Cost Evaluation and Resource Management in Azure ML Deployment

This report evaluates the cloud infrastructure choices, deployment plan, and cost management practices implemented for a sentence-level emotion classification system developed as part of a Natural Language Processing project. The solution processes text, audio, and video transcriptions to infer emotional tone, using a transformer-based model fine-tuned and deployed on Microsoft Azure. To ensure performance and affordability, the project was designed with modular architecture, local-first development, and dynamic scaling in the cloud. This document highlights the technical decisions made, with a primary focus on assessing cost–performance trade-offs and outlining our resource-efficient deployment approach using Azure Kubernetes Service (AKS).

Cloud Architecture and Cost-Aware Design Decisions

We initially deployed our model using Azure's ManagedOnlineEndpoint to validate the deployment logic and quickly test the scoring script. However, we realized this approach introduced higher default compute costs and limited our flexibility in scaling (Microsoft, 2023a). Given our need for GPU acceleration and scalable deployment, we switched to using Azure Kubernetes Service (AKS), which allowed for more control over compute resources, runtime scaling and long-term cost management (Microsoft, 2023b). The earlier endpoints were removed and we transitioned fully to AKS using the cloud_deploy.py and deploy_kubernetes_endpoint.py scripts. These scripts reused registered assets such as models, environments and scoring scripts, helping us avoid redundant deployments.

Resource Optimization Strategies

- Local-first development was prioritized for model loading, user interface testing (app_ui.py), and CLI integration, minimizing reliance on repeated cloud-based trials.
- The model and tokenizer were loaded once during initialization via model_loader.py and reused to prevent redundant memory and compute usage.
- Logging was implemented in score.py at the INFO level enabling debugging without the added cost of real-time monitoring tools (Microsoft, 2023c).
- The inference pipeline was designed to be stateless, optimizing memory usage and allowing deployment on lower-tier AKS node sizes.
- Rather than running endpoints continuously during development, they were deployed only as needed, significantly reducing overall compute time and operational costs.

Monitoring and Budget Awareness

While we did not yet integrate Azure Monitor or enable cost alerts in this testing phase, our system was designed to support these features. In future iterations, we plan to configure autoscaling thresholds, timed shutdown policies for low-demand periods and Azure Budget alerts on the AKS node pool to maintain an effective balance between performance and cost (Microsoft, 2023d). Future improvements will include setting cost alerts at 80% and 100% thresholds and reviewing actual vs. forecasted costs using Azure Cost Management. The logging built into score.py and API_main.py already provides a lightweight yet effective foundation for diagnostics and monitoring during testing phases.

Compute Cost Breakdown and Deployment Efficiency

For model deployment and scalable inference, our emotion classification model was hosted on Azure Kubernetes Service (AKS) using a node with 16 vCPUs, 32 GiB RAM and 1 GPU, comparable to the NC T4 v3 VM priced at $0.815/hour when running Windows-based GPU compute (Microsoft, 2024e). This configuration ensured a balance between performance and cost, supporting scalable inference for our RoBERTa-based classifier. Training was conducted locally on an NVIDIA L40S GPU with 48 GB VRAM; if replicated on Azure, each 30–40-minute session would cost around $0.45–$0.60, with total fine-tuning costs staying under $3 for the project. GPU-based compute was selected for both training and inference due to the transformer model's high parallelization demands, which would be inefficient on CPU. CPU-only nodes were considered for API and UI components, where GPU acceleration is not required. For deployment, running the endpoint for 8 hours per weekday results in an estimated cost of $7.20/day or about $145/month over 20 working days. Given an average response time of 0.2 seconds, the system can handle 1,000–1,500 requests per hour. Assuming 100 users each make 5–10 requests per day, monthly usage would range from 25,000 to 30,000 inferences, translating to an additional inference cost of roughly $20–$40 per month. This offers a sustainable and cost-effective setup for small-scale real-world use.

Table 1. Estimated Monthly Cloud Costs for Emotion Classification Pipeline

To complement the compute estimate, we used the Azure Pricing Calculator to account for all relevant cloud services supporting our deployment. The table below summarizes the projected monthly costs:

Service	Usage Assumption	Monthly Cost ($)
AKS GPU Node (NCas_T4_v3)	1 node x 8hrs/day x 20 days ($0.815/hr)	$145.0
Inference Cost	25,000-30,000 inferences/month (GPU usage + throughput load)	$20-$40
Model Fine-Tuning (Local L40S)	4-5 session x 30-40 min each ($0.50/session est.)	<$3.0
Blob Storage	50 GB for model, logs, transcription	$0.92
Azure Monitor (Logging)	5GB free + 5 GB ($2.3/GB)	$11.5
Container Registry	5GB (Standard Tier it fits in free limit)	$0.00
Data Egress	<1GB (fits in free limit)	$0.00
Total Estimated Monthly Cost (Single Node Deployment)		$180.42 - $200.42

Conclusion

Our deployment of the emotion classification pipeline to Azure Kubernetes Service demonstrates that smart infrastructure decisions can significantly reduce cloud computing costs without compromising performance. By training the model locally, reusing registered assets and leveraging GPU instances only, when necessary, we maintained cost control while meeting all functional requirements. The system is scalable, resource-aware and technically robust, providing a solid foundation for future enhancements such as autoscaling, budget monitoring, or commercial deployment. The result is a deployment pipeline that balances real-world usage requirements with sustainable financial overhead—key criteria for scalable, production-grade ML services.

References

Microsoft. (2023a). *Managed online endpoints*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/machine-learning/how-to-deploy-managed-online-endpoints

Microsoft. (2023b). *What is Azure Kubernetes Service (AKS)?*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/aks/intro-kubernetes

Microsoft. (2023c). *Use logging in Azure Machine Learning*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/machine-learning/how-to-enable-logging

Microsoft. (2023d). *Set budgets and monitor costs in Azure*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/cost-management-billing/costs/cost-mgt-best-practices

Microsoft. (2024e). *Azure Pricing Calculator*. Microsoft Azure. https://azure.microsoft.com/en-us/pricing/calculator/?cdn=disable

OpenAI. (2024). *ChatGPT* (June 2024 version) [OpenAI's ChatGPT (June 2024 version) to assist with spelling corrections, grammar editing, and rewording of some content for clarity and formality]. https://chat.openai.com/