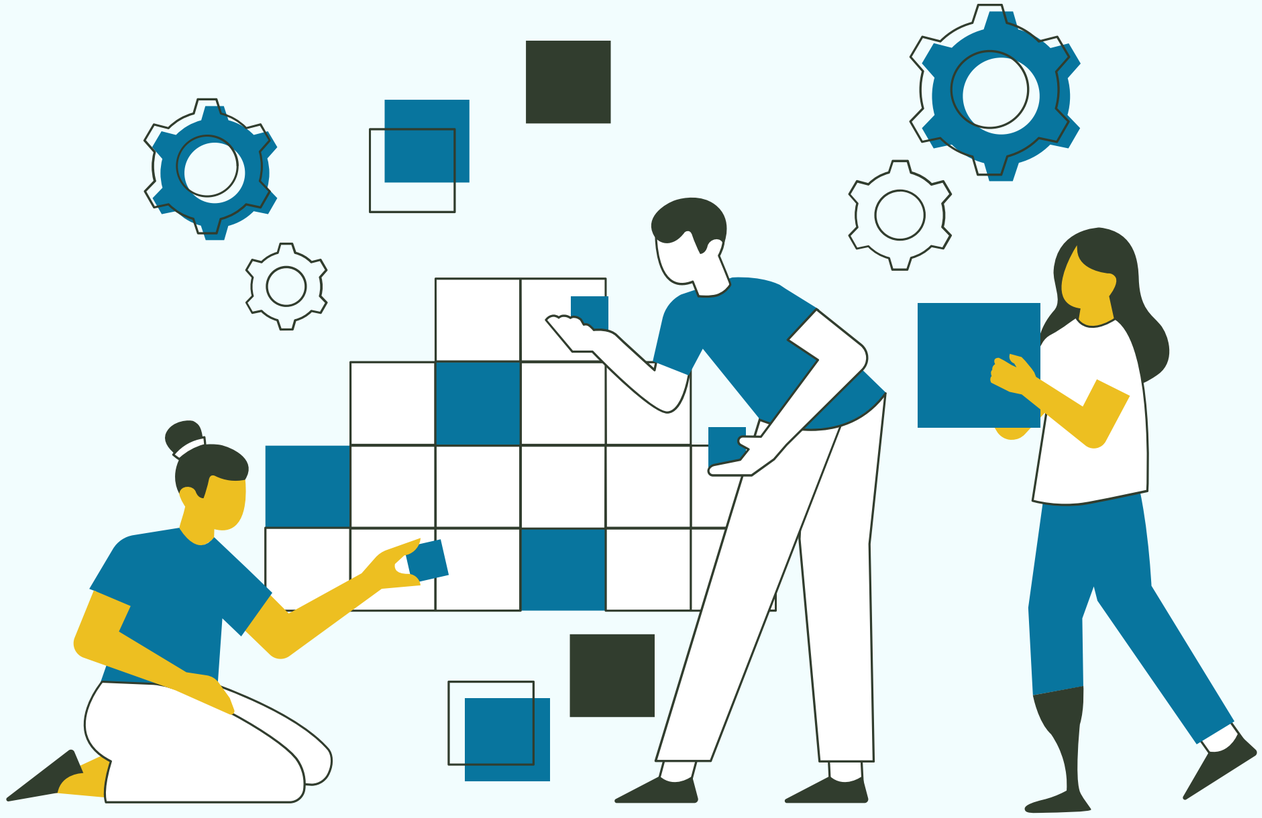
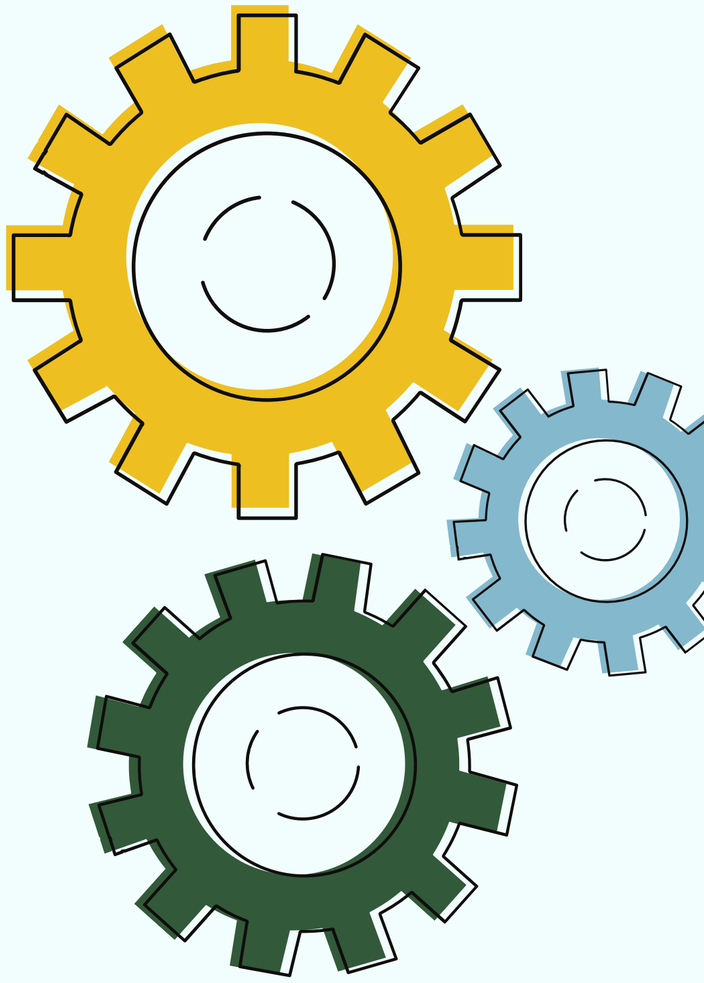
# DEVTrails

## University Hackathon 2025



# TEAM ClusterBusters

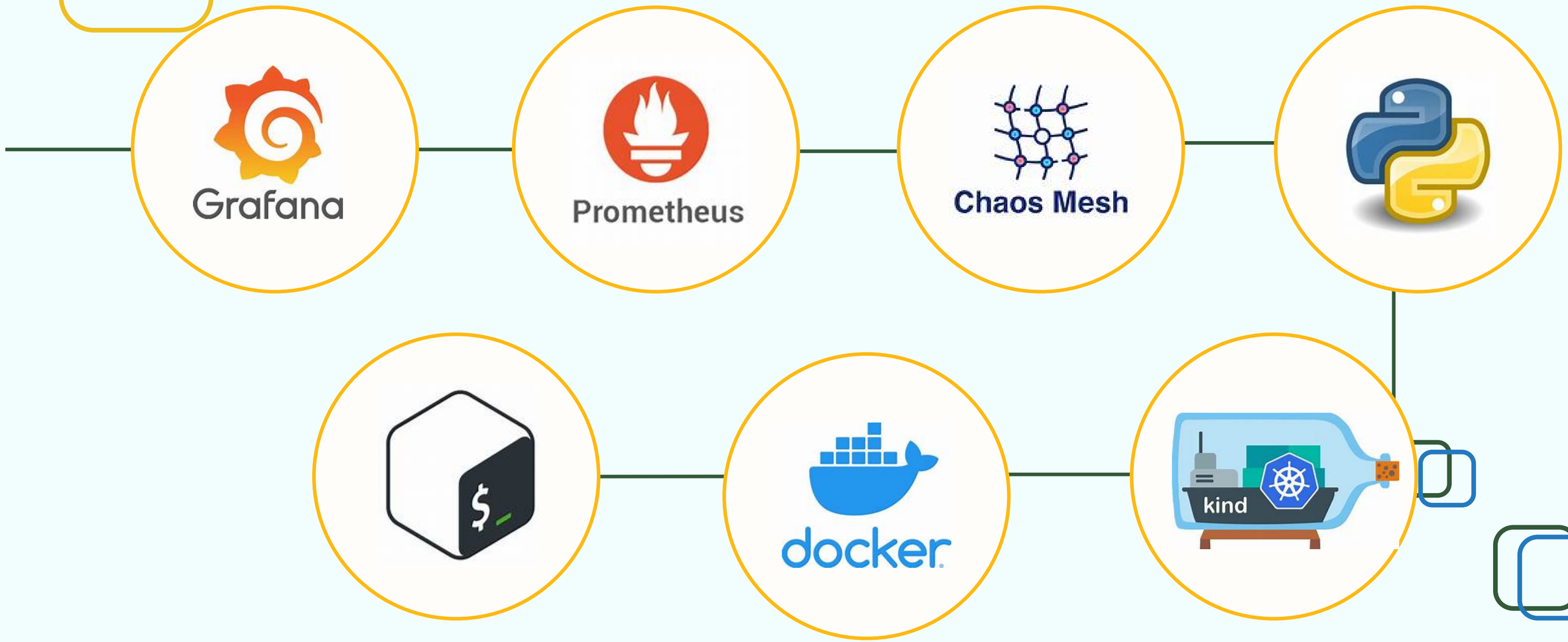
Amrita Vishwa Vidyapeetham, Coimbatore



## Members

- C S Amritha
- Anaswara Suresh M K
- Avi Nair
- Adithya N S
- R. Sruthi

# Techstack



# Dataset Generation

## Metrics Collection Using Prometheus

- Used Prometheus to scrape real-time metrics for Pods, Nodes, and Deployments.
- Created separate PromQL queries for each entity:
  - Nodes
  - Pods
  - Deployments

## Event Collection Using kubectl get events

- Collected Kubernetes events using kubectl get events.
- Grouped events by Node, Pod, and Deployment for structured analysis.

# Dataset Generation

## Error Detection Functions

**Passed metrics and event messages to corresponding error detection functions:**

- **check_node_error(): Checks for node-level errors (e.g., resource exhaustion).**
- **check_pod_error(): Checks for pod-level errors (e.g., crashes, restarts).**
- **check_deployment_error(): Checks for deployment-level errors (e.g., scaling issues).**
- **Each function appends error prediction values (e.g., True or False) to the dataset.**

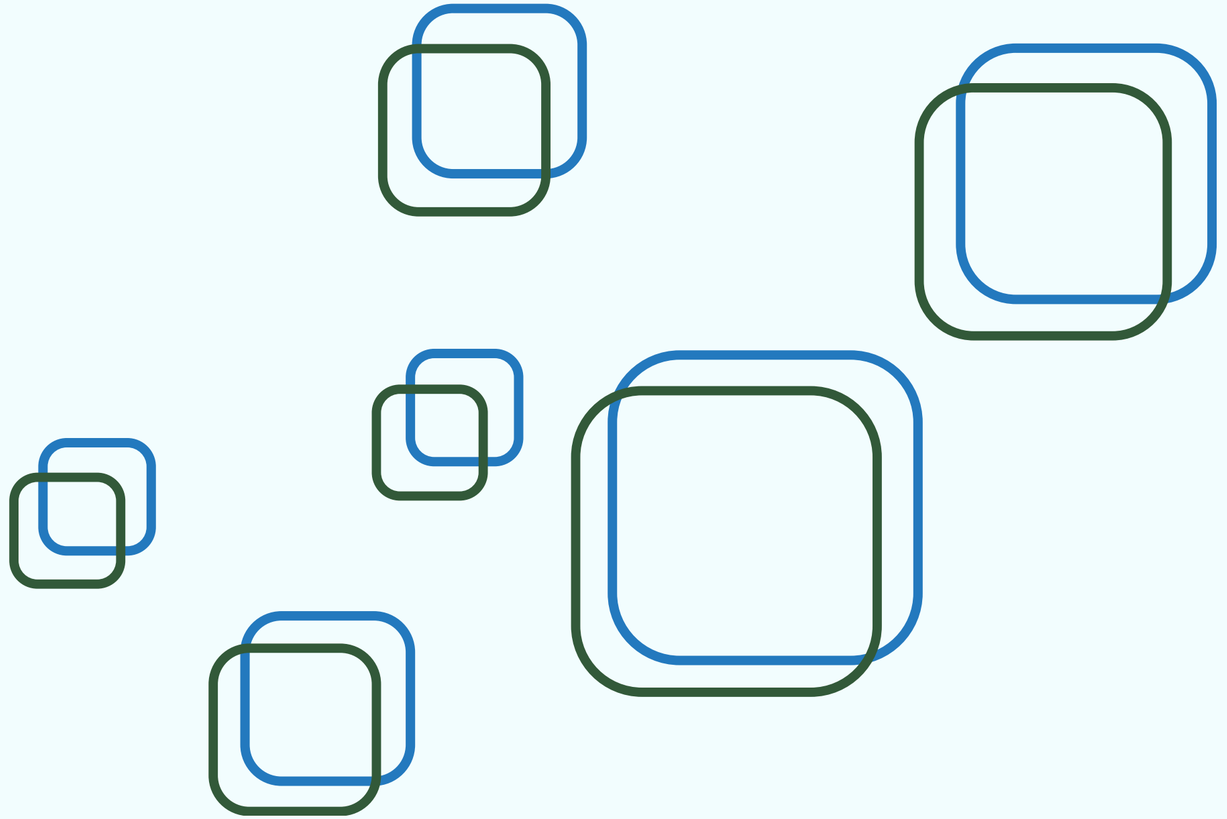
# Dataset Generation

## Data Aggregation

- Combined metrics and error predictions into a single row for each entity:
  - Node Metrics: Appended to all pods under that node.
  - Deployment Metrics: Appended to all nodes and pods under that deployment.

- Final dataset includes:
  - Metrics (CPU, memory, etc.).
  - Event messages.
  - Error prediction values.

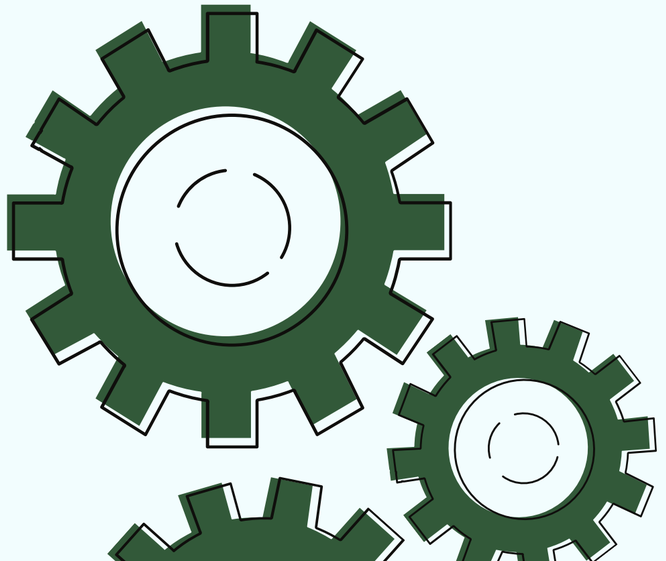
# Model



For our Kubernetes failure prediction system, we experimented with multiple models such as:

- Facebook’s Prophet
- Long Short Term Memory with Isolation Forest
- GRU

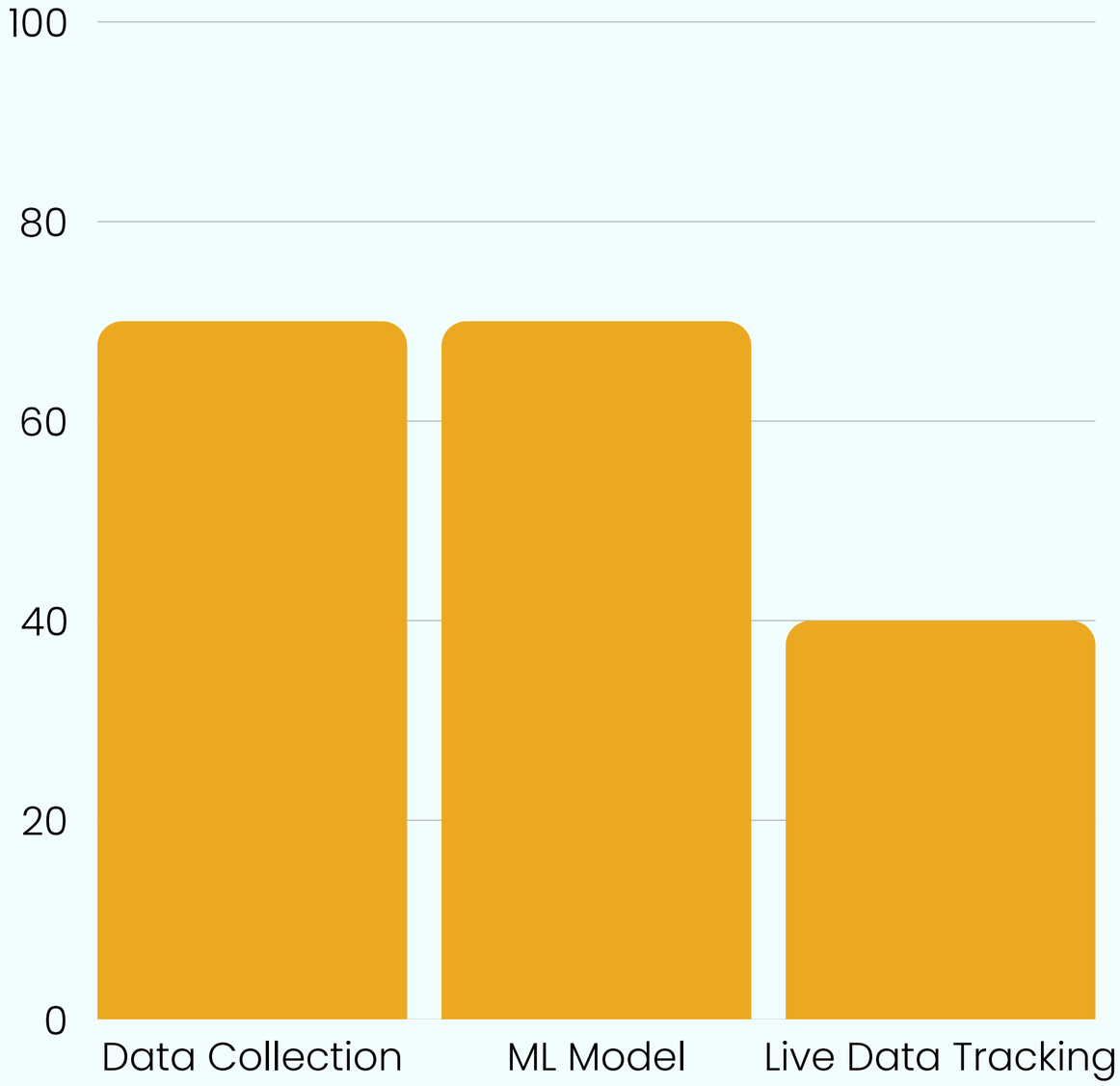
Due to its efficiency and suitability for handling short- to mid-term dependencies in time-series data, GRU became our model of choice for predicting potential failures in Kubernetes clusters.



# How GRU Is Used in the Failure Prediction System

1. Time-series data that is collect is then preprocessed and encoded as sequences using sliding window techniques to create inputs for the GRU model.
2. Data is normalized to ensure consistency and prevent bias in model training.
3. The GRU model is trained on historical time-series data to capture trends and patterns in resource utilization.
4. The model learns to predict future states of key metrics based on past observations.
5. Once deployed, the model continuously forecasts future values of CPU, memory, and other critical metrics.
6. If the predicted values indicate a high probability of resource exhaustion or failure, an early warning alert is triggered.

# Current Status



# Thank You!