Towards a Carrier Grade ONAP Platform

Multi Cloud (MC) Architecture for R2+ & Alignment to S3P

Key Contributors:

Ramki Krishnan, Sumit Verdi, Xinhui Li, Danny Lin, Bin Hu, Gil Hellmann, Bin Yang, Shankar Narayanan, Sastry Isukapalli, Srinivasa Addepalli, Dan Timoney, Brian Freeman, Rajesh Gadiyar, Vimal Begwani, Andrew Philip, Eric Troup, Amir Levy, Alex Vul

Background Material

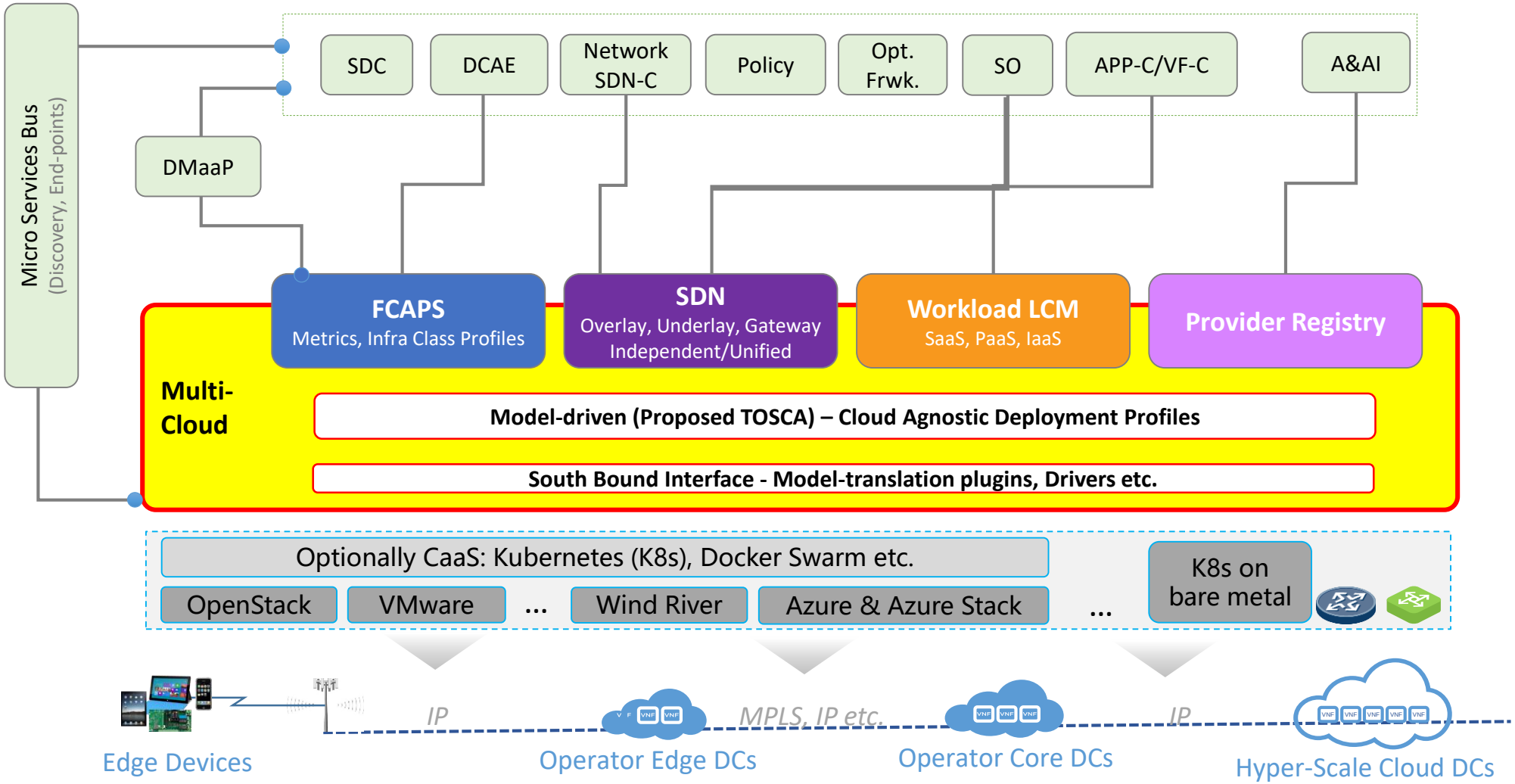
ONAP Paris Workshop on “ONAP Multi Cloud Architectural vision for R2 and beyond”

- Multi-Cloud Architectural Vision Introduction - https://wiki.onap.org/download/attachments/11928197/ONAP-mc-intro.pdf?version=1&modificationDate=1506518564000&api=v2
- Performance & Isolation - "Towards a performance-aware and portable cloud-agnostic Infrastructure" - https://wiki.onap.org/download/attachments/11928197/ONAP-mc-performance-isolation.pdf?version=1&modificationDate=1506518613000&api=v2
- "Architectural options for Multi-vendor SDN Controller and Multi Cloud Deployments in a DC“ - https://wiki.onap.org/download/attachments/11928197/ONAP-mc-sdn.pdf?version=1&modificationDate=1506518708000&api=v2
- FCAPS Telemetry - "Standardized Infrastructure Class statistics Model“ - https://wiki.onap.org/download/attachments/11928197/ONAP-mc-fcaps.pdf?version=2&modificationDate=1506519214000&api=v2
- Minutes on Multi-Cloud follow up and R2 scope discussion (09/28/2017): https://wiki.onap.org/download/attachments/11928197/Multi-Cloud%20follow%20up%20and%20R2%20discussion.pdf?version=3&modificationDate=1506961823000&api=v2

Agenda

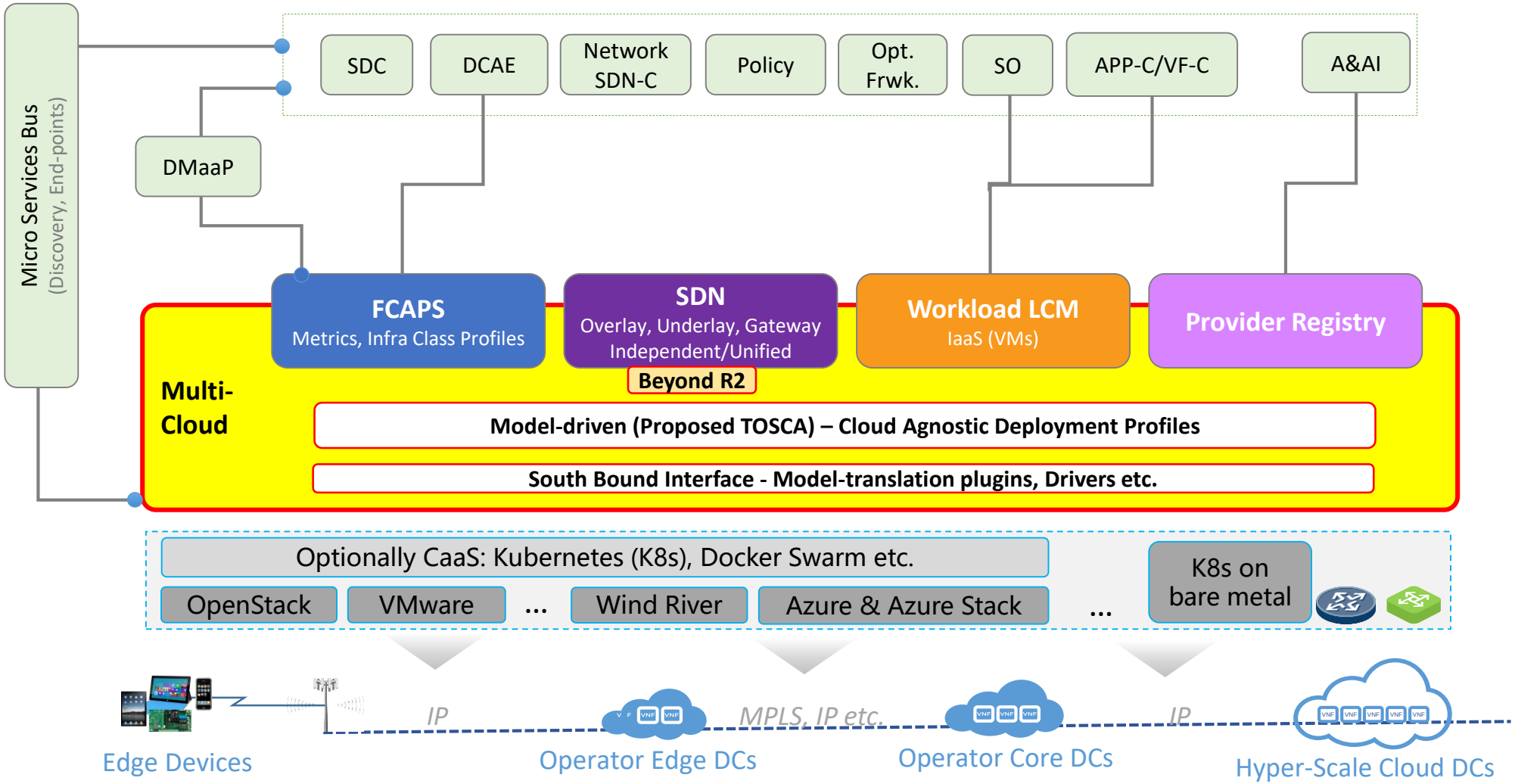
- MC Reference Architecture for R2 & Beyond
- MC High Level SW Architecture for R2
- ONAP R2 Platform S3P Alignment
  - Stability
    - Standardized Data Model for Objects, Capabilities, External Interfaces & Logging
    - Consistent intent-driven APIs for accessing data stores aka “Data Microservices”
  - Scalability for Data Management
    - Push Aggregate Data through Asynchronous Methods

Multi-Cloud Reference Architecture – R2 & Beyond



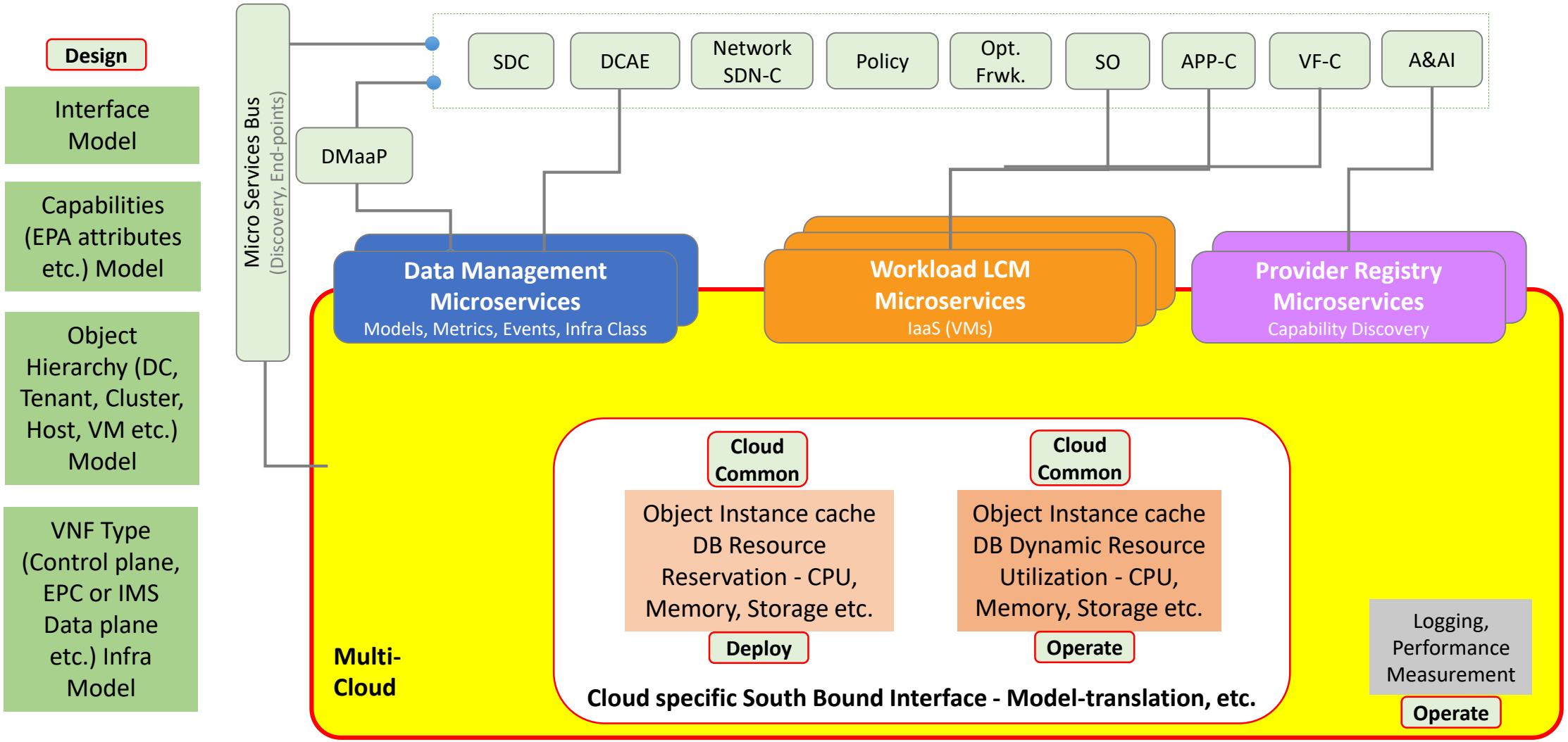
Adapted from ONAP Paris MC Workshop Introduction :
https://wiki.onap.org/download/attachments/11928197/ONAP-mc-intro.pdf?version=1&modificationDate=1506518564000&api=v2

Multi-Cloud Reference Architecture – R2



Adapted from ONAP Paris MC Workshop Introduction :
https://wiki.onap.org/download/attachments/11928197/ONAP-mc-intro.pdf?version=1&modificationDate=1506518564000&api=v2

Multi-Cloud High Level SW Architecture for R2



Multi-Cloud High-level SW Architecture for R2 – Details (1)

- Align with factory microservice constructs
  - Capability/Provider Registry (Catalog); Capacity/Reservation (Inventory); Workload LCM (Request)
  - Data Management - Significant change in resource usage (Publish)
  - Data Management - Bulk data such as Infrastructure Class statistics, Logs (Publish)
  - Data Management - External events (Subscribe)

- Consistent intent-driven APIs for accessing data stores aka “Data Microservices” – goal is to decouple function from type of data store

- Standardized Interface, Object Hierarchy, Capability Models stored in SDC and published through DMaaP
  - Leveraged by consumers such as Multi Cloud, DCAE, APP-C, Policy, OF, SO, A&AI etc.
  - SDC to provide consistent APIs for models data store – have the flexibility of different data stores based on data longevity – e.g. object hierarchy is more likely to change than capability

- Supported Capabilities per Cloud Provider Instance & Capacity/Reservation per object per Cloud Provider Instance stored in A&AI and published through DMaaP
  - Leveraged by consumers such as OF to determine appropriate Multi Cloud instances for homing
  - A&AI to provide consistent APIs – current APIs are graph database centric and may be need to be replaced by a NoSQL database for performance reasons

Multi-Cloud High-level SW Architecture for R2 – Details (2)

- Transaction Logging for Performance Baselining & Troubleshooting
  - Multi Cloud Mediation Layer (plugin independent)
- Workload LCM, e.g. Tenant create
  - If needed, Object instance cache DB (resource reservation) dynamically created in Multi Cloud Plugin
- Data Management subscription for Infrastructure Class, e.g. Tenant utilization monitoring
  - If needed, Object instance cache DB (resource utilization) dynamically created in Multi Cloud Plugin
- Data Management subscription for significant change in resource usage (Publish)
  - If needed, Object instance cache DB (significant change) dynamically created in Multi Cloud Plugin
- State reconciliation following temporary interruptions to the master data store, e.g. A&AI goes down temporarily and gets restarted (edge case which is not handled through HA)
  - Modules such OF which use A&AI can leverage to Multi Cloud object instance cache in Multi Cloud Plugin to make progress
- Objects instances DB cache classified based on functionality and data longevity
  - Object instance resource utilization cache DB is typically long-lived

Stability

- Standardized Data Model for Objects, Capabilities, External Interfaces & Logging
- Generic Data Models which can cover current & future Cloud implementations

Data Model Standardization Scope & Alignment

- Scope
  - Object – DC, Tenant, Cluster, Host, VM etc.
  - Capabilities – Affinity/Anti-affinity, EPA attributes such as NUMA, CPU Pinning, Deployment Template (Heat, ARM etc.) etc.
  - Basic lifecycle operations - create VM (Server), delete VM etc. excluding networking - resource reservation (capacity) for Objects
  - Data collection & management – Resource reservation & Dynamic resource utilization for Objects
  - VNF Types (Control Plane, EPC or IMS Data Plane etc.) – can be specified in terms of capabilities and pre-specified or dynamic capacity from a infrastructure perspective
- Alignment
  - Beyond ONAP - ETSI NFV/OSM

LCM – Create VM (Server) example

Exemplary VM Compute/ Storage TOSCA Model for VNF Type X (e.g. Control Plane)

```
node_templates:
 VMD1:
  type: tosca.nodes.nfv.VMD.xxx
  properties:
   image: cirros-0.3.5-x86_64-disk
  capabilities:
   compute:
    properties:
     disk_size: 40 GB
     mem_size: 4096 MB
     num_cpus: 4
```

Smooth Transition from current OpenStack Heat, Azure & Azure Stack ARM Template – MC Southbound Plugin

- *SO -> MC* - Create VM instance of VNF type X using specified compute/memory/local storage parameters
- *MC -> SO* - New VM instance ID
- *SO -> MC* - Pass Heat or ARM Template with VM instance
- *Retrieve VM instance data*
- OpenStack
  - *Map to appropriate Flavor (represents size & EPA); a new Flavor is created if needed*
  - *Populate Heat Template & call OpenStack to create VM*
- Azure & Azure Stack
  - *Normalize and Round up to appropriate VM size (Ref. 1)*
  - *Intent-based mapping of EPA parameters*
  - *Populate ARM Template & call Azure/Azure Stack to create VM*
- *MC -> SO* –Send successful create VM (Server) response
- *MC -> AAI* – Store cloud specific (OpenStack, Azure etc.) data for MC instance – used for get and delete operations
- *MC – Delete VM instance*

Ref.1: https://docs.microsoft.com/en-us/rest/api/compute/virtualmachines/virtualmachines-list-sizes-region

LCM - Create VM (Server) EPA Asymmetric NUMA example

Exemplary VM EPA (Asymmetric NUMA) TOSCA Model for VNF Type Y (e.g. EPC or IMS Data Plane)

```
VMD1:
 capabilities:
  compute:
   properties:
    disk_size: 40 GB
    mem_size: 4096 MB
    num_cpus: 4
    numa_nodes:
     node0:
      id: 0
      vcpus: [0,1]
      mem_size: 1024 MB
     node1:
      id: 1
      vcpus: [2,3]
      mem_size: 3072 MB
```

Smooth Transition from current OpenStack Heat Template Azure & Azure Stack ARM Template – MC Southbound Plugin

- …
- *SO -> MC* - Create VM instance of VNF type Y using specified compute/memory/local storage parameters and asymmetric NUMA model
- *MC -> SO* - New VM instance ID
- …

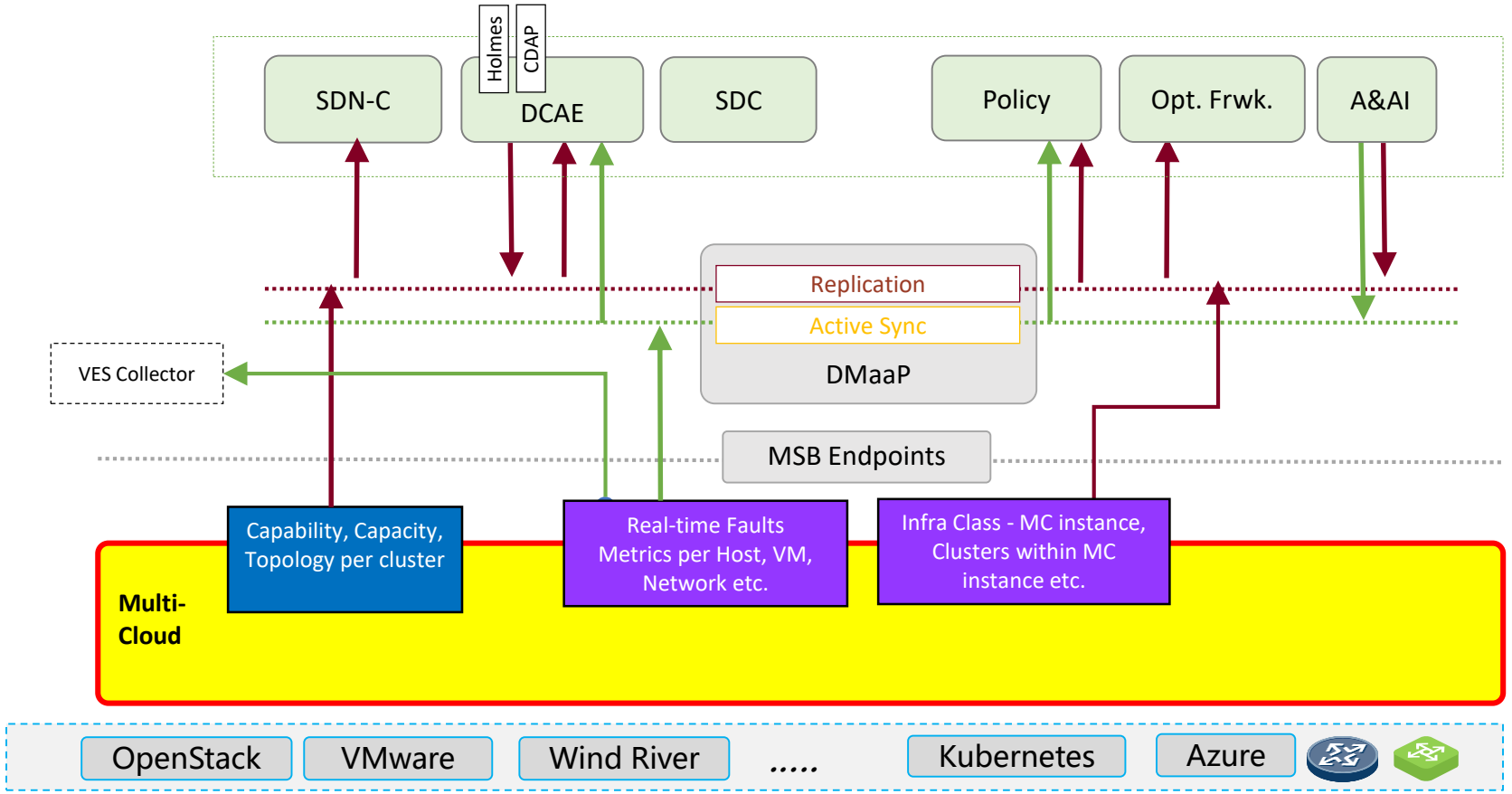
Scalability for Data Management

- Push latest aggregate data through asynchronous methods

Multi-Cloud Data Management Architecture

- **Aggregate Data (Tenant, Cluster etc.)** is key to hierarchical multi-site VNF placement solutions driven by OF, DCAE etc.
  - Data at atomic level (VM, Host etc.) does not scale

- **Asynchronous Push Model for Latest Data** enabled by DMaaP pub-sub active-sync or lazy replication across WAN is key to scale
  - Synchronous Poll model for volumetric data does not scale and often lands up using stale data



Adapted from ONAP Paris MC Workshop FCAPS Telemetry:
https://wiki.onap.org/download/attachments/11928197/ONAP-mc-fcaps.pdf?version=2&modificationDate=1506519214000&api=v2

Data Model Standardization & Scalability Example

Optimization Policy Example

- R1 vCPE use case – Illustrative sequence diagrams (https://wiki.onap.org/display/DW/Residential+Broadband+vCPE+Drafts+for+discussion?preview=%2F10783327%2F16005563%2FvCPE+Use+Case+-+Customer+Service+Instantiation+-+171103.pptx)
- Constraints used by OF
  - VBNG location is fixed based on subscriber
  - VG MUX to VBNG Data Center connectivity latency cannot exceed certain value
- Optimization Policy used by OF
  - Choose optimized multi cloud instance for the placement of VG MUX for a given subscriber based on the above constraints
- Information/Data model standardization example
  - **VNF Type** (EPC CP, PGW DP, SGW DP, BNG DP etc. – where CP is Control Plane and DP is Data Plane)
  - **Resource cluster – group of machines with identical HW configuration including EPA capabilities**
    - VNF Type directly maps to a resource cluster
    - <Host Aggregate> in OpenStack; <Host Aggregate, cluster> in VMware integrated OpenStack
  - **Tenant** (In this example, each VNF maps to a tenant in the infrastructure)
    - <Project> in OpenStack; <Tenant> in VMware integrated OpenStack
- Multi Cloud inputs used by OF
  - **Near-real-time stats per <tenant, resource cluster> at scale** using asynchronous push model using DMaaP

Next Steps

- Proposed presentation on standardized Information Model for Multi Cloud Objects and Capabilities in the upcoming ONAP modelling workshop
- Proposed talk on how standardized Multi Cloud Objects/Capabilities can be leveraged to drive consistent polices, analytics rules and optimization constraints in the upcoming ONAP developer forum
- Besides the common microservice design patterns, drive “data microservices” principles for accessing data stores through consistent intent-driven APIs in the community