Edgility

Serverless Edge Orchestration

Domain 2.0 TLV

Where passion meets technology
AT&T ISRAEL

AT&T

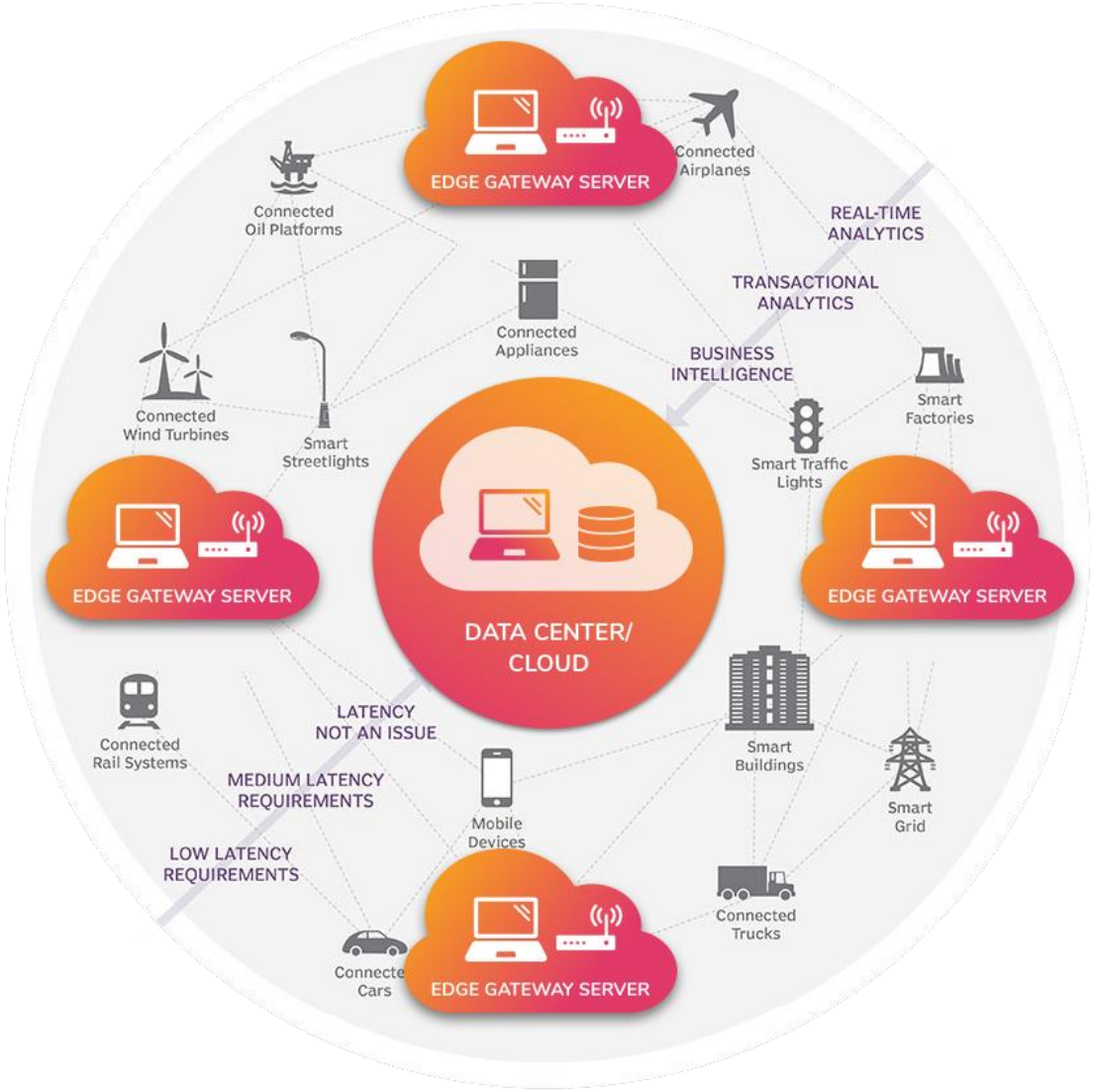
About me

Eden Rozin

CPO, Product Management
AT&T Tel-Aviv, Israel

Edge Computing

- Key enabler for 5G
- Decentralized architecture
- Latency issue mitigation
- Essential for IOT



However…

- Distributed across 1000s of locations
- Limited space & real estate
- Limited cooling and power
- Scarce computing resources
- Significant workload support
- Runs 3rd party software

Orchestration on Edge vs Central Cloud Challenges

	Edge	Central Cloud
Location of application components	Location of nodes plays significant role in application blueprint	Pretty much location-agnostic
Mobility of workloads	Workload transition from one node to the other	Static unless there is a cloud node failure
Workload dynamic	Various applications need to run at various times to serve different needs	Static workload most of the time. Once you deploy a service, it is there forever
Architecture heterogeneity	Edge is made of different nodes, various sizes, vendors and technologies. Large, small, PNFs, Akraino, Green Grass, Azure Edge, etc.	Mostly homogeneous. If it is Openstack, AWS or Azure, it is the same Cloud OS for all nodes, and diversity is considerably small
Latency	Latency and distance from the end consumer plays a major role	Most central cloud apps are not latency-sensitive
Availability of resources	Edge nodes are small; availability of resources for application is not guaranteed	Availability of resources is pretty much guaranteed. This is one of the basic principles of any cloud

Monitoring on Edge vs Central Cloud Challenges

	Edge	Central Cloud
Distributed data collection	Collection needs to be done from thousands of distributed nodes across the network	Everything is centralized and collected to a central DB
Architecture heterogeneity	Edge is made of different vendors; each has its own metrics and APIs	Each cloud vendor has its own collection and monitoring framework (OS Ceilometer, AWS CloudWatch, etc.)
Distributed root cause analysis	Identification of the root cause and its impact on the service in distributed environment	Although it's complicated, it's still simpler than doing it on the edge network
Distributed closed loop	Location and latency take major role in recovery, mitigation plan	Recovery is much simpler. Most of the time it's to spin up another instance

Data Management on Edge vs Central Cloud Challenges

	Edge	Central Cloud
Supporting ACID (transactions)	Distribution and partition of the edge is a challenge for every transactional DB	Everything is in one place; just install SQL DB
High availability of DB	Replication of DB is not practical in most cases	No problem having any H/A solution on central cloud
Latency	Latency requirements prevent using a DB on central cloud; DB needs to be local to the apps	Apps are close to the DB in central cloud, no latency issues
Mobility/Availability of data on the edge nodes	The environment is dynamic so all data needs to be available to all nodes although it is distributed	No such issue in central cloud

Edge Operating System Manifesto

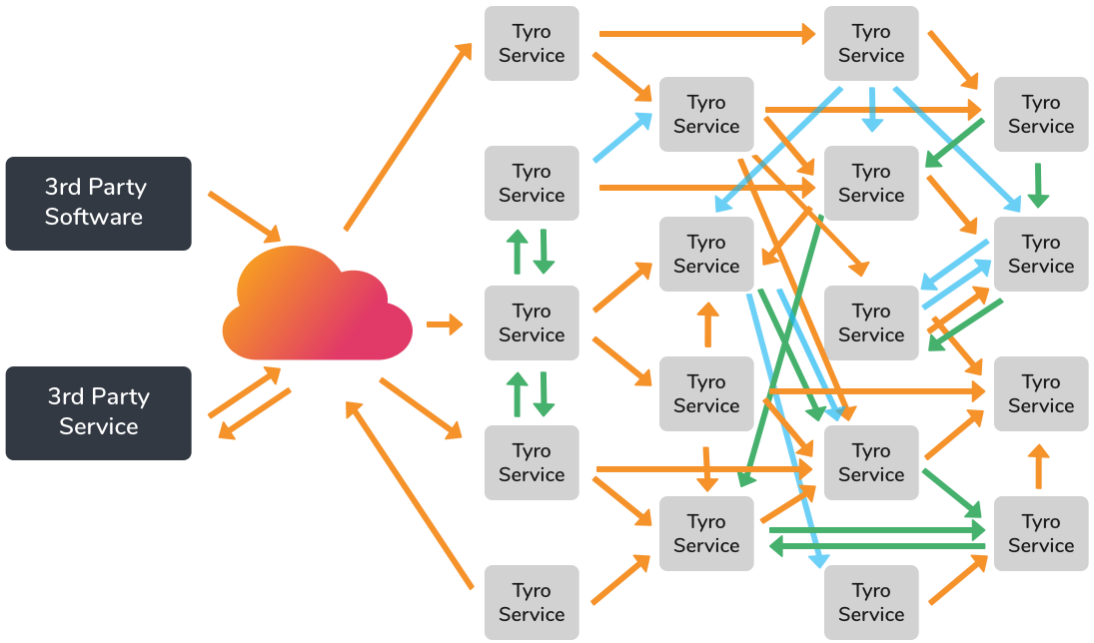
- Treat the Edge as one big distributed compute
- Harness distribution for availability and reliability
- Data is available anywhere on the Edge network
- Execute workload anywhere on the Edge network
- Intelligent resource management
- Location-sensitive workload orchestration
- Expand application beyond Edge boundaries (Public Cloud, DC, etc.)
- No single point of failure

Akraino Edge Stack

- The industry adopted cloud native for edge
- Containers have smaller footprint than VMs
- Improved resource utilization
- Micro-services architecture

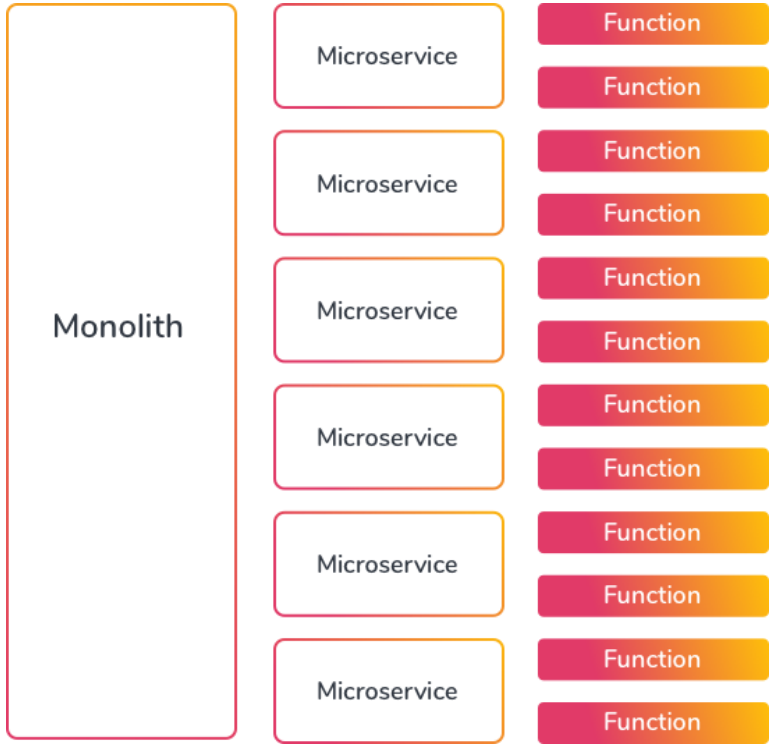
However…

- Integration of new micro-service is complicated
- Permanent allocation of resources
- Container is still larger execution unit

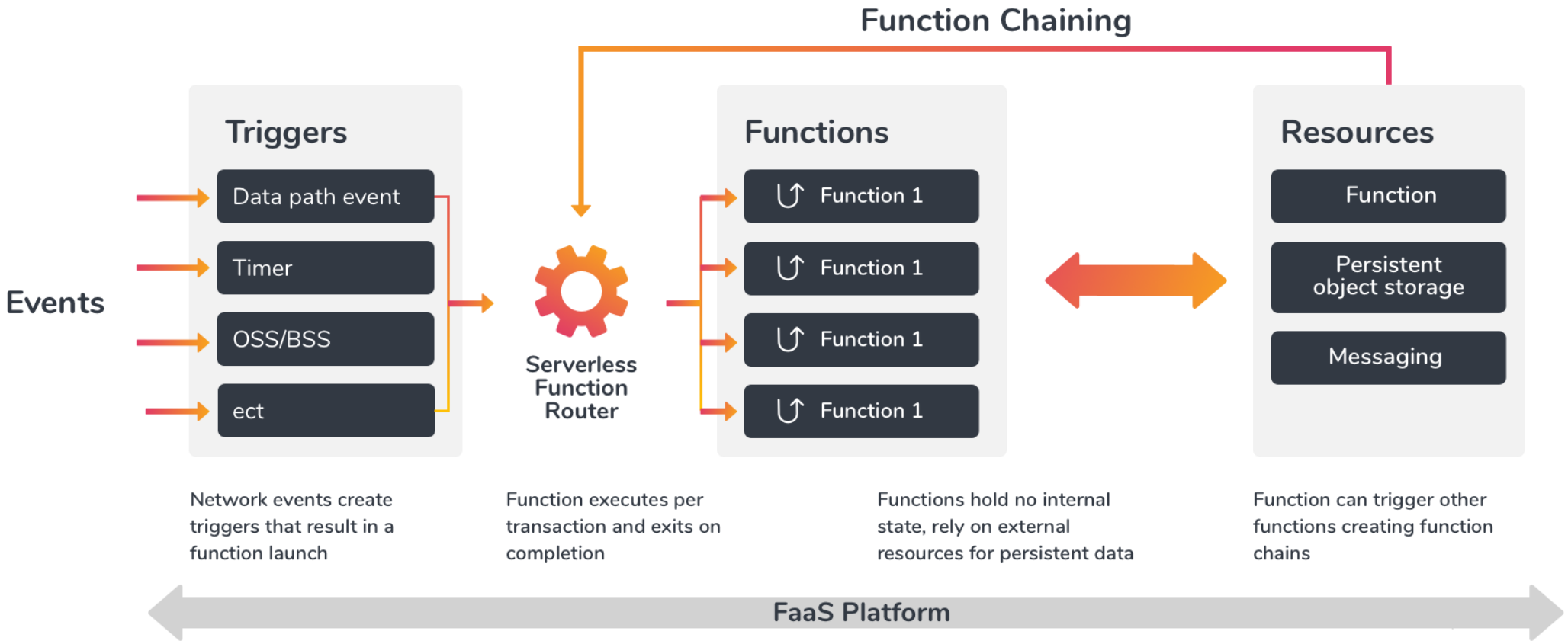


Introducing Serverless
FaaS (Function-as-a-Service)

- Functions are the unit of deployment and scaling
- No machines, VMs, or containers visible in the programming model
- Permanent storage lives elsewhere (SLE)
- Scales per request; Users cannot over- or under-provision capacity
- Never pay for idle (no cold servers/containers or their costs)
- Implicitly fault-tolerant because functions can run anywhere
- Bring Your Own Code (BYOC)
- Metrics and logging are a universal right



Functions in a Nutshell



What is Serverless good for?

GOOD

- Data collection & enrichment
- Mobile backend (Control Plane)
- File processing
- Web backend
- IoT Backend
- Stream processing

NOT GOOD

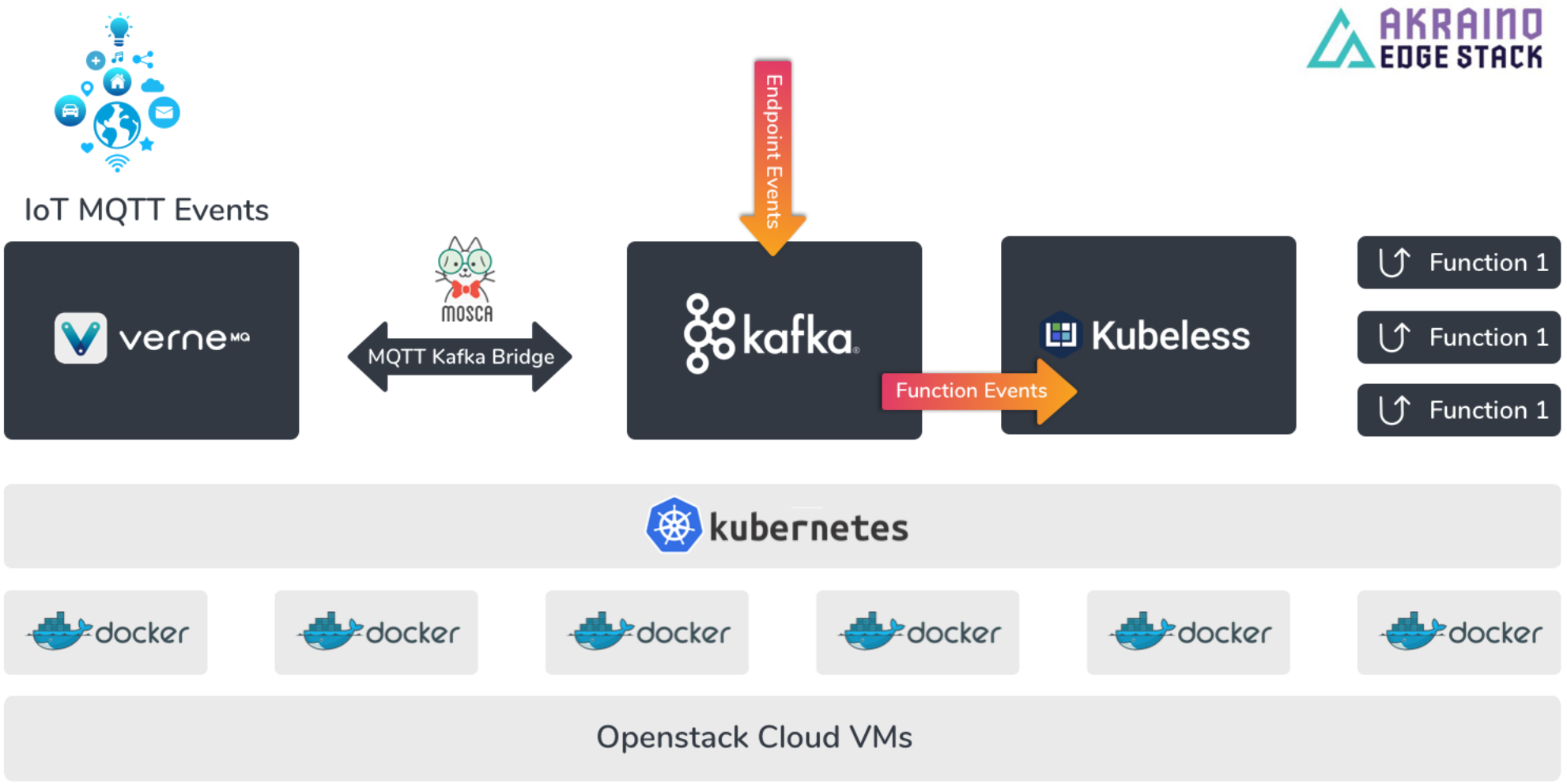
- Long running persistent processes
- Network traffic processing (routers, gateways, firewalls)
- Databases

What did we build?

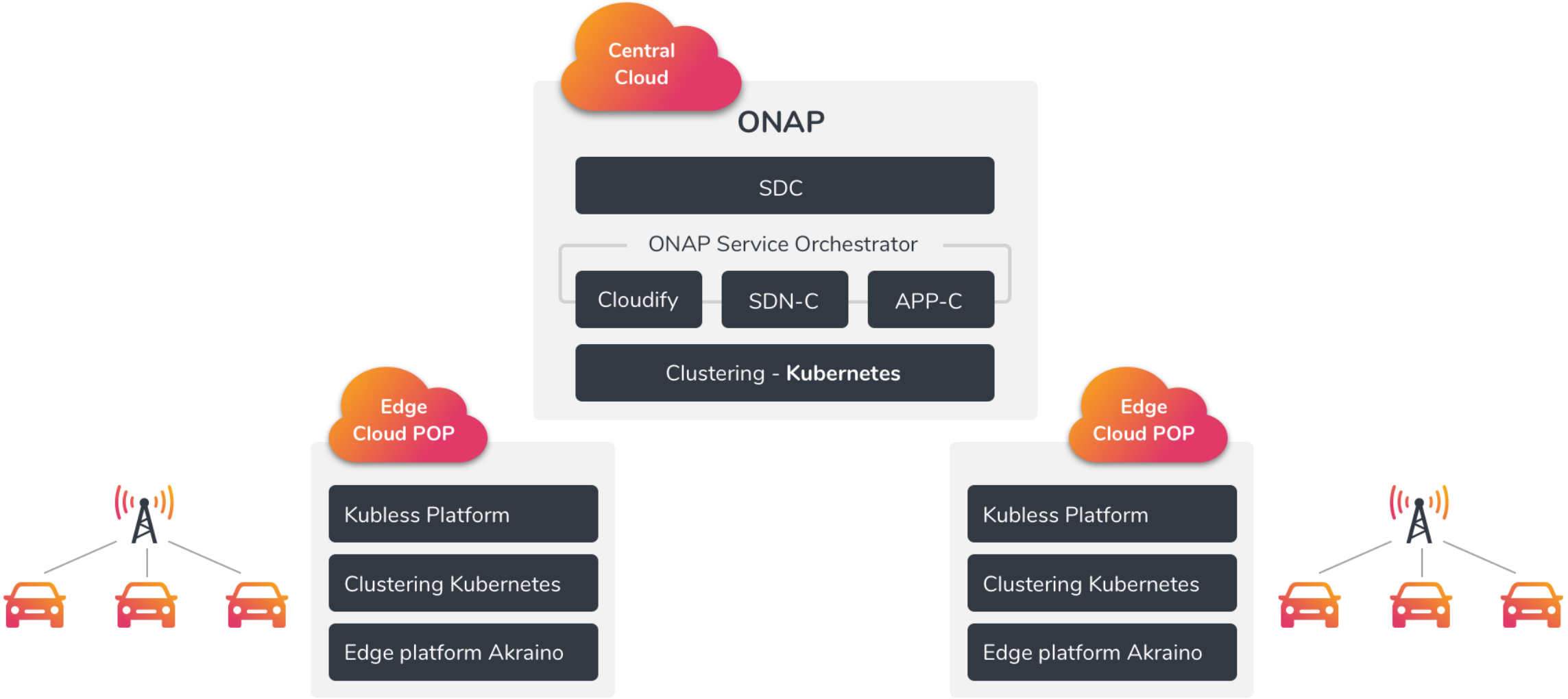
Edgility: Serverless Edge

An open-source attempt to marry serverless with edge, in order to optimize resource management by intelligent orchestration.

Akraino Based Serverless Edge Node with IoT Gateway



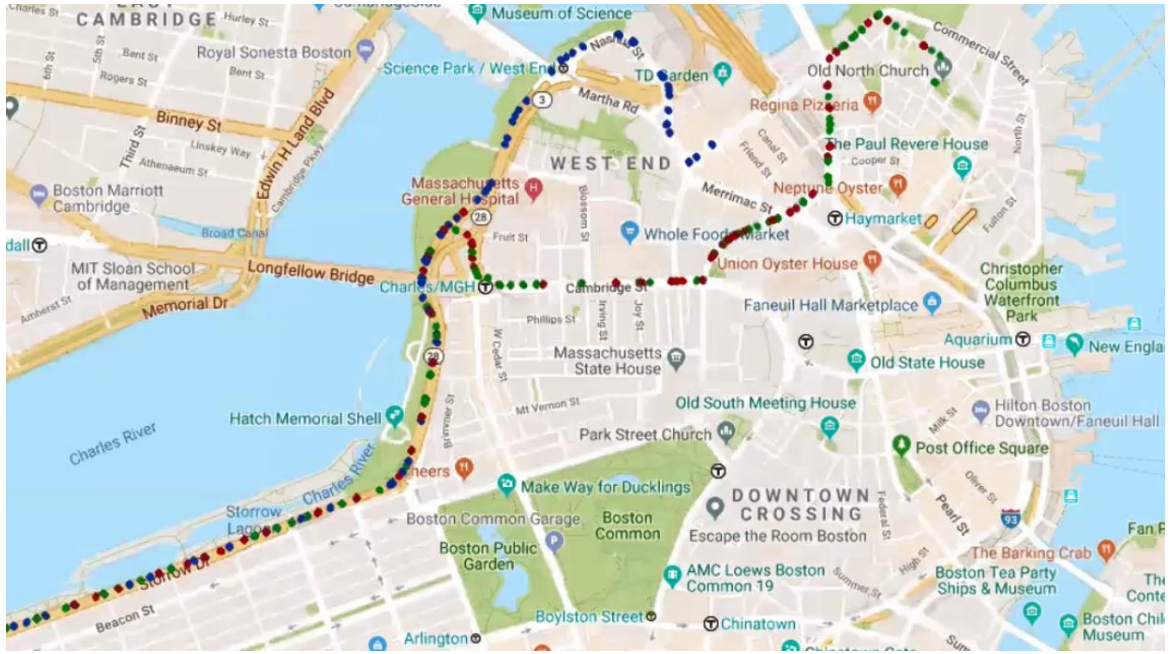
ONAP SDC, SO Orchestration and Monitoring Infrastructure



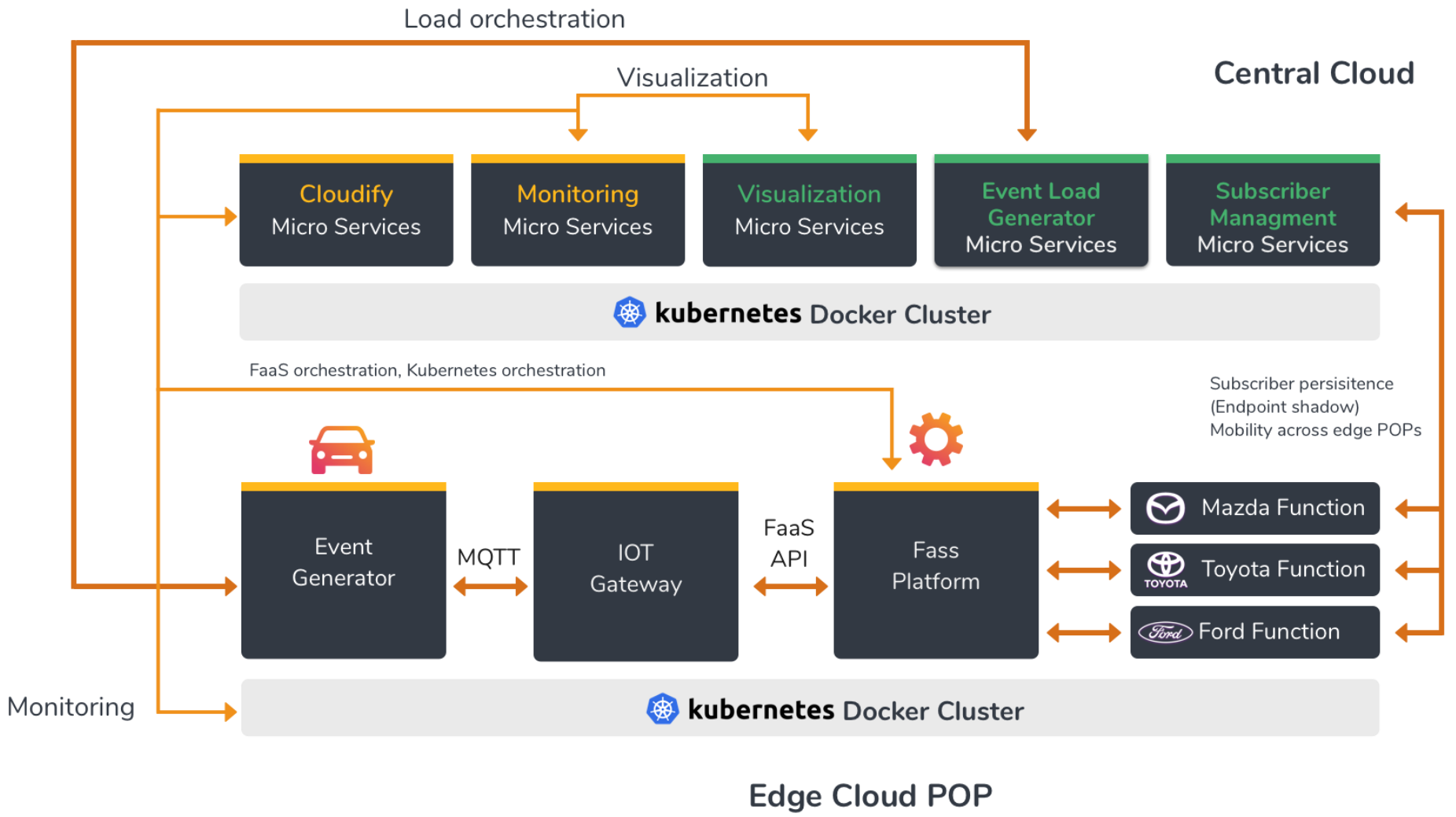
Intelligent Transport System (ITS)

Congestion avoidance system
Re-route connected cars to alternative routes:

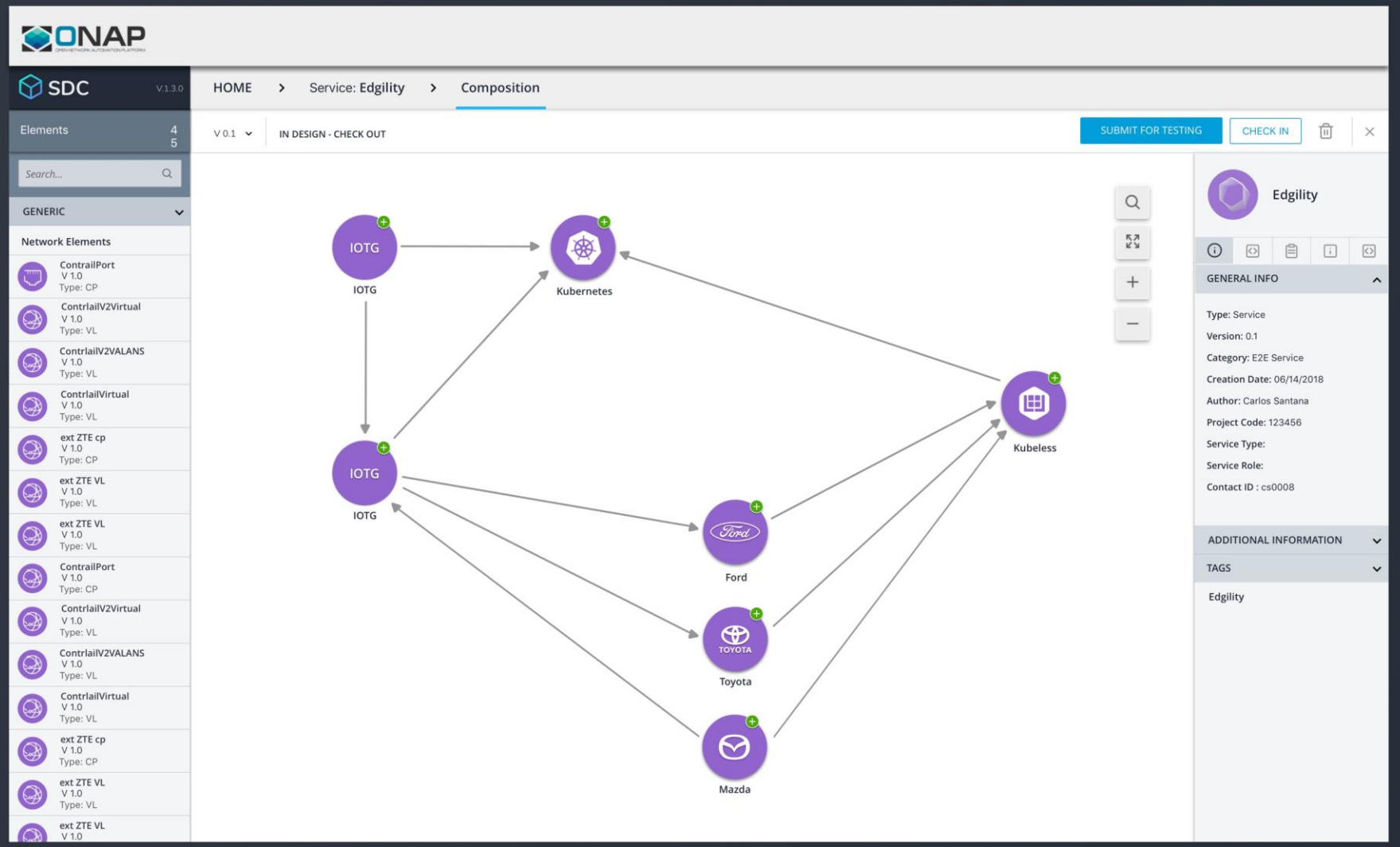
1. Function deployment for each car vendor
2. Function mobility
3. Manual scale-out to accommodate load



Detailed Demo Architecture

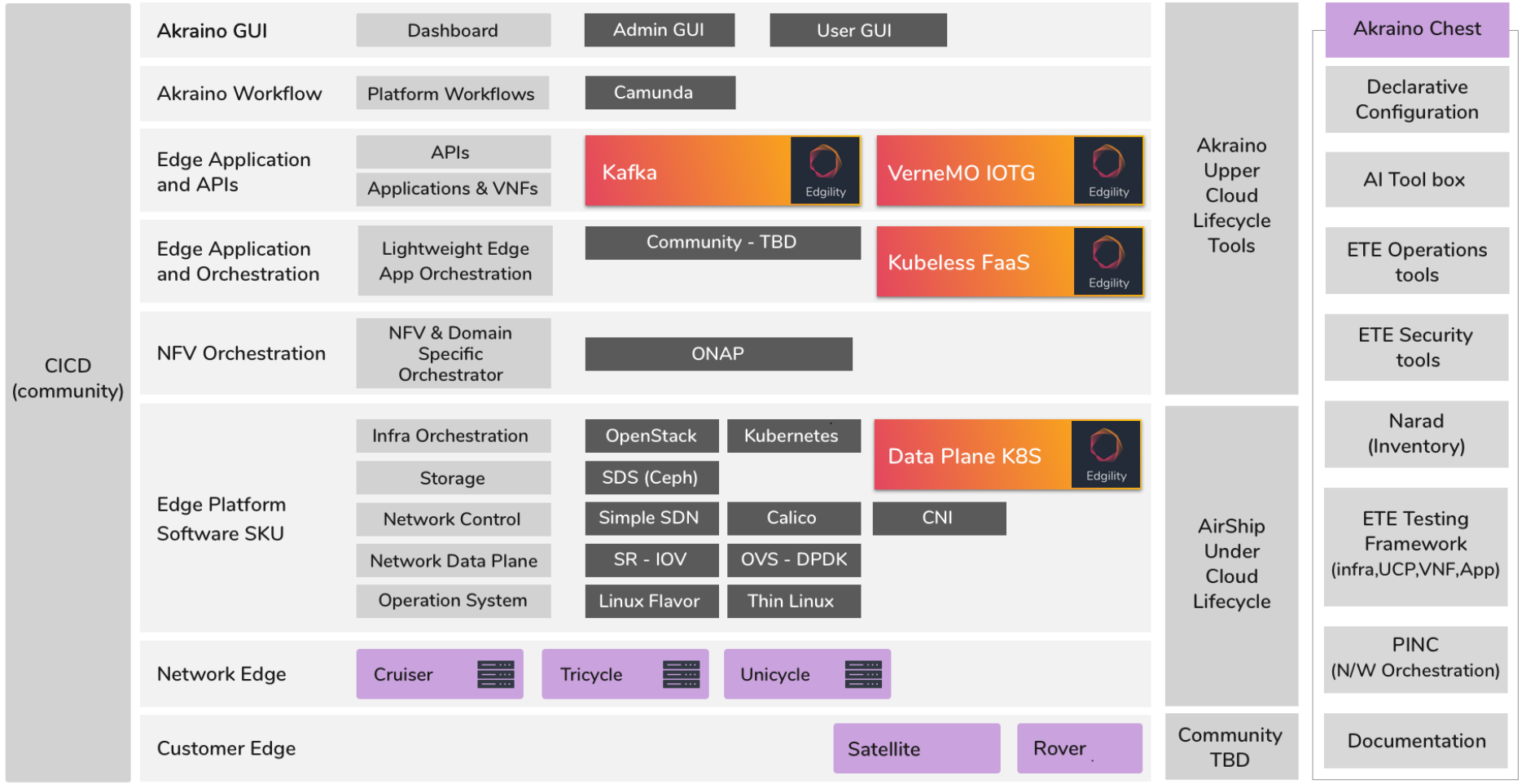


Modeling the Serverless Edge Stack using ONAP SDC



Edgility Code Contribution to Akraino

Akraino Building Blocks



AKRAINO EDGE STACK

Next Steps

Next steps

We are about to start the development of the 2nd phase of **Edgility**: “Dynamicity of Function powered by ML”.

We welcome anyone who would like to join our journey!

Edgility

Thank You!

Contact Details: er434w@att.com