Next Steps Implementation Guide

DevOps Requirements - Detailed Implementation Plan

Based on the "next steps of the assignment.pdf" requirements, this document outlines exactly what needs to be implemented.

Overview of Requirements

The DevOps phase requires:

1. **Dockerfile** (✅ Done) - Expose port 8080, include .dockerignore
2. **Environment Variables** (✅ Done) - MONGODB_URI, credentials, .env.example
3. **Terraform** - Resource group, compute, networking for /rest accessibility
4. **GitHub Actions** - Test on PR, deploy on push, build/push Docker image, terraform apply
5. **Data Validation** - restaurants.json automation (✅ Done - skip invalid rows)

---

1. Terraform Infrastructure (TO DO)

1.1 Directory Structure

Create the following structure:

```
terraform/
├── main.tf           # Main configuration
├── variables.tf       # Input variables
├── outputs.tf         # Output values
├── providers.tf        # Provider configuration
├── container.tf        # Container Instance resource
├── networking.tf       # Virtual network, NSG
└── terraform.tfvars.example  # Example variable values
```

1.2 providers.tf

```hcl
terraform {
  required_version = ">= 1.5.0"

  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 3.80"
    }
  }

  # Local backend (per company clarification)
  backend "local" {
    path = "terraform.tfstate"
  }
}

provider "azurerm" {
  features {}

  # These will come from environment variables or GitHub Secrets:
  # ARM_SUBSCRIPTION_ID
  # ARM_TENANT_ID
  # ARM_CLIENT_ID
  # ARM_CLIENT_SECRET
}
```

1.3 variables.tf

```hcl
variable "resource_group_name" {
  description = "Name of the Azure resource group"
  type        = string
  default     = "rg-restaurant-recommendation"
}

variable "location" {
  description = "Azure region"
  type        = string
  default     = "West Europe"  # Or your preferred region
}

variable "container_name" {
  description = "Name of the container instance"
  type        = string
  default     = "restaurant-api"
}

variable "docker_image" {
  description = "Docker image to deploy"
  type        = string
  # Will be set by CI/CD: ghcr.io/<owner>/restaurant-recommendation:latest
}

variable "mongodb_uri" {
  description = "MongoDB Atlas connection string"
  type        = string
  sensitive   = true
}

variable "cpu_cores" {
  description = "CPU cores for container"
  type        = number
  default     = 1
}

variable "memory_gb" {
  description = "Memory in GB for container"
  type        = number
  default     = 1.5
}

variable "environment" {
  description = "Environment name (dev, staging, prod)"
  type        = string
  default     = "dev"
}
```

1.4 main.tf

hcl

```hcl
# Resource Group
resource "azurerm_resource_group" "main" {
  name     = var.resource_group_name
  location = var.location

  tags = {
    environment = var.environment
    project     = "restaurant-recommendation"
  }
}

# Container Instance
resource "azurerm_container_group" "api" {
  name                = var.container_name
  location            = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
  os_type             = "Linux"
  ip_address_type     = "Public"
  dns_name_label      = "${var.container_name}-${var.environment}"

  container {
    name   = "restaurant-api"
    image  = var.docker_image
    cpu    = var.cpu_cores
    memory = var.memory_gb

    ports {
      port     = 8080
      protocol = "TCP"
    }

    environment_variables = {
      "LOG_LEVEL"  = "INFO"
      "LOG_FORMAT" = "json"
    }

    secure_environment_variables = {
      "MONGODB_URI" = var.mongodb_uri
    }

    liveness_probe {
      http_get {
        path   = "/health"
        port   = 8080
        scheme = "Http"
      }
      initial_delay_seconds = 30
      period_seconds       = 10
      failure_threshold     = 3
    }

    readiness_probe {
      http_get {
        path   = "/health"
        port   = 8080
        scheme = "Http"
      }
      initial_delay_seconds = 10
      period_seconds       = 5
    }
  }

  tags = {
    environment = var.environment
    project     = "restaurant-recommendation"
  }
}
```

1.5 outputs.tf

```hcl
output "api_url" {
  description = "Public URL for the REST API"
  value       = "http://${azurerm_container_group.api.fqdn}:8080"
}

output "api_fqdn" {
  description = "Fully qualified domain name"
  value       = azurerm_container_group.api.fqdn
}

output "api_ip_address" {
  description = "Public IP address"
  value       = azurerm_container_group.api.ip_address
}

output "resource_group_name" {
  description = "Resource group name"
  value       = azurerm_resource_group.main.name
}

output "rest_endpoint" {
  description = "The /rest endpoint URL"
  value       = "http://${azurerm_container_group.api.fqdn}:8080/rest"
}
```

1.6 terraform.tfvars.example

```hcl
# Copy to terraform.tfvars and fill in values
# DO NOT commit terraform.tfvars to git!

resource_group_name = "rg-restaurant-recommendation"
location            = "West Europe"
container_name      = "restaurant-api"
environment         = "dev"

# Docker image (set by CI/CD or manually)
docker_image = "ghcr.io/YOUR_USERNAME/restaurant-recommendation:latest"

# MongoDB Atlas connection (sensitive - use environment variable in CI)
# mongodb_uri = "mongodb+srv://user:pass@cluster.mongodb.net/restaurant_db"
```

---

2. GitHub Actions Workflows (TO DO)

2.1 Directory Structure

```
.github/
└── workflows/
    ├── ci.yml          # Test on PR
    ├── cd.yml          # Deploy on push to main
    └── validate.yml    # Validate restaurants.json
```

2.2 .github/workflows/ci.yml (Test on PR)

yaml

```yaml
name: CI - Test

on:
  pull_request:
    branches: [main]
  push:
    branches: [main]

jobs:
  test:
    name: Run Tests
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.11'
          cache: 'pip'

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install -r requirements.txt

      - name: Run unit tests
        run: pytest tests/ -v --tb=short

      - name: Lint with flake8 (optional)
        run: |
          pip install flake8
          flake8 app/ --max-line-length=120 --ignore=E501
        continue-on-error: true

  validate-data:
    name: Validate restaurants.json
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.11'

      - name: Install dependencies
        run: pip install pydantic

      - name: Validate JSON schema
        run: |
          python -c "
          import json
          from pathlib import Path

          # Load and validate JSON syntax
          data = json.loads(Path('restaurants/restaurants.json').read_text())

          # Check it's a list
          assert isinstance(data, list), 'Expected array'

          # Check each entry has required fields
          required = {'name', 'style', 'address', 'vegetarian', 'openHour', 'closeHour'}
          for i, entry in enumerate(data):
            fields = set(entry.keys())
            missing = required - fields
            extra = fields - required
            if missing or extra:
              print(f'Entry {i}: missing={missing}, extra={extra}')
              # Note: We skip invalid, don't fail (per spec)

          print(f'Validated {len(data)} entries')
          "

  docker-build:
    name: Build Docker Image
    runs-on: ubuntu-latest
    needs: [test]

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Build image (test only)
        uses: docker/build-push-action@v5
        with:
          context: .
          push: false
          tags: restaurant-recommendation:test
          cache-from: type=gha
```

```
          cache-from: type=gha
          cache-to: type=gha,mode=max
```

2.3 .github/workflows/cd.yml (Deploy on Push to Main)

yaml

```yaml
name: CD - Deploy

on:
  push:
    branches: [main]
  workflow_dispatch:  # Allow manual trigger

env:
  REGISTRY: ghcr.io
  IMAGE_NAME: ${{ github.repository }}

jobs:
  build-and-push:
    name: Build & Push Docker Image
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write

    outputs:
      image_tag: ${{ steps.meta.outputs.tags }}

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Log in to GitHub Container Registry
        uses: docker/login-action@v3
        with:
          registry: ${{ env.REGISTRY }}
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Extract metadata
        id: meta
        uses: docker/metadata-action@v5
        with:
          images: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}
          tags: |
            type=sha,prefix=
            type=raw,value=latest

      - name: Build and push
        uses: docker/build-push-action@v5
      with:
        context: .
        push: true
        tags: ${{ steps.meta.outputs.tags }}
        labels: ${{ steps.meta.outputs.labels }}
        cache-from: type=gha
        cache-to: type=gha,mode=max

  deploy:
    name: Deploy to Azure
    runs-on: ubuntu-latest
    needs: [build-and-push]
    environment: production

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v3
        with:
          terraform_version: 1.6.0

      - name: Azure Login
        uses: azure/login@v1
        with:
          creds: ${{ secrets.AZURE_CREDENTIALS }}

      - name: Terraform Init
        working-directory: terraform
        run: terraform init

      - name: Terraform Plan
        working-directory: terraform
        run: |
          terraform plan \
            -var="docker_image=${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:latest" \
            -var="mongodb_uri=${{ secrets.MONGODB_URI }}" \
            -out=tfplan

      - name: Terraform Apply
        working-directory: terraform
        run: terraform apply -auto-approve tfplan

      - name: Output API URL
        working-directory: terraform
        run: |
          echo "API URL: $(terraform output -raw api_url)"
          echo "REST Endpoint: $(terraform output -raw rest_endpoint)"
  smoke-test:
    name: Smoke Test
    runs-on: ubuntu-latest
    needs: [deploy]

    steps:
      - name: Wait for deployment
        run: sleep 60

      - name: Test health endpoint
        run: |
          API_URL="${{ needs.deploy.outputs.api_url }}"
          curl -f "${API_URL}/health" || exit 1

      - name: Test REST endpoint
        run: |
          API_URL="${{ needs.deploy.outputs.api_url }}"
          curl -f "${API_URL}/rest?query=italian" || exit 1
```

2.4 .github/workflows/validate.yml (Optional - Dedicated Validation)

yaml

```yaml
name: Validate Data

on:
  push:
    paths:
      - 'restaurants/restaurants.json'
  pull_request:
    paths:
      - 'restaurants/restaurants.json'

jobs:
  validate:
    name: Validate restaurants.json
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.11'

      - name: Install dependencies
        run: pip install pydantic

      - name: Run validation
        run: |
          python -c "
          import json
          import sys

          required = {'name', 'style', 'address', 'vegetarian', 'openHour', 'closeHour'}
          valid_veg = {'yes', 'no'}

          with open('restaurants/restaurants.json') as f:
              data = json.load(f)

          invalid_count = 0
          for i, entry in enumerate(data):
              errors = []

              # Check fields
              fields = set(entry.keys())
              if missing := required - fields:
                  errors.append(f'missing: {missing}')
              if extra := fields - required:
                  errors.append(f'extra: {extra}')

              # Check vegetarian value
              if entry.get('vegetarian', '').lower() not in valid_veg:
                  errors.append(f'vegetarian must be yes/no')

              if errors:
                  print(f'Entry {i} ({entry.get(\"name\", \"unknown\")}): {errors}')
                  invalid_count += 1

          print(f'\\nTotal: {len(data)} entries, {invalid_count} invalid')

          # Note: We don't fail on invalid entries (per spec: skip invalid rows)
          if invalid_count > 0:
              print('WARNING: Some entries are invalid and will be skipped during loading')
          "
```

3. Required GitHub Secrets

Configure these in GitHub Repository Settings → Secrets and Variables → Actions:

Secret Name	Description	How to Get
`MONGODB_URI`	MongoDB Atlas connection string	Atlas → Connect → Drivers → Copy
`AZURE_CREDENTIALS`	Azure service principal JSON	See below
`AZURE_SUBSCRIPTION_ID`	Azure subscription ID	Azure Portal

Creating Azure Service Principal

```bash
# Create service principal with Contributor role
az ad sp create-for-rbac \
  --name "github-restaurant-app" \
  --role Contributor \
  --scopes /subscriptions/<SUBSCRIPTION_ID> \
  --sdk-auth

# This outputs JSON - copy entire output to AZURE_CREDENTIALS secret
```

Alternative: Individual Azure Secrets

Instead of `AZURE_CREDENTIALS`, you can use:

- `ARM_CLIENT_ID`
- `ARM_CLIENT_SECRET`
- `ARM_SUBSCRIPTION_ID`
- `ARM_TENANT_ID`

---

4. Implementation Checklist

Phase 1: Terraform Setup

- [ ] Create `terraform/` directory
- [ ] Create `providers.tf` with Azure provider
- [ ] Create `variables.tf` with all inputs
- [ ] Create `main.tf` with resource group and container
- [ ] Create `outputs.tf` with API URL outputs
- [ ] Create `terraform.tfvars.example`
- [ ] Add `terraform/` to `.gitignore` for tfvars and tfstate
- [ ] Test locally with `terraform plan`

Phase 2: GitHub Actions

- [ ] Create `.github/workflows/` directory
- [ ] Create `ci.yml` for PR testing
- [ ] Create `cd.yml` for deployment
- [ ] Configure GitHub Secrets
- [ ] Test workflow with a PR

Phase 3: Integration Testing

- [ ] Push to main to trigger deployment
- [ ] Verify container is running in Azure
- [ ] Test `/health` endpoint
- [ ] Test `/rest?query=...` endpoint
- [ ] Verify logging works

Phase 4: Documentation

- [ ] Update README with deployment instructions
- [ ] Document GitHub Secrets requirements
- [ ] Add architecture diagram with Azure

---

5. File Updates Needed

.gitignore additions

````gitignore
# Terraform
terraform/.terraform/
terraform/*.tfstate
terraform/*.tfstate.*
terraform/.terraform.lock.hcl
terraform/terraform.tfvars
terraform/tfplan

# Local environment
.env
*.pem
*.key
````

README.md additions

Add section for production deployment:

````markdown
## Production Deployment

### Prerequisites
- Azure subscription
- GitHub repository with Actions enabled
- MongoDB Atlas cluster

### GitHub Secrets Required
- `MONGODB_URI`: MongoDB Atlas connection string
- `AZURE_CREDENTIALS`: Azure service principal JSON

### Deploy
1. Push to `main` branch
2. GitHub Actions builds and pushes Docker image
3. Terraform applies infrastructure changes
4. Container Instance starts with new image

### Manual Terraform
\`\`\`bash
cd terraform
terraform init
terraform plan -var="mongodb_uri=$MONGODB_URI" -var="docker_image=ghcr.io/user/repo:latest"
terraform apply
\`\`\`
````

---

6. Estimated Effort

Task	Estimated Time
Terraform files	2-3 hours
GitHub Actions workflows	2-3 hours
Testing & debugging	2-4 hours
Documentation updates	1 hour
Total	**7-11 hours**

---

7. Questions to Confirm Before Starting

1. **Azure Region**: Which region to deploy? (affects latency)
2. **Container Size**: 1 CPU / 1.5GB enough?
3. **Domain**: Custom domain or Azure-provided FQDN?
4. **SSL/TLS**: Required for production?
5. **Scaling**: Single instance or multiple?
6. **Monitoring**: Azure Monitor integration needed?