Private LLM Architectures for Enterprise AI

Building secure, scalable language models within enterprise boundaries

45 min read | January 2026 | Architecture

KEY TAKEAWAYS

- Data sovereignty requires complete control over model infrastructure
- Quantisation enables larger models on available hardware without significant quality loss
- Multi-node tensor parallelism is essential for production deployments of 70B+ models
- Security architecture must address encryption at rest, in transit, and in memory

Building Secure, Scalable Language Models Within Enterprise Boundaries

Architetto Research | January 2025

Executive Summary

The adoption of Large Language Models (LLMs) in enterprise environments has accelerated dramatically, yet most organisations remain dependent on third-party API services that process sensitive data outside their security perimeter. This whitepaper presents a comprehensive framework for deploying private LLM infrastructure that maintains data sovereignty, ensures regulatory compliance, and delivers the performance characteristics required for production workloads.

We examine the architectural patterns, infrastructure requirements, security considerations, and operational practices necessary to build and maintain private LLM deployments. The guidance presented here is designed for technology leaders, security architects, and engineering teams evaluating or implementing private AI capabilities.

1. The Case for Private LLM Infrastructure

1.1 The Limitations of API-Based AI Services

© 2026 Architetto. All rights reserved. | architetto.io

Public LLM APIs have democratised access to powerful language models, but they introduce significant constraints for enterprise adoption:

Data Sovereignty Concerns

When data is transmitted to external APIs, organisations lose visibility into how that data is processed, stored, and potentially retained. For regulated industries—financial services, healthcare, legal, and government—this creates compliance risk that is difficult to mitigate through contractual provisions alone.

Unpredictable Costs at Scale

API pricing models based on token consumption can result in unpredictable costs as usage grows. Organisations processing millions of documents or handling high-volume customer interactions may find API costs exceeding the total cost of ownership for private infrastructure.

Latency and Availability Dependencies

Production applications requiring consistent sub-second response times are vulnerable to the latency variability and availability characteristics of shared infrastructure. Mission-critical systems cannot tolerate the outages that periodically affect public AI services.

Limited Customisation

While fine-tuning options exist for some API services, the depth of customisation available with private deployments—including specialised training data, custom tokenisers, and architecture modifications—is fundamentally greater.

1.2 When Private Deployment Makes Sense

Private LLM infrastructure is appropriate when organisations require:

- Processing of data classified as confidential, regulated, or containing personally identifiable information
- Predictable performance characteristics for production applications
- Deep integration with existing security, monitoring, and compliance infrastructure
- Specialised models trained on proprietary datasets
- Air-gapped or isolated network deployments
- Long-term cost optimisation at scale

2. Architectural Foundations

2.1 Reference Architecture Overview

A production-grade private LLM deployment comprises several interconnected layers:

Infrastructure Layer

The compute foundation, typically built on GPU-accelerated servers or specialised AI accelerators. This layer must provide sufficient memory bandwidth and compute capacity to serve models at the

© 2026 Architetto. All rights reserved. | architetto.io

required throughput and latency.

Model Serving Layer

The runtime environment responsible for loading models into memory, managing inference requests, and returning results. This layer handles batching, queuing, and resource allocation.

API Gateway Layer

The interface through which applications interact with the LLM infrastructure. This layer implements authentication, rate limiting, request routing, and protocol translation.

Orchestration Layer

For complex applications, an orchestration layer manages multi-step workflows, retrieval-augmented generation (RAG) pipelines, and agent-based interactions.

Observability Layer

Comprehensive monitoring, logging, and alerting infrastructure to ensure operational visibility and rapid incident response.

2.2 Model Selection Considerations

The choice of base model significantly impacts infrastructure requirements and capability. Key considerations include:

Parameter Count and Memory Requirements

Models range from 1 billion to over 400 billion parameters. Memory requirements scale accordingly:

Model Size	Minimum VRAM (FP16)	Recommended VRAM
7B	14 GB	24 GB
13B	26 GB	48 GB
70B	140 GB	160 GB
405B	810 GB	1 TB+

Quantisation Trade-offs

Quantisation reduces memory requirements by representing weights with lower precision. Common approaches include:

- **FP16**: Full precision, highest quality, maximum memory consumption
- **INT8**: 50% memory reduction with minimal quality loss for most tasks
- **INT4**: 75% memory reduction with moderate quality impact
- **GPTQ/AWQ**: Optimised 4-bit quantisation with quality preservation techniques

Architecture Considerations

© 2026 Architetto. All rights reserved. | architetto.io

Different model architectures offer distinct characteristics:

- **Dense Models**: Traditional transformer architectures with consistent compute requirements per token
- **Mixture of Experts (MoE)**: Sparse architectures that activate only a subset of parameters per token, offering better efficiency at very large scales

2.3 Deployment Patterns

Single-Node Deployment

Appropriate for smaller models (up to ~70B parameters with quantisation) or development environments. Simplest to manage but limited in scalability and redundancy.

Multi-Node Tensor Parallelism

Distributes model layers across multiple GPUs on a single node. Required when model memory requirements exceed single-GPU capacity.

Multi-Node Pipeline Parallelism

Distributes model stages across multiple servers connected via high-speed networking. Enables deployment of the largest models but introduces complexity in synchronisation and fault tolerance.

Hybrid Deployments

Combines tensor and pipeline parallelism across node clusters. Required for production deployments of models exceeding 70B parameters.

3. Infrastructure Requirements

3.1 Compute Hardware

GPU Selection

NVIDIA GPUs dominate the enterprise LLM deployment landscape:

| GPU Model | VRAM | Interconnect | Use Case |

|---|---|---|---|

| A100 (40GB) | 40 GB | NVLink | Production, multi-GPU |

| A100 (80GB) | 80 GB | NVLink | Large models, production |

| H100 (80GB) | 80 GB | NVLink 4.0 | High performance, latest generation |

| L40S | 48 GB | PCIe | Cost-optimised production |

| A10G | 24 GB | PCIe | Smaller models, development |

Interconnect Requirements

© 2026 Architetto. All rights reserved. | architetto.io

For multi-GPU deployments, high-bandwidth interconnects are essential:

- **NVLink**: 600-900 GB/s bidirectional bandwidth for intra-node communication
- **InfiniBand**: 200-400 Gb/s for inter-node communication in distributed deployments
- **Ethernet**: 100 Gb/s minimum for distributed deployments; introduces higher latency than InfiniBand

3.2 Storage Architecture

Model Storage

Models must be accessible with high throughput for loading operations:

- NVMe SSDs for local model caching
- High-performance network storage (NFS, Lustre, or object storage with caching) for model repositories
- Capacity planning should account for multiple model versions and checkpoints

Vector Storage for RAG

Retrieval-augmented generation requires efficient vector similarity search:

- Purpose-built vector databases (Milvus, Qdrant, Weaviate) for large-scale deployments
- PostgreSQL with pgvector extension for simpler requirements
- Index sizing: typically 1-4 bytes per dimension per vector, plus metadata overhead

3.3 Network Architecture

Isolation Requirements

LLM infrastructure should be isolated from general-purpose networks:

- Dedicated VLANs or network segments for AI workloads
- Firewall rules restricting access to authorised applications and administrators
- No direct internet access; all external communication via controlled proxies

Bandwidth Considerations

- Client-to-API traffic: typically modest; LLM requests and responses are text-based
- Inter-node traffic: substantial for distributed inference; requires dedicated high-bandwidth paths
- Storage traffic: high during model loading; can be optimised with local caching

4. Security Architecture

4.1 Data Protection

Encryption at Rest

© 2026 Architetto. All rights reserved. | architetto.io

All persistent data—including models, training data, logs, and vector databases—must be encrypted:

- AES-256 encryption for storage volumes
- Key management via hardware security modules (HSMs) or enterprise key management systems
- Separate key hierarchies for different data classifications

Encryption in Transit

All network communication must be encrypted:

- TLS 1.3 for API traffic
- IPsec or WireGuard for inter-node communication
- Certificate management integrated with enterprise PKI

Memory Protection

Sensitive data may reside in GPU and system memory during inference:

- Secure boot and attestation for compute nodes
- Memory encryption where hardware supports it (AMD SEV, Intel TDX)
- Prompt clearing of sensitive context after request completion

4.2 Access Control

Authentication

- Integration with enterprise identity providers (SAML, OIDC)
- Service accounts with short-lived credentials for application access
- Multi-factor authentication for administrative access

Authorisation

- Role-based access control (RBAC) for model and endpoint access
- Fine-grained permissions for different model capabilities
- Audit logging of all access decisions

4.3 Input and Output Controls

Prompt Injection Mitigation

LLMs are vulnerable to prompt injection attacks that attempt to override system instructions:

- Input validation and sanitisation before processing
- Separation of system prompts from user input in API design
- Output filtering for sensitive patterns
- Rate limiting to prevent abuse

Content Filtering

- Configurable content policies for input and output

© 2026 Architetto. All rights reserved. | architetto.io

- Integration with enterprise data loss prevention (DLP) systems
- Logging of filtered content for security review

5. Operational Excellence

5.1 Model Lifecycle Management

Version Control

- All model artefacts stored in version-controlled repositories
- Clear naming conventions and metadata for model versions
- Rollback capabilities for rapid recovery from problematic deployments

Deployment Automation

- Infrastructure as code for repeatable deployments
- Automated testing pipelines for model validation
- Blue-green or canary deployment strategies for production updates

Capacity Planning

- Baseline throughput testing for all deployed models
- Load testing to establish scaling characteristics
- Proactive capacity additions based on usage trends

5.2 Monitoring and Observability

Performance Metrics

- Request latency (p50, p95, p99)
- Throughput (requests per second, tokens per second)
- Queue depth and wait times
- GPU utilisation and memory consumption

Quality Metrics

- Response quality sampling and human evaluation
- Automated quality checks for common failure modes
- User feedback integration

Security Metrics

- Authentication failure rates
- Blocked request patterns
- Anomaly detection for usage patterns

5.3 Cost Management

© 2026 Architetto. All rights reserved. | architetto.io

Resource Optimisation

- Right-sizing GPU allocations based on actual usage
- Scheduled scaling for predictable workload patterns
- Spot or preemptible instances for development and testing

Usage Tracking

- Per-application and per-user usage attribution
- Chargeback or showback reporting for internal cost allocation
- Trend analysis for budget forecasting

6. Integration Patterns

6.1 Application Integration

Synchronous API

The simplest integration pattern; applications make blocking requests to the LLM API and wait for responses. Appropriate for:

- Interactive applications with modest latency requirements
- Single-turn queries without complex orchestration
- Applications where simplicity is prioritised over throughput

Asynchronous API

Applications submit requests and receive responses via callbacks or polling. Appropriate for:

- Batch processing workloads
- Long-running generation tasks
- High-throughput applications where response ordering is not critical

Streaming API

Responses are delivered incrementally as they are generated. Appropriate for:

- Chat interfaces where users expect immediate feedback
- Long-form generation where complete responses may take several seconds
- Applications that can process partial results

6.2 Retrieval-Augmented Generation

RAG architectures combine LLM capabilities with retrieval from enterprise knowledge bases:

Document Ingestion Pipeline

- Document parsing and chunking
- Embedding generation using specialised models

© 2026 Architetto. All rights reserved. | architetto.io

- Vector storage and indexing

Query Pipeline

- Query embedding generation
- Similarity search against vector database
- Context assembly and prompt construction
- LLM inference with retrieved context

Considerations

- Chunk size affects retrieval precision and context efficiency
- Embedding model selection impacts retrieval quality
- Reranking can improve precision at the cost of latency

6.3 Agentic Architectures

Complex applications may require LLMs to operate as agents that plan, execute actions, and iterate:

Tool Integration

- Well-defined tool interfaces with input/output schemas
- Sandboxed execution environments for code generation
- Authentication and authorisation for tool access

Workflow Orchestration

- State management across multi-turn interactions
- Error handling and recovery strategies
- Human-in-the-loop checkpoints for sensitive operations

7. Compliance and Governance

7.1 Regulatory Considerations

Data Protection Regulations

- GDPR, CCPA, and similar regulations apply to personal data processed by LLMs
- Privacy impact assessments should address LLM processing
- Data subject rights (access, deletion, portability) must be supported

Industry-Specific Regulations

- Financial services: SEC, FCA, and other regulators increasingly scrutinise AI usage
- Healthcare: HIPAA requirements for protected health information
- Government: FedRAMP and similar frameworks for cloud services

7.2 Model Governance

© 2026 Architetto. All rights reserved. | architetto.io

Model Cards and Documentation

- Comprehensive documentation of model capabilities and limitations
- Training data provenance and composition
- Known biases and failure modes

Risk Assessment

- Classification of use cases by risk level
- Appropriate controls for high-risk applications
- Regular review and reassessment

Audit Trail

- Logging of all inference requests with sufficient context for review
- Retention policies aligned with regulatory requirements
- Secure storage of audit logs

8. Conclusion

Private LLM deployment represents a significant undertaking, but it is increasingly essential for organisations with stringent security, compliance, or performance requirements. The architectural patterns, infrastructure considerations, and operational practices described in this whitepaper provide a foundation for successful implementation.

Success depends on:

- Realistic assessment of requirements and constraints
- Appropriate model selection for the target use cases
- Robust security architecture addressing data protection, access control, and LLM-specific vulnerabilities
- Operational maturity in monitoring, lifecycle management, and capacity planning
- Clear governance frameworks aligned with regulatory requirements

Organisations that invest in private LLM capabilities position themselves to leverage AI capabilities that would otherwise be unavailable due to data sensitivity constraints, while maintaining the control and visibility essential for enterprise operations.

References

1. NVIDIA Technical Documentation: GPU Architecture and Performance

2. Hugging Face Model Hub: Model Cards and Documentation

3. MLCommons: ML Systems Benchmark Standards

© 2026 Architetto. All rights reserved. | architetto.io

4. NIST AI Risk Management Framework

5. European AI Act: High-Risk AI System Requirements

This whitepaper is provided for informational purposes. Specific implementation decisions should be made in consultation with qualified technical and legal advisors.

© 2026 Architetto. All rights reserved. | architetto.io