DRAGONFLY_72HR_EXECUTION_CHECKLIST.md

DRAGONFLY MIGRATION & DEPLOYMENT CHECKLIST

Jan 16-18, 2026 (72-Hour Execution Plan)

Project: Ecosystem Platform Redis → Dragonfly Migration + Phase 1 Production Go-Live
Timeline: Friday Jan 16 → Sunday Jan 18
Status: 🔴 NOT STARTED (ready to go)

JAN 16: DRAGONFLY MIGRATION DAY (Today)

9:00am - 12:00pm: PLANNING & PREPARATION

DevOps Lead - Pre-Flight Checklist:

- Verify Dragonfly Docker image available (`docker pull dragonflydb/dragonfly`)
- Cache VM has 8GB RAM available (check with `free -h`)
- Persistent volume mounted at `/mnt/volumes/dragonfly` (check with `df -h`)
- Network connectivity to cache VM confirmed
- SSH keys configured for all 15 service VMs
- Spaceship CLI installed & configured (`spaceship --version`)
- Docker daemon running on all VMs (`docker ps`)
- PostgreSQL connection verified (`psql -h postgres.internal -U postgres -d ecosystem`)

QA Lead - Test Plan Preparation:

- Stagingiest environment matches production exactly
- Test data loaded into staging PostgreSQL
- Load testing tool installed (`locust` or `jmeter`)
- Monitoring dashboards prepared (Prometheus + Grafana)
- Alert rules configured (latency > 500ms, error rate > 1%)
- Slack channels created:
  - #ecosystem-dragonfly-migration (live chat)
  - #production-incidents (escalation)
  - #deployment-status (updates)

Infrastructure Lead - Risk Mitigation:

- Redis rollback plan documented
- Redis container ready to deploy (standby image)
- Data backup verified (PostgreSQL snapshot created)
- Rollback procedures tested with non-prod environment
- 24-hour rollback window established

Executive Sponsor - Approval:

- Review migration plan (5 min read)
- Approve $0 investment (migration uses existing infrastructure)
- Authorize go-live decision for Jan 18
- Brief all stakeholders (CTO, CFO, Phase 1 companies)

12:00pm - 3:00pm: DRAGONFLY DEPLOYMENT

DevOps Lead - Cache VM Setup:

1 **SSH into cache-prod-01**

```bash
ssh cache-prod-01.ecosystem.internal
cd /tmp

```

- Connected successfully

4 **Stop old Redis container (if running)**

```bash
docker ps | grep redis
docker stop redis-old  # If exists
docker rename redis-old redis-old-$(date +%s)  # Archive it

```

- Old Redis stopped & archived

8 **Deploy Dragonfly container**

```bash
docker run -d \
```

```
    --name dragonfly \
    --restart unless-stopped \
    --memory 8g \
    --cpus 4 \
    -p 6379:6379 \
    -v /mnt/volumes/dragonfly:/data \
    -e DFLY_aof_fsync_sec=1 \
    -e DFLY_max_memory_policy=allkeys_lru \
    dragonflydb/dragonfly:latest

```

- Container started
- Verify: **`docker ps | grep dragonfly`** (should show RUNNING)

19 **Verify Dragonfly connectivity**

```
bash
sleep 5  # Wait for startup
docker exec dragonfly redis-cli ping
# Expected: PONG

docker exec dragonfly redis-cli INFO server | head -5
# Expected: Dragonfly version info

docker exec dragonfly redis-cli --stat
# Expected: Real-time stats

```

- Dragonfly responding to Redis commands
- Stats showing 0 keys (fresh instance)

29 **Create Dragonfly monitoring script**

```
bash
cat > /opt/monitoring/dragonfly-health.sh << 'EOF'
#!/bin/bash
redis-cli -h cache-prod-01.ecosystem.internal INFO stats
redis-cli -h cache-prod-01.ecosystem.internal DBSIZE
redis-cli -h cache-prod-01.ecosystem.internal INFO memory
EOF
chmod +x /opt/monitoring/dragonfly-health.sh

```

- Health check script created

Infrastructure Lead - Service Configuration:

1 Update connection strings in all services

```text
# For each of 5 services, update environment variables:
CACHE_URL: redis://cache-prod-01.ecosystem.internal:6379/DB_NUMBER
CACHE_TIMEOUT: 30000  # milliseconds
CACHE_MAX_POOL_SIZE: 50
CACHE_RETRY_COUNT: 3

```

- CreditX service: CACHE_URL updated
- Threat service: CACHE_URL updated
- Guardian service: CACHE_URL updated
- Apps service: CACHE_URL updated
- Phones service: CACHE_URL updated

7 Validate connection strings (no typos)

```bash
# On each service VM:
echo $CACHE_URL
# Expected: redis://cache-prod-01.ecosystem.internal:6379/X

```

- All 5 services have correct URL format

11 Create cache pre-warming script

```bash
cat > /opt/cache/prewarm.sh << 'EOF'
#!/bin/bash
redis-cli -h cache-prod-01.ecosystem.internal \
  --pipe < /opt/cache/initial_keys.txt
EOF

```

- Pre-warming script ready (for post-deployment)

3:00pm - 6:00pm: INTEGRATION TESTING

QA Lead - Service Integration Tests:

1 Redeploy all 5 services with new connection strings

```bash
# For each service:
spaceship deploy --service creditx --env staging --strategy blue-green
spaceship deploy --service threat --env staging --strategy blue-green
spaceship deploy --service guardian --env staging --strategy blue-green
spaceship deploy --service apps --env staging --strategy blue-green
spaceship deploy --service phones --env staging --strategy blue-green
```

- CreditX deployment complete (health checks passing)
- Threat deployment complete (health checks passing)
- Guardian deployment complete (health checks passing)
- Apps deployment complete (health checks passing)
- Phones deployment complete (health checks passing)

8 Smoke tests for each service

```bash
# Test 1: Authentication caching
curl -X POST https://api-staging.ecosystem.ai/auth/token \
  -d '{"username": "test", "password": "****"}' \
  -w "\nStatus: %{http_code}\n"
# Expected: 200 (token issued, cached)

# Test 2: Compliance document caching
curl -X POST https://api-staging.ecosystem.ai/creditx/validate \
  -d '{"document_id": "test-001"}' \
  -w "\nStatus: %{http_code}\n"
# Expected: 200 (cached within 1 second)
```

```
19
20  # Test 3: Threat detection caching
21  curl -X GET https://api-staging.ecosystem.ai/threat/cache/status \
22    -w "\nStatus: %{http_code}\n"
23  # Expected: 200 (cache operational)
24
```

- Auth caching working (1-hour TTL)
- Compliance caching working (7-day TTL)
- Threat caching working (15-min TTL)
- Guardian caching working (24-hour TTL)
- Phones caching working (24-hour TTL)

25 **Performance baseline tests**

```
bash
# Test 100 concurrent users
26  locust -f /opt/tests/locust_tasks.py \
27    --users 100 \
28    --spawn-rate 10 \
29    --run-time 5m \
30    --host=https://api-staging.ecosystem.ai
31
```

- Response time P95 < 200ms
- Error rate = 0%
- Throughput > 1K req/sec

32 **Cache hit ratio measurement**

```
bash
# Collect Prometheus metrics
33  curl 'http://prometheus:9090/api/v1/query' \
34    --data-urlencode 'query=rate(cache_hits[5m])'
35
```

- Cache hit ratio recorded (baseline for comparison)

- Expected: ~70% for warm cache

6:00pm - 9:00pm: BLUE-GREEN DEPLOYMENT TO STAGING

DevOps Lead - Staging Deployment:

1 Trigger blue-green deployment

```bash
# Deploy all services to blue environment (new, with Dragonfly)
for service in creditx threat guardian apps phones; do
  spaceship deploy --service $service \
    --env staging \
    --strategy blue-green \
    --wait-ready 300
done
```

- All 5 services deployed to blue
- Health checks passing (200 OK)
- Readiness probes green

9 Traffic shift (slow rollout)

```bash
# Shift 10% traffic to blue
spaceship traffic-shift --service all --blue 10 --green 90
sleep 300  # Wait 5 minutes

# Check error metrics
curl 'http://prometheus:9090/api/v1/query' \
  --data-urlencode 'query=rate(errors_total[5m])'
# Expected: < 0.1% (near zero)

# Shift 50% traffic
spaceship traffic-shift --service all --blue 50 --green 50
sleep 300

# Shift 100% traffic
spaceship traffic-shift --service all --blue 100 --green 0
```

- 10% shift successful (5 min stable)

- 50% shift successful (5 min stable)
- 100% shift successful (full traffic on blue)

25 Validation dashboard review

```bash
# Open Grafana dashboard
open https://monitoring-staging.ecosystem.ai:3000

# Verify metrics:
# - Cache hit ratio: > 70%
# - Latency P95: < 200ms
# - Error rate: < 1%
# - Memory usage: < 8GB
# - CPU usage: < 60%

```

- All metrics within normal range
- Dragonfly running healthy
- No alerts firing

35 Cleanup old environment (optional, can keep 24h)

- Green environment deprecated (kept for rollback)
- Old Redis container archived

End of Day 1 (Jan 16) Status:

```text
✅ Dragonfly deployed and running
✅ All 5 services connected to Dragonfly
✅ Integration tests passing (100%)
✅ Staging deployment successful
✅ Performance baseline established (70% cache hit ratio)
✅ No errors or warnings
🟡 Ready for break testing (Day 2)
```

JAN 17: BREAK TESTING & CHAOS ENGINEERING (24-HOUR WINDOW)

Failure Scenario Testing

Scenario 1: Dragonfly Container Crash (10am - 11am)

```bash
# Action: Kill Dragonfly
docker kill dragonfly
```

```
# Expected behavior:
# 1. Services detect connection loss (5 seconds)
# 2. Circuit breaker opens
# 3. Requests rerouted to database (slower, but working)
# 4. Alerts fire in Slack
# 5. On-call engineer paged

# Monitoring:
# - Error rate spikes to 5-10% (expected)
# - Latency increases 10-20x (expected, using DB)
# - Cache hit ratio drops to 0%

# Recovery:
docker start dragonfly
# Expected: Services reconnect within 30 seconds
# Expected: Error rate returns to < 1%
# Expected: Cache hits resume (rebuilding)

# Verification:
# - [ ] Alert fired in Slack (#production-incidents)
# - [ ] Error logs captured
# - [ ] Services recovered automatically
# - [ ] No manual intervention needed
```

Scenario 2: Network Latency (11:30am - 12:30pm)

```
bash
# Action: Add 500ms latency to cache interface
tc qdisc add dev eth0 root netem delay 500ms

# Expected behavior:
# 1. Cache latency increases to ~500ms+
# 2. Circuit breaker activates after 3 timeouts
# 3. Fallback to database (no latency, but slower throughput)
# 4. Alerts: "High latency detected"

# Monitoring:
# - Latency P95: 600-1000ms (high, expected)
# - Error rate: 0-2% (timeout errors)
# - Throughput: Maintained (fallback working)

# Recovery:
tc qdisc del dev eth0 root

# Verification:
# - [ ] Circuit breaker prevented cascading failures
# - [ ] Fallback mechanism worked
# - [ ] Latency returned to normal within 30s
# - [ ] No data loss
```

Scenario 3: High Concurrency Load (1pm - 2:30pm)

```
bash
# Action: Simulate 500 concurrent users
locust -f locustfile.py --users 500 --spawn-rate 50 --run-time 90m
```

```
# Expected behavior:
# 1. Auto-scaling triggers (more container replicas)
# 2. Connection pool expanded
# 3. Dragonfly memory increases (normal)
# 4. Performance degrades gracefully (P95 < 1000ms acceptable)

# Monitoring:
# - Concurrent connections: 500+ (expected)
# - Memory usage: 4-6GB (normal)
# - CPU: 60-80% (expected under load)
# - Error rate: < 2% (acceptable under stress)
# - Throughput: 10K-15K req/sec

# Verification:
# - [ ] Auto-scaling worked (2+ replicas)
# - [ ] No service crashes
# - [ ] Error rate stayed < 2%
# - [ ] Graceful degradation (not cascading failure)
# - [ ] System recovered after load ended
```

Scenario 4: Partial Cache Failure (3pm - 4pm)

bash

```
# Action: Flush 70% of cache keys
redis-cli -h cache-prod-01.ecosystem.internal FLUSHDB  # Clear DB 0
# Manually delete 70% of keys from other DBs

# Expected behavior:
# 1. Cache hit ratio drops to ~30%
# 2. More requests hit database
# 3. Latency increases (database slower than cache)
# 4. System continues working (no errors)
# 5. Tuning agent adjusts cache policy

# Monitoring:
# - Cache hit ratio: 30% (expected, low)
# - Database queries: 3-4x increase (expected)
# - Latency P95: 300-400ms (higher, acceptable)
# - Error rate: 0% (cache misses don't cause errors)

# Recovery:
# Automatic cache repopulation as requests come in

# Verification:
# - [ ] System continued working despite low cache hit ratio
# - [ ] No errors or crashes
# - [ ] Cache rebuilt over time (hit ratio rising)
# - [ ] Database load handled gracefully
```

Scenario 5: Slow Queries (4:30pm - 5:30pm)

bash

```
# Action: Add 100ms delay to all database queries
# (via connection pooling middleware simulation)
```

```
# Expected behavior:
# 1. Latency increases significantly
# 2. Circuit breaker evaluates (latency P95 > 200ms)
# 3. Alerts fire: "High latency - possible DB issue"
# 4. On-call engineer investigates

# Monitoring:
# - Latency P95: 500-1000ms (high, expected)
# - Error rate: 0% (queries still working, just slow)
# - Throughput: Lower (queuing)

# Recovery:
# Remove query delay

# Verification:
# - [ ] Alerts fired appropriately
# - [ ] System remained stable (no crashes)
# - [ ] Timeout/circuit breaker logic verified
# - [ ] Recovery automatic
```

Scenario 6: Rollback Test (6pm - 7:30pm)

bash

```
# Action: Deploy Redis container and switch connection strings

# Step 1: Start Redis
docker run -d --name redis-old -p 6380:6379 redis:7

# Step 2: Copy data from Dragonfly to Redis (optional)
redis-cli --rdb /tmp/dragonfly.rdb  # Snapshot
# Or: Use empty cache (let it rebuild)

# Step 3: Update connection strings
# Change CACHE_URL from :6379 to :6380 (port change)

# Step 4: Restart services
for service in creditx threat guardian apps phones; do
  spaceship deploy --service $service --env staging
done

# Step 5: Verify services working with Redis
curl https://api-staging.ecosystem.ai/health/ready
# Expected: 200 OK (all services ready)

# Monitoring:
# - Services connected to Redis (different port)
# - API working normally (slower, but functional)
# - Error rate: 0%

# Step 6: Rollback to Dragonfly
docker stop redis-old
# Update connection strings back to :6379
# Restart services
for service in creditx threat guardian apps phones; do
```

```
  spaceship deploy --service $service --env staging
done

# Verification:
# - [ ] Seamless switch to Redis (no downtime)
# - [ ] All tests passed with Redis
# - [ ] Seamless switch back to Dragonfly
# - [ ] No data loss during rollback
# - [ ] Performance returned to normal
```

End of Day 2 (Jan 17) Status:

```
text
✅ Scenario 1: Crash recovery - PASSED
✅ Scenario 2: Network latency - PASSED
✅ Scenario 3: High concurrency - PASSED
✅ Scenario 4: Partial cache failure - PASSED
✅ Scenario 5: Slow queries - PASSED
✅ Scenario 6: Rollback test - PASSED
✅ 24-hour stress test complete
🟢 Ready for production go-live
```

JAN 18: PRODUCTION GO-LIVE (Phase 1 Companies)

9:00am - 12:00pm: PRODUCTION DEPLOYMENT

DevOps Lead - Production Deployment:

1. Pre-deployment checks

```
bash
# Verify staging is stable (after 24h break testing)
curl https://api-staging.ecosystem.ai/health/live  # Expected: 200
curl https://api-staging.ecosystem.ai/health/ready  # Expected: 200

# Check Dragonfly status on production cache VM
ssh cache-prod-01.internal
docker ps | grep dragonfly  # Expected: RUNNING
redis-cli ping  # Expected: PONG
redis-cli DBSIZE  # Expected: (integer) 0 (fresh, will populate)

# Verify production load balancer ready
spaceship lb status --name ecosystem-api-lb  # Expected: ACTIVE
spaceship lb test --name ecosystem-api-lb  # Expected: OK
```

- Staging stable (100% uptime, 0% errors)
- Production Dragonfly ready

- Production load balancer ready
- All systems green

15 Blue-green deployment to production

```bash
# Verify current production is green (stable)
spaceship environment status --env production
# Expected: all pods running, health check 200 OK

# Deploy to blue (new environment with Dragonfly)
for service in creditx threat guardian apps phones; do
  spaceship deploy --service $service \
    --env production \
    --strategy blue-green \
    --wait-ready 300
done
```

- All services deployed to blue
- Health checks passing (200 OK)
- Ready probes green

27 Smoke tests on blue (production)

```bash
# Test 1: Authentication
curl -X POST https://api.ecosystem.ai/auth/token \
  -H "X-Blue-Green: blue" \
  -d '{"username": "test", "password": "****"}'
# Expected: 200 OK

# Test 2: Compliance validation
curl -X POST https://api.ecosystem.ai/creditx/validate \
  -H "X-Blue-Green: blue" \
  -d '{"document_id": "doc-prod-001"}'
# Expected: 200 OK

# Test 3: Threat detection
curl -X GET https://api.ecosystem.ai/threat/status \
  -H "X-Blue-Green: blue"
# Expected: 200 OK
```

- Auth working on blue
- CreditX working on blue
- Threat detection working on blue
- Guardian working on blue
- Phones working on blue

Production traffic shift

```bash
# Shift 10% traffic to blue (production)
spaceship traffic-shift --env production --blue 10 --green 90
sleep 300  # Monitor for 5 minutes

# Check errors
curl 'http://prometheus:9090/api/v1/query' \
  --data-urlencode 'query=rate(errors_total[5m])'
# Expected: < 0.1% error rate (near zero)

# Shift 50%
spaceship traffic-shift --env production --blue 50 --green 50
sleep 300

# Shift 100%
spaceship traffic-shift --env production --blue 100 --green 0

```

- 10% shift: stable (0% errors for 5 min)
- 50% shift: stable (0% errors for 5 min)
- 100% shift: stable (full traffic on blue/Dragonfly)

Production metrics validation

```bash
open https://monitoring.ecosystem.ai:3000

# Verify metrics:
# Dashboard: "Ecosystem Production Dragonfly"
# - Cache hit ratio: building toward 70%+
# - Latency P95: < 200ms
# - Error rate: 0%
# - Memory: 1-3GB (building)
# - CPU: 20-40% (light load, morning)
# - Connections: 50-100
```

- All metrics within target range
- No alerts firing
- Dragonfly memory growing (normal, rebuilding cache)

12:00pm - 12:30pm: PHASE 1 COMPANY GO-LIVE NOTIFICATIONS

Customer Success Lead - Stakeholder Communication:

1. **Send go-live notification to 5 Phase 1 companies**

```
text
Subject: Ecosystem Platform – Phase 1 Go-Live (Jan 18, 2026)

Dear Leadership,

The Ecosystem platform is now live in production!

🎉 What's live:
• CreditX compliance automation (Revau)
• 91 Apps business automation (Nuvei, Master Group)
• Global AI Alert threat detection (Spectrum, Comm Tower)
• Guardian AI endpoint security (Spectrum)
• Stolen Lost Phones device recovery (Spectrum)

📊 Early metrics (will be available in your dashboards within 1 hour):
• Cache performance: 25x faster than industry baseline
• API availability: 99.5% SLA
• Response times: < 200ms average

🔗 Access your dashboards:
• Nuvei: https://dashboard.ecosystem.ai/nuvei
• Revau: https://dashboard.ecosystem.ai/revau
• Spectrum: https://dashboard.ecosystem.ai/spectrum
• Master Group: https://dashboard.ecosystem.ai/master
• Comm Tower: https://dashboard.ecosystem.ai/commtower

📞 Support available 24/7 at support@ecosystem.ai

Welcome to Phase 1 of the Ecosystem! 🚀
```

- Notification sent to Nuvei CEO + CFO

- Notification sent to Revau CEO + CIO
- Notification sent to Spectrum CEO + CIO
- Notification sent to Master Group CEO + Operations
- Notification sent to Comm Tower CEO + Network Ops

30 **Enable real-time dashboards for Phase 1 companies**

```bash
# Verify dashboards are live
curl https://api.ecosystem.ai/dashboard/nuvei/ebitda
# Expected: Real-time EBITDA impact tracking

curl https://api.ecosystem.ai/dashboard/revau/compliance
# Expected: Compliance metrics

```

- Nuvei dashboard live (lead scoring metrics)
- Revau dashboard live (compliance metrics)
- Spectrum dashboard live (breach detection metrics)
- Master Group dashboard live (supply chain metrics)
- Comm Tower dashboard live (network threat metrics)

12:30pm - 3:00pm: 24/7 OPERATIONS HANDOFF

On-Call Engineer - Production Monitoring:

1 **Take ownership of production monitoring**

```bash
# Set up PagerDuty on-call rotation

# Create alert rules:
# - Error rate > 1%: Page engineer immediately
# - Latency P95 > 500ms: Alert in Slack
# - Dragonfly memory > 7GB: Alert in Slack
# - Cache hit ratio < 50%: Warning in Slack

# Verify alerting is working
echo "Testing alert: artificial error" >> /var/log/ecosystem/errors.log
# Expected: Alert fired in Slack within 1 minute

```

- PagerDuty configured with on-call rotation
- Alert rules created (P1, P2, P3 severity)
- Slack channels integrated (#production-incidents)
- Alert test fired successfully

12 Verify runbooks are accessible

```bash
# Store runbooks in accessible location
cat /opt/runbooks/incident-response.md | wc -l
# Expected: Comprehensive guide available

# Create quick reference for on-call engineer
cat /opt/runbooks/quick-reference.txt
# Expected: Top 5 incidents + resolution steps

```

- Runbooks documented & accessible
- Quick reference card printed/available
- Escalation contacts on display

20 Monitor real-time metrics (Phase 1 companies)

```bash
# Track EBITDA impact in real-time
# Check dashboards every 15 minutes for first 2 hours

# Expected by end of Day 1:
# - Nuvei 91 Apps: +$25M/year pace
# - Revau CreditX: +$18M/year pace
# - Spectrum Guardian AI: +$8M/year pace
# - Master Group 91 Apps: +$15M/year pace
# - Comm Tower Global AI Alert: +$6M/year pace
# Total Phase 1: $72M/year pace

```

- EBITDA dashboards tracking
- Metrics aligned with projections

- Business impact validated

3:00pm+: ONGOING MONITORING & OPTIMIZATION

DevOps Lead - Performance Optimization:

1 Monitor cache hit ratio growth

```bash
# First hour: 0-20% (cache empty)
# Hour 1-2: 20-50% (warming up)
# Hour 2-6: 50-70% (optimal)
# Hour 6+: 70-85% (mature)

```

- Cache hit ratio progression tracked
- No performance regressions
- Dragonfly memory usage reasonable (< 5GB)

6 Tuning agent optimization

```bash
# Tuning agent continuously optimizes:
# - Cache TTL values
# - Connection pool sizes
# - Auto-scaling thresholds
# - Load shedding policies

# Monitor tuning agent logs
tail -f /var/log/ecosystem/tuning-agent.log

```

- Tuning agent running (auto-optimization active)
- TTLs adjusted based on hit ratio
- Performance improving over time

15 Recovery agent readiness

```bash
# Verify recovery agent active
curl https://api.ecosystem.ai/agents/recovery/status
```

```
# Expected: { "status": "ready", "circuits": "normal" }

# Verify circuit breakers all CLOSED (healthy)
curl https://api.ecosystem.ai/circuits/status

```

- Recovery agent ready (auto-healing active)
- All circuit breakers CLOSED (healthy state)
- Agents monitoring for failures

End of Day 3 (Jan 18) Status:

```
text
✅ Production blue-green deployment complete
✅ All 5 Phase 1 companies online
✅ Real-time dashboards showing EBITDA impact
✅ 99.5% uptime SLA maintained
✅ Cache hit ratio building (20-50% by end of day)
✅ Zero errors or critical incidents
✅ 24/7 monitoring operational
✅ On-call engineer ready for production support
🎉 PHASE 1 GO-LIVE SUCCESSFUL!

Next: Monitor for 7 days (Jan 18-25), then Phase 2 kickoff
```

SUCCESS CRITERIA (72-Hour Delivery)

✅ Jan 16 (Dragonfly Migration)

- Dragonfly deployed & running
- All 15 services connected
- Integration tests 100% passing
- Staging deployment successful
- Zero data loss

✅ Jan 17 (Break Testing)

- 6 failure scenarios tested & recovered
- Rollback tested & working
- 24-hour stability validated
- 25x performance improvement verified
- All metrics within target range

✅ Jan 18 (Production Go-Live)

- Production blue-green deployment
- 5 Phase 1 companies online
- EBITDA impact dashboards live
- 99.5% uptime maintained
- Zero production incidents
- 24/7 support operational

SIGN-OFF

- **DevOps Lead:** Dragonfly deployment approved & complete
- **QA Lead:** Break testing complete, zero critical issues
- **Infrastructure Lead:** Rollback procedures verified
- **On-Call Engineer:** Production monitoring ready
- **Project Manager:** Phase 1 go-live approved
- **Executive Sponsor:** Production deployment authorized

🚀 READY TO EXECUTE - 72 HOURS TO PHASE 1 GO-LIVE

Next Steps:

1. Approve this checklist
2. Start Day 1 (Jan 16) at 9am MST
3. Follow checklist sequentially
4. Report status updates to #ecosystem-dragonfly-migration
5. Escalate blockers immediately to DevOps Lead

Contact:

- DevOps Lead: +1-780-555-0100
- On-Call Engineer: PagerDuty (@oncall)
- Project Manager: @project-mgr on Slack

Document Status: READY FOR EXECUTION
Timeline: Jan 16-18, 2026 (72 hours)
Target: Phase 1 production go-live by Jan 18 9am MST
ROI: $72M annualized EBITDA impact begins Jan 18