Azure Deployment Guide - MEDEVAC Application

Step-by-Step Instructions for First-Time Setup

Prerequisites

- ☑ Azure subscription with appropriate permissions
- ☑ GitHub account with repository access
- ☑ Node.js 18+ installed locally
- ☑ Azure CLI installed (`az --version` to verify)
- ☑ Git installed and configured

🚀 Phase 1: Initial Azure Setup (30 minutes)

Step 1: Login and Configure Azure CLI

```
# Login to Azure
az login

# List available subscriptions
az account list --output table

# Set the correct subscription (replace with your subscription ID)
az account set --subscription "your-subscription-id"

# Set your existing resource group name as a variable for easy management
$resourceGroup = "IAF-MEDEVAC"

# Verify you're using the correct subscription and resource group exists
az account show
az group show --name $resourceGroup
```

Step 2: Get Resource Group Location

```
# Get the location of your existing resource group
$location = az group show --name $resourceGroup --query "location" --output tsv
Write-Host "Using resource group: $resourceGroup in location: $location"
```

Step 3: Set Storage Account Variables (Already Created)

```
# Your storage account was created through Azure portal
$storageAccountName = "medevaciafstorage"

# Get the connection string for your existing storage account
```

```
$connectionString = az storage account show-connection-string `
  --name $storageAccountName `
  --resource-group $resourceGroup `
  --query "connectionString" --output tsv

Write-Host "Using existing storage account: $storageAccountName"
Write-Host "Connection String: $connectionString"
```

Step 4: Create Blob Storage Containers

```
# Create containers for different data types using your connection string
az storage container create --name "perdiem-cache" --connection-string
$connectionString
az storage container create --name "rate-history" --connection-string
$connectionString
az storage container create --name "scraper-logs" --connection-string
$connectionString
az storage container create --name "application-data" --connection-string
$connectionString

# Verify containers were created
az storage container list --connection-string $connectionString --output table
```

Phase 2: Security Setup (20 minutes)

Step 5: Create Azure Key Vault

```
# Create Key Vault (name must be globally unique - POC version)
$keyVaultName = "kv-medevac-poc" + (Get-Random -Maximum 999)
az keyvault create `
  --name $keyVaultName `
  --resource-group $resourceGroup `
  --location $location `
  --sku "standard"

Write-Host "Created Key Vault: $keyVaultName"
# Get your user principal name for access policy
az ad signed-in-user show --query "userPrincipalName" --output tsv
```

Step 6: Store Secrets in Key Vault

```
# Store the storage connection string
az keyvault secret set `
  --vault-name "kv-medevac-prod" `
  --name "StorageConnectionString" `
```

```
  --value "YOUR_CONNECTION_STRING_HERE"

# Store other application secrets (replace with actual values)
az keyvault secret set `
  --vault-name "kv-medevac-prod" `
  --name "StateDeptApiKey" `
  --value "your-state-dept-api-key-if-any"

# Verify secrets were stored
az keyvault secret list --vault-name "kv-medevac-prod" --output table
```

💻 Phase 3: Backend API Deployment (25 minutes)

Step 7: Create App Service Plan

```
# Create App Service Plan (Basic B1 for POC - much cheaper!)
az appservice plan create `
  --name "plan-medevac-poc" `
  --resource-group $resourceGroup `
  --location $location `
  --sku "B1" `
  --is-linux true
```

Step 8: Create App Service for Backend API

```
# Create App Service (POC naming)
az webapp create `
  --name "medevac-poc-api" `
  --resource-group $resourceGroup `
  --plan "plan-medevac-poc" `
  --runtime "NODE:18-lts"

# Enable system-assigned managed identity
az webapp identity assign `
  --name "medevac-poc-api" `
  --resource-group $resourceGroup

# Get the principal ID (save this for Key Vault access)
$principalId = az webapp identity show `
  --name "medevac-poc-api" `
  --resource-group $resourceGroup `
  --query "principalId" --output tsv

Write-Host "Principal ID: $principalId"
```

Step 9: Grant App Service Access to Key Vault

```
# Grant the App Service access to Key Vault secrets
az keyvault set-policy `
  --name "kv-medevac-prod" `
  --object-id $principalId `
  --secret-permissions get list
```

Step 10: Configure App Service Settings

```
# Configure application settings
az webapp config appsettings set `
  --name "medevac-prod-api" `
  --resource-group "rg-medevac-prod" `
  --settings `
    AZURE_STORAGE_CONNECTION_STRING="@Microsoft.KeyVault(VaultName=kv-medevac-
prod;SecretName=StorageConnectionString)" `
    KEY_VAULT_URL="https://kv-medevac-prod.vault.azure.net/" `
    NODE_ENV="production" `
    FRONTEND_URL="https://medevac-prod.azurestaticapps.net"
```

Phase 4: Frontend Deployment (20 minutes)

Step 11: Prepare Your Code for Deployment

```
# Navigate to your project directory
cd "c:\Users\VSACK\medevacform"

# Create production build
npm install
npm run build

# Verify build was successful
ls build
```

Step 12: Create Azure Static Web App (Free Tier)

```
# Create Static Web App Free tier (perfect for POC!)
az staticwebapp create `
  --name "medevac-poc" `
  --resource-group $resourceGroup `
  --source "https://github.com/your-username/medevacform" `
  --location "East US2" `
  --branch "main" `
  --app-location "/" `
```

```
  --output-location "build" `
  --sku "Free"
```

Important: The above command will provide you with a GitHub workflow file. Copy this content and create `.github/workflows/azure-static-web-apps-[random-string].yml` in your repository.

Step 13: Configure Static Web App Routing

Create a `staticwebapp.config.json` file in your project root:

```
# Create the configuration file
@"
{
  "routes": [
    {
      "route": "/api/*",
      "rewrite": "https://medevac-prod-api.azurewebsites.net/api/*"
    },
    {
      "route": "/*",
      "serve": "/index.html",
      "statusCode": 200
    }
  ],
  "responseOverrides": {
    "404": {
      "rewrite": "/index.html"
    }
  },
  "globalHeaders": {
    "Cache-Control": "public, max-age=31536000, immutable"
  }
}
"@ | Out-File -FilePath "staticwebapp.config.json" -Encoding UTF8
```

⚡ Phase 5: Azure Functions (Scraper Service) (25 minutes)

Step 14: Create Function App (Consumption Plan - Free!)

```
# Create Function App with Consumption plan (pay-per-use, much cheaper for POC)
az functionapp create `
  --name "medevac-poc-functions" `
  --resource-group $resourceGroup `
  --consumption-plan-location $location `
  --runtime "node" `
  --runtime-version "18" `
  --functions-version "4" `
  --storage-account $storageAccountName
```

```
# Enable managed identity for Functions
az functionapp identity assign `
  --name "medevac-poc-functions" `
  --resource-group $resourceGroup

# Get Function App principal ID
$functionPrincipalId = az functionapp identity show `
  --name "medevac-poc-functions" `
  --resource-group $resourceGroup `
  --query "principalId" --output tsv

# Grant Functions access to Key Vault
az keyvault set-policy `
  --name $keyVaultName `
  --object-id $functionPrincipalId `
  --secret-permissions get list
```

Step 15: Configure Function App Settings

```
# Configure Function App settings
az functionapp config appsettings set `
  --name "medevac-prod-functions" `
  --resource-group "rg-medevac-prod" `
  --settings `
    AZURE_STORAGE_CONNECTION_STRING="@Microsoft.KeyVault(VaultName=kv-medevac-
prod;SecretName=StorageConnectionString)" `
    KEY_VAULT_URL="https://kv-medevac-prod.vault.azure.net/" `
    FUNCTIONS_WORKER_RUNTIME="node"
```

Phase 6: Monitoring Setup (15 minutes)

Step 16: Create Application Insights

```
# Create Application Insights
az monitor app-insights component create `
  --app "medevac-prod-insights" `
  --location "East US" `
  --resource-group "rg-medevac-prod" `
  --kind "web"

# Get the instrumentation key
$instrumentationKey = az monitor app-insights component show `
  --app "medevac-prod-insights" `
  --resource-group "rg-medevac-prod" `
  --query "instrumentationKey" --output tsv

Write-Host "Instrumentation Key: $instrumentationKey"
```

Step 17: Connect Services to Application Insights

```
# Add Application Insights to App Service
az webapp config appsettings set `
  --name "medevac-prod-api" `
  --resource-group "rg-medevac-prod" `
  --settings APPINSIGHTS_INSTRUMENTATIONKEY=$instrumentationKey

# Add Application Insights to Function App
az functionapp config appsettings set `
  --name "medevac-prod-functions" `
  --resource-group "rg-medevac-prod" `
  --settings APPINSIGHTS_INSTRUMENTATIONKEY=$instrumentationKey
```

🚀 Phase 7: Deploy Your Application Code (30 minutes)

Step 18: Deploy Backend API

```
# Navigate to your server directory
cd "c:\Users\VSACK\medevacform\server"

# Create a package.json if it doesn't exist or update it
@"
{
  "name": "medevac-api",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "express": "^4.18.0",
    "cors": "^2.8.5",
    "@azure/storage-blob": "^12.0.0",
    "@azure/keyvault-secrets": "^4.0.0",
    "@azure/identity": "^3.0.0",
    "axios": "^1.0.0",
    "cheerio": "^1.0.0"
  }
}
"@ | Out-File -FilePath "package.json" -Encoding UTF8

# Install dependencies
npm install

# Create deployment zip
Compress-Archive -Path ".\*" -DestinationPath "..\server-deploy.zip" -Force
```

```
# Deploy to App Service
az webapp deployment source config-zip `
  --name "medevac-prod-api" `
  --resource-group "rg-medevac-prod" `
  --src "..\server-deploy.zip"
```

Step 19: Deploy Functions (if you have function code)

```
# Navigate to your functions directory (create if it doesn't exist)
cd "c:\Users\VSACK\medevacform"
mkdir azure-functions -Force
cd azure-functions

# Create basic function structure
mkdir scraper-function -Force
cd scraper-function

# Create function.json
@"
{
  "bindings": [
    {
      "authLevel": "function",
      "type": "httpTrigger",
      "direction": "in",
      "name": "req",
      "methods": ["get", "post"]
    },
    {
      "type": "http",
      "direction": "out",
      "name": "res"
    }
  ]
}
"@ | Out-File -FilePath "function.json" -Encoding UTF8

# Create index.js (basic scraper function)
@"
const { DefaultAzureCredential } = require('@azure/identity');
const { BlobServiceClient } = require('@azure/storage-blob');

module.exports = async function (context, req) {
    context.log('Per diem scraper function triggered');

    const locationCode = req.params.locationCode || req.query.locationCode;

    if (!locationCode) {
        context.res = {
            status: 400,
```

```
            body: { error: 'Location code is required' }
        };
        return;
    }

    try {
        // Use managed identity to connect to storage
        const credential = new DefaultAzureCredential();
        const blobServiceClient = new BlobServiceClient(
            'https://medevacprodstorage.blob.core.windows.net',
            credential
        );

        // For now, return mock data
        const mockData = {
            locationCode: locationCode,
            rates: [{
                maxLodging: 200,
                mieRate: 75,
                maxPerDiem: 275,
                effectiveDate: new Date().toISOString()
            }],
            lastUpdated: new Date().toISOString()
        };

        context.res = {
            status: 200,
            body: mockData
        };
    } catch (error) {
        context.log.error('Error:', error);
        context.res = {
            status: 500,
            body: { error: 'Internal server error' }
        };
    }
};
"@ | Out-File -FilePath "index.js" -Encoding UTF8

# Go back to functions root and deploy
cd ..
func azure functionapp publish medevac-prod-functions --force
```

Step 20: Deploy Frontend via GitHub

```
# Navigate back to your main project
cd "c:\Users\VSACK\medevacform"

# Commit your changes to trigger deployment
git add .
```

```
git commit -m "Configure for Azure deployment"
git push origin main
```

☑ Phase 8: Verification & Testing (20 minutes)

Step 21: Test Your Deployment

```
# Test the App Service API
curl https://medevac-prod-api.azurewebsites.net/api/health

# Test the Function App
curl https://medevac-prod-functions.azurewebsites.net/api/scraper-function?
locationCode=11410

# Get your Static Web App URL
az staticwebapp show `
  --name "medevac-prod" `
  --resource-group "rg-medevac-prod" `
  --query "defaultHostname" --output tsv
```

Step 22: Configure Custom Domain (Optional)

```
# For Static Web App custom domain
az staticwebapp hostname set `
  --name "medevac-prod" `
  --resource-group "rg-medevac-prod" `
  --hostname "your-domain.com"

# For App Service custom domain
az webapp config hostname add `
  --webapp-name "medevac-prod-api" `
  --resource-group "rg-medevac-prod" `
  --hostname "api.your-domain.com"
```

🔧 Phase 9: Post-Deployment Configuration (15 minutes)

Step 23: Set Up Monitoring Alerts

```
# Create action group for notifications
az monitor action-group create `
  --name "medevac-alerts" `
  --resource-group "rg-medevac-prod" `
  --short-name "medevac" `
  --email-receivers name="admin" email-address="your-email@domain.com"
```

```
# Create alert rule for high error rate
az monitor metrics alert create `
  --name "High Error Rate" `
  --resource-group "rg-medevac-prod" `
  --scopes "/subscriptions/$(az account show --query id -o tsv)/resourceGroups/rg-medevac-prod/providers/Microsoft.Web/sites/medevac-prod-api" `
  --condition "avg requests/failed > 10" `
  --window-size "5m" `
  --evaluation-frequency "1m" `
  --action-groups "medevac-alerts"
```

Step 24: Set Up Backup Policies

```
# Enable backup for App Service (requires higher tier)
az webapp config backup update `
  --webapp-name "medevac-prod-api" `
  --resource-group "rg-medevac-prod" `
  --container-url "https://medevacprodstorage.blob.core.windows.net/backups" `
  --frequency 1440 `
  --retain-one true
```

📋 Final Checklist

☑ Deployment Verification

- ☐ Resource group created successfully
- ☐ Storage account and containers accessible
- ☐ Key Vault storing secrets correctly
- ☐ App Service running and responding
- ☐ Function App deployed and accessible
- ☐ Static Web App deployed from GitHub
- ☐ Application Insights collecting data
- ☐ All services can authenticate with managed identity
- ☐ Frontend can communicate with backend API
- ☐ Per diem scraping function works

☑ Security Checklist

- ☐ Managed identities configured for all services
- ☐ Secrets stored in Key Vault (not in app settings)
- ☐ HTTPS enforced on all endpoints
- ☐ CORS configured properly
- ☐ Access policies configured correctly
- ☐ No hardcoded secrets in code

☑ Monitoring Setup

- ☐ Application Insights configured
- ☐ Log Analytics workspace connected
- ☐ Alert rules created
- ☐ Notification channels configured
- ☐ Performance monitoring enabled

🆘 Troubleshooting Common Issues

Issue 1: App Service Not Starting

```
# Check logs
az webapp log tail --name "medevac-prod-api" --resource-group "rg-medevac-prod"

# Check configuration
az webapp config appsettings list --name "medevac-prod-api" --resource-group "rg-medevac-prod"
```

Issue 2: Key Vault Access Denied

```
# Verify managed identity is assigned
az webapp identity show --name "medevac-prod-api" --resource-group "rg-medevac-prod"

# Check Key Vault access policies
az keyvault show --name "kv-medevac-prod" --resource-group "rg-medevac-prod" --query "properties.accessPolicies"
```

Issue 3: Static Web App Not Deploying

```
# Check GitHub Actions workflow
# Go to your GitHub repo > Actions tab
# Look for workflow runs and check for errors

# Get deployment token for manual deployment
az staticwebapp secrets list --name "medevac-prod" --resource-group "rg-medevac-prod"
```

Issue 4: Function App Not Working

```
# Check Function App logs
az functionapp logs tail --name "medevac-prod-functions" --resource-group "rg-medevac-prod"
```

```
# Test function directly
curl -X POST "https://medevac-prod-functions.azurewebsites.net/api/scraper-function" -H "Content-Type: application/json" -d '{"locationCode": "11410"}'
```

📞 Support Resources

- **Azure Documentation**: https://docs.microsoft.com/azure/
- **Azure CLI Reference**: https://docs.microsoft.com/cli/azure/
- **Application Insights**: https://docs.microsoft.com/azure/azure-monitor/app/app-insights-overview
- **Azure Functions**: https://docs.microsoft.com/azure/azure-functions/
- **Static Web Apps**: https://docs.microsoft.com/azure/static-web-apps/

💰 Cost Management

Monitor your Azure spending:

```
# Check current costs
az consumption usage list --start-date "2024-12-01" --end-date "2024-12-03"

# Set up budget alerts
az consumption budget create --budget-name "medevac-monthly" --amount 300 --resource-group "rg-medevac-prod" --time-grain "Monthly"
```

Expected Monthly Costs (POC/MVP):

- App Service (Basic B1): ~$14/month
- Function App (Consumption): ~$0-5/month (pay per execution)
- Static Web Apps (Free): ~$0/month
- Storage Account (Hot): ~$10/month (small usage)
- Application Insights: ~$0/month (free tier)
- Key Vault: ~$3/month
- **Total: ~$27-32/month** 💰 (92% cost savings!)

🎉 **Congratulations!** Your MEDEVAC application is now deployed to Azure with enterprise-grade security, monitoring, and scalability!

Remember to:

1. **Test thoroughly** in your development environment first
2. **Monitor costs** regularly
3. **Keep secrets secure** and rotate them periodically
4. **Review logs** and metrics regularly
5. **Keep your dependencies updated**