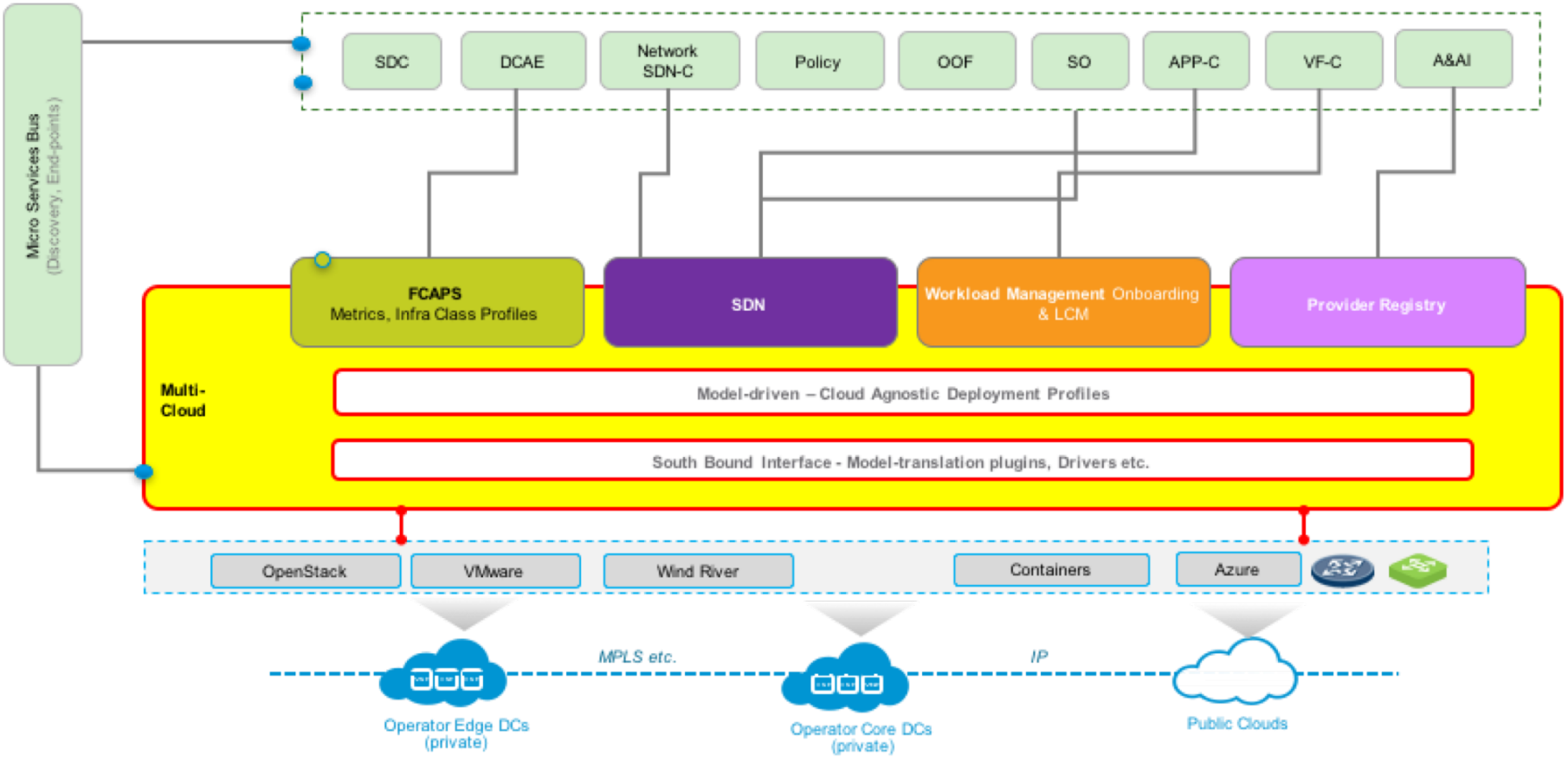
跨云管理

VMware: Xinhui Li, Ethan Lynn, Bin Sun, Liang Ke

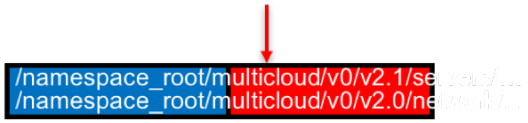
Architecture of Multi VIM/Cloud



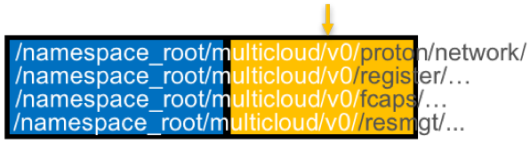
Multi Cloud R1 API

- All API starts with **MultiCloud Name Space**, followed by **functional module name space**

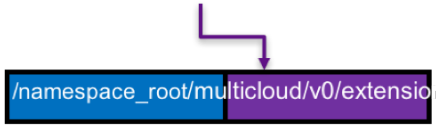
- Support existing **OpenStack APIs** as default functional modules. Minimal code changes to existing ONAP modules that already use OpenStack.



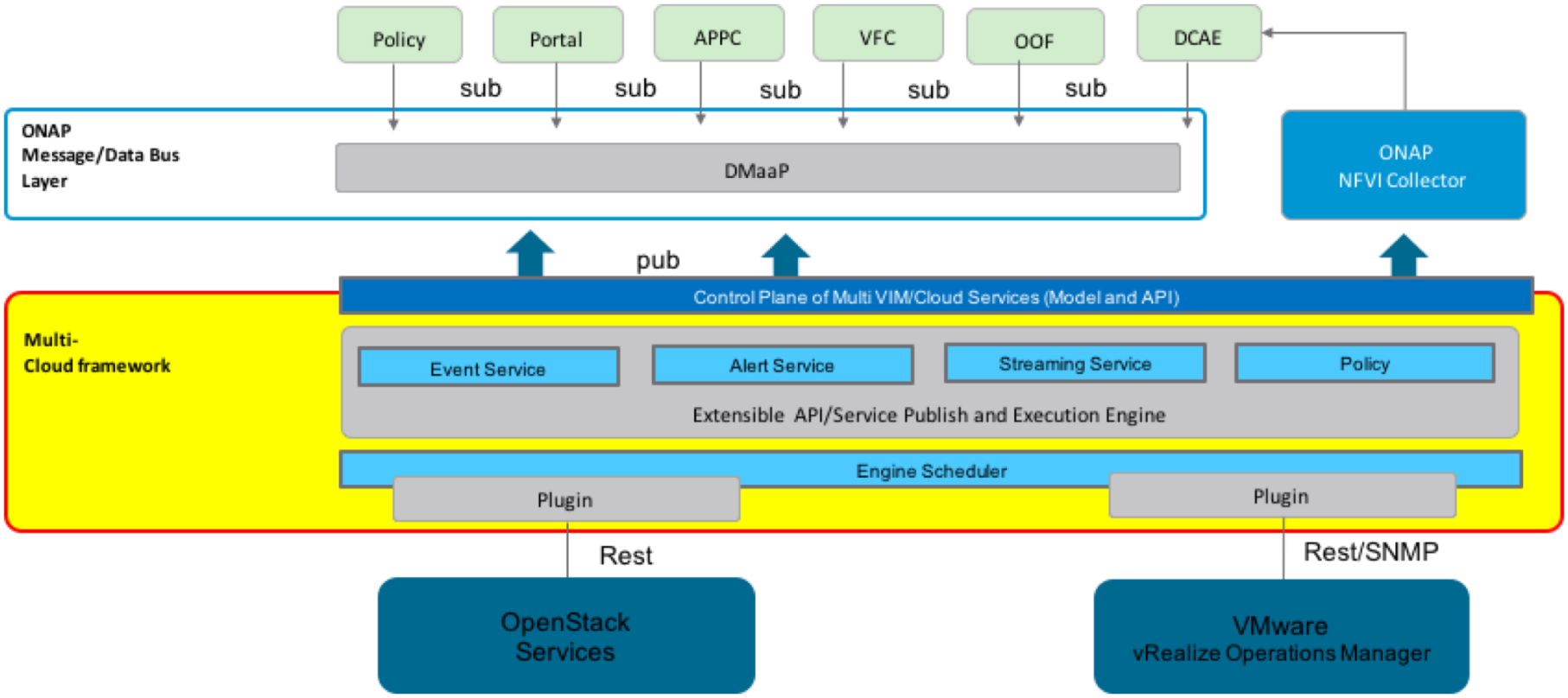
- New name space for **common cloud functionality**.



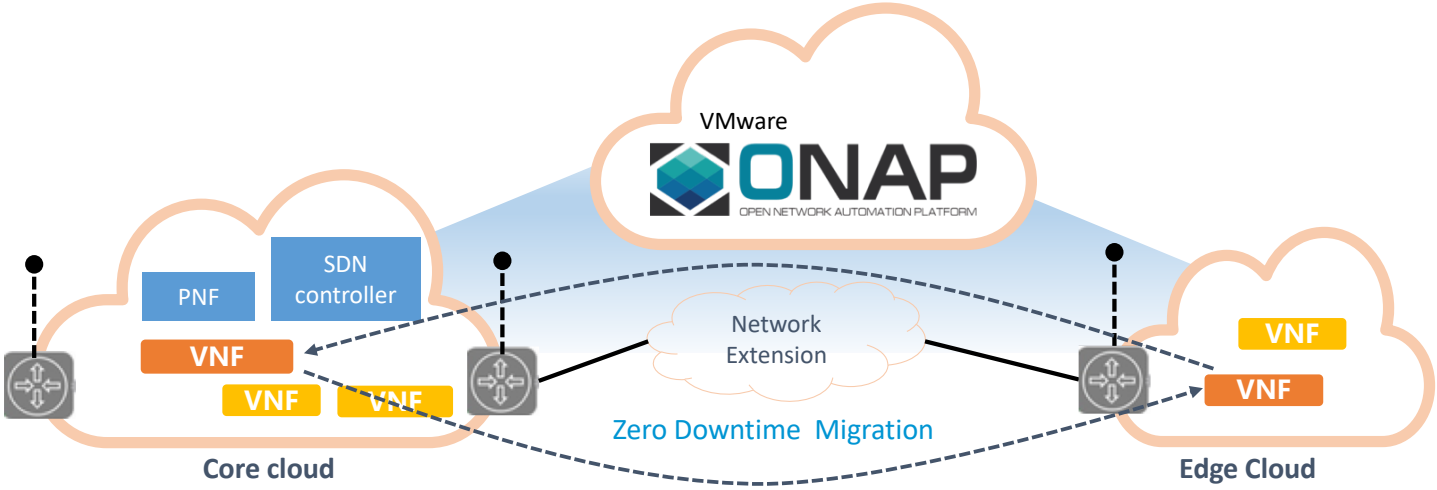
- Also support **3rd party extension** beyond common functionalities



R2 Event Federation



R3 Across-Clouds Management and Replication



Onboarding

ETSI compliant VSP onboarding

Instantiation

Multiple VIM/Cloud provisioning and VNF level protection

Seamless replication

No downtime during replication and recovery

Agenda

- Kick-off
- Instantiation
- Closed Loop
- Development and Diagnosis

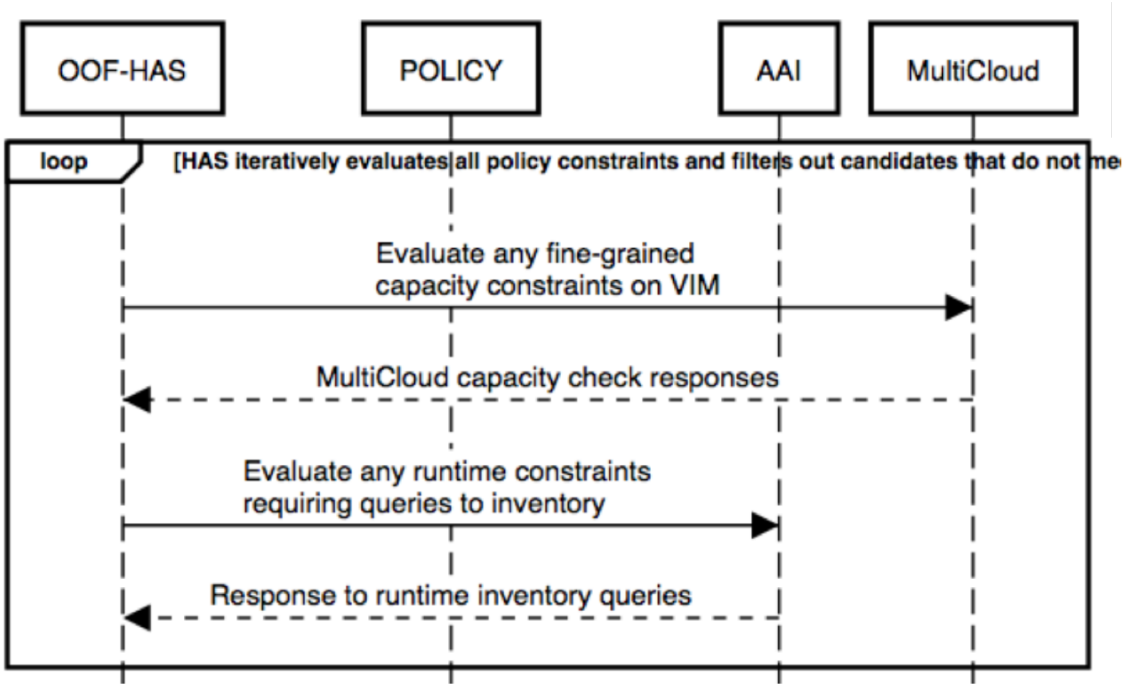
Current

Capacity Check API

- Input is vCPU/Mem/Vol requirements, Output is suitable VIMs

HPA Discovery

- MultiCloud could discover HPA capabilities on VIM registration and generates into A&AI



Improvement – check_vim_capacity

- Check_vim_capacity
- Input

```
{
  "vCPU": int,  // number of cores
  "Memory": float,  // size of memory, GB
  "Storage": int, //GB
  "VIMs": array  // VIMs OOF wish to check with
}
```

- Outp

```
{
  "VIMs": array  // VIMs satisfy with this resource requirement
}
```

- Only accept a list of candidate VIMs, could not check against all VIMs.
  - An empty list indicates to check against all VIMs

Improvement - Up to date cloud capacity

Cons of check_vim_capacity API .

- Check_vim_capacity shadows the capacities.
- Could only check through cloud native APIs, like openstack API

Improvement

- New API to expose those, like v0/cloud_capacities ?
- Regularly push to DMaaP, like vesagent do? Who will consume? How to avoid polling?

Improvement - Reservation

Resource reservation

- Race between check and deploy
- Need to lock resource during check for deployment.
- Align with cloud agnostic intent?

Improvement - registry

Dynamic discover HPA features

- Currently discover through flavor extra_spec, only trigger once in VIM registration
- Change to regular update Or
- Enhance registry API to be re-triggerable

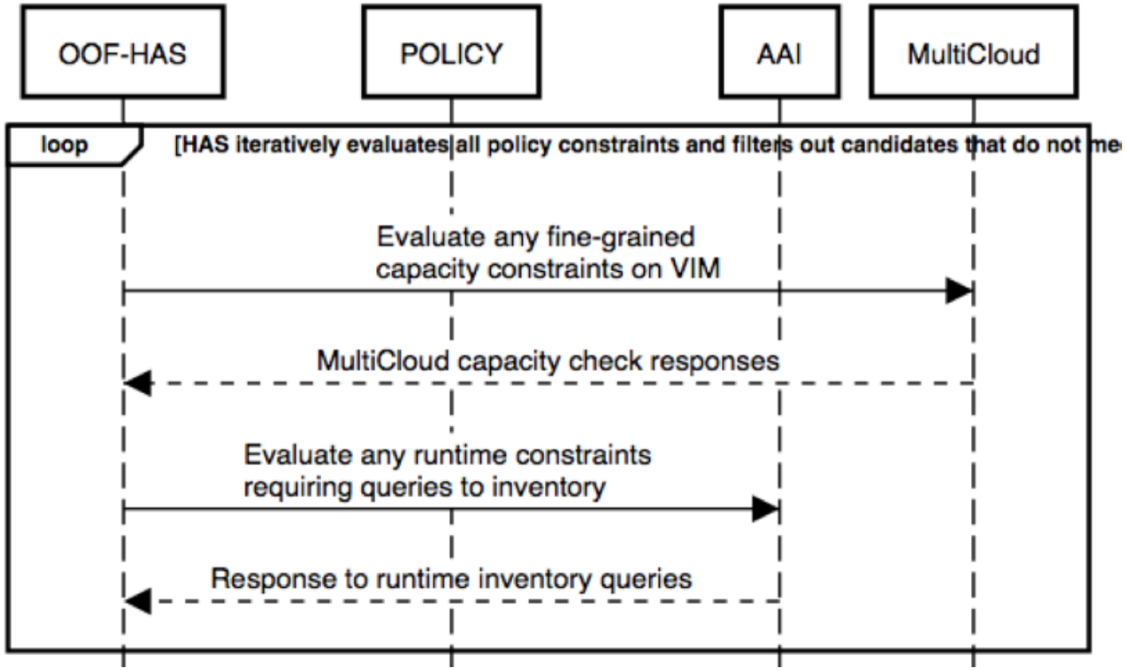
Current capacity/capability Workflow in MultiCloud

Capacity Check API

- Input is vCPU/Mem/Vol requirements, Output is suitable VIMs

HPA Discovery

- MultiCloud could discover HPA capabilities on VIM registration and generates into A&AI



Improvement – check_vim_capacity

- Check_vim_capacity
- Input

```
{
  "vCPU": int,  // number of cores
  "Memory": float,  // size of memory, GB
  "Storage": int, //GB
  "VIMs": array  // VIMs OOF wish to check with
}
```

- Output

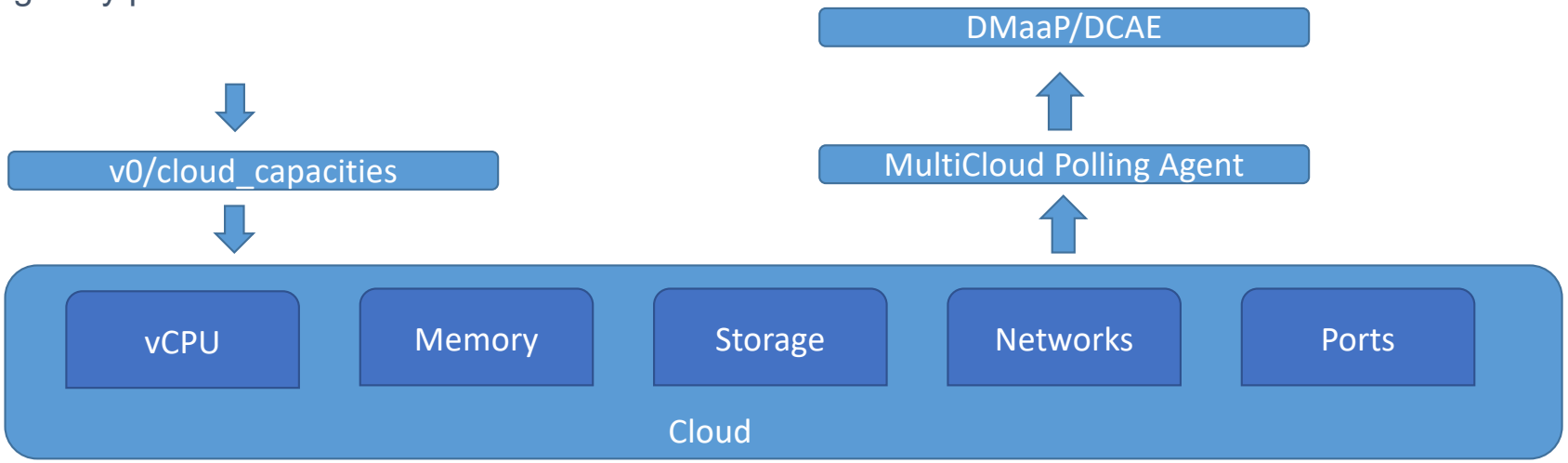
```
{
  "VIMs": array  // VIMs satisfy with this resource requirement
}
```

- Only accept a list of candidate VIMs, could not check against all VIMs.
  - An empty list indicates to check against all VIMs

Improvement - Up to date cloud capacity

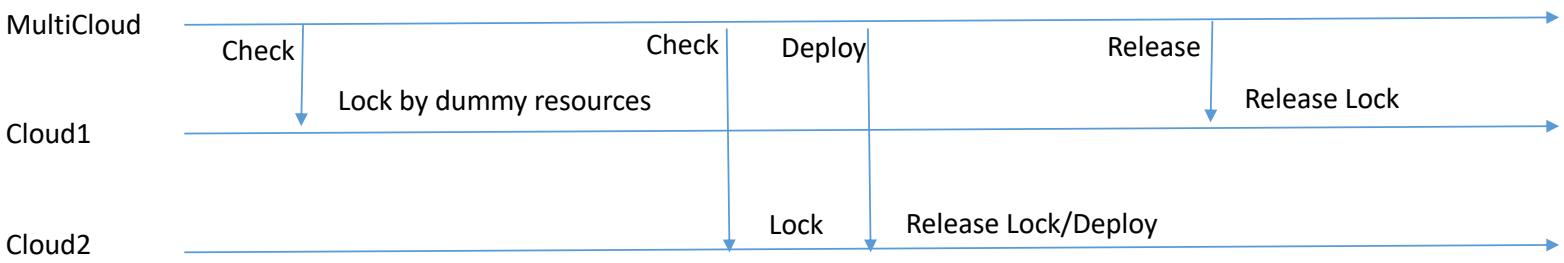
- Disvantage of check_vim_capacity API .
  - Check_vim_capacity shadows the capacities.
  - Could only check through cloud native APIs, like openstack API

- Improvement
  - New API to expose those, like v0/cloud_capacities .
  - Regularly push to DMaaP



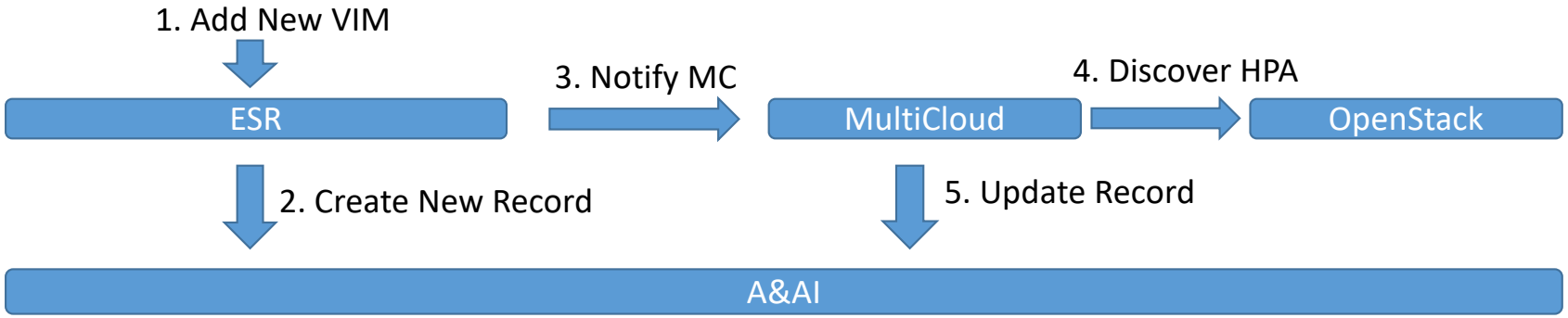
Improvement - Reservation

- Resource reservation
  - Race between check and deploy
  - Need to lock resource during check for deployment.
  - Align with cloud agnostic intent



Improvement - Registry

- Dynamic discover HPA features
  - Currently discover through flavor extra_spec, only trigger once in VIM registration
  - Change to regular update
  - Enhance registry API to be re-triggerable

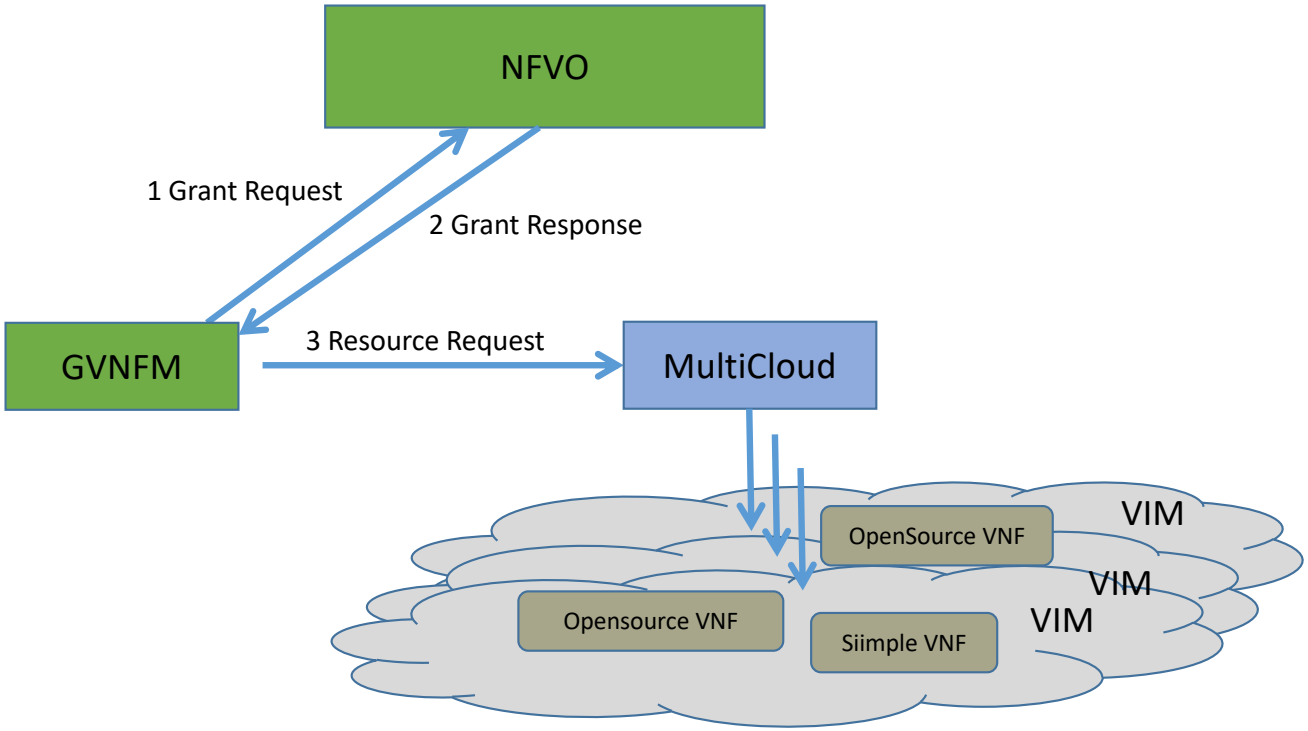


Agenda

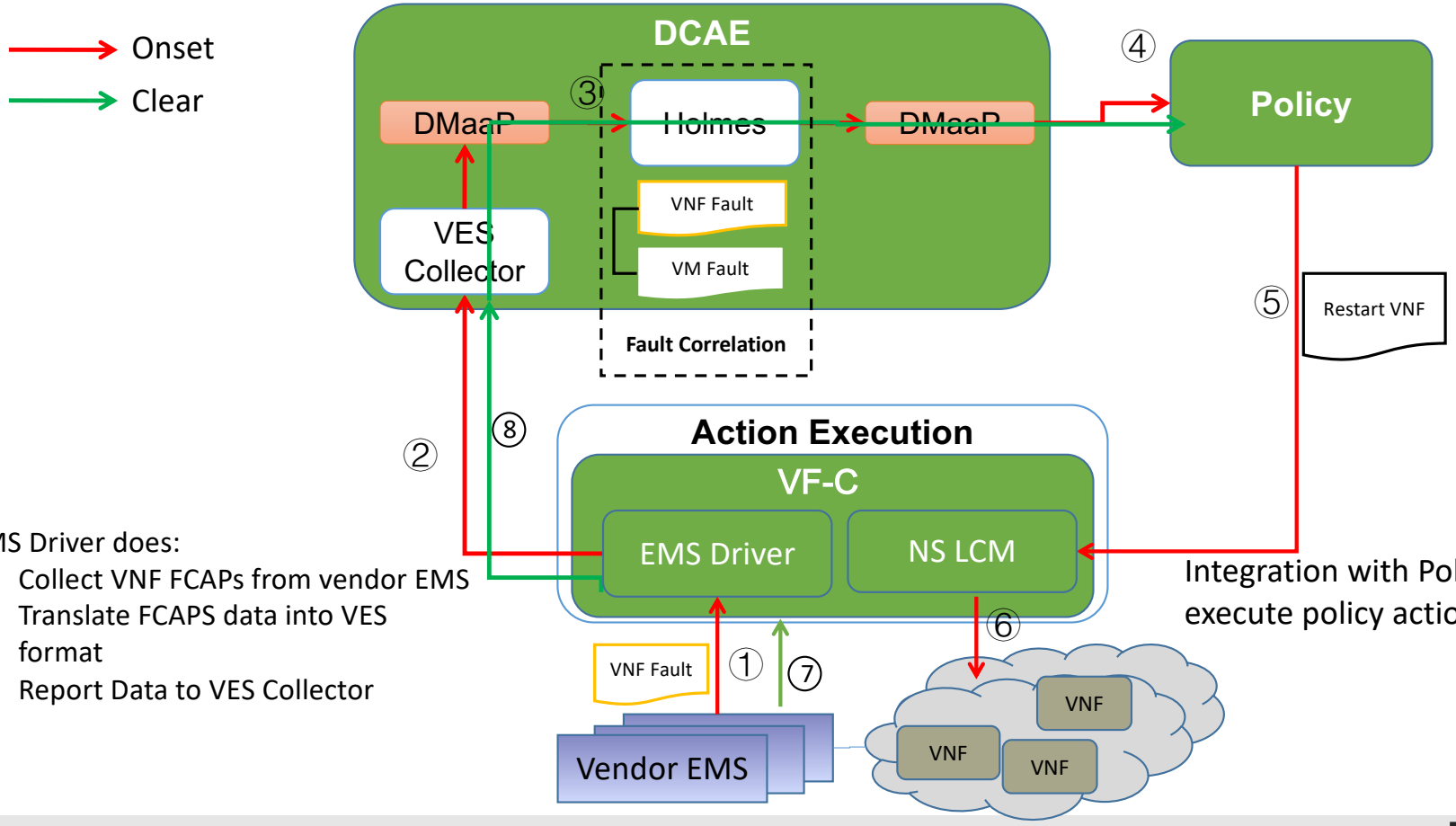
- Kick-off
- Instantiation
- **Closed Loop**
- Development and Diagnosis

MultiCloud & VFC – Direct Mode VIM Integration

Support direct mode via MultiCloud



Architecture Facts – Close Loop Integration

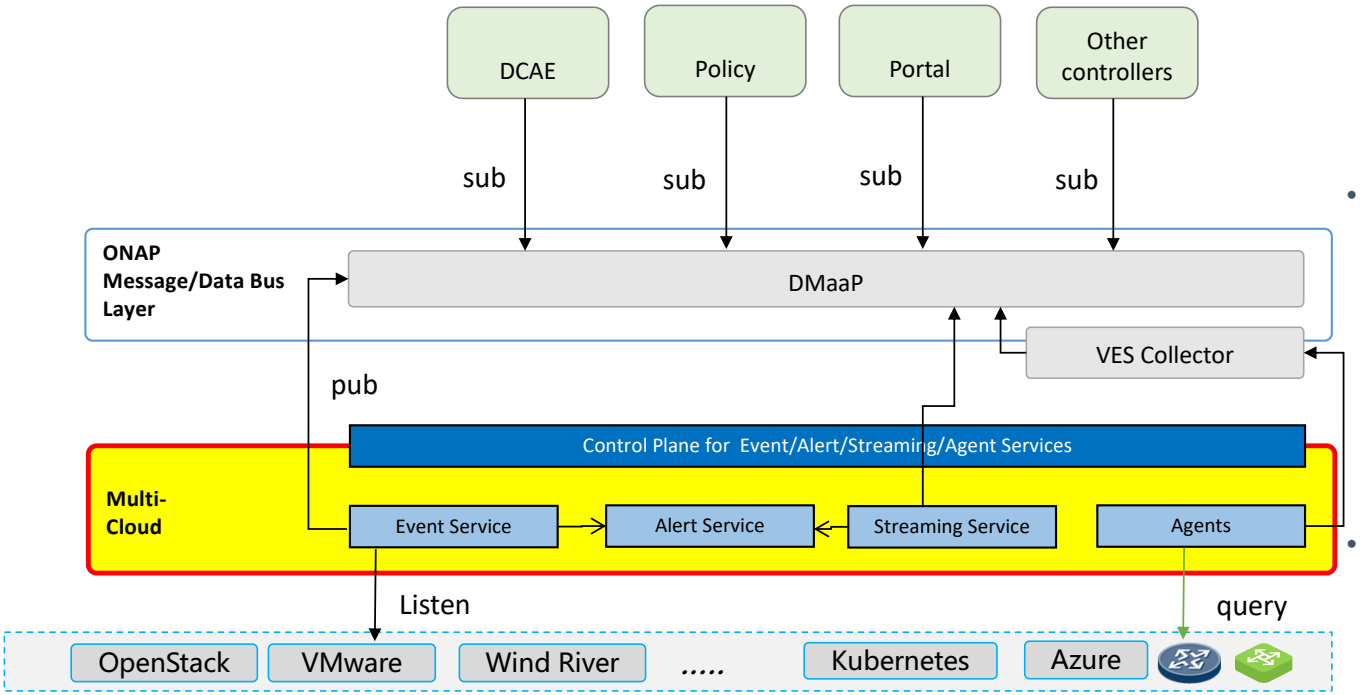


EMS Driver does:

1. Collect VNF FCAPs from vendor EMS
2. Translate FCAPS data into VES format
3. Report Data to VES Collector

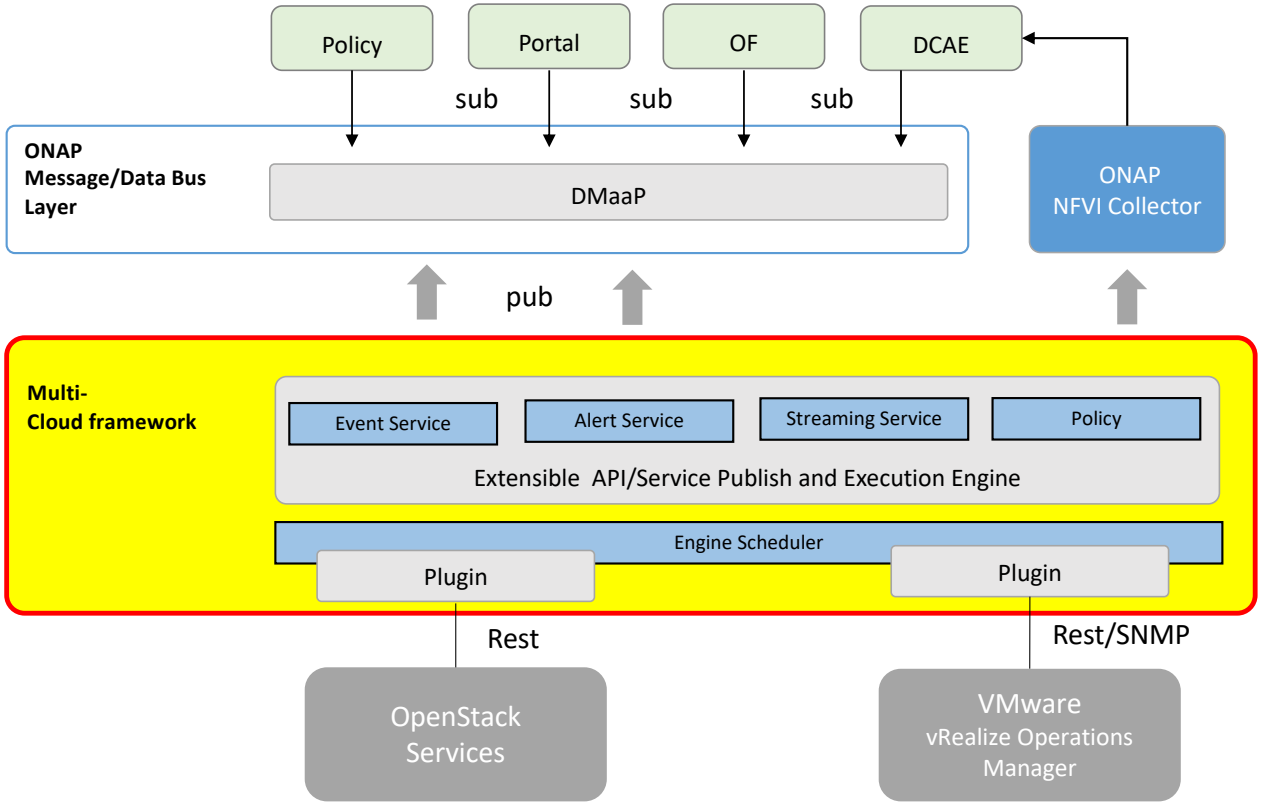
Integration with Policy to execute policy action

Alert/Event/Meter Federation Framework



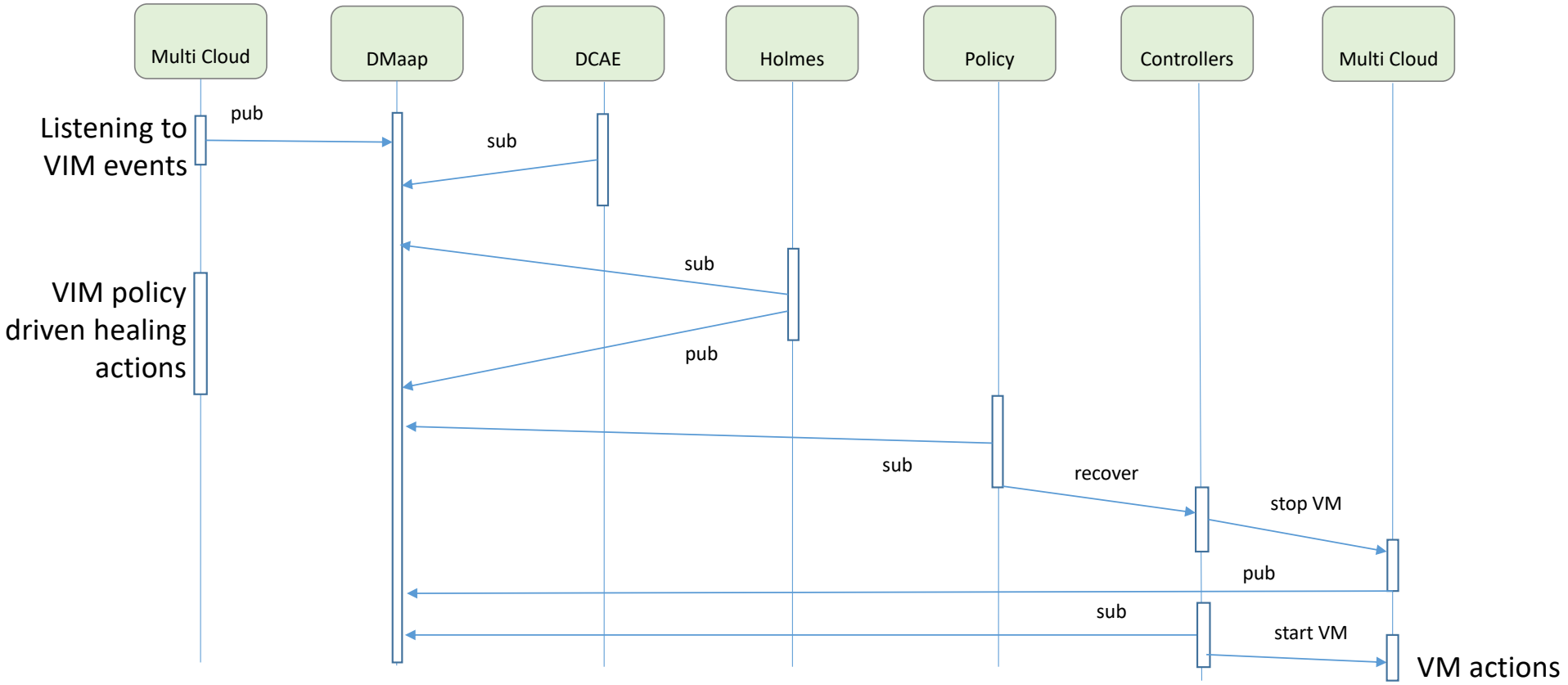
- **Event Service**
  - Federate events from different VIM providers with ONAP Message/Data bus services
  - Allow to be configured by the control plane about listener and endpoints
  - Not only events of different backend clouds, but also events from VIM controllers
- **Streaming Service**
  - Federate 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

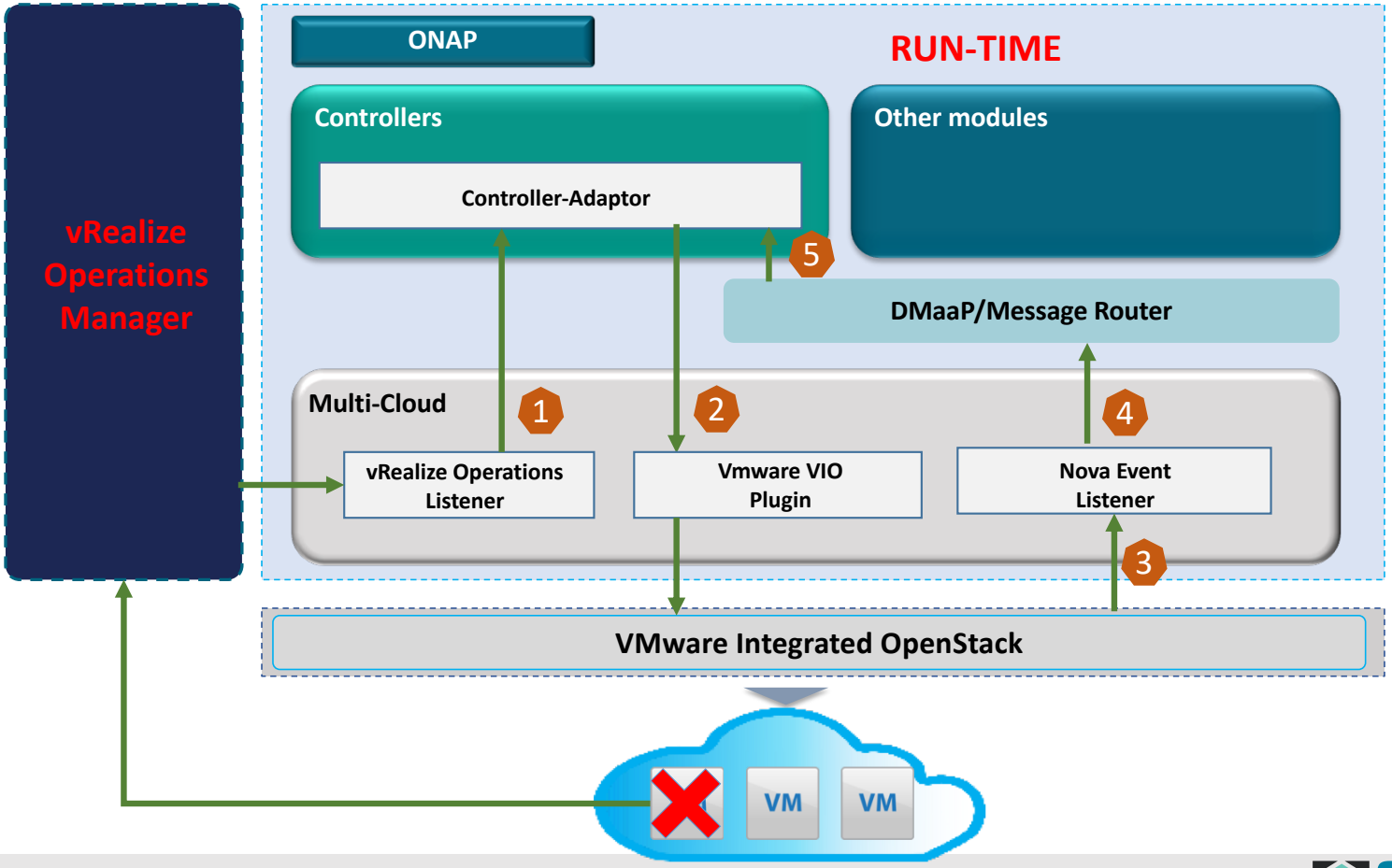


- **Extensible MC framework**
  - Pluggable framework
  - Intelligence contained
  - Information models standardized
  - Customizable to work with upper layers
  - Policy-driven data distribution
- **Compensate each other with different Telemetry:**
  - OpenStack control plane service status
  - Underlying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

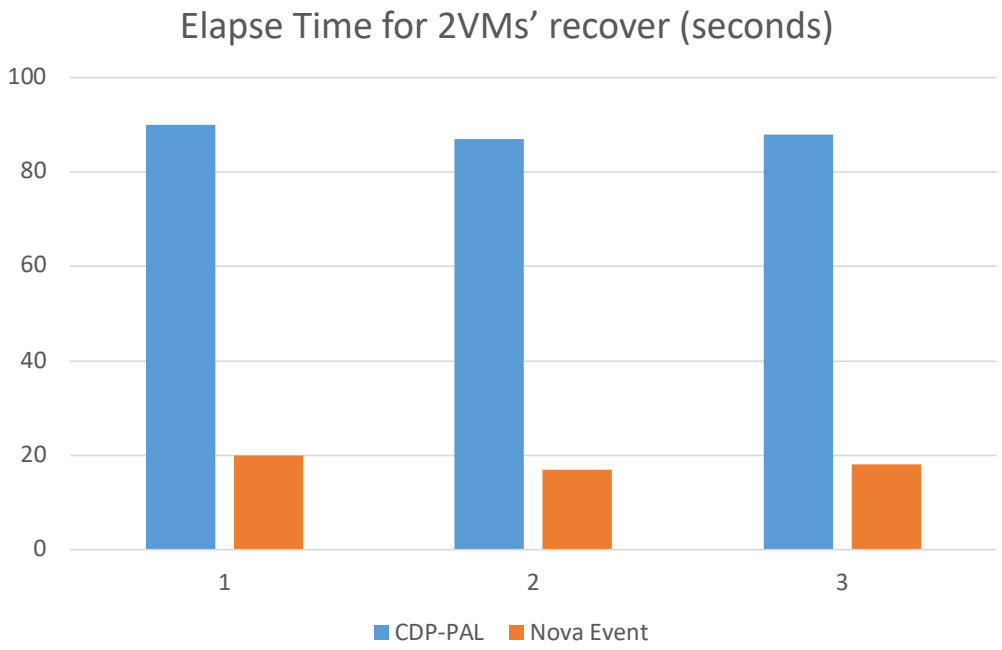
Alert/Event/Meter Federation Flow Chart

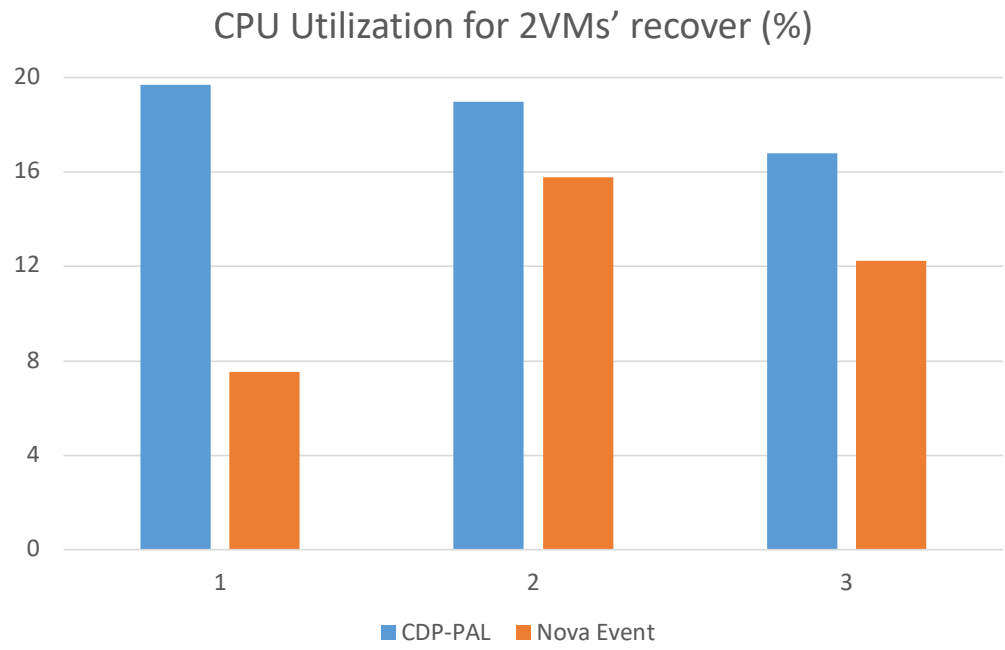


Multi VIM/Cloud based Close Loop



Performance Improvement





Agenda

- Kick-off
- Instantiation
- Closed Loop
- **Development and Diagnosis**

Topics

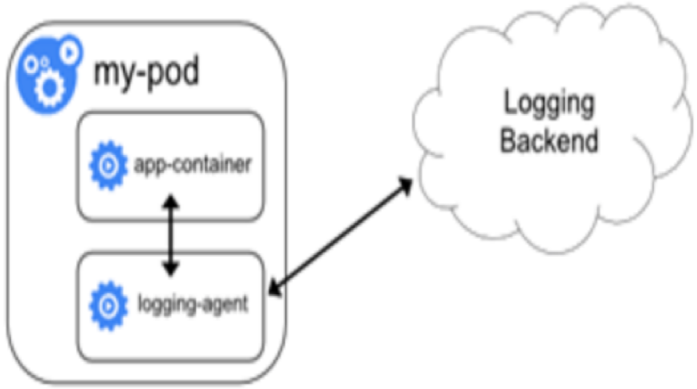
- Filebeat as a sidecar container collect Multivim logs(R2)
- onappylog a python library(R2)
- Integration with POMBA(R3)
- Discussion

Filebeat

The multivim container is deployed in K8s platform, each pod associated with filebeat sidecar container to collect logs. Filebeat fetch the logs through shared volume in pod.

Good: decoupled, easy to maintain and upgrade.

Bad: Need more resource for each pod.



onappylog library

Extend python built-in log library to satisfy the needs of logging team.
It is similar to log4j library widely used in java project.
Feture:

1 Compatible with built-in log library
2 Support Mapped Diagnostic Context(MDC)
3 Configuration file with yaml template
4 Support runtime reloading configuration

Shortage:

1 Not support python3
2 Not support MDC inheriting for child thread （python dose not has this ability）
3 Not support Markers

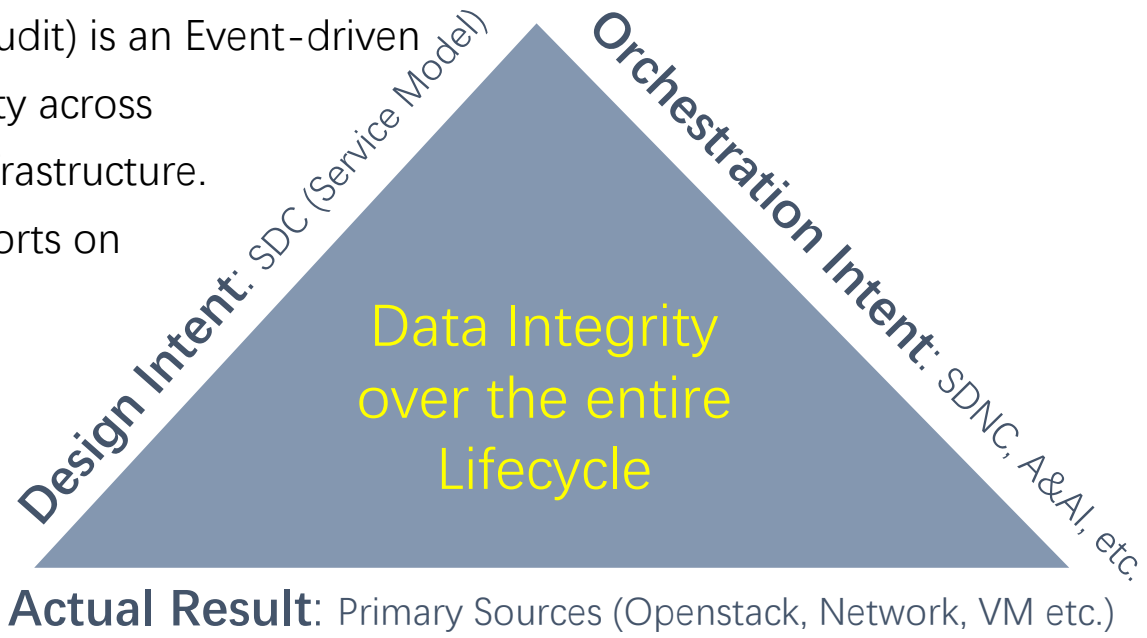
Reference: https://github.com/onap/logging-analytics/tree/master/pylog

Integration with POMBA

Background:

POMBA (Post Orchestration Model Based Audit) is an Event-driven auditing platform checking the data integrity across NFV orchestration environment and NFV infrastructure.

In short, It provides an ONAP audit and reports on any data integrity issues found



Context Aggregator

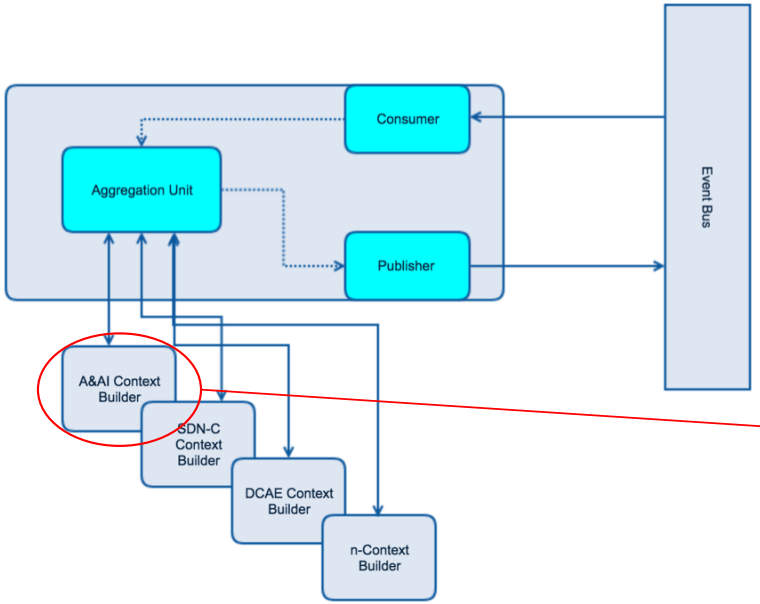


figure 1

Aggregator is configured to subscribe the Message Router topic, then orchestrates the calls to various context builders to obtain data.

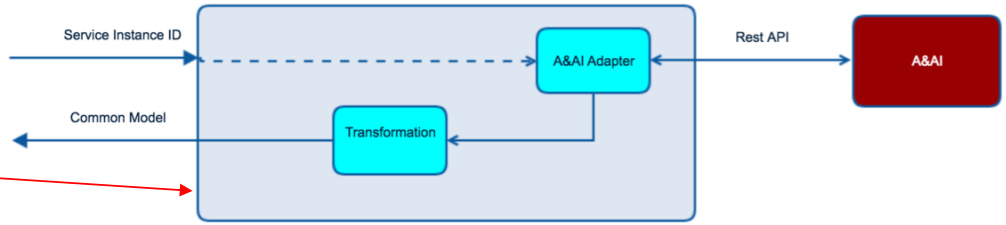
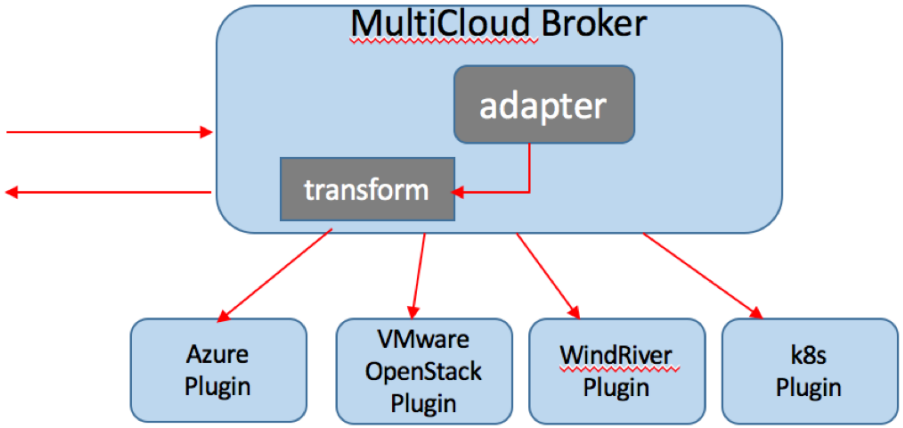


figure 2

Context builder could provider information from a particular data source and present it via a well defined common data model.

Mode reference:https://wiki.onap.org/display/DW/Context+Builders+Mapping+to+Common+Model

Context builder-multivim



Broker has NBI API will forward request to correct plugin. It is reasonable to implement adaptor in broker that expose context builder NBI API.

Discussion

- On my opinion, Data integrity is a pain point for vim registry.
- Dose POMBA is the best way to resolve this problem? Any other Ideas?

Thank You!