INDUSTRIAL EDGE INSIGHTS PLATFORM

LEGAL NOTICES AND DISCLAIMERS

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at www.intel.com.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

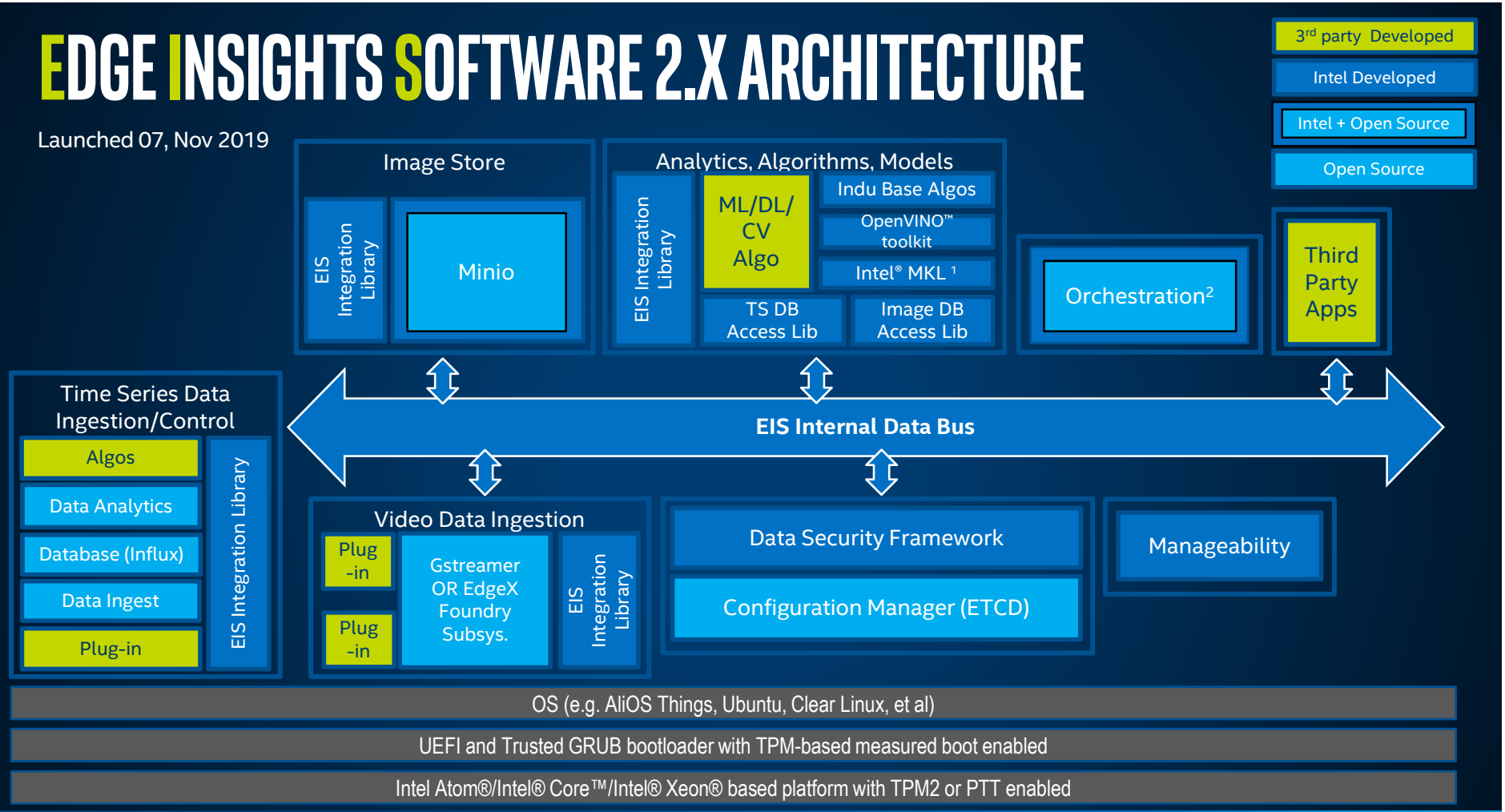
This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

Intel, the Intel logo, and other Intel are trademarks of Intel Corporation in the U.S. and/or other countries.

Other names and brands may be claimed as the property of others.

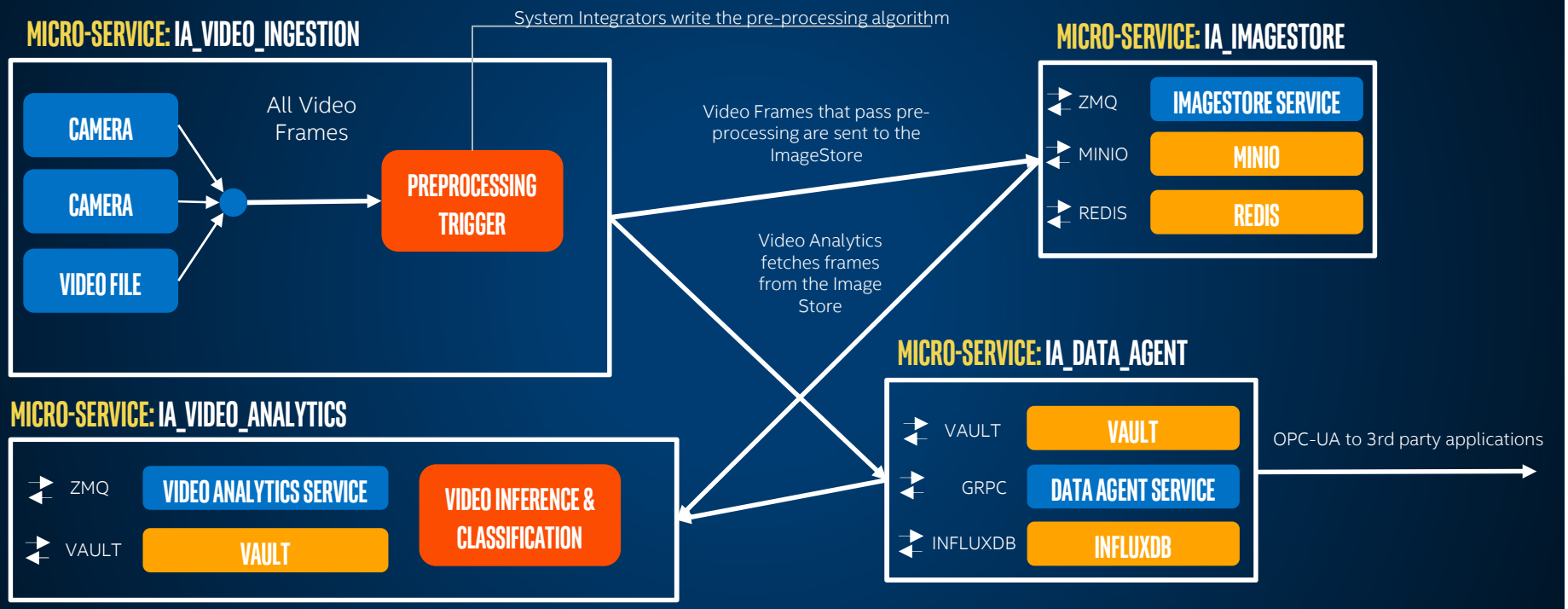
EDGE INSIGHTS SOFTWARE 2.X ARCHITECTURE

Launched 07, Nov 2019



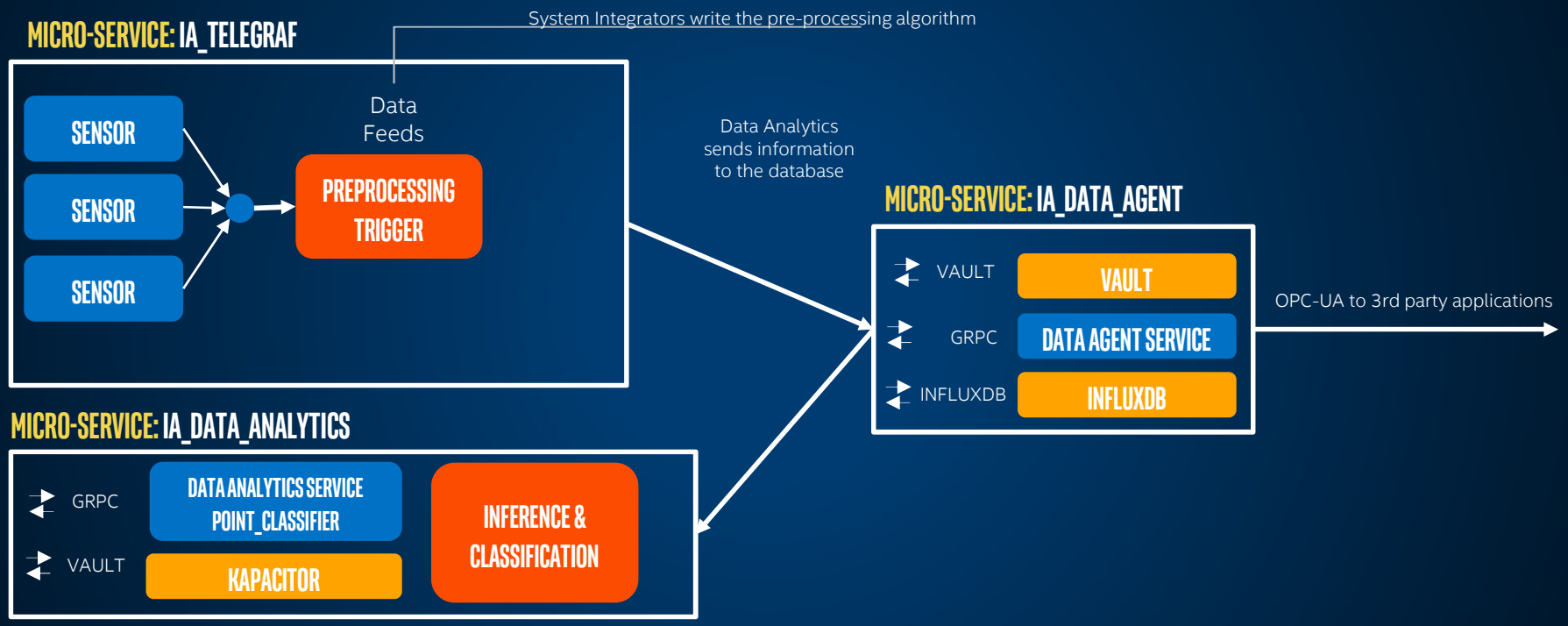
1. Intel® Math Kernel Library (Intel® MKL)
2. Targeted for a future Industrial Edge Insights software release

INTEL EDGE INSIGHTS SOFTWARE: VIDEO ANALYTICS



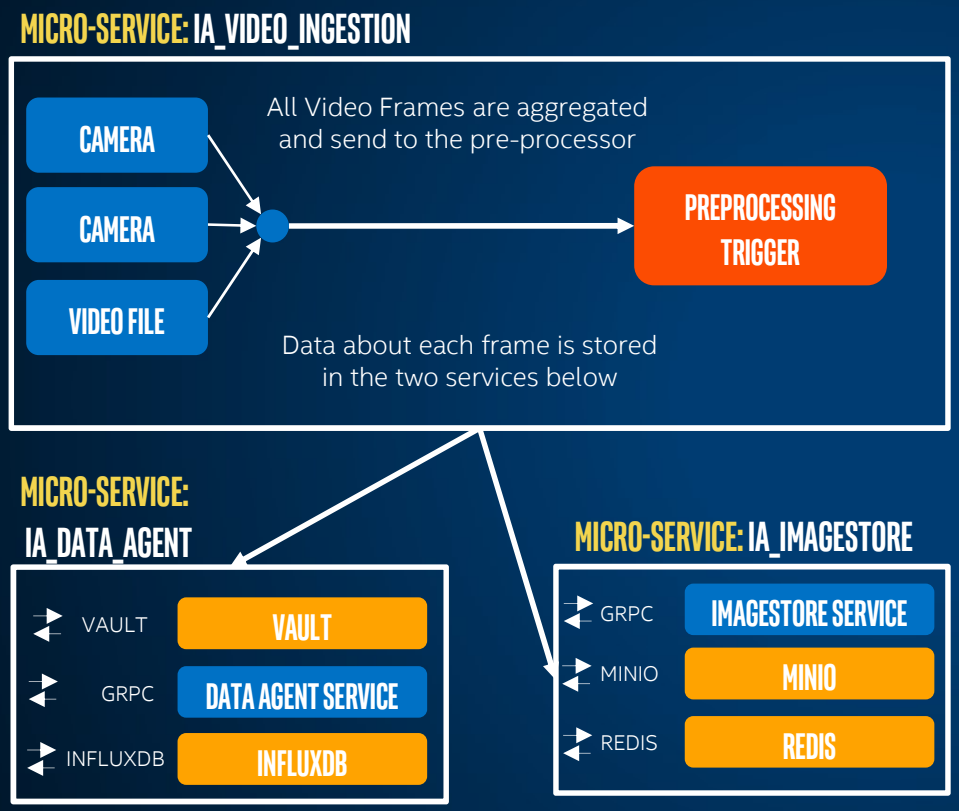
This diagram is base on EIS version 1.5

INTEL EDGE INSIGHTS SOFTWARE: TIME SERIES ANALYTICS



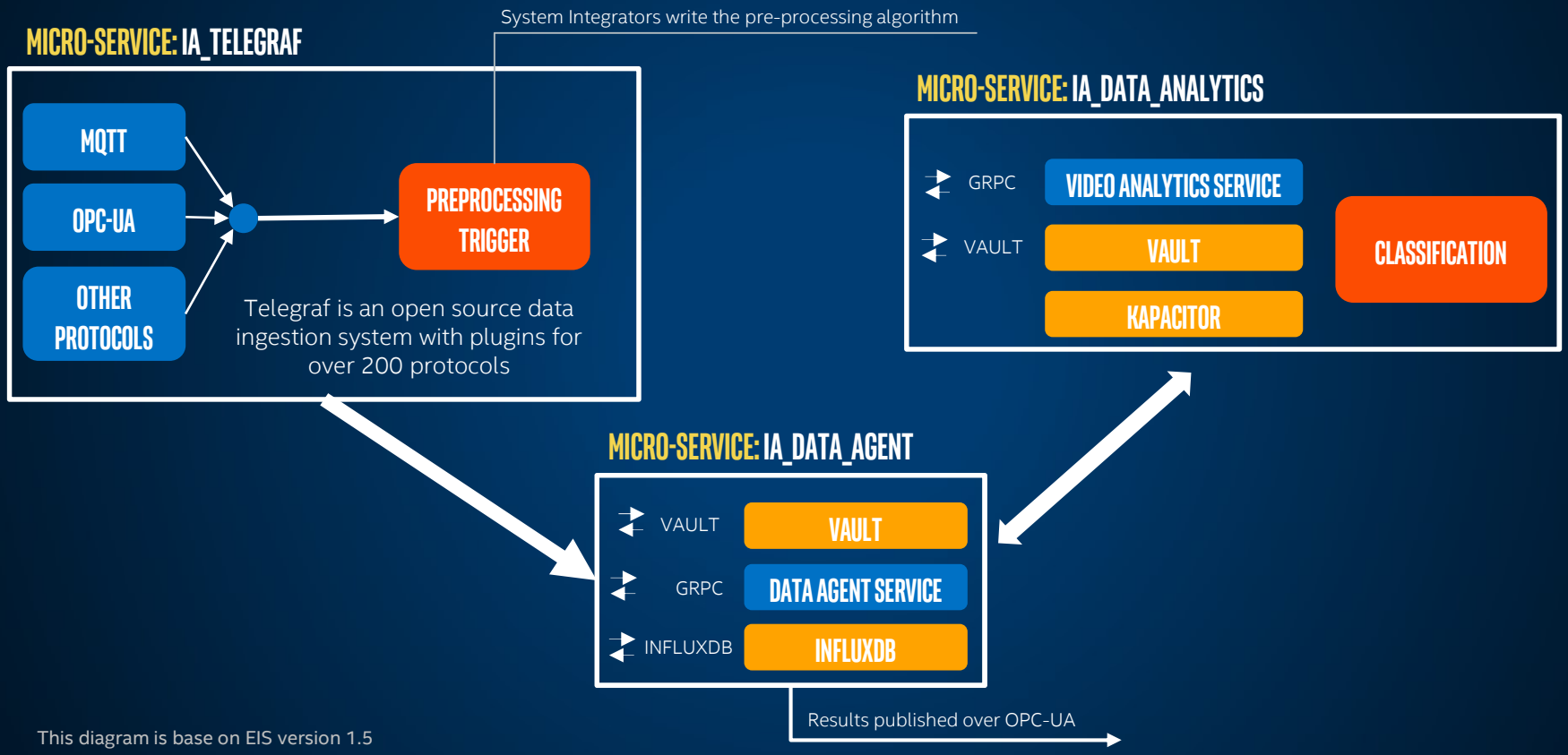
This diagram is base on EIS version 1.5

PIPELINE WALKTHROUGH: VIDEO INGESTION



1. Video inference is an expensive operation, particular when there are many video sources
2. The pre-processing step lets the application filter video frames that can be easily classified by with faster, simpler algorithms
3. The application developer creates a pre-processing function to filter incoming video frames.
4. Video frame that pass the pre-processing algorithm are stored in the Image Store and meta-data about the image is stored by the Data Agent

INTEL EDGE INSIGHTS SOFTWARE: DATA ANALYTICS

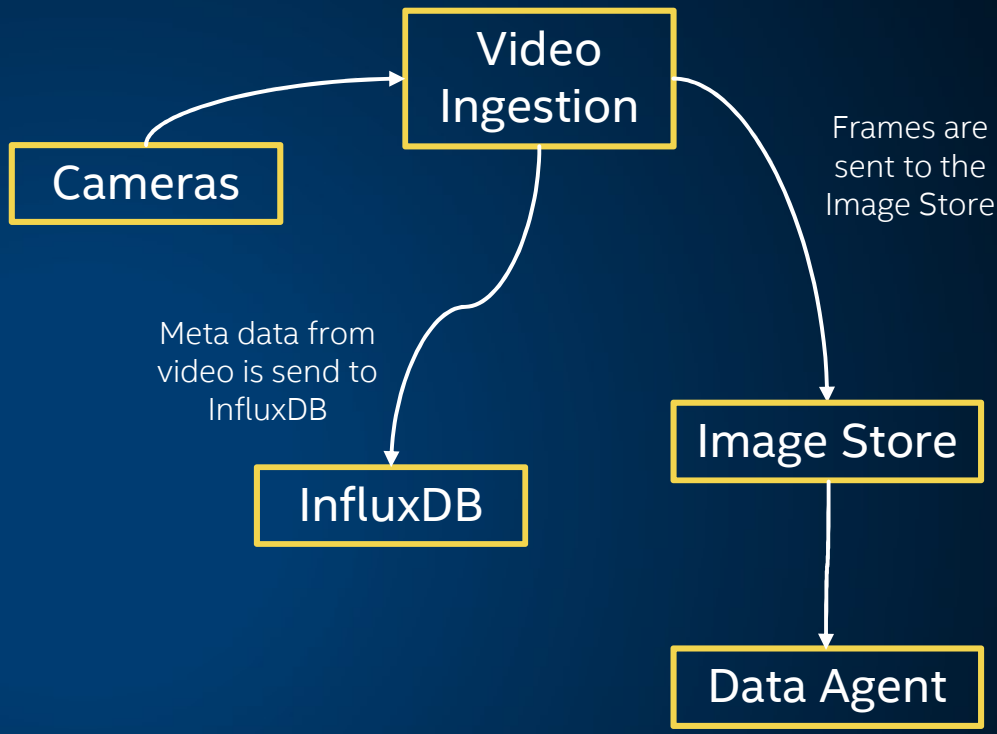


This diagram is base on EIS version 1.5

STEP 2: STREAM MANAGER

The Stream Manager subscribes to the stream results. The policy of stream export is set in the Stream Manager.

The data is published over OPC-UA to an external OPC-UA server and can be integrated into the existing environment and 3rd party applications.



MODIFY .ENV FILE

The .env file contain all the environmental variables that are needed to build the docker containers that host the Intel Edge Insights Software.

```
# ---------------------------------
COMPOSE_HTTP_TIMEOUT=200
COMPOSE_PROJECT_NAME=iei_docker_network

# ---------------------------------
# IEI configs
# ---------------------------------
NO_PROXY=localhost,127.0.0.1,ia_influxdb,ia_telegraf,ia_imagest
DATA_AGENT_GRPC_SERVER=localhost
IMAGESTORE_GRPC_SERVER=localhost
INFLUX_SERVER=localhost
KAPACITOR_SERVER=localhost
HOST_IP=localhost

# In case multiple RTSP cameras are used, update the value for
RTSP_CAMERA_IP=

# Refer the below link for available logging level for Gstreame
# https://gstreamer.freedesktop.org/documentation/tutorials/bas
GST_DEBUG=1

# Docker security
IEI_USER_NAME=ieiuser
IEI_UID=5317

# IEI_SERVICES allows to selectively build and run the requirec
IEI_SERVICES=services_pointdata.json

# CONFIG_FILE has all the required information for video usecas
# classification algos configuration
CONFIG_FILE=factory_pcbdemo.json

# This is the path where IEI package is installed
IEI_INSTALL_PATH=/opt/intel/iei

# DEV_MODE if set `true` allows one to run IEI in non-secure mc
DEV_MODE=true

# PROFILING if set to `true` captures IEI's performance/benchma
PROFILING=false
```

IEI SERVICES – MICROSERVICE ARCHITECTURE

IEI_SERVICES should point to a JSON file that enumerates the EIS services

Video Analytics Services:

- ia_video_ingestion
- Ia_imagestore
- ia_video_analytics

Time Series Services:

- ia_telegraf
- ia_data_analytics

```
{
    "iei_services": [
        {
            "name": "ia_video_ingestion",
            "dockerignore": "VideoIngestion/.dockerignore"
        },
        {
            "name": "ia_imagestore",
            "dockerignore": "ImageStore/.dockerignore"
        },
        {
            "name": "ia_video_analytics",
            "dockerignore": "DataAnalytics/VideoAnalytics/.dockerignore"
        }
    ]
}
```

STEP 1A: VIDEO INGESTION

This module ingests video frames from a video source like video file or basler/RTSP/USB camera using gstreamer pipeline and publishes the (metadata, frame) tuple to the data bus.

RESPONSIBILITIES OF VIDEO INGESTOR

- The Video Ingestion module starts capturing frames from Basler camera / RTSP camera (or video file) and sends the frames to the Trigger Algorithm.
- Uses Trigger Functions to select Key Frames
- Queues Key Frames to be published on the Data Bus
- Triggers are defined by the application developer

./docker_setup/config/algo_config/factory_pcbdemo.json

```
{
        "ingestor": {
            "video_src": "./test_videos/pcb_d2000.avi",
            "encoding": {
                "type": "jpg",
                "level": 100
            },
            "loop_video": "true"
        },
        "filter": {
            "name": "pcb_filter",
            "queue_size": 10,
            "max_workers": 5,
            "training_mode": "false",
            "n_total_px": 300000,
            "n_left_px": 1000,
            "n_right_px": 1000
        }
}
```

CAPTURING VIDEO FROM CAMERA OR FILE SOURCES

Step-1.a: The Video Ingestion module starts capturing frames from Basler camera / RTSP camera (or video file) and sends the frames to the Trigger Algorithm.

```
"data_ingestion_manager": {
    "ingestors": {
        "video": {
            "vi_queue_size":5,
            "streams": {
                "capture_streams": {
                    "cam_serial1": {
                        "video_src": "rtspsrc location=\"rtsp://localhost:8554/\" latency=100 ! rtph264depay ! h264parse ! mfxdecode ! videoconvert ! appsink",
                        "encoding": {
                            "type": "jpg",
                            "level": 100
                        },
                        "img_store_type": "inmemory_persistent",
                        "poll_interval": 0.2
                    },
                    "cam_serial2": {
                        "video_src": "rtspsrc location=\"rtsp://localhost:8554/\" latency=100 ! rtph264depay ! h264parse ! mfxdecode ! videoconvert ! appsink",
                        "encoding": {
                            "type": "jpg",
                            "level": 100
                        },
                        "img_store_type": "inmemory_persistent",
                        "poll_interval": 0.2
                    },
                    "cam_serial3": {
                        "video_src": "rtspsrc location=\"rtsp://localhost:8554/\" latency=100 ! rtph264depay ! h264parse ! mfxdecode ! videoconvert ! appsink",
                        "encoding": {
                            "type": "jpg",
                            "level": 100
                        },
                        "img_store_type": "inmemory_persistent",
                        "poll_interval": 0.2
                    },
```

STEP 1B: POINT DATA INGESTION

Point Data mode allows the ingestion of time series based like sensor readings that have a time stamp

SERVICES IN POINT DATA MODE

- Telegraf is used to read multiple data formats and store them in InfluxDB
- Uses Trigger Functions to select Key Frames
- Queues Key Frames to be published on the Data Bus
- Triggers are defined by the application developer

./docker_setup/config/algo_config/factory_pcbdemo.json

```
{
        "ingestor": {
            "video_src": "./test_videos/pcb_d2000.avi",
            "encoding": {
                "type": "jpg",
                "level": 100
            },
            "loop_video": "true"
        },
        "filter": {
            "name": "pcb_filter",
            "queue_size": 10,
            "max_workers": 5,
            "training_mode": "false",
            "n_total_px": 300000,
            "n_left_px": 1000,
            "n_right_px": 1000
        }
}
```

TRIGGER ALGORITHM

- The Trigger Algorithm will determine the relevant frames that are to go to the Classifier.
- To create a custom trigger
  - Create a class derived from BaseTrigger
  - Override the constructor, __init__()
  - Enumerate the supported ingestors
  - Override the on_data() function, which is called when for each video frame.
  - Use OpenVINO or other libraries or methods to analyze the video and decide whether to pass it to the classifier or not.

```python
import logging
import cv2
import numpy as np
from . import BaseTrigger


class Trigger(BaseTrigger):
    """Bypass trigger to send all the frames without any trigger logic to
    select key frames.
    """

    def __init__(self, training_mode):
        """Constructor.
        """
        super(Trigger, self).__init__()
        self.log = logging.getLogger(__name__)
        self.training_mode = training_mode
        self.count = 0
        self.startSignal = True

    def get_supported_ingestors(self):
        return ['video', 'video_file']

    def on_data(self, ingestor, data):
        """Process video frames as they are received and call the callback
        registered by the `register_trigger_callback()` method if the frame
        should trigger the execution of the classifier.

        Parameters
        ----------
        ingestor : str
            String name of the ingestor which received the data
        data : tuple
            Tuple of (camera serial number, camera frame)
        """
        if self.training_mode is True:
            self.send_start_signal(data, -1)
            cv2.imwrite("./frames/"+str(self.count)+".png", data[1])
        else:
            # Send trigger start signal and send frame to classifier
            if self.startSignal:
                self.send_start_signal(data, -1)
                self.startSignal = False
            # Sending Frames to Store
            self.log.debug("Sending frame")
            self.send_data(data, 1)
```

KAPACITOR

- Kapacitor subscribes to the Meta Data stream. All the streams in InfluxDB are subscribed as default by Kapacitor.
- Kapacitor is configured by TICKscript
- Kapacitor is an alert and monitoring system

- Intel Edge Insights Software uses Kapacitor to subscribe to data coming into InfluxDB, passing it to a Classifier and then writing the results back to Influxdb.
- This workflow is defined in a TICK script. Users and define their own TICK scripts.

The following handlers are currently supported:

Alerta: Sending alerts to Alerta.

Email: To send alerts by email.

HipChat: Sending alerts to the HipChat service.

Kafka: Sending alerts to an Apache Kafka cluster.

MQTT: Publishing alerts to an MQTT broker.

OpsGenie: Sending alerts to the OpsGenie service.

PagerDuty: Sending alerts to the PagerDuty service.

Pushover: Sending alerts to the Pushover service.

Sensu: Sending alerts to Sensu.

Slack: Sending alerts to Slack.

SNMP Trap: Posting to SNMP traps.

Talk: Sending alerts to the Talk service.

Telegram: Sending alerts to Telegram.

VictorOps: Sending alerts to the VictorOps service.

CLASSIFIER

The Classifier UDF (User Defined Function) receives the Meta Data from Kapacitor. It then invokes the UDF classifier algorithm.

The Classifier Algorithm is defined by the application developer.

OpenVINO is used for deep learning based visual inspection.

TICK Script

1. Defines the subscription of Kapacitor to InfluxDB
2. Kapacitor runs the classifier for each frame
3. Frame Classification results calculated
4. Kapacitor writes the results back to influxdb

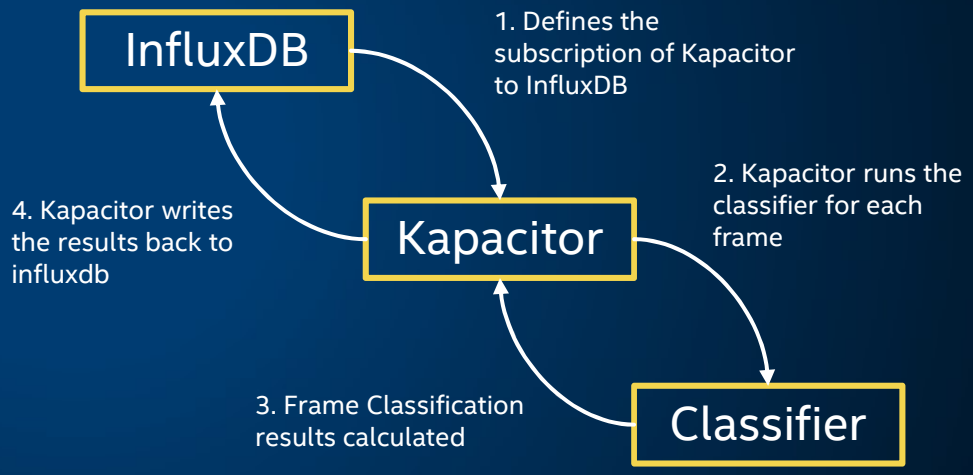


IMAGE STORE AND PUBLISHERS

IMAGE STORE

Redis - in-memory data structure store, used as a database, cache and message broker

Minio - Open Source, Enterprise-Grade, Amazon S3 Compatible Object Storage

OPC-UA AND REST PUBLISHERS

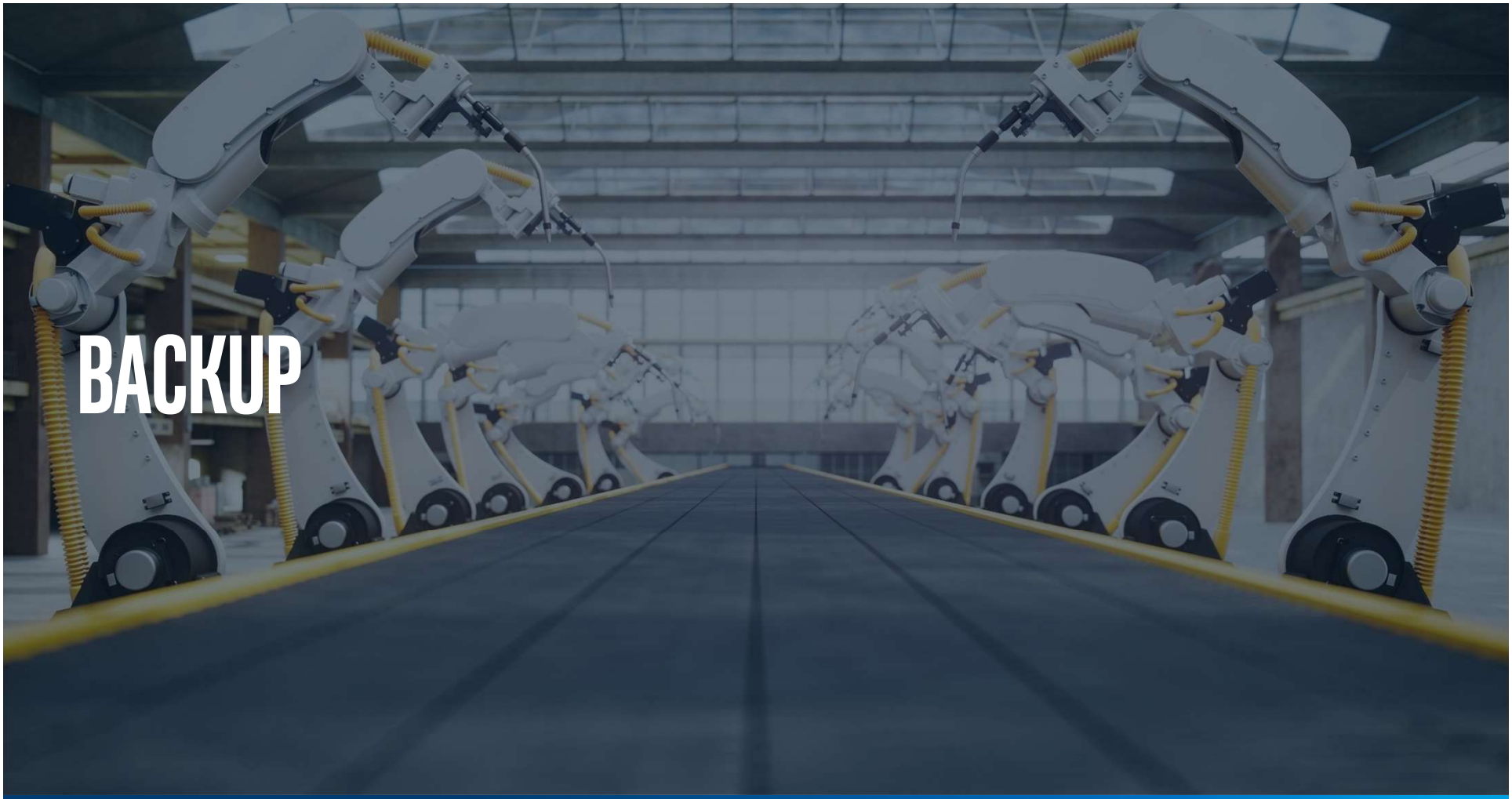
Responsible for allowing data access to the EIS Data Bus through industry standard protocols

VISUALIZE THE DATA WITH CHRONOGRAF

Chronograf lets you visualize the results of a query to the time series database.

Measurements can be grouped by intervals, single point data, or data over time.


intel®


BACKUP

INTEL'S INDUSTRIAL EDGE SOFTWARE FEATURES

VERSION 1.0

Silicon Support

- Intel® Xeon® SP processor (Skylake)
- Intel® Xeon® D processor (Broadwell)
- Intel® Core™ i5/ i7/ i9 processor
- FPGA, Movidius VPUs, SSDs

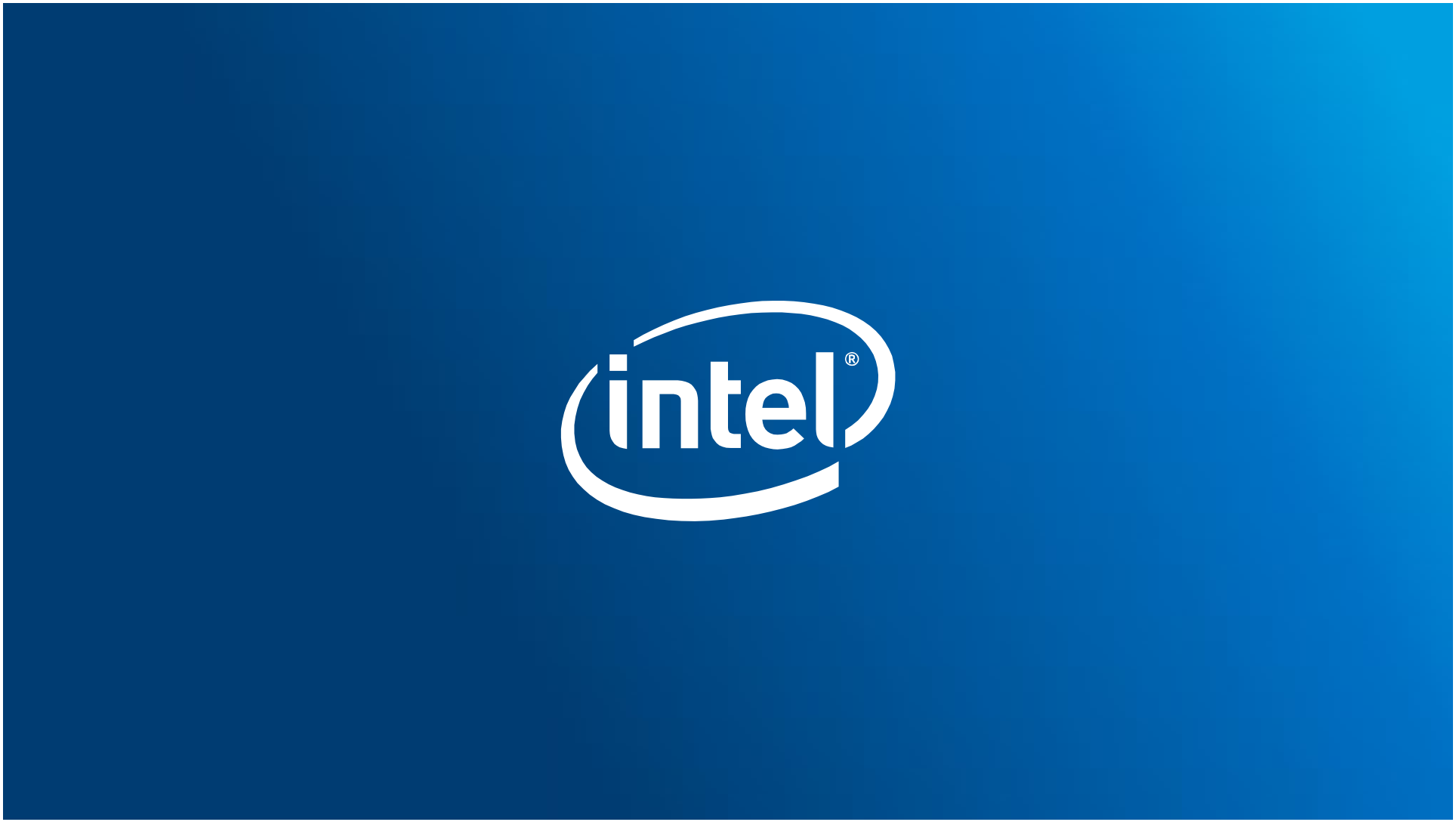
Key Features

- Time series data ingestion, storage and data analytics
- Video data ingestion (Basler GigE Vision* & RTSP) and storage (Redis* & MinIO*)
- Hardware-accelerated media encode/decode (Intel® Media SDK)
- IA Optimized computer vision inference (OpenVINO™ toolkit)
- Available for use with full suite of PCIe*-based Intel accelerator cards (FPGA, VPU)
- Security enabled (SAFE ratified architecture)

VERSION 1.5

Key Features

- Validated on Ubuntu* 18.04
- Camera support: Basler* with Gstreamer pipeline, USB camera
- Support for multiple video ingestion streams with improved build time (up to 180 fps WLs)
- ImageStore* optimization for persistent store
- Video analytics support without Kapacitor*
- Machine data use cases with Telegraf*, Kapacitor-based UDF and Data Agent
- Simple visualizer
- OPC-UA pub-sub speed optimization (for machine data)
- Low memory footprint support (max 2GB RAM use cases)
- Support for Docker* registry
- UX/DX improvements (dev mode, selective container builds)


intel