Deployment Guide

DevOps Best Practices for Students

State University of Zanzibar (SUZA)

BSc Computer Science

Contents

1 Introduction

This guide covers deployment strategies and DevOps practices for student software development projects. It includes instructions for various hosting platforms and deployment methods.

2 Pre-Deployment Checklist

2.1 Code Preparation

All features are implemented and tested

Code has been reviewed

All tests pass

No hardcoded credentials or secrets

Environment variables are configured

README is updated

Dependencies are up to date

2.2 Security Checklist

HTTPS is configured

Passwords are hashed

SQL injection prevention

XSS protection enabled

CORS properly configured

Rate limiting implemented

3 Environment Configuration

3.1 Environment Variables

Store sensitive configuration in environment variables, not in code.

Example .env file:

```
# Database
DATABASE_URL=postgresql://user:pass@host:5432/db
DB_HOST=localhost
DB_PORT=5432
DB_NAME=myapp
DB_USER=admin
DB_PASSWORD=secretpassword
```

```
# Application
NODE_ENV=production
PORT=3000
SECRET_KEY=your-secret-key

# External APIs
API_KEY=your-api-key
```

Important: Never commit .env files to Git. Add to .gitignore:

```
.env
.env.local
.env.production
```

4 Deployment Options

4.1 Option 1: Vercel (Frontend/Full-Stack)

Best for: React, Next.js, Vue, static sites

Steps:

1. Create account at https://vercel.com
2. Connect GitHub repository
3. Configure build settings:

   ```
   Build Command: npm run build
   Output Directory: dist (or build)
   Install Command: npm install
   ```

4. Add environment variables in dashboard
5. Deploy

4.2 Option 2: Heroku (Full-Stack)

Best for: Node.js, Python, Java backends

Steps:

1. Install Heroku CLI

   ```
   # macOS
   brew install heroku/brew/heroku

   # Ubuntu
   sudo snap install heroku --classic
   ```

2. Login and create app

```
heroku login
heroku create your-app-name
```

3. Add Procfile to project root

```
web: npm start
# or for Python
web: gunicorn app:app
```

4. Configure environment variables

```
heroku config:set DATABASE_URL=your-db-url
heroku config:set SECRET_KEY=your-secret
```

5. Deploy

```
git push heroku main
```

4.3 Option 3: Railway

Best for: Databases, backends, full-stack apps
Steps:

1. Create account at https://railway.app
2. Create new project
3. Connect GitHub repository
4. Add database service if needed (PostgreSQL, MySQL)
5. Configure environment variables
6. Deploy automatically on push

4.4 Option 4: GitHub Pages (Static Sites)

Best for: HTML/CSS/JS sites, documentation
Steps:

1. Go to repository Settings
2. Navigate to Pages section
3. Select source branch (main or gh-pages)
4. Select folder (root or /docs)
5. Save and wait for deployment

For React/Vue builds:

```
# Install gh-pages
npm install gh-pages --save-dev

# Add to package.json scripts
"predeploy": "npm run build",
"deploy": "gh-pages -d build"

# Deploy
npm run deploy
```

5 Docker Deployment

5.1 Creating a Dockerfile

Node.js Example:

```
# Use official Node.js image
FROM node:18-alpine

# Set working directory
WORKDIR /app

# Copy package files
COPY package*.json ./

# Install dependencies
RUN npm ci --only=production

# Copy source code
COPY . .

# Expose port
EXPOSE 3000

# Start application
CMD ["npm", "start"]
```

Python Example:

```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5000
```

```
CMD ["python", "app.py"]
```

5.2 Docker Commands

```
# Build image
docker build -t myapp .

# Run container
docker run -p 3000:3000 myapp

# Run with environment variables
docker run -p 3000:3000 --env-file .env myapp

# List containers
docker ps

# Stop container
docker stop container_id
```

5.3 Docker Compose

docker-compose.yml:

```
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=postgres://user:pass@db:5432/mydb
    depends_on:
      - db

  db:
    image: postgres:15
    environment:
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=pass
      - POSTGRES_DB=mydb
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```

```
# Start services
docker-compose up -d

# Stop services
```

```
docker-compose down

# View logs
docker-compose logs -f
```

6 Database Deployment

6.1 PostgreSQL on Railway/Heroku

1. Add PostgreSQL addon
2. Get DATABASE_URL from dashboard
3. Update application to use DATABASE_URL

6.2 MongoDB Atlas (Free Tier)

1. Create account at https://www.mongodb.com/atlas
2. Create free cluster
3. Whitelist IP addresses (or 0.0.0.0/0 for development)
4. Create database user
5. Get connection string
6. Update MONGODB_URI in environment

7 CI/CD with GitHub Actions

7.1 Basic Workflow

Create `.github/workflows/deploy.yml`:

```
name: Deploy

on:
  push:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
      - run: npm ci
      - run: npm test
  deploy:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Deploy to Heroku
        uses: akhileshns/heroku-deploy@v3.12.14
        with:
          heroku_api_key: ${{secrets.HEROKU_API_KEY}}
          heroku_app_name: "your-app-name"
          heroku_email: "your-email@example.com"
```

8 Post-Deployment

8.1 Monitoring

- Check application logs regularly
- Set up uptime monitoring (UptimeRobot, Pingdom)
- Monitor error rates

8.2 Rollback Procedure

```
# Heroku rollback
heroku rollback v123

# Git-based rollback
git revert HEAD
git push origin main
```

9 Deployment Checklist Template

Task	**Status**	**Notes**
Code merged to main		
All tests passing		
Environment variables set		
Database migrated		
Deployment successful		
Application accessible		
Basic functionality verified		
Team notified		

10 Common Issues and Solutions

Issue	**Solution**
Build fails	Check build logs, ensure dependencies are in package.json
App crashes on start	Check for missing environment variables
Database connection fails	Verify DATABASE_URL and network access
Port already in use	Use process.env.PORT for dynamic port
CORS errors	Configure CORS middleware properly