# 🐳 Docker Deployment Guide - Azure AI Foundry Invoice Management System

## 📋 Overview

This guide provides comprehensive instructions for deploying the Azure AI Foundry Invoice Management System using Docker and Docker Compose.

---

## 🚀 Quick Start

### Prerequisites

- Docker Engine 20.10+
- Docker Compose 2.0+
- 4GB+ RAM available
- Azure account with required services

### 1. Clone and Setup

```
git clone <repository-url>
cd Azure_AI_Foundry

# Copy environment template
cp env.example .env

# Edit .env with your Azure credentials
nano .env
```

### 2. Start Basic Services

```
# Make build script executable
chmod +x docker-scripts/build.sh
```

```
# Start application with Redis
./docker-scripts/build.sh start
```

### 3. Access Application

- **Main Application**: http://localhost:8501
- **Redis GUI**: http://localhost:8081 (if monitoring enabled)

---

## 🏗️ Architecture Overview

### Services Included

| Service | Port | Description |
|---|---|---|
| `invoice-app` | 8501 | Main Streamlit application |
| `redis` | 6379 | Queue management and caching |
| `redis-commander` | 8081 | Redis GUI (monitoring profile) |
| `prometheus` | 9090 | Metrics collection (monitoring profile) |
| `grafana` | 3000 | Dashboards (monitoring profile) |

### Docker Network

- **Network**: `invoice-network` (172.20.0.0/16)
- **Type**: Bridge network with custom subnet
- **DNS**: Automatic service discovery

---

## ⚙️ Configuration

### Environment Variables

#### Required Azure Configuration

```
# Azure CosmosDB
AZURE_COSMOS_ENDPOINT=https://your-cosmos.documents.azure.com:443/
AZURE_COSMOS_KEY=your-cosmos-key
```

```
AZURE_COSMOS_DATABASE=InvoicesDB
AZURE_COSMOS_CONTAINER=container2

# Azure AI Search
AZURE_SEARCH_ENDPOINT=https://your-search.search.windows.net
AZURE_SEARCH_KEY=your-search-key
AZURE_SEARCH_INDEX=container2

# Azure AI Project
AZURE_AI_PROJECT_CONNECTION_STRING=your-ai-project-connection

# Azure Blob Storage
AZURE_STORAGE_CONNECTION_STRING=your-storage-connection
AZURE_STORAGE_CONTAINER=invoices
```

### Optional Configuration

```
# Redis (automatically configured for Docker)
REDIS_URL=redis://redis:6379

# Performance
MAX_WORKERS=3
CACHE_TTL=300

# Monitoring
GRAFANA_PASSWORD=admin123
LOG_LEVEL=INFO
```

---

## 🛠️ Build Scripts Usage

### Available Commands

```
# Build Docker image only
./docker-scripts/build.sh build

# Build with docker-compose
./docker-scripts/build.sh build-compose

# Start basic services (app + redis)
./docker-scripts/build.sh start

# Start with full monitoring stack
./docker-scripts/build.sh start-monitoring
```

```
# Stop all services
<span id="instant-markdown-cursor"></span>
./docker-scripts/build.sh stop

# Restart services
./docker-scripts/build.sh restart

# View logs
./docker-scripts/build.sh logs
./docker-scripts/build.sh logs invoice-app

# Check health
./docker-scripts/build.sh health

# Clean up everything
./docker-scripts/build.sh cleanup
```

### Manual Docker Compose Commands

```
# Basic deployment
docker-compose up -d

# With monitoring
docker-compose --profile monitoring up -d

# View logs
docker-compose logs -f invoice-app

# Scale workers (if needed)
docker-compose up -d --scale invoice-app=2

# Stop services
docker-compose down

# Clean up with volumes
docker-compose down -v --remove-orphans
```

---

## 📊 Monitoring Setup

### Enable Full Monitoring Stack

```
./docker-scripts/build.sh start-monitoring
```

### Access Monitoring Services

#### Grafana Dashboard

- **URL**: http://localhost:3000
- **Username**: admin
- **Password**: admin123 (configurable via `GRAFANA_PASSWORD`)

#### Prometheus Metrics

- **URL**: http://localhost:9090
- **Targets**: http://localhost:9090/targets

#### Redis Commander

- **URL**: http://localhost:8081
- **Connection**: Automatic to Redis container

### Available Metrics

- Invoice generation performance
- Queue statistics
- Cache hit rates
- System health scores
- Error rates and response times

---

## 🔧 Development Setup

### Local Development with Docker

```
# Build development image
docker build --target dependencies -t azure-invoice-dev .

# Run development container with volume mounts
docker run -it --rm \
  -p 8501:8501 \
  -v $(pwd):/app \
  -v $(pwd)/.env:/app/.env \
  azure-invoice-dev \
  streamlit run app.py --server.port=8501 --server.address=0.0.0.0
```

### Development with Docker Compose

```
# docker-compose.dev.yml
version: '3.8'
services:
  invoice-app-dev:
    build:
      context: .
      target: dependencies
    volumes:
      - .:/app
      - ./.env:/app/.env
    ports:
      - "8501:8501"
    environment:
      - REDIS_URL=redis://redis:6379
    depends_on:
      - redis
    command: streamlit run app.py --server.port=8501 --server.address=0.0.0.
```

## 🔒 Security Considerations

### Production Security

#### 1. Environment Variables

```
# Use Docker secrets in production
echo "your-cosmos-key" | docker secret create cosmos_key -
echo "your-search-key" | docker secret create search_key -
```

#### 2. Network Security

```
# Restrict external access
services:
  invoice-app:
    ports:
      - "127.0.0.1:8501:8501"  # Bind to localhost only
```

#### 3. User Permissions

- Application runs as non-root user (`appuser`)
- Read-only filesystem where possible
- Minimal system dependencies

#### 4. Secrets Management

```
# Use Azure Key Vault in production
# Mount secrets as files instead of environment variables
volumes:
  - /var/secrets/cosmos-key:/run/secrets/cosmos-key:ro
```

---

## 📈 Performance Optimization

### Resource Limits

```
services:
  invoice-app:
    deploy:
      resources:
        limits:
          cpus: '2.0'
          memory: 4G
        reservations:
          cpus: '1.0'
          memory: 2G
```

### Redis Optimization

```
redis:
  command: redis-server --appendonly yes --maxmemory 1gb --maxmemory-policy
```

### Multi-stage Build Benefits

- **Smaller image size**: ~500MB vs 1.5GB
- **Faster deployments**: Cached dependency layers
- **Security**: No build tools in production image

---

## 🚀 Production Deployment

### Azure Container Instances

```
# Create resource group
az group create --name invoice-app-rg --location eastus

# Deploy container
az container create \
  --resource-group invoice-app-rg \
  --name azure-invoice-app \
  --image your-registry/azure-invoice-app:latest \
  --ports 8501 \
  --environment-variables \
    AZURE_COSMOS_ENDPOINT=$AZURE_COSMOS_ENDPOINT \
    AZURE_COSMOS_KEY=$AZURE_COSMOS_KEY \
  --cpu 2 \
  --memory 4
```

### Azure Container Apps

```
# container-app.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-invoice-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: azure-invoice-app
  template:
    metadata:
      labels:
        app: azure-invoice-app
    spec:
      containers:
      - name: invoice-app
        image: your-registry/azure-invoice-app:latest
        ports:
        - containerPort: 8501
        env:
        - name: AZURE_COSMOS_ENDPOINT
          valueFrom:
            secretKeyRef:
              name: azure-secrets
              key: cosmos-endpoint
```

### Docker Swarm

```
# Initialize swarm
docker swarm init

# Deploy stack
docker stack deploy -c docker-compose.yml invoice-stack
```

---

## 🔍 Troubleshooting

### Common Issues

#### 1. Application Won't Start

```
# Check logs
./docker-scripts/build.sh logs invoice-app

# Common causes:
# - Missing .env file
# - Invalid Azure credentials
# - Port conflicts
```

#### 2. Redis Connection Issues

```
# Check Redis health
docker exec azure-invoice-redis redis-cli ping

# Check network connectivity
docker exec azure-invoice-app ping redis
```

#### 3. Performance Issues

```
# Check resource usage
docker stats

# Check application metrics
curl http://localhost:8501/metrics
```

#### 4. Build Failures

```
# Clean build cache
docker builder prune

# Rebuild without cache
docker-compose build --no-cache
```

### Health Checks

```
# Application health
curl -f http://localhost:8501/_stcore/health

# Redis health
docker exec azure-invoice-redis redis-cli ping

# Full system health
./docker-scripts/build.sh health
```

---

## 📝 Maintenance

### Updates and Upgrades

```
# Pull latest images
docker-compose pull

# Rebuild and restart
./docker-scripts/build.sh stop
./docker-scripts/build.sh build-compose
./docker-scripts/build.sh start
```

### Backup and Restore

```
# Backup Redis data
docker exec azure-invoice-redis redis-cli BGSAVE
docker cp azure-invoice-redis:/data/dump.rdb ./backup/

# Backup volumes
docker run --rm -v invoice_redis-data:/data -v $(pwd)/backup:/backup alpine
```

### Log Management

```
# Rotate logs
docker-compose logs --no-color > logs/app-$(date +%Y%m%d).log

# Clean old logs
docker system prune -f
```

---

## 🎯 Best Practices

### 1. Image Management

- Use specific tags, not `latest`
- Implement multi-stage builds
- Regular security scanning

### 2. Configuration

- Use environment-specific configs
- Never commit secrets to version control
- Use Docker secrets for sensitive data

### 3. Monitoring

- Enable health checks
- Monitor resource usage
- Set up alerting

### 4. Backup Strategy

- Regular data backups

- Test restore procedures
- Document recovery processes

---

## 📞 Support

### Getting Help

- Check logs: `./docker-scripts/build.sh logs`
- Health check: `./docker-scripts/build.sh health`
- Clean restart: `./docker-scripts/build.sh cleanup && ./docker-scripts/build.sh start`

### Reporting Issues

Include the following information:

- Docker version: `docker --version`
- Compose version: `docker-compose --version`
- System resources: `docker system df`
- Application logs: `./docker-scripts/build.sh logs`

---

🎉 **Your Azure AI Foundry Invoice Management System is now fully containerized and ready for production deployment!**