## Lab 7: Deploy a Flask API Service on Kubernetes

## Microservices and Cloud Computing - CSC 5201 301

### Hudson Arney

### 7 October 2024

### Use Case: **Real-Time Sports Event Data Ingestion and Analytics**

I wanted to focus on a project for sports analytics and I think this would be a great use case for Kubernetes.

### Scenario:

To start in baseball, data is captured using a combination of advanced sensors, radar systems, and high-speed cameras installed throughout the stadium. This is done through Statcast which tracks many detailed metrics in real time such as player speed, bat speed, throw velocity, launch angle, and exit velocity, while other systems monitor player positioning, ball movement, and matchups. I believe you could ingest this real time data into the Kubernetes app through various APIs.

Using Kubernetes could ensure the platform can scale dynamically to handle the significant data generated during games, especially in critical moments. Kubernetes manages microservices that process the ingested data, allowing for autoscaling, fault tolerance, and high availability. I think where it could become really powerful is with predicting the outcomes of different strategies, pitch choices, defensive setups, certain lineups, substitutions, etc. For instance, when a new game begins, we can scale-up automatically to quickly create multiple instances of our model (running on separate nodes) training microservices to evaluate different strategies—such as predicting the effectiveness of specific pitch types, defensive setups, or player lineups. And just as fast, scale back down for times where key decisions are not as necessary.

### Key Challenges Solved by Kubernetes:

1. **Scalability:** The system needs to handle peak loads during high-demand games or tournaments. Kubernetes can automatically scale up to manage the burst of data.
2. **Real-Time Data Processing:** The system ingests live data streams, so Kubernetes can orchestrate microservices that process this data and ensure near-instantaneous analysis.
3. **Fault Tolerance:** Kubernetes ensures that data processing continues smoothly by rescheduling tasks on other nodes in case of a node or pod failure, critical for uninterrupted live analytics.
4. **Cost Efficiency:** With auto-scaling, the system only consumes resources when needed, which is especially important when handling fluctuating data streams from games.

### Workflow:

1. **Data Ingestion Microservice:** Kubernetes runs pods that consume real-time sports data from sources like APIs, sensors, or IoT devices in the stadium (e.g., ball tracking or player movement sensors).
2. **Data Processing and Aggregation Service:** These microservices process raw data into meaningful insights—calculating player performance metrics (e.g., speed, accuracy) and game statistics (e.g., score, possession time). Kubernetes automatically scales this service based on incoming data volume. Most importantly, comparing to other possible situations to pick the best.
3. **Analytics and Prediction Microservice:** Machine learning models deployed on Kubernetes analyze data and provide real-time predictions, such as win probability.
4. **Data Storage:** Use Persistent Volumes for storing processed data and player statistics history.
5. **API Service for Consumers:** End users (could be the actual team or a sports gambler) can query the analytics via a REST API. Kubernetes manages the load by distributing the API requests to different pods.

### Key Kubernetes Components:

* **Horizontal Pod Autoscaler (HPA):** Automatically scales the pods based on CPU or memory usage to handle fluctuating traffic.
* **StatefulSets for Data:** Kubernetes StatefulSets ensure stable, unique storage for each instance of data processing or database pods, ensuring persistent access to historical data.
* **GPU Nodes:** If the analytical models involve deep learning for predictions, Kubernetes can allocate GPU resources dynamically for training and running these models in real time.

Submit a snapshot of your web page showing the Hello World

A computer screen with a white background

Description automatically generated

Snapshot for the results of adding a user

A screenshot of a computer

Description automatically generated

**Challenges/Feedback**: Maybe I’m just getting used to the formula, but this was the easiest lab so far for me. The only problem I had at all came from installing minikube and setting the PATH. Once I found the file location for that it was smooth sailing.   
  
Again, I appreciate you going into detail on each step of why everything has a meaning. I was wondering why this lab seemed more tame than previous, but then I saw lab 8’s title which looks like an expansion.  
  
  
**New Comment:** I apologize for missing the use case initially. I found that really fun! I love when we can use a little bit of freedom to apply knowledge to a subject we choose. I am considering making something like this for my final project!