Contents

Cato Incident Response Runbook

Severity Levels

Level	Description	Response Time	Example
SEV1	Complete outage, all users affected	5 minutes	Shadow Self down
SEV2	Degraded service, >50% affected	15 minutes	High latency
SEV3	Minor degradation, <50% affected	1 hour	Cache miss spike
SEV4	Cosmetic/minor, no user impact	24 hours	Dashboard error

Incident Response Process

1. Detection

Automatic Alerts: - CloudWatch alarms → PagerDuty → On-call - Custom metrics → Slack #cato-alerts

Manual Detection: - User reports via support - Dashboard anomalies

2. Triage

```
1. Acknowledge alert in PagerDuty
2. Join #cato-incident Slack channel
3. Assess severity level
```

4. Start incident log

3. Mitigation

First 5 Minutes: - Check dashboard for obvious issues - Review recent deployments - Check AWS Health Dashboard

Mitigation Strategies: - Failover to healthy region - Scale up resources - Enable emergency mode - Rollback recent changes

4. Resolution

- Fix root cause
- Verify service restored
- Close incident

5. Post-Incident

- Blameless postmortem within 48 hours
- Update runbooks with learnings
- Implement preventive measures

Common Incidents

Shadow Self Endpoint Down

Symptoms: - 5XX errors from `/api/admin/cato/shadow-self/*` - High latency on introspection queries - Circuit breaker OPEN

Diagnosis:

```
# Check endpoint status
aws sagemaker describe-endpoint --endpoint-name cato-shadow-self

# Check CloudWatch logs
aws logs filter-log-events \
  --log-group-name /aws/sagemaker/Endpoints/cato-shadow-self \
  --filter-pattern "ERROR"
```

Mitigation:

```
# 1. Check if instances are healthy
aws sagemaker describe-endpoint --endpoint-name cato-shadow-self \
  --query 'ProductionVariants[0].CurrentInstanceCount'

# 2. If 0 instances, restart endpoint
aws sagemaker update-endpoint \
  --endpoint-name cato-shadow-self \
  --endpoint-config-name cato-shadow-self-config
```

```
# 3. If still failing, rollback to previous config
aws sagemaker list-endpoint-configs --name-contains cato-shadow-self
aws sagemaker update-endpoint \
  --endpoint-name cato-shadow-self \
  --endpoint-config-name cato-shadow-self-config-v1
```

Root Cause Investigation: - Check for OOM errors (model too large) - Check for GPU driver issues - Check for container crash loop

---

High Latency

Symptoms: - p99 latency > 2 seconds - User complaints about slow responses - Timeout errors

Diagnosis:

```
# Check component latencies
aws cloudwatch get-metric-statistics \
  --namespace AWS/SageMaker \
  --metric-name ModelLatency \
  --dimensions Name=EndpointName,Value=cato-shadow-self \
  --start-time $(date -u -v-1H +%Y-%m-%dT%H:%M:%SZ) \
  --end-time $(date -u +%Y-%m-%dT%H:%M:%SZ) \
  --period 60 \
  --statistics p99
```

Mitigation:

```
# 1. Scale up Shadow Self
aws sagemaker update-endpoint-weights-and-capacities \
  --endpoint-name cato-shadow-self \
  --desired-weights-and-capacities VariantName=primary,DesiredInstanceCount=50

# 2. Scale up Ray Serve
kubectl scale deployment cato-orchestrator -n cato --replicas=50

# 3. Check cache hit rate - if low, investigate cache issues
```

Root Causes: - Insufficient capacity - Cache miss spike - Bedrock throttling - Network latency

---

Cache Miss Spike

Symptoms: - Cache hit rate drops below 70% - Higher than expected LLM costs - Increased latency

Diagnosis:

```
# Check cache stats
redis-cli -h cato-cache.xxx.use1.cache.amazonaws.com INFO stats
```

```
# Check for recent cache invalidations
aws cloudwatch get-metric-statistics \
  --namespace Custom/Cato \
  --metric-name CacheInvalidations
```

Mitigation:

```
# 1. Check if learning update invalidated too much
# Review recent domain updates

# 2. If cache is undersized, scale up
aws elasticache modify-replication-group \
  --replication-group-id cato-cache \
  --cache-node-type cache.r7g.2xlarge
```

Root Causes: - Aggressive cache invalidation after learning - Cache eviction due to size limits - Query pattern change

---

Budget Exceeded (Emergency Mode)

Symptoms: - Mode shows "EMERGENCY" in dashboard - Curiosity exploration stopped - Limited responses

Diagnosis:

```
# Check budget status
curl -H "Authorization: Bearer $TOKEN" \
  https://api.cato.thinktank.ai/api/admin/cato/budget/status
```

Mitigation:

```
# 1. Increase budget if approved
curl -X PUT \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"monthlyLimit": 1000}' \
  https://api.cato.thinktank.ai/api/admin/cato/budget/config

# 2. Or wait for next month reset
```

Prevention: - Set appropriate budget limits - Monitor spend daily - Enable budget alerts

---

Bedrock Throttling

Symptoms: - `ThrottlingException` in logs - Increased error rate - Fallback to Haiku/static responses

Diagnosis:

```
# Check Bedrock quotas
aws service-quotas get-service-quota \
  --service-code bedrock \
  --quota-code L-XXXXXXXX
```

Mitigation:

```
# 1. Request quota increase
aws service-quotas request-service-quota-increase \
  --service-code bedrock \
  --quota-code L-XXXXXXXX \
  --desired-value 10000

# 2. Enable more aggressive caching
# Reduce cache similarity threshold temporarily

# 3. Shift more traffic to self-hosted Shadow Self
```

DynamoDB Hot Partition

Symptoms: - `ProvisionedThroughputExceededException` - Slow memory reads/writes - Specific domains affected

Diagnosis:

```
# Check consumed capacity by partition
aws cloudwatch get-metric-statistics \
  --namespace AWS/DynamoDB \
  --metric-name ConsumedReadCapacityUnits \
  --dimensions Name=TableName,Value=cato-semantic-memory
```

Mitigation:

```
# 1. Switch to on-demand billing if not already
aws dynamodb update-table \
  --table-name cato-semantic-memory \
  --billing-mode PAY_PER_REQUEST

# 2. Add GSI to distribute load
# Requires table redesign

# 3. Enable DAX for read caching
```

Emergency Contacts

Role	Contact	Escalation
On-call Engineer	PagerDuty	Auto

Role	Contact	Escalation
Engineering Lead	@lead-eng Slack	15 min
VP Engineering	Phone	30 min (SEV1 only)
AWS TAM	aws-support@company.com	As needed

Post-Incident Template

```
# Incident Post-Mortem: [TITLE]

**Date:** YYYY-MM-DD
**Duration:** X hours Y minutes
**Severity:** SEVX
**Author:** [Name]

## Summary
Brief description of what happened.

## Timeline
- HH:MM - Event detected
- HH:MM - On-call paged
- HH:MM - Mitigation started
- HH:MM - Service restored

## Root Cause
What caused the incident.

## Impact
- Users affected: X
- Revenue impact: $Y
- SLA impact: Z minutes

## Mitigation
What was done to fix it.

## Prevention
Action items to prevent recurrence:
- [ ] Action 1 - Owner - Due date
- [ ] Action 2 - Owner - Due date

## Lessons Learned
What we learned from this incident.
```