KoachSmart: Deployment and Infrastructure Readiness Plan

Disclosure on Research Methodology: In order to prepare these answers, I used Perplexity/Claude as a research and validation tool. I want to be clear, though: even though I have a solid understanding of the architecture and principles, I lack practical production experience overseeing infrastructure on my own. With the right help and direction, I'm determined to acquire these abilities through this initiative. Instead of making assumptions that could lead to problems later, I will record all I learn and be ready to seek assistance when necessary.

1. Load Assumptions & Traffic Spikes

- What are the estimated concurrency numbers (e.g., 50, 100, 300 users) during a recruitment drive?
Ans:- 50
- Since resume uploads are I/O heavy, how will we prevent them from blocking Gunicorn workers?
By using direct-to-object-storage uploads
- Are you planning direct-to-object-storage uploads (presigned POST) or proxying via Flask?
direct-to-object-storage uploads
- Action: Please document the expected concurrency range and confirm the upload flow.

During recruitment efforts, KoachSmart anticipates 50 concurrent users; if this number consistently exceeds 40, it will scale up. By completely avoiding the Flask proxy, direct-to-object-storage uploads utilizing presigned POST URLs avoid Gunicorn worker blocking.Why: Direct uploads offload to storage providers like Cloudflare R2 or Backblaze B2; I/O-heavy resume uploads (500KB–2MB) tie up workers and cause queueing and latency spikes for other requests.How: Frontend asks backend for a 15-minute presigned URL, uploads straight to storage, and alerts backend to save metadata in Neon Postgres. With no server stress, this can process about 200 uploads in 48 hours.

2. CI/CD & Rollback Reality Check

- How will partial DB migration failures be handled, given that Alembic rollbacks can be complex?
I have no idea about Alembic rollbacks.
- Do we have a staging environment (e.g., a Neon branch or separate lightweight VPS) to test releases?
Yes we have a staging/development environment on Neon
- Action: Please share a plan for a minimal staging setup.

At present, I do not yet have hands-on production experience executing and rolling back Alembic migrations independently. I understand the conceptual flow of schema migrations, but I want to be transparent that rollback mechanics (especially in production environments) are an area I am actively learning.

Risk Acknowledgement

- Schema migrations are inherently risky if partially applied.
- Downgrades are not always trivial if data transformations or destructive changes are involved.
- Relying purely on automated rollback without validation can introduce data inconsistency.

Mitigation Plan (Conservative Approach)

1. **Staging First Always**
   - Every migration will be tested on a Neon staging branch before production.
   - Application smoke tests will validate schema compatibility.
2. **Backward-Compatible Migrations**
   - Prefer additive changes (new columns, nullable fields) over destructive changes.
   - Avoid dropping columns or renaming in the same release whenever possible.
3. **Pre-Deployment Backup**
   - Take a database snapshot before production migration (Neon snapshot).
   - This allows recovery even if downgrade scripts are unsafe.
4. **Controlled Rollout**
   - Run migrations manually during deployment (not fully automated initially).
   - Validate app health before opening traffic.
5. **Skill Ramp-Up**
   - I will practice Alembic upgrade / downgrade workflows in a sandbox project.

    - Document real rollback procedures in the infra runbook before production usage.

Staging Setup

- Lightweight Hetzner VPS
- Neon staging branch database
- Staging-prefixed object storage bucket

This approach prioritizes safety and learning while reducing operational risk.

3. Monitoring & Alerting
- We need to define concrete alerting. Which specific triggers will notify you of an issue (e.g., CPU/RAM spikes, disk limits, 5xx errors, or Neon connection issues)?
- Action: Define 5–7 essential alerts for day-one production.

Day-one alerts cover CPU >80% (10min), RAM >85% (10min), P99 latency >2s, 5xx >1% (5min), Neon pool exhaustion, disk >85%, Gunicorn restarts >5/hour. Use tools like Monit or Datadog free tier for email notifications.

Why: Proactive triggers catch issues before outages; symptom-based (latency) over raw metrics spots user impact early.

How: Install Monit on VPS for CPU/RAM/disk; Prometheus exporter for Gunicorn restarts and Neon latency; external UptimeRobot for 5xx.

4. Security & Compliance
- The approach to secrets and file access is solid.
- Action: Please confirm if basic audit logs (tracking who accessed which resume and when) will be implemented at the application level for compliance.

WHY This Matters

Legal and Compliance:

- If a resume is misused, we need to know who accessed it
- GDPR/data protection requires tracking access to personal data
- In case of disputes, audit trail protects the company

Security:

- Detect unauthorized access patterns
- Identify insider threats
- Forensics after security incidents

What Gets Logged

Critical Actions:

- ✅ Resume uploaded
- ✅ Resume viewed
- ✅ Resume downloaded
- ✅ Resume deleted
- ✅ User login/logout
- ✅ User created/modified/deleted
- ✅ Permission changes
- ✅ Failed access attempts (403 errors)

Not Logged (too noisy):

- General page views
- API health checks
- Static file requests

5. Cost Projection
- Object storage and egress costs can vary.
- Action: Please sanity-check storage and egress assumptions based on realistic resume sizes and provide a "worst reasonable month" cost estimate.

WHY This Matters

Budget Planning:

- Need realistic estimates to get approval
- Prevent surprise bills
- Know when to optimize vs. upgrade

Sustainability:

- Ensure costs scale predictably with usage
- Avoid free tier traps that become expensive suddenly

Worst Reasonable Month" Scenario

Assumptions for Heavy Recruitment Month:

- 3 simultaneous recruitment drives

- 1000 total applications
- 1000 resume uploads (avg 1.5MB each)
- 5000 resume views (recruiters screening)
- 500 resume downloads

Calculations:

Storage:

- Resumes: 1000 × 1.5MB = 1.5GB
- Database: 0.5GB (candidate data, applications)
- **Total: 2GB** (still within 250GB free tier)

Egress (Bandwidth):

- Resume views (streaming): 5000 × 1.5MB = 7.5GB
- Resume downloads: 500 × 1.5MB = 0.75GB
- **Total: 8.25GB** (still within 250GB free tier)

When Costs Will Increase

Trigger Point 1: Database > 0.5GB or Compute > 191h/month

- **Happens when:** ~5000+ active candidates in database
- **Action:** Upgrade to Neon Launch plan: ₹1560/month ($19)
- **New monthly total:** ₹2033 ($25)

Trigger Point 2: Object Storage > 10GB

- **Happens when:** ~6000+ resumes stored
- **Action:** Already in free tier up to 250GB, no action needed
- **If we hit 250GB:** Costs increase by ~₹390/month for next 250GB

Trigger Point 3: VPS CPU/RAM Insufficient

- **Happens when:** Consistent >70% CPU or >80% RAM
- **Action:** Upgrade to CX31 (2 vCPU, 8GB RAM): ₹635/month (€6.49)
- **New monthly total:** ~₹1278

6. Ownership & Continuity
- To mitigate single-point-of-failure risk, we need to ensure the infrastructure is documented.
- Action: Outline an infra runbook covering the architecture diagram, deployment steps, scaling triggers, and incident response basics.

Runbook Maintenance Plan

How I'll Keep It Updated:

1. **After Every Incident:**
   - Document what went wrong
   - Add solution to runbook
   - Update incident response section
2. **After Every Deployment:**
   - Note any new steps required
   - Update deployment procedure if changed
3. **Monthly Review:**
   - Read through entire runbook
   - Verify commands still work
   - Update contact information
4. **Version Control:**
   - Runbook lives in Git repository (docs/infrastructure/)
   - Changes tracked via commits
   - Team can review/suggest improvements

Knowledge Transfer Plan

How Others Can Learn:

1. **Onboarding New Team Member:**
   - Share runbook as first resource
   - Pair with them on first deployment
   - Have them handle second deployment (with supervision)
   - Third deployment = they're independent
2. **Emergency Backup:**
   - Designate backup person
   - Walk them through runbook once
   - Give them read-only access to infrastructure
   - Full access credentials in secure vault
3. **Documentation Locations:**
   - Runbook: adding it on gitrepo