

Here's a **very simple Streamlit app** that is perfect for **testing Docker builds and pushing to Azure Container Registry (ACR)**.

It's intentionally minimal so you can easily confirm that:

- Docker image builds correctly
  - Container runs
  - Image pushes to ACR
  - App works after deployment
- 

## 1 Simple Streamlit app (app.py)

```
import streamlit as st
import socket
import datetime

st.set_page_config(page_title="ACR Docker Test", layout="centered")

st.title("🚀 Streamlit Docker Test App")

st.write("This app is running inside a Docker container.")

st.subheader("Container Info")
st.write(f"**Hostname:** {socket.gethostname()}")
st.write(f"**Current time:** {datetime.datetime.utcnow()} UTC")

st.success("If you see this, your Docker image works! 🎉")
```

---

## 2 requirements.txt

```
streamlit==1.29.0
```

(You can change the version later if needed.)

---

## 3 Minimal Dockerfile

```
# Use a small Python base image
FROM python:3.10-slim

# Set working directory
WORKDIR /app
```

```
# Copy dependency file  
COPY requirements.txt .  
  
# Install dependencies  
RUN pip install --no-cache-dir -r requirements.txt  
  
# Copy app source  
COPY app.py .  
  
# Expose Streamlit port  
EXPOSE 8501  
  
# Run Streamlit  
CMD ["streamlit", "run", "app.py", "--server.port=8501", "--server.address=0.0.0.0"]
```

---

## 4 Build & run locally (test first)

```
docker build -t streamlit-test .  
docker run -p 8501:8501 streamlit-test
```

Open:

👉 <http://localhost:8501>

---

## 5 Push image to Azure Container Registry (ACR)

### Login to Azure & ACR

```
az login  
az acr login --name <your-acr-name>
```

### Tag the image

```
docker tag streamlit-test <your-acr-name>.azurecr.io/streamlit-test:v1
```

### Push to ACR

```
docker push <your-acr-name>.azurecr.io/streamlit-test:v1
```

---

## 6 Verify image in ACR

```
az acr repository list --name <your-acr-name> --output table
```

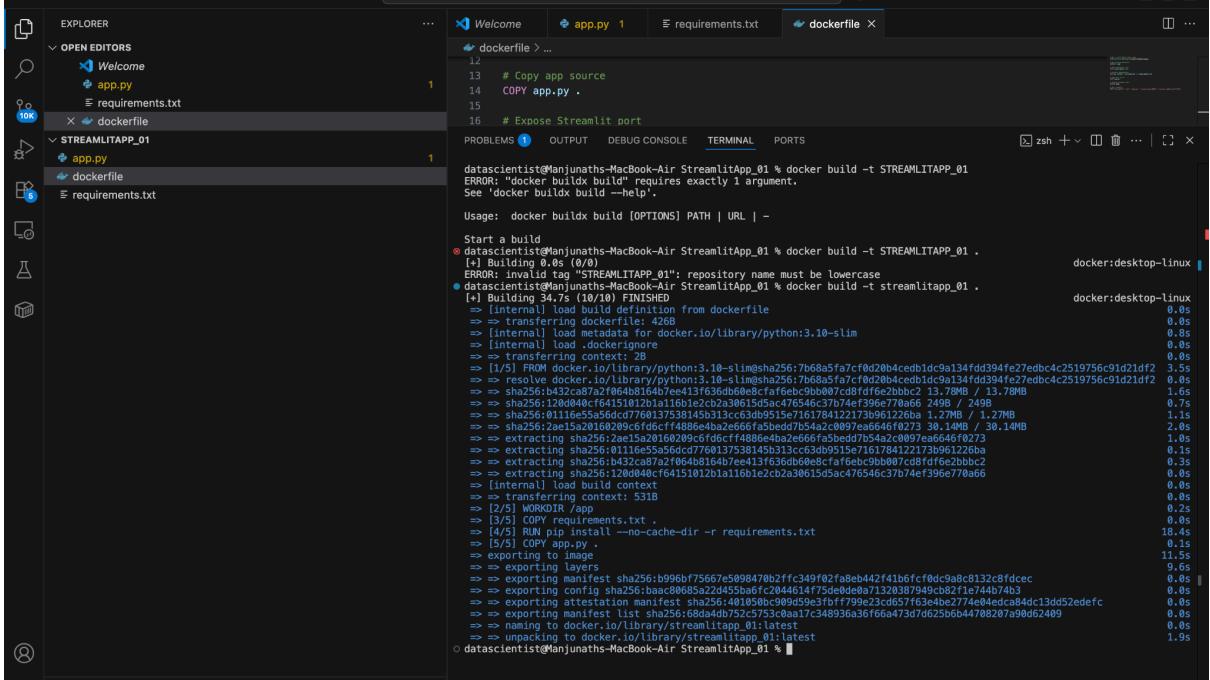
## ✓ Why this app is good for testing ACR

- No external dependencies
- Fast build
- Uses default Streamlit port (8501)
- Shows container hostname (confirms containerized execution)
- Works in **Azure App Service / Azure Container Apps / AKS**

If you want, I can also provide:

- Azure **Container App** deployment
- Azure **App Service for Containers**
- **ACR authentication via managed identity**
- **Multi-arch Dockerfile**
- **CI/CD (GitHub Actions)**

Just tell me 



The screenshot shows the Visual Studio Code interface with the Streamlit application structure. In the Explorer sidebar, there are folders for 'OPEN EDITORS' (Welcome, app.py, requirements.txt) and 'STREAMLITAPP\_01' (app.py, dockerfile, requirements.txt). The 'dockerfile' tab in the editor shows a Dockerfile with the following content:

```
12
13 # Copy app source
14 COPY app.py .
15
16 # Expose Streamlit port
```

The 'TERMINAL' tab shows the command: `dataScientist@Manjunatha-MacBook-Air StreamlitApp_01 % docker build -t STREAMLITAPP_01`. The output of the build command is displayed, showing the Docker build process step-by-step, including pulling Python 3.10-slim, extracting files, and copying the requirements file.

dataScientist@Manjunaths-MacBook-Air StreamlitApp\_01% **docker build -t streamlitapp\_01 .**

**(Note dot (.) need to be added at end )**

[+] Building 34.7s (10/10) FINISHED

docker:desktop-linux  
=> [internal] load build definition from dockerfile  
0.0s  
=> => transferring dockerfile: 426B 0.0s  
=> [internal] load metadata for docker.io/library/python:3.10-slim  
0.8s  
=> [internal] load .dockerignore 0.0s  
=> => transferring context: 2B 0.0s  
=> [1/5] FROM  
docker.io/library/python:3.10-slim@sha256:7b68a5fa7cf0d20b4cedb1dc9a134fdd394fe27ed  
bc4c2519756c91d21df2 3.5s  
=> => resolve  
docker.io/library/python:3.10-slim@sha256:7b68a5fa7cf0d20b4cedb1dc9a134fdd394fe27ed  
bc4c2519756c91d21df2 0.0s  
=> => sha256:b432ca87a2f064b8164b7ee413f636db60e8cfaf6ebc9bb007cd8fdf6e2bbbc2  
13.78MB / 13.78MB 1.6s  
=> =>  
sha256:120d040cf64151012b1a116b1e2cb2a30615d5ac476546c37b74ef396e770a66 249B  
/ 249B 0.7s  
=> =>  
sha256:01116e55a56dcd7760137538145b313cc63db9515e7161784122173b961226ba  
1.27MB / 1.27MB 1.1s  
=> => sha256:2ae15a20160209c6fd6cff4886e4ba2e666fa5bedd7b54a2c0097ea6646f0273  
30.14MB / 30.14MB 2.0s  
=> => extracting  
sha256:2ae15a20160209c6fd6cff4886e4ba2e666fa5bedd7b54a2c0097ea6646f0273  
1.0s  
=> => extracting  
sha256:01116e55a56dcd7760137538145b313cc63db9515e7161784122173b961226ba  
0.1s  
=> => extracting  
sha256:b432ca87a2f064b8164b7ee413f636db60e8cfaf6ebc9bb007cd8fdf6e2bbbc2  
0.3s  
=> => extracting  
sha256:120d040cf64151012b1a116b1e2cb2a30615d5ac476546c37b74ef396e770a66  
0.0s  
=> [internal] load build context 0.0s  
=> => transferring context: 531B 0.0s  
=> [2/5] WORKDIR /app 0.2s  
=> [3/5] COPY requirements.txt . 0.0s  
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt  
18.4s  
=> [5/5] COPY app.py . 0.1s

```
=> exporting to image                                11.5s
=> => exporting layers                             9.6s
=> => exporting manifest
sha256:b996bf75667e5098470b2ffc349f02fa8eb442f41b6fcf0dc9a8c8132c8fdcec
0.0s
=> => exporting config
sha256:baac80685a22d455ba6fc2044614f75de0de0a71320387949cb82f1e744b74b3
0.0s
=> => exporting attestation manifest
sha256:401050bc909d59e3fbff799e23cd657f63e4be2774e04edca84dc13dd52edefc
0.0s
=> => exporting manifest list
sha256:68da4db752c5753c0aa17c348936a36f66a473d7d625b6b44708207a90d62409
0.0s
=> => naming to docker.io/library/streamlitapp_01:latest
0.0s
=> => unpacking to docker.io/library/streamlitapp_01:latest
```

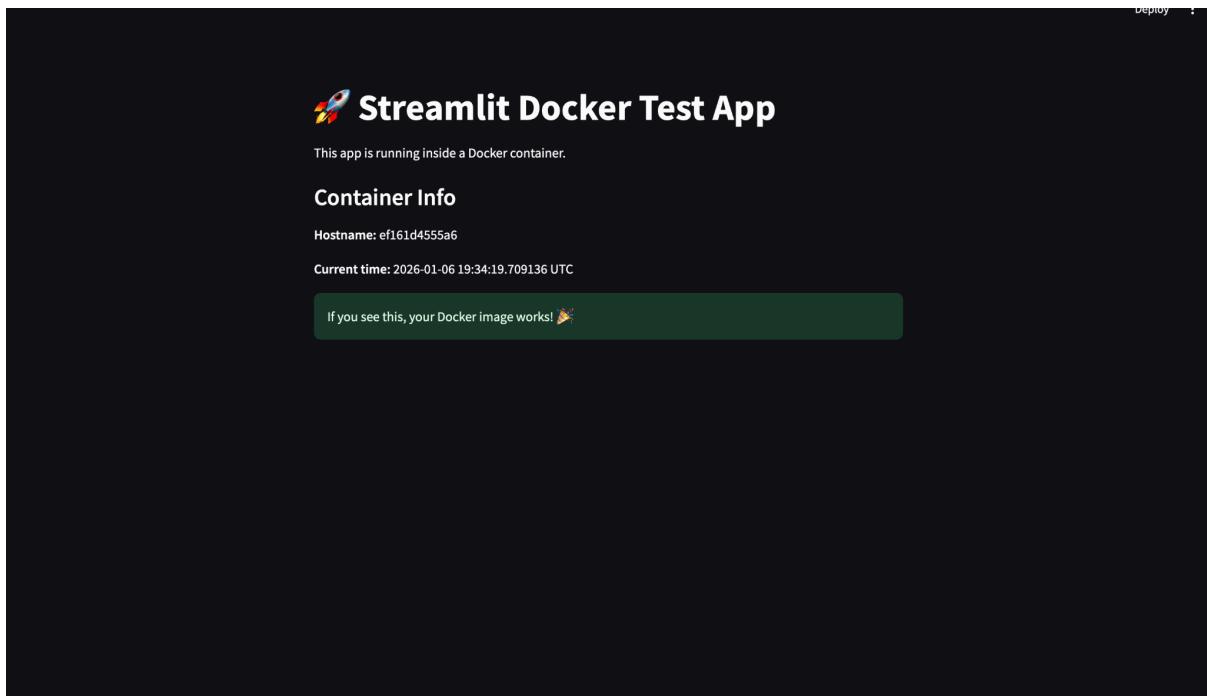
```
dataScientist@Manjunaths-MacBook-Air StreamlitApp_01 % docker run -p 8501:8501
streamlitapp_01
```

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.

You can now view your Streamlit app in your browser.

URL: <http://0.0.0.0:8501> (not this below is the correct)

<http://localhost:8501/>



## Now logging to Azure

```
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ az login  
bash: az: command not found  
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ brew --version  
Homebrew 4.5.4  
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ brew update
```

```
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ az --version  
bash: az: command not found  
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ brew list | grep azure  
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ brew list | grep azure  
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ brew update  
==> Updating Homebrew...  
Already up-to-date.  
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ brew list | grep azure
```

## Install Azure cli

```
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ brew install azure-cli
```

```
Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ az --version  
azure-cli          2.81.0
```

core	2.81.0
telemetry	1.1.0

## Dependencies:

**msal** 1.34.0b1  
**azure-mgmt-resource** 23.3.0

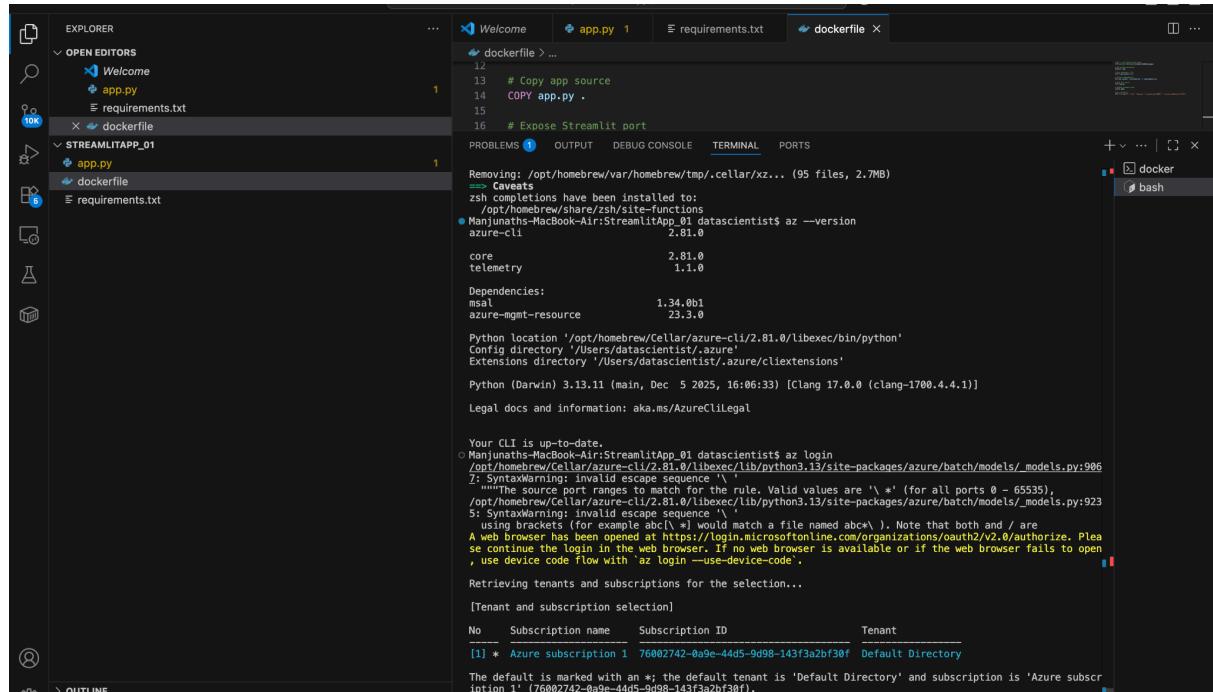
Login to Azure using below command

**Login to Azure using below command**

## az login

az login

```
az acr login --name StreamlitApp7
```



**Uppercase characters are detected in the registry name. When using its server url in docker commands, to avoid authentication errors, use all lowercase.**

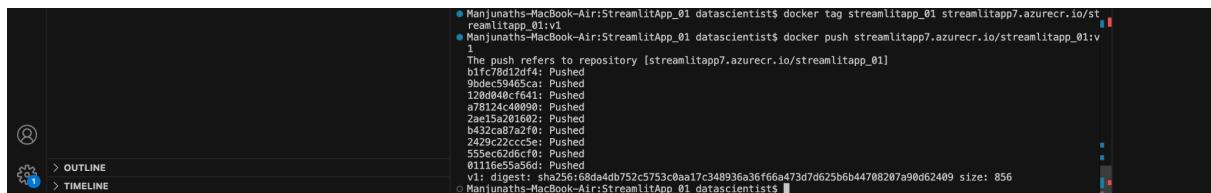
**Login Succeeded**

**Tag the image**

```
docker tag streamlitapp_01 streamlitapp7.azurecr.io/streamlitapp_01:v1
```

**Pushing to ACR**

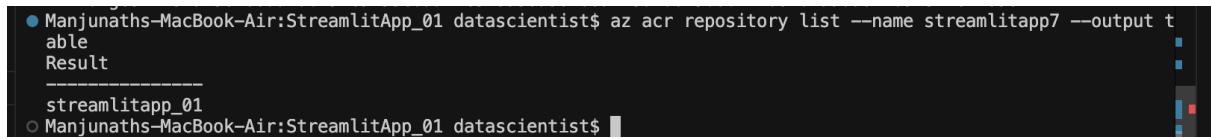
```
docker push streamlitapp7.azurecr.io/streamlitapp_01:v1
```



```
● Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ docker tag streamlitapp_01 streamlitapp7.azurecr.io/st
reamlitapp_01:v1
● Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ docker push streamlitapp7.azurecr.io/streamlitapp_01:v
1
The push refers to repository [streamlitapp7.azurecr.io/streamlitapp_01]
b1fcf8d12df4: Pushed
9bdcf59465ca: Pushed
120dd048cf641: Pushed
a78124c40090: Pushed
2ae15a201602: Pushed
b414e03a0000: Pushed
2429c22ccc5e: Pushed
555ec62d6cf0: Pushed
01116e55a56d: Pushed
v1: digest: sha256:68da4db752c5753c0aa17c348936a36f66a473d7d625b6b44708207a90d62409 size: 856
○ Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$
```

**Verify image in ACR**

```
az acr repository list --name streamlitapp7 --output table
```



```
● Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$ az acr repository list --name streamlitapp7 --output t
able
Result
-----
streamlitapp_01
○ Manjunaths-MacBook-Air:StreamlitApp_01 datascientist$
```

StreamlitApp7 | Repositories ⭐ ⋮

Container registry

Search Refresh Manage Deleted Repositories

Access control (IAM)

Tags Quick start Resource visualizer Events

Settings

- Access keys
- Encryption
- Identity
- Networking
- Microsoft Defender for Cloud
- Properties
- Locks

Services

- Repositories
- Webhooks
- Geo-replications
- Tasks
- Connected registries
- Cache

Repository permissions

Tokens

Search to filter repositories ...

Repositories ↑↓

streamlitapp\_01 Cache Rule

A screenshot of the Azure Container Registry interface. The top navigation bar shows "StreamlitApp7 | Repositories" and "Container registry". Below the bar are search and refresh buttons, and links for "Access control (IAM)", "Manage Deleted Repositories", and a search bar for filtering repositories. The left sidebar has sections for "Tags", "Quick start", "Resource visualizer", "Events", and "Settings". Under "Settings", there are links for "Access keys", "Encryption", "Identity", "Networking", "Microsoft Defender for Cloud", "Properties", and "Locks". Below "Settings" is a "Services" section with links for "Repositories", "Webhooks", "Geo-replications", "Tasks", "Connected registries", and "Cache". Under "Repositories", there is a link for "Repository permissions" and "Tokens". A search bar at the bottom of the sidebar allows filtering by repository name. The main content area shows a single repository named "streamlitapp\_01" with a "Cache Rule" link.

## Using Azure App Service

**Create Web App** ...

**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

Secure unique default hostname on. [More about this update ⓘ](#)

Publish \*  Code  Container

Operating System \*  Linux  Windows

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

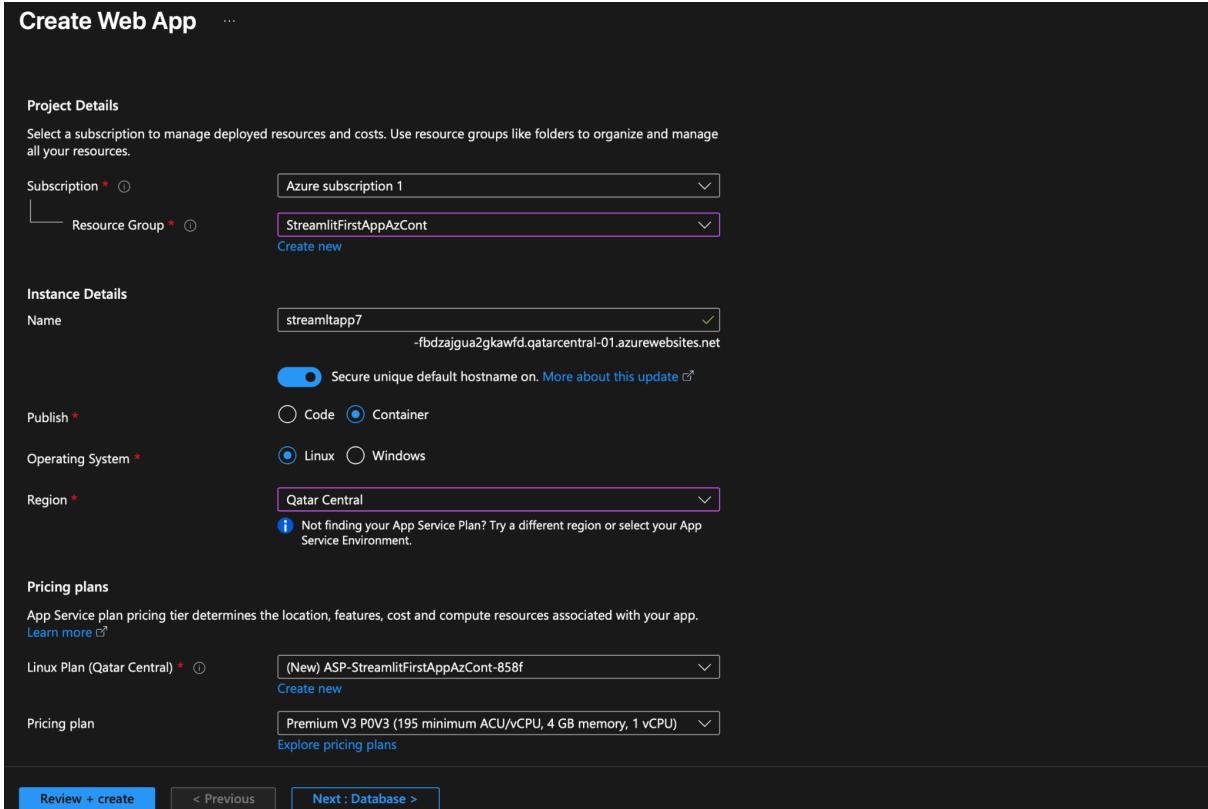
**Pricing plans**

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

Linux Plan (Qatar Central) \* ⓘ   [Create new](#)

Pricing plan   [Explore pricing plans](#)

[Review + create](#) [< Previous](#) [Next : Database >](#)



## Create Web App

Basics Database Container Networking Monitor + secure Tags Review + create

Select your preferred source for container images. You can change these settings and other dependencies after creating the app. [Learn more](#)

Enhanced configuration with sidecar support on [Learn More](#)

**Image Source \***

Quickstart  
 Azure Container Registry  
 Other container registries

**Name \***

main

**Azure container registry options**

**Registry \*** StreamlitApp7

**Authentication \***  Managed identity  
 Admin credentials

**Identity \*** (New) ua-id-86d2

**Image \*** streamlitapp\_01

**Tag \*** v1

**Port** 80

**Startup Command** Example: /bin/bash -c echo hello; sleep 10000

[Review + create](#) [< Previous](#) [Next : Networking >](#)

**When using managed identity, image and tag fields will not auto populate. Please manually enter the image and tag below.**

## Create Web App

Basics Database Container Networking Monitor + secure Tags Review + create

Select your preferred source for container images. You can change these settings and other dependencies after creating the app. [Learn more](#)

Enhanced configuration with sidecar support on [Learn More](#)

**Image Source \***

Quickstart  
 Azure Container Registry  
 Other container registries

**Name \***

main

**Azure container registry options**

**Registry \*** StreamlitApp7

**Authentication \***  Managed identity  
 Admin credentials

**Identity \*** (New) ua-id-86d2

**Image \*** streamlitapp\_01

**Tag \*** v1

**Port** 80

**Startup Command** Example: /bin/bash -c echo hello; sleep 10000

[Review + create](#) [< Previous](#) [Next : Networking >](#)

**When using managed identity, image and tag fields will not auto populate. Please manually enter the image and tag below.**

yes! If you must deploy in Qatar Central, then Azure Container Apps (ACA) won't work because it isn't available there yet.

The alternative is Azure App Service for Containers, which supports Qatar Central. 

With Azure App Service, you can:

- Run Docker containers directly
  - Expose a public URL
  - Choose the Qatar Central region
  - Scale easily (manual or automatic)
- 

## ◆ Step 1: Create a Resource Group (if not already)

```
az group create --name streamlitfirstappazcont --location qatarcentral
```

---

## ◆ Step 2: Create an App Service plan

This defines the compute tier:

```
az appservice plan create --name streamlit-app-plan --resource-group  
streamlitfirstappazcont --sku B1 --is-linux
```

- **--sku B1** → Basic tier (cheap, good for testing)
- **--is-linux** → required for Docker containers

---

```
az appservice plan create --name streamlit-app-plan --resource-group  
streamlitfirstappazcont --sku S1 --is-linux --location uaenorth
```

---

## ◆ Step 3: Create the Web App using your Docker image from ACR

```
az webapp create \
--resource-group streamlitfirstappazcont \
--plan streamlit-app-plan \
--name streamlit-webapp-qc \
--deployment-container-image-name streamlitapp7.azurecr.io/streamlitapp_01:v1
```

- --name must be globally unique → e.g., streamlit-webapp-qc
  - --deployment-container-image-name → full ACR image name
- 

## ◆ Step 4: Configure ACR authentication (if needed)

```
az webapp config container set \
--name streamlit-webapp-qc \
--resource-group streamlitfirstappazcont \
--docker-custom-image-name streamlitapp7.azurecr.io/streamlitapp_01:v1 \
--docker-registry-server-url https://streamlitapp7.azurecr.io \
--docker-registry-server-user <ACR_USERNAME> \
--docker-registry-server-password <ACR_PASSWORD>
```

You can get username/password from ACR:

```
az acr credential show --name streamlitapp7
```

---

## ◆ Step 5: Browse your app

```
az webapp show --name streamlit-webapp-qc --resource-group
streamlitfirstappazcont --query defaultHostName
```

Example output:

"streamlit-webapp-qc.qatarcentral.azurewebsites.net"

Open in your browser:

<https://streamlit-webapp-qc.qatarcentral.azurewebsites.net>

---

 Advantages of App Service over ACA here:

- Can deploy in Qatar Central
  - Docker image support
  - Managed scaling & HTTPS
  - Easy to update container image
- 

If you want, I can write a full copy-paste script that:

1. Creates the RG
2. Creates App Service Plan
3. Deploys your Docker image from ACR
4. Outputs the public URL

...all ready for Qatar Central.

Do you want me to do that?

---

=====

**Below is information for SKU , only for understanding purpose**

Absolutely! Azure App Service plans come in tiers that determine CPU, memory, scaling, and features. Here's a clean breakdown:

---

#### ◆ App Service Tiers (Linux / Docker)

Tier	SKU Examples	Memory / CPU	Features
Free (F1)	F1	1 GB memory shared	Only for dev/testing, no custom domain, 60 min/day CPU
Shared (D1)	D1	Shared CPU	Dev/test, low-cost
Basic (B1/B2/B3)	B1: 1 Core, 1.75 GB B2: 2 Core, 3.5 GB B3: 4 Core, 7 GB	Dedicated VM	Supports custom domains, SSL, manual scale 1–3

<b>Standard (S1/S2/S3)</b>	<b>S1: 1 Core, 1.75 GB S2: 2 Core, 3.5 GB S3: 4 Core, 7 GB</b>	<b>Dedicated VM</b>	<b>Auto-scale (manual/metric), staging slots, custom domains</b>
<b>Premium v2 (P1v2/P2v2/P3v2)</b>	<b>P1v2: 1 Core, 3.5 GBP P2v2: 2 Core, 7 GBP P3v2: 4 Core, 14 GB</b>	<b>Dedicated VM</b>	<b>Auto-scale, faster processors, better networking</b>
<b>Premium v3 (P1v3/P2v3/P3v3)</b>	<b>P1v3: 1 Core, 3.5 GBP P2v3: 2 Core, 7 GBP P3v3: 4 Core, 14 GB</b>	<b>Dedicated VM</b>	<b>Latest hardware, faster scaling, VNET integration, staging slots</b>
<b>Isolated (I1/I2/I3)</b>	<b>I1: 1 Core, 3.5 GB</b>	<b>Dedicated, isolated VNET</b>	<b>For highly secure apps, private networking, very high scale</b>

---

◆ Examples for Medium / Higher tiers

Use case	SKU to use
Medium (production, moderate traffic)	<b>S1 (Standard), B2 (Basic medium)</b>
Higher (heavy traffic, high reliability)	<b>S3 (Standard), P1v2 / P2v2 (Premium)</b>
Enterprise / isolated	<b>I1 / I2</b>

---

◆ Command Example for Medium / Higher

**Standard S1 (good production tier):**

```
az appservice plan create \
--name streamlit-app-plan \
--resource-group streamlitfirstappazcont \
--sku S1 \
--is-linux
```

**Premium P1v2 (higher performance):**

```
az appservice plan create \
--name streamlit-app-plan \
--resource-group streamlitfirstappazcont \
--sku P1v2 \
--is-linux
```

---

 Tip:

- B1/B2/B3 → cheaper, limited scale, good for testing / small apps
  - S1/S2/S3 → production-ready, auto-scaling available
  - P1v2+ → high performance, better hardware, faster scaling
- 

If you want, I can prepare a ready-to-run deployment script for your Streamlit Docker app on a Standard (S1) or Premium (P1v2) plan in Qatar Central, so you just copy-paste and it will be live with a public URL.

Do you want me to do that?

---

Perfect! Let's deploy your Streamlit app to Azure Container Apps (ACA) step by step. This is modern, serverless, and will give you a public URL to access your Streamlit app.

---

## ◆ Step 0: Prerequisites

Make sure:

1. Azure CLI is installed ✓ (az --version)
  2. You are logged in ✓ (az login)
  3. Your Docker image is in ACR ✓ (streamlitapp\_01 in streamlitapp7)
  4. You have a resource group ready (streamlit-rg) ✓
- 

## ◆ Step 1: Create a Container Apps environment

The environment is like a network boundary for container apps. Run:

```
az containerapp env create \
--name streamlit-env \
--resource-group streamlitfirstappazcont \
--location qatarcentral
```

- **streamlit-env** → environment name
  - **streamlit-rg** → your resource group
  - **eastus** → region (match your RG location)
- 

## ◆ Step 2: Deploy your container app

We'll deploy the image from your ACR:

```
az containerapp create \
--name streamlit-app \
--resource-group streamlit-rg \
--environment streamlit-env \
--image streamlitapp7.azurecr.io/streamlitapp_01:v1 \
--target-port 8501 \
--ingress 'external' \
--cpu 0.5 \
--memory 1.0Gi
```

## Explanation:

- `--target-port 8501` → Streamlit default port
  - `--ingress 'external'` → gives it a public URL
  - `--cpu 0.5` → half a CPU
  - `--memory 1.0Gi` → 1 GB RAM
- 

## ◆ Step 3: Check deployment status

```
az containerapp show \
--name streamlit-app \
--resource-group streamlit-rg \
--query properties.configuration.ingress.fqdn
```

- This will return the public URL for your app.
- Example output:

"streamlit-app.eastus.azurecontainerapps.io"

---

## ◆ Step 4: Open in browser

Visit the URL in your browser:

<https://<your-app-name>.<region>.azurecontainerapps.io>

You should see your Streamlit Docker app running live! 🎉

---

## ◆ Optional: Scaling

You can scale up/down with:

```
az containerapp update \
--name streamlit-app \
--resource-group streamlit-rg \
--min-replicas 1 \
--max-replicas 3
```

This way, it can auto-scale depending on traffic.

---

## Summary

You now have:

1. Streamlit app containerized 
  2. Image pushed to Azure Container Registry 
  3. Running in Azure Container Apps with public URL 
  4. Can scale resources as needed 
- 

If you want, I can also give you a ready-to-use single script that:

- Builds the Docker image
- Tags it
- Pushes to ACR
- Deploys to ACA

...all in one command sequence.

Do you want me to do that next?

**StreamlitApp7** Container registry

Test if this ACR is accessible from connected AKS Check AKS clusters pulling from this registry Run Canipull to validate access to this ACR

Search Move Delete

Access control (IAM)

Tags Quick start Resource visualizer Events Settings

Microsoft Defender for Cloud Properties Locks Services

Repositories Webhooks Geo-replications Tasks Connected registries Cache

Repository permissions Tokens

Essentials

Resource group (move) : StreamlitFirstAppAzCont Location : Qatar Central Subscription (move) : Azure subscription 1 Subscription ID : 76002742-0a9e-44d5-9d98-143f3a2bf30f Soft delete (Preview) : Disabled Tags (edit) : Add tags

Login server : streamlitapp7.azurecr.io Creation date : 06/01/2026, 22:01 GMT+3 Provisioning state : Succeeded Pricing plan : Standard Domain name label scope : Unsecure

Get started Monitoring Capabilities (9) Tutorials

*Click on this*

Simplify container lifecycle management

Container registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments. [Learn more](#)

**Push a Container Image**  
Get instructions on how to store your container images in your container registry. [Learn more about container images](#)

**Manage your access controls**  
Secure your container registry by configuring how users interact with your container registry

**Deploy a container image**  
Explore the latest ways to cont streamline artifact deployment Registry - or choose one and g

**Push an image** **Manage access controls** **Deploy an image**

The screenshot shows the Azure Container Registry (ACR) interface for the 'StreamlitApp7' resource. On the left, a sidebar lists various services: Access control (IAM), Tags, Quick start, Resource visualizer, Events, Settings, Microsoft Defender for Cloud, Properties, Locks, Services, Repositories (which is highlighted with a yellow circle), Webhooks, Geo-replications, Tasks, Connected registries, Cache, Repository permissions, and Tokens. The main content area displays 'Essentials' information such as Resource group, Location, Subscription, Soft delete status, and Tags. It also features a 'Get started' tab and navigation links for Monitoring, Capabilities (9), and Tutorials. A large central section is titled 'Simplify container lifecycle management' and provides links for Push a Container Image, Manage your access controls, and Deploy a container image. A handwritten note 'Click on this' is overlaid on the Services section.