Multi VIM/Cloud Evolvement for Carrier Grade Support

VMWare: Xinhui Li, Ramki Krishnan, Sumit Verdi
China Mobile: Lingli Deng, Chengli Wang, Yuan Liu, Yan Yang
ATT: Bin Hu
Wind River: Gil Hellmann, Bin Yang
Huawei: Gaoweitao, Jianguo Zeng, Chris Donley

Retro to R1

- Use case and integration support:
    - Keystone V2 vs. V3
    - Polling based alert emit
    - Unknown VM status and standalone event system
    - Runtime Only
    - Concurrency

- Multi Cloud for S3p
    - Improvement of Keystone Proxy
    - FCAPS modeling
    - Event/Alert/Meters Federation
    - Runtime + onboarding
    - Framework Scalability
    - Platform awareness
    - Workload orchestration

Alert/Event/Meter Federation Framework

- Different Cloud Backend -> different Alerts/Events/Meters, two roles:
  - Multi Tenancy workload
  - Admin

- Motivation:
  - Despite Alerts, Events and metrics are needed from VIM controller
    - Close the Service Resilience loop in ONAP needs underlying events (from not only resources but also vim controller)
    - Close the Auto-Scaling Resilience loop in ONAP needs underlying Meters/Alerts etc.
  - Aligning/translating FCAPS modeling is needed from different VIM controllers
  - Framework enhancement is needed
    - Allowing for meaningful close loop for handling time-sensitive event/alerts
    - Allowing streaming mechanism for handling accuracy-sensitive metrics
    - Extending ves agent implemented in R1 which only emit vm abnormal alert for more period data collection
  - Visualization on Portal

FCAPS Modeling

- Different categories of Data: resource status, alerts, performance, and so on
  - Interface protocol
  - Data format
  - Metrics
  - Logs

- All the data should be gotten based on the same pre-condition
  - Period or event?
  - NTP and facilitation requirements
  - Delay sensitive
  - Different data sources

- Need spec and modeling efforts to formulate these detentions

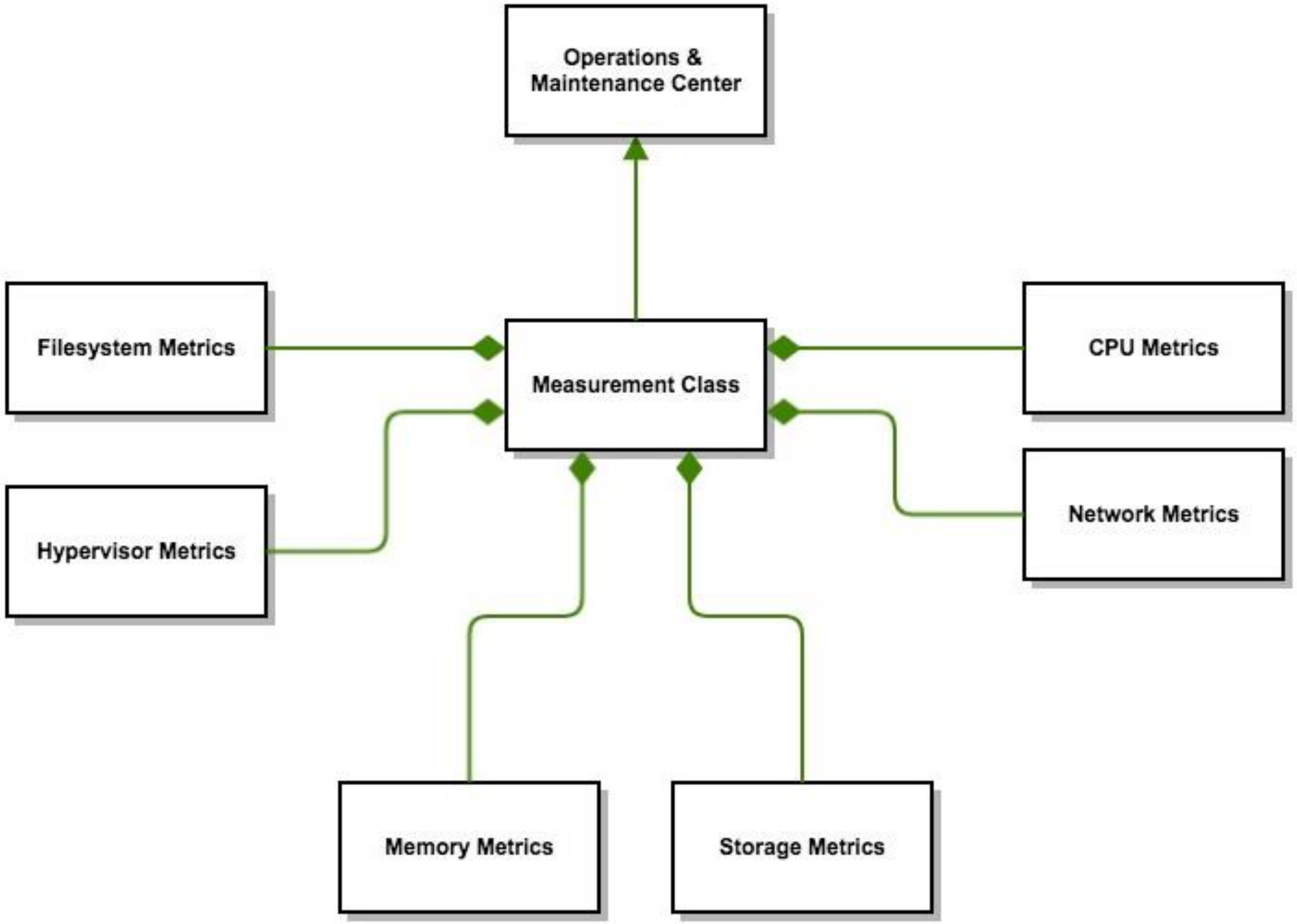
Alert/Event/Meter Federation Framework

- Different Cloud Backend -> different Alerts/Events/Meters, two roles:
  - Multi Tenancy workload
  - Admin

- Use cases:
  - Close the Service Resilience loop in ONAP needs underlying events(from not only resources but also vim controller)
  - Fcaps propose the requirements for monitoring and management of Alerts/Events/Meters
  - Visualization on Portal

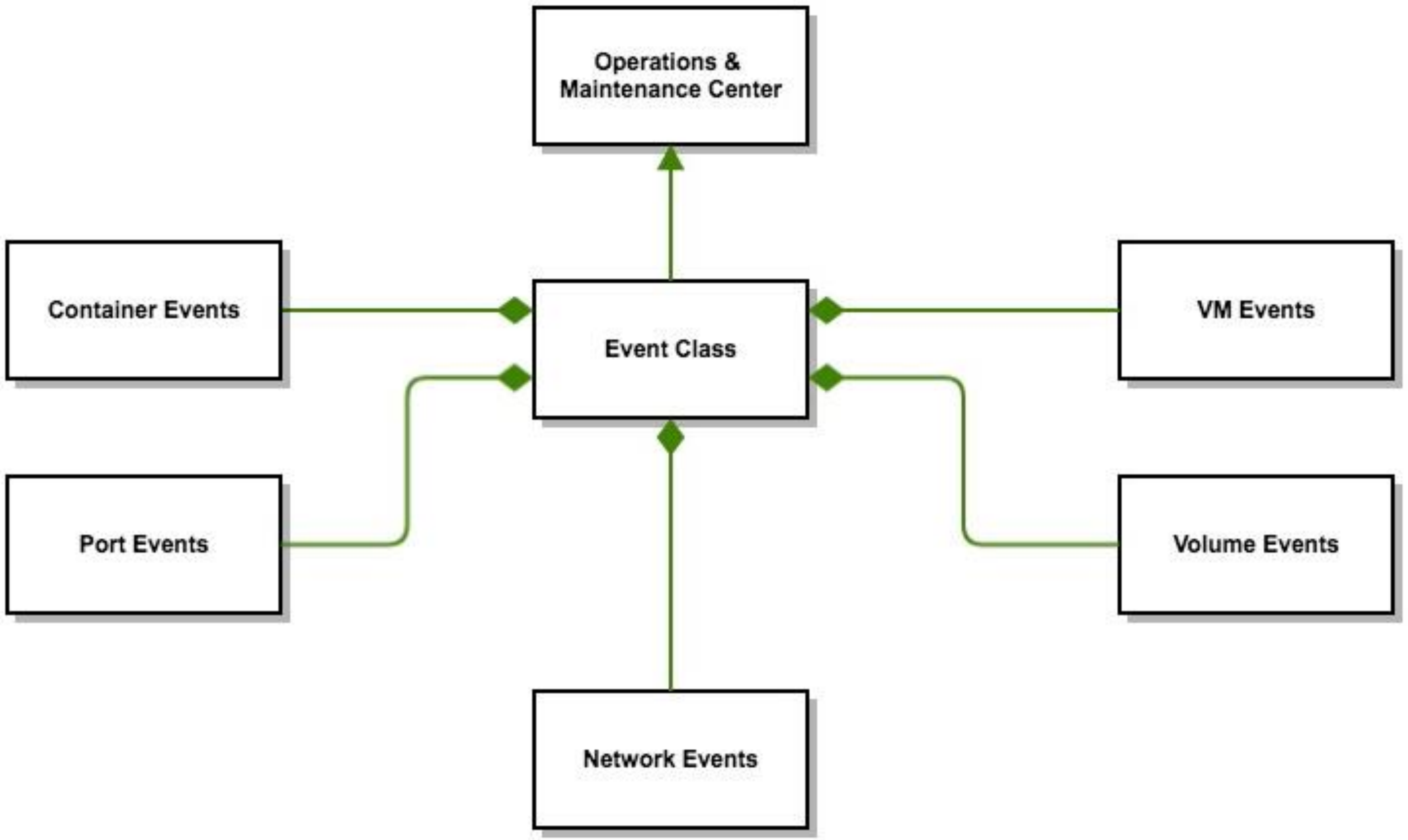
Measurement Class

Measurement class represent the metrics collected from various of sources. It could carry the cpu metrics, memory metrics or fs metrics collected in a period of time.



Event Class

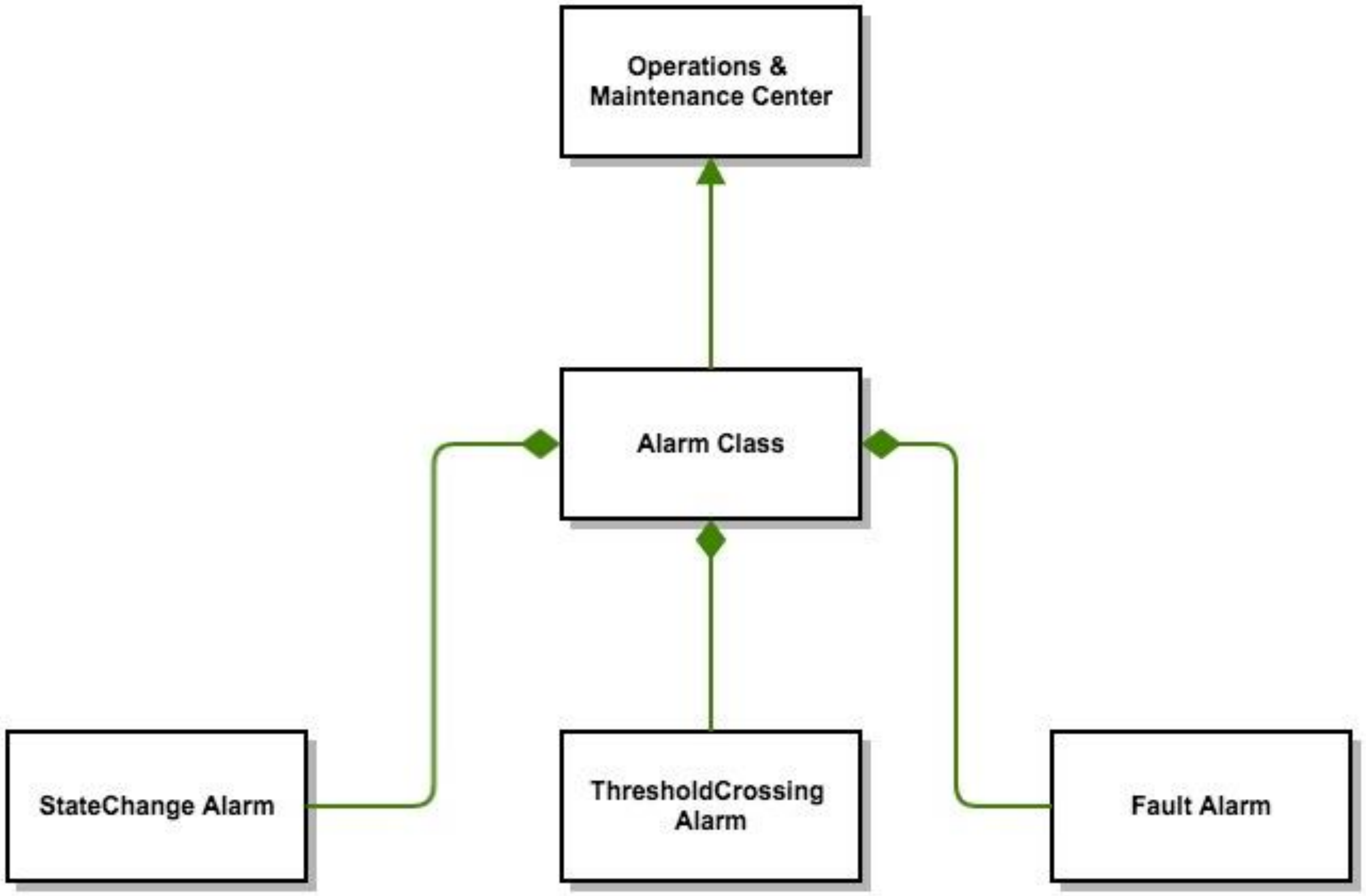
Event class represent the VIM specific events. For example, Virtual Machine poweroff events, Port creation/deletion events.



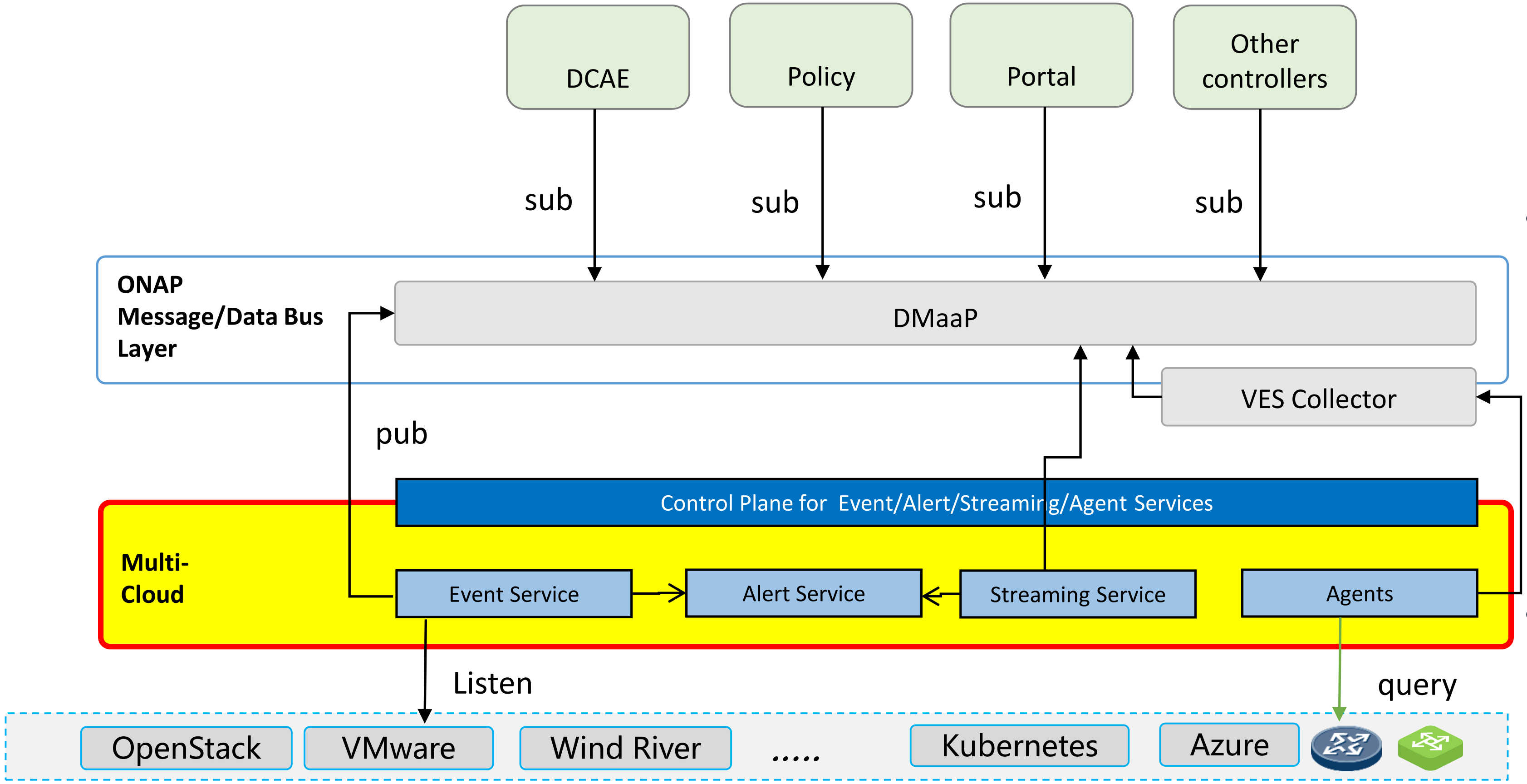
Alarm Class

Alarm class represent the alarms triggered by VIM. There are some basic alarms like:

- Fault Alarm
- Threshold Crossing Alarm
- State Change Alarm

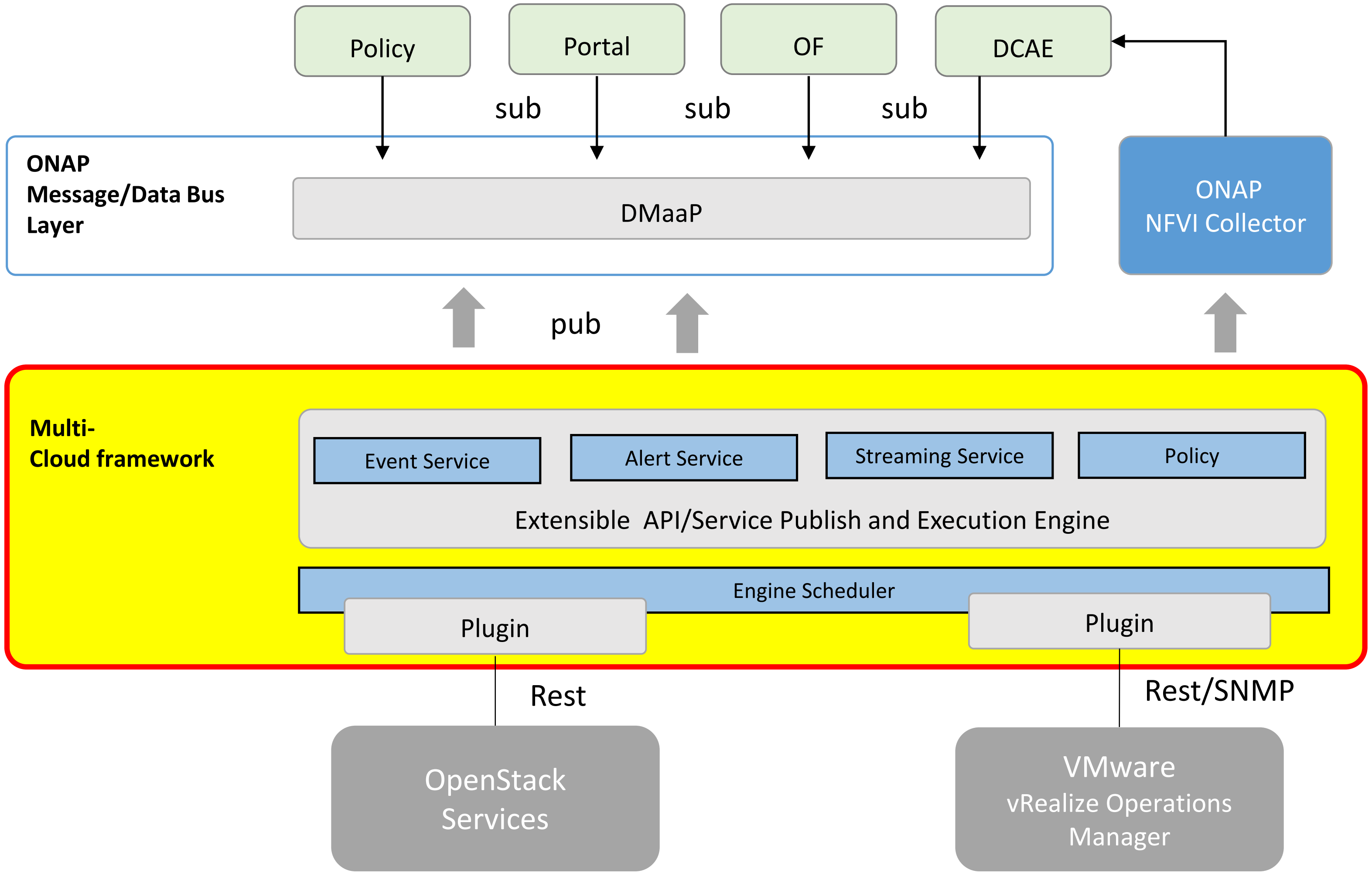


Alert/Event/Meter Federation Framework



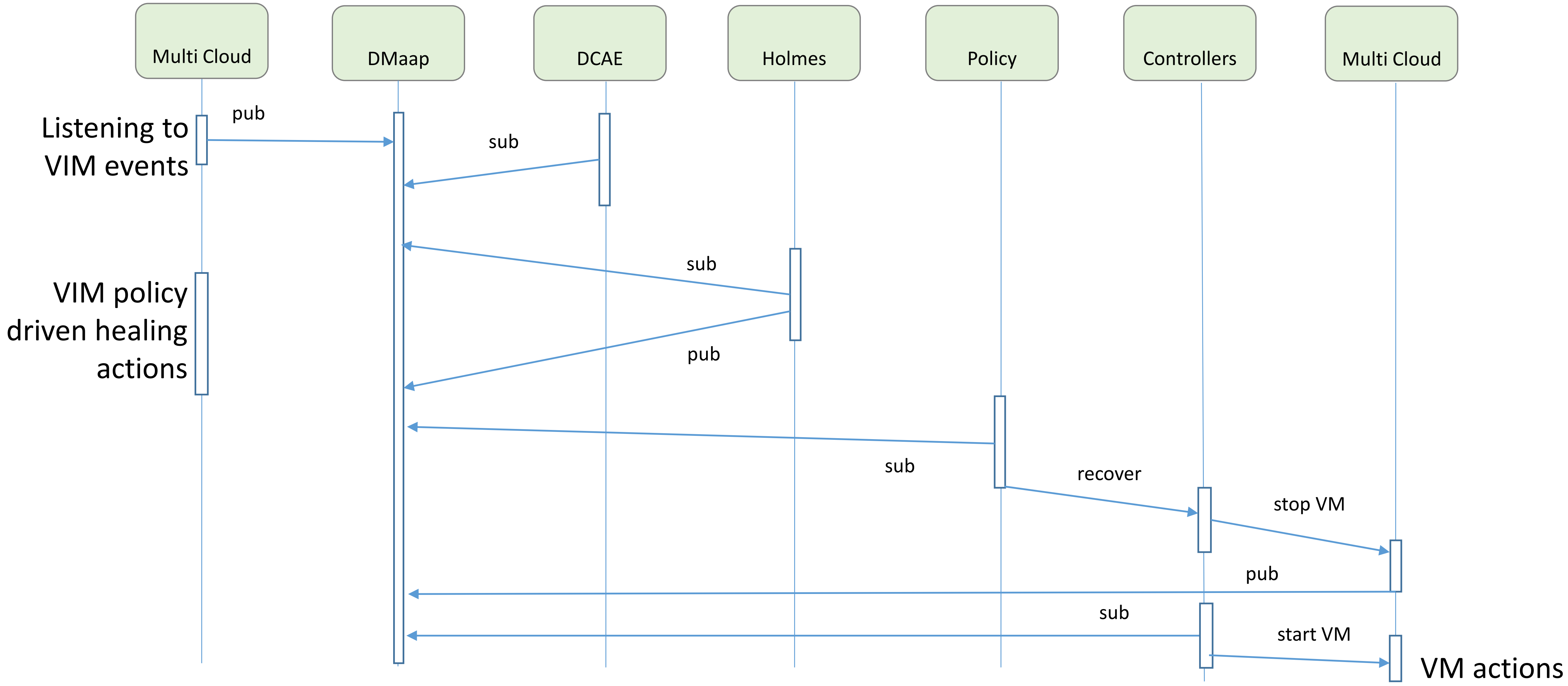
- **Event Service**
  - Federate events from different VIM providers with ONAP Message/Data bus services
  - Allow to be configured by the control plane about listener and endpoints
  - Not only events of different backend clouds, but also events from VIM controllers
- **Streaming Service**
  - Federate meters from different VIM providers with ONAP Message/Data bus services
  - Allow to be configured by the control plane about gate rate and water mark
  - Achieve a ideal long term output rate which should be faster or at least equal to the long term data input rate
- **Alert Service**
  - Alert comes from meters or events, or pre-defined in different backend Clouds
  - Allow to be customized

Operational Intelligence Federation Framework

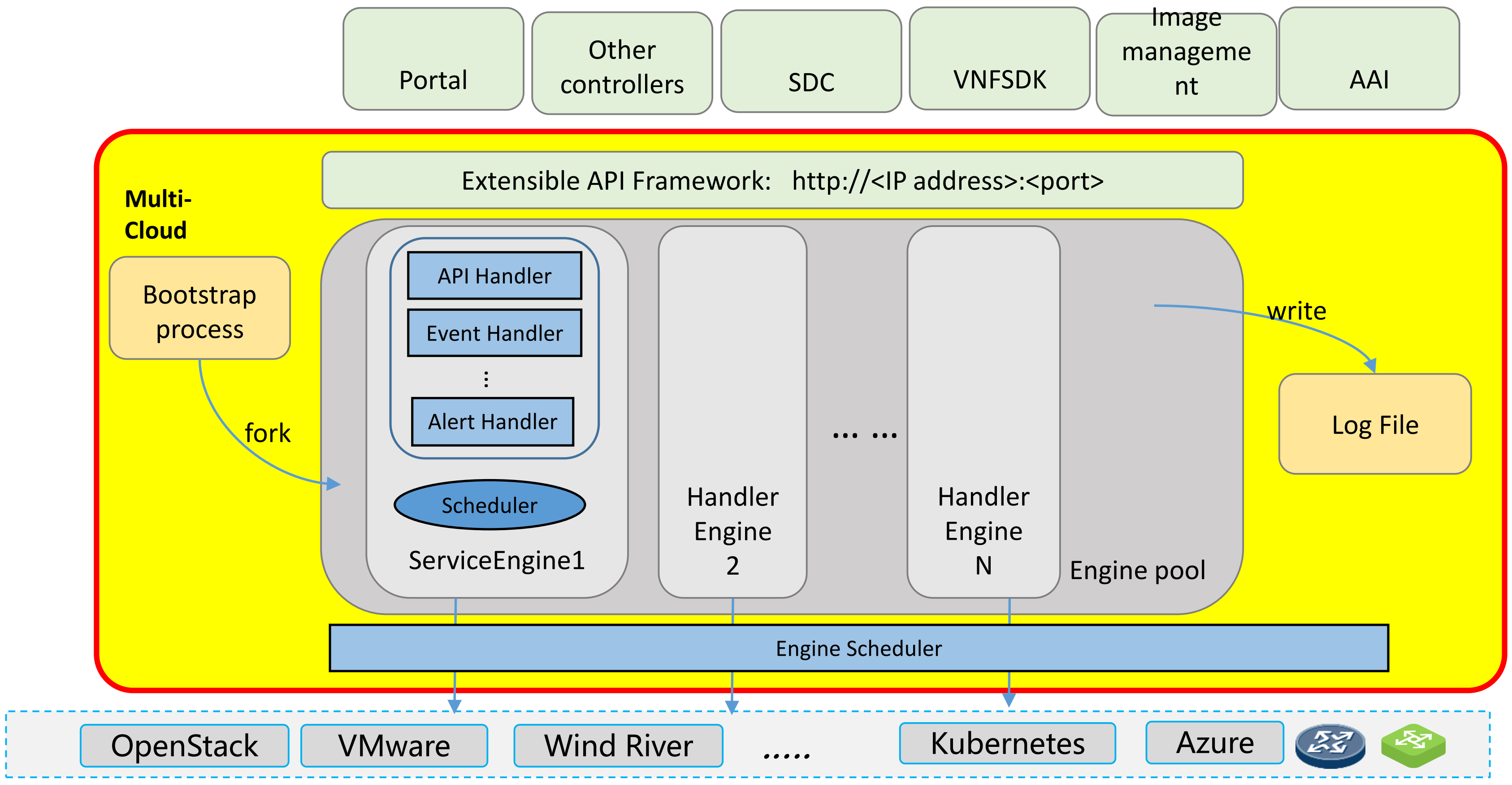


- **Extensible MC framework**
  - Pluggable framework
  - Intelligence contained
  - Information models standardized
  - Customizable to work with upper layers
  - Policy-driven data distribution
- **Compensate each other with different Telemetry:**
  - OpenStack control plane service status
  - Underlying OS/hypervisor status
  - Customizable by policy

Alert/Event/Meter Federation Flow Chart



Multi VIM/Cloud Framework Improvement



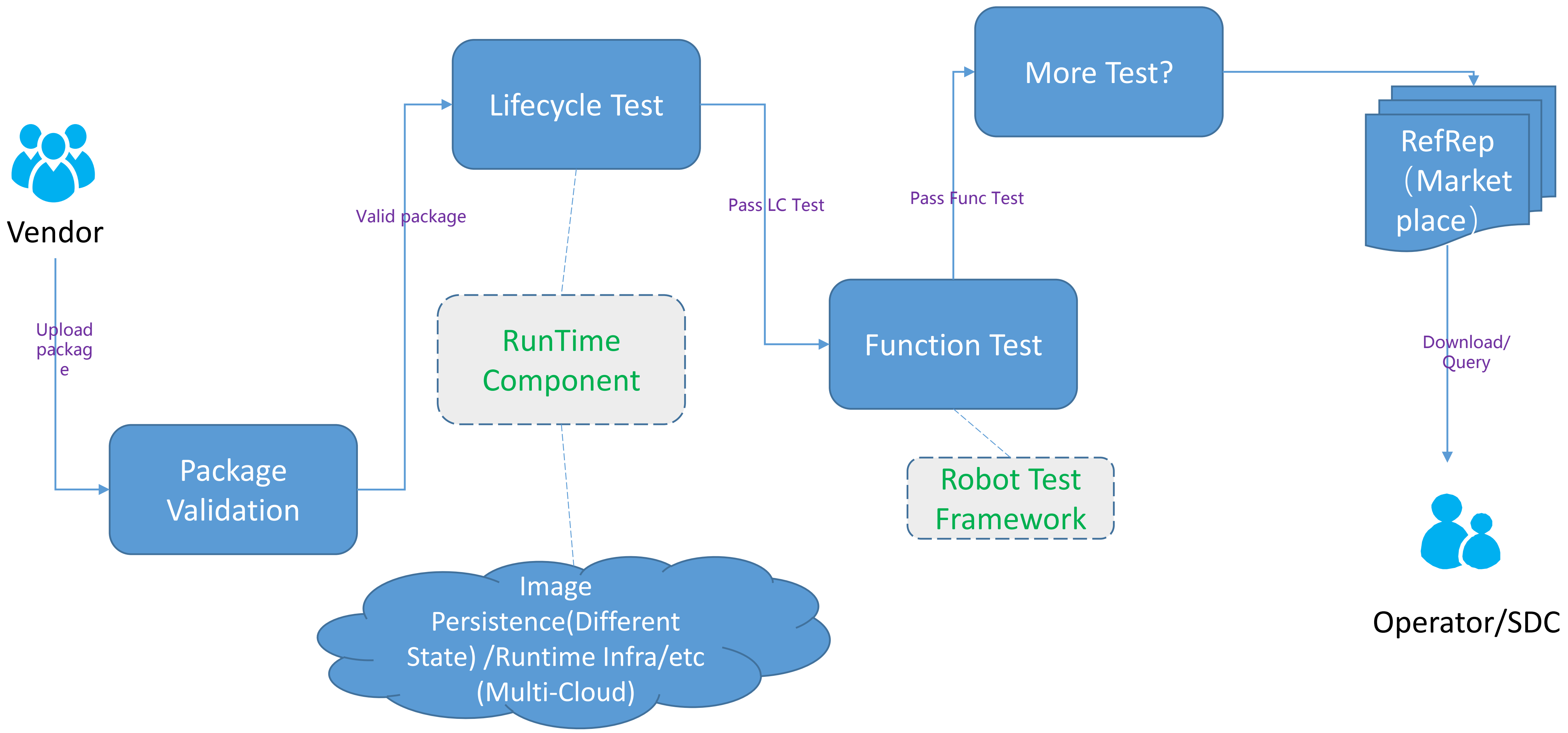
- **Performance improvement**
  - The number of Handler Engines is configurable, and is the number of CPU cores by default.
  - Each Handler Engine will use scheduler to do multiple jobs.
- **Stability improvement**
  - Bootstrap process will watch handler engine and respawn it if any exits unexpectedly.
  - Handler engines are run as process so that memory and resources are isolated.
  - Store log into files
- **Scalability improvement**
  - Bootstrap process uses fork to create multiple Handler Engines
  - Multiple Handler Engines share a single socket with bind to the <IP address>:<port> of API server
  - Extend northbound API by using yaml file

Support Onboarding

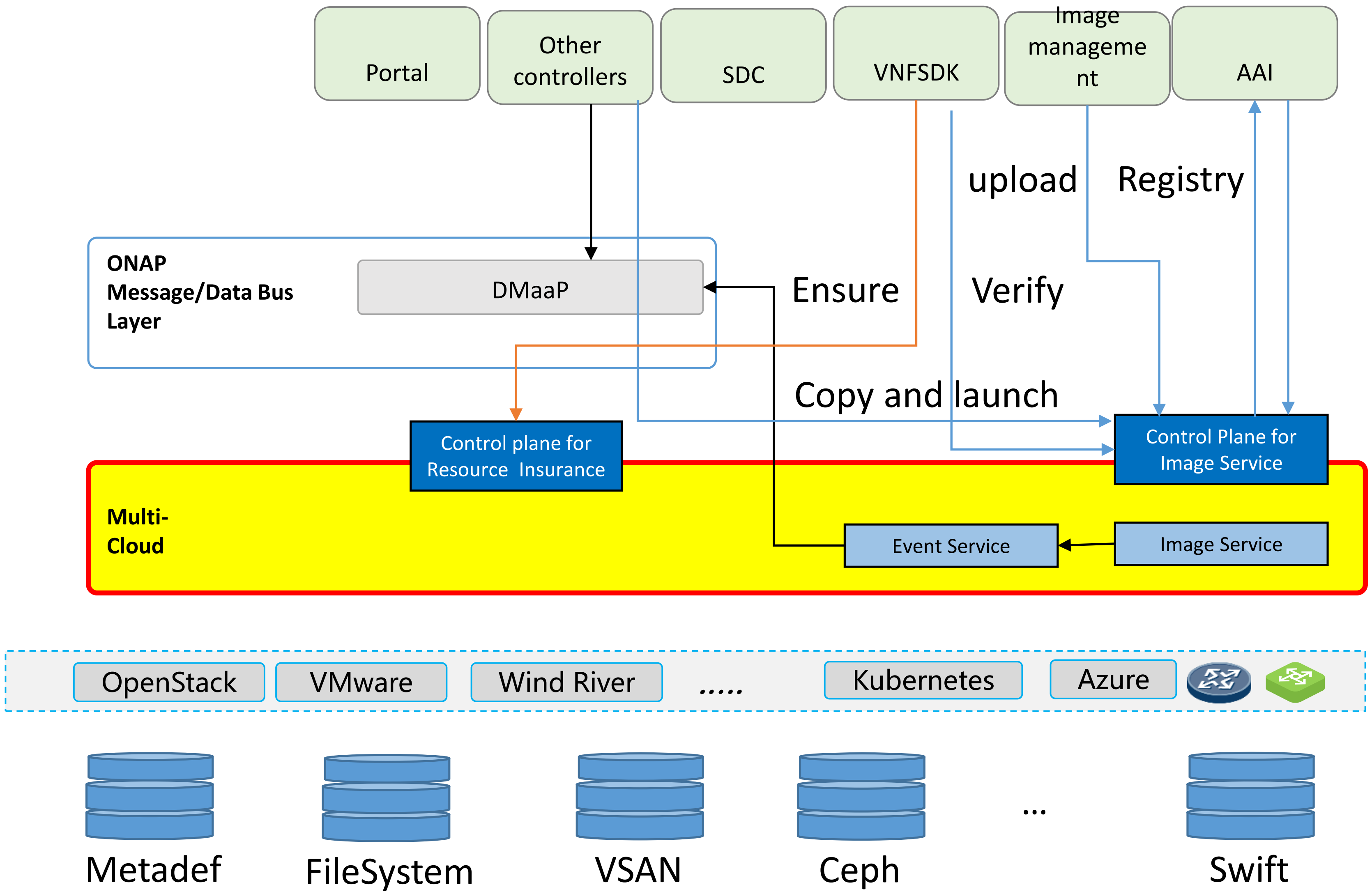
- VNF on-boarding and upgrade are:
  - Current No formal way for a VNF to ask for resources and ensure they can be and will be granted at install and reserved thru operation
  - Limited security, isolation, scalability, self healing – VIM & VNF specific
  - Efficiency: in many cases, no other VNF runs on the same platform
  - Operator, VNF vendor and VIM specific – procedures are specifically tailored to each VIM infrastructure, operator and VNF vendor

- Multi Cloud will help:
  - Automatically ensure capacity and capability of required VIM & infrastructure resources
  - Verify/Validate that
    - VIM Image （qcow2 or vmdk)
    - VIM requirements can be met

Onboarding


Vendor
Upload package
Package Validation
Valid package
Lifecycle Test
RunTime Component
Image Persistence(Different State) /Runtime Infra/etc (Multi-Cloud)
Pass LC Test
Function Test
Robot Test Framework
Pass Func Test
More Test?
RefRep（Market place）
Download/ Query
Operator/SDC

Multi VIM/Cloud Extention for Onboarding



- **Image service to handle**
  - Different kinds of storage for image file and meta reservoir
  - Support both Docker or VM mages
  - Help VNF onboarding and runtime optimization
- **Primary operations**
  - Image upload/download
  - Image registry/un-registry
  - Copy from one VIM to another VIM
  - Copy from one data store to another data store
  - Launch instances
- **Transparent to the up layer**
  - Auto handling across vim copy and capacity management

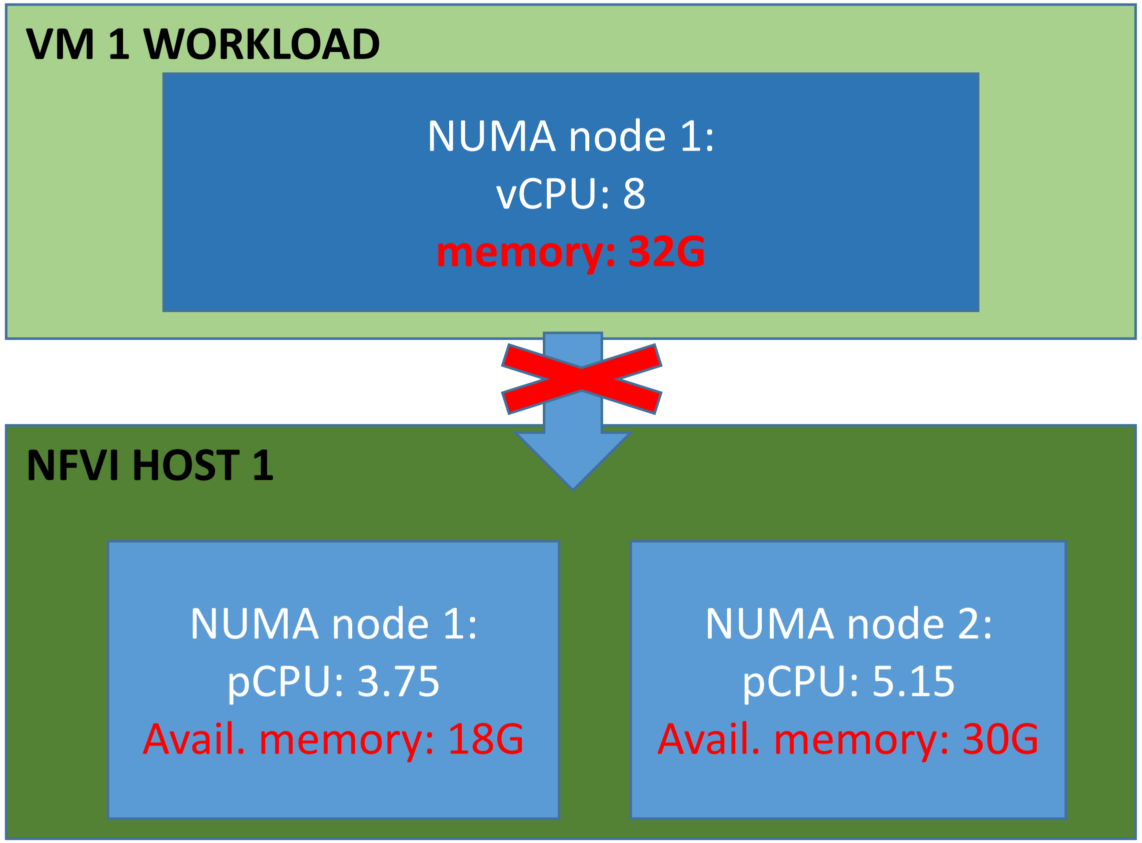
NFVI Host NUMA resource exposure

- NUMA awareness
  - Reduce memory access latency
  - Fit VM workload NUMA requirement to available NFVI Host NUMA resource
- NUMA requirement from VM workload
  - How many NUMA nodes
  - vCPU list for each NUMA node
  - Memory size for each NUMA node
- NUMA resource of the NFVI
  - NFVI Host list with NUMA awareness capability
  - Number of NUMA nodes
  - pCPU list for each NUMA node: total and available
  - Memory size/range for each NUMA node: total and available

Mismatching NUMA requirement to NUMA resource, case 1

- VNF usually consumes big chunk of memory
- Un-balanced usage of NUMA resource in NFVI hosts
- Waste usage of memory usage
- What usually observed: The avail. memory are more than request, but no NFVI host could be placed for a VNF

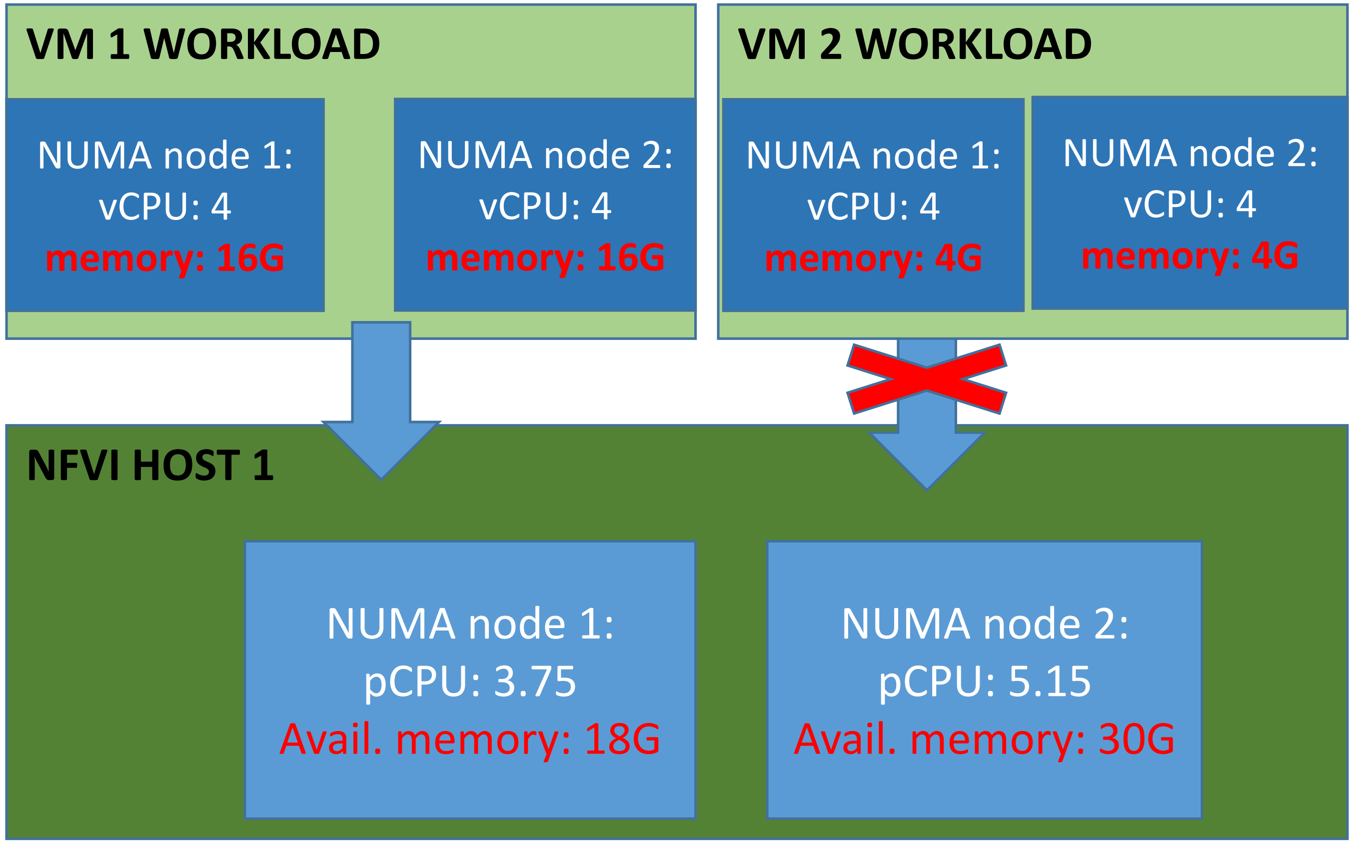
Case 1: Single NUMA node request memory more than avail. memory of any single NUMA node resource, but less than total avail. Memory resource



Mismatching NUMA requirement to NUMA resource, case 2

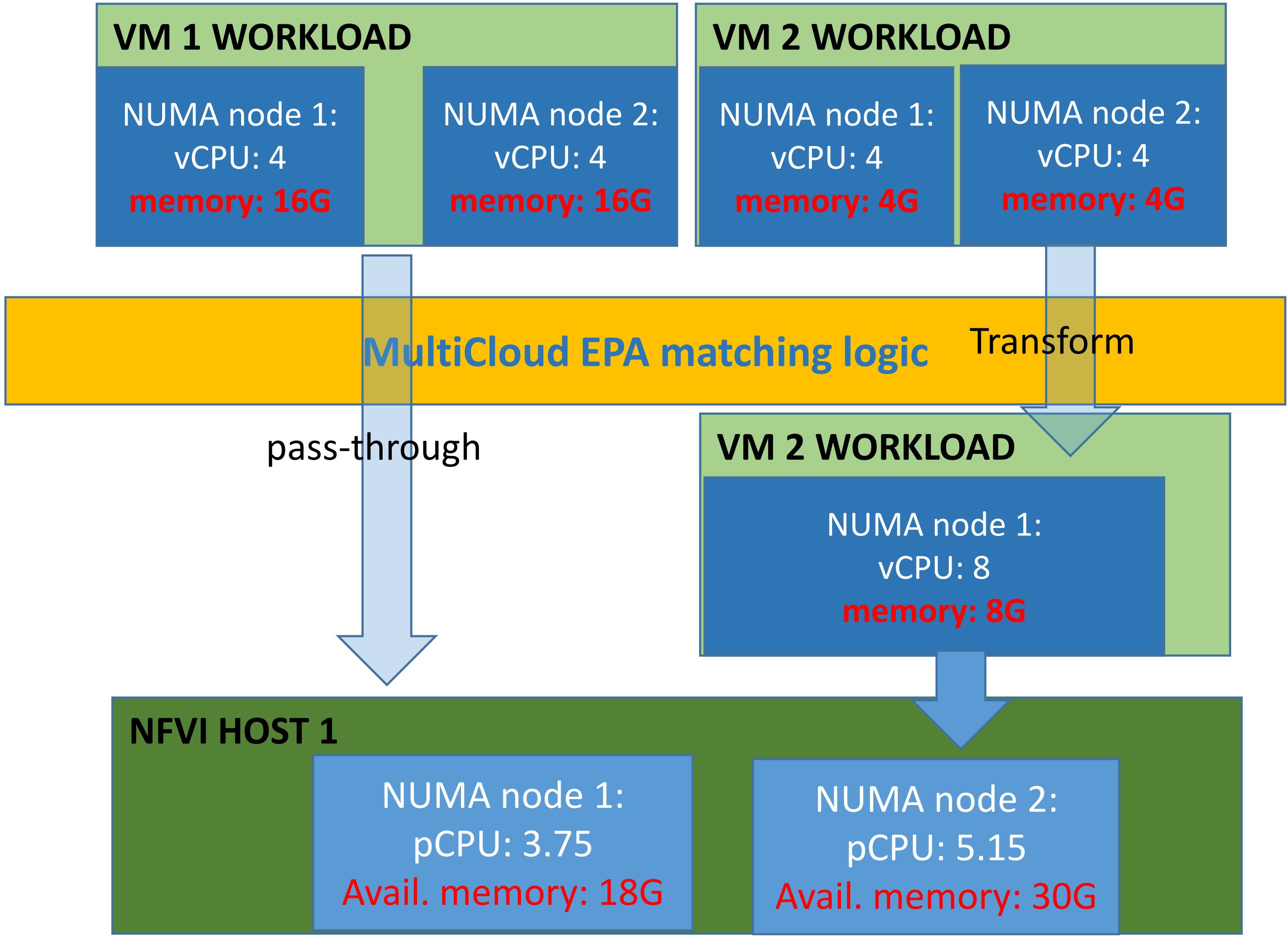
- There is no single predefined NUMA requirement fit for dynamically changed NUMA resource

Case 1: Single NUMA node request memory more than avail. memory of any single NUMA node resource, but less than total avail. Memory resource

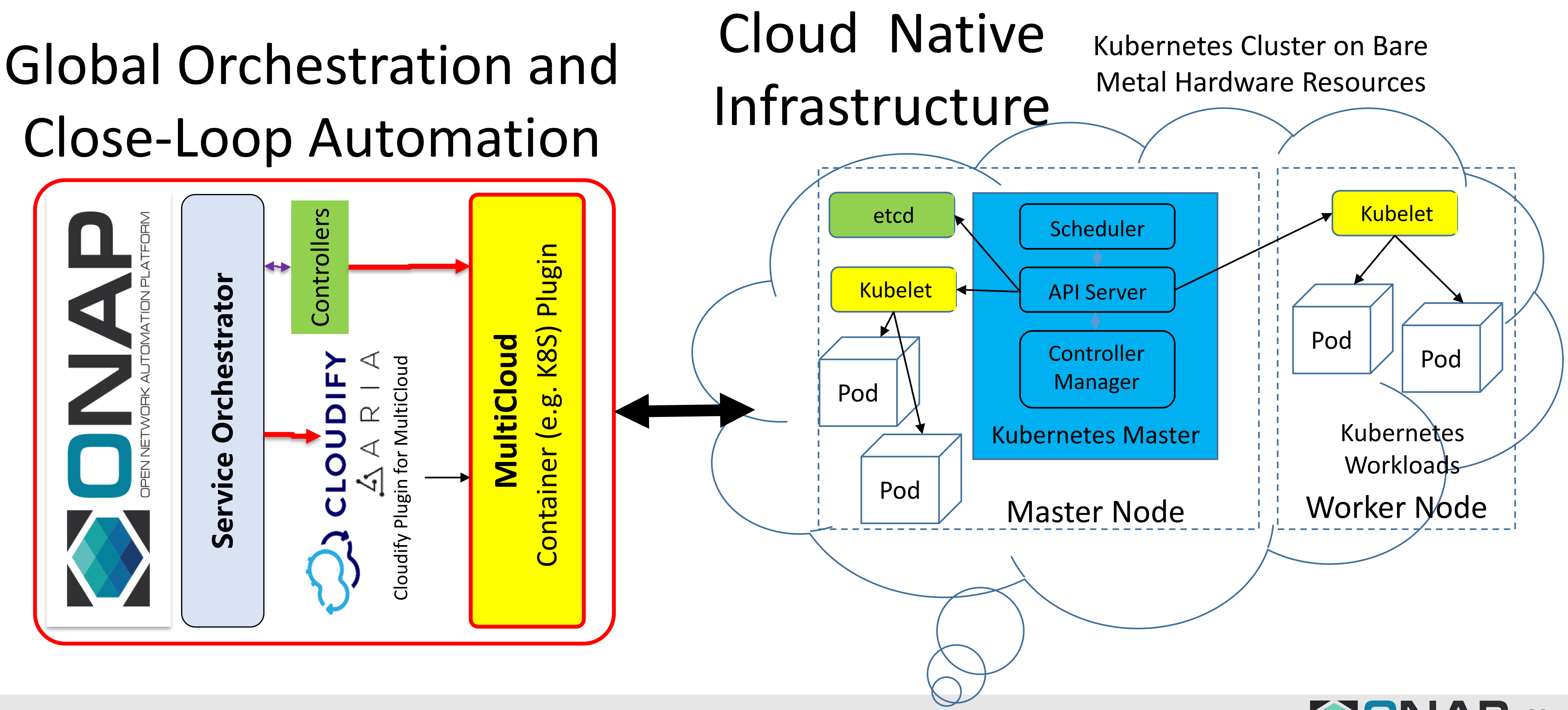


ONAP MultiCloud Tune the NUMA requirements

- MultiCloud tune the NUMA requirement to fit the VM workloads into NFVI Host NUMA resource
- NFVI Host NUMA resource must be exposed/updated into ONAP data store



Enabling Cloud Native Workloads Orchestration via MultiCloud

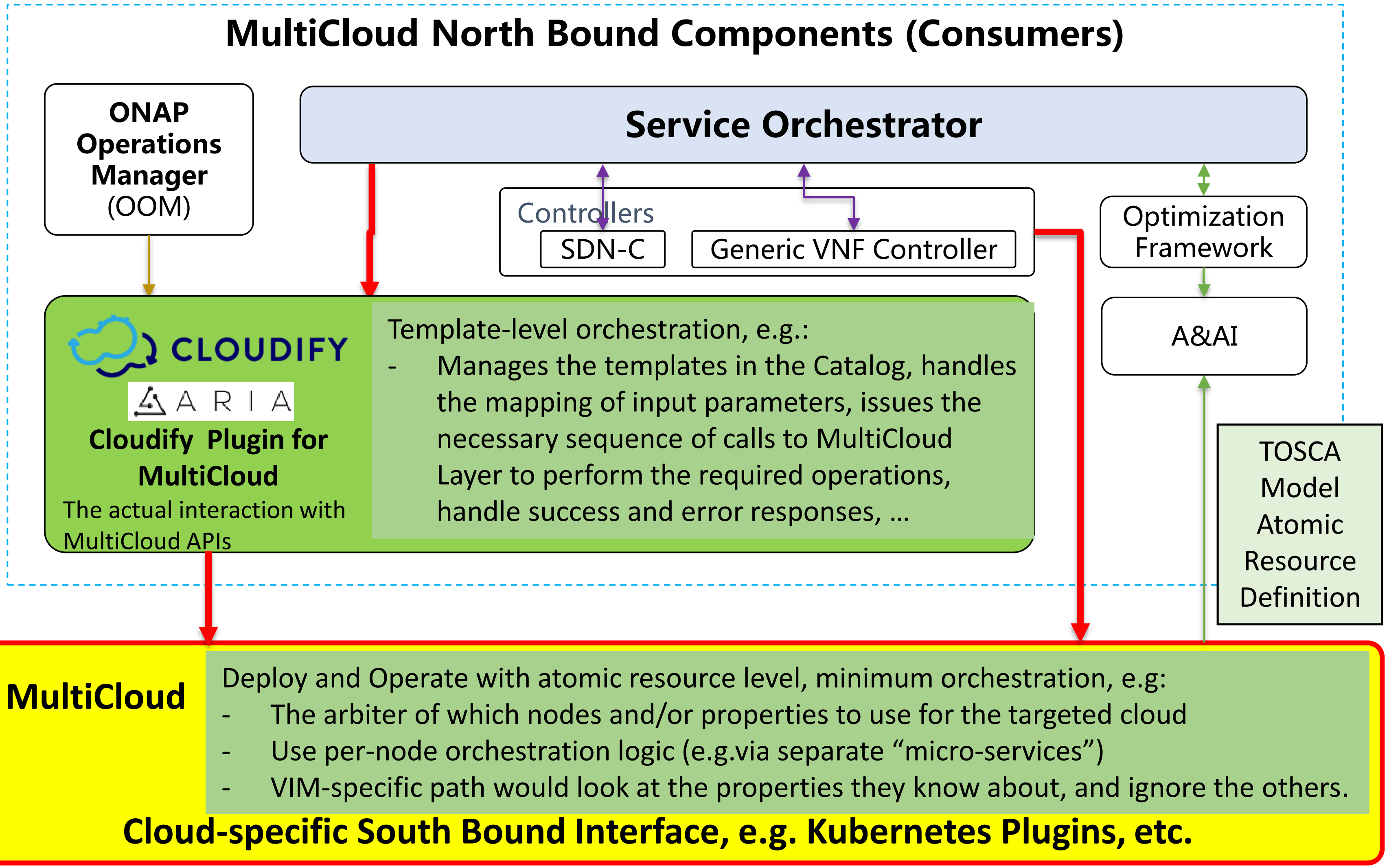


Enabling Cloud Native Workloads Orchestration via MultiCloud

ONAP
OPEN NETWORK AUTOMATION PLATFORM

Atomic Resource Definition:
- Define new data and/or node types to represent individual cloud resources in cloud-agnostic fashion
- Custom data/node types are included for EPA etc.

Cloudify Plugin for MultiCloud:
- A generic, thin API proxy
- Define the superset of all properties of each cloud-agnostic node type
- A single service template can support multiple clouds by “model + inputs”



Backup Slides

Enabling Cloud Native Workloads Orchestration via MultiCloud

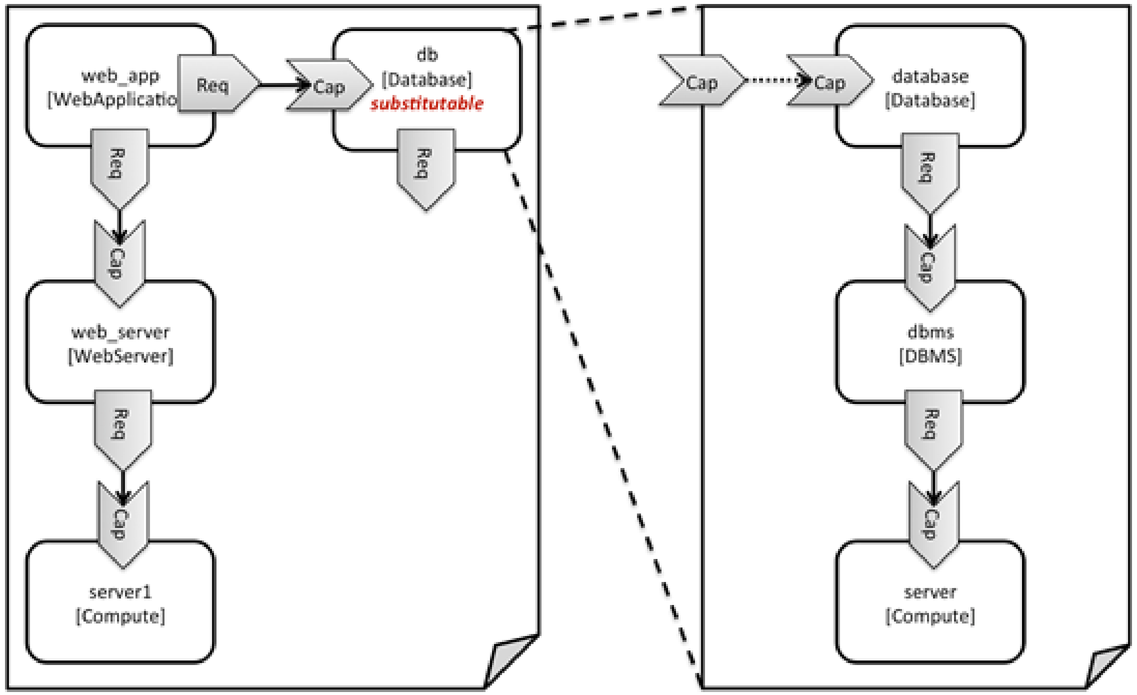
- Decoupling template level operations from atomic resource level operation
- Atomic Resource Definition
  - TOSCA data and/or node types to represent the individual cloud resources in cloud-agnostic fashion. It also includes custom data and/or node types per cloud type
  - Define the superset of all properties of each cloud-agnostic node type
- Template-level Orchestration
  - A single template can support multiple clouds because of “model + inputs with runtime resolution”
  - Orchestration is primarily done within TOSCA Orchestrator on the north of MultiCloud Layer
  - E.g.: manages the templates in the Catalog, handles the mapping of input parameters, issues the necessary sequence of calls to MultiCloud Layer to perform the required operations, handle success and error responses, etc.
- Minimum Orchestration at Atomic Resource Level in MultiCloud Layer
  - MultiCloud Layer would be the arbiter of which nodes and/or properties to use for the particular target cloud
  - Per-node orchestration logic can be used in MultiCloud Layer (e.g.via separate “micro-services”), and the VIM-specific paths would look at the properties they know about, and ignore the others.

Enabling Cloud Native Workloads Orchestration via MultiCloud

Service Orchestrator

CLOUDIFY

ARIA




```
tosca_definitions_version: tosca_simple_yaml_1_0

topology_template:
  description: Template of an application connecting to a database.

  node_templates:
    web_app:
      type: tosca.nodes.WebApplication.MyWebApp
      requirements:
        - host: web_server
        - database_endpoint: db

    web_server:
      type: tosca.nodes.WebServer
      requirements:
        - host: server

    server:
      type: tosca.nodes.Compute
      # details omitted for brevity

    db:
      # This node is abstract (no Deployment or Implementation artifacts on create)
      # and can be substituted with a topology provided by another template
      # that exports a Database type's capabilities.
      type: tosca.nodes.Database
      properties:
        user: my_db_user
        password: secret
        name: my_db_name
```

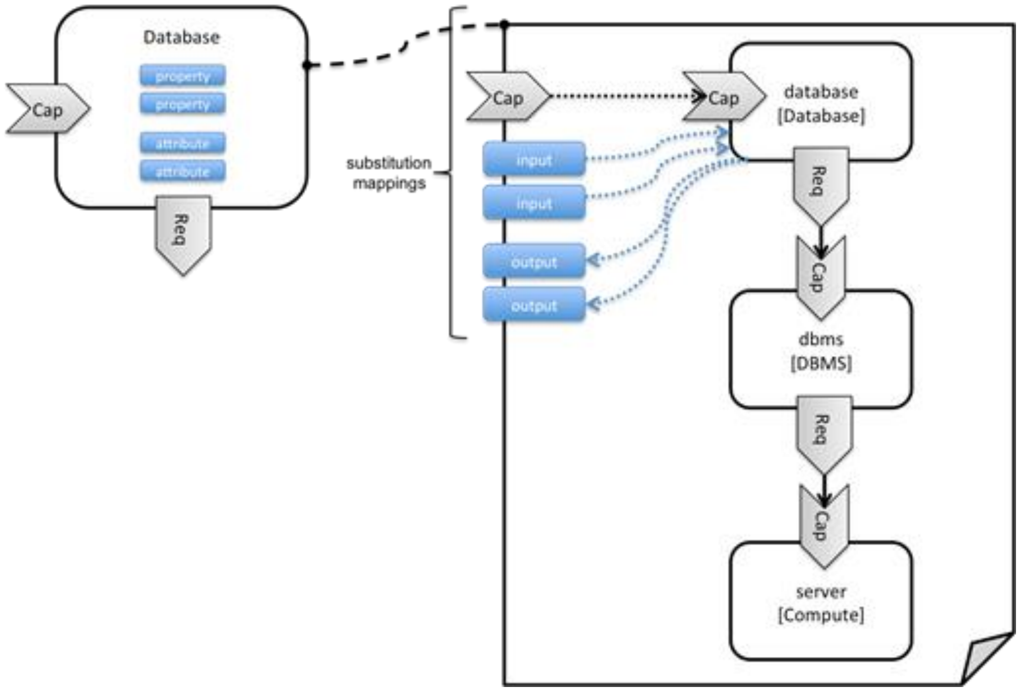
Example cited from [TOSCA-Simple-Profile-YAML-v1.1].

Enabling Cloud Native Workloads Orchestration via MultiCloud

Service Orchestrator

CLOUDIFY

ARIA




```yaml
tosca_definitions_version: tosca_simple_yaml_1_0

topology_template:
  description: Template of a database including its hosting stack.

  inputs:
    db_user:
      type: string
    db_password:
      type: string
    # other inputs omitted for brevity

  substitution_mappings:
    node_type: tosca.nodes.Database
    capabilities:
      database_endpoint: [ database, database_endpoint ]

  node_templates:
    database:
      type: tosca.nodes.Database
      properties:
        user: { get_input: db_user }
        # other properties omitted for brevity
      requirements:
        - host: dbms

    dbms:
      type: tosca.nodes.DBMS.MySQL
      # details omitted for brevity

    server:
      type: tosca.nodes.Compute
      # details omitted for brevity
```

Enabling Cloud Native Workloads Orchestration via MultiCloud

Service Orchestrator

CLOUDIFY ARIA

Cloudify Plugin for MultiCloud:
The actual interaction with MultiCloud APIs

- Issues the sequence of calls to MultiCloud Layer to perform the required operations
- Handle success and error responses

Template-level orchestration, e.g.:

1. Instantiate a Compute host A (to host a database server)
2. Deploy MySQL on the Compute host A
3. Create a Database on MySQL by getting the input of database name, username and password, and input of artifacts of database content
4. Instantiate a Compute host B (to host the web server)
5. Deploy web server on the Compute host B
6. Deploy the web application on the web server
7. Configure the web application to connect to the Database

MultiCloud

Deploy and Operate with atomic resource level, minimum orchestration, e.g:

- Find a cloud, if not specified, with MySQL and Apache web server capability. E.g. a K8S cluster
- Prepare deployment profile, including specific capability request, and affinity rules
- Deploy and configure accordingly through Kubernetes Plugin.

Cloud-specific South Bound Interface, e.g. Kubernetes Plugins, etc.