

Odin's Eye - Azure Deployment Guide

Business: Viking Restaurant Consultants LLC

Application: Odin's Eye (P&L Converter)

Version: 1.0.0

Last Updated: October 2025

Table of Contents

1. [Overview](#)
 2. [Prerequisites](#)
 3. [Before You Begin](#)
 4. [Quick Start](#)
 5. [Detailed Deployment Steps](#)
 6. [Environment Variables Configuration](#)
 7. [Post-Deployment Steps](#)
 8. [Troubleshooting](#)
 9. [Managing Your Application](#)
 10. [Cost Optimization](#)
 11. [Security Best Practices](#)
 12. [FAQ](#)
-

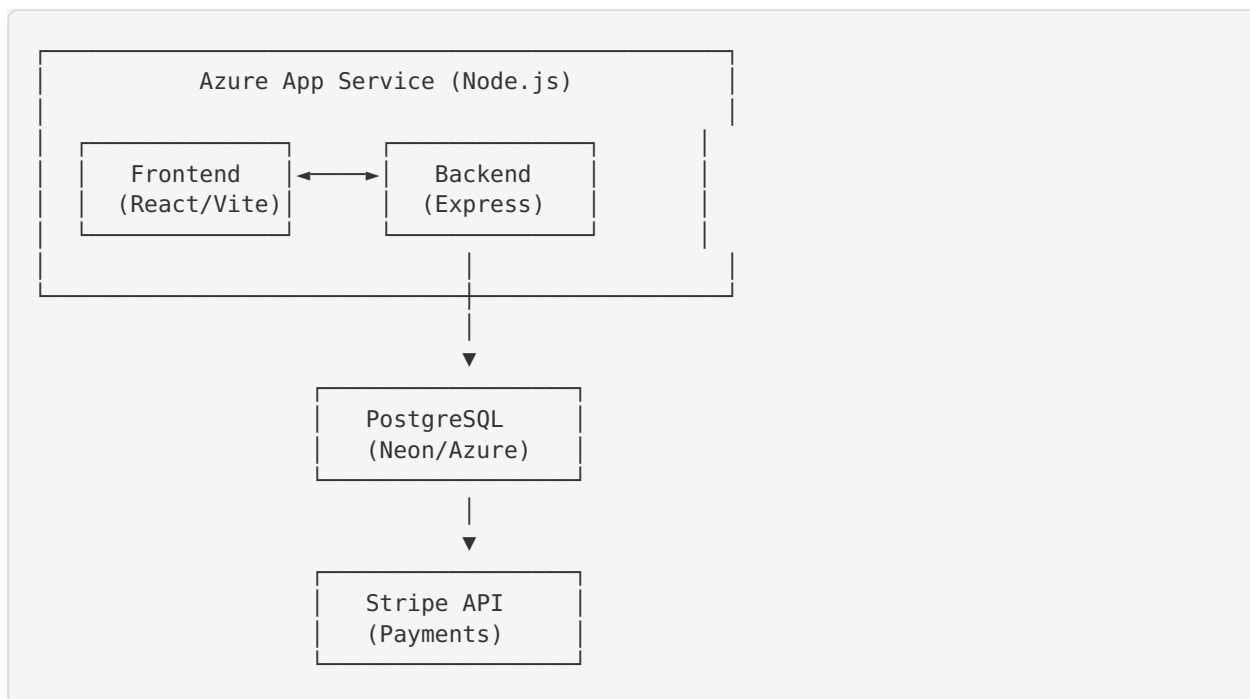
Overview

This deployment package contains everything you need to deploy the Odin's Eye application to Microsoft Azure. The application is a powerful P&L (Profit & Loss) converter designed specifically for restaurant operations.

What's Included

-  **PowerShell deployment script** (`scripts/deploy-odins-eye.ps1`) - For Windows users
-  **Bash deployment script** (`scripts/deploy-odins-eye.sh`) - For Linux/Mac users
-  **Application package** (`odins-eye-app.zip`) - Ready-to-deploy application
-  **Environment template** (`.env.template`) - Configuration template
-  **This comprehensive guide** - Step-by-step instructions

Architecture



Prerequisites

Required Software

Before running the deployment scripts, ensure you have the following installed:

1. Azure CLI (Required)

- **Windows:** Download from [Azure CLI Installer](https://aka.ms/installazurecliwindows) (<https://aka.ms/installazurecliwindows>)

- **macOS:**

```
bash
```

```
brew update && brew install azure-cli
```

- **Linux (Ubuntu/Debian):**

```
bash
```

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

Verify installation:

```
bash
```

```
az --version
```

2. Node.js 20.x or higher (Required)

- Download from nodejs.org (<https://nodejs.org/>)

- **Verify installation:**

```
bash
```

```
node --version # Should be v20.x or higher
```

```
npm --version
```

3. PowerShell 7.x (Windows users)

- Download from [PowerShell GitHub](https://github.com/PowerShell/PowerShell/releases) (<https://github.com/PowerShell/PowerShell/releases>)

- **Verify installation:**

```
powershell
```

```
$PSVersionTable.PSVersion
```

4. jq (Linux/Mac users, optional but recommended)

- **macOS:** `brew install jq`
- **Linux:** `sudo apt-get install jq`

Azure Account Requirements

- ☒ Active Azure subscription
- ☒ Subscription ID (default: `5e0e2c8e-e8b7-4cb0-8e5e-c8e7e8b7e8b7`)
- ☒ Contributor or Owner role on the subscription
- ☒ Sufficient quota for:
 - 1 Resource Group
 - 1 App Service Plan (B1 tier minimum)
 - 1 App Service (Web App)

External Services

1. PostgreSQL Database (Required)

You need a PostgreSQL database. Recommended options:

- **Neon** (neon.tech) - Free tier available, serverless
- **Azure Database for PostgreSQL** - Managed service
- **Supabase** - Free tier available
- **Heroku Postgres** - Free tier available

You'll need the connection string in this format:

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

2. Stripe Account (Required)

- Create an account at stripe.com (https://stripe.com)
- Navigate to **Developers → API Keys**
- Copy your:
 - **Publishable key** (starts with `pk_`)
 - **Secret key** (starts with `sk_`)

Before You Begin

1. Gather Required Information

Create a checklist and gather the following:

- ☐ Azure Subscription ID
- ☐ Stripe Publishable Key
- ☐ Stripe Secret Key
- ☐ PostgreSQL Database URL
- ☐ Preferred Azure region (default: East US)

2. Create Your Database

If you don't have a PostgreSQL database yet:

Option A: Neon (Recommended for development)

1. Go to neon.tech (<https://neon.tech>)
2. Sign up for a free account
3. Create a new project
4. Copy the connection string

Option B: Azure Database for PostgreSQL

```
# Create database server
az postgres server create \
  --resource-group viking-restaurant-rg \
  --name odins-eye-db-server \
  --location eastus \
  --admin-user dbadmin \
  --admin-password <YourPassword> \
  --sku-name B_Gen5_1

# Create database
az postgres db create \
  --resource-group viking-restaurant-rg \
  --server-name odins-eye-db-server \
  --name odinseyedb
```

3. Set Up Environment Variables (Optional)

To avoid entering credentials during deployment, set environment variables:

Windows (PowerShell):

```
$env:STRIPE_PUBLISHABLE_KEY = "pk_test_..."
$env:STRIPE_SECRET_KEY = "sk_test_..."
$env:DATABASE_URL = "postgresql://..."
```

Linux/Mac (Bash):

```
export STRIPE_PUBLISHABLE_KEY="pk_test_..."
export STRIPE_SECRET_KEY="sk_test_..."
export DATABASE_URL="postgresql://..."
```

Or create a `.env` file in the deployment package directory using the provided template.

Quick Start

For Windows Users (PowerShell)

1. Open PowerShell 7 as Administrator
2. Navigate to the deployment package:

```
powershell
cd path\to\odins-eye-deployment-package
```

3. Run the deployment script:

```
powershell  
.\scripts\deploy-odins-eye.ps1
```

For Linux/Mac Users (Bash)

1. Open Terminal
2. Navigate to the deployment package:

```
bash  
cd path/to/odins-eye-deployment-package
```

3. Make the script executable:

```
bash  
chmod +x scripts/deploy-odins-eye.sh
```

4. Run the deployment script:

```
bash  
./scripts/deploy-odins-eye.sh
```

The script will:

- ☒ Check prerequisites
- ☒ Authenticate with Azure
- ☒ Create or reuse existing resources
- ☒ Configure environment variables
- ☒ Build and deploy the application
- ☒ Display the application URL

Expected Duration: 5-10 minutes

Detailed Deployment Steps

Step 1: Authenticate with Azure

```
az login
```

This will open a browser window. Sign in with your Azure account credentials.

Verify authentication:

```
az account show
```

Switch subscription (if needed):

```
az account set --subscription "5e0e2c8e-e8b7-4cb0-8e5e-c8e7e8b7e8b7"
```

Step 2: Run the Deployment Script

Windows (PowerShell):

```
# Basic deployment
.\scripts\deploy-odins-eye.ps1

# With custom parameters
.\scripts\deploy-odins-eye.ps1 `
  -SubscriptionId "your-subscription-id" `
  -Region "westus2" `
  -ResourceGroup "custom-rg-name"

# Skip build step (if already built)
.\scripts\deploy-odins-eye.ps1 -SkipBuild
```

Linux/Mac (Bash):

```
# Basic deployment
./scripts/deploy-odins-eye.sh

# With custom parameters
./scripts/deploy-odins-eye.sh \
  --subscription "your-subscription-id" \
  --region "westus2" \
  --resource-group "custom-rg-name"

# Skip build step (if already built)
./scripts/deploy-odins-eye.sh --skip-build

# Show help
./scripts/deploy-odins-eye.sh --help
```

Step 3: Monitor Deployment

The script will display progress messages:

```
=====
Odin's Eye Azure Deployment Script
=====

Business: Viking Restaurant Consultants LLC
Application: Odin's Eye (P&L Converter)
Target Subscription: 5e0e2c8e-e8b7-4cb0-8e5e-c8e7e8b7e8b7
Target Region: eastus

=====
Step 1: Checking Prerequisites
=====
▶ Checking Azure CLI installation...
✓ Azure CLI is installed (version 2.54.0)
...
```

Step 4: Handle Environment Variables

If you didn't set environment variables beforehand, the script will prompt you:

Please enter Stripe Publishable Key:
 STRIPE_PUBLISHABLE_KEY: pk_test_...




Please enter Stripe Secret Key:
 STRIPE_SECRET_KEY: sk_test_...

Please enter PostgreSQL Database URL:
 DATABASE_URL: postgresql://...

Tip: Copy and paste these values to avoid typos.

Step 5: Wait for Deployment

The deployment process includes:

1.  Resource provisioning (1-2 minutes)
2.  Application upload (1-2 minutes)
3.  Build and compilation on Azure (2-5 minutes)

Total time: 5-10 minutes

Environment Variables Configuration

Required Environment Variables

Variable	Description	Example
STRIPE_PUBLISHABLE_KEY	Stripe publishable API key	pk_test_51A...
STRIPE_SECRET_KEY	Stripe secret API key	sk_test_51A...
DATABASE_URL	PostgreSQL connection string	postgresql:// user:pass@host:5432/db

Automatically Set Variables

These are configured by the deployment script:

Variable	Value	Description
NODE_ENV	production	Node.js environment
WEB-SITE_NODE_DEFAULT_VERSION	~20	Node.js version
SCM_DO_BUILD_DURING_DEPLOYMENT	true	Enable build on deployment

Updating Environment Variables

After deployment, you can update variables using Azure CLI:

```
# Update a single variable
az webapp config appsettings set \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg \
  --settings "STRIPE_SECRET_KEY=sk_live_new_key"

# Update multiple variables
az webapp config appsettings set \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg \
  --settings \
    "VAR1=value1" \
    "VAR2=value2"

# Restart app to apply changes
az webapp restart \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg
```

Or use the Azure Portal:

1. Navigate to your App Service
2. Go to **Settings → Configuration**
3. Click + **New application setting**
4. Enter name and value
5. Click **Save** then **Continue**

Post-Deployment Steps

1. Run Database Migrations

After deployment, you need to initialize the database schema:

Option A: Using Azure SSH

```
# SSH into the app
az webapp ssh \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg

# Once connected, run migrations
cd /home/site/wwwroot
npm run db:push
```

Option B: Using Local Connection

```
# Set database URL locally
export DATABASE_URL="your-database-url"

# Navigate to app directory
cd app/

# Run migrations
npm run db:push
```


2. Verify Application

Visit your application URL (provided at the end of deployment):

```
https://odins-valhalla.azurewebsites.net
```

Expected behavior:

- ☒ Landing page loads
- ☒ Login/Register functionality works
- ☒ No JavaScript errors in console
- ☒ Stripe integration loads

3. Test Core Functionality

1. **Create an account** or log in
2. **Upload a test P&L file** (Excel/CSV)
3. **Verify data processing**
4. **Test Stripe payment flow** (use test cards)
5. **Export results** (PDF/Excel)

4. Configure Custom Domain (Optional)

```
# Add custom domain
az webapp config hostname add \
  --webapp-name odins-valhalla \
  --resource-group viking-restaurant-rg \
  --hostname odins-eye.yourdomain.com

# Configure SSL
az webapp config ssl bind \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg \
  --certificate-thumbprint <thumbprint> \
  --ssl-type SNI
```

5. Enable Application Insights (Recommended)

```
# Create Application Insights
az monitor app-insights component create \
  --app odins-eye-insights \
  --location eastus \
  --resource-group viking-restaurant-rg

# Get instrumentation key
INSTRUMENTATION_KEY=$(az monitor app-insights component show \
  --app odins-eye-insights \
  --resource-group viking-restaurant-rg \
  --query instrumentationKey -o tsv)

# Configure app to use Application Insights
az webapp config appsettings set \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg \
  --settings "APPINSIGHTS_INSTRUMENTATIONKEY=$INSTRUMENTATION_KEY"
```

Troubleshooting

Common Issues and Solutions

1. Azure CLI Not Found

Error:

```
'az' is not recognized as an internal or external command
```

Solution:

- Install Azure CLI from [microsoft.com/azure/cli](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli) (<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>)
- Restart your terminal after installation
- Verify with: `az --version`

2. Authentication Failed

Error:

```
Please run 'az login' to setup account
```

Solution:

```
az login
az account set --subscription "5e0e2c8e-e8b7-4cb0-8e5e-c8e7e8b7e8b7"
```

3. App Name Already Exists

Error:

```
The webapp 'odins-almanac' already exists
```

Solution:

- The script handles this automatically by checking both `odins-almanac` and `odins-valhalla`
- If both exist and you want a new one, edit the script to use a different name

4. Deployment Package Not Found

Error:

```
Deployment package not found: odins-eye-app.zip
```

Solution:

- Ensure `odins-eye-app.zip` is in the deployment package root directory
- If missing, the script should have created it during the build step
- Manually create it:

```
bash
```

```
cd app/
```

```
zip -r ../odins-eye-app.zip . -x "*.git*" -x "*node_modules/*"
```

5. Application Not Starting

Symptoms:

- App service shows “Your web app is running and waiting for your content”
- Or shows an error page

Solutions:

Check logs:

```
az webapp log tail \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg
```

Common causes:

- Missing environment variables
- Database connection issue
- Build failed on Azure

Fix:

1. Verify environment variables:

```
bash
az webapp config appsettings list \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg
```

1. Check deployment logs:

```
bash
az webapp log deployment show \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg
```

2. Restart the app:

```
bash
az webapp restart \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg
```

6. Database Connection Error

Error in logs:

```
Error: connect ECONNREFUSED
Could not connect to database
```

Solutions:

1. **Verify DATABASE_URL format:**

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

2. **Check database firewall rules** (for Azure PostgreSQL):

```
bash
az postgres server firewall-rule create \
  --resource-group viking-restaurant-rg \
```

```
--server-name odins-eye-db-server \
--name AllowAzureServices \
--start-ip-address 0.0.0.0 \
--end-ip-address 0.0.0.0
```

3. Test connection from local machine:

```
bash
psql "postgresql://username:password@host:port/database?sslmode=require"
```

7. Stripe Integration Not Working

Symptoms:

- Payment buttons don't appear
- Stripe checkout doesn't load

Solutions:

1. Verify Stripe keys are set:

```
bash
az webapp config appsettings list \
--name odins-valhalla \
--resource-group viking-restaurant-rg \
--query "[?name=='STRIPE_PUBLISHABLE_KEY' || name=='STRIPE_SECRET_KEY']"
```

2. Check browser console for JavaScript errors

3. Verify keys are not expired in Stripe Dashboard

8. Out of Memory Errors

Error in logs:

```
JavaScript heap out of memory
```

Solution:

Upgrade to a higher App Service Plan:

```
az appservice plan update \
--name viking-app-service-plan \
--resource-group viking-restaurant-rg \
--sku B2
```

9. Slow Performance

Solutions:

1. Enable Always On:

```
bash
az webapp config set \
--name odins-valhalla \
--resource-group viking-restaurant-rg \
--always-on true
```

2. Upgrade App Service Plan:

```
bash
```

```
az appservice plan update \  
  --name viking-app-service-plan \  
  --resource-group viking-restaurant-rg \  
  --sku P1V2
```

3. Enable HTTP/2:

```
bash  
az webapp config set \  
  --name odins-valhalla \  
  --resource-group viking-restaurant-rg \  
  --http2-enabled true
```

Managing Your Application

Viewing Logs

Real-time logs:

```
az webapp log tail \  
  --name odins-valhalla \  
  --resource-group viking-restaurant-rg
```

Download logs:

```
az webapp log download \  
  --name odins-valhalla \  
  --resource-group viking-restaurant-rg \  
  --log-file app-logs.zip
```

Restarting the Application

```
az webapp restart \  
  --name odins-valhalla \  
  --resource-group viking-restaurant-rg
```

Stopping the Application

```
az webapp stop \  
  --name odins-valhalla \  
  --resource-group viking-restaurant-rg
```

Starting the Application

```
az webapp start \  
  --name odins-valhalla \  
  --resource-group viking-restaurant-rg
```

Scaling the Application

Vertical scaling (bigger instance):

```
az appservice plan update \
  --name viking-app-service-plan \
  --resource-group viking-restaurant-rg \
  --sku P1V2
```

Horizontal scaling (more instances):

```
az appservice plan update \
  --name viking-app-service-plan \
  --resource-group viking-restaurant-rg \
  --number-of-workers 3
```

Backing Up Data

```
# Create storage account for backups
az storage account create \
  --name odineyebackups \
  --resource-group viking-restaurant-rg \
  --location eastus \
  --sku Standard_LRS

# Configure backup
az webapp config backup create \
  --resource-group viking-restaurant-rg \
  --webapp-name odins-valhalla \
  --backup-name daily-backup \
  --container-url <storage-container-url>
```

Redeploying the Application

To redeploy after making changes:

1. **Update the application code** in the `app/` directory
2. **Rebuild the package:**

```
bash
cd app/
npm run build
cd ..
zip -r odins-eye-app.zip app/ -x "*.git*" -x "**node_modules/*"
```

3. **Redeploy:**

```
bash
az webapp deploy \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg \
  --src-path odins-eye-app.zip \
  --type zip
```

Deleting Resources

 **WARNING: This will permanently delete all resources and data!**

```
# Delete entire resource group
az group delete \
  --name viking-restaurant-rg \
  --yes \
  --no-wait
```

Or delete individual resources:

```
# Delete just the web app
az webapp delete \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg
```

Cost Optimization

Estimated Monthly Costs

Resource	SKU	Estimated Cost
App Service Plan	B1	~\$13.14/month
App Service	Included	\$0
PostgreSQL (Neon)	Free tier	\$0
Stripe	Pay-per-transaction	2.9% + \$0.30 per transaction
Total		~\$13-15/month

Cost-Saving Tips

- 1. **Use Free Tier Database:**
 - Neon PostgreSQL (free tier: 0.5 GB storage)
 - Supabase (free tier: 500 MB storage)

- 2. **Stop App When Not in Use:**

```
bash
az webapp stop --name odins-valhalla --resource-group viking-restaurant-rg
```

- 3. **Use B1 for Development, Scale Up for Production:**
 - Development: B1 (\$13/month)
 - Production: P1V2 (\$73/month) or higher

- 4. **Enable Auto-scaling** only when needed

- 5. **Monitor Usage:**

```
bash
az monitor metrics list \
  --resource odins-valhalla \
  --resource-group viking-restaurant-rg \
```

```
--resource-type "Microsoft.Web/sites" \  
--metric "CpuTime"
```

Security Best Practices

1. Use Managed Identity

Enable managed identity to avoid storing credentials:

```
az webapp identity assign \  
--name odins-valhalla \  
--resource-group viking-restaurant-rg
```

2. Enable HTTPS Only

```
az webapp update \  
--name odins-valhalla \  
--resource-group viking-restaurant-rg \  
--https-only true
```

3. Set Minimum TLS Version

```
az webapp config set \  
--name odins-valhalla \  
--resource-group viking-restaurant-rg \  
--min-tls-version 1.2
```

4. Enable Security Headers

Add custom headers via Application Settings:

```
az webapp config appsettings set \  
--name odins-valhalla \  
--resource-group viking-restaurant-rg \  
--settings \  
  "X-Frame-Options=DENY" \  
  "X-Content-Type-Options=nosniff"
```

5. Rotate Secrets Regularly

Update Stripe keys and database passwords periodically.

6. Enable Azure AD Authentication (Optional)

```
az webapp auth update \  
--name odins-valhalla \  
--resource-group viking-restaurant-rg \  
--enabled true \  
--action LoginWithAzureActiveDirectory
```


7. Use Key Vault for Secrets (Advanced)

```
# Create Key Vault
az keyvault create \
  --name odinseye-vault \
  --resource-group viking-restaurant-rg \
  --location eastus

# Store secrets
az keyvault secret set \
  --vault-name odinseye-vault \
  --name stripe-secret-key \
  --value "sk_live_..."

# Reference in App Service
az webapp config appsettings set \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg \
  --settings "STRIPE_SECRET_KEY=@Microsoft.KeyVault(SecretUri=https://odinseye-vault.vault.azure.net/secrets/stripe-secret-key/)"
```

FAQ

Q: Can I use this deployment on other cloud providers?

A: This package is specifically designed for Azure. However, the application itself (in `app/`) can be deployed to:

- AWS Elastic Beanstalk
- Google Cloud App Engine
- Heroku
- DigitalOcean App Platform
- Any Node.js hosting provider

You'll need to adapt the deployment process for each platform.

Q: What if I already have “Odins Almanac” running?

A: The script will automatically detect it and reuse it. If you want to create a new app instead, edit the script and change the `APP_NAME_SECONDARY` variable to a different name.

Q: Can I use MySQL instead of PostgreSQL?

A: The application is built for PostgreSQL using Drizzle ORM. While theoretically possible, it would require code modifications. PostgreSQL is recommended.

Q: How do I upgrade Node.js version?

A: Update the `WEBSITE_NODE_DEFAULT_VERSION` setting:

```
az webapp config appsettings set \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg \
  --settings "WEBSITE_NODE_DEFAULT_VERSION=~21"
```

Q: Can I run this locally for development?

A: Yes! Navigate to the `app/` directory and:

```
npm install
npm run dev
```

Create a `.env` file with your environment variables.

Q: How do I enable custom domains?

A: See [Post-Deployment Steps → Configure Custom Domain](#)

Q: What's the difference between “Odins Almanac” and “Odins Valhalla”?

A: These are just names for the Azure App Service. The script checks if “Odins Almanac” exists first (to reuse it). If not, it creates “Odins Valhalla”. Both run the same application.

Q: How do I update the application after deployment?

A: See [Managing Your Application → Redeploying the Application](#)

Q: Is there a staging environment?

A: You can create deployment slots for staging:

```
az webapp deployment slot create \
  --name odins-valhalla \
  --resource-group viking-restaurant-rg \
  --slot staging
```

Q: How do I monitor application health?

A: Use Application Insights (see [Post-Deployment Steps → Enable Application Insights](#))

Support and Resources

Official Documentation

- [Azure App Service Documentation](https://docs.microsoft.com/en-us/azure/app-service/) (https://docs.microsoft.com/en-us/azure/app-service/)
- [Azure CLI Reference](https://docs.microsoft.com/en-us/cli/azure/) (https://docs.microsoft.com/en-us/cli/azure/)
- [Node.js on Azure](https://docs.microsoft.com/en-us/azure/developer/javascript/) (https://docs.microsoft.com/en-us/azure/developer/javascript/)

Community Resources

- [Azure Community Support](https://azure.microsoft.com/en-us/support/community/) (https://azure.microsoft.com/en-us/support/community/)
- [Stack Overflow - Azure Tag](https://stackoverflow.com/questions/tagged/azure) (https://stackoverflow.com/questions/tagged/azure)

Business Contact

Viking Restaurant Consultants LLC

For application-specific questions and support.


Changelog

Version 1.0.0 (October 2025)

- Initial deployment package release
 - PowerShell and Bash deployment scripts
 - Comprehensive documentation
 - Azure App Service deployment
 - PostgreSQL and Stripe integration
-

License

This deployment package is provided by Viking Restaurant Consultants LLC for the Odin's Eye application.

 **Congratulations!** You're now ready to deploy Odin's Eye to Azure. If you encounter any issues not covered in this guide, please consult the Troubleshooting section or reach out for support.

Good luck with your deployment! 