DevOps Tutorial - Quick Guide

🚀 What is DevOps?

DevOps = **Dev**elopment + **Op**erations

A culture and set of practices that combines software development and IT operations to:

- **Shorten development cycles**
- **Increase deployment frequency**
- **Deliver reliable releases**
- **Improve collaboration**

🔄 DevOps Lifecycle

```
Plan → Code → Build → Test → Release → Deploy → Operate → Monitor
  ↑                                                        ↓
  ←←←←←←←←←←←← Feedback Loop ←←←←←←←←←←←←←←←←←←←←←←←←←←←←
```

🛠️ Core DevOps Tools & Practices

1. Version Control

- **Git** (GitHub, GitLab, Azure DevOps)
- **Branching strategies** (GitFlow, GitHub Flow)
- **Pull/Merge requests** for code review

2. Continuous Integration (CI)

- **Automated builds** on code commits
- **Automated testing** (unit, integration, E2E)
- **Code quality checks** (linting, security scans)

Popular CI Tools:

- GitHub Actions
- Azure DevOps Pipelines
- Jenkins
- GitLab CI/CD

3. Continuous Deployment (CD)

- **Automated deployments** to environments
- **Environment promotion** (Dev → Test → Prod)
- **Rollback capabilities**

4. Infrastructure as Code (IaC)

- **Terraform** - Multi-cloud infrastructure
- **ARM Templates** - Azure resources
- **CloudFormation** - AWS resources
- **Ansible** - Configuration management

5. Containerization

- **Docker** - Application containerization
- **Kubernetes** - Container orchestration
- **Docker Compose** - Multi-container applications

6. Monitoring & Logging

- **Application monitoring** (Application Insights, New Relic)
- **Infrastructure monitoring** (Prometheus, Grafana)
- **Log aggregation** (ELK Stack, Splunk)

📋 DevOps Pipeline Example

Simple CI/CD Pipeline:

```yaml
# GitHub Actions example
name: CI/CD Pipeline

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

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'
      - name: Install dependencies
        run: npm install
      - name: Run tests
        run: npm test
      - name: Run linting
        run: npm run lint

  deploy:
    needs: test
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'
    steps:
      - name: Deploy to production
        run: echo "Deploying to production..."
```

🎯 DevOps Benefits

For Development Teams:

- ✅ **Faster feedback** on code changes

- ✅ **Automated testing** catches bugs early
- ✅ **Consistent environments** (dev = prod)
- ✅ **Reduced manual work**

For Operations Teams:

- ✅ **Predictable deployments**
- ✅ **Better monitoring** and alerting
- ✅ **Infrastructure automation**
- ✅ **Faster incident response**

For Business:

- ✅ **Faster time to market**
- ✅ **Higher quality software**
- ✅ **Reduced downtime**
- ✅ **Better customer satisfaction**

🚀 Getting Started with DevOps

Step 1: Version Control

```
# Initialize Git repository
git init
git add .
git commit -m "Initial commit"
git remote add origin <repository-url>
git push -u origin main
```

Step 2: Automated Testing

```
# Frontend tests
npm test

# Backend tests
dotnet test

# Add to CI pipeline
```

Step 3: Basic CI Pipeline

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

```
name: CI
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Run tests
```

```
        run: |
          npm install
          npm test
```

Step 4: Containerization

Create `Dockerfile`:

```
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

Step 5: Deployment

```
# Build and deploy container
docker build -t myapp .
docker run -p 3000:3000 myapp
```

📊 DevOps Metrics

Key Performance Indicators (KPIs):

1. **Deployment Frequency** - How often you deploy
2. **Lead Time** - Time from code commit to production
3. **Mean Time to Recovery (MTTR)** - Time to fix issues
4. **Change Failure Rate** - % of deployments causing failures

Example Targets:

- 🎯 **Deploy multiple times per day**
- 🎯 **Lead time < 1 hour**
- 🎯 **MTTR < 1 hour**
- 🎯 **Change failure rate < 15%**

🔧 DevOps for Your Project

Current State Analysis:

Your TextSubmissionAPI project already has:

- ✅ **Version Control** (Git)
- ✅ **Automated Testing** (89 tests)
- ✅ **Frontend Tests** (Angular/Jasmine)
- ✅ **Backend Tests** (.NET/xUnit)

Next DevOps Steps:

1. Add CI Pipeline

```yaml
# .github/workflows/ci.yml
name: CI/CD
on: [push, pull_request]
jobs:
  frontend-tests:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
      - run: |
          cd frontend/text-submission-app
          npm install
          ng test --watch=false --browsers=ChromeHeadless

  backend-tests:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-dotnet@v3
      - run: |
          cd backend
          dotnet test TextSubmissionAPI.Tests
```

2. Containerize Applications

```dockerfile
# Frontend Dockerfile
FROM node:18-alpine
WORKDIR /app
COPY frontend/text-submission-app .
RUN npm install && ng build
EXPOSE 4200
CMD ["ng", "serve", "--host", "0.0.0.0"]
```

```dockerfile
# Backend Dockerfile
FROM mcr.microsoft.com/dotnet/aspnet:9.0
WORKDIR /app
COPY backend/TextSubmissionAPI/bin/Release/net9.0/publish .
EXPOSE 80
ENTRYPOINT ["dotnet", "TextSubmissionAPI.dll"]
```

3. Add Docker Compose

```yaml
# docker-compose.yml
version: '3.8'
services:
  frontend:
    build: ./frontend
    ports:
```

```
      - "4200:4200"

  backend:
    build: ./backend
    ports:
      - "5000:80"
    environment:
      - ASPNETCORE_ENVIRONMENT=Production

  database:
    image: mcr.microsoft.com/mssql/server:2022-latest
    environment:
      - ACCEPT_EULA=Y
      - SA_PASSWORD=YourPassword123!
    ports:
      - "1433:1433"
```

🎓 Learning Path

Beginner (1-2 months):

1. Master Git workflows
2. Set up basic CI with GitHub Actions
3. Learn Docker basics
4. Implement automated testing

Intermediate (3-6 months):

1. Infrastructure as Code (Terraform)
2. Kubernetes fundamentals
3. Monitoring and logging
4. Security scanning

Advanced (6+ months):

1. Multi-cloud strategies
2. Advanced orchestration
3. Site Reliability Engineering (SRE)
4. DevSecOps practices

🔗 Useful Resources

- **Documentation:** GitHub Actions, Docker, Kubernetes
- **Learning:** Azure DevOps Labs, Katacoda
- **Tools:** DevOps Roadmap, CNCF Landscape

DevOps is a journey, not a destination. Start small, automate incrementally, and focus on continuous improvement! 🚀