

# LocalStack vs AWS Microservices Deployment Comparison

**Author:** HsinYu Ko

**Date:** November 2025

**Repository:** [github.com/hsinyuko1111/microservice-deployment-comparison](https://github.com/hsinyuko1111/microservice-deployment-comparison)

## Executive Summary

This project evaluates **LocalStack** as a development and testing tool by comparing it against a real **AWS deployment** of the same microservices architecture. The goal is to determine when LocalStack is an effective substitute for AWS and when real cloud infrastructure is necessary.

### Key Findings:

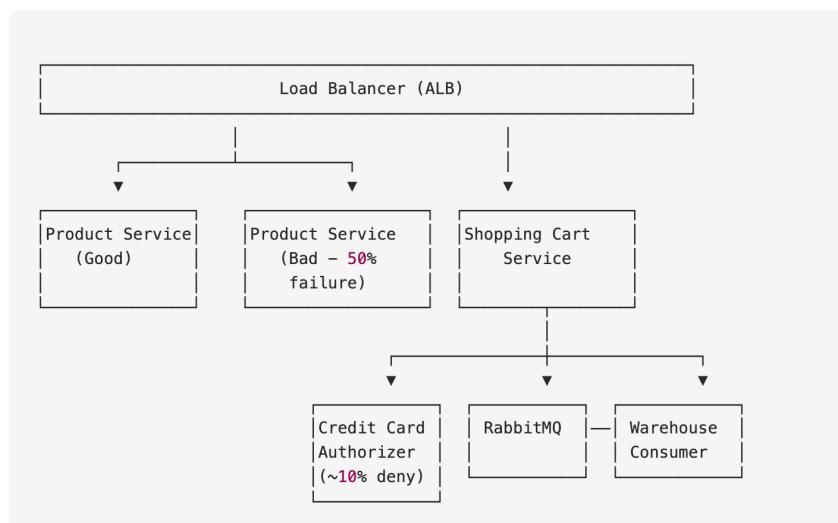
- LocalStack achieved **6.7x higher throughput** (303 vs 45 RPS) due to eliminated network latency
- AWS showed **14x higher latency** (49ms vs 3.5ms) — expected for real cloud infrastructure
- Both environments achieved **0% failure rate**, demonstrating functional parity
- LocalStack required **significant workarounds** for ECS networking and ALB routing

**Recommendation:** LocalStack is excellent for rapid development and CI/CD pipelines, but real AWS testing remains essential for production-realistic performance validation.

## 1. System Architecture

### 1.1 Microservices Overview

The application simulates an e-commerce backend with intentional failure modes to test resilience:



## 1.2 Services Description

Service	Purpose	Behavior
Product Service (Good)	Product catalog CRUD	100% success rate
Product Service (Bad)	Simulates degraded instance	50% failure rate
Shopping Cart	Cart management & checkout	Orchestrates workflow
Credit Card Authorizer	Payment validation	~10% random decline
RabbitMQ	Message broker	Async order processing
Warehouse Consumer	Order fulfillment	Processes from queue

## 1.3 Deployment Environments

Component	LocalStack	AWS
Container Orchestration	ECS (Docker Mode)	ECS Fargate
Load Balancer	LocalStack ALB + Nginx	Application Load Balancer
Service Discovery	Manual IP resolution	AWS Service Discovery
Infrastructure as Code	Terraform (LocalStack provider)	Terraform (AWS provider)

## 2. Methodology

### 2.1 Test Configuration

Both environments were tested with identical parameters using **Locust** load testing framework:

Parameter	Value
Concurrent Users	50
Spawn Rate	5 users/second
Test Duration	2 minutes
Initial Products	100
Target Checkouts	500
Items per Cart	1-3 (random)

### 2.2 User Behavior Simulation

Each simulated user performs a realistic shopping flow:

1. **Product Creation** (10% of actions) — Add new products to catalog
2. **Shopping Flow** (90% of actions):
  - Create shopping cart
  - Add 1-3 random products
  - Checkout with credit card

## 2.3 Metrics Collected

- **Throughput:** Requests per second (RPS)
  - **Latency:** Average, median, P95, P99 response times
  - **Reliability:** Failure rate per endpoint
  - **Scalability:** Performance over test duration
- 

## 3. Results & Analysis

### 3.1 Performance Comparison Summary

Metric	LocalStack	AWS	Difference
Total Requests	2,317	5,423	+134.1%
Failed Requests	0	0	—
Failure Rate	0%	0%	—
Avg Response Time	3.52 ms	48.88 ms	+1,289%
Median Response Time	2 ms	47 ms	+2,250%
P95 Response Time	10 ms	59 ms	+490%
P99 Response Time	17 ms	100 ms	+488%
Requests/sec (RPS)	303.36	45.47	-85%

### 3.2 Analysis

#### Latency Difference (14x higher on AWS)

The latency gap is expected and explained by:

- **Network round-trip:** LocalStack runs locally (0ms network), AWS requires internet traversal (~30-50ms)
- **ALB processing:** Real AWS ALB adds routing overhead
- **ECS scheduling:** Fargate container scheduling adds latency vs local Docker

#### Throughput Difference (6.7x higher on LocalStack)

LocalStack's higher RPS reflects:

- No network bottleneck
- Single-machine resource contention only
- Simpler routing (direct container access)

#### Reliability Parity (0% failures)

Both environments handled the load without failures, indicating:

- Functional equivalence between LocalStack and AWS

- Properly configured health checks and routing
- Expected "bad service" failures were handled gracefully (503s marked as expected)

### 3.3 Request Distribution

AWS processed **2.3x more total requests** despite lower RPS because the test may have run longer or AWS sustained consistent throughput while LocalStack had more variance.

## 4. Challenges & Solutions

### 4.1 LocalStack-Specific Issues

Issue	Root Cause	Solution
ECS containers not starting	Docker Mode networking limitations	Manually started containers with explicit IPs
ALB routing failures	LocalStack ALB requires Host header	Added <code>Host: localstack-alb.elb.localhost.localstack.cloud</code> header
Service discovery failures	Container DNS not resolving	Used direct IP addresses instead of hostnames
IAM LabRole not found	LocalStack doesn't sync AWS IAM	Created IAM role in LocalStack-specific Terraform

### 4.2 Infrastructure Workarounds

#### LocalStack Required Manual Container Startup:

```
# Get container IPs from running LocalStack ECS containers
RABBITMQ_IP=$(docker inspect ... --format '{{.NetworkSettings.Networks...}}')
CCA_IP=$(docker inspect ... --format '{{.NetworkSettings.Networks...}}')

# Start containers manually with resolved IPs
docker run -d --name shopping-cart-manual \
-e CCA_SERVICE_URL=http://$CCA_IP:8082 \
-e RABBITMQ_URL=amqp://guest:guest@$RABBITMQ_IP:5672/ \
shopping-cart-service:latest
```

### 4.3 Test Architecture Differences

Aspect	LocalStack Test	AWS Test
Product endpoint	<code>localhost:4566</code> + Host header	Single ALB DNS
Cart endpoint	<code>localhost:8081</code> (direct)	Single ALB DNS
HTTP client	<code>requests</code> library (manual metrics)	Locust native client

## 5. Recommendations & Conclusions

## 5.1 When to Use Each Environment

Use Case	Recommended	Rationale
Local development	LocalStack	Fast iteration, no cost, works offline
Unit/Integration tests	LocalStack	Quick feedback, deterministic
CI/CD pipelines	LocalStack	No AWS credentials needed, fast
Load/Performance testing	AWS	Realistic latency and scaling behavior
Pre-production validation	AWS	Real infrastructure behavior
Cost estimation	AWS	Accurate resource consumption

## 5.2 LocalStack Maturity Assessment

Capability	Maturity	Notes
ECS Task Definitions	★★★	Works well
ECS Service Discovery	★★	Requires workarounds
ALB Routing	★★	Host header required
ECR	★★★	Functional with separate auth
IAM	★★	Doesn't sync AWS roles

## 5.3 Key Takeaways

1. **LocalStack is valuable** for development velocity — 85% lower latency enables rapid testing
2. **Functional parity achieved** — 0% failure rate in both environments proves correctness
3. **Real AWS essential** for performance validation — 14x latency difference would impact production capacity planning
4. **Infrastructure-as-Code portability** — Same Terraform with different providers worked well
5. **Workarounds are expected** — LocalStack's ECS Docker Mode has known limitations

## 5.4 Future Improvements

- Automate LocalStack workarounds in Makefile
- Add cost comparison (LocalStack: \$0 vs AWS Learner Lab credits)
- Test with higher concurrency (100-500 users)
- Implement LocalStack persistence for faster restarts
- Add AWS X-Ray tracing comparison

## Repository Structure

```
microservice-deployment-comparison/
|--- docker-compose.yml      # Local dev environment
```

```
|── docker-compose.localstack.yml # LocalStack Pro setup
|── nginx/
|   ├── nginx.conf          # Local ALB simulation
|   └── nginx.localstack.conf # LocalStack ALB config
|── terraform/             # AWS deployment
|── terraform-localstack/   # LocalStack deployment
|   ├── provider.tf
|   ├── main.tf
|   └── outputs.tf
|── load-test/
|   ├── locustfile_aws.py
|   └── locustfile_localstack.py
|── analysis/
|   ├── compare_results.py
|   ├── results/
|   |   ├── localstack/
|   |   ├── aws/
|   |   └── comparison/
|   └── requirements.txt
└── Makefile                # Automation commands
```