fog Ø5

End to End Compute, Storage and Networking.

Angelo Corsaro, PhD
Chief Technology Officer
ADLINK Tech. Inc.
Co-Chair, OMG DDS-SIG
angelo.corsaro@adlinktech.com

Gabriele Baldoni
Junior Technologist
Advanced Technologies Office
ADLINK Technologies Inc.
gabriele.baldoni@adlinktech.com

PRISMTECH™
AN ADLINK COMPANY

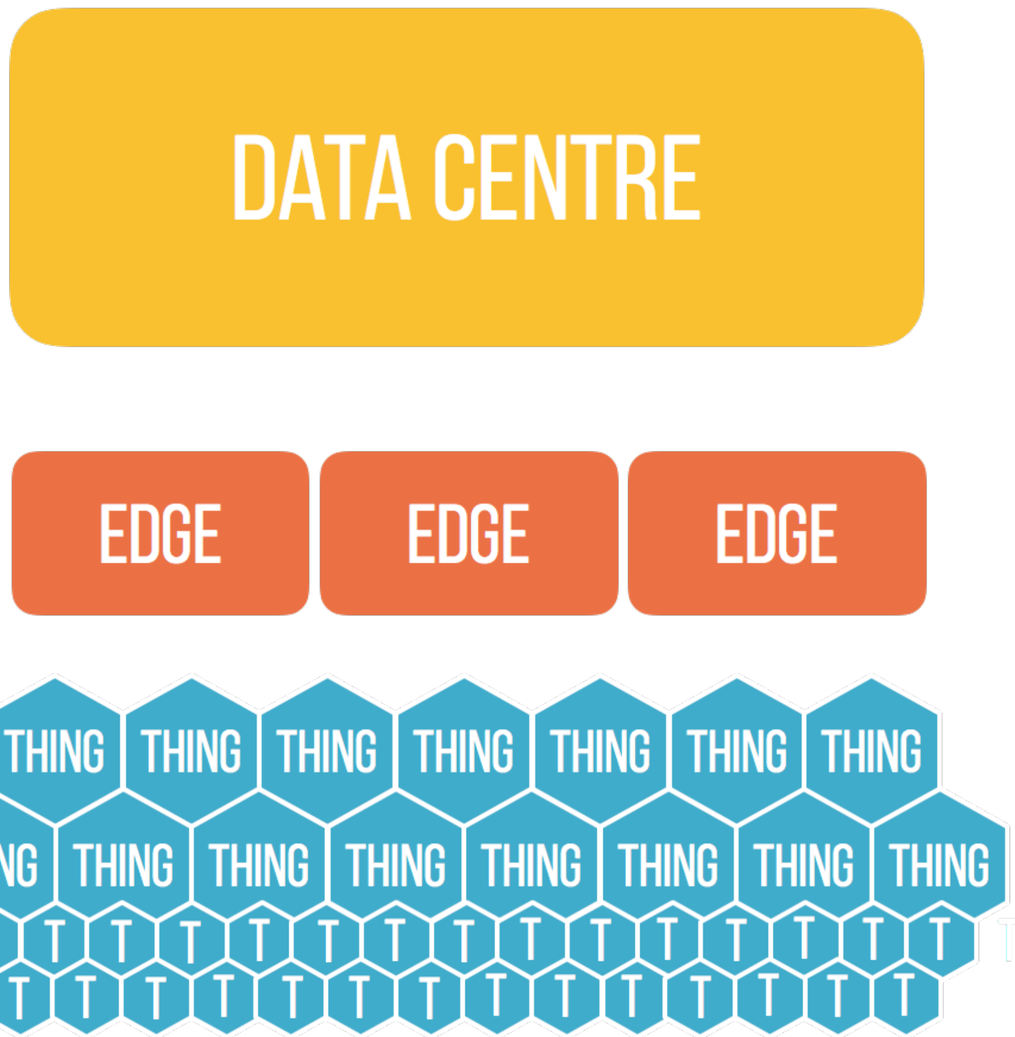
Hardware Tiers in IoT

A generic IoT/IIoT system has **three** different **hardware tiers**

Off-premises data-centre which may be private or public

On-premises edge infrastructure

Things with computational, communication and storage capacity



Cloud-Centric

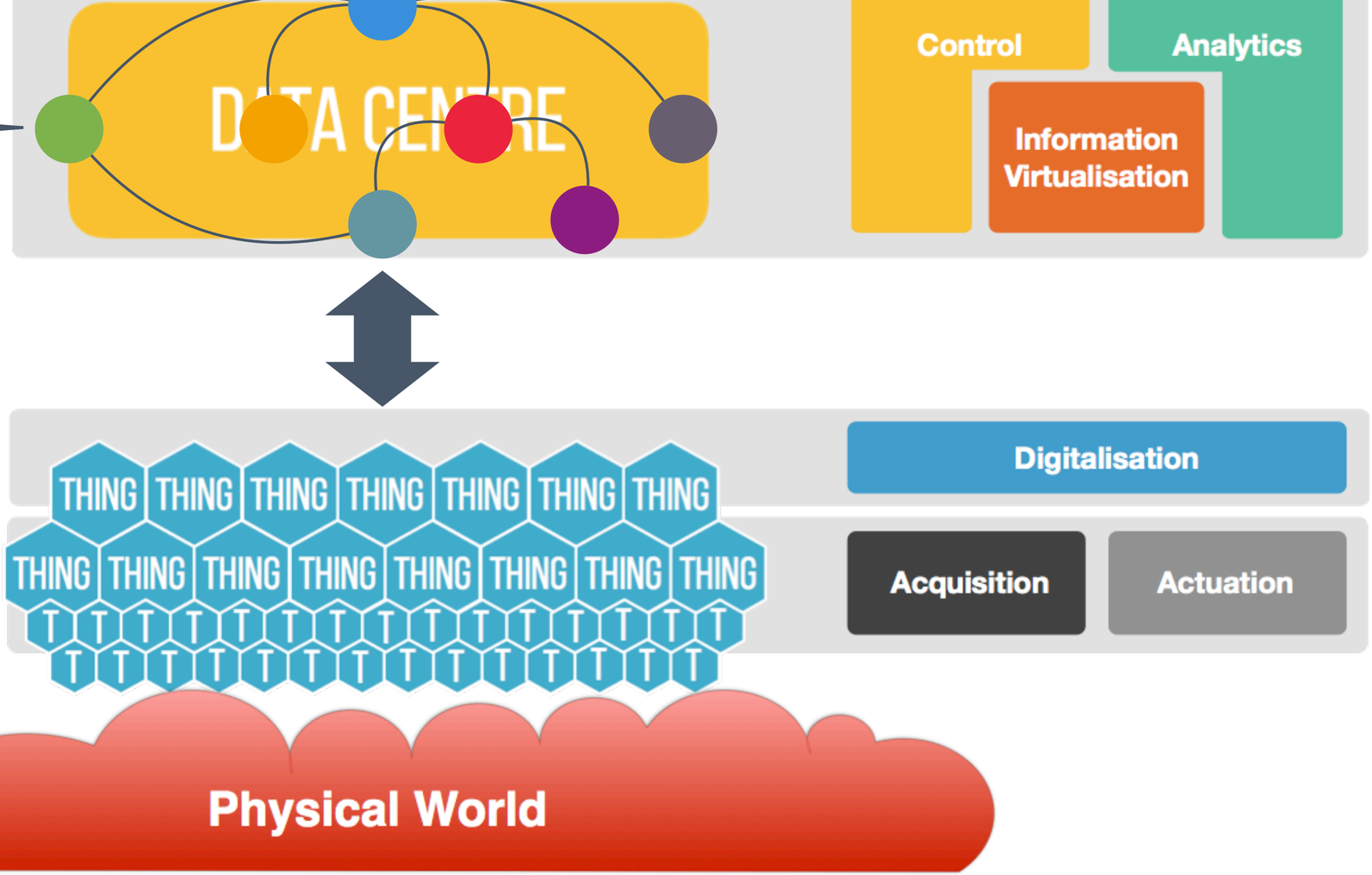
The early days of IoT/IIoT have been biased by a **cloud centric perspective**

The cloud infrastructure is **mature** and **operationally convenient**…

Yet **cloud centric architectures don't fit well** for a **large class** of **IoT/IIoT applications**

Cloud Centric

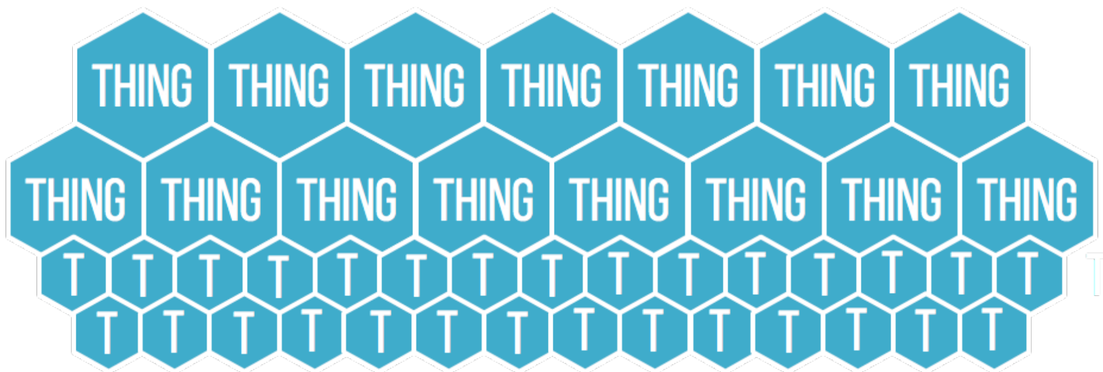
The IoT application is deployed, managed and monitored using the Cloud IaaS infrastructure



The Reality is

Only a **restricted class of IoT/ IIoT applications** that are **compatible** with the **cloud-centric model**

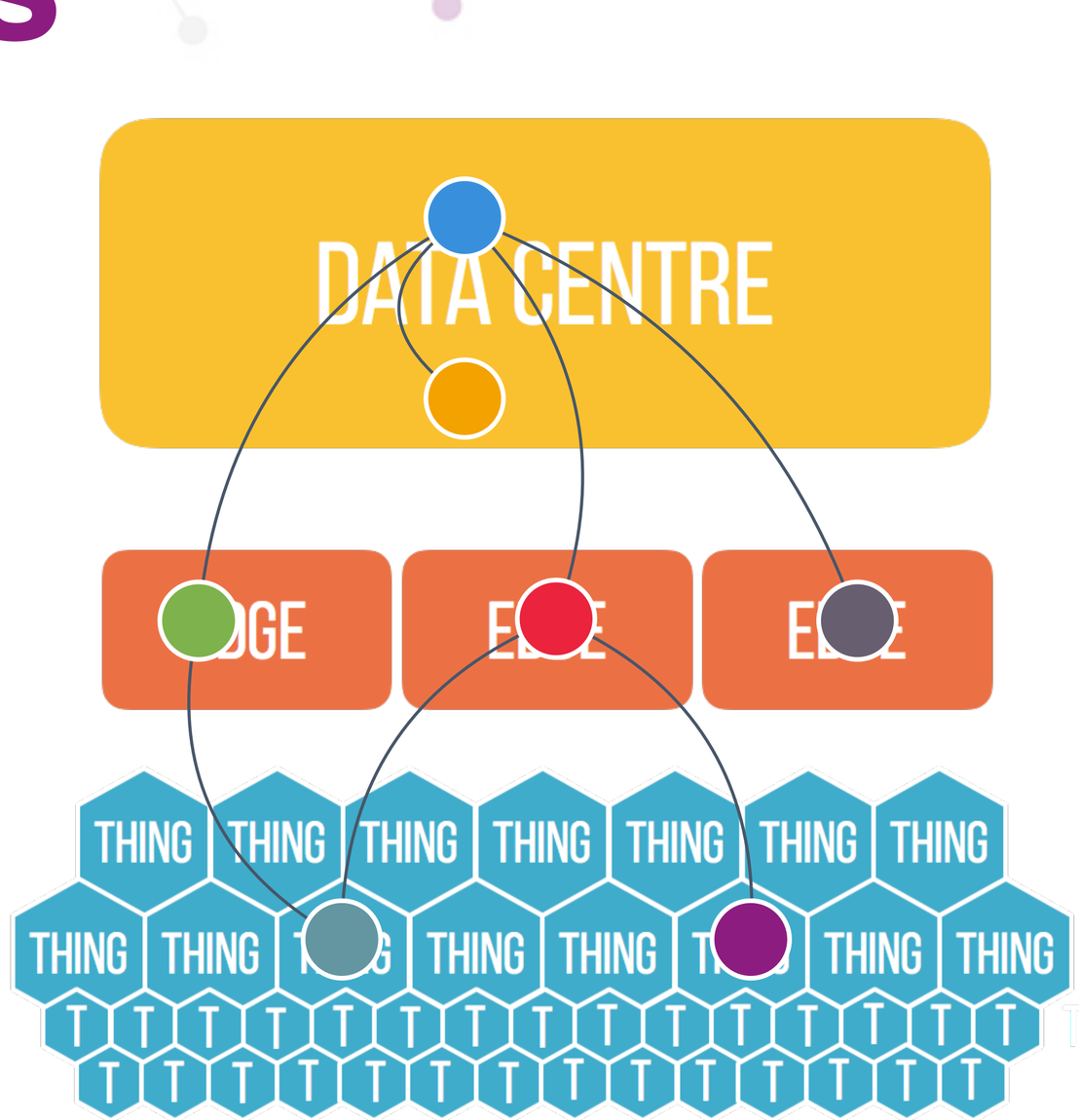
In general the **entities defining** an **application** need to be **distributed** across the **three tiers**



The Reality is

Thus we need to be able to **provision**, **manage** and **monitor** applications **across** these **tiers**

Consequently we have to **virtualise** the **infrastructure end-to-end**

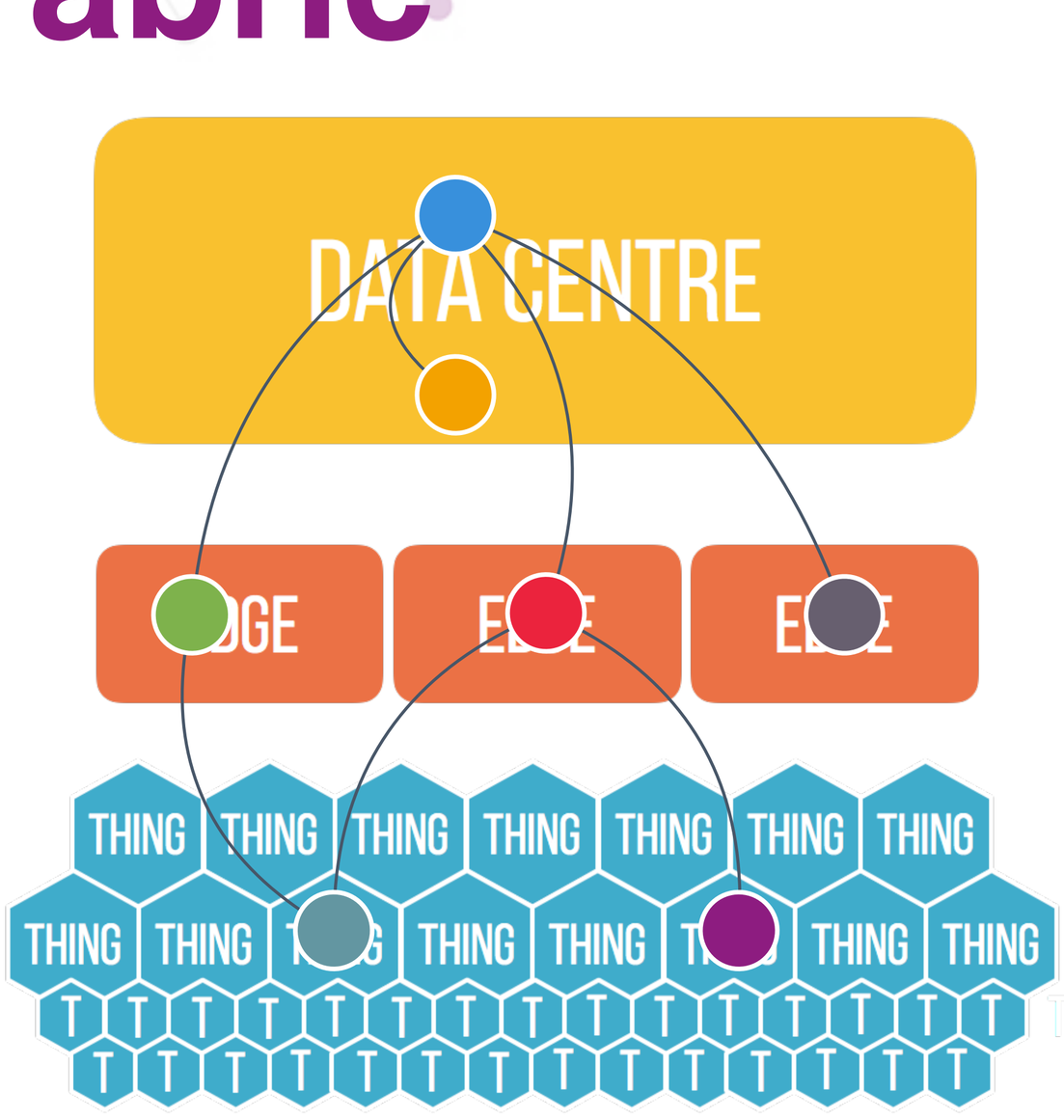


End-to-End Fabric

We need to provide the abstraction of an **end-to-end compute**, **storage** and **communication fabric**

Yet the **requirements** of applications deployed **across the three tiers** as well as the technology ecosystem are **extremely different**

What is the solution?

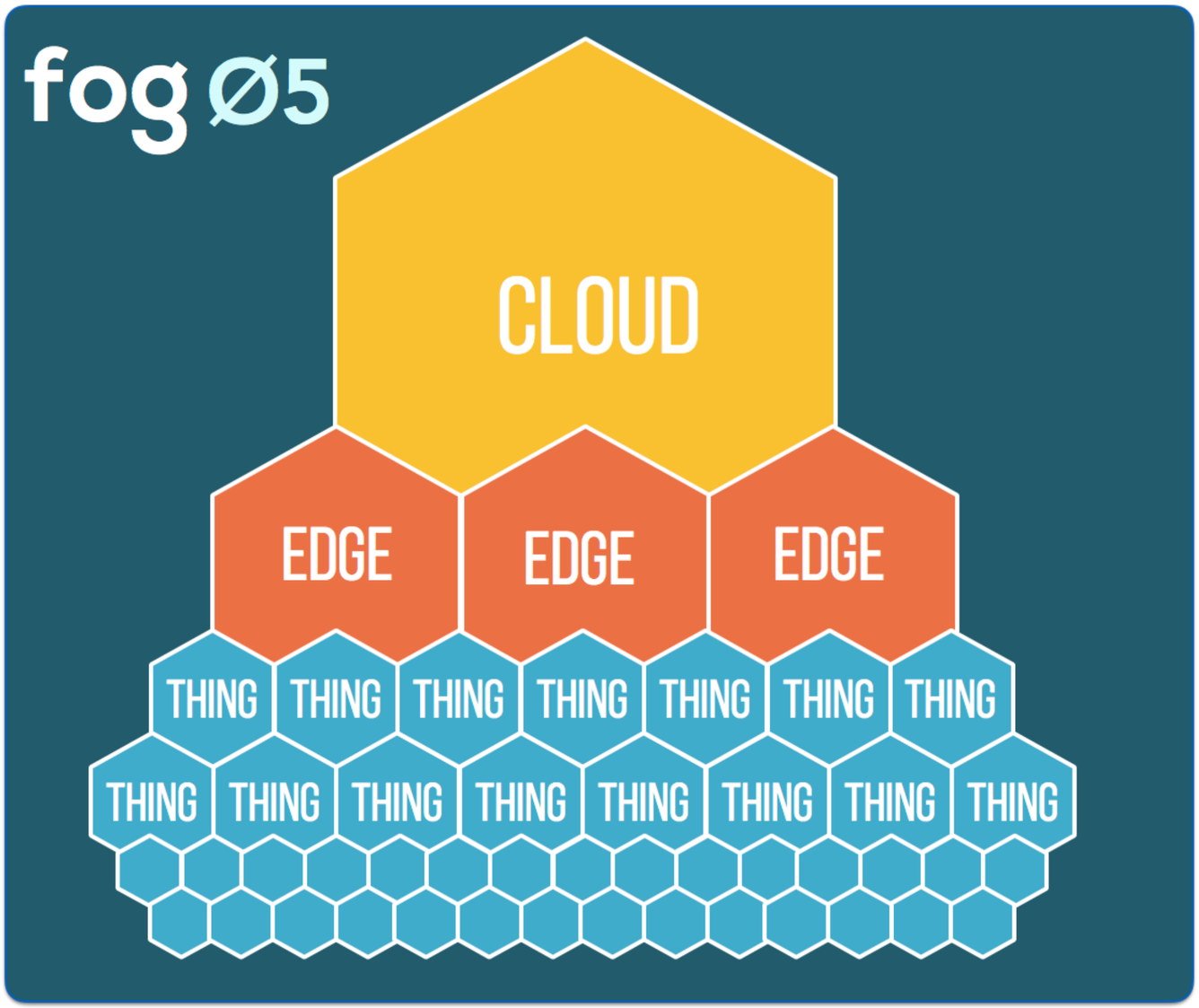


fog Ø5

Compute Fabric

The **fogOS** unifies the compute fabric that spans across things, edge and cloud infrastructure

It unifies administration, management and monitoring end-to-end

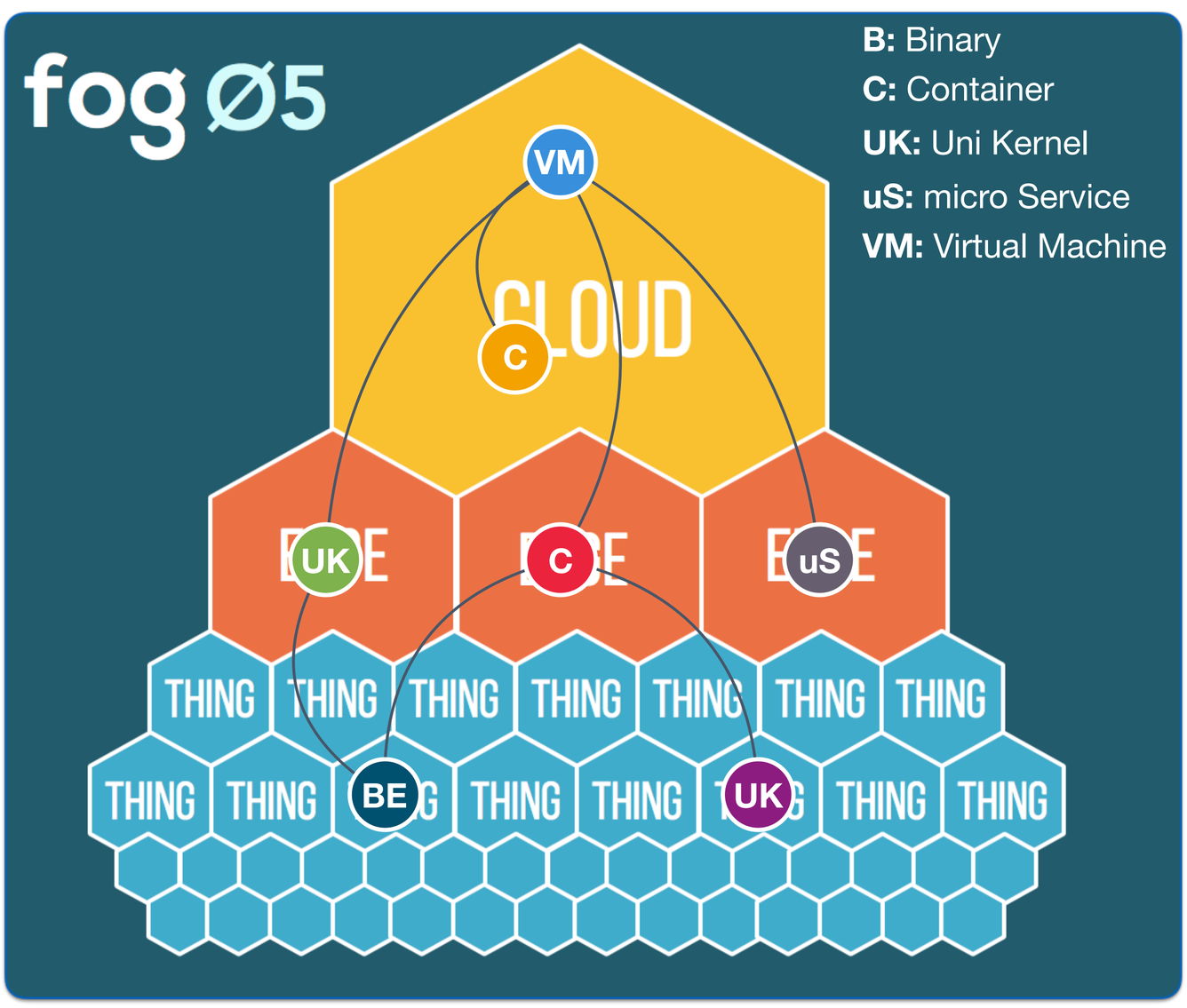


One App, Many Entity Kinds

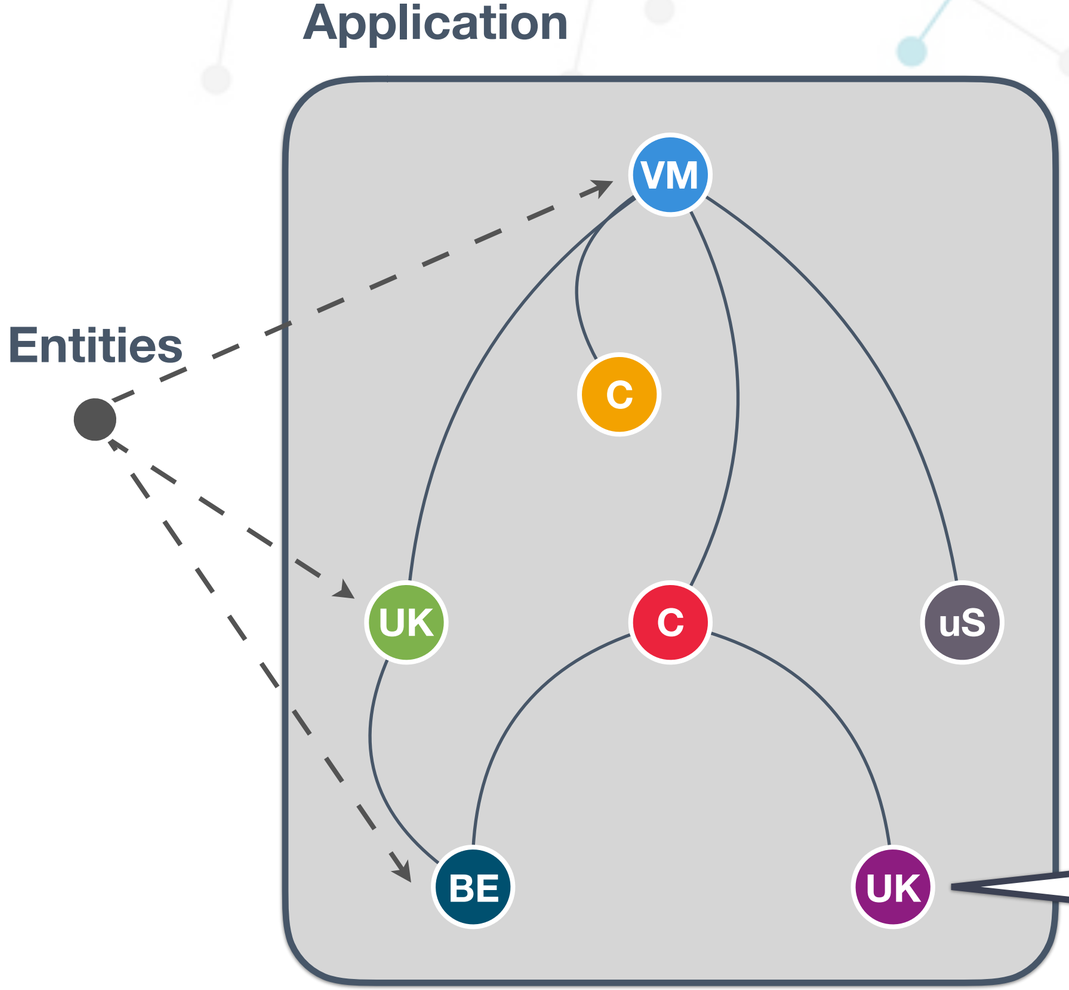
A **fogOS** application is a graph of deployable entities, such as VM, Containers, Uni-Kernels, Binaries, etc.

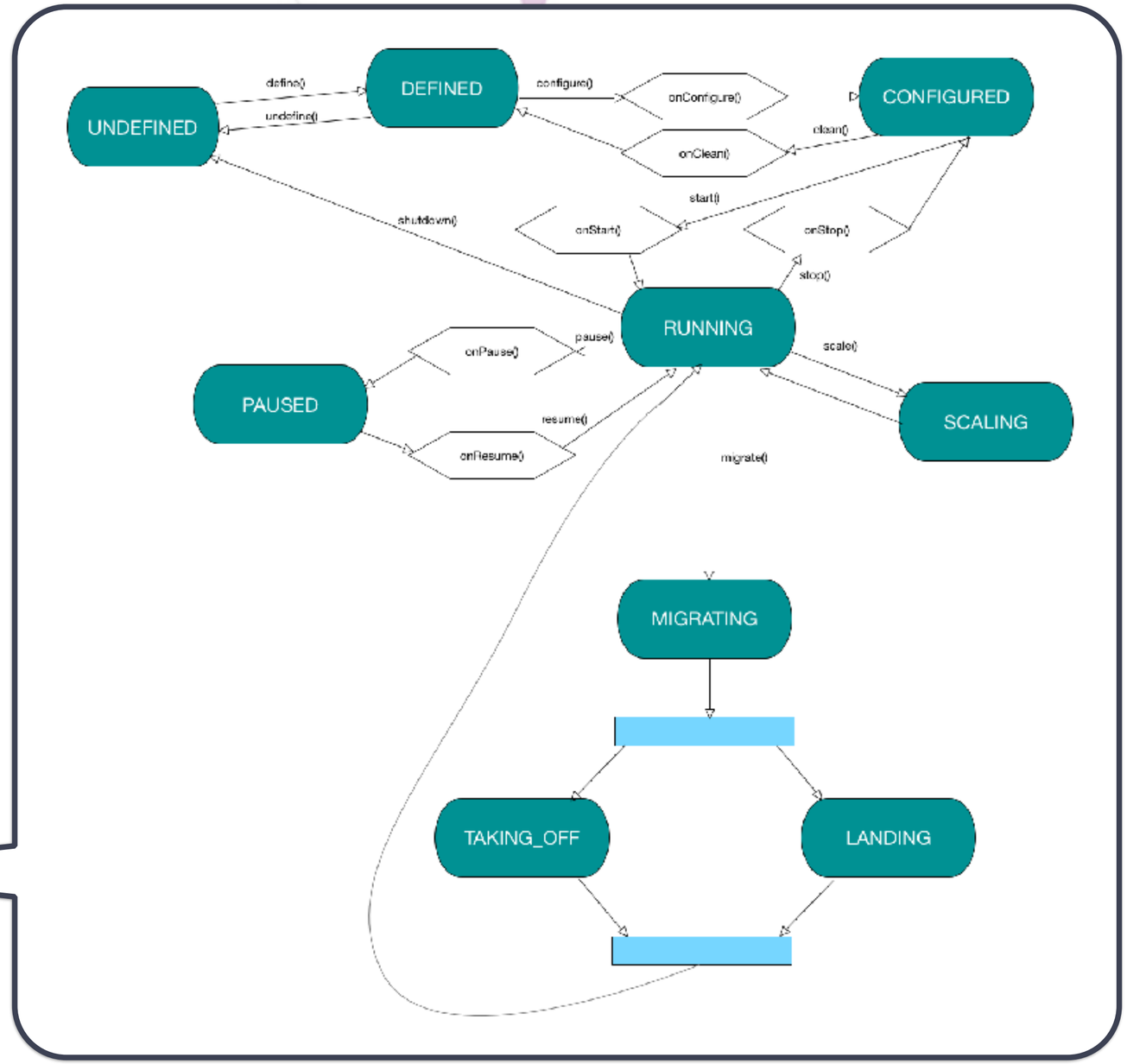
These entities can have deployment affinity w.r.t. to each other as well as with respect to compute, storage, I/O and accelerates resources

fogOS uses plug-in for dealing with different kinds of entities



FogOS Application



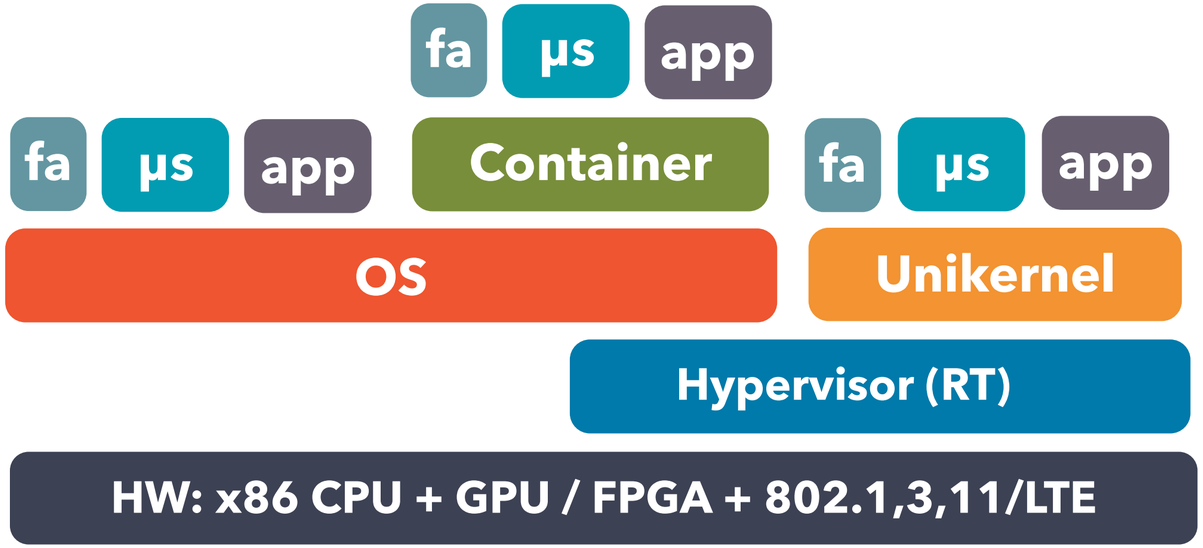


fog Ø5

Architecture

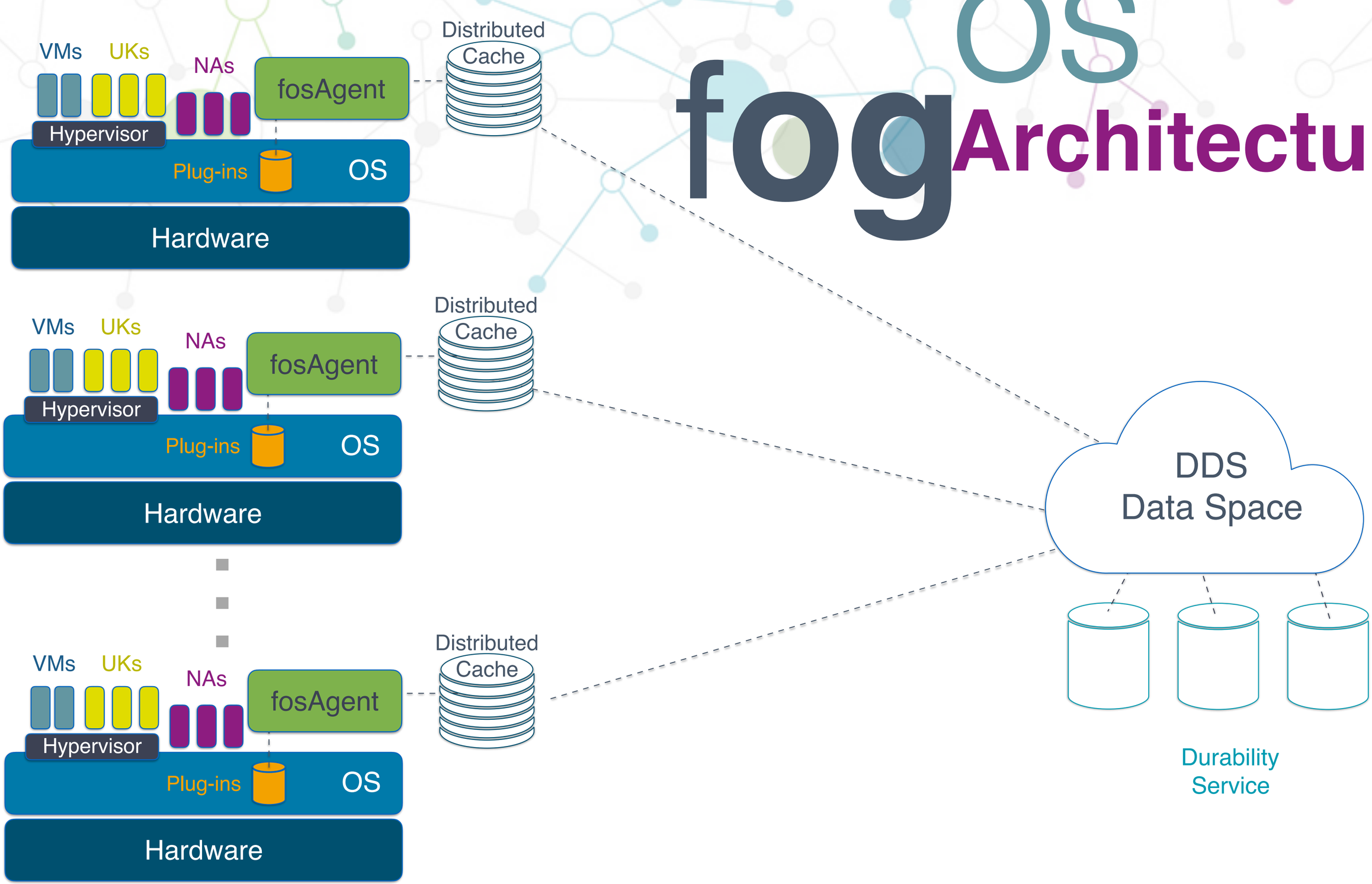
fog OS Architecture

fogOS is an infrastructure to provision, manage and monitor applications composed by different kinds of deployable bundles, ranging from a micro-service to a full VM



fa: fog agent
µS: micro-service
app: application

fog OS Architecture

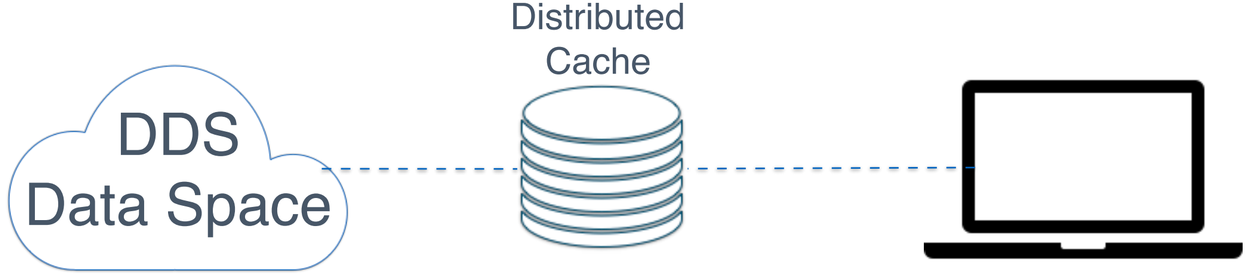


fog Ø5

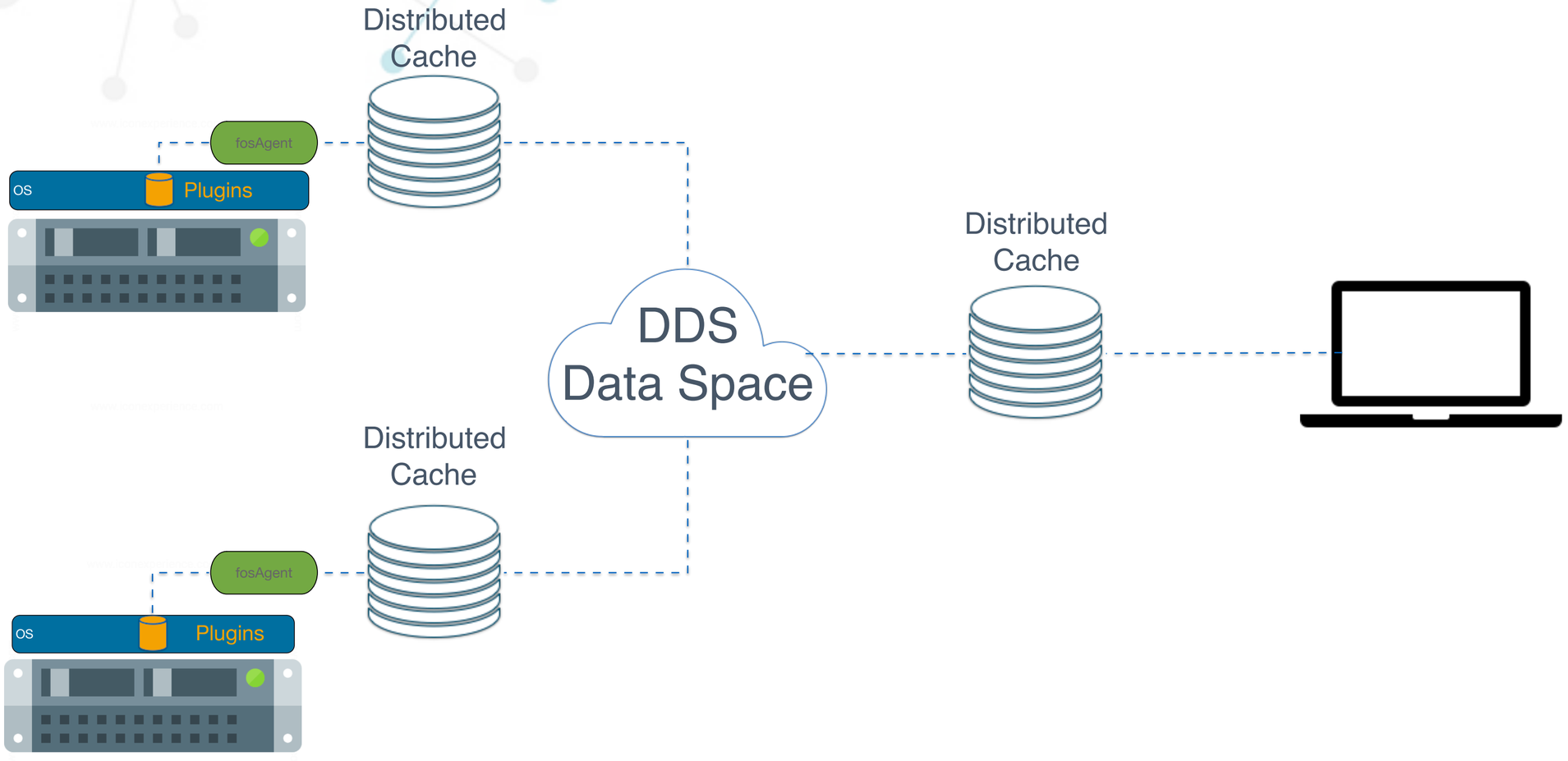
Demo

Demo Description

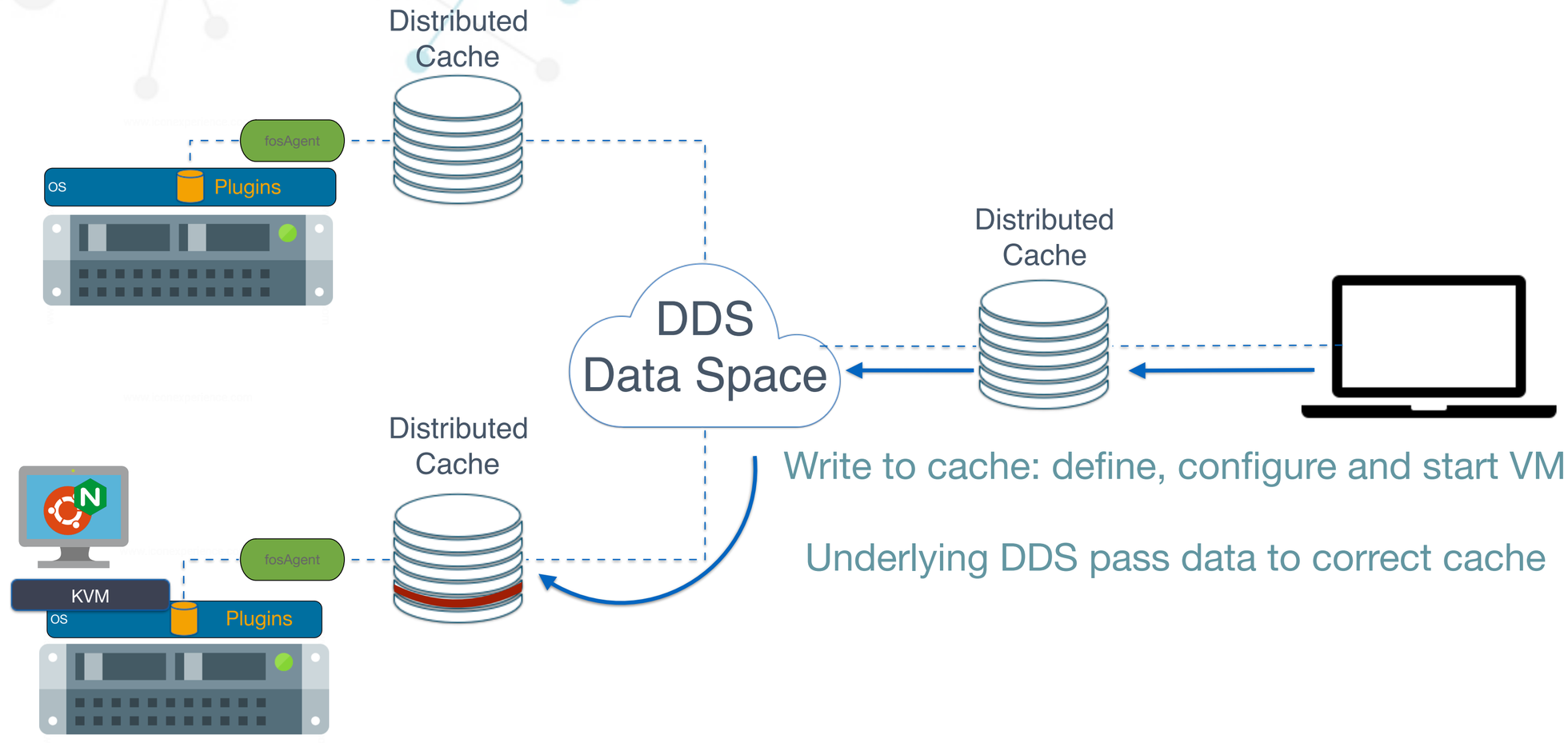
Demo Architecture



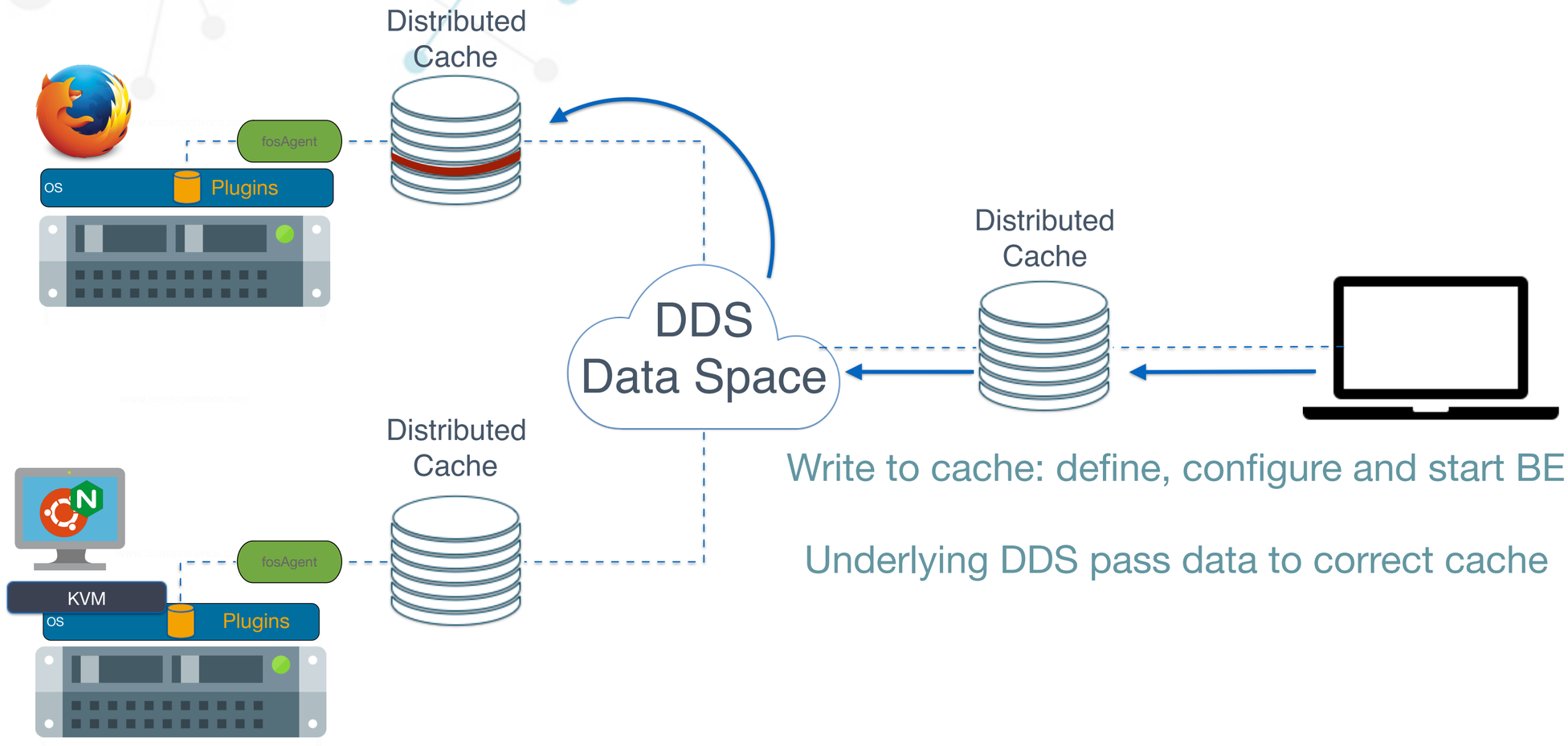
Demo Architecture



Demo Architecture



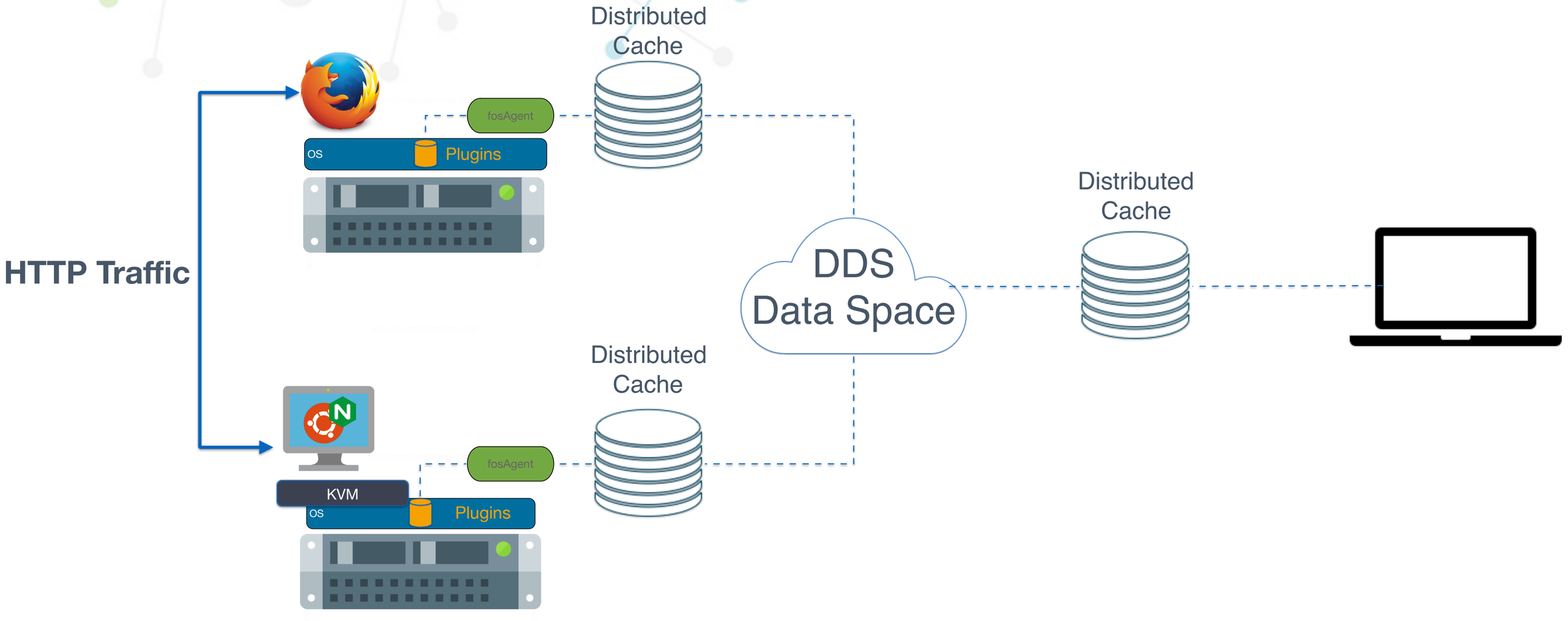
Demo Architecture



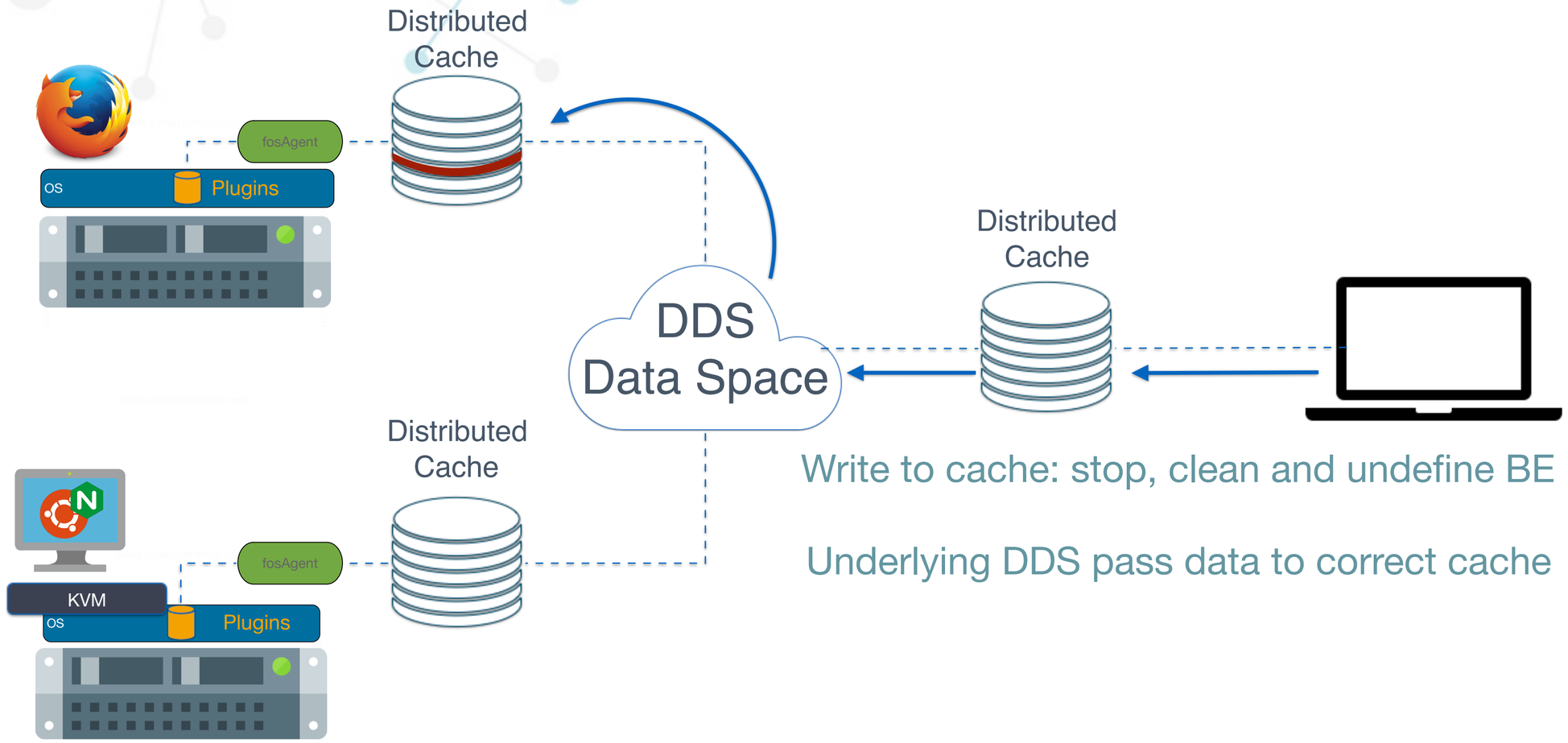
Write to cache: define, configure and start BE

Underlying DDS pass data to correct cache

Demo Architecture



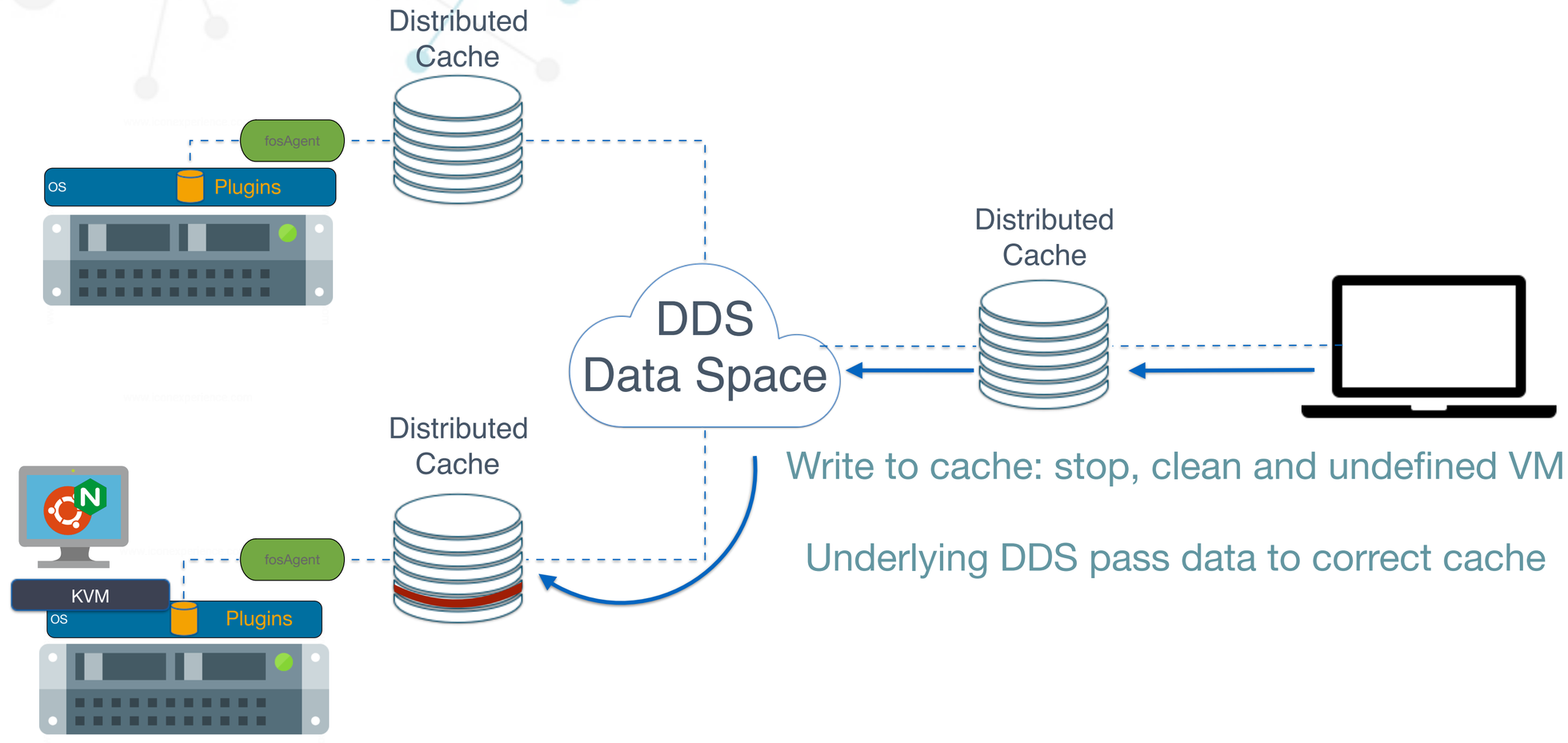
Demo Architecture



Write to cache: stop, clean and undefine BE

Underlying DDS pass data to correct cache

Demo Architecture



fog Ø5 R&D

R&D Directions

Static Provisioning

- Entity Model
- Application Model
- Application Dependency Graph
- Plugins for Unikernel, Container, VM, Binaries, ROS2 nodes

Affinity Provisioning

- Optimal (near-optimal) placement of application entities w.r.t. resource requirements, affinity and communication flows
- Network Virtualisation

Dynamic Res. Mngmt

- Dynamic adaptation of entities to match resource availability and workloads

- End-to-End Network Virtualisation
- Accelerators Virtualisation
- Real-Time Hypervisors
- WAN-resiliance
- Fog PaaS

Dynamic Res. Mngmt

fog Ø5 Collaborations

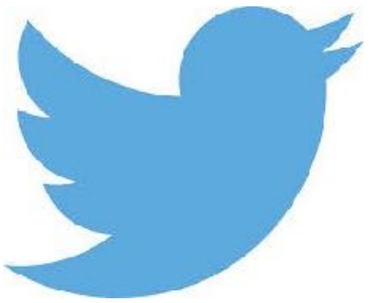
Collaboration Model

fogOS will be released as **Open Source** by the **end of the year**

We would like III to join our Open Source project and make **fogOS** the **de-facto standard fog-computing IaaS**

fog Ø5

Next Steps?


Appendix

Distributed Cache

The FogOS distributed Cache is the abstraction used to operate on the system.

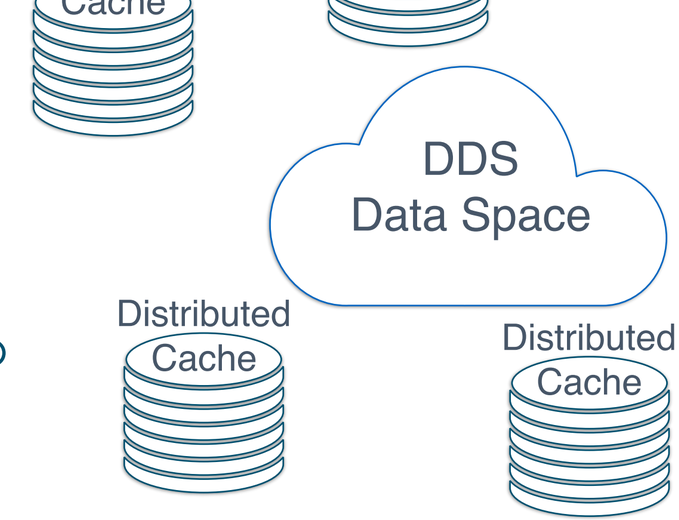
Each cache as a root URI, such as `fos://system-id/node-id` that identifies its "root"

All resources that are below the cache root are considered as local and always kept in memory.

Non local resources are cached in a fixed size sub cache and thus may be evicted as a consequence of a conflict

Local miss are resolved using a distributed look-up

At start-up the cache is populated through data coming from the agent and the durability service



Distributed Cache

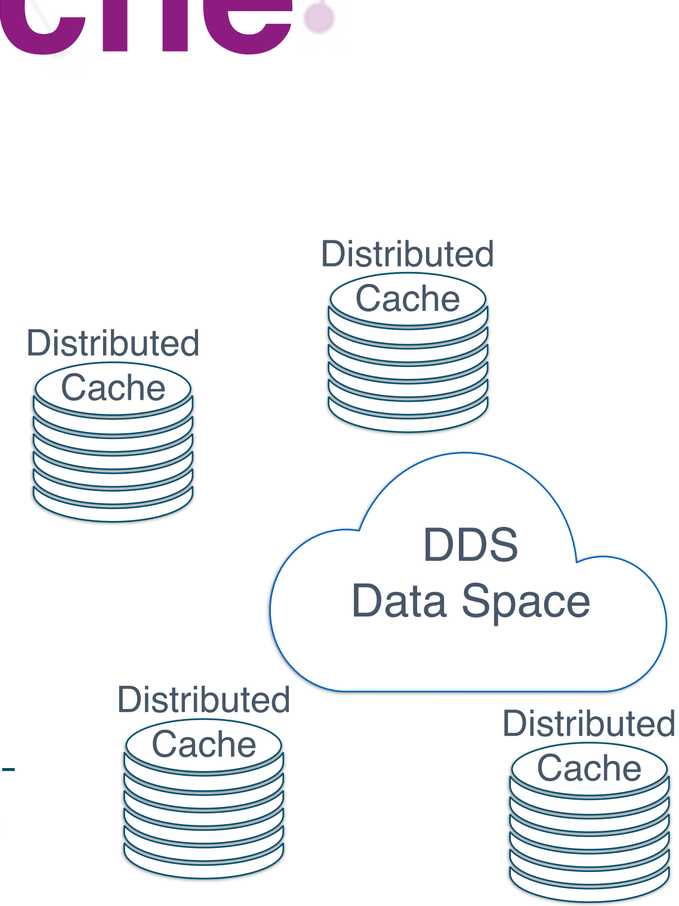
The distributed cache stores key values.

The key is the URI of the resource, such as:
fos://system-id/node-id/plugins

The value is a JSON object describing the resource

Notice that the cache maintains the state of the system. Its persistent portion makes it possible to pre-provision a node or restart in a known configuration after a crash.

As such monitoring the system is equivalent to observing cache resources. Likewise controlling the system is equivalent to putting removing resources.



Distributed Cache Operations

The operations supported by the cache are:

- put, pput, dput
- get
- observe
- remove