Azure Portal Deployment Guide for FastAPI App

Prerequisites

- Azure account with active subscription
- Your FastAPI code ready in a GitHub repository or local Git
- Azure AD app registration already created

Step 1: Create Azure Cache for Redis

1. **Navigate to Azure Portal** (https://portal.azure.com)
2. **Create Redis Cache**:
   - Click **"Create a resource"** → Search for **"Azure Cache for Redis"**
   - Click **"Create"**
   - Fill in the form:
     - **Subscription**: Select your subscription
     - **Resource group**: Create new → Name it `myapp-rg`
     - **DNS name**: `myapp-redis` (must be globally unique)
     - **Location**: Select your preferred region (e.g., East US)
     - **Cache type**: Basic C0 (for testing) or Standard C1 (for production)
     - **Clustering**: Disabled
   - Click **"Review + create"** → **"Create"**
   - ⏱️ This takes 15-20 minutes to deploy
3. **Get Redis Connection Details** (after deployment):
   - Go to your Redis resource
   - Click **"Access keys"** in the left menu
   - Copy the **Primary connection string** (save this for later)

Step 2: Create App Service for Backend

1. **Create Web App**:
   - Click **"Create a resource"** → **"Web App"**
   - Fill in the basics:
     - **Subscription**: Select your subscription
     - **Resource Group**: Select existing → `myapp-rg`

- **Name**: `myapp-backend` (will be myapp-backend.azurewebsites.net)
- **Publish**: Code
- **Runtime stack**: Python 3.11
- **Operating System**: Linux
- **Region**: Same as your Redis cache

2. **App Service Plan**:
   - Click **"Create new"**
   - **Name**: `myapp-plan`
   - **Sku and size**: Click "Change size"
     - For testing: **B1** (Basic)
     - For production: **P1V2** (Premium V2)
   - Click **"Review + create"** → **"Create"**

Step 3: Configure Web App Settings

1. **Navigate to your Web App** (after deployment)
2. **Configure Application Settings**:
   - In left menu, go to **Settings** → **Configuration**
   - Click **"+ New application setting"** for each of these:

Name	Value
`AZURE_CLIENT_ID`	Your Azure AD app client ID
`AZURE_CLIENT_SECRET`	Your Azure AD app secret
`AZURE_TENANT_ID`	Your tenant ID (or "common")
`REDIRECT_URI`	`https://myapp-backend.azurewebsites.net/auth/microsoft/callback`
`SECRET_KEY`	Generate a random 32-character string
`ALGORITHM`	`HS256`
`ACCESS_TOKEN_EXPIRE_MINUTES`	`30`
`REFRESH_TOKEN_EXPIRE_DAYS`	`7`
`FRONTEND_URL`	`https://myapp-backend.azurewebsites.net` (update later)
`BACKEND_URL`	`https://myapp-backend.azurewebsites.net`
`ENVIRONMENT`	`production`
`CORS_ORIGINS`	`["https://myapp-backend.azurewebsites.net"]`
`REDIS_HOST`	Your Redis hostname (e.g., `myapp-redis.redis.cache.windows.net`)
`REDIS_PORT`	`6380`
`REDIS_PASSWORD`	Your Redis access key
`REDIS_SSL`	`true`
`REDIS_SSL_CERT_REQS`	`required`
`SECURE_COOKIES`	`true`

3. **Configure General Settings**:
   - Still in **Configuration**, click **"General settings"** tab
   - **Stack settings**:
     - **Stack**: Python
     - **Major version**: Python 3
     - **Minor version**: Python 3.11
   - **Startup Command**:

     ```
     gunicorn -w 4 -k uvicorn.workers.UvicornWorker main:app
     ```

   - Click **"Save"** at the top
   - Click **"Continue"** when prompted about restart
4. **Configure HTTPS**:
   - Go to **Settings** → **TLS/SSL settings**

- Set **HTTPS Only**: On

Step 4: Prepare Your Code for Deployment

1. **Create required files in your project root**: **requirements.txt**:

```txt
fastapi==0.104.1
uvicorn[standard]==0.24.0
gunicorn==21.2.0
msal==1.25.0
python-jose[cryptography]==3.3.0
python-multipart==0.0.6
redis==5.0.1
slowapi==0.1.9
pydantic-settings==2.1.0
python-dotenv==1.0.0
httpx==0.25.2
aiofiles==23.2.1
```

.deployment:

```ini
[config]
SCM_DO_BUILD_DURING_DEPLOYMENT=true
```

2. **Update your code structure**:
   - Ensure `main.py` is in the root directory
   - Move other Python files to appropriate locations
   - Update import paths if necessary

Step 5: Deploy Your Code

Option A: Deploy from Local Git

1. **In Azure Portal**, go to your Web App
2. Go to **Deployment** → **Deployment Center**
3. **Source**: Select **Local Git**
4. Click **Save**
5. Go to **Deployment Center** → **Local Git/FTPS credentials**
6. Copy the **Git Clone Uri**

7. Note the **Username** (looks like `$myapp-backend`)
8. **In your local terminal**:

```bash
git remote add azure https://myapp-backend.scm.azurewebsites.net/myapp-backend.git
git push azure main:master
```

   - Use the username and password from step 6-7

Option B: Deploy from GitHub

1. **Push your code to GitHub**
2. **In Azure Portal**, go to your Web App
3. Go to **Deployment** → **Deployment Center**
4. **Source**: Select **GitHub**
5. **Authorize** Azure to access your GitHub
6. Select:
   - **Organization**: Your GitHub username
   - **Repository**: Your repo name
   - **Branch**: main (or master)
7. Click **Save**

Step 6: Deploy Frontend

Option A: Use Azure Static Web Apps (Recommended)

1. **Create Static Web App**:
   - Click **"Create a resource"** → **"Static Web App"**
   - Fill in:
     - **Subscription**: Your subscription
     - **Resource group**: `myapp-rg`
     - **Name**: `myapp-frontend`
     - **Plan type**: Free
     - **Region**: Select your region
     - **Deployment details**:
       - **Source**: GitHub
       - **GitHub Account**: Sign in and authorize

        - **Organization**: Your username
        - **Repository**: Your repo (or create new)
        - **Branch**: main
    - **Build Details**:
        - **Build Presets**: Custom
        - **App location**: `/frontend`
        - **Api location**: Leave empty
        - **Output location**: Leave empty
- Click **"Review + create"** → **"Create"**

Option B: Use Azure Blob Storage

1. **Create Storage Account**:
    - Click **"Create a resource"** → **"Storage account"**
    - Fill in:
        - **Subscription**: Your subscription
        - **Resource group**: `myapp-rg`
        - **Storage account name**: `myappfrontend` (must be globally unique)
        - **Region**: Same as other resources
        - **Performance**: Standard
        - **Redundancy**: LRS (Locally-redundant storage)
    - Click **"Review + create"** → **"Create"**
2. **Enable Static Website** (after deployment):
    - Go to your storage account
    - In left menu, under **Data management** → **Static website**
    - **Static website**: Enabled
    - **Index document name**: `index.html`
    - **Error document path**: `404.html` (optional)
    - Click **Save**
    - Copy the **Primary endpoint** URL (this is your frontend URL)
3. **Upload Frontend Files**:
    - In left menu, go to **Data storage** → **Containers**

- Click on **$web** container
- Click **Upload**
- Upload your HTML files (make sure to update the API_URL in the files first)

Step 7: Update Azure AD App Registration

1. Go to **Azure Active Directory → App registrations**
2. Click on your app
3. Go to **Authentication**
4. Under **Redirect URIs**, add:
   - `https://myapp-backend.azurewebsites.net/auth/microsoft/callback`
5. Click **Save**

Step 8: Update CORS Settings

1. Go back to your Web App in Azure Portal
2. Go to **API → CORS**
3. Add your frontend URL to allowed origins:
   - If using Static Web Apps: `https://myapp-frontend.azurewebsites.net`
   - If using Blob Storage: Your primary endpoint URL
4. Click **Save**

Step 9: Test Your Deployment

1. **Test Backend Health**:
   - Navigate to: `https://myapp-backend.azurewebsites.net/health`
   - You should see a JSON response with health status
2. **Test Frontend**:
   - Navigate to your frontend URL
   - Try the login flow

Step 10: Monitor and Troubleshoot

1. **View Logs**:
   - In your Web App, go to **Monitoring → Log stream**
   - Watch real-time logs during testing

2. **Enable Application Insights** (Optional):
    - In your Web App, go to **Settings** → **Application Insights**
    - Click **Turn on Application Insights**
    - Create new or select existing
    - Click **Apply**

Troubleshooting Common Issues

"Application Error" or 502/503 errors

1. Check **Log stream** for detailed errors
2. Verify all environment variables are set correctly
3. Ensure `requirements.txt` is in root directory
4. Check startup command is correct

Redis Connection Errors

1. Verify Redis is fully deployed (can take 20 minutes)
2. Check Redis hostname and password are correct
3. Ensure Redis SSL settings match your configuration

Authentication Issues

1. Verify redirect URIs in Azure AD match exactly
2. Check CORS settings include your frontend URL
3. Ensure cookies are configured for production

Slow Performance

1. Scale up your App Service Plan
2. Enable **Always On** in Configuration → General settings
3. Consider using Redis Premium tier for better performance

Next Steps

1. **Custom Domain**:
    - Go to **Settings** → **Custom domains**
    - Add your domain and configure DNS
2. **Scaling**:

- Go to **Settings** → **Scale up** (App Service plan)
- Or **Settings** → **Scale out** (instance count)

3. **Backup**:
   - Go to **Settings** → **Backups**
   - Configure automated backups
4. **Security**:
   - Enable **Azure AD authentication** at the App Service level
   - Configure **IP restrictions** if needed
   - Use **Azure Key Vault** for secrets