GitHub Secrets Setup for Azure Deployment

This document provides a complete guide for setting up the required GitHub secrets for Azure deployment.

Required Secrets

The following secrets must be configured in your GitHub repository for successful Azure deployment:

1. AZURE_RESOURCE_GROUP

Purpose: The name of your Azure resource group where all resources will be created.

Example Value: `ai-event-planner-rg`

How to set:

1. Go to your Azure portal
2. Create a resource group or use an existing one
3. Copy the exact resource group name

2. AZURE_LOCATION

Purpose: The Azure region where your resources will be deployed.

Example Values:

- `eastus`
- `westus2`
- `centralus`
- `westeurope`

How to set:

1. Choose an Azure region close to your users
2. Verify the region supports the services you need
3. Use the Azure CLI name format (lowercase, no spaces)

3. AZURE_CREDENTIALS

Purpose: Azure service principal credentials for authentication.

Format: JSON string containing authentication details

How to create:

1. **Create a Service Principal**:

```
az ad sp create-for-rbac --name "ai-event-planner-sp" \
  --role contributor \
  --scopes /subscriptions/{subscription-
id}/resourceGroups/{resource-group-name} \
  --sdk-auth
```

2. **Example output** (save this entire JSON as the secret):

```
{
  "clientId": "xxxx-xxxx-xxxx-xxxx",
  "clientSecret": "xxxx-xxxx-xxxx-xxxx",
  "subscriptionId": "xxxx-xxxx-xxxx-xxxx",
  "tenantId": "xxxx-xxxx-xxxx-xxxx"
}
```

3. **Alternative method using Azure Portal**:
    - Go to Azure Active Directory → App registrations → New registration
    - Create app registration for "ai-event-planner-sp"
    - Generate a client secret
    - Assign Contributor role to your resource group

4. DATABASE_URL

Purpose: PostgreSQL connection string for your Azure database.

Format: `postgresql://username:password@hostname:port/database?sslmode=require`

Example: `postgresql://eventplanner:mypassword@ai-event-db.postgres.database.azure.com:5432/eventplanner?sslmode=require`

How to create:

1. Create an Azure Database for PostgreSQL server
2. Create a database called `eventplanner`
3. Enable SSL connections
4. Use the connection string format above

5. SECRET_KEY

Purpose: Application secret key for JWT tokens and session encryption.

Format: Long random string (minimum 32 characters)

Example: `your-super-secret-key-here-minimum-32-chars-long`

How to generate:

```
# Using Python
python -c "import secrets; print(secrets.token_urlsafe(32))"

# Using OpenSSL
openssl rand -base64 32
```

6. OPENAI_API_KEY

Purpose: OpenAI API key for AI agent functionality.

Format: String starting with `sk-`

Example: `sk-1234567890abcdef...`

How to get:

1. Go to OpenAI Platform
2. Create a new API key
3. Copy the key (starts with `sk-`)

Optional Secrets

These secrets enhance functionality but are not required for basic deployment:

GOOGLE_API_KEY

Purpose: Google Services integration (Maps, Calendar, etc.)

Format: String

How to get:

1. Go to Google Cloud Console
2. Enable required APIs (Maps, Calendar, etc.)
3. Create credentials → API Key

TAVILY_API_KEY

Purpose: Web search functionality

Format: String

How to get:

1. Sign up at Tavily
2. Get your API key from the dashboard

STORAGE_CONNECTION_STRING

Purpose: Azure Storage account for file uploads

Format: Connection string

Example:
`DefaultEndpointsProtocol=https;AccountName=mystorageaccount;AccountKey=...`

How to Add Secrets to GitHub

1. **Navigate to your repository on GitHub**
2. **Click Settings (in the repository, not your profile)**
3. **Go to Secrets and variables → Actions**
4. **Click "New repository secret"**
5. **Enter the secret name and value**
6. **Click "Add secret"**

Screenshot Guide:

```
Repository → Settings → Secrets and variables → Actions → New repository secret
```

Validation

The workflow will automatically validate all required secrets before deployment and provide helpful error messages if any are missing.

Troubleshooting

"Secret not found" errors

- Ensure secret names match exactly (case-sensitive)
- Verify secrets are added to the correct repository
- Check that secrets are repository secrets, not environment secrets

Database connection issues

- Verify PostgreSQL server is running and accessible
- Check that SSL is enabled (required for Azure)
- Ensure database exists and user has proper permissions
- Test connection string format

Azure authentication issues

- Verify service principal has correct permissions
- Ensure JSON format is correct for AZURE_CREDENTIALS
- Check that subscription ID and tenant ID are correct

Runtime issues

- Ensure Python 3.10 is supported in your selected Azure region
- Verify App Service plan supports Linux containers

Security Best Practices

1. **Rotate secrets regularly** (especially API keys)
2. **Use separate secrets for different environments** (dev, staging, prod)
3. **Never commit secrets to code**
4. **Limit service principal permissions** to minimum required
5. **Monitor secret usage** in Azure and other platforms
6. **Use strong, unique passwords** for database accounts

Next Steps

After setting up all required secrets:

1. **Push to main branch** to trigger deployment
2. **Monitor GitHub Actions** for deployment progress
3. **Check Azure portal** to verify resources are created
4. **Test the deployed application**