**SOF/C Tactical OPE Toolkit: Maintenance and Monitoring Plan**

# **Introduction**

The SOF/C Operation Planning of the Environment (OPE) application is deployed within Microsoft Azure Cloud. It leverages the Azure OpenAI Service to host both the Visual Language Model (VLM) and Large Language Model (LLM), enabling advanced AI-driven analytics. The application aggregates geospatial data via the Google Maps API and applies AI-managed solutions to perform analysis on the collected data.

The application is available in two versions to meet the customer’s operational requirements: a cloud-based deployment within Microsoft Azure and a standalone version. The solution is containerized, comprising separate frontend and backend components to support modularity and scalability across deployment environments.

**Model Monitoring**

## **Data Monitoring**

The application has undergone comprehensive testing with a variety of input scenarios, including invalid coordinates, incorrect API keys, and nonexistent locations. As a result, the application is designed to return clear and informative error messages when input criteria are not met.

The models powering the LLM and VLM are configured to return appropriate errors if the requested data is unavailable or does not match known parameters. Furthermore, any requests involving sensitive or inappropriate content are filtered, and such information is automatically excluded from the response.

## **Model Performance Monitoring**

As the application remains in use, existing AI models will inevitably be retired and replaced with newer, more efficient versions. When this occurs, AI2C or the designated organization maintaining the application code will be responsible for implementing the necessary updates. This process will involve updating the remote repository and executing the associated deployment workflow.

For standalone deployments, additional steps such as remote scanning of user workstations may be required to verify that the latest updates have been properly applied and are functioning as intended.

## **Monitoring Infrastructure**

The infrastructure is deployed on Microsoft Azure Kubernetes Service (AKS), utilizing two initial pods within the cluster. To evaluate performance under load, the application was vertically scaled using a locust.py script and the Apache JMeter application, simulating 10 concurrent users executing queries. Based on the results of this testing, an assessment was conducted to determine the number of pods required to support this user load efficiently, leading to the implementation of horizontal pod autoscaling.

Following the scaling implementation, it was determined that monitoring capabilities were necessary to ensure the application performs as expected within the cluster. For the cloud deployment, Azure’s managed Prometheus solution was selected to collect time-series metrics and generate performance alerts. Additionally, the managed Grafana service will be utilized to provide dashboards and visualizations based on the data collected by Prometheus.

To monitor the standalone application, users can utilize Docker Desktop to inspect the status of running containers. By accessing the container view and selecting the three-dot (ellipsis) menu, the **"View Details"** option provides access to logs, resource usage statistics, and image inspection. This interface enables users to assess container health, identify potential resource overcommitment, and troubleshoot issues within the standard deployment environment.

# **Model Maintenance**

## **Retraining and Updates**

Since the AI model will utilize a managed service, retraining or manual updates will not be necessary. The only requirement is to ensure that the model remains within its support lifecycle and is not approaching retirement. Responsibility for monitoring this will fall to either AI2C or the organization deploying the application.

## **Version Control**

The application code is hosted in a GitHub repository that is integrated with the Managed Identity of the AKS cluster. When updates are pushed from the frontend or backend repositories to the main branch, an automated workflow pipeline is triggered. This pipeline builds the Docker image, uploads it to the Azure Container Registry, and deploys it to the AKS cluster. This process ensures that the cluster consistently runs the latest version of the application.

Organizational policy will dictate when changes are promoted to the main repository. Prior to deployment, the frontend and backend container images will be tested locally using Docker Desktop to verify compatibility and ensure stable operation.

# **Operational Considerations**

## **Resource Management**

Resources will be monitored using Azure Monitor and the Managed Prometheus and Grafana service. This will ensure that the resources have not been overcommitted as the user base increases. However, an accurate assessment of the resources must be made monthly to ensure that the resources are handling the increased usage and are scaled appropriately.

To monitor the LLM and VLM services, Azure AI Studio can be utilized to track Azure OpenAI resource usage and performance. Additionally, the Google Maps API Dashboard provides visibility into API key management and usage metrics, supporting effective monitoring and maintenance.

## **Security and Compliance**

As the model transitions into production, the application will need to be hardened in accordance with the customer’s network security requirements. This process will include a comprehensive code review of both the frontend and backend containers to ensure alignment with the security standards established by the customer’s security team.

## **Incident Management**

As the application transitions to the customer’s network, we will establish an Incident Response strategy for both the cloud and standalone versions.

**Standalone:**

The standalone model will consist of the previously mentioned version control system to ensure an up-to-date and functioning application is deployed to a remote repository. Additionally, within this repository will be a README file that will contain point of contact information to allow users to address issues with designers.

**Cloud:**

The cloud version will still be linked to the remote repository and be controlled with a workflow to ensure the latest version has been loaded into the AKS Cluster. Additionally, the cloud resources will be monitored by the customer’s cloud administrators. The cloud administrators will ensure that the resource is functional and is being monitored per their policy.

# **Documentation and Knowledge Transfer**

The remote repository includes a README file that provides detailed information on updating, loading, and managing the current application. The customer intends to establish a centralized remote repository to store both the application code and associated documentation. This approach will support continuity and maintainability as the application progresses toward production deployment.

# **Conclusion**

As the application approaches deployment in the production environment, increased collaboration between the design team and the customer’s network team will be essential. This joint effort will ensure a smooth transition and ongoing maintenance of the application. This is particularly critical in areas such as application security and the development of an incident response plan.