ADMINISTRATOR TRAINING

Anti-Call Masking Platform

Duration: ~30 minutes
Audience: System Administrators, DevOps
Version: 2.0 | January 2026

PREREQUISITES

Before this training, ensure you have:

- Completed the System Overview training
- Admin credentials for the platform
- Access to the infrastructure (SSH/VPN)
- Familiarity with Docker and Linux

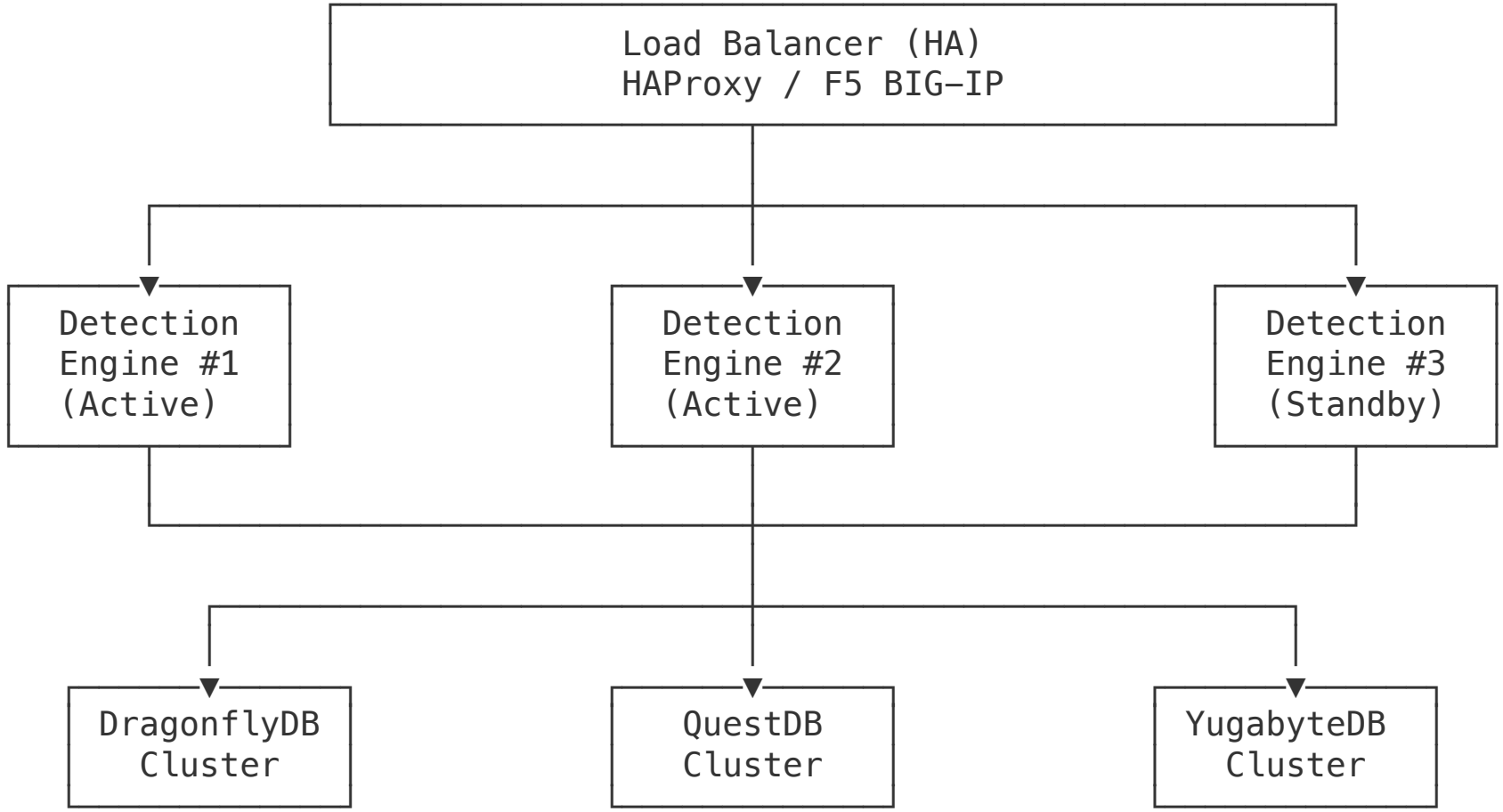
AGENDA

1. Infrastructure Overview
2. Deployment & Configuration
3. Monitoring & Health Checks
4. Database Administration
5. Security Management
6. Maintenance Procedures
7. Troubleshooting Guide

INFRASTRUCTURE OVERVIEW

Understanding the deployment architecture

PRODUCTION ARCHITECTURE



SERVICE INVENTORY

Service	Port	Instances	Memory
detection-engine	8080	3-6	64GB each
management-api	8081	2-3	32GB each
dragonfly	6379	3	128GB each
questdb	9000/9009	2-3	64GB each
yugabytedb	5433	3-5	64GB each
clickhouse	8123/9000	2-6	128GB each

NETWORK REQUIREMENTS

INTERNAL NETWORK:

- 25-100 Gbps between nodes
- <1ms latency requirement
- Dedicated VLAN

EXTERNAL ACCESS:

- SIP Switch: TCP 8080
- Dashboard: TCP 443
- NCC SFTP: TCP 22 outbound

DEPLOYMENT & CONFIGURATION

Managing platform deployments

DOCKER COMPOSE DEPLOYMENT

```
# Navigate to deployment directory
cd /opt/acm

# Pull latest images
docker compose pull

# Start all services
docker compose up -d

# Verify all services are running
docker compose ps

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

CONFIGURATION FILES

File	Purpose
`docker-compose.yml`	Service definitions
`config/detection.toml`	Detection engine config
`config/prometheus.yml`	Metrics scraping
`.env`	Environment variables
`config/ncc/`	NCC integration settings

KEY CONFIGURATION PARAMETERS

```
# config/detection.toml

[detection]
# Number of distinct A-numbers to trigger alert
threshold = 5

# Time window in seconds
window_seconds = 5

# Enable automatic call disconnection
auto_disconnect = true

# Cooldown between alerts for same B-number
cooldown_seconds = 60

[performance]
```

ENVIRONMENT VARIABLES

```
# .env file

# Database connections
DRAGONFLY_URL=redis://dragonfly:6379
QUESTDB_URL=http://questdb:9000
YUGABYTE_URL=postgres://yugabyte:5433/acm

# NCC Integration
NCC_API_URL=https://api.ncc.gov.ng
NCC_SFTP_HOST=sftp.ncc.gov.ng
NCC_OPERATOR_ID=OPERATOR001

# Security
API_KEY_SALT=your-secure-salt
JWT_SECRET=your-jwt-secret
```

Security: Never commit .env files to version control!

ROLLING UPDATES

```
# Update detection engine with zero downtime

# 1. Pull new image
docker compose pull detection-engine

# 2. Scale up (add new instance)
docker compose up -d --scale detection-engine=4

# 3. Wait for health check
sleep 30

# 4. Remove old containers
docker compose up -d --scale detection-engine=3

# 5. Verify
docker compose ps
```

MONITORING & HEALTH CHECKS

Keeping the system healthy

GRAFANA DASHBOARDS

> Access Grafana at: `http://grafana:3000`

KEY DASHBOARDS:

- **ACM Overview** - System-wide metrics
- **Detection Engine** - CPS, latency, alerts
- **Database Health** - All database metrics
- **NCC Compliance** - Report status

CRITICAL METRICS TO MONITOR

Metric	Warning	Critical
Detection Latency P99	> 0.8ms	> 1ms
CPU Usage	> 70%	> 85%
Memory Usage	> 80%	> 90%
DragonflyDB Memory	> 75%	> 90%
Disk Usage	> 70%	> 85%
Error Rate	> 0.1%	> 1%

HEALTH CHECK ENDPOINTS

```
# Detection Engine health
curl http://detection-engine:8080/health
# Response: {"status":"healthy","version":"2.0.0"}

# Management API health
curl http://management-api:8081/health
# Response: {"status":"healthy","db":"connected"}

# Full system check
curl http://management-api:8081/health/full
# Response: {"detection":"healthy","cache":"healthy",
#            "timeseries":"healthy","sql":"healthy"}
```

PROMETHEUS ALERTS

```
# alertmanager rules

groups:
  - name: acm-critical
    rules:
      - alert: HighDetectionLatency
        expr: acm_detection_latency_p99 > 0.001
        for: 2m
        labels:
          severity: critical
        annotations:
          summary: "Detection latency exceeds 1ms"

      - alert: ServiceDown
        expr: up{job="detection-engine"} == 0
        for: 30s
```

LOG MANAGEMENT

```
# View detection engine logs
docker compose logs -f detection-engine | grep ERROR

# View last 100 lines
docker compose logs --tail=100 detection-engine

# Export logs for analysis
docker compose logs detection-engine > /tmp/logs.txt

# Using Loki queries (Grafana)
{job="detection-engine"} |= "ERROR"
{job="detection-engine"} | json | level="error"
```

DATABASE ADMINISTRATION

Managing the data layer

DRAGONFLYDB MANAGEMENT

```
# Connect to DragonflyDB
docker compose exec dragonfly redis-cli

# Check memory usage
INFO memory

# Check connected clients
CLIENT LIST

# View sliding window keys
KEYS "window:*" | head -10

# Monitor real-time commands
MONITOR
```

> DragonflyDB is in-memory only. Data loss on restart is expected and handled by design.

QUESTDB MANAGEMENT

```
-- Access via HTTP console: http://questdb:9000

-- Check table sizes
SELECT table_name,
       sum(numRows) as rows,
       sum(diskSize) / 1024 / 1024 as size_mb
FROM tables()
GROUP BY table_name;

-- Check partition info
SELECT * FROM table_partitions('call_events');

-- Drop old partitions (retention)
ALTER TABLE call_events
DROP PARTITION WHERE timestamp < dateadd('d', -90, now());
```

YUGABYTEDB MANAGEMENT

```
# Connect to YugabyteDB
docker compose exec yugabytedb ysqlsh -h localhost

# Check cluster health
SELECT * FROM yb_servers();

# Check replication status
SELECT * FROM yb_table_replication_info();

# Vacuum tables
VACUUM ANALYZE alerts;
VACUUM ANALYZE whitelist;
```

CLICKHOUSE MANAGEMENT

```
-- Connect: clickhouse-client

-- Check table sizes
SELECT table,
       formatReadableSize(sum(bytes)) as size,
       sum(rows) as rows
FROM system.parts
WHERE active
GROUP BY table;

-- Check merges in progress
SELECT * FROM system.merges;

-- Optimize table (force merge)
OPTIMIZE TABLE analytics_events FINAL;
```

BACKUP PROCEDURES

```
# YugabyteDB backup
docker compose exec yugabytedb ysql_dump -h localhost acm > backup.sql

# ClickHouse backup
docker compose exec clickhouse \
  clickhouse-client --query="BACKUP TABLE analytics_events TO Disk('backups',

# QuestDB backup (copy data directory)
docker compose stop questdb
cp -r ./data/questdb ./backups/questdb-$(date +%Y%m%d)
docker compose start questdb
```

SECURITY MANAGEMENT

Protecting the platform

API KEY MANAGEMENT

```
# Generate new API key
curl -X POST http://management-api:8081/admin/api-keys \
  -H "Authorization: Bearer $ADMIN_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "opensips-integration",
    "scopes": ["events:write", "alerts:read"],
    "expires_in_days": 365
  }'

# Revoke API key
curl -X DELETE http://management-api:8081/admin/api-keys/KEY_ID \
  -H "Authorization: Bearer $ADMIN_TOKEN"

# List all API keys
curl http://management-api:8081/admin/api-keys \
```

USER MANAGEMENT

```
# Create user
curl -X POST http://management-api:8081/admin/users \
  -H "Authorization: Bearer $ADMIN_TOKEN" \
  -d '{
    "username": "analyst1",
    "email": "analyst1@company.com",
    "role": "analyst",
    "mfa_required": true
  }'

# Reset password
curl -X POST http://management-api:8081/admin/users/USER_ID/reset-password \
  -H "Authorization: Bearer $ADMIN_TOKEN"

# Disable user
curl -X PATCH http://management-api:8081/admin/users/USER_ID \
```

TLS CERTIFICATE MANAGEMENT

```
# Check certificate expiration
openssl x509 -in /etc/ssl/acm/server.crt -noout -enddate

# Generate new certificate (Let's Encrypt)
certbot certonly --standalone -d acm.yourcompany.com

# Renew certificates
certbot renew

# Verify certificate chain
openssl verify -CAfile ca.crt server.crt
```

> Set calendar reminders to renew certificates 30 days befor
> expiration!

AUDIT LOG REVIEW

```
-- YugabyteDB audit queries

-- Recent admin actions
SELECT timestamp, user_id, action, details
FROM audit_log
WHERE action LIKE 'admin.%'
ORDER BY timestamp DESC
LIMIT 100;

-- Failed login attempts
SELECT timestamp, user_id, ip_address
FROM audit_log
WHERE action = 'auth.login_failed'
  AND timestamp > now() - interval '24 hours'
ORDER BY timestamp DESC;
```

MAINTENANCE PROCEDURES

Keeping the system running smoothly

DAILY CHECKS

- **08:00** - Verify NCC report upload success
- **09:00** - Review overnight alerts
- **12:00** - Check system metrics
- **17:00** - Review daily statistics

WEEKLY MAINTENANCE

- Review and rotate logs
- Check disk space trends
- Verify backup integrity
- Review security alerts
- Update documentation if needed
- Test DR failover (monthly)

DATA RETENTION

Data Type	Retention	Action
Call Events	90 days	Auto-purge
Alerts	365 days	Archive then purge
Audit Logs	5 years	Archive to cold storage
Analytics	365 days	Aggregate then purge
Metrics	30 days	Auto-purge

SCALING PROCEDURES

```
# Scale up detection engines
docker compose up -d --scale detection-engine=5

# Verify load balancer configuration
curl http://load-balancer/stats

# Scale down (during low traffic)
docker compose up -d --scale detection-engine=3

# Add new database node (YugabyteDB)
docker compose exec yugabytedb yb-admin \
  add_node new-node.internal:7100
```

TROUBLESHOOTING GUIDE

Common issues and solutions

HIGH LATENCY

> **Symptom:** Detection latency > 1ms

DIAGNOSTIC STEPS:

1. Check DragonflyDB memory: `redis-cli INFO memo`
2. Check CPU usage: `docker stats`
3. Check network latency: `ping -c 10 dragonfly`
4. Review recent config changes

SOLUTIONS:

SERVICE NOT STARTING

```
# Check service logs
docker compose logs detection-engine | tail -50

# Check for port conflicts
netstat -tlnp | grep 8080

# Check resource limits
docker inspect detection-engine | grep -A 10 Resources

# Restart with verbose logging
LOG_LEVEL=debug docker compose up detection-engine
```

DATABASE CONNECTION ISSUES

```
# Test DragonflyDB connection
docker compose exec detection-engine \
  redis-cli -h dragonfly PING

# Test QuestDB connection
curl http://questdb:9000/exec?query=SELECT%201

# Test YugabyteDB connection
docker compose exec management-api \
  psql -h yugabytedb -p 5433 -U yugabyte -c "SELECT 1"

# Check DNS resolution
docker compose exec detection-engine \
  nslookup dragonfly
```

NCC REPORT FAILURES

> NCC reports must be uploaded by 06:00 WAT daily

```
# Check report generation
docker compose logs ncc-reporter | grep -i error

# Verify SFTP connectivity
sftp -i /etc/ncc/key ncc@sftp.ncc.gov.ng

# Manual report upload
docker compose exec ncc-reporter \
  ./upload-report.sh /reports/daily-$(date +%Y%m%d).csv

# Verify credentials
cat /etc/ncc/credentials.json | jq .
```

EMERGENCY PROCEDURES

Complete System Failure:

1. Notify SOC team immediately
2. Enable bypass mode on SIP switch (if available)
3. Check infrastructure (power, network, storage)
4. Restart services: `docker compose restart`
5. If persistent, initiate DR failover
6. Document incident for post-mortem

ESCALATION MATRIX

Severity	Response Time	Escalate To
Critical	Immediate	On-call + Manager
High	15 minutes	On-call Engineer
Medium	1 hour	Platform Team
Low	Next business day	Support Queue

SUMMARY

- **Infrastructure:** Multi-node, highly available deployment
- **Deployment:** Docker Compose with rolling updates
- **Monitoring:** Grafana, Prometheus, Loki stack
- **Databases:** DragonflyDB, QuestDB, YugabyteDB, ClickHouse
- **Security:** API keys, TLS, audit logging
- **Maintenance:** Daily checks, weekly tasks, retention

RESOURCES

DOCUMENTATION:

- `docs/manuals/ADMIN_MANUAL.md`
- `docs/runbook.md`
- `docs/technical/SYSTEM_REQUIREMENTS.md`

SUPP

- Sl
- Er
  pl
- O

QUESTIONS?

Administrator Training Complete

Next: Practice in staging environment
Schedule: Hands-on lab session