🚀 Telecom Demo - Complete Startup & Execution Guide

Pre-Flight Checklist ✅

CRITICAL: Complete this checklist BEFORE attempting the demo to ensure 100% success rate.

1. Docker Desktop Status

```
# Check Docker is running
docker --version
docker ps

# Expected: Version output and running containers
# ❌ Error: "Cannot connect to Docker daemon" → Start Docker Desktop
```

2. Kubernetes Cluster Health

```
# Verify Kubernetes is enabled and running
kubectl get nodes

# Expected: STATUS = Ready
# ❌ Error: "connection refused" → Enable Kubernetes in Docker Desktop
```

3. Core Services Status

```
# Check all pods are running
kubectl get pods -n ecommerce-ai

# Expected: All pods show STATUS = Running
# ❌ Any pod showing "Pending/Error/CrashLoopBackOff" → Restart that service
```

Complete Startup Sequence 🔧

Follow this sequence **exactly** - each step depends on the previous ones.

STEP 1: Start Docker Desktop 🐳

```
# macOS: Open Docker Desktop application
open -a Docker
```

```
# Wait for Docker to fully start (green status icon)
# Verify with:
docker ps
```

What this does: Starts the container runtime that hosts Kubernetes and all services.

Common Issues:

- Docker Desktop not installed → Download from docker.com
- "Docker Desktop starting..." → Wait 2-3 minutes for full startup
- Port conflicts → Close other applications using ports 7687, 9092

STEP 2: Verify Kubernetes Services ☸️

```
# Check all services are running
kubectl get pods -n ecommerce-ai

# Should show:
# kafka-confluent-xxx         Running
# neo4j-release-0             Running
```

What this does: Confirms your database and messaging services are operational.

If services are not running:

```
# Restart Kafka
kubectl rollout restart deployment/kafka-confluent -n ecommerce-ai

# Restart Neo4j
kubectl rollout restart statefulset/neo4j-release -n ecommerce-ai

# Wait 2-3 minutes, then verify:
kubectl get pods -n ecommerce-ai
```

STEP 3: Network Connectivity Health Check 🌐

```
# Test Neo4j connectivity
nc -zv localhost 7687
# Expected: "Connection to localhost port 7687 [tcp] succeeded!"

# Test Kafka connectivity
nc -zv localhost 9092
# Expected: "Connection to localhost port 9092 [tcp] succeeded!"

# Test Neo4j Browser (optional)
curl -s http://localhost:7474 | grep -q "neo4j" && echo "Neo4j Browser OK" || ecl
```

What this does: Verifies network paths from your Mac to the containerized services.

If connectivity fails:

```
# Check service endpoints
kubectl get svc -n ecommerce-ai

# Look for LoadBalancer services with localhost:
# neo4j-lb-neo4j         LoadBalancer   localhost   7474:xxx/TCP,7687:xxx/TCP
# kafka-confluent-service LoadBalancer  localhost   9092:xxx/TCP
```

STEP 4: Database Validation 💾

```
# Navigate to project directory
cd /Users/timtadeo/Desktop/TelcoDemoDAIR

# Test Neo4j connection with credentials
python3 -c "
from neo4j import GraphDatabase
try:
    driver = GraphDatabase.driver('bolt://localhost:7687', auth=('neo4j', 'passw
    with driver.session() as session:
        result = session.run('MATCH (t:Tower) RETURN count(t) as towers')
        count = result.single()['towers']
        print(f'✅ Neo4j OK: {count} towers found')
    driver.close()
except Exception as e:
    print(f'❌ Neo4j Failed: {e}')
"
```

Expected Output: `✅ Neo4j OK: 100 towers found`

If database is empty or fails:

```
# Re-seed the database
python3 seed_neo4j.py

# Expected output:
# ✅ Constraints and indexes created successfully
# ✅ Seeded 4 carriers
# ✅ Seeded 100 towers
```

STEP 5: Test Individual Scripts 🧪

```
# Test tower data generator (30-second test)
timeout 30s python3 TelcoTowerDataV14.py

# Expected output:
# ✅ Loaded 100 towers from Neo4j
# 🚀 Starting Tower Data Generator...
# ✅ Tower status sent: TWR_0001 to tower-status

# Test CDR generator (30-second test)
```

```
timeout 30s python3 TelcoCDRDataKafkaV11.py

# Expected output:
# ✅ Loaded 100 towers and 4 carriers from Neo4j
# 🚀 Starting Telecom CDR Generator...
# ✅ CDR sent: CDR_123456 (telecom-cdr)
```

What this does: Validates that your Python scripts can connect to both Neo4j and Kafka independently.

If scripts fail:

- **Neo4j errors**: Check password in scripts matches `password123`
- **Kafka errors**: Verify Kafka service is running and accessible
- **Import errors**: Run `pip install faker confluent-kafka neo4j flask`

STEP 6: Start Flask Webhook Service 🌐

```
# Start the webhook listener
python3 run_demo.py

# Expected output:
# 🌐 Telecom Demo Webhook Listener Starting...
# 📍 Listening on port 5001
# * Running on http://127.0.0.1:5001
```

What this does: Creates the API endpoint that n8n will call to trigger your demo.

Keep this terminal open - you'll see real-time demo output here.

If port 5001 is in use:

```
# Find what's using the port
lsof -i :5001

# Kill the process or change port in run_demo.py
```

STEP 7: Test Local Webhook 🔧

Open a new terminal and test:

```
# Test health endpoint
curl -X GET http://localhost:5001/webhook/health

# Expected response:
# {"status": "healthy", "timestamp": "2025-01-20T..."}

# Test demo trigger (1-minute test)
curl -X POST http://localhost:5001/webhook/start-demo \
  -H "Content-Type: application/json" \
```

```
  -d '{"duration_minutes": 1}'

# Expected response:
# {"status": "success", "message": "Demo started successfully", ...}
```

What this does: Confirms your Flask service can receive requests and start the scripts.

Watch the Flask terminal - you should see both scripts start and generate data.

STEP 8: Expose to Internet 🌍

In another new terminal:

```
# Start ngrok tunnel
ngrok http 5001

# Expected output:
# Forwarding  https://xxxx-68-118-242-43.ngrok-free.app -> http://localhost:5001
```

Copy the HTTPS URL - you'll need this for n8n.

Test public access:

```
# Replace with your actual ngrok URL
curl -X GET https://your-ngrok-url.ngrok-free.app/webhook/health

# Expected: Same healthy response as local test
```

STEP 9: Configure n8n Workflow ⚡

In your n8n Cloud interface:

1. **Update HTTP Request URL** with your ngrok URL:

   ```
   https://your-ngrok-url.ngrok-free.app/webhook/start-demo
   ```

2. **Verify configuration**:
   - Method: `POST`
   - Headers: `Content-Type: application/json`
   - Body Parameters: `duration_minutes: 2`

3. **Save workflow**

Demo Execution & Monitoring 🎬

Execute the Demo

1. **Click "Execute workflow"** in n8n
2. **Monitor Flask terminal** for real-time output:

```
📞 Demo start request received - Duration: 2 minutes
[TelcoTowerDataV14.py] ✅ Loaded 100 towers from Neo4j
[TelcoCDRDataKafkaV11.py] ✅ Loaded 100 towers and 4 carriers from Neo4j
[TelcoTowerDataV14.py] ✅ Tower status sent: TWR_0008 to tower-status
[TelcoCDRDataKafkaV11.py] ✅ CDR sent: CDR_782139 (telecom-cdr)
[TelcoCDRDataKafkaV11.py] 💾 Stored interaction #1 in Neo4j (Status: DROPPED)
```

Success Indicators

- ✅ n8n shows green checkmark and success response
- ✅ Flask terminal shows both scripts running with data flow
- ✅ Scripts complete after specified duration
- ✅ No error messages in any terminal

Expected Data Volume (2-minute demo)

- **Tower Messages**: ~60 (1 every 2 seconds)
- **CDR Messages**: ~120 (1 every 1 second)
- **Neo4j Records**: ~12 (every 10th CDR stored)

Health Check Commands 🩺

Use these commands to diagnose issues during demo:

Quick Service Check

```
# All-in-one health check
echo "=== Docker ===" && docker ps --format "table {{.Names}}\t{{.Status}}" | hea
echo "=== Kubernetes ===" && kubectl get pods -n ecommerce-ai --no-headers | awk
echo "=== Connectivity ===" && nc -zv localhost 7687 && nc -zv localhost 9092
echo "=== Flask ===" && curl -s http://localhost:5001/webhook/health | grep -o ''
```

Neo4j Data Check

```
# Verify recent data
python3 -c "
from neo4j import GraphDatabase
driver = GraphDatabase.driver('bolt://localhost:7687', auth=('neo4j', 'password1
```

```
with driver.session() as session:
    towers = session.run('MATCH (t:Tower) RETURN count(t)').single()[0]
    recent = session.run('MATCH (i:Interaction) WHERE i.Timestamp >= datetime() -
    print(f'Towers: {towers}, Recent Interactions: {recent}')
driver.close()
"
```

Common Error Scenarios & Solutions 🔧

Error Symptom	Cause	Solution
`Cannot connect to Docker daemon`	Docker Desktop stopped	Start Docker Desktop application
`connection refused` to kubectl	Kubernetes disabled	Enable Kubernetes in Docker Desktop settings
`ServiceUnavailable: localhost:7687`	Neo4j service down	`kubectl rollout restart statefulset/neo4j-release -n ecommerce-ai`
`ApiVersionRequest failed`	Kafka service down	`kubectl rollout restart deployment/kafka-confluent -n ecommerce-ai`
`The endpoint xxx.ngrok-free.app is offline`	ngrok tunnel expired	Restart ngrok, update n8n URL
`port 5001 already in use`	Previous Flask instance running	`lsof -i :5001` then kill process
`No towers found`	Database not seeded	Run `python3 seed_neo4j.py`
`Import errors`	Missing Python packages	Run pip install command

Pre-Demo Final Checklist ✅

Complete this 2-minute checklist before EVERY demo execution:

```
# 1. Docker running?
docker ps > /dev/null && echo "✅ Docker OK" || echo "❌ Start Docker Desktop"

# 2. Services running?
```

```
kubectl get pods -n ecommerce-ai | grep -q "Running" && echo "✅ K8s OK" || echo

# 3. Connectivity working?
nc -zv localhost 7687 > /dev/null 2>&1 && nc -zv localhost 9092 > /dev/null 2>&1

# 4. Flask responsive?
curl -s http://localhost:5001/webhook/health | grep -q "healthy" && echo "✅ Flas

# 5. ngrok tunnel active?
curl -s https://your-ngrok-url.ngrok-free.app/webhook/health | grep -q "healthy"
```

Only proceed with n8n execution if ALL checks show ✅

Troubleshooting Quick Commands 🚨

If demo fails mid-execution:

```
# Stop all running processes
pkill -f "python3.*Telco"
pkill -f "run_demo.py"

# Restart everything fresh
kubectl rollout restart deployment/kafka-confluent -n ecommerce-ai
kubectl rollout restart statefulset/neo4j-release -n ecommerce-ai

# Wait 3 minutes, then restart from STEP 6
```

Emergency reset: If nothing works, restart Docker Desktop completely and begin from STEP 1.

Following this guide ensures your demo will work perfectly on the first try every time! 🎯