Presentazione FLUIDOS

Daniele Cacciabue

What is FLUIDOS

- FLUIDOS (Flexible, scaLable, secUre, and decentralIseD Operating System) aims to leverage the enormous, unused processing capacity at the edge, scattered across heterogeneous edge devices that struggle to integrate with each other and to coherently form a seamless computing continuum.

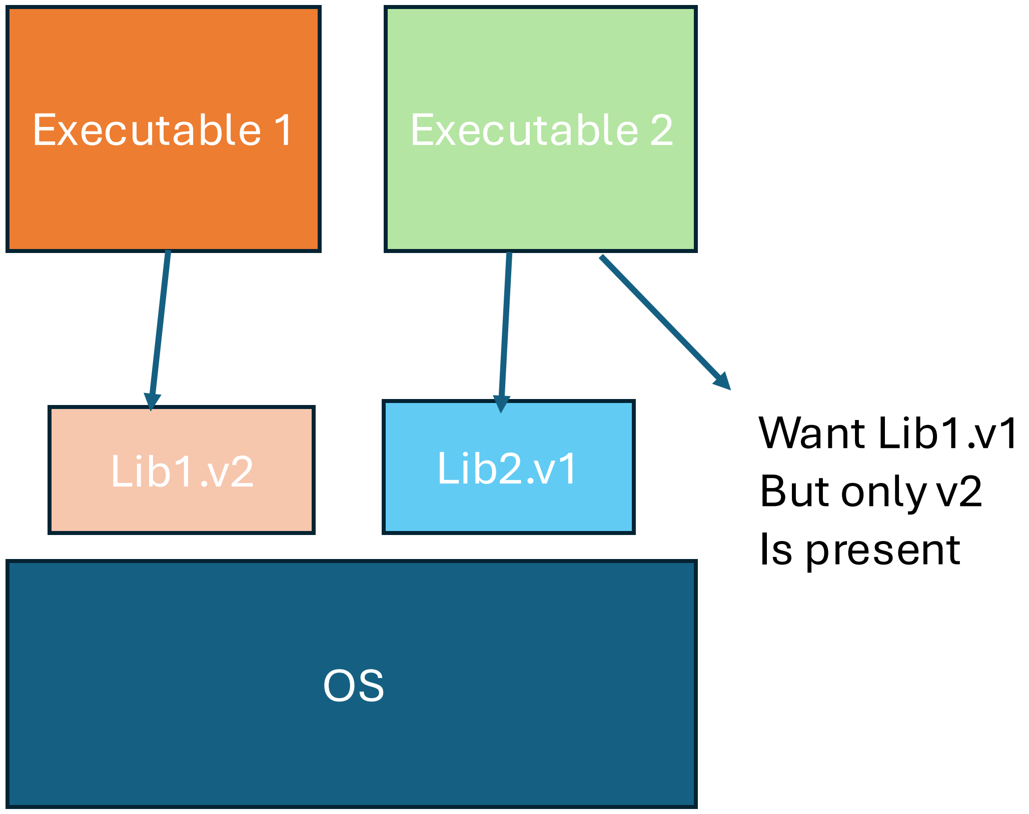
Underlying Technologies

docker

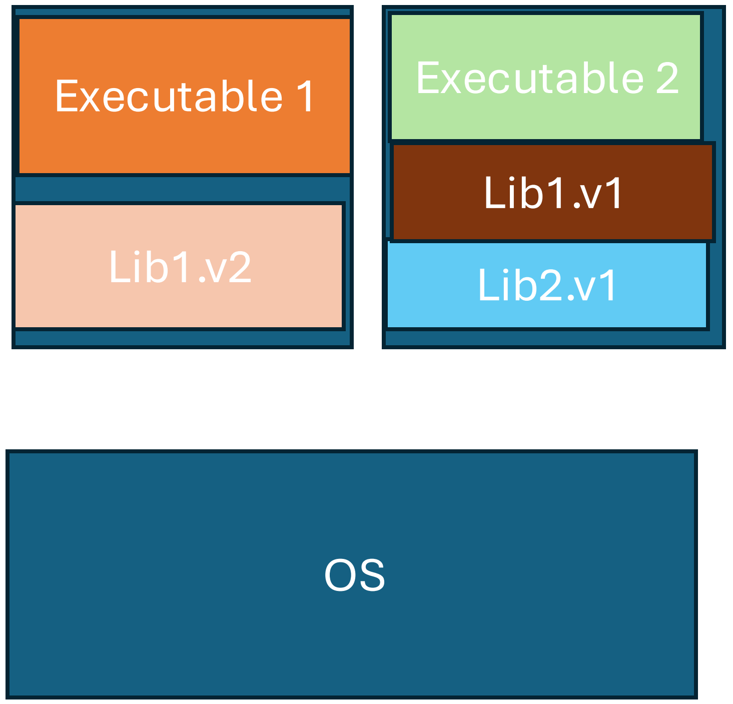
Containerization

- A lightweight form of virtualization that packages an application and its dependencies into a single, isolated unit called a container.
- Benefits:
  - **Portability**: Containers can run consistently across different environments, from development to production.
  - **Efficiency**: Containers share the host OS kernel, making them more resource-efficient than traditional virtual machines.
  - **Scalability**: Easily scale applications up or down by adding or removing containers.
  - **Isolation**: Each container operates independently, ensuring that issues in one container do not affect others.
  - **Rapid Deployment**: Containers can be quickly started, stopped, and replicated, speeding up the deployment process.

Before and After Containerization



Without containerization
The host system needs all dependencies installed

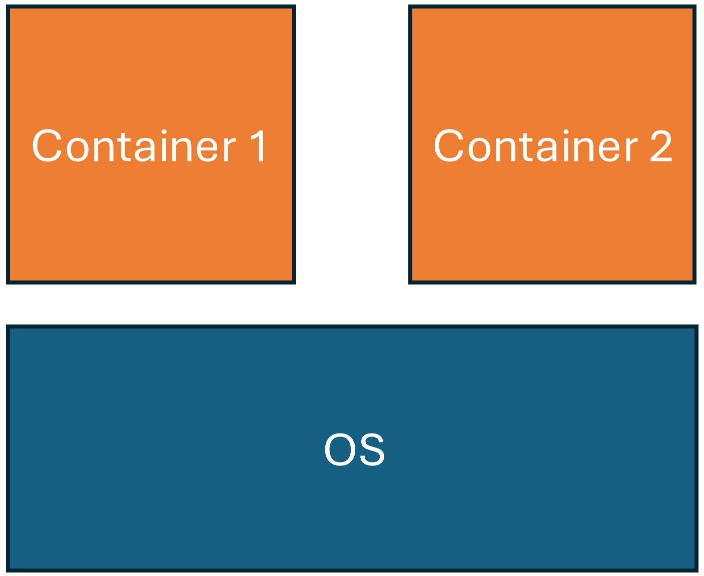


With containerization
Each container contains its dependencies.

Kubernetes

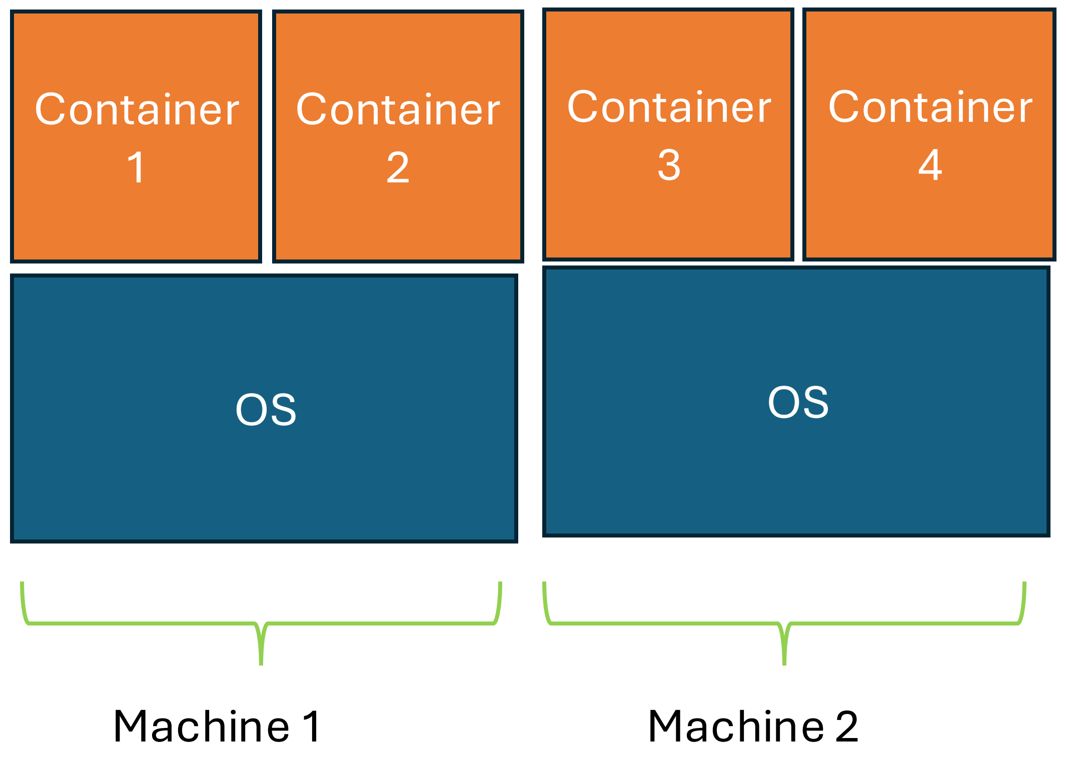
- Open-source platform for automating deployment, scaling, and operation of application containers.
- Key Features:
  - Automated Deployment and Scaling: Automatically deploys and scales applications.
  - Self-Healing: Restarts failed containers and replaces them.
  - Service Discovery and Load Balancing: Exposes containers using DNS names or IP addresses.
  - Storage Orchestration: Automatically mounts storage systems.
  - Secret and Configuration Management: Manages sensitive information and configuration.

Before K8s



- Who restarts containers?
- How do containers find each other?
- How to scale to multiple machines?
- What if a container depends on another container?

After K8s



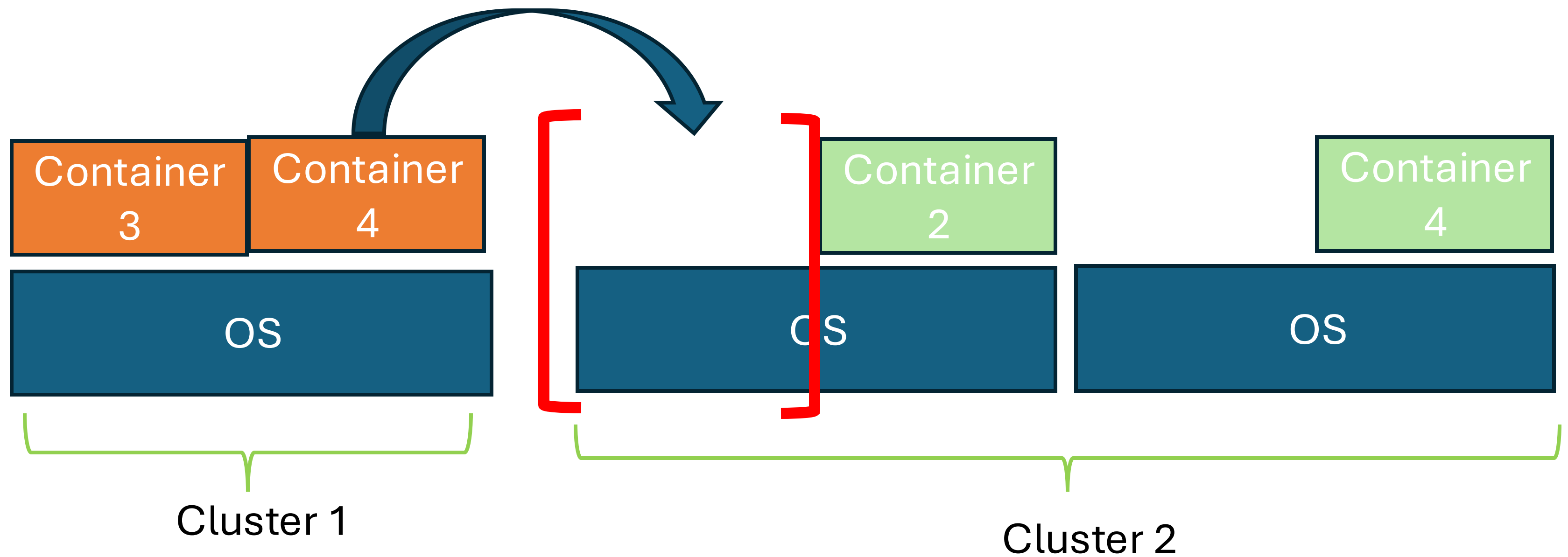
- Single point of management for multiple nodes (machines).
- We are able to control all aspects of a container lifecycle.
- Containers can be accessed via clearly-defined interfaces called "services".
- Containers are scheduled on nodes in a seamless manner.

Liqo

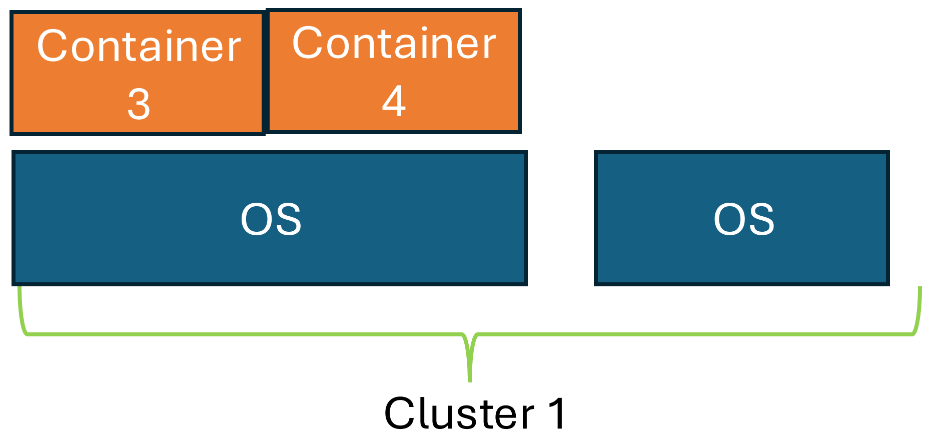
- **Liqo**: An open-source project that enables dynamic and seamless Kubernetes multi-cluster topologies, supporting heterogeneous on-premise, cloud, and edge infrastructures.
- Features:
  - **Peering**: Automatic peer-to-peer establishment of resource and service consumption relationships between independent and heterogeneous clusters.
  - **Offloading**: Seamless workload offloading to remote clusters without requiring any modification to Kubernetes or the applications themselves.
  - **Network Fabric**: Transparent multi-cluster pod-to-pod and pod-to-service connectivity, regardless of the underlying configurations and CNI plugins.
  - **Storage Fabric**: Support for remote execution of stateful workloads, extending standard high availability deployment techniques to multi-cluster scenarios.

Liqo: Virtual Node

Two clusters are peered
The first cluster can deploy on the second Cluster.



After the peering, the first cluster sees the other cluster as a "virtual node"



FLUIDOS

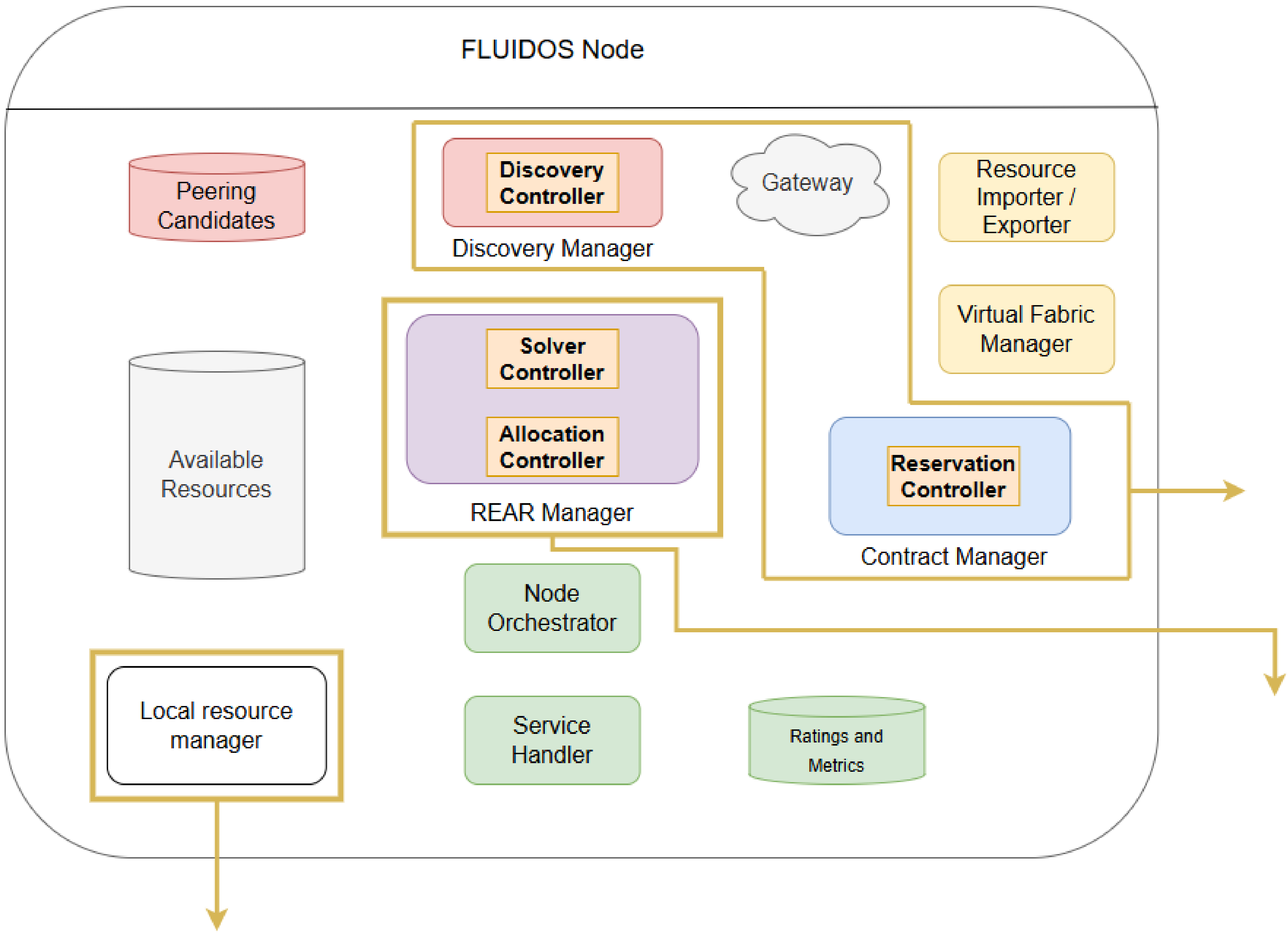
- FLUIDOS builds on top of Liqo providing:
  - Ways for the two clusters to "discover" each other.
  - Ways to limit the amount and kind of resources that a cluster provides to other clusters.
  - A way for a cluster to not only provide slices of its clusters but access to its services.
  - A Zero-Trust security model:
    - every K8s cluster mantains its independence.
    - Using a contract-based system. Any access to a given service is recorded and can potentially be leveraged

FLUIDOS Node Architecture

- **FLUIDOS Node**: K8s Cluster on which FLUIDOS has been installed.
- The FLUIDOS Node is composed of the following components:
  - Local ResourceManager
  - Available Resources
  - Discovery Manager
  - Peering Candidates
  - REAR Manager
  - Contract Manager
- The components are developed using Kubernetes controllers with Kubebuilder, making extensive use of the Kubernetes API and Custom Resource Definitions (CRDs).

Basic k8s things to know

- **Pod:** single deployment unit, it can be a process or coupled processes.
- **Namespace**: a way to divide cluster resources between multiple users or teams, providing a scope for names to avoid conflicts.
- **Services**: abstractions that define a logical set of Pods and a policy to access them, enabling reliable communication within the cluster.
- **Controllers**: is a control loop that watches the state of your cluster and makes or requests changes to achieve the desired state.
- **CRDs**: a way to extend the Kubernetes API by defining new custom resource types.


FLUIDOS Node
Peering Candidates
Discovery Controller
Discovery Manager
Gateway
Resource Importer / Exporter
Virtual Fabric Manager
Available Resources
Solver Controller
Allocation Controller
REAR Manager
Reservation Controller
Contract Manager
Node Orchestrator
Local resource manager
Service Handler
Ratings and Metrics

Terminology

- **Flavour:** It is the kind of resource being provided by a cluster, the same cluster can provide more than one Flavour. It can be of many types:
  - **K8s_slice**: subset of the cluster. When advertising a K8s_slice, a cluster states how much RAM, CPU etc. It can provide the other cluster.
  - **Service:** A Flavour used to advertise a service present in the other cluster that can be used. For instance a database or an HTTP server.
  - **… :** New kind of Flavours are being defined and probably will be. For instance a flavour of type Car.
- **Known Clusters**: Clusters that have been discovered by the Network Manager.
- **Peering Candidates**: List of Flavours available from the Known clusters.
- **Contract:** resource stating the flavour that has been purchased, when, and for how long.

Network Manager

- Currently it is just a process that discovers nearby FLUIDOS Nodes and adds them into the list of "known clusters".
- Its temporary implementation is based on multicast but will be substituted in the future (neuropil).
- Currently works only in the same LAN.

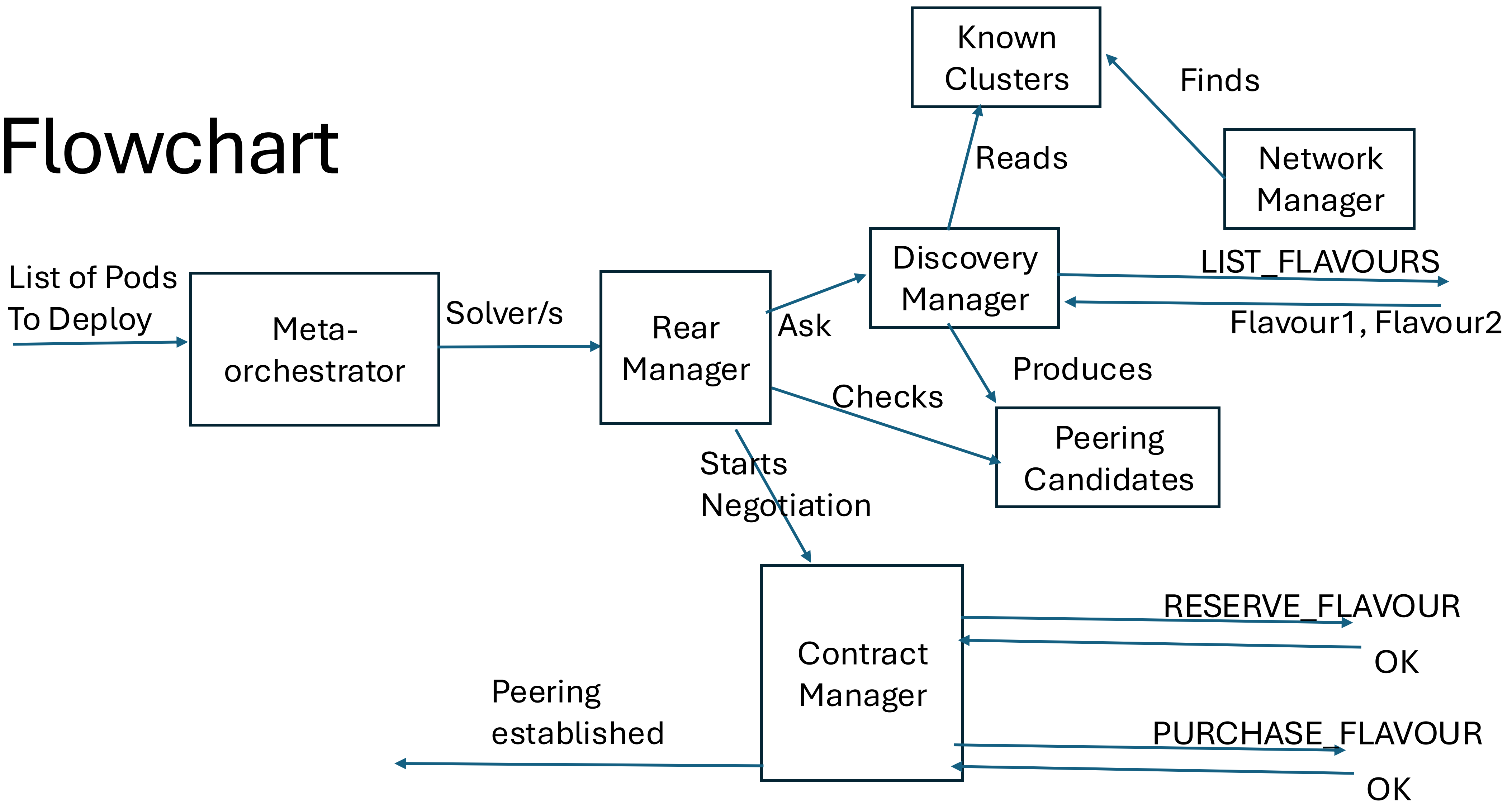
Local Resource Manager

- Developed through a Kubernetes controller.
- Monitors internal resources of individual nodes within a FLUIDOS Node (cluster).
- Generates a Flavour Custom Resource (CR) for each node.
- Stores CRs within the cluster for further management and utilization.

Available Resources

- **Critical part of FLUIDOS**: Manages and stores Flavours.
- **Two primary data structures**:
  - **Free Flavours**: Stores Flavours as defined in the REAR component.
  - **Reserved Flavours**: Stores resource allocation objects/documents as Flavours.
- **Centralized repository**: Provides a list of resources when queried.
- **Integration with Kubernetes' etcd**: Ensures efficient storage and retrieval of data.
- **Facilitates resource management**: Enables efficient allocation and utilization of computing resources within the FLUIDOS ecosystem.

Flowchart



Known Clusters
Finds
Reads
Network Manager
Discovery Manager
LIST_FLAVOURS
Flavour1, Flavour2
List of Pods To Deploy
Meta-orchestrator
Solver/s
Rear Manager
Ask
Produces
Checks
Peering Candidates
Starts Negotiation
Contract Manager
RESERVE_FLAVOUR
OK
PURCHASE_FLAVOUR
OK
Peering established

For More Information

- REAR Protocol Demo
- FLUIDOS Node Demo

Discovery Manager

- **Critical part of resource discovery**: Operates as a Kubernetes controller.
- **Monitors Discovery Custom Resources (CRs)**: Generated by the Solver Controller.
- Primary Objectives
    - **Populating Peering Candidates Table (Client)**: starting from the "known clusters found by the Network Manager.
    - Identifies suitable resources ("Flavours") based on initial REAR protocol messages.
    - **Discovery (Client)**: Initiates a LIST_FLAVOURS message, broadcasting it to all known FLUIDOS Nodes.
    - **Offering Appropriate Flavours (Provider)**: Provides Flavours that best match incoming requests.

Peering Candidates Component

- **Manages a dynamic list of nodes/flavours**: Suitable for establishing peering connections.
- **Continuously updated**: By the Discovery Manager.
- **Stored as Custom Resources**: Uses appropriate Custom Resources for storage.

REAR Manager

- **Orchestrates service provisioning**: Receives solving requests and translates them into resource or service requests.
- **Looks up external resources**: Searches for suitable external resources.
- **Initiates Discovery**: If no Peering Candidates are found.
- **Triggers Reservation phase**: If a suitable candidate is found.
- **Allocates resources**: Upon successful fulfillment of the process.
- **Stores contracts**: Manages contract storage.
- **Starts Peering phase**: Optionally initiates the Peering phase.

Contract Manager

- **Manages reserve and purchase of resources**: Handles negotiation and management of resource contracts between nodes.
- **Initiates Reserve phase**: Sends a RESERVE_FLAVOUR message when a suitable peering candidate is identified and a Reservation is forged.
- **Proceeds to Purchase phase**: Sends a PURCHASE_FLAVOUR message upon successful reservation of resources.
- **Stores contracts**: Manages the storage of received contracts.

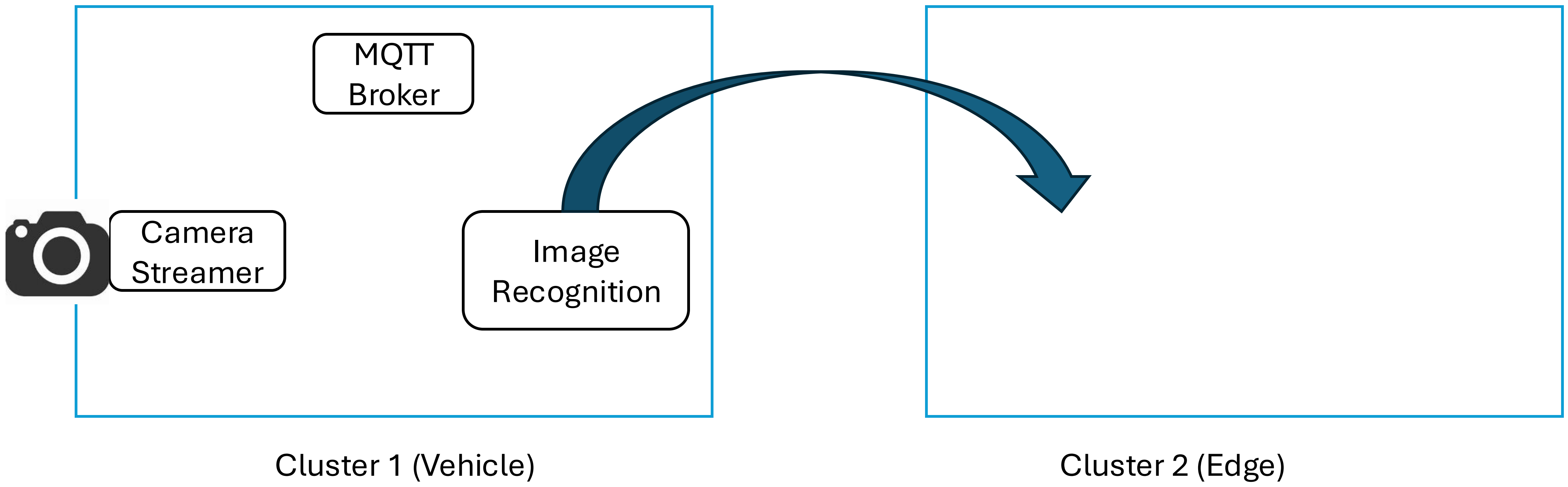
Node Orchestrator

- It is a component that manages the FLUIDOS Node, its main functions are:
    - Given a set of workloads to deploy on FLUIDOS, arranges Liqo peerings to one or more FLUIDOS Nodes by leveraging all the other FLUIDOS Components.
    - In practice, it makes it so every workload will be able to be deployed satisfying one or more criteria (low-latency, carbon footprint, etc).
- It just creates the solver when a new Fluidos Deployment is created, which culminates with a peering.
- It does not decide where the pod will be scheduled, that will be up to the default Kubernetes scheduler.

Work Packages

- WP1 = Core Principles
- WP2 = Reference Architecture
- WP3 = Fluidos Node
- WP4 = Fluidos Continuum
- WP5 = Security
- WP6 = Energy
- WP7 = Use cases
- WP8 = Community and Open Calls
- WP9 = Integration and Testing
- WP10 = Project Coordination

Demo diagram



Important K8s resources in FLUIDOS

- Knownclusters
- Discovery
- Solver
- Reservation
- Contract

Resources

- fluidos-project/Docs: This repository is the starting point for all the documents about the architecture and the implementation of FLUIDOS.
- fluidos-project/node: FLUIDOS Node and its essential components
- FLUIDOS - Official Website
- FLUIDOS - YouTube
- Fluidos Node Orchestrator - Github
- Liqo Documentation