CDW Documentation

Azure Webapp Deployment

Azure Webapp Deployment

Final File Contents (cloudadvisor.zip)

The archive used for deployment contained:

- `app.py` – Flask app connecting to Azure OpenAI, handling routing and logic
- `requirements.txt` – Dependencies: `flask` and `openai`
- `templates/index.html` – Web interface to enter questions
- `static/placeholder.txt` – Keeps the `static/` folder intact (required by some deployments)
- `.DS_Store` – Ignorable macOS metadata file

What app.py Does

- Reads environment variables `OPENAI_API_KEY` and `OPENAI_API_ENDPOINT`
- Initializes the OpenAI client using `openai>=1.0.0` standards
- Serves an HTML form from `templates/index.html`
- Responds to POST requests at `/ask` by querying the OpenAI API with a system prompt and user message
- Returns the AI's response back to the frontend for display

⚙ Required Azure Steps (End-to-End)

1. Resource Group & App Service Plan

```
bashCopyEditaz group create --name don-test-rg --location eastus
az appservice plan create --name cloudadvisor-plan --resource-group don-test-rg --sku B1 --is-linux
```

2. Web App (Python on Linux)

```
bashCopyEditaz webapp create \
  --name cloudadvisor \
  --resource-group don-test-rg \
  --plan cloudadvisor-plan \
  --runtime "PYTHON|3.10"
```

3. Set Required App Settings

```
bashCopyEditaz webapp config appsettings set \
```

```
  --name cloudadvisor \
  --resource-group don-test-rg \
  --settings \
  OPENAI_API_KEY="your-key" \
  OPENAI_API_ENDPOINT="https://your-endpoint.openai.azure.com/" \
  SCM_DO_BUILD_DURING_DEPLOYMENT=true \
  WEBSITES_ENABLE_APP_SERVICE_STORAGE=false
```

4. Restrict IP Access (Optional)

Ensure only your IP (e.g., `162.231.206.200/32`) can access the site using Bicep or:

```
bashCopyEditaz webapp config access-restriction add \
  --resource-group don-test-rg \
  --name cloudadvisor \
  --rule-name AllowMyIP \
  --action Allow \
  --ip-address 162.231.206.200/32 \
  --priority 100
```

Deploy the App

```
bashCopyEditaz webapp deploy \
  --resource-group don-test-rg \
  --name cloudadvisor \
  --src-path ./cloudadvisor.zip \
  --type zip
```

Startup & Troubleshooting

Start App If Not Running Automatically:

```
bashCopyEditaz webapp restart --name cloudadvisor --resource-group don-test-rg
```

Common Debugging Tools:

- Kudu Console: https://cloudadvisor.scm.azurewebsites.net/DebugConsole
- Logs: `LogFiles/application`
- App Settings: `az webapp config appsettings list`

Azure OpenAI Network Access

If getting a `403 - Access denied due to Virtual Network/Firewall rules` error:

- Go to Azure OpenAI resource
- Under **Networking**, choose **Allow access from: All Networks**
- Click **Save**

Final Notes

This final architecture is secure (IP restricted), scalable, and portable. The problems you faced—deployment timing, startup configs, OpenAI SDK changes, and environmental variables—are now all resolved.

app.py

```
from flask import Flask, request, jsonify, render_template
from openai import AzureOpenAI
import os

app = Flask(__name__, template_folder="templates")

# These must be set in environment variables or app settings
api_key = os.getenv("OPENAI_API_KEY")
endpoint = os.getenv("OPENAI_API_ENDPOINT")  # No trailing slash
deployment_name = "gpt-4.1"  # Replace with your deployed model name

# Create AzureOpenAI client
client = AzureOpenAI(
    api_key=api_key,
    api_version="2023-12-01-preview",
    azure_endpoint=endpoint  # This must not have trailing '/'
)

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/ask", methods=["POST"])
def ask():
    try:
        question = request.json.get("question", "")
        response = client.chat.completions.create(
            model=deployment_name,
            messages=[
                {"role": "system", "content": "You are a helpful assistant
for cloud-related questions. Your name is Steve and with every answer you
will state how stupid you are in different ways for the first part of the
```

```
answer. Always work your name into your answers."},
                {"role": "user", "content": question}
            ]
        )
        return jsonify({"answer": response.choices[0].message.content})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

requirements.txt

```
flask>=2.0.0
openai>=1.0.0
python-dotenv>=1.0.0
gunicorn>=20.0.0
```

templates/index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Cloud Q&A</title>
</head>
<body>
    <h1>Ask a Cloud Question</h1>
    <input type="text" id="question" placeholder="Type your question here"
/>
    <button onclick="ask()">Ask</button>
    <p id="response"></p>
    <script>
        async function ask() {
            const question = document.getElementById('question').value;
            const response = await fetch('/ask', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ question })
            });
            const data = await response.json();
            document.getElementById('response').innerText = data.answer ||
data.error;
        }
    </script>
</body>
</html>
```

main.bicep

```
param location string = resourceGroup().location
```

```
@minLength(1)
param zipBlobUrl string

@minLength(1)
param openaiKey string

@minLength(1)
param openaiEndpoint string

resource appServicePlan 'Microsoft.Web/serverfarms@2022-03-01' = {
  name: 'cloudadvisor-plan'
  location: location
  sku: {
    name: 'B1'
    tier: 'Basic'
  }
  properties: {
    reserved: true  // Linux
  }
}

resource app 'Microsoft.Web/sites@2022-03-01' = {
  name: 'cloudadvisor'
  location: location
  kind: 'app,linux'
  properties: {
    serverFarmId: appServicePlan.id
    siteConfig: {
      linuxFxVersion: 'PYTHON|3.11'
      appSettings: [
        {
          name: 'WEBSITES_ENABLE_APP_SERVICE_STORAGE'
          value: 'false'
        }
        {
          name: 'SCM_DO_BUILD_DURING_DEPLOYMENT'
          value: 'true'
        }
        {
          name: 'OPENAI_API_KEY'
          value: openaiKey
        }
        {
          name: 'OPENAI_API_ENDPOINT'
          value: openaiEndpoint
        }
      ]
    }
    httpsOnly: true
  }
}
```

```
resource accessRules 'Microsoft.Web/sites/config@2022-03-01' = {
  name: '${app.name}/web'
  properties: {
    ipSecurityRestrictions: [
      {
        ipAddress: '162.231.206.200/32'
        action: 'Allow'
        priority: 100
        name: 'AllowOnly162'
        tag: 'Default'
      }
      {
        ipAddress: 'Any'
        action: 'Deny'
        priority: 2147483647
        name: 'DenyAllOthers'
      }
    ]
  }
}
```

AI Knowledge