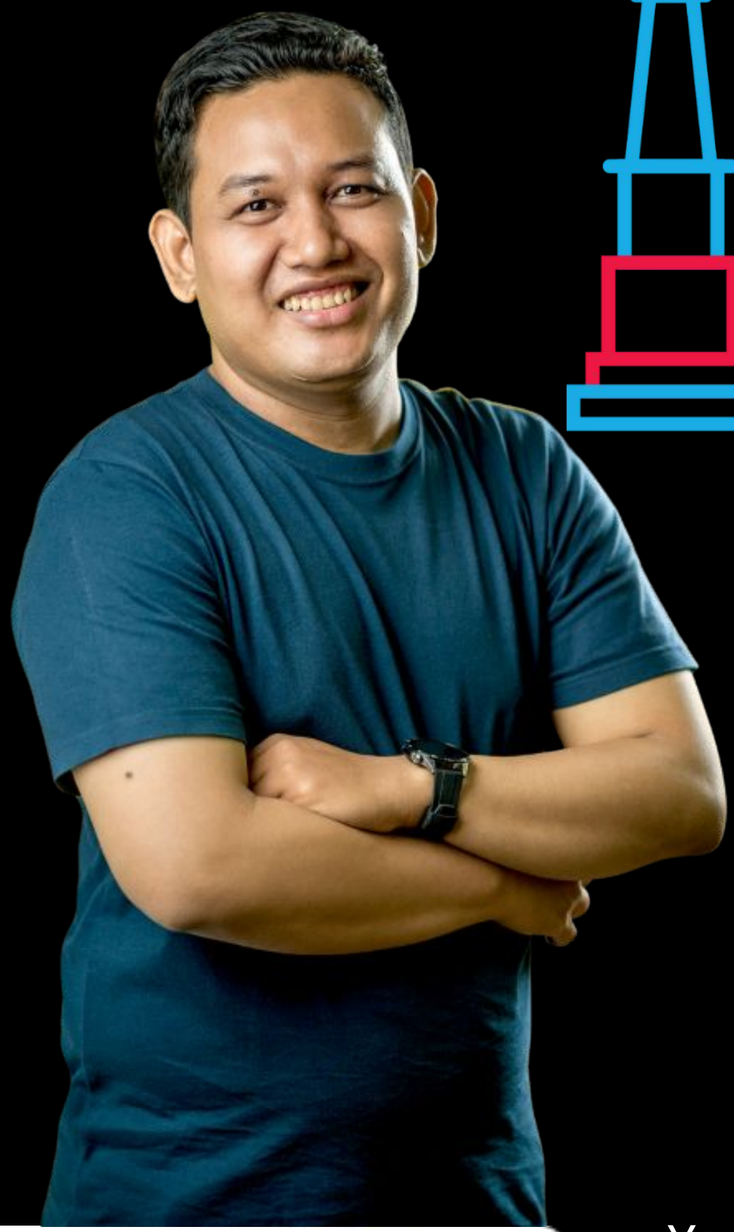
Distributed Data & Training on Kubernetes: How We Use Ray Stack at Scale

El Muhammad Cholid Hidayatullah
Founder of DevOps Focus Group

Yogyakarta, 19 July 2025

El Muhammad Cholid Hidayatullah

- Founder DevOps Focus Group
- IT Consultant at Aliz Tech Kft.
- Principal Cloud Engineer at Kourai Inc.
- Ex System Engineer Manager at Privy



Yogyakarta, 19 July 2025

EasyStack open cloud computing
AMD
datacomm
flexi
WOWRACK
ZConverter Cloud
SIVALI CLOUD TECHNOLOGY
boer technology
NASHTAGROUP
nevacloud

Let’s talk about

- Our vision at Kourai Inc.
- How our architecture look like at a glance
- Why we choose Kubernetes
- How Ray help us to solve our top 4 challenges
- Key benefits of using our existing stacks
- Challenges & lessons learned of Ray implementation
- Our future direction

Problem

We're losing the fight against drowning.

Thousands of families around the globe suffer the loss of a child every year because of drowning. So far, no one has been able to create an effective pool security system to prevent these tragedies.

#1

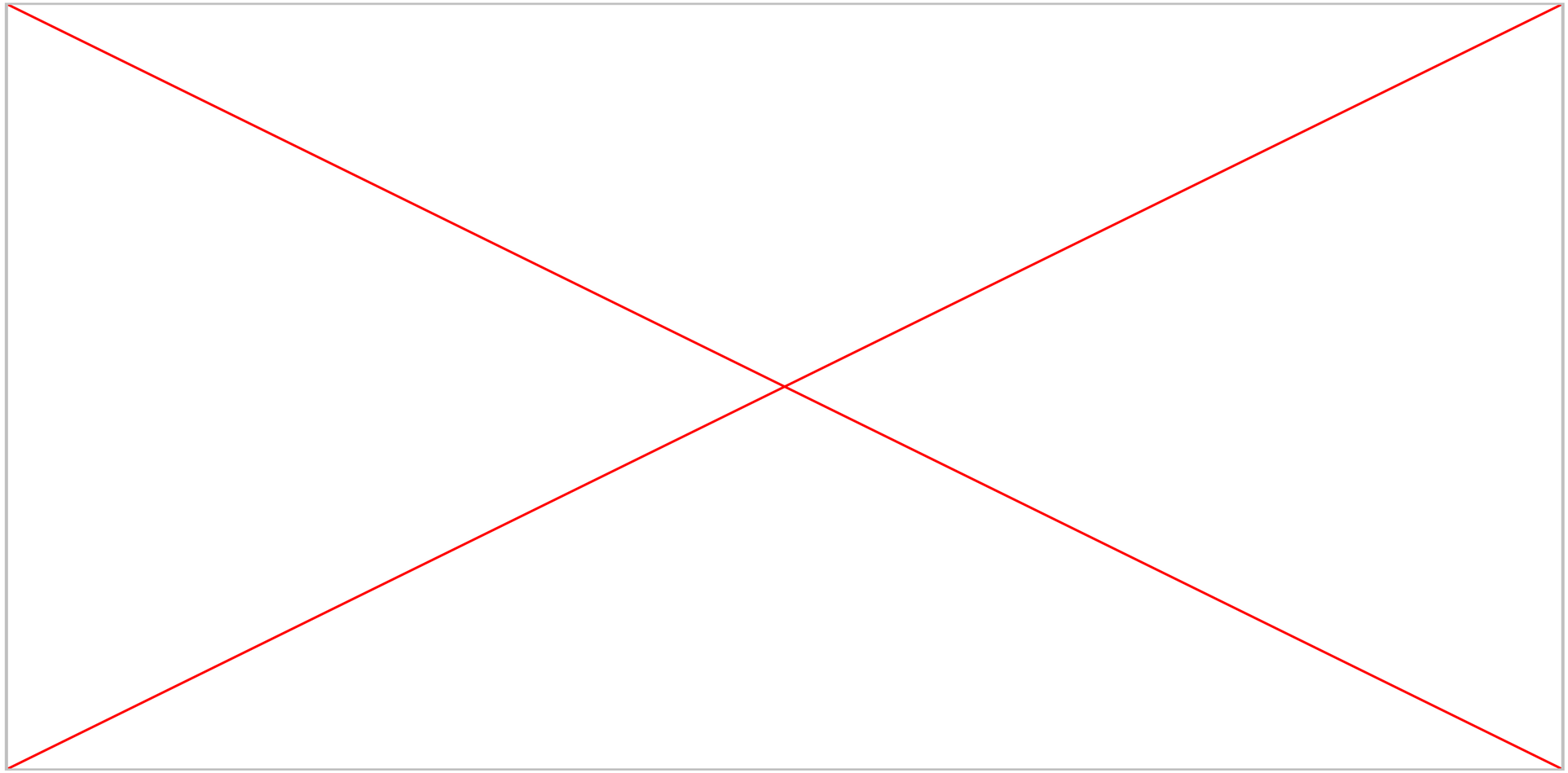
cause of accidental death for children 4 & under

11 deaths

per day from drowning in the US and 8x that in serious injury

70%

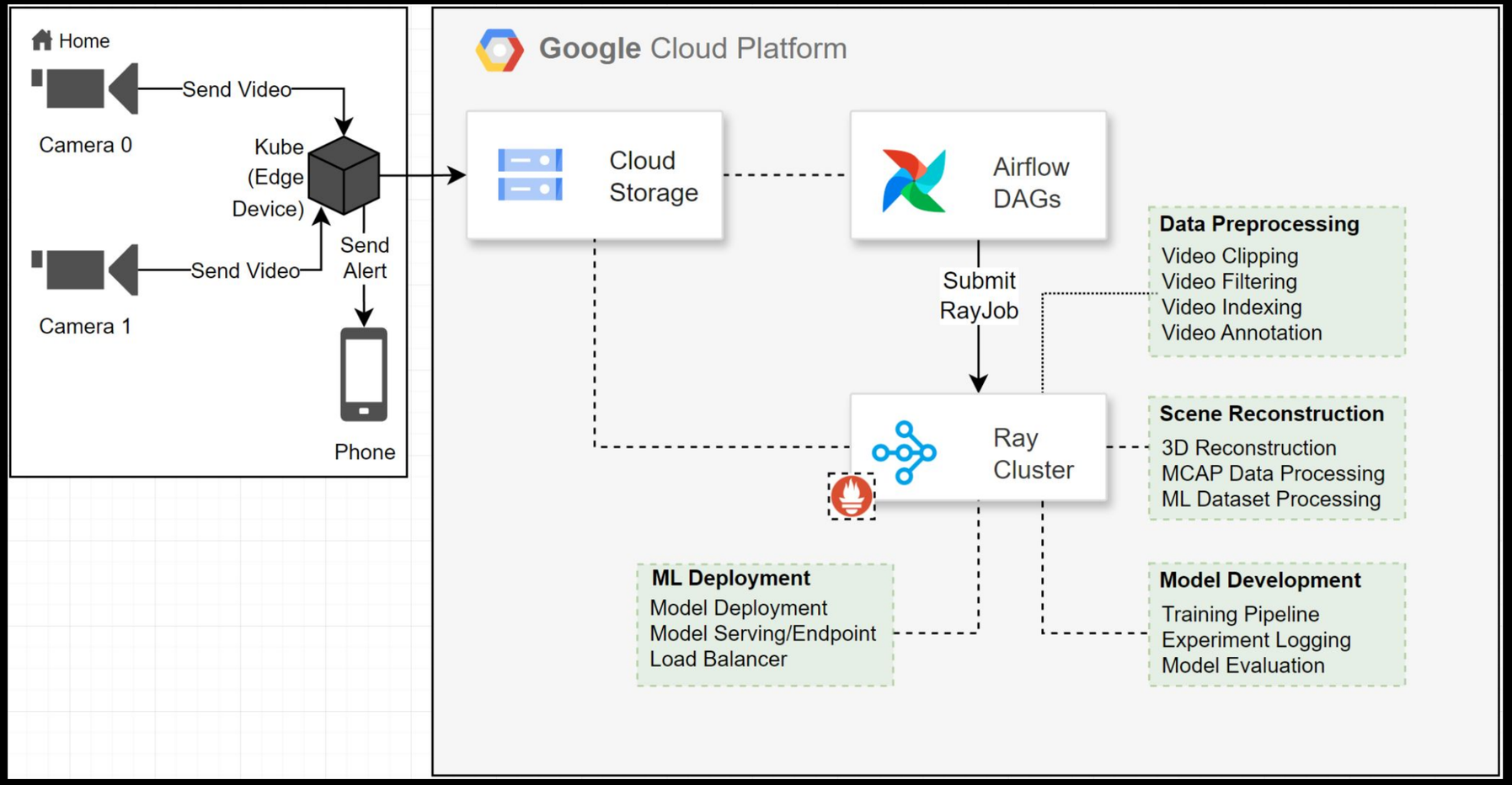
of child drownings at home occur during non-swim time



Real-time 3D monitoring of the pool area

Contextual, escalating alerts when someone is in danger

Our Infrastructure at Glance (Super Simplified)



Main challenges

- We needed quick, cost-effective spin-up/spin-down of servers for intermittent development
- Managing numerous disparate tools and APIs for different tasks
- Workloads demanded varied resources. Some require CPU-only, while others requiring specific GPU types
- Massive Video Datasets require taylor-made handler

Why we choose GKE Autopilot Cluster?

01

Fully managed

Google handles cluster operations, patching, upgrades, freeing our engineers to focus on AI.

02

Cost Optimized

Pay only for consumed resources (pods), not underlying nodes. We don't pay for idle servers.

03

Operational Simplicity

We can opt-in GPU resource seamlessly just by adding few lines on our YAML manifest.

This 3 point easily help us to overcome our scaling and on-demand request challenge

So, what is distribute data and training actually?

Why we choose Ray?

Our AI development involves diverse tasks: video preprocessing, scene reconstruction, model development, and ML deployment. Each often required different tools.

01 Ray provides a single, consistent, and unified Pythonic API

02 Effortlessly scales our Python code from a laptop to a large GKE cluster

03 Supports various ML libraries (PyTorch, TensorFlow) and data formats

04 Built-in mechanisms handle node failures, ensuring our long-running experiments

How Ray Stacks help us to solve our top 4 challenges?

RayCluster - Nodes Management of Ray

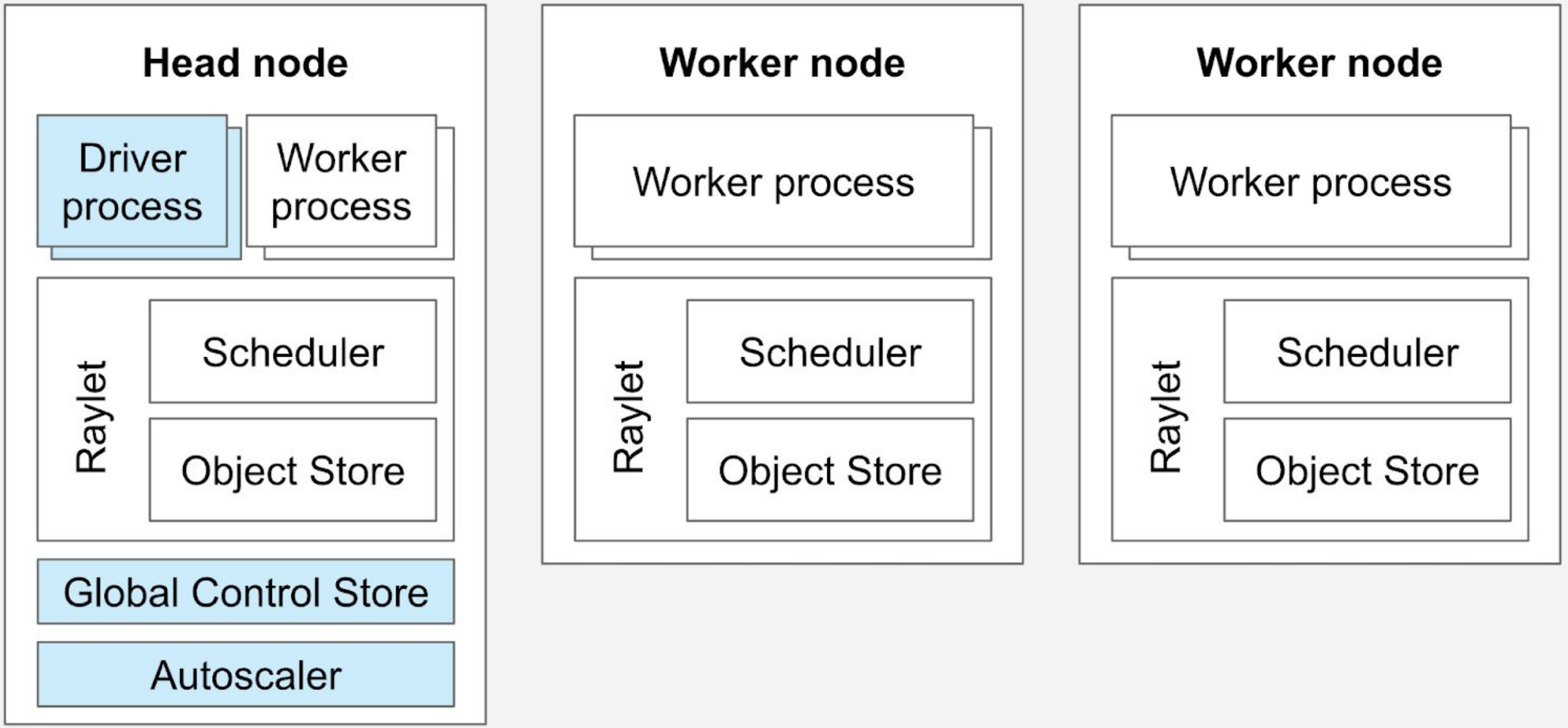
Problem

Our development workloads are highly heterogeneous. Some tasks (like initial video parsing) are CPU-intensive, while others demand specific GPU types.

How RayCluster solve our challenges

- Leverage GKE autopilot capabilities to provision nodes based on request, without the need to create node group or instance group first
- Create 3 different cluster for data processing , 3D reconstruction clusters, and training
- Enable autoscaling sidecar on head node, set the worker to scale to 0
- Use burstable pods config for CPU and Memory dependant worker
- Opt-in for spots instance for non-critical process

How RayCluster Work (Simplified)



How Ray Stacks help us to solve our top 4 challenges?

KubeRay - Ray Operator for Kubernetes

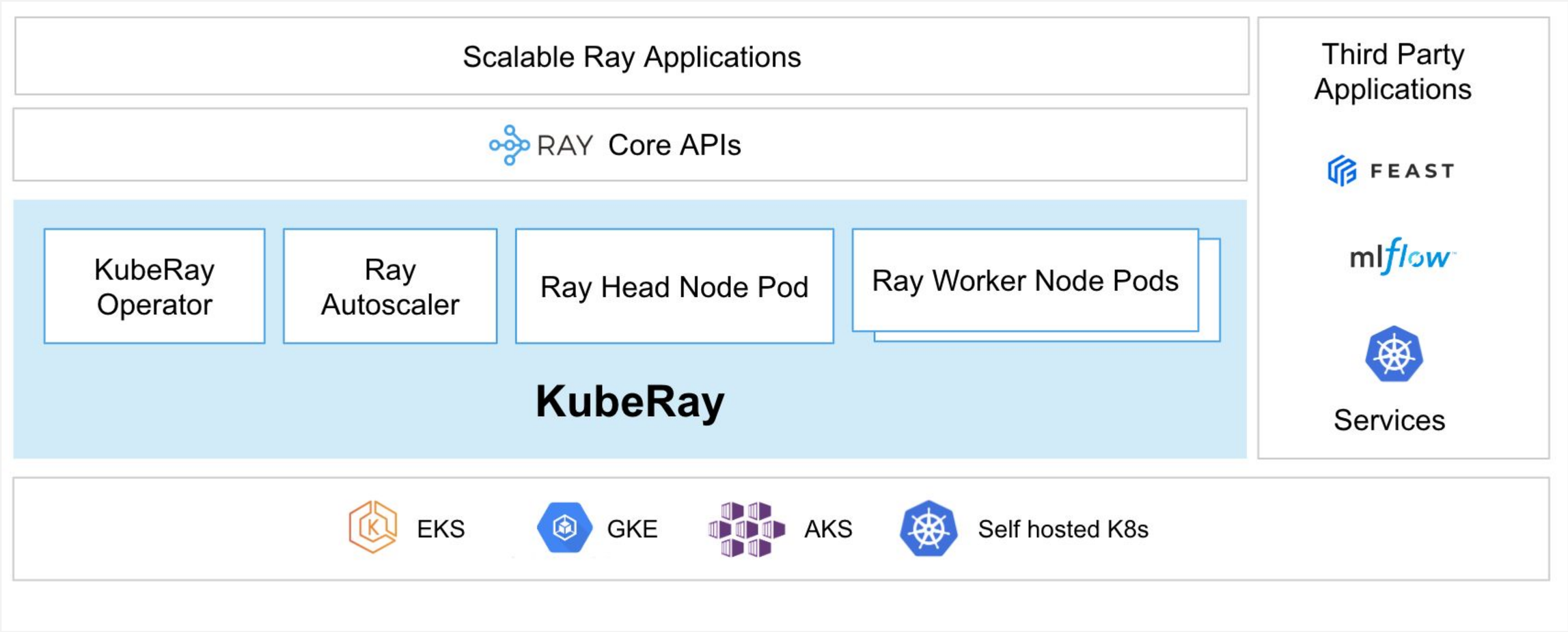
Problem

Deploying and managing Ray clusters directly on Kubernetes was complex. This involved writing intricate YAML, handling service discovery, etc.

Solution

- Deploy KubeRay using to work as kubernetes operator to simplify Ray clusters deployment.
- After KubeRay installation, we simply need to create RayCluster YAML, including head and worker group configuration.
- For GKE, you can enable KubeRay directly on the console when creating the clusters
- Installation from Ray official repository is recommended

Ray Inside Kubernetes



How Ray Stacks help us to solve our top 4 challenges?

RayData - Scalable Data Processing

Problem

We're dealing with a rapidly growing volume of video data ~10 TBs for both processing and training. Efficiently ingesting, decoding, transforming, and sharding this data from Cloud Storage into a format suitable for distributed AI training is a massive bottleneck in our development cycle.

How we use RayData to solve our challenges

- RayData reads raw video clips from GCS in parallel across all available Ray workers.
- Efficiently extracts frames from video files and performs initial scene reconstruction (e.g., identifying key objects, camera motion) across the distributed dataset.
- Applies transformations like resizing, normalization, and augmentation across the distributed dataset, ensuring consistency.
- Automatically shards data for efficient distributed training with *{REDACTED}*.

How Ray Stacks help us to solve our top 4 challenges?

RayTrain - Distributed AI Model Training

Problem

Training deep learning models for complex behavior recognition (drowning, gestures, age) on our massive datasets is computationally intensive and time-consuming

Solution

- We train deep learning models to identify drowning behaviors, hand gestures, and age groups, leveraging the processed data from RayData.
- RayTrain automatically distributes the training workload across multiple GPUs/CPUs on our RayCluster nodes.
- Automatically handles worker failures during training, resuming from checkpoints, which is critical for long experiments.
- Integrates the training with RayTune for efficient hyperparameter optimization, speeding up model iteration.

Overall Impact

- We now run millions process (job) simultaneously with full on-demand and scalable infrastructure
- We are centralizing our data and training pipeline in RayStacks
- We are able to handle ~10TB of data with only 4 individual.
- We are successfully decrease our monthly infrastructure cost by ~91%

Lesson Learned (Challenges & Solution)

Challenges	Solutions
Ray can run jobs/task on Head Node and causing OOM.	Set RayParams `num_cpus=0` and `mem=0`
By default RayCluster API are open to public and no RBAC	Use `kube-rbac-proxy` sidecar to allow token-based authentication
Disk usage issue on high-memory requirement workloads	Increase `object-store-memory` value in RayParams to prevent object spilling
Zombie nodes issue	Decrease `failureThreshold` value on readinessProbe
Job taking too long to start	Use prebuild docker image with ray as base image

Our Future Direction

01

Ray Serve

Deploying trained models directly on Ray Serve for low-latency inference at scale, potentially closer to the edge.

02

Enhanced Onsite Integration

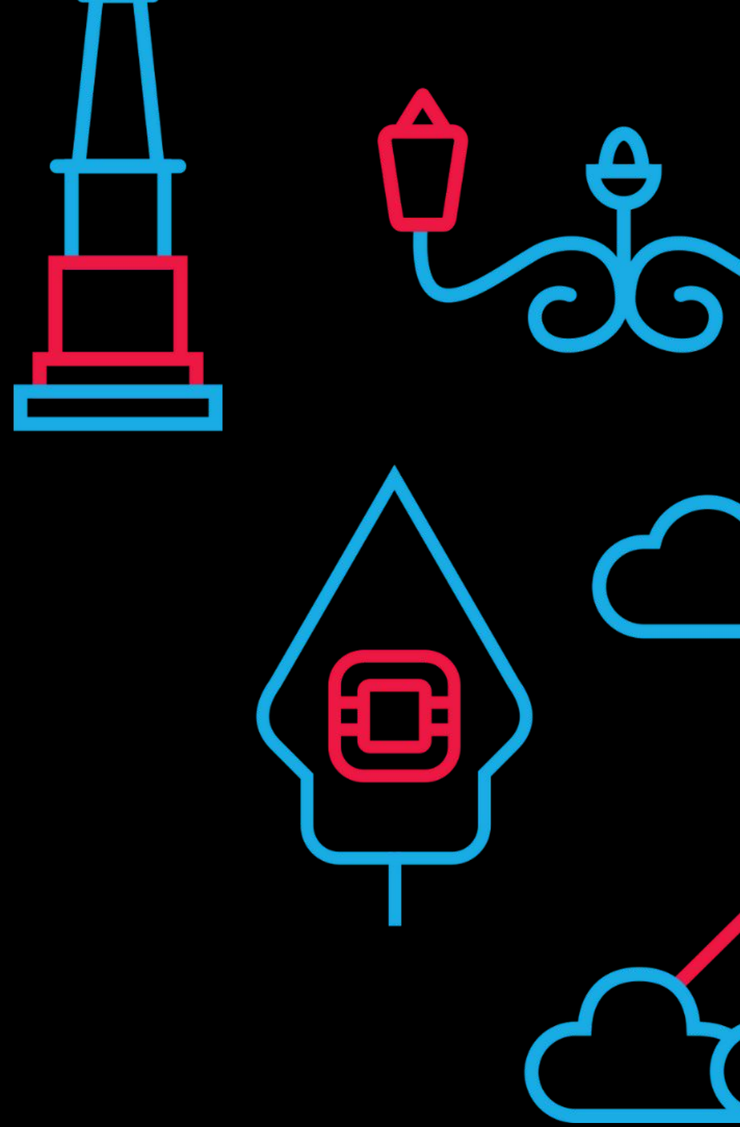
Deeper integration with onsite notification devices, potentially using Ray for managing local inference and alert logic.

03

Federated Learning

Integrating federated learning approaches with Ray to train models on edge data without centralizing raw video

THANK YOU



EasyStack open cloud computing

AMD

datacomm flexi WOWRACK

ZConverter Cloud

SIVALI CLOUD TECHNOLOGY

boer technology

NASHTAGROUP

nevacloud

Yogyakarta, 19 July 2025