# DEMO: Adaptive Application and System Management Across the Mobile-Edge-Cloud Continuum: A Smart Agriculture Use Case

Author Names Removed due to Double-Blind Submission Scheme
*Affiliation X*
*e-mail@xxx.yyy*

***Abstract*—Modern distributed applications are characterized by high complexity as well as strict quality of service requirements. Such applications consist of several components that communicate with each other to perform several tasks and often exploit edge computing to achieve improved performance compared to communicating with the cloud. Numerous real-world edge computing scenarios require operation across heterogeneous nodes, with the need to satisfy both application- and system-level constraints while supporting node mobility. In this demo, we introduce a framework for dynamic and adaptive management of distributed applications and systems spanning the mobile-edge-cloud continuum. To demonstrate the flexibility of our framework, we showcase a smart agriculture scenario including a tractor and a drone as mobile nodes to support efficient characterization and treatment of field properties.**

## 1. Introduction

Distributed applications have gained widespread adoption due to their ability to handle complex and resource-intensive tasks across different environments. It is common for such applications to comprise of several microservices, thus significantly affecting the operation of cloud data centers by accelerating scalability and increasing cost-efficiency. An increasing number of modern workloads are container-based, allowing effortless deployment and migration. Furthermore, a plethora of applications are no longer restricted to the cloud but also involve mobile IoT devices, such as smartphones and vehicles like cars or even drones. In this case, edge computing can improve application Quality of Service (QoS) by moving computations closer to the points where data is produced while leading to better overall system performance and stability, as it reduces data traffic and resource pressure to the cloud. To effectively harness the potential of edge computing, it is important to support flexible and adaptive deployment and orchestration of such applications so that edge resource usage adapts to the dynamically changing position of the mobile nodes.

In this demo, we showcase a framework that enables flexible application deployment and orchestration across the entire system continuum. We support modular applications where the developer merely provides the individual components as containers and annotates them with their resource, deployment, and interaction requirements. Based on this information, our framework deploys the application components on a suitable cluster, comprising cloud, edge, and mobile IoT nodes, and adapts this deployment at runtime without any intervention from the application owner or system administrator.

## 2. Design & Implementation

### 2.1. Conceptual Architecture

Our framework comprises a hierarchical multi-agent system which interacts with two other basic subsystems: Container Orchestration and Telemetry. The agent hierarchy follows the orchestration architecture, which is logically organized into the Continuum, Cluster, and Node layers. Communication between the three subsystems (agents, container orchestration, and telemetry) is performed through appropriate interfaces at every layer. All three subsystems have internal communication across different levels, while the agent communicates with the system Actors at the continuum-level. The node agent employs a controller to execute the commands for low-level configuration knobs. Our framework accomplishes the management by using the available telemetry data that is emitted from every layer to get the necessary insights. This design enables the usage of different approaches (potentially ML-based) to augment the framework's decision-making process to achieve the goals of the application and the system. The overall status and results of the decisions are communicated to system Actors.

### 2.2. Implementation

For the continuum- and cluster-level orchestration, we leverage Karmada [1] and Kubernetes [2] respectively. In addition, we use OpenTelemetry [3] as the telemetry collection and transmission mechanism across the different layers of the continuum. The multi-agent system is implemented with the SPADE framework [4], capturing continuum, cluster, and node agents. System information and application status are stored and updated using Redis [5] data structures. Moreover, we provide a CLI allowing users to deploy, remove, and monitor applications.

## 3. Smart Agriculture Use Case

### 3.1. Application description

The application focuses on the characterization and localization of field properties and the targeted application of products, using a system that consists of a device mounted on a tractor and a drone overflying the field. The key objective of the application is to achieve very high accuracy regarding the characterization of field properties, both in terms of quantification and localization. This information is then used to control, in real-time, a product application device on the tractor, to release the right amount of product on top of relevant positions, as the tractor moves above the area. Increased accuracy leads to operation efficacy, reduced usage of products, and cost savings for the farmer.

Both the tractor-mounted device and the drone are equipped with cameras and GPS. The device on the tractor features additional sensors. The data provided by these sensors, including the video/images from the cameras, are processed on board to detect and localize field properties in the parts of the field in front of the tractor. The drone plays the role of an optional, assistive subsystem, used to increase the accuracy of field characterization and localization. It can be used to produce such information before the tractor starts moving in the field or operate in tandem with the tractor to generate live information at runtime. The software component pipelines on the two nodes work synergistically: the drone sends properties of the scanned areas to the localization module of the on-tractor system.

### 3.2. Instantiation of our Framework

Figure 1 depicts our framework's instantiation for the use case to be demonstrated. The system infrastructure consists of four nodes, represented as Virtual Machines (VMs), connected through a VPN. Two nodes host the Continuum and the Cluster Agents respectively, operating on top of Karmada and Kubernetes based on their management layer. The two worker nodes implement the tractor and the drone respectively, with simulated mobility. To be able to perform effortless testing and validation of our framework before deploying it on the field, we simulate node mobility in a lab environment, having (virtual) mobile nodes that run the Software-In-The-Loop (SITL) configuration of the Ardupilot autopilot [6]. The system and application software on the mobile nodes, in our case a drone and a tractor, communicates with the respective autopilot using the MAVLink protocol [7] via the MAVProxy library [8] and the DroneKit software [9]. Finally, we use Mission Planner [10] and a Grafana dashboard [11] for the visualization of telemetry data related to node mobility and application performance respectively.

We use an application that mimics the behavior of the smart agriculture use case application and consists of three components: (i) the field analysis component for the tractor, (ii) the field analysis component for the drone, and (ii) the drone driver/controller. For practical reasons, the drone

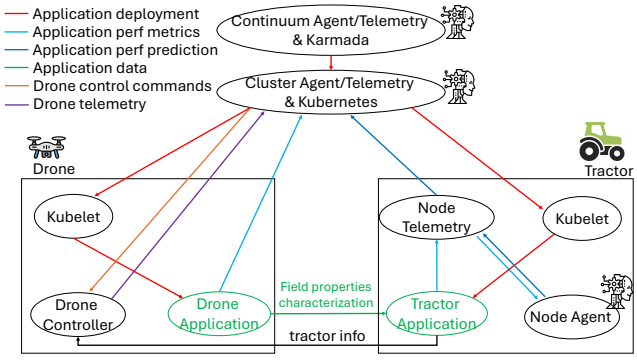


Figure 1. Framework instantiation for the smart agriculture use case.

controller is pre-installed on the drone node. Thus, the application components that are managed by our framework are the two field analysis components. The tractor component receives the data generated by the drone component, and considers this extra information to improve field analysis. It also sends tractor position/speed information to the drone controller. Both field analysis components emit application performance metrics via telemetry. The drone controller component emits state information via telemetry, based on whether the drone is ready to operate, is following the tractor or is returning to land (when explicitly disengaged or running out of battery).

In this demo scenario, our framework is responsible for monitoring the status of the worker nodes as well as application-related metrics via telemetry and for adapting the orchestration using two basic (re)configuration actions: (i) start/stop the drone operation based on the time windows which are estimated to be beneficial for the application without wasting drone's energy (batteries) using a simple heuristic approach, and (ii) deploy/remove the drone component when the drone is following the tractor/returning to its home location, respectively. Notably, the heuristic approach could be replaced by alternative (potentially machine learning-based) methods to predict the application performance and adapt accordingly.

## References


[1] "Karmada," https://karmada.io/.

[2] "Kubernetes," https://kubernetes.io/.

[3] "OpenTelemetry," https://opentelemetry.io/.

[4] "SPADE," https://spade-mas.readthedocs.io/en/latest/readme.html.

[5] "Redis," https://redis.io/.

[6] "Ardupilot SITL," http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html.

[7] "MAVLink," https://mavlink.io/en/.

[8] "MAVProxy," https://ardupilot.org/mavproxy/.

[9] "Dronekit," http://dronekit.io/.

[10] "Mission Planner," https://ardupilot.org/planner/.

[11] "Grafana," https://grafana.com/.