

Open in app ↗



Search

Write



★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Deploy Python FastAPI using Azure Container Registry

Harsh Bakshi · [Follow](#)

9 min read · Aug 8



5



```
C:\MCN Solution\FastAPI Article>python -m uvicorn main:app --reload
INFO:      Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:      Started reloader process [19736] using statreload
INFO:      Started server process [7368]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
```

Introduction

In the world of software development, containerization has become a game-changer. It allows developers to create, deliver, and run applications seamlessly across different environments. One powerful platform that simplifies container management is Azure Container Registry (ACR), offered by Microsoft Azure. It has gained popularity among developers for its robustness and scalability. In this article, we will explore the possibilities of deploying a Python FastAPI application using Azure Container Registry.

FastAPI is known for its speed, simplicity, and flexibility, making it a favorite among developers. By combining the strengths of FastAPI and Azure Container Registry, we can ensure reliable and scalable development and deployment. So, let's dive into the exciting world of containerized web apps with FastAPI and Azure Container Registry!

What is FastAPI?

FastAPI is a modern and high-performance web framework used for building APIs with Python. It offers simplicity, efficiency, and ease of use, making it a popular choice for developers who want to create robust and fast web applications. One of FastAPI's notable features is its ability to automatically generate detailed and interactive API documentation using Python-type hints. This feature not only helps developers understand the available endpoints but also allows them to interact with the API directly through user-friendly interfaces like Swagger UI or ReDoc.

How does FastAPI work?

Here's a simplified explanation:

- **Easy to Use:** FastAPI provides a user-friendly approach to defining and handling API endpoints.
- **Fast and Efficient:** FastAPI utilizes asynchronous programming, enabling it to handle multiple requests simultaneously, resulting in impressive speed and efficiency.
- **Automatic Documentation:** With FastAPI, your API documentation is generated automatically based on the code you write. This saves you time and effort in maintaining separate documentation.
- **Data Validation:** FastAPI ensures that the data sent to your API is valid and correct, enhancing the reliability of your application.

- **URL Routing:** FastAPI helps organize your API endpoints using URLs, simplifying the management and maintenance of your web application.
- **Request Handling:** FastAPI automatically parses request data (e.g., JSON payloads or form data) and provides easy access within the view function. It also handles serialization of response data into the appropriate format.

Prerequisites

- Python 3.7+
- Docker 4.10+
- Ubuntu 22.04.2 LTS

Installation

Step 1: Install FastAPI and Uvicorn using Python's pip:

```
pip install fastapi
pip install "uvicorn[standard]"
```

Step 2: Create a file named *main.py* with the following code:

```
##### main.py #####

from fastapi import FastAPI

app = FastAPI()

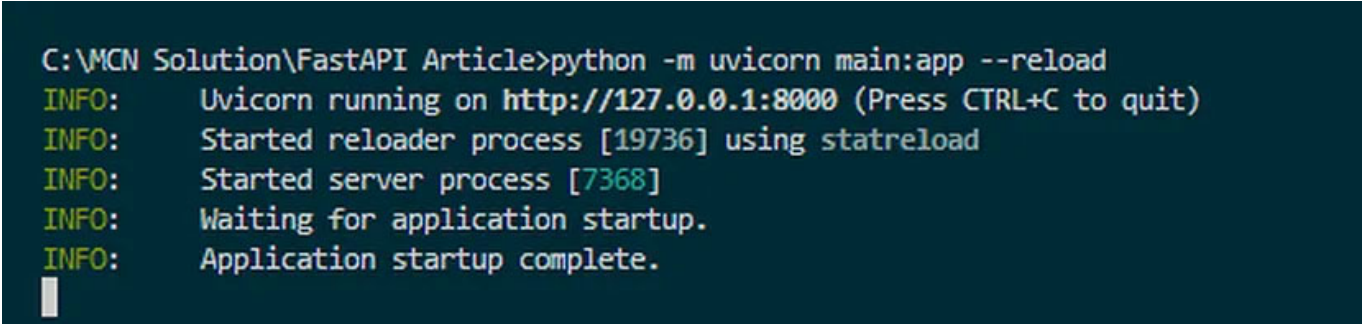
@app.get("/")
def read_root():
    return "Hello": "World"
```

This code sets up a simple GET endpoint for demonstration purposes.

Step 3: Run the program:

```
python -m uvicorn main:app --reload
```

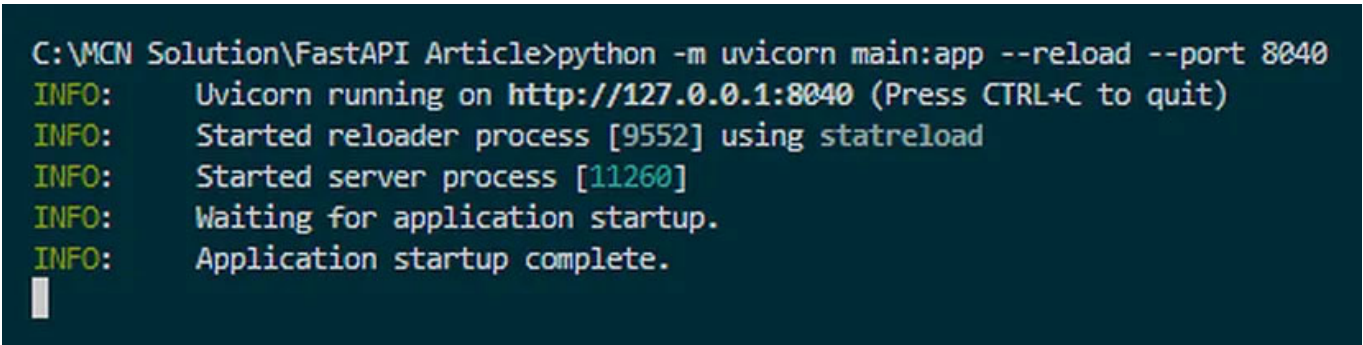
After executing this command, you will see your program running on port 8000:

A terminal window with a dark blue background. The prompt is 'C:\MCN Solution\FastAPI Article>'. The command entered is 'python -m uvicorn main:app --reload'. The output shows several INFO messages: 'Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)', 'Started reloader process [19736] using statreload', 'Started server process [7368]', 'Waiting for application startup.', and 'Application startup complete.'.

```
C:\MCN Solution\FastAPI Article>python -m uvicorn main:app --reload
INFO:      Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:      Started reloader process [19736] using statreload
INFO:      Started server process [7368]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
```

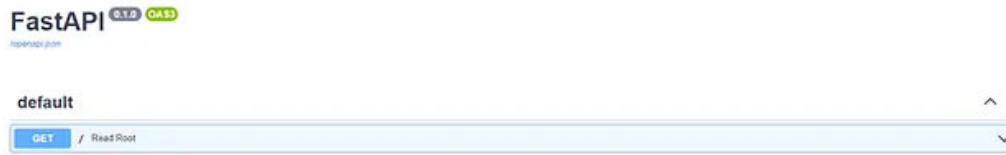
You can also specify a different port of your choice using the command:

```
python -m uvicorn main:app --reload --port 8040
```

A terminal window with a dark blue background. The prompt is 'C:\MCN Solution\FastAPI Article>'. The command entered is 'python -m uvicorn main:app --reload --port 8040'. The output shows several INFO messages: 'Uvicorn running on http://127.0.0.1:8040 (Press CTRL+C to quit)', 'Started reloader process [9552] using statreload', 'Started server process [11260]', 'Waiting for application startup.', and 'Application startup complete.'.

```
C:\MCN Solution\FastAPI Article>python -m uvicorn main:app --reload --port 8040
INFO:      Uvicorn running on http://127.0.0.1:8040 (Press CTRL+C to quit)
INFO:      Started reloader process [9552] using statreload
INFO:      Started server process [11260]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
```

After running this command, you can access the Swagger documentation at <http://127.0.0.1:8040/docs>.

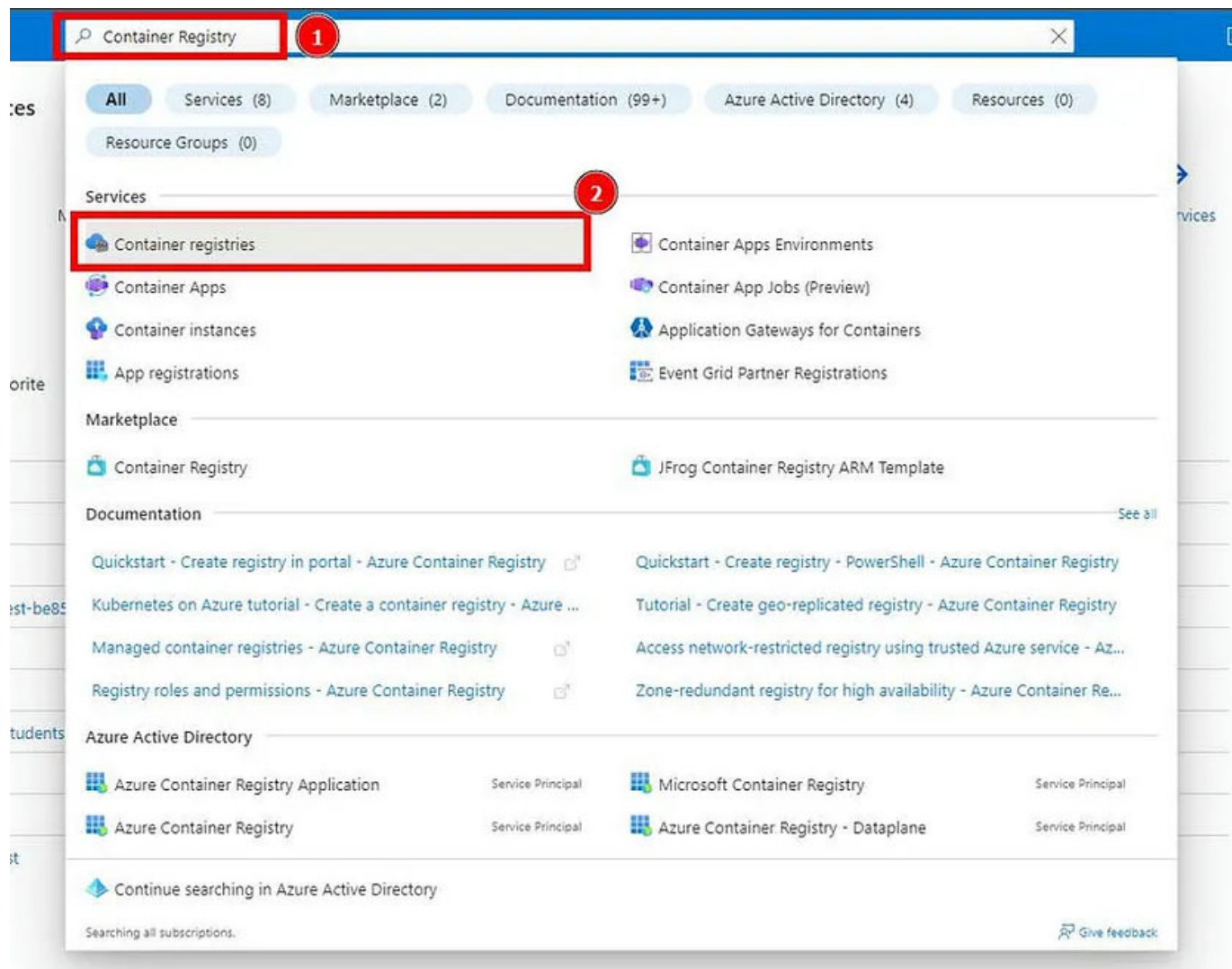


Setup the Azure Container Registry

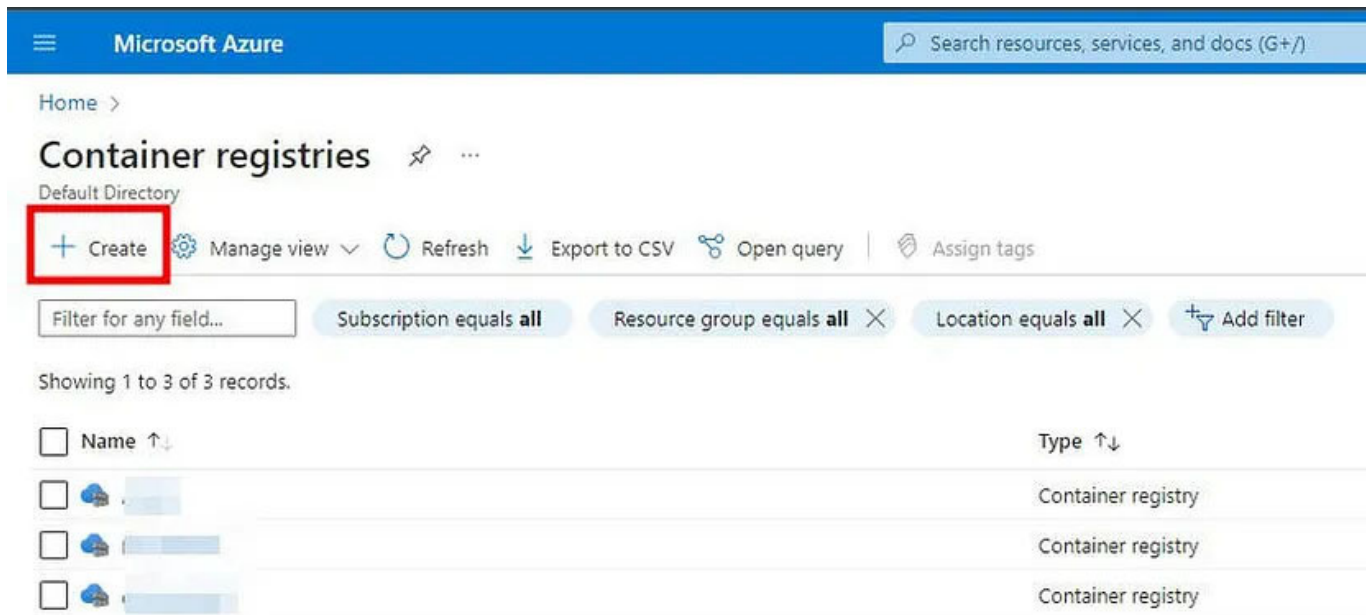
Azure Container Registry acts as a centralized repository for containerized applications. It stores and manages these containers, ensuring easy access, sharing, and downloading. Think of it as a library where you can borrow program containers. By leveraging Azure Container Registry, developers and teams can efficiently manage and distribute their software.

Step 1: On the Azure portal, search for Azure Container Registry.

Step 2: Select Azure Container Registry from the results.



Step 3: Click on Create.



The screenshot shows the Microsoft Azure portal interface for 'Container registries'. The top navigation bar includes the Microsoft Azure logo and a search bar. Below the navigation bar, the 'Container registries' page is displayed. A red box highlights the '+ Create' button. The page also features a 'Default Directory' section with various action buttons like 'Manage view', 'Refresh', 'Export to CSV', 'Open query', and 'Assign tags'. A filter bar is present with a search input and several filter buttons. Below the filter bar, it states 'Showing 1 to 3 of 3 records.' and displays a table of existing container registries.

<input type="checkbox"/>	Name ↑↓	Type ↑↓
<input type="checkbox"/>	[Redacted]	Container registry
<input type="checkbox"/>	[Redacted]	Container registry
<input type="checkbox"/>	[Redacted]	Container registry


Step 4: Select the subscription type and create a new Resource Group.

Provide a meaningful name for your Registry. Click on Review + Create and verify all the details. Finally, click Create to create your Container Registry.

Microsoft Azure

Search resources, services, and docs (G+)

Home > Container registries >

 **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 *

Azure for Students

1

Resource group *

(New) TestAPI

Create new

2

Instance details

Registry name *

TestFastAPIArticle

.azurecr.io

3

Location *

East US

Availability zones ⓘ

☐ Enabled

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

SKU * ⓘ

Standard

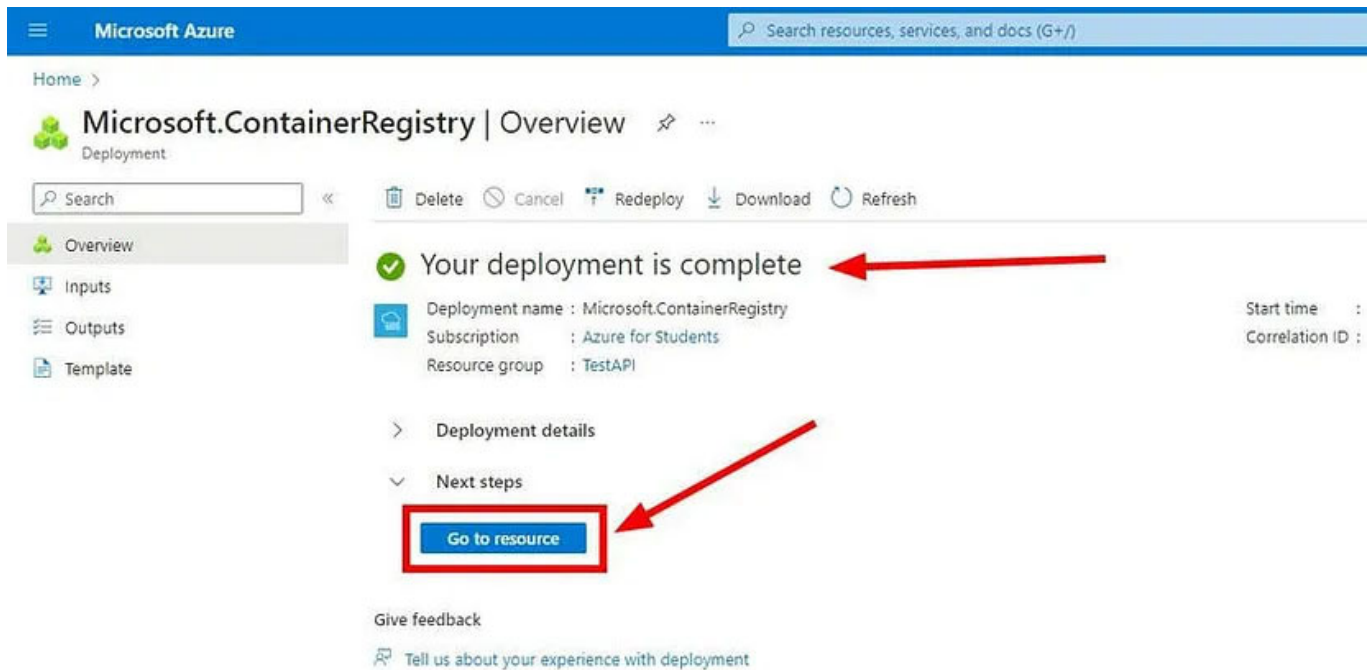
Review + create

4

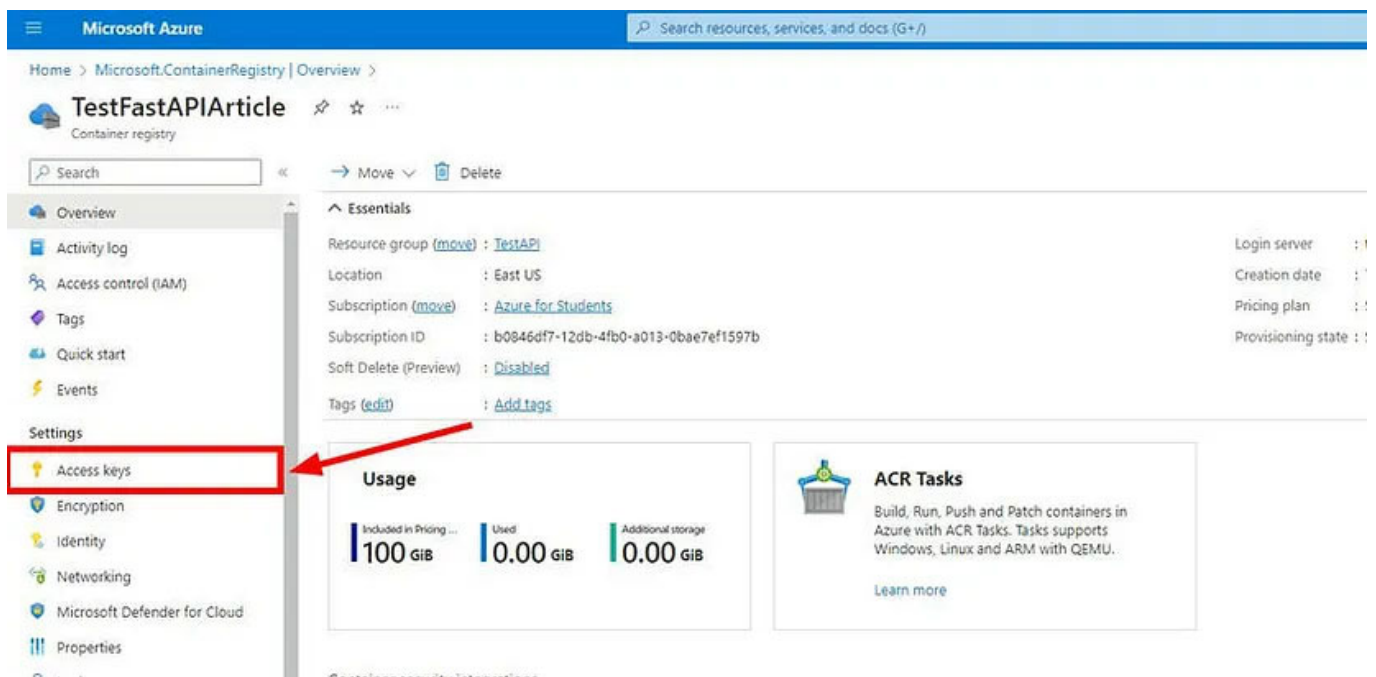
< Previous

Next: Networking >

Step 5: Wait for a minute until your Container Registry is created. Once you see the “deployment is complete” message, your Container is ready. Click on the Go to Resource button.

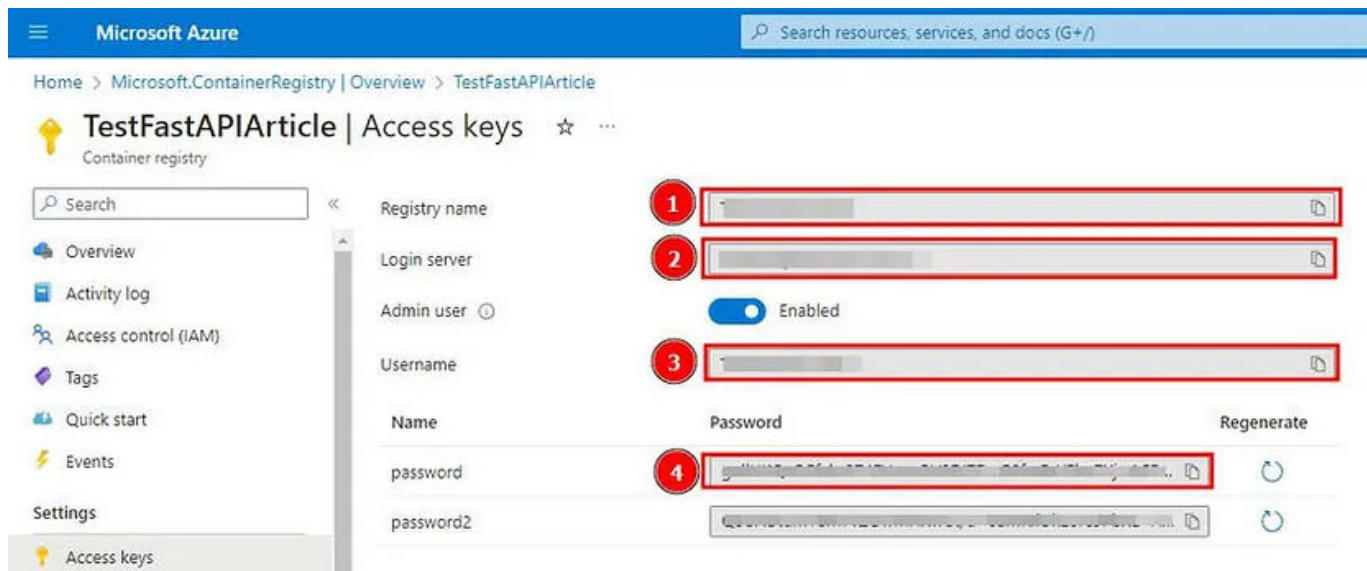


Step 6: You will now see the Container Registry Dashboard. To push your FastAPI code into this container, you will need an access key. Click on Access Keys in the left navigation bar.



Step 7: In the Access Keys section, you will find your Registry name, Login server, and Admin user information. By default, the Admin User is disabled.

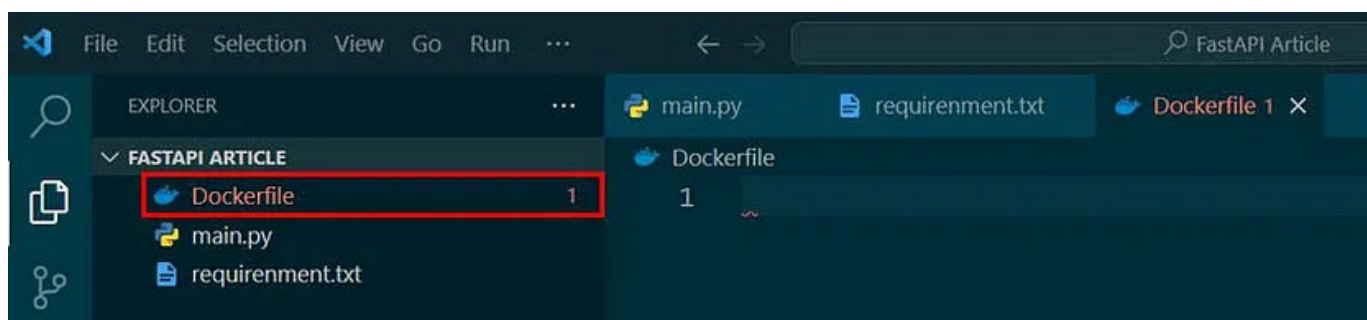
Enable it to see your Username and two Passwords: Password1 and Password2. Make a note of these credentials. They will be used to push your FastAPI code into the container.



Create Container Image using Docker Build

Before creating an image, make sure to freeze your Python modules in a requirement.txt file.

Step 1: Create a Dockerfile:



Step 2: Configure the Dockerfile with the necessary details, such as the Python version, directory copying, and the run command:

```
FROM python:3.10

WORKDIR /data
COPY . /data

RUN pip install -r requirement.txt

COPY . .

CMD [ "uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80" ]
```

Copy and paste this code into your Dockerfile.

Step 3: Log in to your Azure Container registry using the WSL terminal and the login command provided:

```
docker login [Your Azure Login server URL] -u [Username] -p [Password]
```

Note: Replace [Your Azure Login server URL], [Username], and [Password] with your container credentials. Ensure that your Docker runs in the background.



Step 4: Build your container image using the following command:

```
docker build -t [Azure Login server]/[Container name]:[Tag] .
```

Note: Replace [Azure Login server], [Container name], and [Tag] with your container credentials. The Tag helps version your container image.

```
divyansh.gupta@DESKTOP-RSPQ1UT:/mnt/c/KN Solution/FastAPI Article$ docker build -t testfastapi:v1 .  
[*] Building 2.5s (10/10) FINISHED  
-> [internal] load build definition from Dockerfile  
-> >> transferring dockerfile: 38B  
-> [internal] load .dockerignore  
-> >> transferring context: 2B  
-> [internal] load metadata for docker.io/library/python:3.10  
-> [internal] load build context  
-> >> transferring context: 92B  
-> [1/5] FROM docker.io/library/python:3.10@sha256:b24f2d9aaa0093ee4fa95a1a574badbaed4d5c7fda68ae609b8145973183f8b  
-> >> resolve docker.io/library/python:3.10@sha256:b24f2d9aaa0093ee4fa95a1a574badbaed4d5c7fda68ae609b8145973183f8b  
-> CACHED [2/5] WORKDIR /data  
-> CACHED [3/5] COPY . /data  
-> CACHED [4/5] RUN pip install -r requirement.txt  
-> CACHED [5/5] COPY . .  
-> >> exporting layers  
-> >> writing image sha256:84306e3e17d673f8a9ba3f6f102944c001d57ec573c0fa08b26d7ad7da2860  
-> >> naming to testfastapiarticle.guuprr.io/testfastapi:v1
```

Step 5: Test your image before pushing it to the main server using the following command:

```
docker run -p 8040:80 [Azure Login Server]/[Container name]:[Tag]
```

Note: Replace [Azure Login server], [Container name], and [Tag] with your container credentials.

By running this command, your image will be accessible on <http://localhost:8040/docs>. Test all your endpoints, and if everything works fine, you can proceed to push the image to your Azure Container Registry.

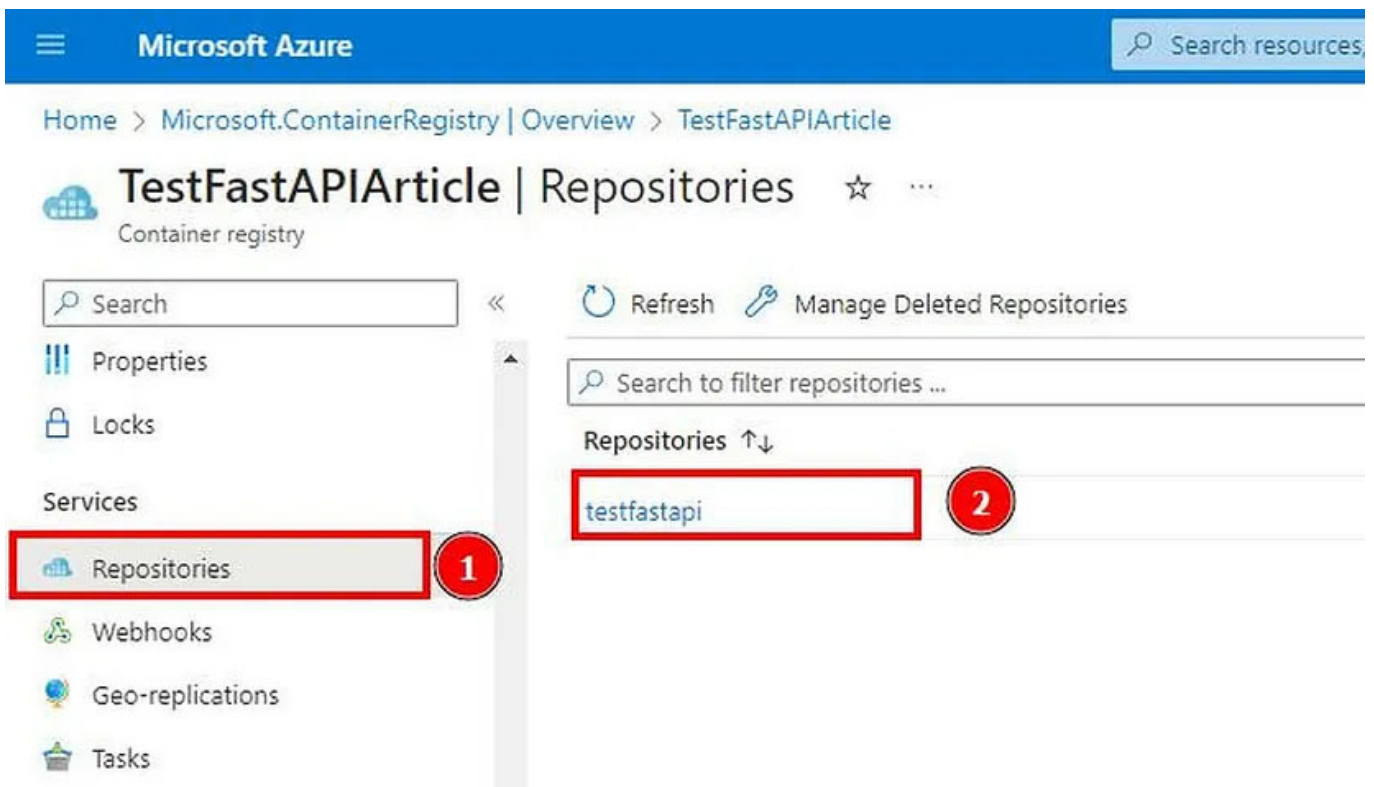
Step 6: Push the image to Azure Container Registry using the following command:

```
docker push [Azure Login server]/[Container name]:[Tag]
```

Note: Replace [Azure Login server], [Container name], and [Tag] with your container credentials.

```
divyansh_gupta@DESKTOP-R5PQ1UT:/mnt/c/MCN Solution/FastAPI Article$ docker push t[redacted] /testfastapi:v1
The push refers to repository [testfastapiarticle.azurecr.io/testfastapi]
50569d340189: Pushed
a7ad2214ba59: Pushed
10abc389df70: Pushed
22811a905058: Pushed
590598117129: Pushed
980fef3df566: Pushed
9bd21ac8c9e8: Pushed
c5f1d4dd95f0: Pushed
6a25221bdf24: Pushed
b578f477cd5d: Pushed
b298f9991a11: Pushed
c94dc8fa3d89: Pushed
v1: digest: sha256:1d66e683d6030b172cae0132429cc0414431d0a8a25c0323a4df805003e6c5e8 size: 2840
```

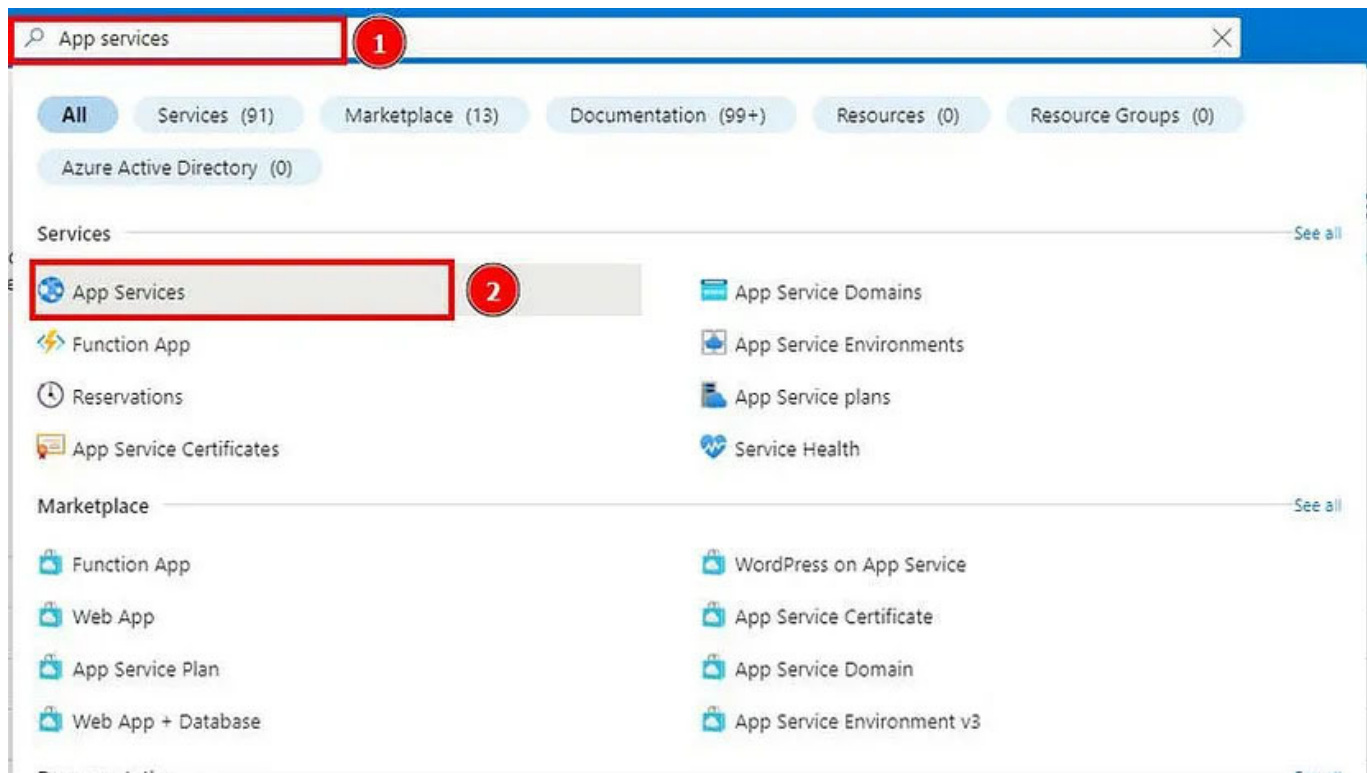
Step 7: Check the image in your Azure Container Registry by clicking on Repositories in the left navigation bar. You will see your image listed in the Repositories section.



Create an App Service for Deployment

Azure App Service is a cloud computing platform provided by Microsoft. It allows developers to build, deploy, and manage web applications and APIs without worrying about underlying infrastructure. Think of it as a virtual space where you can create and host your website or web application without needing to set up and maintain servers. Azure App Service takes care of the technical details, allowing you to focus on building your application and making it accessible to users on the internet.

Step 1: Search for App Service on the Azure portal and select App Service. Click on the Create icon and choose Web App.



Step 2: Create a Web App. Select the subscription and Resource Group, provide a meaningful URL, select Docker Container, and choose Linux as the operating system. Click Next: Docker.

Microsoft Azure

Search resources, services, and documentation

Home > App Services >

Create Web App

Basics Docker Networking 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 *

Azure for Students

1

Resource Group *

TestAPI

2

Create new

Instance Details

Need a database? [Try the new Web + Database experience.](#)

Name *

TestFastAPIArticle

3

Publish *

☐ Code

☒ Docker Container

☐ Static Web App

4

Operating System *

☒ Linux

☐ Windows

5

Region *

East US

Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

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

Linux Plan (East US) *

Create new

Pricing plan

Basic B1 (100 total ACU, 1.75 GB memory, 1 vCPU)

Zone redundancy

Review + create

< Previous

Next : Docker >

6

Step 3: Set up Docker. Select Single Container as the option, choose Azure Container Registry as the Image Source, and select the Registry you created

earlier. The image name and tag will be automatically detected. Click on Review + Create, verify all the details, and then click Create.

Microsoft Azure

Home > App Services >

Create Web App

Basics Docker Networking 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 (1)

Image Source: Azure Container Registry (2)

Azure container registry options

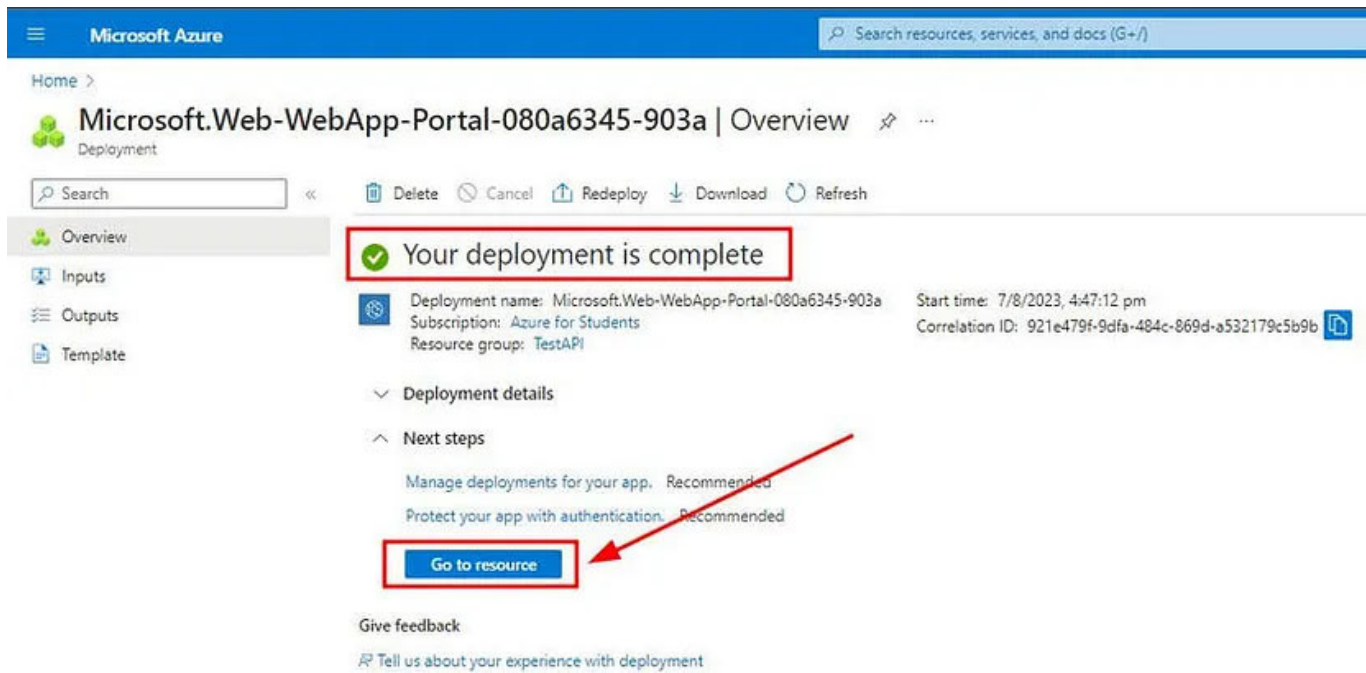
Registry *: TestFastAPIArticle (3)

Image *: testfastapi (4)

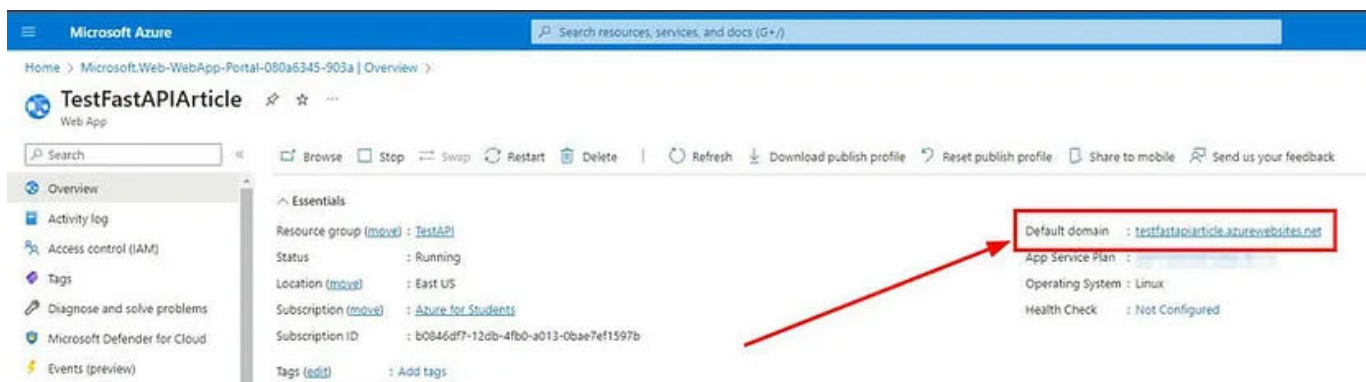
Tag *: v1 (5)

Startup Command ⓘ

Step 4: Wait for a minute for your container to deploy. Once you see the deployment complete screen, click on the Go to Resource button.



Step 5: The deployment is complete, and you will see the URL where your FastAPI application is hosted.



Congratulations! You have successfully hosted your FastAPI application in Azure using Azure Container Registry.

Conclusion

Deploying Python FastAPI using Azure Container Registry simplifies and accelerates the process of launching web applications on the internet. It's like packaging your app in a box and storing it in a secure place called Azure Container Registry, ensuring easy management and sharing of your

applications. By leveraging Azure Container Registry, you can ensure that your apps run smoothly in different environments and can scale efficiently to handle increased traffic.

At Skrots, we also offer similar services to our clients. We provide seamless deployment of Python FastAPI applications using our own container registry solution. Just like Azure Container Registry, our platform ensures reliability, scalability, and easy management of containerized applications. If you want to experience the power of containerized web apps with FastAPI, we recommend visiting Skrots to explore our services. Feel free to check out all the services we provide at <https://skrots.com/services>. Thank you for choosing Skrots!

FAQs

Q: What is Azure Container Registry, and why should I use it to deploy my FastAPI application?

A: Azure Container Registry is a secure and private container registry service provided by Microsoft Azure. You should use it to deploy your FastAPI application because it simplifies container management, storage, and distribution, ensuring a reliable and scalable deployment process.

Q: How does containerization benefit my FastAPI application deployment?

A: Containerization packages your FastAPI application and its dependencies into a single container, ensuring consistency and easy portability. It eliminates the challenges caused by differences in environments, making deployment smoother and more reliable.

Q: Can I use Azure Container Registry with other Azure services?

A: Absolutely! Azure Container Registry seamlessly integrates with other Azure services, enabling features like continuous integration and continuous deployment (CI/CD), auto-scaling, and monitoring. This allows you to build a comprehensive application ecosystem in the Azure cloud.

Q: How does Azure Container Registry help with scaling my FastAPI application?

A: Azure Container Registry simplifies scaling by allowing the distribution and deployment of multiple instances of your containerized app across various servers. This ensures effective handling of increased traffic and user demand.

Q: Can I automatically update my deployed FastAPI application when I make changes to the code?

A: Absolutely! Azure Container Registry can be integrated with Azure services like Azure Kubernetes Service (AKS) or Azure Web App, enabling automatic updates to your application whenever you push changes to the container registry. This eliminates the need for manual intervention and ensures your app stays up-to-date effortlessly.

[ad_1]

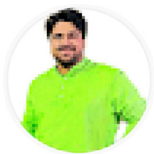
At Skrots, we also offer similar services to our clients. We provide seamless deployment of Python FastAPI applications using our own container registry solution. Just like Azure Container Registry, our platform ensures reliability, scalability, and easy management of containerized applications. If you want to experience the power of containerized web apps with FastAPI, we

recommend visiting [Skrots](https://skrots.com) to explore our services. Feel free to check out all the services we provide at https://skrots.com/services. Thank you for choosing Skrots!

Thanks, Harsh

Founder | CEO — Skrots

Learn more about our blog at [Blog at Skrots](https://blog.skrots.com). Checkout our list of services on [Skrots](https://skrots.com). Give a look at our website design at [Skrots](https://skrots.com). Checkout our LinkedIn Page at [LinkedIn.com](https://www.linkedin.com/company/skrots). Check out our original post at https://blog.skrots.com/deploy-python-fastapi-usign-azure-container-registry/?feed_id=1151&_unique_id=64d283914994d.

[Azure](#)[AI](#)[ArtificialIntelligence](#)[Cloud](#)

Written by Harsh Bakshi

34 Followers

Follow

