v1
        with:
          creds: ${{ secrets.AZURE_CREDENTIALS }}

      - name: Login to Azure Container Registry
        uses: azure/docker-login@v1
        with:
          login-server: fastapicd.azurecr.io
          username: ${{ secrets.REGISTRY_USERNAME }}
          password: ${{ secrets.REGISTRY_PASSWORD }}

      - name: Build and push container image to registry
        run: |
          docker build . -t fastapicd.azurecr.io/fastapi-cd:${{ github.sha }}
          docker push fastapicd.azurecr.io/fastapi-cd:${{ github.sha }}

      - name: Deploy to App Service
        uses: azure/webapps-deploy@v2
        with:
          app-name: 'fastapi-cd'
          images: 'fastapicd.azurecr.io/fastapi-cd:${{ github.sha }}'
          slot-name: 'staging'

      - name: Azure logout
        run: |
          az logout

```

Now, whenever we do a git push to the main branch, Github Actions will run a deployment job that consists of the 5 steps above, each with its own name.

Push this new file to Github using the git commands:

```
git add .  
git commit -m "github actions deployment workflow"  
git push
```

Service Principal

To automate our deployment workflow with Github Actions, we need to give the actions runner a service principal to authenticate to Azure, and perform the app deployment.

In VS Code terminal, run:

```
az ad sp create-for-rbac --name "github-actions" --role  
contributor --scopes  
/subscriptions/<GUID>/resourceGroups/fastapi-cd --sdk-auth
```

You can get the subscription GUID from the subscription you used to create the resource group.

You will get a response like below:

```
{  
  "clientId": "<clientId>",  
  "clientSecret": "<clientSecret>",  
  "subscriptionId": "<subscriptionId>",  
  "tenantId": "<tenantId>",  
  "activeDirectoryEndpointUrl": "https://login.microsoftonline.com",  
  "resourceManagerEndpointUrl": "https://management.azure.com/",  
  "activeDirectoryGraphResourceId": "https://graph.windows.net/",  
  "sqlManagementEndpointUrl":  
"https://management.core.windows.net:8443/",  
  "galleryEndpointUrl": "https://gallery.azure.com/",  
  "managementEndpointUrl": "https://management.core.windows.net/"  
}
```

Copy the response and save it. You won't be able to see it again.

Head over to Github settings, and create 3 Github Secrets:

- AZURE CREDENTIALS: Entire JSON response from above.
- REGISTRY_USERNAME: clientId value from JSON response.
- REGISTRY_PASSWORD: clientSecret value from JSON response.

Actions secrets / New secret

Name

AZURE_CREDENTIALS


Value

```
{  
  "clientId":   
  "clientSecret":   
  "subscriptionId":   
  "tenantId":   
  "activeDirectoryEndpointUrl": "https://login.microsoftonline.com",  
  "resourceManagerEndpointUrl": "https://management.azure.com/",  
  "activeDirectoryGraphResourceId": "https://graph.windows.net/",  
  "sqlManagementEndpointUrl": "https://management.core.windows.net:8443/",  
  "galleryEndpointUrl": "https://gallery.azure.com/",  
  "managementEndpointUrl": "https://management.core.windows.net/"  
}
```

Add secret

Configure App Service to use Github Actions for continuous deployment

In the Deployment Center pane of the App Service, link your Github repo main branch, and configure deployment to use Github Actions.



fastapi-cd | Deployment Center

App Service

Save

Discard

Browse

Manage publish profile

Leave

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Security

Events (preview)

Deployment

Quickstart

Deployment credentials

Deployment slots

Deployment Center

Settings

Configuration

Container settings (Classic)

Authentication

Authentication (classic)

Application Insights

Identity

Backups

Custom domains

TLS/SSL settings

Networking

Scale up (App Service plan)

Scale out (App Service plan)

Settings

Logs

FTPS credentials

Use these settings to configure your container app deployment model and regist

Source*

☐ Container Registry: Setup up your app to pu

☒ GitHub Actions: Build, deploy, and manage y

☐ Azure Pipelines: Configure a robust deploym

GitHub Actions

If you can't find an organization or repository, you may need to enable additional

Signed in as

nathancheng-data [Change Account](#)

Organization*

nathancheng-data

Repository*

fastapi-cd

Branch*

main

Workflow Option*

☐ Add a workflow: Add a new workflow file 'm

☒ Use available workflow: Use one of the work

Registry settings

Registry source

Azure Container Registry

Subscription ID*

Registry*

fastapicd

Image*

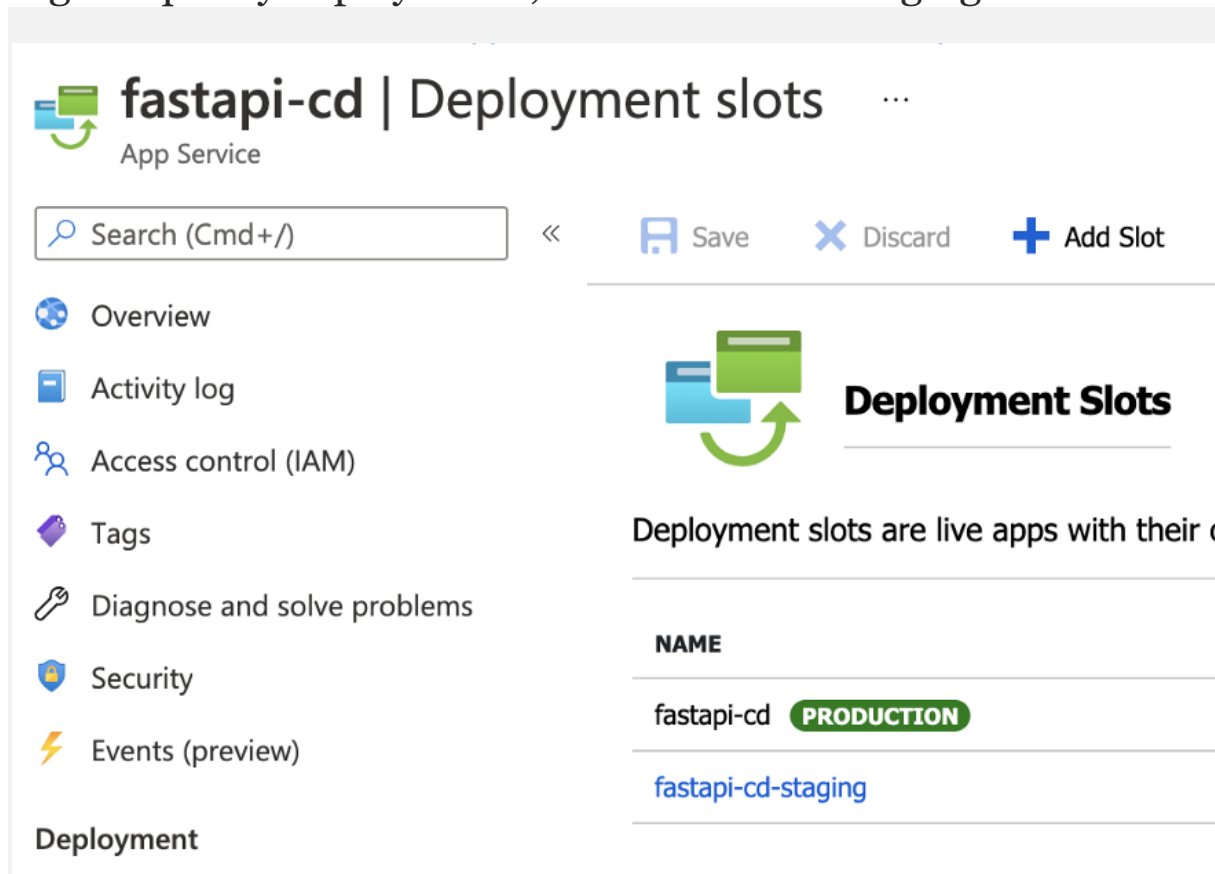
fastapi-cd

Tag

Tagged with GitHub commit ID

Create staging deployment slot

The current app is running in a production slot. To guarantee higher-quality deployments, we will create a staging slot.



Slots are useful for multiple reasons:

- Changes can be released to a smaller % of users in a staging slot, to validate new releases before swapping the staging slot with production.
- Staging slot app instances are warmed up before being swapped into production, guaranteeing that the app is responsive to requests.
- If something goes wrong with the new release, swaps can be performed again to roll back the changes.

Check that the staging app is running like the production app by going to <https://<your-webapp-name>-staging.azurewebsites.net/docs>.

Configure the same deployment center settings for the staging slot as you did for production.

When you're comfortable swapping the staging app over to production, simply click on Swap in the Deployment Slots pane of the App Service.

I would like to give a big shoutout to all the extensive documentation and YouTube videos that helped to complete the task:

- [Deploy a custom container to App Service using Github Actions](#)
- [Github Action repo — deploy to App Service](#)
- [Set up staging slots in App Service](#)
- [Learn Docker in 7 Easy Steps — Full Beginner's Tutorial](#)
- [Containerize FastAPI app with Docker](#)
- [Push Docker image to Azure Container Registry](#)