Aurora AI – Deployment Guide

---

1. Overview

Aurora AI can be deployed in three modes:

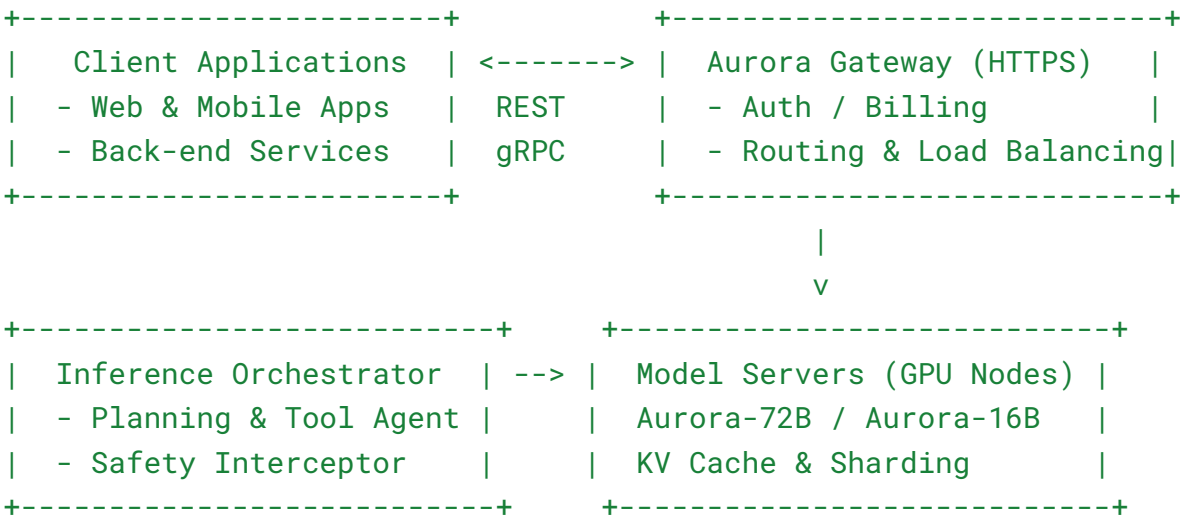
1. **Cloud-Hosted (Managed)** – Aurora is hosted by the Aurora Platform.
2. **Hybrid Deployment** – Aurora model + vector store runs in your infrastructure; coordination & safety layers run in Aurora Cloud.
3. **Self-Hosted (Enterprise Edition)** – Full model hosting, orchestration, safety, and inference layers deployed entirely in your VPC or on-prem cluster.

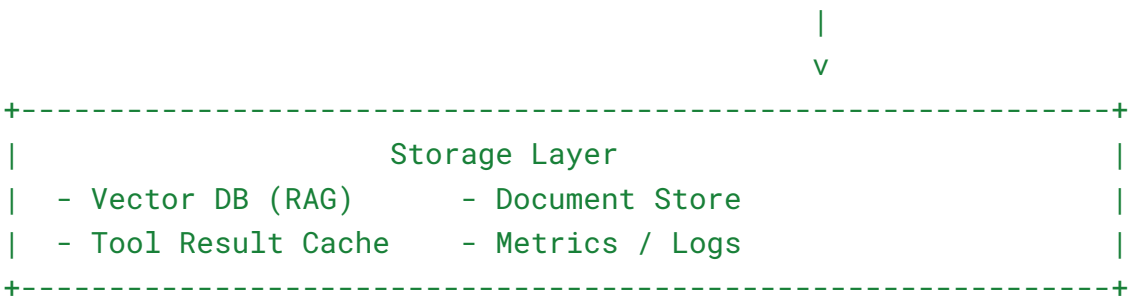
This document describes the architecture, prerequisites, resource requirements, deployment steps, and operational best practices for all modes.

---

2. Architecture Overview

2.1 High-Level Architecture Diagram (Textual)





---

3. System Requirements

3.1 Hardware Requirements for Self-Hosting

Aurora-72B (flagship model)

Component	Requirement
GPUs	8–32 × A100 80GB **or** H100 80GB
GPU Interconnect	NVLink or NVSwitch recommended
CPU	64+ vCPUs
RAM	512 GB minimum
Storage	4–8 TB NVMe SSD (RAID recommended)
Network	40–100 Gbit InfiniBand or equivalent

Aurora-16B

Component	Requirement
GPUs	2–4 × A100 40/80GB
RAM	128–256 GB
Network	10–25 Gbit

Aurora-3B Edge

Can run on consumer-grade hardware:

Component	Requirement
GPUs	1 × RTX 4090 or A6000
RAM	64 GB
Disk	1 TB SSD

---

4. Deployment Options

4.1 Option A – Managed Cloud Deployment

Prerequisites

- Aurora Cloud account
- Active API credentials
- Optional: VPC peering or private link

Setup Steps

1. Create an Aurora Cloud project.
2. Configure per-tenant rate limits.
3. Add service accounts (optional).
4. Upload custom tools (via Tool Registry API).
5. Configure embeddings or vector search integration.
6. Optionally whitelist outbound/inbound CIDRs.

Pros: Zero maintenance, auto-scaling, highest uptime.
Cons: No access to raw model weights.

4.2 Option B – Hybrid Deployment

In this model:

- Aurora inference runs **in your VPC**.
- Aurora security & safety enforcement runs **via Aurora Cloud Gateway**.

Use Case Examples

- GovCloud / FedRAMP environments
- Enterprises needing data residency guarantees
- Regulated industries

Prerequisites

- Kubernetes cluster (K8s 1.25+)
- NVIDIA GPU operator installed
- Fast interconnect between nodes
- Access to Aurora Hybrid Gateway

Components Installed Locally

- Aurora Inference Runtime
- Aurora KV Cache Server
- Aurora Chunk-Streaming Loader
- Vector Storage (Milvus, Pinecone, Weaviate, or internal)

Deployment Steps

Install Aurora Operator:

```
kubectl apply -f aurora-operator.yaml
```

1.

Deploy GPU node pool:

```
kubectl apply -f gpu-node-pool.yaml
```

2.

Deploy Aurora-72B inference shard set:

```
kubectl apply -f aurora72b-sharded.yaml
```

3.

Register your cluster with the Aurora Cloud Control Panel.

```
auroractl register --cluster-id your-cluster
```

4.

Validate inference health:

```
auroractl health --cluster your-cluster
```

5.

4.3 Option C – Full Self-Hosted Enterprise Deployment

This deploys:

- Model server
- Planner/agent component
- Safety interceptor
- Routing/gRPC gateway

- Rate-limit engine
- Observability stack

Recommended Orchestration Stacks

- **Kubernetes** (preferred)
- **Bare metal + Docker Compose** (small teams)
- **Nomad** or **OpenShift** (regulated govt environments)

Deployment Topology

```
Aurora Gateway (NGINX / Envoy)
        |
Inference Orchestrator (Go service)
        |
+-----------+-----------+
|           |           |
Model Shard 1  Model Shard 2  ...
(2–8 GPUs each)
```

---

5. Installation Procedures

5.1 Kubernetes Deployment

Step 1 – Install NVIDIA GPU Operator

```
kubectl create -f https://nvidia.github.io/gpu-operator.yaml
```

Step 2 – Download Aurora Helm Chart

```
helm repo add aurora https://repo.aurora.ai/charts
helm repo update
```

Step 3 – Install Aurora Runtime

```
helm install aurora-inference aurora/aurora-runtime \
  --set model=aurora-72b-agent-v3 \
  --set replicas=4 \
  --set gpu.type="a100" \
  --set gateway.enabled=true
```

Step 4 – Expose Gateway

```
kubectl apply -f ingress.yaml
```

Step 5 – Validate

```
curl https://<gateway>/v1/models \
  -H "Authorization: Bearer <token>"
```

5.2 Docker Compose Deployment (Local or Small Teams)

For Aurora-3B or Aurora-16B only.

docker-compose.yaml

```
version: "3.9"
services:
  aurora:
    image: aurora/aurora-runtime:latest
    environment:
      - MODEL=aurora-16b-assistant
    ports:
      - "8080:8080"
    deploy:
      resources:
        reservations:
          devices:
            - capabilities: ["gpu"]
```

Run

```
docker compose up -d
```

---

6. Scaling & Performance

6.1 Horizontal Scaling

Aurora supports:

- **Stateless gateway scaling** (unlimited)
- **Model-shard replication**
- **KV cache synchronization**

Scaling Recommendation

Model	Instances	Notes
72B	4–8	High traffic; use KV-L2 cache server
16B	2–4	Balanced workloads
3B	1–2	Low-latency inference

6.2 Performance Optimization

- Enable **tensor parallelism** & **pipeline parallelism**
- Use **flash-attention kernels**
- Enable **int8 or int4 quantization**
- Use **persistent KV cache**
- Pin pods to GPU nodes

---

7. Observability & Logging

Aurora ships with a full monitoring stack:

7.1 Metrics (Prometheus)

- `aurora_inference_latency_seconds`
- `aurora_generated_tokens_total`
- `aurora_gpu_utilization_percent`
- `aurora_safety_interventions_total`
- `aurora_tool_calls_total`

7.2 Logging (ELK or OpenSearch)

- Streaming logs for each request
- Tool call traces
- Safety filters applied

7.3 Distributed Tracing

- OpenTelemetry support
- Traces include:
    - Planner
    - Model inference
    - Tool call execution

    - Safety layer decisions

---

8. Security & Compliance

8.1 Authentication

- JWT or API-key based
- Mutual TLS for internal components

8.2 Data Handling

- No data stored by default unless explicitly enabled
- Optional encrypted conversation storage
- Optional redaction middleware

8.3 Isolation Options

- Namespace isolation (K8s)
- GPU partitioning (MIG)
- Dedicated per-customer clusters

---

9. Disaster Recovery & High Availability

9.1 Recommended Strategy

- Use multi-zone GPU clusters
- Enable Aurora state store replication (etcd or Aurora-KV)
- Keep vector DBs replicated

9.2 Backup Cadence

Component	Backup Frequency
Model weights	Static; no backup needed
Config & Tools	Daily
Logs	7–14 days retention
Vector DB	4–6 hours

---

10. Appendix

10.1 Example Health Check

```
curl https://my-aurora-internal/v1/health
```

Response:

```
{
  "status": "ok",
  "gpu": {
    "temperature": 68,
    "memory_used_gb": 53.2,
    "utilization": 89
  },
  "model": {
    "loaded": "aurora-72b-agent-v3",
    "ready": true
  }
}
```