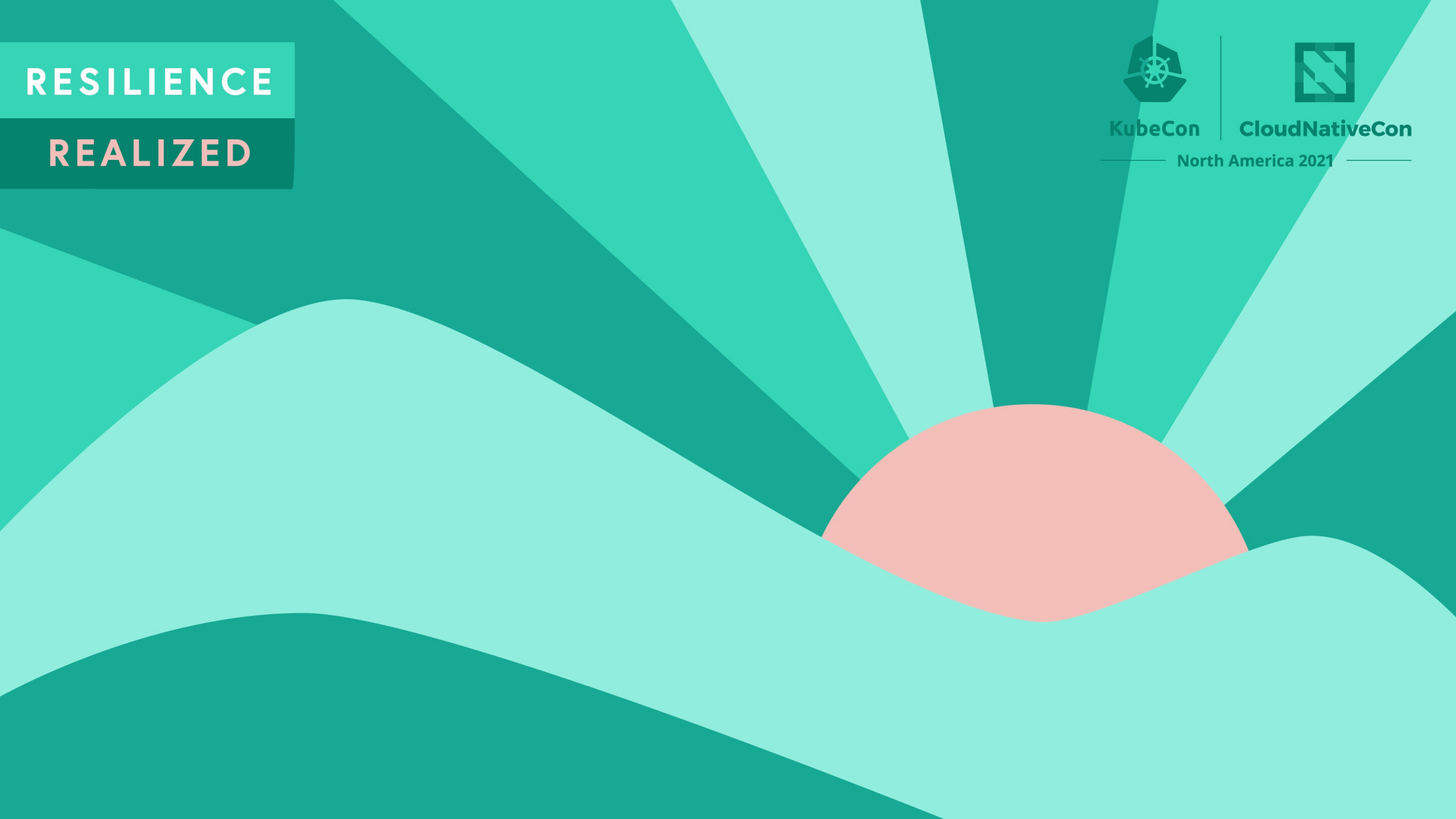
RESILIENCE REALIZED



RESILIENCE
REALIZED

AIoT Ops - Using Kubernetes and ML Ops to build Edge ML applications

Asheesh Goja, Cisco

About Me

Senior Solutions Architect @ Cisco

Team : Emerging Technologies and Incubation

CISCO

Topics

Talk – *Why, What and how of AIoT ML Ops*

- Why
  - AIoT
  - ML Ops
- What
  - Emergent behaviors
  - AIoT solution Patterns
  - Reference Architecture
- How
  - Reference implementation using K3S, TFLite and Argo
  - Deployment topology
  - Electrical circuit setup

Demo - *Step by step guide to build and deploy an AIoT MLOps solution*

- Run the demo in a fully configured environment and show
  - ML workflow DAGs – Run Extract , Drift , Quantize and Train jobs
  - Perform Edge Inferences on Coral TPU cluster
  - Perform MCU inference on ESP32 RTOS
  - Introduce fault on the motor and detect it
- Tear it down the demo and show how to rebuild:
  - Infrastructure
    - Setup K3S and Strimzi
    - Setup Argo workflows
  - Platform
    - Setup model registry
    - Setup container registry
    - Build and deploy protocol bridge and FOTA
  - Application
    - Configure Argo DAGs to deploy Extract, drift detection, training and quantization modules
    - Build and deploy AIoT ML module to TPU and MCU devices
    - Show HA and Scaling capabilities

AIoT – Artificial Intelligence of Things

Problem – By 2025

- 55 billion IoT devices
- 73.1 Zettabytes of data
- Data deluge – more data less insight
- Privacy concerns
- Wide cyber attack vector with internet connected devices

The solution - Embed ML in IoT

- Perform inferences close to sensors and actuators
- Minimize need for cloud connectivity
- Execute close loop anomaly detection and intervention
- Enable operations that require low latency communication
- Improve s/n ratio - Reduce inconsequential data uploads to the cloud
- Generate strong privacy protection – keep data on the device or edge tier

Emergent behaviors and considerations

- Computational complexity impacts resource availability
  - Training vs. Inference complexity
  - Space, time and sample complexity
  - Computation, memory, bandwidth, energy
  - Accuracy, resource and complexity tradeoffs
- Limited memory, processing power and energy capacity impacts inference capability
- ML frameworks are too bulky for embedded devices
- Insufficient metrics to measure model performance
  - MACs and FLOPs inadequate
- Optimization strategies introduce training and model entropy
  - Quantization error
  - Errors introduced due to binarization of activations and weights
- Multiple incompatible computational hardware architectures
  - ARM64 vs x86 vs AMD64 vs ARM Cortex-M

ML Model	Size	Training	Inference
Random Forest	$O(N_{tree}\, m)$	$O(N_{tree}\, mn\, (\log(m))$	$O(N_{tree}\, (\log(m))$
Logistic Regression	**O(n)**	**$O(mn^2 + n^3)$**	**O(n)**
Naive Bayes	O(nc)	O(Mn + nc)	O(nc)
Support-vector machine	O(n)	$O(m^2n)$	$O(m_{sv}n)$
m samples, c classes and n dimensions			

AIoT Solution Patterns

ML Ops

- Express ML Ops dependencies as DAGs
- Manage extract, drift detection, training, validation and quantization on the edge tier
- Use workflow pipelines for continuous evaluation and training
- Use hardware accelerator aware pod placement strategies

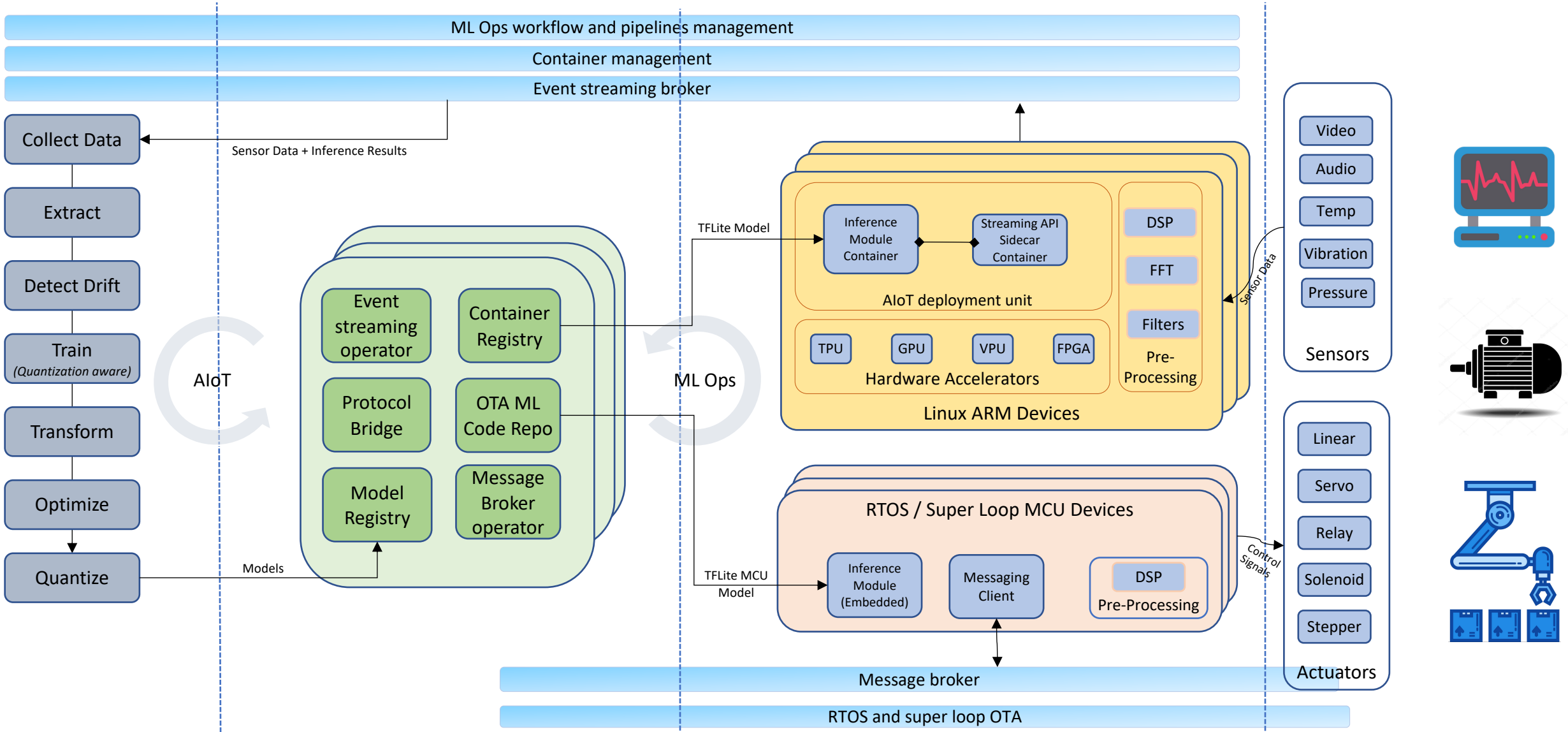
Architecture

- Energy efficient hardware acceleration
- Containerization
- Streaming API sidecar
- Decouple embedded inference from communication
- Automated container orchestration on edge devices

Embedded ML

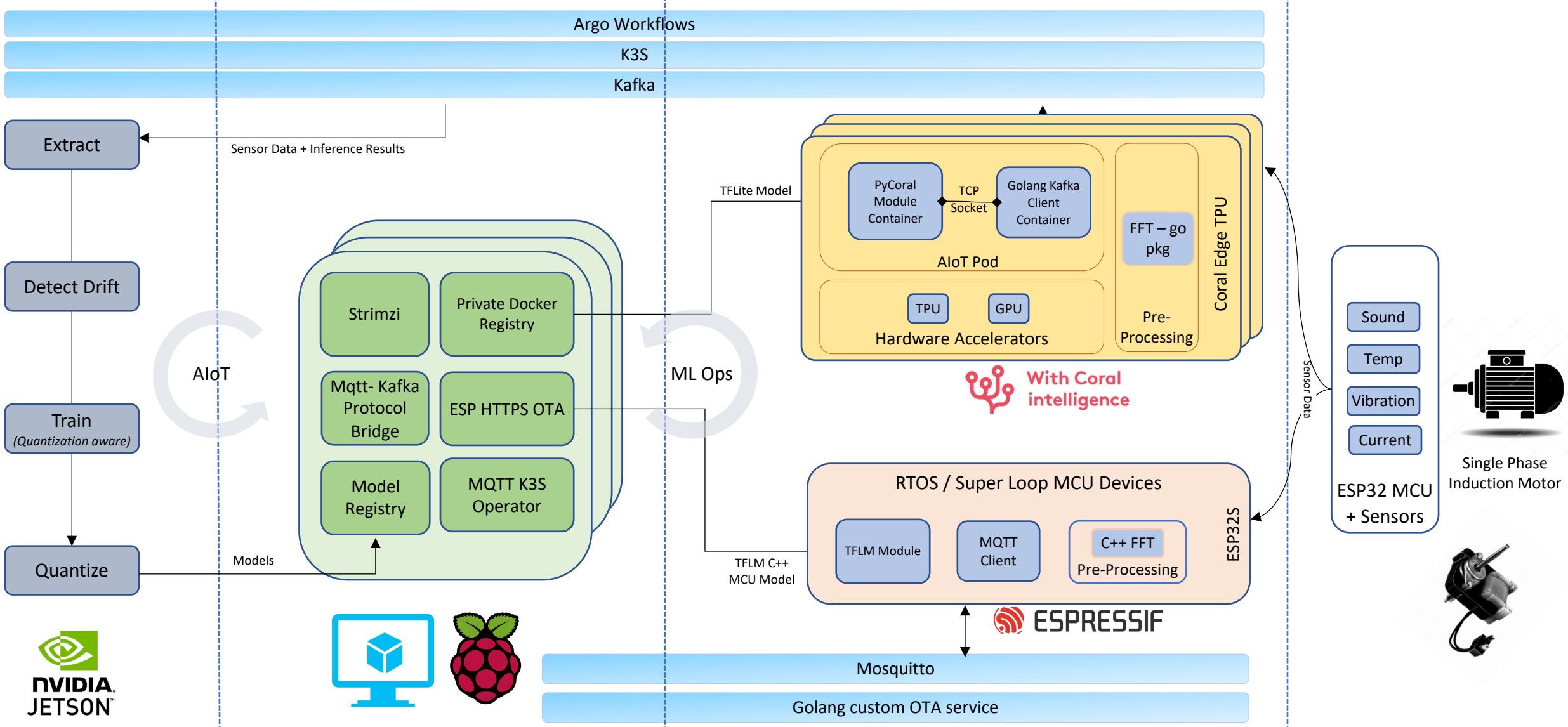
- Decouple training from inference
- Introduce preprocessing Inferencing at analog tier
- Use early termination and cascaded models
- Binarize weights and activations at train time
- Use bit-shift operations over multiplications
- Explore model partitioning, edge caching and input filtering
- Perform data compression or sparsification
- Use Transfer Learning
- Use DSP to filter out noise

AIoT ML Ops Reference Architecture



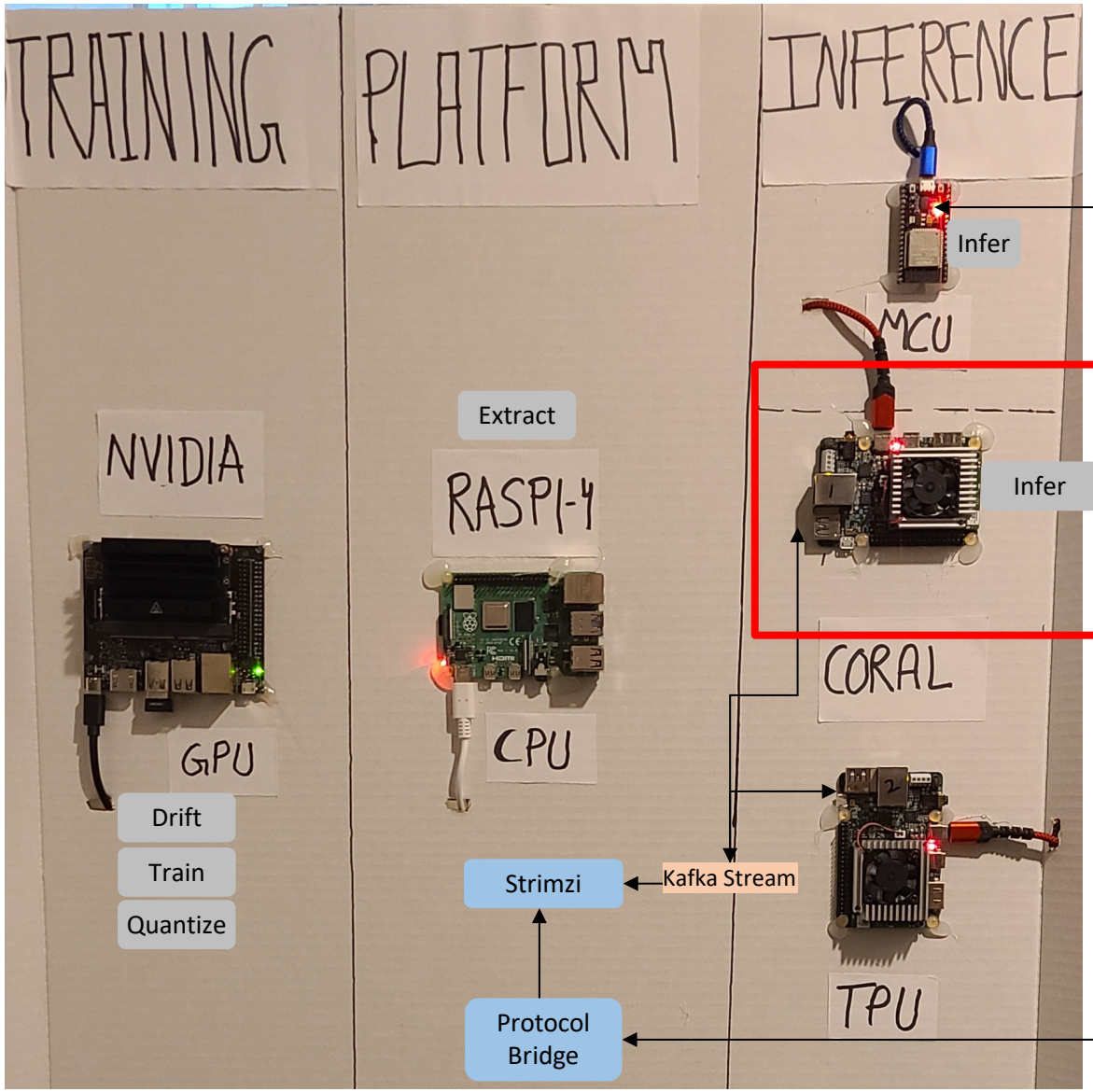
Reference Implementation- Strmizi, K3S, Argo and TFLite

KubeCon CloudNativeCon North America 2021

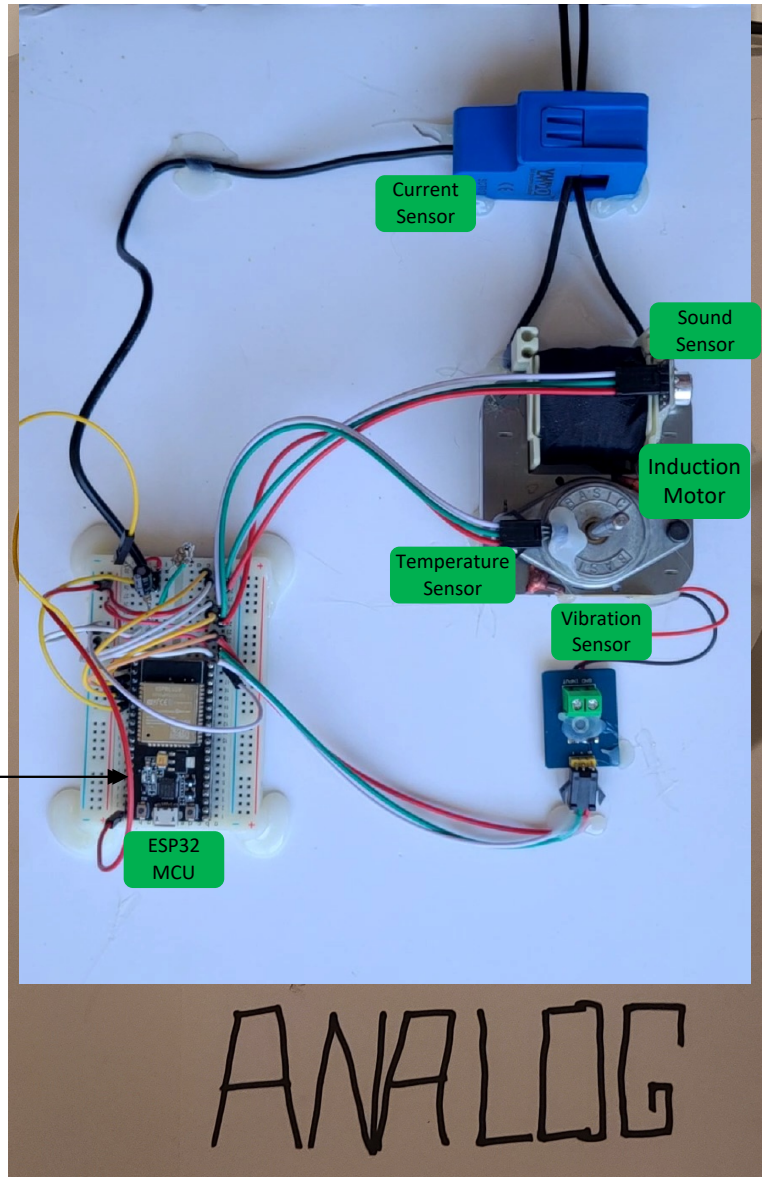


Let’s start building…Demo Time

Demo- On stage setup

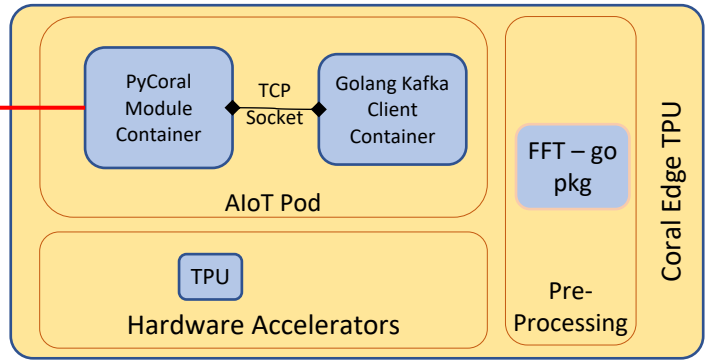


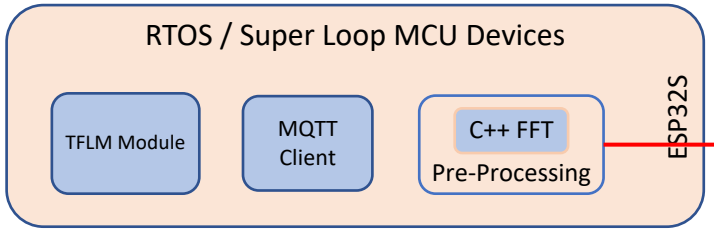
MQTT



Under the hood – TPU and MCU Inference modules


```python
#uncomment this section to te  Quick Fix... (⌘.)
from pycoral.adapters import classify
from pycoral.adapters import common
from pycoral.utils.dataset import read_label_file
from pycoral.utils.edgetpu import make_interpreter
from periphery import GPIO

ledB = GPIO("/dev/gpiochip2", 9, "out")  # 16
ledG = GPIO("/dev/gpiochip4", 10, "out") # 18
ledR = GPIO("/dev/gpiochip4", 12, "out") # 22

try:
    current = np.float32(json.loads(msg)["current"])
    temperature = np.float32(json.loads(msg)["temperature"])
    vibration = np.float32(json.loads(msg)["vibration"])
    sound = np.float32(json.loads(msg)["sound"])

    telemetry = [current, temperature, vibration, sound]

    np_arr_64 = np.array(telemetry)
    np_arr_f32 = np_arr_64.astype(np.float32)

    inp_details = interpreter.get_input_details()
    out_details = interpreter.get_output_details()

    interpreter.set_tensor(inp_details[0]['index'], [telemetry])

    interpreter.invoke()

    output_details = interpreter.get_output_details()
    predictions = interpreter.get_tensor(output_details[0]['index'])

    output_index = interpreter.get_output_details()[0]["index"]
    ten = interpreter.get_tensor(output_index)

    fault = float(('%f' % ten))

    hn_ip_tuple = getHostnameAndIpAddress()

    logToFile(bcolors.BOLD +
        "Edge TPU logistic regression inference value : {}".format(fault))
```



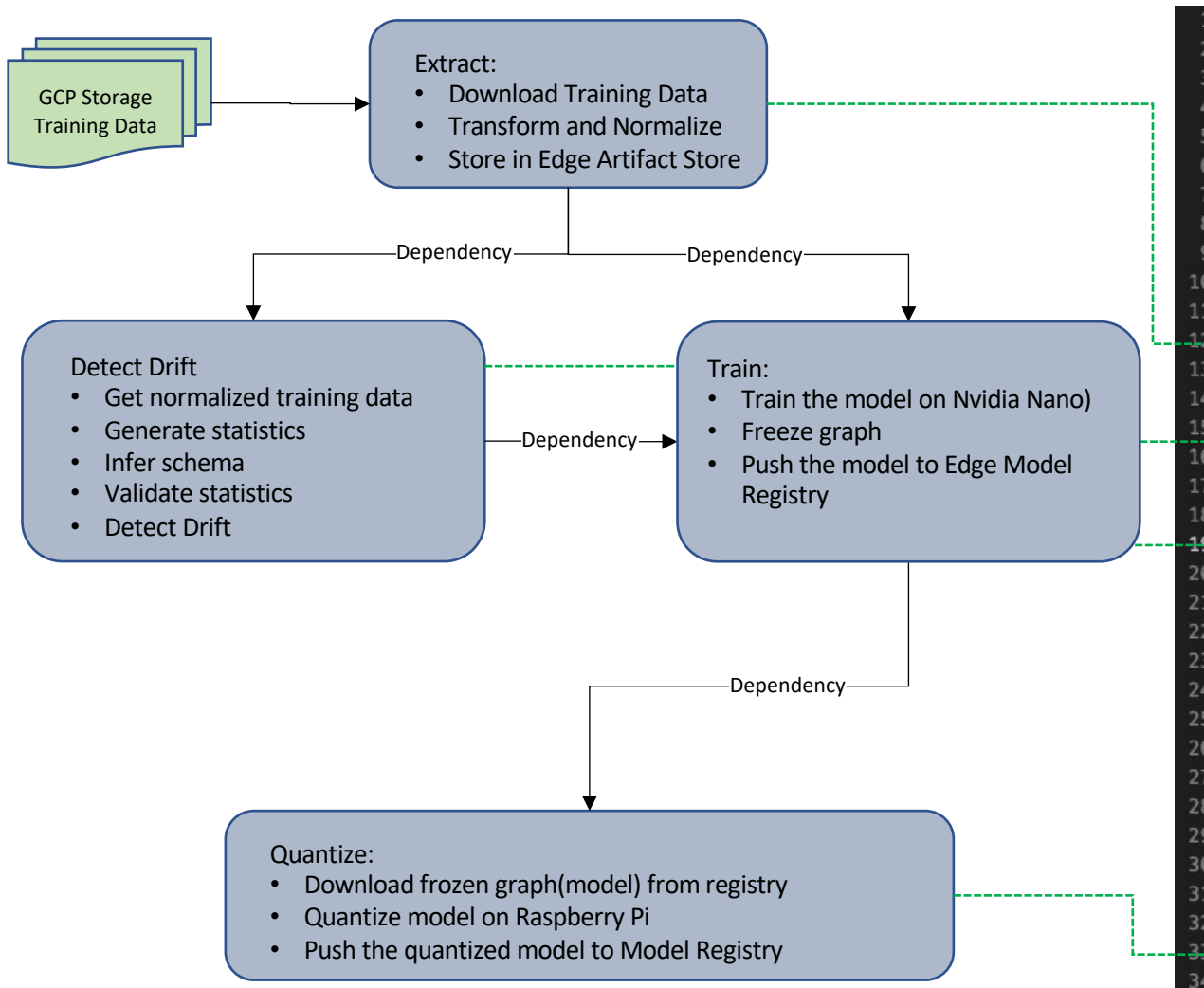



```cpp
#include "tensorflow/lite/micro/all_ops_resolver.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"

interpreter = new tflite::MicroInterpreter( model, *resolver, tensor_a

TfLiteStatus allocate_status = interpreter->AllocateTensors();
if (allocate_status != kTfLiteOk)
{
    TF_LITE_REPORT_ERROR(error_reporter, "AllocateTensors() failed");
    return;
}

size_t used_bytes = interpreter->arena_used_bytes();
TF_LITE_REPORT_ERROR(error_reporter, "Used bytes %d\n", used_bytes);

// Obtain pointers to the model's input and output tensors.
input = interpreter->input(0);
output = interpreter->output(0);

Serial.printf("mqtt message for TFLM inference - Current: %d, Temp: %d, Vibration: %d, S

input->data.f->getInputBuffer()[0] = current;
input->data.f->getInputBuffer()[1] = temperature;
input->data.f->getInputBuffer()[2] = vibration;
input->data.f->getInputBuffer()[3] = sound;

interpreter->Invoke();
float result = output->data.f[0];

Serial.printf("performing tflm inference on the mqtt mesage result = %.2f \n", result);

if (result > 0.7)
{
  for (int i = 0; i < 30; i++)
  {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(100);
    digitalWrite(LED_BUILTIN, LOW);
    delay(100);
  }

  char sensor_data[255];
  snprintf(sensor_data, 255, " {\"inference-level\": \"%d\",\"chipID\": \"%d\",\"current
          2, chipId, current, temperature, vibration, sound);
  client.publish(outTopic, sensor_data);
```

PyCoral Logistic Regression Module

C++ TFLM Logistic Regression

Under the hood : Argo DAG - Train and Quantize Pipeline



MLOps Train/Quantize Pipeline

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: kubecon-aiotdemo-dag-
spec:
  entrypoint: kubecon-aiotdemo-dag
  templates:
  - name: kubecon-aiotdemo-dag
    dag:
      tasks:
      - name: extract
        template: extract-template
        arguments:
          parameters:
          - name: message
            value: ""
      - name: validate
        dependencies: [extract]
        template: validate-template
        arguments:
          parameters:
          - name: message
            value: ""
      - name: train
        dependencies: [validate,extract]
        template: train-template
        arguments:
          parameters:
          - name: message
            value: ""
      - name: quantize
        dependencies: [train]
        template: quantize-template
        arguments:
          parameters:
          - name: message
            value: ""
```

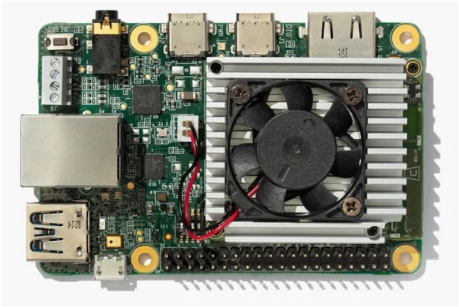
```yaml
- name:  extract-template
  inputs:
    parameters:
    - name: message
  container:
    image: docker.35.202.84.160.nip.io:5000/extract_module:latest
    securityContext:
      privileged: true
    env:
    - name: TRAINING_DATA_UPLOAD_REGISTRY_URL
      value: "http://35.236.22.237:30007/uploadTrainingData"
    - name: GCP_BUCKET
      value: "kubecon2021-aiot-mlops-demo"
  nodeSelector:
    kubernetes.io/hostname : "kubecon-aiot-control-node"

- name:  validate-template
  inputs:
    parameters:
    - name: message
  container:
    image: docker.35.202.84.160.nip.io:5000/validation_module:latest
    env:
    - name: TRAINING_DATA_URL
      value: "http://35.236.22.237:30007/training_data"
  nodeSelector:
    kubernetes.io/hostname : "kubecon-aiot-control-node"

- name:  train-template
  inputs:
    parameters:
    - name: message
  container:
    image: docker.35.202.84.160.nip.io:5000/training-module:latest
    env:
    - name: MODEL_REGISTRY_URL
      value: "http://35.236.22.237:30007/uploadModel"
  nodeSelector:
    kubernetes.io/hostname : "agentnode-nvidia-jetson"

- name:  quantize-template
  inputs:
    parameters:
    - name: message
  container:
    image: docker.35.202.84.160.nip.io:5000/quantize-module:latest
    securityContext:
      privileged: true
    env:
    - name: MODEL_DOWNLOAD_REGISTRY_URL
      value: "http://35.236.22.237:30007/full"
```

Argo DAG YAML

Edge ML Devices and Accelerators

Google Coral Edge TPU

- CPU - Quad Cortex-A53 @ 1.5 GHz
- GPU - Vivante GC7000Lite
- TPU - Google Edge TPU coprocessor
- VPU - 4Kp60 HEVC/H.265
- Memory - 1 GB LPDDR4

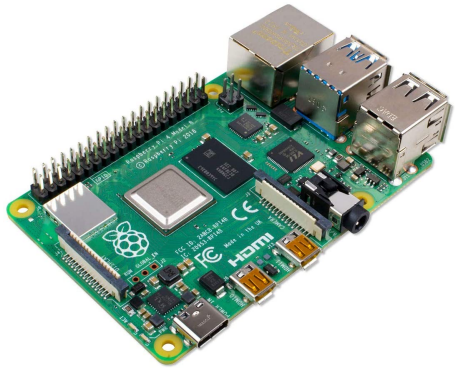


NVIDIA Jetson Nano

- CPU – Quad-core ARM® A57 @ 1.43 GHz
- GPU - 128-core NVIDIA Maxwell™
- Memory - 2 GB 64-bit LPDDR4

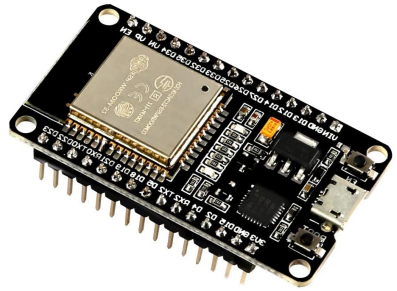
Raspberry Pi 4

- CPU - Quad Cortex-A72 @ 1.5GHz
- Memory - 4GB LPDDR4



ESP32S

- MCU – Dual Core Xtensa® 32-bit LX6 @ 40Mhz
- Memory - 448 KB ROM, 520 KB SRAM



AIoT Demo – Deployment Topology

Argo Workflows

K3S

Kafka

Extract

Detect Drift

Train

Quantize

Containers

Training Jobs

K3S Server Nodes (HA Topology)

Private Container Registry

Argo workflows Operator

Model Registry

Strimzi Kafka Operator

Protocol Bridge

Message Broker operator

Containers

ML Ops Services

Agent Node 3

Agent Node 2

Agent Node 1

Raspberry Pi 4

Nvidia Jetson Nano

Coral Edge TPU

Logistical Regression ML Module

Event Streaming API Sidecar

Containers

MCU Device 2

MCU Device 1

ATmega328

ESP32S

TFLM Module

MQTT Client

C++ FFT Pre-Processing

Embedded ML

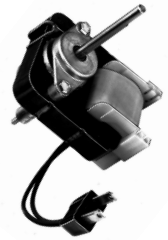
Sound

Temp

Vibration

Current

Sensors



AC Motor

Mosquitto

Golang custom OTA service

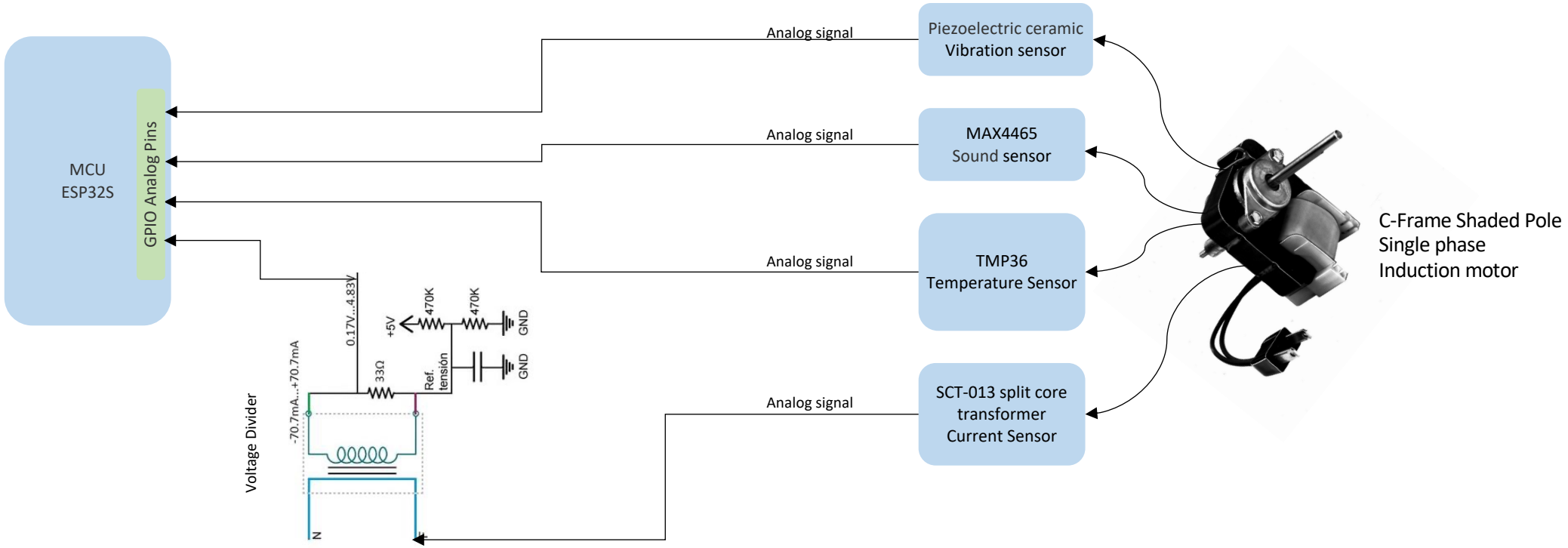
Training Tier

Platform Tier

Inference and IoT Tier

Analog Tier

Electrical Circuit Setup



Live Demo – K3S, Strimzi and Argo Setup

K3S Cluster

```
root@kubecon-aiot-master2:/home/agoja# kubectl get nodes -o wide -w
NAME                     STATUS   ROLES                  AGE     VERSION        INTERNAL-IP      EXTERNAL-IP     OS-IMAGE                         KERNEL-VERSION     CONTAINER-RUNTIME
kubecon-aiot             Ready    control-plane          6d10h   v1.21.4+k3s1   10.128.15.203    35.202.84.160   Ubuntu 20.04.3 LTS               5.11.0-1018-gcp    containerd://1.4.9-k3s1
follower-node-coral-tpu3 Ready    <none>                 19m     v1.21.4+k3s1   10.0.0.229       <none>          Mendel GNU/Linux 5 (Eagle)       4.14.98-imx        containerd://1.4.9-k3s1
follower-node-raspi      Ready    <none>                 31m     v1.21.4+k3s1   10.0.0.130       <none>          Raspbian GNU/Linux 10 (buster)   5.10.60-v7l+       containerd://1.4.9-k3s1
follower-node-coral-tpu2 Ready    <none>                 23m     v1.21.4+k3s1   10.0.0.210       <none>          Mendel GNU/Linux 5 (Eagle)       4.14.98-imx        containerd://1.4.9-k3s1
follower-node-coral-tpu1 Ready    <none>                 24m     v1.21.4+k3s1   10.0.0.15        <none>          Mendel GNU/Linux 5 (Eagle)       4.14.98-imx        containerd://1.4.9-k3s1
```

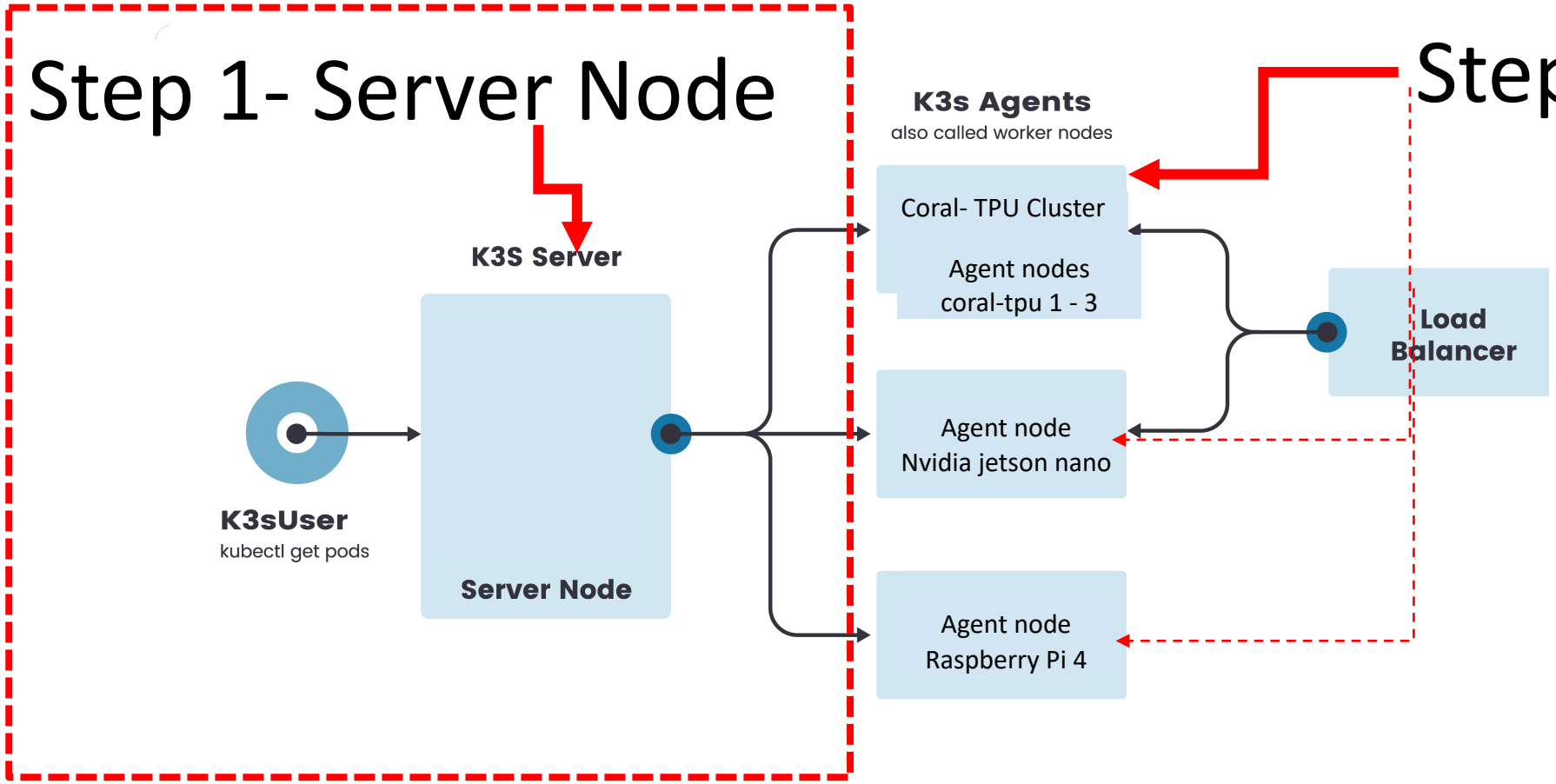
Strimzi Kafka

```
NAME                                          READY   STATUS    RESTARTS   AGE     IP            NODE           NOMINATED NODE   READINESS GATES
strimzi-cluster-operator-7695cb5f7f-5ptqk     1/1     Running   0          6m46s   10.42.0.44    kubecon-aiot   <none>           <none>
my-cluster-zookeeper-0                        1/1     Running   0          2m16s   10.42.0.45    kubecon-aiot   <none>           <none>
my-cluster-kafka-0                            1/1     Running   0          101s    10.42.0.46    kubecon-aiot   <none>           <none>
my-cluster-entity-operator-876668cdf-hczpq    3/3     Running   0          66s     10.42.0.47    kubecon-aiot   <none>           <none>
```

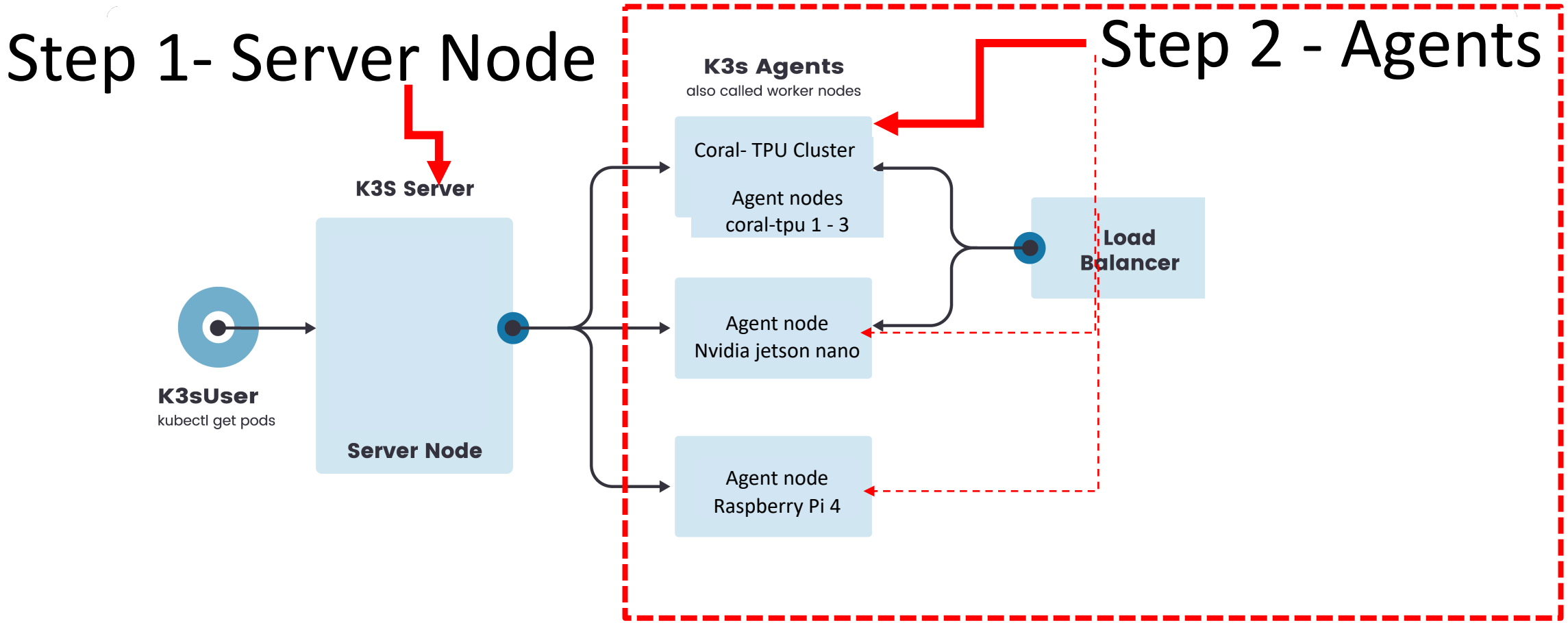
Argo ML Ops Deployments

```
NAME                                          READY   STATUS    RESTARTS   AGE     IP            NODE                   NOMINATED NODE   READINESS GATES
strimzi-cluster-operator-7695cb5f7f-5ptqk     1/1     Running   0          11m     10.42.0.44    kubecon-aiot           <none>           <none>
my-cluster-zookeeper-0                        1/1     Running   0          6m36s   10.42.0.45    kubecon-aiot           <none>           <none>
my-cluster-kafka-0                            1/1     Running   0          6m1s    10.42.0.46    kubecon-aiot           <none>           <none>
my-cluster-entity-operator-876668cdf-hczpq    3/3     Running   0          5m26s   10.42.0.47    kubecon-aiot           <none>           <none>
coral-python-deployment-77645bbb66-48kjq      2/2     Running   0          11s     10.42.7.7     follower-node-coral-tpu1   <none>           <none>
coral-python-deployment-77645bbb66-xz6qr      2/2     Running   0          11s     10.42.9.3     follower-node-coral-tpu3   <none>           <none>
coral-python-deployment-77645bbb66-nzch7      2/2     Running   0          11s     10.42.8.3     follower-node-coral-tpu2   <none>           <none>
```

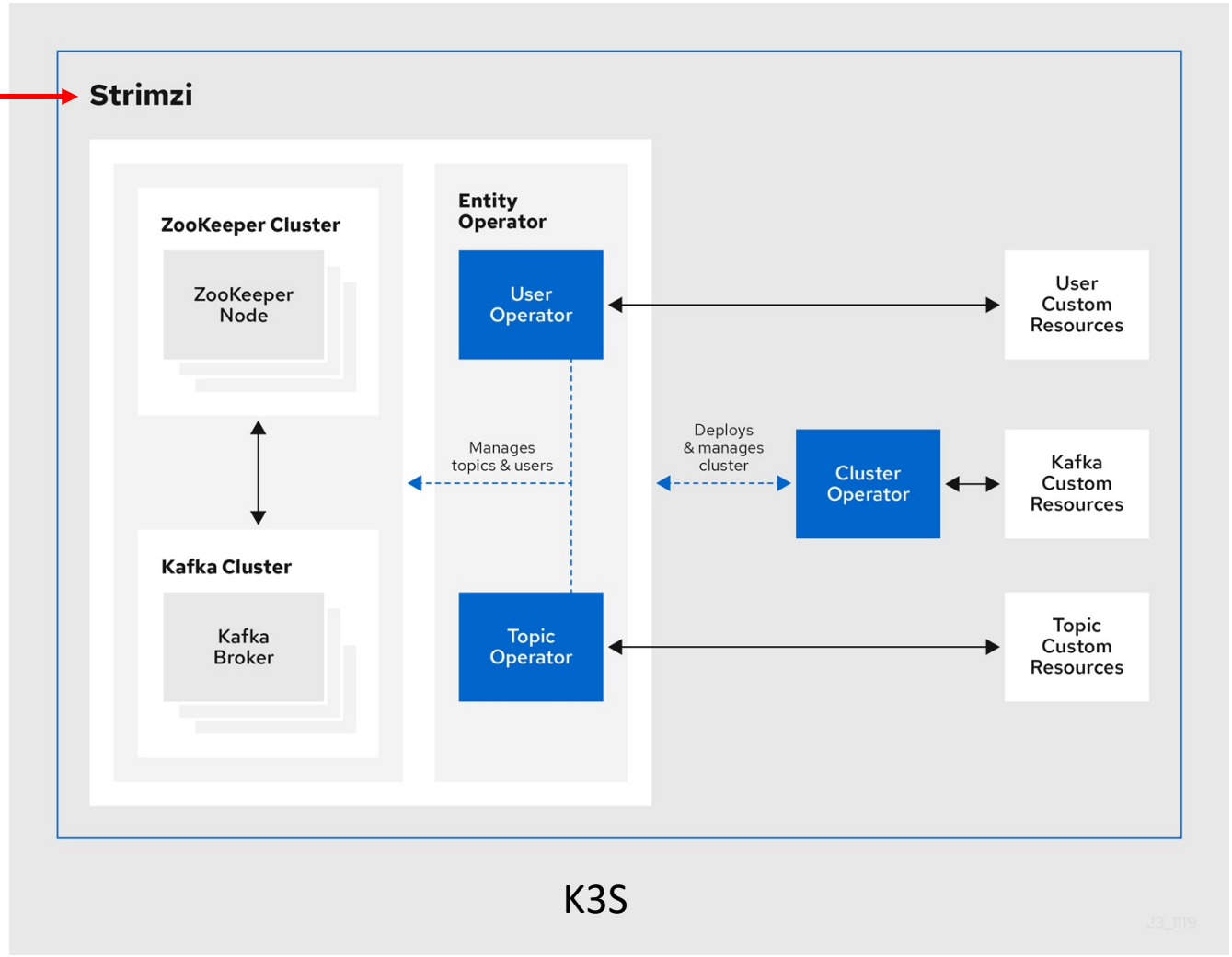
K3S Architecture



K3S Architecture



Strimzi Kafka Architecture



Argo Workflows Architecture