PLEDGER

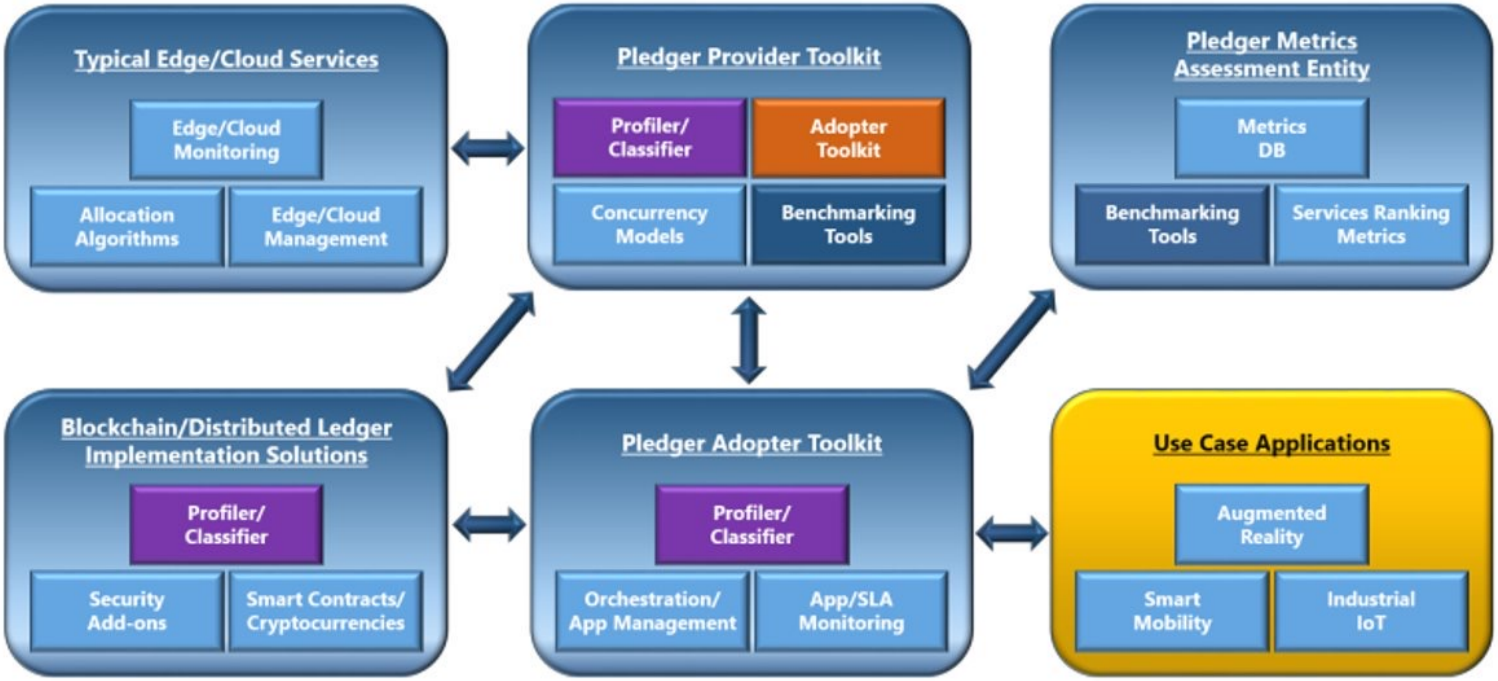
Resource optimisation on the edge: progress and challenges

Francesco Iadanza
Engineering Ingegneria Informatica

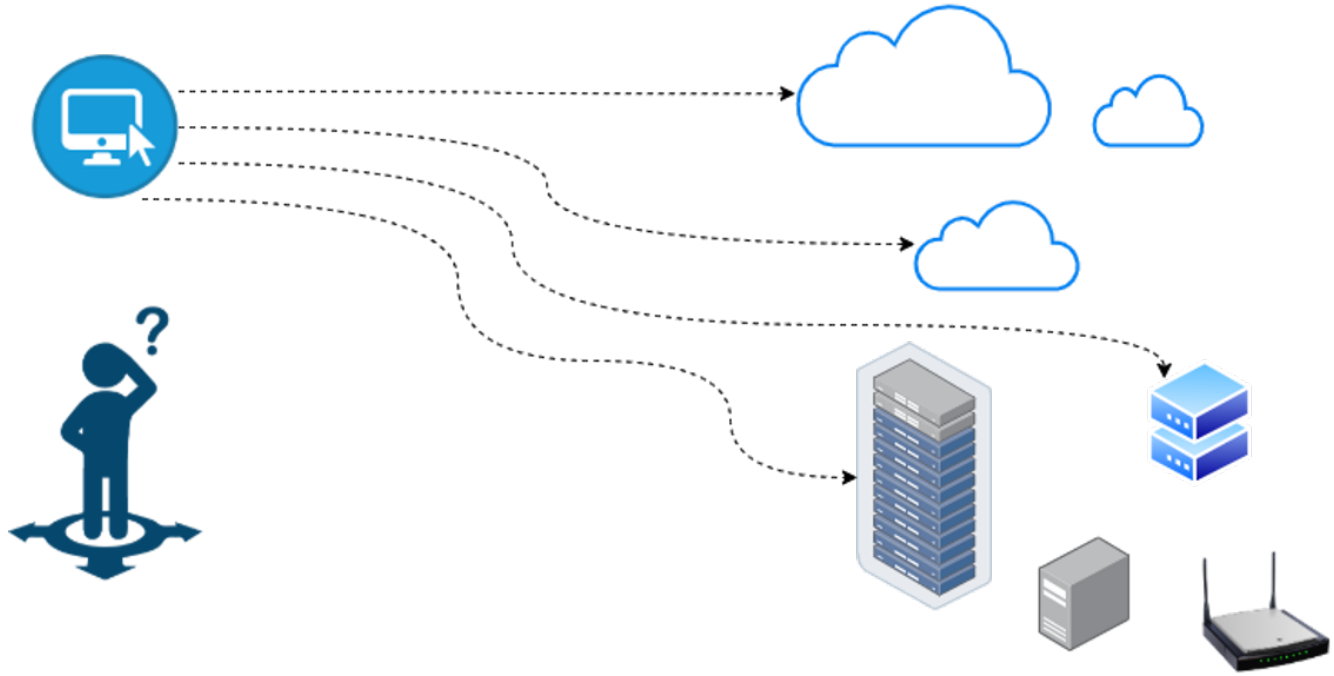
This project has received funding from the European Union's Horizon 2021 research and Innovation Programme under Grant Agreement No. 871536

Pledger project

Goal: deliver a new **architectural paradigm** and a **toolset** for edge computing **providers** and **adopters**, by coupling the benefits of low latencies on the edge, with the robustness and resilience of cloud infrastructures.

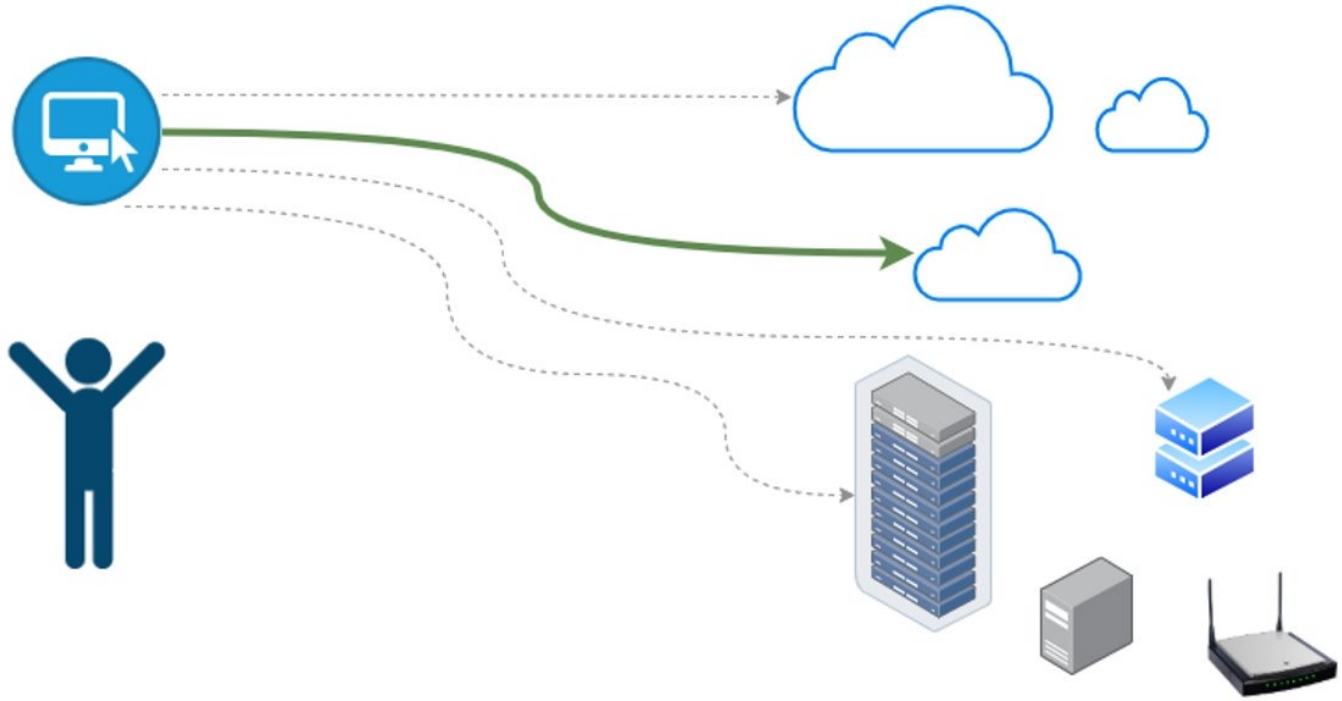


Cloud/Edge infrastructure/nodes selection and QoS/QoE



What actions should I take to make sure my Application fulfills some QoS/QoE in a cloud/edge infrastructure?

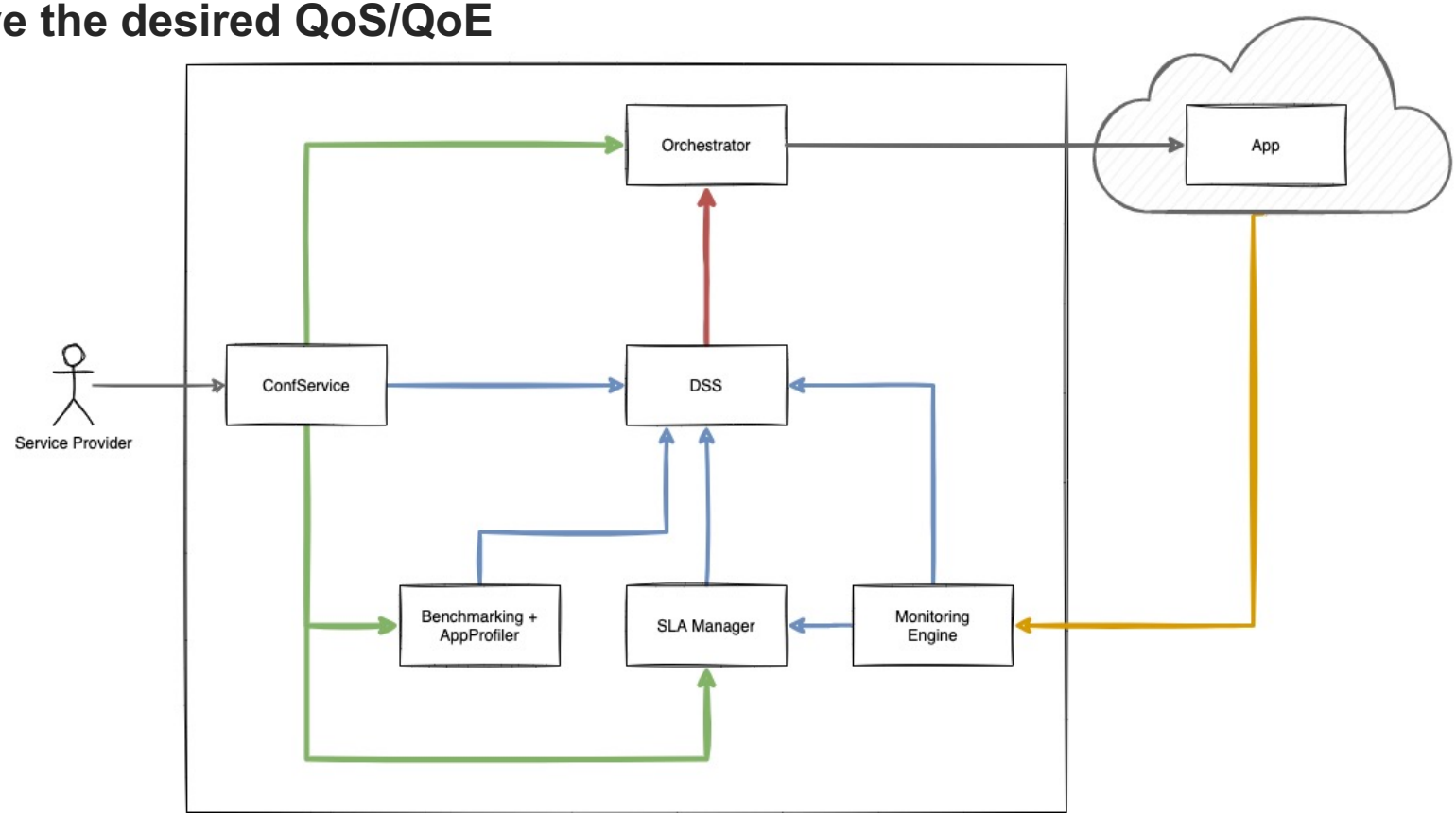
The Pledger workflow



- **Identify** metrics that define the QoS/QoE of the App
- **Measure** the metrics on different providers
- **Monitor** SLA violations and warnings
- **Rank** deployment options managing infrastructures variety
- **Choose** the best infrastructure/nodes for a given App

The DSS and the core components in Pledger

Goal: produce **optimal decisions** for the **resource management/offloading** to **achieve the desired QoS/QoE**



DSS optimisations

The scenarios explored by the DSS are with **private**, **scarce** and **diverse** edge resources

the DSS receives:

- ✓ **deployment option rankings** based on **SP preferences** to leverage on edge **variety**
- ✓ **monitoring data, classification data and SLA violations/warnings**

…then it:

- ✓ creates a **resource allocation model** based on **thresholds** (request/limit) for Apps
- ✓ **changes resource allocation/thresholds** to reduce resource waste and SLA violations
- ✓ **offloads** (when needed) Apps using the **deployment option rankings**

DSS optimisations – some details (1/4)

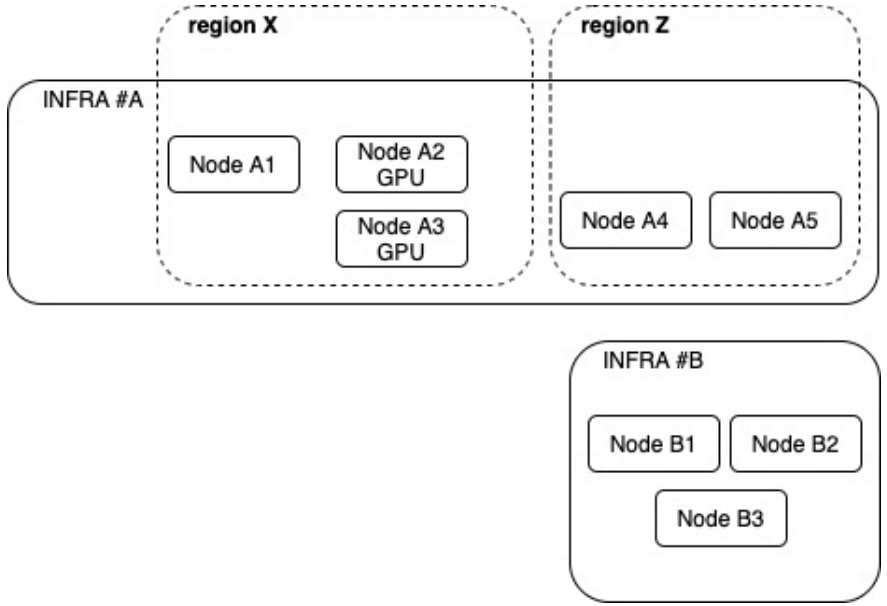
How to manage **edge resources variety**?

Configuration service:

- extracts HW infrastructure **features** (eg. GPU)
- allows SP to **label** Infrastructure Nodes with domain specific information (eg. location etc.)
- allows SP to define deployment **constraints**, for example to privilege edge over cloud

DSS:

- works within such constraints depending on resource requests and availability



example of SP deployment constraints:

1) prefer deployment on Infra #A on region #Z
2) if no resources prefer region #X but only with GPU
3) if no resources prefer Infra #B

default: get back when resources are available again

DSS optimisations – some details (2/4)

How to manage **edge resources allocation**?

DSS inspires to Kubernetes QoS model, based on resource thresholds (min/max)

min («request») represents what an App is guaranteed to get if requested.

- impact: which Apps will be hosted on a resource-limited infrastructure (eg. edge)

max («limit») avoid an App to take too many resources.

- impact: which App will be evicted, that would lead to SLA violations

Thresholds need to be carefully chosen to use edge as much as possible and avoid violations

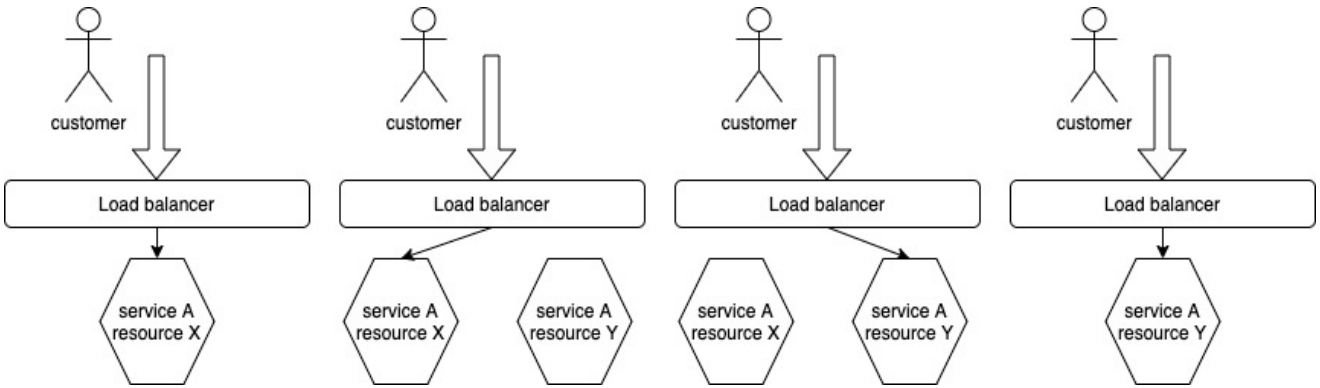
DSS uses **SLA warnings** to **increase/decrease** the **resources** (cpu/memory)
If no resources are available, it **offloads** using the rankings (edge/cloud-to-edge/cloud)

DSS optimisations – some details (3/4)

How to choose **which** App to **optimise resource** first?

The DSS uses SP ”weights” to compute App critical score based on SLA violations

Also, in case of **scale up/down,** load-balancer is used to avoid service disruption and the DSS monitors the Apps **time to boot** to prioritise those that are faster



DSS optimisations – some details (4/4)

How to choose among multiple deployment options (Nodes) with the **same ranking**?
The DSS uses

- **Benchmarking** based on **App profile**, to prioritise Nodes which better fits an App

How to choose among different edge nodes?
The DSS uses

- **edge nodes latency,** to prioritise Nodes with lower values (**edge-to-edge** offloading)

What if a **custom** automation is needed?
The DSS integrates with **NodeRed** to allow custom rules through an easy UI

Next steps and challenges

- ✓ Finalize the integration with the use cases and validate the algorithm with i2CAT
- ✓ Include far-edge devices to support edge-to-edge offloading based on edge node latency
- ✓ Include energy benchmarks on edge devices to optimize service allocation
- ✓ Experiment distributed load-balancing in multi-clusters (eg. for weighted load distribution)

PLEDGER

Thanks for your attention!

PLEDGER www.pledger-project.eu

twitter.com/Pledgerproject

facebook.com/pledgerproject

linkedin.com/in/pledger-project