

What Makes a Production- Ready Agent System?

Complete Checklist • Common Gaps •
Evaluation Framework • Migration Path

Systems Ship, Demos Don't





Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

The 3 AM Page

February 12th, 2024. 3:47 AM.

PagerDuty alert: "Agent system down. 1,247 users affected."

-  Worked perfectly in staging
-  100 test cases passed
-  Demo impressed stakeholders
-  Complete failure in production after 3 hours

The Question

What's the difference between a POC and a production-ready system?

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

What Does Production-Ready Mean?

✗ Most Teams Think

"The agent works, ship it."

✓ Reality

"Works reliably under **all conditions**"

All Conditions Means:

- External API timing out
- Unexpected user input
- Cost spike at 3 AM

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

- PII accidentally leaked
- LLM rate limited
- Database connection dropped
- Network partition
- Memory leak after 10,000 requests

The Production-Ready Checklist

Five Categories

1. Architecture

Separated layers, retry logic, circuit breakers

2. Observability

Tracing, cost tracking, alerts

3. Security

Validation, PII detection, rate limiting

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

4. Compliance

Data residency, audit logs, human-in-loop

5. Operations

Deployment pipeline, rollback, runbooks

Most teams: 30-40% checked

Production systems: 100% checked

Category 1:

Architecture

- ❑ **Separated Layers** - Not monolithic
- ❑ **Clear Failure Domains** - Isolated failures
- ❑ **Retry Logic** - Exponential backoff for external calls
- ❑ **Circuit Breakers** - Stop calling failed services

POC Architecture vs Production

POC

```
main.py  
(everything in one file)
```

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Production

```
|— api/  
|— agent/  
|— tools/  
└— database/
```

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Circuit Breaker Example

```
circuit_breaker = CircuitBreaker(  
    failure_threshold=10,  
    recovery_timeout=60  
)  
  
if circuit_breaker.is_open():  
    return cached_response  
else:  
    try:  
        return external_api.call()  
    except:  
        circuit_breaker.record_failure()
```

Real Incident: The \$2,000 Runaway Loop

API timing out. Agent retrying infinitely. No circuit breaker.

40,000 failed API calls in 3 hours

\$0.05 per call = \$2,000 wasted

Circuit breaker would have cost: \$0

Category 2:

Observability

- ❑ **Distributed Tracing** - See entire request flow
- ❑ **Cost Per Request** - Track LLM + API costs
- ❑ **Error Correlation** - Which user? Which tool? What input?
- ❑ **Alert Thresholds** - Critical, Warning, Info levels

Alert Configuration

Level	Condition	Action
Critical	Error rate > 10%	PagerDuty
Warning	Latency > 30s	

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Info

Daily cost summary

Email

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Real Incident: The Slow Agent Mystery

95% of requests: 2 seconds
5% of requests: **45 seconds**
No tracing = No idea why

The Fix

- Added LangSmith tracing
- Found: One specific tool timing out on 5% of requests
- Root cause: No exponential backoff on retries
- Fix: 5 minutes

Result: P95 latency dropped from 45s to 2s

Finding without tracing: Would have taken days

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Category 3: Security

- ❑ **Input Validation** - Never trust user input
- ❑ **PII Detection** - SSN, credit cards, phone numbers
- ❑ **Rate Limiting** - Per user, per IP, per API key
- ❑ **Auth + AuthZ** - Who are you? What can you do?

Rate Limiting Example

```
@rate_limit(requests=100, window=3600) # 100/hour
def agent_endpoint(user_id):
    return agent.run()
```

Without rate limiting: One user can consume entire API budget

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Real Incident: The \$8,000 Bill

The Problem

No rate limiting. One user's infinite loop.

160,000 calls in one day

\$0.05 per call = \$8,000

With Rate Limiting

100 calls/hour limit would have stopped it

Actual cost: \$5 instead of \$8,000

3 lines of code saved \$7,995

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Category 4: Compliance

- **Data Residency** - EU users → EU servers
- **Audit Logging** - 6 years (HIPAA), 7 years (financial)
- **Human-in-Loop** - For high-risk actions
- **Retention Policies** - How long to keep data

High-Risk Actions

- Financial transactions > \$1,000
- Data deletion
- Account changes
- Medical diagnoses
- Legal advice

Real Incident: The Compliance Audit

The Failure

Healthcare agent. No audit logs.

HIPAA audit: **Failed**

Fine: \$50,000

The Costs

Implementing After Fine

\$50,000 + 6 weeks of work

Join the Community – community.nachiketh.in

More Launch

Scroll to navigate
Production-Ready Agent Systems

2 days of work

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Category 5: Operations

- ❑ **Automated Deployment** - CI/CD pipeline
- ❑ **Rollback Strategy** - < 5 minutes to rollback
- ❑ **Incident Runbooks** - Documented fixes
- ❑ **Monitoring Dashboards** - Real-time metrics

Deployment Pipeline

1. Commit code
2. Run tests (unit, integration, E2E)
3. Deploy to staging
4. Run smoke tests
5. Deploy to 10% of production
6. Monitor for 1 hour
7. Deploy to 100% if healthy

Five Gaps Between POC and Production

Gap	POC	Production
Error Handling	Basic try-catch	Retry, fallbacks, circuit breakers
Cost Tracking	None	Per-request attribution
Observability	Print statements	Distributed tracing
Security	Hardcoded keys	Key management, rate limiting
Testing	Manual	Automated pipelines

Bridging these gaps: 4-6 weeks of work

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Gap 1: Error Handling

POC Error Handling

```
try:
    response = llm.call(prompt)
    return response
except Exception as e:
    return f"Error: {e}"
```

Production Error Handling

```
@retry(max_attempts=3, backoff=exponential)
@circuit_breaker(threshold=5, timeout=60)
async def call_llm(prompt):
    try:
        response = await llm.call(prompt)
        return response
    except APITimeout:
        return await secondary_llm.call(prompt)
    except RateLimit as e:
        await asyncio.sleep(e.retry_after)
        return await llm.call(prompt)
```

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

```
except APIError:  
    return cache.get_similar_response(prompt)
```

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Gap 2: Cost Tracking

Real Incident: The Invisible Cost Spike

No cost tracking. Monthly AWS bill: \$4,000 → \$12,000

No idea why.

Spent 2 weeks investigating.

Found: One user's automation script. 50,000 requests/day.

With Cost Attribution

Would have found in 5 minutes

Time saved: 2 weeks = \$40,000

Cost to implement: 1 day = \$2,000

ROI: 20x

Production Readiness Score

Objective Scoring System

Five categories. Each worth 20 points. Total: 100 points.

70+ Points

Production-ready ✓

50-69 Points

Needs work

<50 Points

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Scoring Breakdown

- **Architecture:** 20 points (4 items × 5 points)
- **Observability:** 20 points (4 items × 5 points)
- **Security:** 20 points (4 items × 5 points)
- **Compliance:** 20 points (4 items × 5 points)
- **Operations:** 20 points (4 items × 5 points)

Real System Examples

Healthcare Agent: 85/100 ✓ Production-Ready

Architecture	18/20
Observability	20/20
Security	20/20
Compliance	20/20
Operations	7/20

Customer Service Agent: 62/100 ⚠ Needs Work

Architecture	15/20
Observability	10/20
Security	

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

4-Week Migration Plan

Week 1: Observability

Distributed tracing, cost tracking, alerts

20 points

Week 2: Security

Validation, PII detection, rate limiting, auth

20 points

Week 3: Architecture

Separate layers, retry logic, circuit breakers

20 points

Join the Community – community.nachiketh.in

ce + Ops

Scroll to navigate
Production-Ready Agent Systems

Data residency, audit logs, deployment pipeline

40 points

Total: 160 hours = 4 weeks = \$32,000 in engineering cost

Compare to: \$50,000 incidents, \$100,000 breaches, \$50,000 fines

ROI: 3-5x in first year

Staged Rollout Strategy

5 Stages Over 2 Weeks

Stage	Traffic %	Duration	Goal
Internal Testing	0%	3 days	Fix obvious bugs
Beta Users	0%	4 days	Validate with real users
10% Rollout	10%	3 days	Catch scale issues
50% Rollout	50%	2 days	Verify performance
100% Rollout	100%	Ongoing	Monitor closely

Don't Deploy to 100% on Day One

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Staged rollout finds issues before they affect thousands of users

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Post-Launch Monitoring

First 72 Hours: Constant Monitoring

- Check dashboards every hour
- Alert channel open 24/7
- Engineer on-call
- Incident response team ready

Success Metrics

Metric	Target
Uptime	>99.9% (<43 min downtime/month)
Error Rate	<0.1% (1 error per 1,000 requests)
P95 Latency	<5 seconds

Cost per Request

Within 10% of budget

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Summary

Production-Ready Checklist

- Architecture: Separated layers, retry logic, circuit breakers
- Observability: Tracing, cost tracking, alerts
- Security: Validation, PII detection, rate limiting, auth
- Compliance: Data residency, audit logs, human-in-loop
- Operations: Deployment pipeline, rollback, runbooks

The Gap

Most Teams

30-40% checked

Production Systems

100% checked

Bridging the gap: 4-6 weeks of focused work

Investment: \$32,000 in engineering time

Return: Avoid \$50,000+ incidents

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Ready to Ship Production Systems?

Join the Community

If you just want to follow along, discuss, and learn over time. Join the community:

<https://community.nachiketh.in>

OR

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Agentic AI Enterprise Mastery Bootcamp

**If you want a structured path, Join the Agentic AI
Enterprise Mastery Bootcamp:**

<https://bootcamp.nachiketh.in>

Production isn't about working.
It's about working reliably under all conditions.

Join the Community – community.nachiketh.in

Scroll to navigate
Production-Ready Agent Systems

Join the Community – community.nachiketh.in

Scroll to navigate
[Production-Ready Agent Systems](#)