Document covering deployment strategy for serving a fine-tuned GPT-2 model (with LoRA/Adapter modifications) to handle approximately 500 concurrent requests.

Target Specifications

- Concurrent Users: ~500 simultaneous requests
- Availability: 99.9% uptime target
- Scalability: Auto-scaling based on demand

Proposed Architecture

1. Containerization (Docker)

- This will ensure consistent deployment environments across development, staging, and production.

We will package the entire model application, including dependencies, model weights, and runtime environment into Docker containers. This will simplify version management and rollbacks.

2. Container Orchestration (Kubernetes)

- The purpose is to manage multiple container instances with automatic scaling and load balancing.

Deploy containers as Kubernetes pods across multiple nodes.
Because Kubernetes provides built-in load balancing, health monitoring, automatic restarts on failures, and horizontal pod autoscaling based on CPU/GPU metrics. This will ensure the system can handle traffic fluctuations automatically.

3. Request Management with Message Queues

- We will implement Apache Kafka to decouple request ingestion from model inference processing.

Flow:
Client Request --> API Gateway --> Kafka Queue --> Model Pods --> Response Cache --> Client

This will prevent system overload during traffic spikes and enable asynchronous processing, and provide request durability.
And will act as a buffer between incoming requests and available processing capacity

4. Dynamic Batching Implementation

Instead of processing requests individually, collect multiple requests arriving within a short time window (e.g., 50ms) and process them as a batch. This will improve GPU throughput while maintaining acceptable latency.

5. Redis Caching Layer

To reduce redundant computations for frequently requested prompts.

Strategy -

- Cache responses using hashed prompt + model configuration as keys
- Implement 1-hour TTL to balance freshness and efficiency
- Expected 30-40% cache hit rate for typical workloads
- Significantly reduces computational load and improves response times

6. Infrastructure Requirements

Hardware Specifications:

- GPU Nodes: 3-5 nodes with NVIDIA T4/A10 GPUs
- Memory: 32GB RAM minimum per node
- Storage: SSD storage for fast model loading (100GB)
- Network: High-bandwidth (10Gbps) for inter-node communication

Cloud Provider:

- AWS: EKS with g4/p3 instance types

7. API Service Layer

FastAPI-based service providing RESTful endpoints for model inference.
Features:

- Asynchronous request handling
- Integration with caching and batching layers
- Request validation and error handling
- Health check endpoints for monitoring

8. Monitoring and Observability

Metrics Collection: Prometheus for gathering system and application metrics.
Visualization: Grafana dashboards for real-time monitoring.
Key Metrics:

- Request count and rate
- Response latency (P50, P95, P99)
- GPU utilization and memory usage
- Cache hit rates
- Queue depths and processing times

Performance Estimates

Expected Throughput

- Single Pod: ~100 requests/minute with dynamic batching
- 5-Pod Cluster: ~500 requests/minute capacity
- With Caching: Effective capacity of 700+ requests/minute (40% cache hit rate)

Resource Utilization

- GPU Usage: 70-85% average utilization
- Memory: ~12GB per pod average
- Network: ~200Mbps per pod