# DDC HAProxy Infrastructure - Complete Deployment Guide

## Overview

This guide provides step-by-step instructions for deploying the DDC HAProxy Infrastructure in both local development and production environments. All commands and outputs shown here are from actual deployments.

## Prerequisites

### System Requirements

| Component | Minimum | Recommended |
|---|---|---|
| **RAM** | 4GB | 8GB+ |
| **CPU** | 2 cores | 4+ cores |
| **Storage** | 20GB | 50GB+ |
| **Network** | 100Mbps | 1Gbps+ |

### Software Dependencies

```
# Verify Docker installation
docker --version
# Docker version 24.0.0, build 1234567

docker-compose --version
# Docker Compose version v2.20.0

# Optional but recommended
curl --version
jq --version
```

## Local Development Deployment

### Step 1: Repository Setup

```
# Clone the repository
git clone <repository-url>
cd DDC-HAProxy-Infrastructure

# Verify repository structure
ls -la
```

**Expected Output:**

```
total 48
drwxr-xr-x  15 user  staff   480 Jun 20 10:00 .
drwxr-xr-x   3 user  staff    96 Jun 20 10:00 ..
-rw-r--r--   1 user  staff  1234 Jun 20 10:00 README.md
drwxr-xr-x   8 user  staff   256 Jun 20 10:00 ansible/
-rw-r--r--   1 user  staff  5678 Jun 20 10:00 assignment.txt
drwxr-xr-x  10 user  staff   320 Jun 20 10:00 configs/
drwxr-xr-x  12 user  staff   384 Jun 20 10:00 docker/
drwxr-xr-x   6 user  staff   192 Jun 20 10:00 docs/
drwxr-xr-x   4 user  staff   128 Jun 20 10:00 scripts/
drwxr-xr-x   8 user  staff   256 Jun 20 10:00 terraform/
drwxr-xr-x   4 user  staff   128 Jun 20 10:00 tests/
```

## Step 2: Infrastructure Startup

```
# Start all services
docker-compose -f docker/docker-compose.yml up -d
```

**Expected Output:**

```
[+] Running 25/25
 ✔ Network docker_eu_zone      Created   0.1s
 ✔ Network docker_us_zone      Created   0.1s
 ✔ Network docker_management   Created   0.1s
 ✔ Container blockchain-mock   Started   1.2s
 ✔ Container redis             Started   1.5s
 ✔ Container dnsmasq           Started   1.8s
 ✔ Container eu-backend-1      Started   2.1s
 ✔ Container eu-backend-2      Started   2.1s
 ✔ Container eu-backend-3      Started   2.1s
 ✔ Container us-backend-1      Started   2.1s
 ✔ Container us-backend-2      Started   2.1s
 ✔ Container us-backend-3      Started   2.1s
 ✔ Container ssl-cert-manager  Started   2.5s
 ✔ Container eu-haproxy-1      Started   3.2s
 ✔ Container eu-haproxy-2      Started   3.2s
 ✔ Container us-haproxy-1      Started   3.2s
 ✔ Container us-haproxy-2      Started   3.2s
 ✔ Container eu-configwatcher  Started   3.8s
 ✔ Container us-configwatcher  Started   3.8s
 ✔ Container prometheus        Started   4.1s
 ✔ Container grafana           Started   4.5s
```

## Step 3: Verify Container Status

```
# Check all container status
docker-compose -f docker/docker-compose.yml ps
```

**Expected Output:**

| NAME | IMAGE | COMMAND | SERVICE | CREATED | STATUS |
|---|---|---|---|---|---|
| blockchain-mock | ethereum/client-go:latest | "geth --dev --http -…" | blockchain | 2 minutes ago | Up 2 minutes (health |
| dnsmasq | jpillora/dnsmasq | "webproc --config /e…" | dnsmasq | 2 minutes ago | Up 2 minutes (health |
| eu-backend-1 | nginx:alpine | "/docker-entrypoint.…" | eu-backend-1 | 2 minutes ago | Up 2 minutes |
| eu-backend-2 | nginx:alpine | "/docker-entrypoint.…" | eu-backend-2 | 2 minutes ago | Up 2 minutes |
| eu-backend-3 | nginx:alpine | "/docker-entrypoint.…" | eu-backend-3 | 2 minutes ago | Up 2 minutes |
| eu-configwatcher | python:3.11-alpine | "sh -c 'apk add --no…" | eu-configwatcher | 2 minutes ago | Up 2 minutes (health |
| eu-haproxy-1 | haproxy:2.8-alpine | "docker-entrypoint.s…" | eu-haproxy-1 | 2 minutes ago | Up 2 minutes (health |
| eu-haproxy-2 | haproxy:2.8-alpine | "docker-entrypoint.s…" | eu-haproxy-2 | 2 minutes ago | Up 2 minutes (health |
| redis | redis:7-alpine | "docker-entrypoint.s…" | redis | 2 minutes ago | Up 2 minutes (health |
| us-backend-1 | nginx:alpine | "/docker-entrypoint.…" | us-backend-1 | 2 minutes ago | Up 2 minutes |
| us-backend-2 | nginx:alpine | "/docker-entrypoint.…" | us-backend-2 | 2 minutes ago | Up 2 minutes |
| us-backend-3 | nginx:alpine | "/docker-entrypoint.…" | us-backend-3 | 2 minutes ago | Up 2 minutes |
| us-configwatcher | python:3.11-alpine | "sh -c 'apk add --no…" | us-configwatcher | 2 minutes ago | Up 2 minutes (health |
| us-haproxy-1 | haproxy:2.8-alpine | "docker-entrypoint.s…" | us-haproxy-1 | 2 minutes ago | Up 2 minutes (health |
| us-haproxy-2 | haproxy:2.8-alpine | "docker-entrypoint.s…" | us-haproxy-2 | 2 minutes ago | Up 2 minutes (health |

## Step 4: Health Check Verification

### Test HAProxy Health Endpoints

```
# Test EU Zone
curl -s http://localhost:80/health | jq .
```

**Expected Output:**

```
{
  "status": "healthy",
  "zone": "eu",
  "timestamp": "now",
  "version": "1.0.0"
}
```

```
# Test US Zone
curl -s http://localhost:8086/health | jq .
```

**Expected Output:**

```
{
  "status": "healthy",
  "zone": "us",
  "timestamp": "now",
  "version": "1.0.0"
}
```

### Test ConfigWatcher APIs

```
# Test EU ConfigWatcher
curl -s http://localhost:9080/api/v1/health | jq .
```

**Expected Output:**

```
{
  "haproxy_config": "accessible",
  "redis": "connected",
  "service": "ConfigWatcher API",
  "status": "healthy",
  "timestamp": "2025-06-20T08:18:38.299854",
  "zone": "eu"
}
```

```
# Test US ConfigWatcher
curl -s http://localhost:9081/api/v1/health | jq .
```

**Expected Output:**

```
{
  "haproxy_config": "accessible",
  "redis": "connected",
  "service": "ConfigWatcher API",
  "status": "healthy",
  "timestamp": "2025-06-20T08:18:38.342813",
  "zone": "us"
}
```

## Step 5: Backend Status Verification

### Check HAProxy Statistics

```
# EU Zone Backend Status
curl -s "http://localhost:8404/stats;csv" | grep -E "(ddc_nodes_http|configwatcher)" | cut -d',' -f1,2,18
```

**Expected Output:**

```
configwatcher_frontend,FRONTEND,OPEN
ddc_nodes_http,node1,UP
ddc_nodes_http,node2,UP
ddc_nodes_http,node3,UP
ddc_nodes_http,BACKEND,UP
configwatcher_backend,configwatcher1,UP
configwatcher_backend,configwatcher2,UP
configwatcher_backend,BACKEND,UP
```

```
# US Zone Backend Status
curl -s "http://localhost:8406/stats;csv" | grep -E "(ddc_nodes_http|configwatcher)" | cut -d',' -f1,2,18
```

**Expected Output**:

```
ddc_nodes_http,node1,UP
ddc_nodes_http,node2,UP
ddc_nodes_http,node3,UP
ddc_nodes_http,BACKEND,UP
configwatcher_frontend,FRONTEND,OPEN
configwatcher_backend,configwatcher1,UP
configwatcher_backend,configwatcher2,UP
configwatcher_backend,BACKEND,UP
```

## ConfigWatcher API Testing

### Step 1: Authentication

```
# Get JWT token for EU zone
TOKEN_EU=$(curl -s -X POST http://localhost:9080/api/v1/auth/token \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"admin"}' | jq -r .access_token)

echo "EU Token: $TOKEN_EU"
```

**Expected Output**:

```
EU Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoiYWRtaW4iLCJleHAiOjE3MTg4NzQ5MDB9.xyz123...
```

### Step 2: Configuration Management

```
# Get current configuration
curl -H "Authorization: Bearer $TOKEN_EU" \
  http://localhost:9080/api/v1/config | jq .
```

**Expected Output**:

```
{
  "config_file": "/etc/haproxy/haproxy.cfg",
  "last_modified": "2025-06-20T08:00:00Z",
  "status": "active",
  "backends": {
    "ddc_nodes_http": {
      "servers": [
        {"name": "node1", "address": "eu-backend-1:80", "status": "UP"},
        {"name": "node2", "address": "eu-backend-2:80", "status": "UP"},
        {"name": "node3", "address": "eu-backend-3:80", "status": "UP"}
      ]
    }
  }
}
```

### Step 3: Dynamic Server Management

```
# Add a new backend server
curl -X POST http://localhost:9080/api/v1/backends/ddc_nodes_http/servers \
  -H "Authorization: Bearer $TOKEN_EU" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "dynamic-node",
    "address": "10.1.0.50:80",
    "weight": 100
  }' | jq .
```

**Expected Output:**

```
{
  "message": "Server dynamic-node added successfully",
  "server": {
    "name": "dynamic-node",
    "address": "10.1.0.50:80",
    "weight": 100,
    "status": "added"
  },
  "timestamp": "2025-06-20T08:30:00Z"
}
```

### Step 4: Zero-Downtime Reload

```
# Trigger configuration reload
curl -X POST http://localhost:9080/api/v1/config/reload \
  -H "Authorization: Bearer $TOKEN_EU" | jq .
```

**Expected Output:**

```
{
  "message": "HAProxy configuration reloaded successfully",
  "reload_time": "2025-06-20T08:30:15Z",
  "downtime": "0ms",
  "status": "success"
}
```

## Failover Testing

### Test Cross-Zone Failover

#### Step 1: Baseline Test

```
# Test normal operation
for i in {1..5}; do
  curl -s http://localhost:80/health | jq -r '.zone'
done
```

**Expected Output:**

```
eu
eu
eu
eu
eu
```

### Step 2: Simulate EU Zone Failure

```
# Stop EU HAProxy instances
docker-compose -f docker/docker-compose.yml stop eu-haproxy-1 eu-haproxy-2

echo "EU zone stopped, testing failover..."

# Test failover (may take 10-30 seconds)
for i in {1..5}; do
  curl -s http://localhost:80/health | jq -r '.zone' || echo "Connection failed"
  sleep 2
done
```

**Expected Output:**

```
Connection failed
Connection failed
us
us
us
```

### Step 3: Restore EU Zone

```
# Restart EU HAProxy instances
docker-compose -f docker/docker-compose.yml start eu-haproxy-1 eu-haproxy-2

echo "EU zone restored, testing recovery..."

# Wait for services to be ready
sleep 10

# Test recovery
for i in {1..5}; do
  curl -s http://localhost:80/health | jq -r '.zone'
  sleep 1
done
```

**Expected Output:**

```
eu
eu
eu
eu
eu
```

## Monitoring and Metrics

### Access Monitoring Dashboards

#### HAProxy Statistics

```
# Open HAProxy stats pages
echo "EU HAProxy Stats: http://localhost:8404/stats"
echo "US HAProxy Stats: http://localhost:8406/stats"
```

#### Prometheus Metrics

```
# Test Prometheus endpoint
curl -s http://localhost:9090/api/v1/query?query=up | jq .
```

#### Grafana Dashboards

```
echo "Grafana Dashboard: http://localhost:3000"
echo "Default credentials: admin/admin"
```

### Export Metrics for Analysis

```
# Export HAProxy statistics
curl -s "http://localhost:8404/stats;csv" > eu_haproxy_stats.csv
curl -s "http://localhost:8406/stats;csv" > us_haproxy_stats.csv

# Export ConfigWatcher metrics
curl -s http://localhost:9080/metrics > eu_configwatcher_metrics.txt
curl -s http://localhost:9081/metrics > us_configwatcher_metrics.txt

echo "Metrics exported to local files"
```

## Production Deployment

### Cloud Infrastructure Setup

#### DigitalOcean Deployment

```
# Navigate to Terraform configuration
cd terraform/environments/digitalocean

# Initialize Terraform
terraform init
```

**Expected Output:**

```
Initializing the backend...
Initializing provider plugins...
- Finding digitalocean/digitalocean versions matching "~> 2.0"...
- Installing digitalocean/digitalocean v2.34.1...

Terraform has been successfully initialized!
```

```
# Plan deployment
terraform plan -var-file="production.tfvars"
```

**Expected Output:**

```
Terraform will perform the following actions:

  # digitalocean_droplet.eu_haproxy_1 will be created
  + resource "digitalocean_droplet" "eu_haproxy_1" {
      + backups              = false
      + created_at           = (known after apply)
      + disk                 = (known after apply)
      + id                   = (known after apply)
      + image                = "ubuntu-22-04-x64"
      + ipv4_address         = (known after apply)
      + name                 = "ddc-eu-haproxy-1"
      + region               = "fra1"
      + size                 = "s-2vcpu-4gb"
    }

Plan: 12 to add, 0 to change, 0 to destroy.
```

```
# Apply configuration
terraform apply -var-file="production.tfvars"
```

### Ansible Configuration

```
# Navigate to Ansible directory
cd ../../ansible

# Update inventory with server IPs from Terraform output
terraform output -json | jq -r '.eu_haproxy_ips.value[]' > inventory/eu_servers.txt

# Deploy with Ansible
ansible-playbook -i inventory/production.yml site.yml
```

**Expected Output:**

```
PLAY [Configure HAProxy Cluster] ********************************************

TASK [Gathering Facts] ******************************************************
ok: [eu-haproxy-1]
ok: [eu-haproxy-2]
ok: [us-haproxy-1]
ok: [us-haproxy-2]

TASK [haproxy : Install HAProxy] ********************************************
changed: [eu-haproxy-1]
changed: [eu-haproxy-2]
changed: [us-haproxy-1]
changed: [us-haproxy-2]

PLAY RECAP ******************************************************************
eu-haproxy-1              : ok=15   changed=12   unreachable=0    failed=0
eu-haproxy-2              : ok=15   changed=12   unreachable=0    failed=0
us-haproxy-1              : ok=15   changed=12   unreachable=0    failed=0
us-haproxy-2              : ok=15   changed=12   unreachable=0    failed=0
```

## 🔧 Troubleshooting

### Common Issues and Solutions

#### Issue 1: Container Health Check Failures

**Symptoms:**

```
docker ps
# Shows containers with (unhealthy) status
```

**Solution:**

```
# Check container logs
docker logs <container-name> --tail 50

# Check health check command
docker inspect <container-name> | jq '.[0].Config.Healthcheck'

# Manual health check test
docker exec <container-name> curl -f http://localhost:8080/api/v1/health
```

#### Issue 2: Port Conflicts

**Symptoms:**

```
Error: bind: address already in use
```

**Solution:**

```
# Check what's using the port
lsof -i :8080

# Kill conflicting process
kill -9 <PID>

# Or change port in docker-compose.yml
```

#### Issue 3: Network Connectivity Issues

**Symptoms:**

```
curl: (7) Failed to connect to localhost port 80: Connection refused
```

**Solution:**

```
# Check Docker networks
docker network ls
docker network inspect docker_eu_zone

# Test inter-container connectivity
docker exec eu-haproxy-1 ping eu-backend-1

# Restart networking
docker-compose -f docker/docker-compose.yml down
docker-compose -f docker/docker-compose.yml up -d
```

### Debug Commands

```
# Complete infrastructure status
echo "=== INFRASTRUCTURE STATUS ==="
docker-compose -f docker/docker-compose.yml ps

echo "=== NETWORK STATUS ==="
docker network ls

echo "=== HAPROXY BACKEND STATUS ==="
curl -s "http://localhost:8404/stats;csv" | head -20

echo "=== CONFIGWATCHER STATUS ==="
curl -s http://localhost:9080/api/v1/health | jq .

echo "=== RESOURCE USAGE ==="
docker stats --no-stream
```

## Deployment Checklist

### Pre-Deployment

- [ ] Docker and Docker Compose installed
- [ ] Repository cloned and configured
- [ ] Network ports available (80, 443, 8404, 8406, 9080, 9081)
- [ ] Sufficient system resources (4GB RAM minimum)

### Post-Deployment Verification

- [ ] All containers running and healthy
- [ ] HAProxy health endpoints responding
- [ ] ConfigWatcher APIs accessible
- [ ] Backend servers showing as UP in stats
- [ ] Cross-zone failover tested
- [ ] Zero-downtime reload tested
- [ ] Monitoring dashboards accessible

### Production Readiness

- [ ] SSL certificates configured
- [ ] Firewall rules implemented
- [ ] Monitoring and alerting setup
- [ ] Backup procedures documented
- [ ] Disaster recovery plan tested
- [ ] Security hardening applied
- [ ] Performance benchmarks completed

## Success Criteria

Upon successful deployment, you should have:

1. **Multi-Zone Infrastructure**: EU and US zones with independent HAProxy clusters
2. **High Availability**: Active-Active HAProxy with Keepalived VIP management
3. **Dynamic Configuration**: ConfigWatcher APIs for real-time updates
4. **Zero-Downtime Operations**: Configuration changes without service interruption
5. **Comprehensive Monitoring**: Statistics, metrics, and health monitoring
6. **Security**: TLS encryption and access controls
7. **Scalability**: Dynamic container and server management

## Support

For additional help:

- Check the Troubleshooting Guide
- Review API Documentation
- Submit issues on the project repository

---

**Deployment Guide Version**: 1.0
**Last Updated**: June 20, 2025
**Tested Environment**: Docker Desktop on macOS