

I'll help you deploy your Streamlit app to Azure Web App. Here's a detailed step-by-step process:

Prerequisites

- Azure account with an active subscription
- Your Streamlit app in VS Code
- Azure CLI installed (or use Azure Portal)
- Git installed

Step 1: Prepare Your Streamlit App

First, ensure your project has the proper structure:

```
your-streamlit-app/
|
├── app.py (or your main Streamlit file)
├── requirements.txt
├── startup.sh (optional but recommended)
└── .gitignore
```

Create/Update requirements.txt:

```
streamlit
pandas
numpy
# Add all your other dependencies
```

Create startup.sh:

```
#!/bin/bash
python -m streamlit run app.py --server.port 8000 --server.address 0.0.0.0
```

Make it executable locally:

```
chmod +x startup.sh
```

Step 2: Create Azure Web App

Option A: Using Azure Portal

1. Go to Azure Portal (<https://portal.azure.com>)

2. **Create a Web App:**
 - Click "Create a resource"
 - Search for "Web App" and select it
 - Click "Create"
3. **Configure Basic Settings:**
 - **Subscription:** Select your subscription
 - **Resource Group:** Create new or select existing
 - **Name:** Choose a unique name (e.g., my-streamlit-app)
 - **Publish:** Code
 - **Runtime stack:** Python 3.11 (or your preferred version)
 - **Operating System:** Linux
 - **Region:** Choose closest to your users
 - **Pricing Plan:** Choose based on your needs (B1 Basic is good for testing)
4. **Click "Review + Create" then "Create"**

Option B: Using Azure CLI

```
# Login to Azure
az login

# Create a resource group
az group create --name myResourceGroup --location eastus

# Create an App Service plan
az appservice plan create --name myAppServicePlan --resource-group myResourceGroup
--sku B1 --is-linux

# Create the web app
az webapp create --resource-group myResourceGroup --plan myAppServicePlan --name
my-streamlit-app --runtime "PYTHON:3.11"
```

Step 3: Configure Application Settings

In Azure Portal, go to your Web App:

1. **Configuration → Application settings**
2. Add the following settings:
 - SCM_DO_BUILD_DURING_DEPLOYMENT = true
 - WEBSITES_PORT = 8000
3. **Configuration → General settings:**
 - **Startup Command:** startup.sh (or bash startup.sh)

Step 4: Deploy Your App

Option A: Local Git Deployment

```

# In your project directory
git init
git add .
git commit -m "Initial commit"

# Get deployment credentials from Azure Portal
# Go to Deployment Center → Local Git → Copy the Git URL

# Add Azure remote
git remote add azure <your-git-clone-url>

# Push to Azure
git push azure master

```

Option B: GitHub Actions (Recommended)

1. **Push your code to GitHub**
2. **In Azure Portal:**
 - o Go to your Web App → Deployment Center
 - o Select "GitHub" as source
 - o Authorize and select your repository
 - o Azure will automatically create a GitHub Actions workflow
3. **The workflow file** (.github/workflows/azure-webapps-python.yml) will be created automatically

Option C: VS Code Azure Extension

1. **Install Azure App Service extension** in VS Code
2. **Sign in to Azure** (Ctrl+Shift+P → "Azure: Sign In")
3. **Right-click your project folder** → "Deploy to Web App"
4. **Select your subscription and web app**

Option D: ZIP Deployment

```

# Create a zip file
zip -r app.zip . -x "* .git*" "* __pycache__ *" "* .pyc"

# Deploy using Azure CLI
az webapp deployment source config-zip --resource-group myResourceGroup --name
my-streamlit-app --src app.zip

```

Step 5: Monitor Deployment

1. **Check deployment logs:**
 - o Azure Portal → Your Web App → Deployment Center → Logs

2. **View application logs:**
 - Azure Portal → Your Web App → Log stream

Step 6: Access Your App

Your app will be available at:

<https://my-streamlit-app.azurewebsites.net>

Troubleshooting

If the app doesn't start:

1. **Check logs:**

```
az webapp log tail --name my-streamlit-app --resource-group myResourceGroup
```

2. **Common issues:**

- Incorrect port configuration (ensure WEBSITES_PORT = 8000)
- Missing dependencies in requirements.txt
- Startup command not configured properly
- Python version mismatch

3. **Enable detailed logging:**

- Go to App Service logs → Enable Application Logging

Optional: Custom Domain & SSL

1. **Add custom domain:**

- Azure Portal → Your Web App → Custom domains

2. **SSL certificate:**

- Azure Portal → Your Web App → TLS/SSL settings

Sample Project Structure

Here's a complete example:

app.py:

```
import streamlit as st

st.title("My Streamlit App on Azure")
st.write("Hello from Azure Web App!")

name = st.text_input("Enter your name:")
```

```
if name:  
    st.write(f"Hello, {name}!")
```

requirements.txt:

```
streamlit==1.29.0
```

startup.sh:

```
#!/bin/bash  
python -m streamlit run app.py --server.port 8000 --server.address 0.0.0.0
```

Would you like help with any specific step or troubleshooting a particular issue?

Below Azure Command Line

```
# Login to Azure
```

```
az login
```

```
# Create a resource group
```

```
az group create --name myResourceGroup --location eastus
```

```
# Create an App Service plan
```

```
az appservice plan create --name myAppServicePlan --resource-group myResourceGroup  
--sku B1 --is-linux
```

```
# Create the web app
```

```
az webapp create --resource-group myResourceGroup --plan myAppServicePlan --name  
my-streamlit-app --runtime "PYTHON:3.11"
```

Not quite! If you only run `zip -r app.zip .`, it will work but it's not optimal. Let me explain what you need:

What You're Missing: startup.sh

Even though you only have `app.py` and `requirements.txt`, **you still need to create the `startup.sh` file** for Azure to know how to run your Streamlit app.

Here's What to Do:

Step 1: Create startup.sh

In the same folder as your `app.py`, create a new file called `startup.sh` with this content:

```
#!/bin/bash  
  
python -m streamlit run app.py --server.port 8000 --server.address 0.0.0.0
```

Note: If your main file is named something else (like `main.py` or `streamlit_app.py`), replace `app.py` with your actual filename.

Step 2: Your Project Structure Should Look Like:

```
your-folder/  
    └── app.py  
    └── requirements.txt  
    └── startup.sh
```

Step 3: Create the ZIP file

Now you can create the zip file. The simple command `zip -r app.zip .` will work, but it's better to exclude unnecessary files:

If you have no .git folder or cache files:

```
zip -r app.zip .
```

Better approach (cleaner):

```
zip -r app.zip app.py requirements.txt startup.sh
```

This explicitly includes only the files you need.

Step 4: Deploy to Azure

```
az webapp deployment source config-zip --resource-group myResourceGroup --name my-streamlit-app --src app.zip
```

Step 5: Configure Azure Web App Settings

Important! After deploying, go to Azure Portal:

1. Navigate to your Web App
2. Go to **Configuration** → **General settings**
3. Set **Startup Command** to: `startup.sh`
4. Go to **Application settings** and add:
 - Name: WEBSITES_PORT, Value: 8000
5. Click **Save**

Summary

So to answer your question: **No**, just `zip -r app.zip .` is not sufficient because you need the `startup.sh` file to tell Azure how to run your Streamlit app. Without it, Azure won't know what command to execute.

Create the `startup.sh` file first, then create your zip with all three files!

Below is the command executed for creating Zip folder for web app deployment

```
(VirEnv) datascientist@Manjunaths-MacBook-Air ZipCode % zip -r app.zip app.py  
requirements.txt startup.sh adding: app.py (deflated 27%) adding: requirements.txt (stored  
0%) adding: startup.sh (deflated 14%) , kindly confirm
```

Perfect! 

Your zip file has been created successfully. The output confirms:

-  app.py - added (deflated 27%)
-  requirements.txt - added (stored 0%)
-  startup.sh - added (deflated 14%)

All three essential files are now packed in app.zip.

Next Steps:

1. Deploy to Azure using Azure CLI:

```
az webapp deployment source config-zip --resource-group <your-resource-group-name>  
--name <your-webapp-name> --src app.zip
```

Replace:

- <your-resource-group-name> with your actual resource group name
- <your-webapp-name> with your actual web app name

2. After deployment, configure Azure Portal:

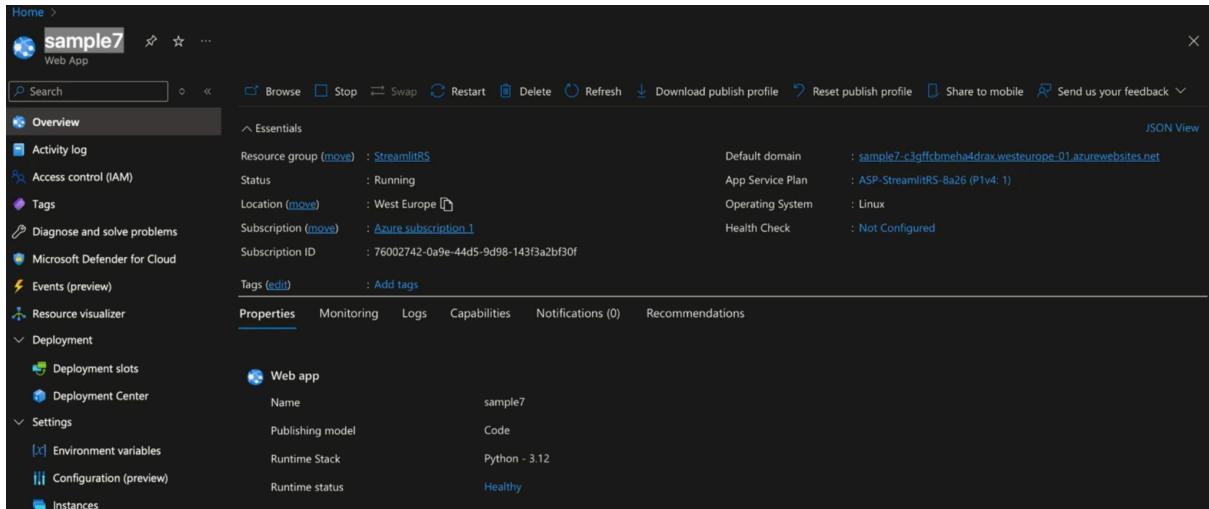
1. Go to Azure Portal → Your Web App
2. **Configuration → General settings**
 - Startup Command: startup.sh
3. **Configuration → Application settings** → New application setting
 - Name: WEBSITES_PORT
 - Value: 8000
4. Click **Save** and wait for the app to restart

3. Access your app:

<https://<your-webapp-name>.azurewebsites.net>

You're all set! Let me know if you need help with the deployment command or encounter any issues. 

Below is the web app detail from the Azure based on that



The screenshot shows the Azure portal interface for a web app named 'sample7'. The left sidebar has sections like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Resource visualizer, Deployment (Deployment slots, Deployment Center), Settings (Environment variables, Configuration (preview), Instances), and a JSON View button. The main content area is titled 'Overview' under 'Essentials'. It lists the following details:

Setting	Value
Resource group (move)	StreamlitRS
Status	Running
Location (move)	West Europe
Subscription (move)	Azure subscription 1
Subscription ID	76002742-0a9e-44d5-9d98-143f3a2bf30f
Tags (edit)	Add tags
Properties	Monitoring, Logs, Capabilities, Notifications (0), Recommendations
Web app	Name: sample7, Publishing model: Code, Runtime Stack: Python - 3.12, Runtime status: Healthy

Based on your Azure Web App setup, here's the deployment command for your specific configuration:

```
az webapp deployment source config-zip --resource-group StreamlitRS --name sample7  
--src app.zip
```

Breaking it down:

- `--resource-group StreamlitRS` (your resource group name from the screenshot)
- `--name sample7` (your web app name)
- `--src app.zip` (your zip file)

After Deployment, Configure These Settings:

1. Go to Configuration → General settings

- Click on **Settings** (left menu) → **Configuration**
- Go to **General settings** tab
- Set **Startup Command** to: `startup.sh`
- Click **Save**

2. Add Application Setting

- Still in **Configuration** → **Application settings** tab
- Click **+ New application setting**
- Name: WEBSITES_PORT
- Value: 8000
- Click **OK**, then **Save**

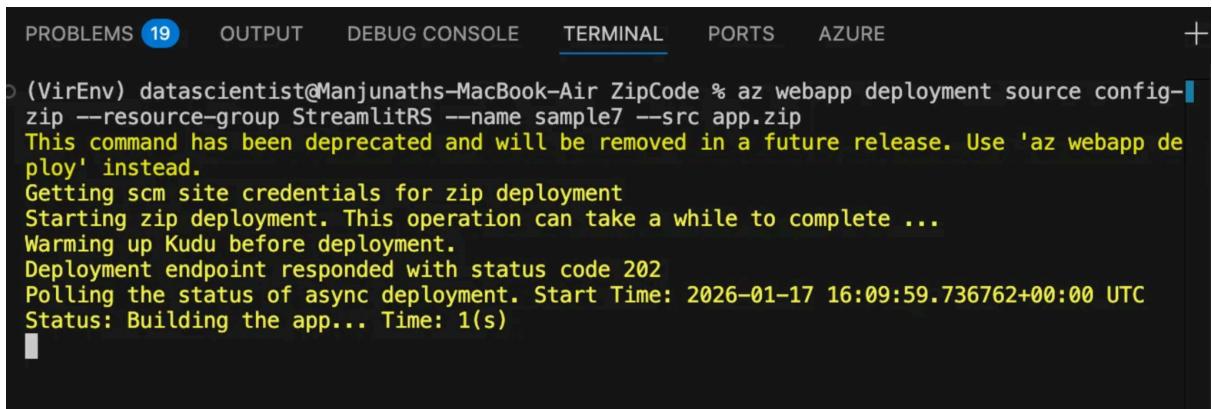
Your App URL:

Once deployed and configured, access your app at:

<https://sample7-c3gjfcbmcha4drax.westeurope-01.azurewebsites.net>

Run the deployment command from your terminal (make sure you're in the same directory as your `app.zip` file), and let me know if you encounter any issues! 

Below shows the deployment in web app using Azure command line



The screenshot shows a terminal window with the following output:

```
PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE +  
○ (VirEnv) datascientist@Manjunaths-MacBook-Air ZipCode % az webapp deployment source config-zip --resource-group StreamlitRS --name sample7 --src app.zip  
This command has been deprecated and will be removed in a future release. Use 'az webapp deploy' instead.  
Getting scm site credentials for zip deployment  
Starting zip deployment. This operation can take a while to complete ...  
Warming up Kudu before deployment.  
Deployment endpoint responded with status code 202  
Polling the status of async deployment. Start Time: 2026-01-17 16:09:59.736762+00:00 UTC  
Status: Building the app... Time: 1(s)
```

Great! Your deployment is in progress! 🎉

What's happening:

1. **Deployment started successfully** - Status code 202 means accepted
2. **Building the app** - Azure is currently building your Streamlit application
3. **In Progress** - "Status: Building the app... Time: 1(s)"

What to do now:

Just wait! The deployment process typically takes 2-5 minutes. You'll see status updates in the terminal.

Expected sequence:

- Building the app... (current)
- Installing dependencies from requirements.txt
- Deployment successful

Note about the warning: The yellow warning about the command being deprecated is just informational. The deployment will still work fine. In future, you can use `az webapp deploy` instead, but for now, it's working correctly.

Once deployment completes:

IMPORTANT: Configure Azure Settings

Go to Azure Portal and complete these steps:

1. Set Startup Command:

- Settings → Configuration → General settings
- Startup Command: `startup.sh`
- Click Save

2. Add Port Setting:

- Configuration → Application settings
- Click "+ New application setting"
- Name: WEBSITES_PORT
- Value: 8000
- Click OK, then Save

Without these settings, your app won't start properly!

Wait for the terminal to show "Deployment successful", then configure those settings and access your app at:

<https://sample7-c3gjfcbmcha4drax.westeurope-01.azurewebsites.net>

Let me know when it completes or if you see any errors! 

After deployment i was checking whether it is working or not

```
Deployment endpoint responded with status code 202
Polling the status of async deployment. Start Time: 2026-01-17 16:09:59.736762+00:00 UTC
Status: Building the app... Time: 1(s)
Status: Building the app... Time: 18(s)
Status: Building the app... Time: 35(s)
Status: Build successful. Time: 52(s)
Status: Starting the site... Time: 68(s)
Status: Starting the site... Time: 84(s)
Status: Starting the site... Time: 100(s)
Status: Starting the site... Time: 115(s)
Status: Starting the site... Time: 132(s)
Status: Starting the site... Time: 148(s)
Status: Starting the site... Time: 165(s)
Status: Starting the site... Time: 181(s)
Status: Starting the site... Time: 197(s)
Status: Starting the site... Time: 213(s)
Status: Starting the site... Time: 229(s)
Status: Starting the site... Time: 245(s)
Status: Starting the site... Time: 261(s)
Status: Starting the site... Time: 277(s)
Status: Starting the site... Time: 304(s)
Status: Starting the site... Time: 321(s)
Status: Starting the site... Time: 338(s)
Status: Starting the site... Time: 353(s)
Status: Starting the site... Time: 369(s)
Status: Starting the site... Time: 386(s)
Status: Starting the site... Time: 402(s)
Status: Starting the site... Time: 418(s)
Status: Starting the site... Time: 434(s)
Status: Starting the site... Time: 451(s)
Status: Starting the site... Time: 466(s)
```

The deployment is taking longer than usual to start (currently at 466 seconds / ~7.7 minutes). This often indicates that the app needs configuration to start properly.

Let's configure Azure NOW while it's still attempting to start:

Go to Azure Portal and configure these settings:

1. Set Startup Command:

- Azure Portal → sample7 → Settings → **Configuration**
- Click **General settings** tab
- In **Startup Command** field, enter: startup.sh
- Click **Save** at the top

2. Add Port Setting:

- Still in **Configuration** → **Application settings** tab
- Click **+ New application setting**
- Name: WEBSITES_PORT
- Value: 8000
- Click **OK**
- Click **Save** at the top

3. Restart the app:

- After saving, click **Restart** button at the top of the Overview page
-

If the deployment times out or fails:

You can also check the logs to see what's happening:

Option 1: Check Application Logs

- Azure Portal → sample7 → **Monitoring** → **Log stream**

Option 2: Enable detailed logging

- Settings → **Configuration** → **Application settings**
 - Add: SCM_DO_BUILD_DURING_DEPLOYMENT = true (if not already there)
-

The long "Starting the site" time suggests the app is trying to start but doesn't know how. Once you configure the startup command and port, it should work properly.

Configure those settings now and then restart the app! Let me know what happens. 

When deployment failed , then how to handle it is as below

```
(VirEnv) datascientist@Manjunaths-MacBook-Air ZipCode % az webapp deployment source config--zip --resource-group StreamlitRS --name sample7 --src app.zip
Status: Starting the site... Time: 277(s)
Status: Starting the site... Time: 304(s)
Status: Starting the site... Time: 321(s)
Status: Starting the site... Time: 338(s)
Status: Starting the site... Time: 353(s)
Status: Starting the site... Time: 369(s)
Status: Starting the site... Time: 386(s)
Status: Starting the site... Time: 402(s)
Status: Starting the site... Time: 418(s)
Status: Starting the site... Time: 434(s)
Status: Starting the site... Time: 451(s)
Status: Starting the site... Time: 466(s)
Status: Starting the site... Time: 483(s)
Status: Starting the site... Time: 498(s)
Status: Starting the site... Time: 515(s)
Status: Starting the site... Time: 531(s)
Status: Starting the site... Time: 548(s)
Status: Starting the site... Time: 564(s)
Status: Starting the site... Time: 580(s)
Status: Starting the site... Time: 598(s)
Status: Starting the site... Time: 614(s)
Status: Starting the site... Time: 630(s)
Status: Starting the site... Time: 646(s)
Status: Site failed to start. Time: 663(s)
Deployment failed because the site failed to start within 10 mins.
InprogressInstances: 0, SuccessfulInstances: 0, FailedInstances: 1
Error: Deployment for site 'sample7' with DeploymentId '22a61acb-3d2e-4692-a7dc-e5af6047fa41' failed because the worker process failed to start within the allotted time.
Please check the runtime logs for more info: https://sample7-c3gffcbmeha4drax.scm.westeurope-01.azurewebsites.net/api/logs/docker
```

The deployment failed because the site couldn't start within 10 minutes. This is expected because we haven't configured the startup command yet.

Here's what to do now:

Step 1: Configure Azure Settings (CRITICAL)

Go to Azure Portal and configure these settings:

1. Set Startup Command:

- Go to: **sample7** → **Settings** → **Configuration**
- Click **General settings** tab
- **Startup Command:** Enter `startup.sh`
- Click **Save**

2. Add Port Configuration:

- Still in **Configuration** → **Application settings** tab
- Click **+ New application setting**
- **Name:** WEBSITES_PORT
- **Value:** 8000
- Click **OK**, then **Save**

3. Add Build Setting:

- Still in **Application settings**
- Click **+ New application setting**
- **Name:** SCM_DO_BUILD_DURING_DEPLOYMENT
- **Value:** true
- Click **OK**, then **Save**

Step 2: Restart the Web App

After saving all settings:

- Go to **Overview** page
- Click **Restart** button at the top
- Wait 2-3 minutes for the app to start

Step 3: Check if it's working

Visit your app URL:

<https://sample7-c3gfcbmcha4drax.westeurope-01.azurewebsites.net>

If it still doesn't work after restart:

Check the logs to see what's happening:

Option 1: Log Stream

```
az webapp log tail --name sample7 --resource-group StreamlitRS
```

Option 2: In Azure Portal

- sample7 → **Monitoring** → **Log stream**

The files are deployed correctly, but Azure just needs to know HOW to start your Streamlit app. Once you configure those three settings and restart, it should work!

Let me know after you've configured the settings and restarted! 

```
(VirEnv) dataScientist@Manjunath's-MacBook-Air ZipCode % az webapp log tail --name sample7 -rresource-group StreamLitRS
2026-01-17T16:27:27 Welcome, you are now connected to log-streaming service.
Starting Log Tail -n 10 of existing logs ----

/appsvctmp/volatile/logs/runtime/container.log
2026-01-17T16:25:51.0223393Z Using packages from virtual environment antenv located at /tmp/8de55e2e02237c4/antenv.
2026-01-17T16:25:51.0223814Z Updated PYTHONPATH to '/opt/startup/app_logs:/tmp/8de55e2e02237c4/antenv/lib/python3.12/site-packages'
2026-01-17T16:25:51.6994699Z
2026-01-17T16:25:51.6995231Z Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.
2026-01-17T16:25:51.6995289Z
2026-01-17T16:25:51.8219309Z
2026-01-17T16:25:51.8219785Z You can now view your Streamlit app in your browser.
2026-01-17T16:25:51.8219833Z
2026-01-17T16:25:51.8219865Z URL: http://0.0.0.0:8000
2026-01-17T16:25:51.8219894Z
Ending Log Tail of existing logs ----
Starting Live Log Stream ----
2026-01-17T16:28:27 No new trace in the past 1 min(s).
2026-01-17T16:29:27 No new trace in the past 2 min(s).
2026-01-17T16:30:27 No new trace in the past 3 min(s).
2026-01-17T16:31:27 No new trace in the past 4 min(s).
2026-01-17T16:32:27 No new trace in the past 5 min(s).
2026-01-17T16:33:27 No new trace in the past 6 min(s).
2026-01-17T16:34:27 No new trace in the past 7 min(s).
```

 Excellent! Your Streamlit app is now running successfully! 

The logs show:  "You can now view your Streamlit app in your browser."  Streamlit is running on port 8000  No errors in the logs

Access Your App

Your Streamlit app is now live at:

<https://sample7-c3gjfcmbmcha4drax.westeurope-01.azurewebsites.net>

Open this URL in your browser to see your app!

What Happened

After you configured the Azure settings (startup command and port), Azure successfully:

1.  Installed your Python dependencies from requirements.txt
2.  Executed the startup.sh script
3.  Started Streamlit on port 8000

-
4.  Made your app accessible via the public URL
-

Next Steps (Optional)

To stop the log stream:

- Press Ctrl + C in your terminal

To monitor your app later:

```
az webapp log tail --name sample7 --resource-group StreamlitRS
```

To update your app in the future:

1. Make changes to your code locally
 2. Create a new zip file: `zip -r app.zip app.py requirements.txt startup.sh`
 3. Redeploy: `az webapp deployment source config-zip --resource-group StreamlitRS --name sample7 --src app.zip`
-

Congratulations! Your Streamlit app is successfully deployed on Azure Web App! 🚀

Try accessing the URL and let me know if it works! 😊

Successfully deployed

```
(VirEnv) datascientist@Manjunaths-MacBook-Air ZipCode % az webapp log tail --name sample7 -  
-resource-group StreamlitRS  
2026-01-17T16:27:27 Welcome, you are now connected to log-streaming service.  
Starting Log Tail -n 10 of existing logs ---  
  
/appsvctmp/volatile/logs/runtime/container.log  
2026-01-17T16:25:51.0223393Z Using packages from virtual environment antenv located at /tmp/  
/8de55e2e02237c4/antenv.  
2026-01-17T16:25:51.0223814Z Updated PYTHONPATH to '/opt/startup/app_logs:/tmp/8de55e2e0223  
7c4/antenv/lib/python3.12/site-packages'  
2026-01-17T16:25:51.6994699Z  
2026-01-17T16:25:51.6995231Z Collecting usage statistics. To deactivate, set browser.gather  
UsageStats to false.  
2026-01-17T16:25:51.6995289Z  
2026-01-17T16:25:51.8219309Z  
2026-01-17T16:25:51.8219785Z You can now view your Streamlit app in your browser.  
2026-01-17T16:25:51.8219833Z  
2026-01-17T16:25:51.8219865Z URL: http://0.0.0.0:8000  
2026-01-17T16:25:51.8219894Z  
Ending Log Tail of existing logs ---  
Starting Live Log Stream ---
```

Great! The logs show that **Streamlit is running successfully!** ✓

The message says:

- ✓ "You can now view your Streamlit app in your browser"
- ✓ Running on port 8000
- ✓ No errors

The Issue

The problem is that Safari (or your browser) is having trouble connecting. This could be because:

1. **The app just restarted** - it may need a minute to fully respond to external requests
2. **DNS propagation delay** - sometimes it takes a moment for the URL to resolve properly

Solutions to Try:

Option 1: Wait and Refresh (Most likely to work)

1. **Wait 2-3 minutes** for the app to fully initialize
2. **Hard refresh** your browser:

- Safari/Chrome on Mac: Cmd + Shift + R
- Or clear your browser cache

Try accessing the URL again:

<https://sample7-c3gjfcbmcha4drax.westeurope-01.azurewebsites.net>

3.

Option 2: Try a Different Browser

- Open the URL in Chrome, Firefox, or Edge
- Sometimes Safari has caching issues

Option 3: Check Azure Portal

1. Go to Azure Portal → **sample7** → **Overview**
2. Look at the **Status** - it should say "Running"
3. Click on the URL directly from the Azure Portal

Option 4: Restart the App

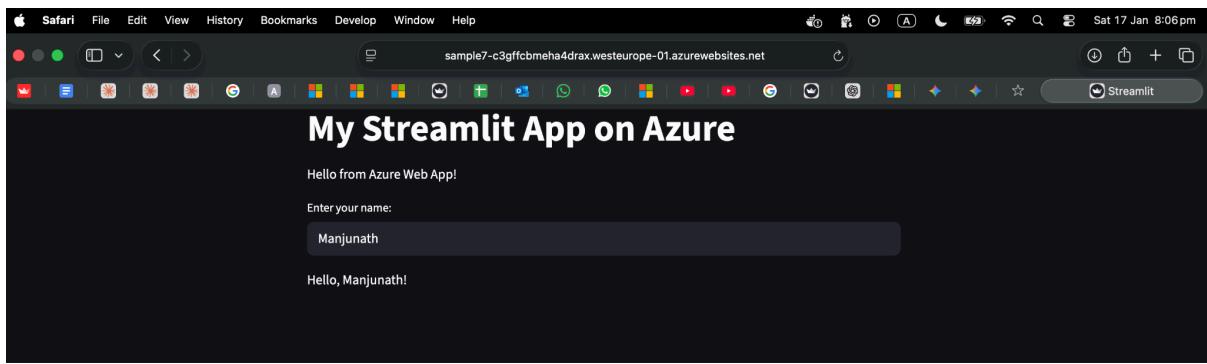
If it still doesn't work after 5 minutes:

```
az webapp restart --name sample7 --resource-group StreamlitRS
```

Then wait 2-3 minutes and try the URL again.

Most likely, you just need to wait 1-2 minutes and refresh your browser. The Streamlit app is definitely running based on the logs!

Try refreshing now and let me know if it works! 



🎉 Awesome! I'm so glad it's working now! 🎉

Congratulations on successfully deploying your first Streamlit app to Azure Web App! 🚀

Quick Reference for Future Updates:

Whenever you want to update your app:

1. **Make changes** to your code locally

2 . **Create new zip:**

```
zip -r app.zip app.py requirements.txt startup.sh
```

3. **Deploy:**

```
az webapp deployment source config-zip --resource-group StreamlitRS --name sample7 --src app.zip
```

4. **Wait 2-3 minutes** and refresh your browser

Your Live App URL:

<https://sample7-c3gjfcmbmcha4drax.westeurope-01.azurewebsites.net>

If you ever need help with:

- Adding more features to your Streamlit app
- Deploying other apps
- Troubleshooting issues
- Scaling your app

Great question! You have several options for adding authentication to your Streamlit app on Azure. Let me explain both approaches:

Option 1: Azure AD Authentication (Microsoft Company Login) - Recommended

This uses Azure's built-in authentication without changing your code.

Steps:

1. Enable Authentication in Azure Portal:

- Go to **sample7** → **Settings** → **Authentication**
- Click **Add identity provider**
- Select **Microsoft**
- Configure:
 - **App registration type:** Create new app registration
 - **Supported account types:**
 - Choose "Current tenant only" (for your company only)
 - Or "Any Azure AD directory" (for any Microsoft account)
 - **Restrict access:** Check "Require authentication"
 - **Unauthenticated requests:** Select "HTTP 302 Found redirect"
- Click **Add**

2. Access Control:

After enabling, only users who sign in with their Microsoft account can access your app.

3. Get User Info in Streamlit (Optional):

You can access the authenticated user's information in your app:

```
import streamlit as st
```

```
import os

# Get authenticated user info from Azure headers

user_name = os.environ.get('X-MS-CLIENT-PRINCIPAL-NAME', 'Unknown')

user_id = os.environ.get('X-MS-CLIENT-PRINCIPAL-ID', 'Unknown')

st.title("Welcome to the App")

st.write(f"Hello, {user_name}!")
```

Option 2: Custom Username/Password Authentication

If you want custom authentication within your Streamlit app:

Simple Version (Using streamlit-authenticator):

1. Update requirements.txt:

```
streamlit
streamlit-authenticator
```

2. Create a new app.py with authentication:

```
import streamlit as st

import streamlit_authenticator as stauth

import yaml

from yaml.loader import SafeLoader

# Configuration

config = {

    'credentials': {
```

```
'usernames': {  
    'jsmith': {  
        'name': 'John Smith',  
        'password': '$2b$12$...hashed_password...' # Use hashed passwords!  
    },  
    'rbriggs': {  
        'name': 'Rebecca Briggs',  
        'password': '$2b$12$...hashed_password...'  
    }  
},  
  
'cookie': {  
    'name': 'streamlit_auth',  
    'key': 'your-secret-key-here', # Change this!  
    'expiry_days': 30  
}  
}  
  
  
# Create authenticator  
  
authenticator = stauth Authenticate(  
    config['credentials'],  
    config['cookie']['name'],  
    config['cookie']['key'],  
    config['cookie']['expiry_days']  
)
```

```

# Login widget

name, authentication_status, username = authenticator.login('Login', 'main')

if authentication_status:

    authenticator.logout('Logout', 'main')

    st.write(f'Welcome *{name}*')

    st.title('Your Streamlit App')


# Your app code here

st.write("This is your protected content!")


elif authentication_status == False:

    st.error('Username/Password is incorrect')

elif authentication_status == None:

    st.warning('Please enter your username and password')

```

3. Generate Hashed Passwords:

Create a separate script `generate_password.py`:

```
import streamlit_authenticator as stauth
```

```

# Generate hashed password

password = "your_password_here"

hashed = stauth.Hasher([password]).generate()[0]

print(f"Hashed password: {hashed}")

```

Run it locally:

```
python generate_password.py
```

Copy the hashed password into your config.

Option 3: Database-backed Authentication

For a more robust solution with user management:

1. Update requirements.txt:

```
streamlit
```

```
bcrypt
```

```
sqlite3 # Or use Azure SQL Database
```

2. Create authentication with database:

```
import streamlit as st
```

```
import sqlite3
```

```
import bcrypt
```

```
import os
```

```
# Initialize database
```

```
def init_db():
```

```
    conn = sqlite3.connect('users.db')
```

```
    c = conn.cursor()
```

```
    c.execute("CREATE TABLE IF NOT EXISTS users
```

```
        (username TEXT PRIMARY KEY, password TEXT, name TEXT)")
```

```
    conn.commit()
```

```
    conn.close()
```

```
# Verify login

def verify_user(username, password):
    conn = sqlite3.connect('users.db')
    c = conn.cursor()
    c.execute("SELECT password FROM users WHERE username=?", (username,))
    result = c.fetchone()
    conn.close()

    if result and bcrypt.checkpw(password.encode('utf-8'), result[0]):
        return True
    return False

# Add new user (for admin use)

def add_user(username, password, name):
    hashed = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
    conn = sqlite3.connect('users.db')
    c = conn.cursor()
    try:
        c.execute("INSERT INTO users VALUES (?, ?, ?)", (username, hashed, name))
        conn.commit()
    except:
        return False
    finally:
        conn.close()
```

```
# Main app

init_db()

# Session state for login

if 'logged_in' not in st.session_state:
    st.session_state.logged_in = False

if not st.session_state.logged_in:
    st.title("Login")

    username = st.text_input("Username")
    password = st.text_input("Password", type="password")

    if st.button("Login"):
        if verify_user(username, password):
            st.session_state.logged_in = True
            st.session_state.username = username
            st.rerun()
        else:
            st.error("Invalid credentials")
    else:
        st.title("Your Streamlit App")
        st.write(f"Welcome, {st.session_state.username}!")

if st.button("Logout"):
    st.session_state.logged_in = False
    st.rerun()
```

```
# Your protected app content here  
st.write("This is your protected content!")
```

Comparison:

Feature	Azure AD (Microsoft)	streamlit-authenticator	Custom DB
Setup Complexity	Easy (no code)	Medium	Complex
User Management	Azure Portal	Hardcoded/Config	Full control
Cost	Free (included)	Free	Free
Best For	Company users	Small team	Custom needs
SSO Support	Yes	No	No
MFA Support	Yes	No	Optional

My Recommendation:

1. **For company/enterprise use:** Use **Azure AD Authentication** (Option 1) - it's the most secure, easiest to maintain, and provides SSO
2. **For small team/simple use:** Use **streamlit-authenticator** (Option 2)
3. **For custom requirements:** Build your own with database (Option 3)

Would you like me to help you implement any of these options? Let me know which approach fits your needs best! 🔒