ONAP
OPEN NETWORK AUTOMATION PLATFORM

Enabling Workloads Orchestration in Containers via MultiCloud

Bin Hu (AT&T), Ramki Krishnan (VMWare), Gil Hellmann (Wind River)

Key Contributors:

AT&T: Bin Hu, John Choma, Jack Murray, Brian Freeman, Oliver Spatscheck, Catherine Lefevre, Vimal Begwani, Bryan Sullivan, Gil Bullard, Claude Noshpitz, John Ng, Habib Torab, Dean Bragg
Intel: Alex Vul
VMWare: Ramki Krishnan, Sumit Verdi, Xinhui Li
Wind River: Gil Hellmann, Bin Yang

Agenda

- **Cloud Native Workloads and Use Cases**
- Cloud Native Orchestration
- MultiCloud Reference Architecture – Long-Term Vision
- Enabling Cloud Native Workloads Orchestration via MultiCloud
- Next Steps

Cloud Native Workloads – Characteristics and Benefits

Containerized, Microservice

Decentralized governance

Independently deployable, upgradeable and replaceable

Decentralized data management

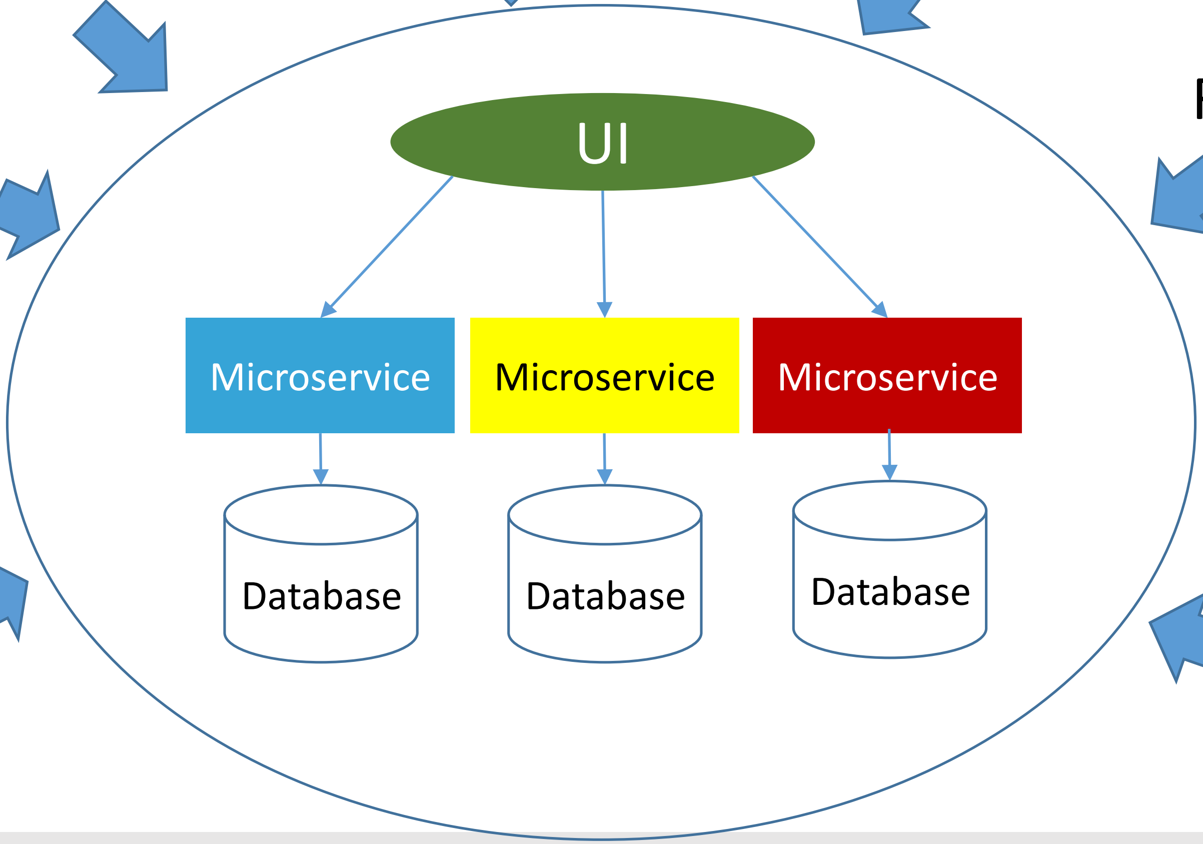
Focused on business capabilities

CI / CD, automated infrastructure

Product mentality, you build, you run it

“Failure” in design

Smart endpoints and dumb pipe

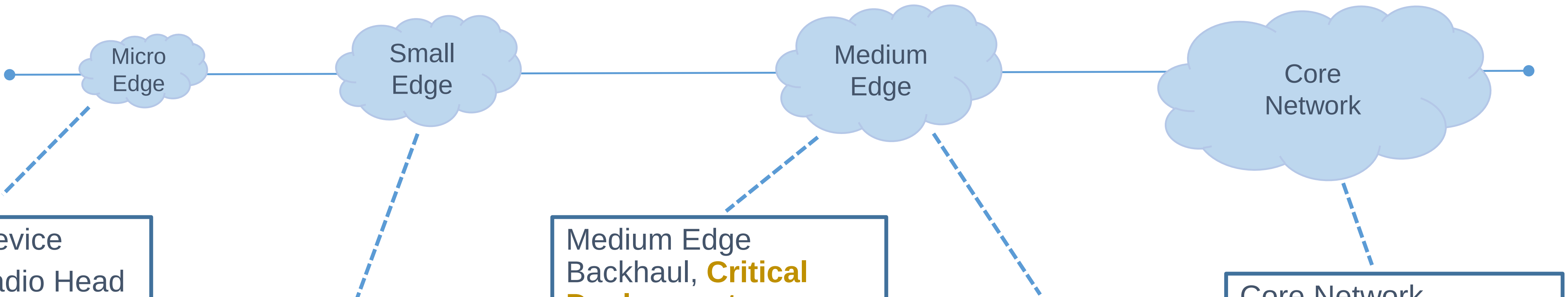


Cloud Native Workloads – NFV and Edge Use Cases

Key Industries:
Telco (e.g. NFV, 5G, and IoT),
Retail (e.g. IoT, Supply Chain)

Use cases were identified in OpenDev 2017

Key Application Domains:
Surveillance (e.g. CCTV),
Telematics, Enterprise Security



Micro Edge Device
- Remote Radio Head
- Remote Radio Unit
- CPE / set top box

Runs a single instance, instance changes infrequently

Small Edge Device
- Retail Wi-Fi
- POS for a store
- Cell tower site

Multiple instances, instances change occasionally

Medium Edge Backhaul, **Critical Deployment**
- Cloud-RAN
- Big cell site
- National Broadband Network Point of Interconnect (NBN-POI)

Multiple instances, instances change daily

Medium Edge Backhaul, **Non-Critical Deployment**
- Big box retail
- Cloudlet

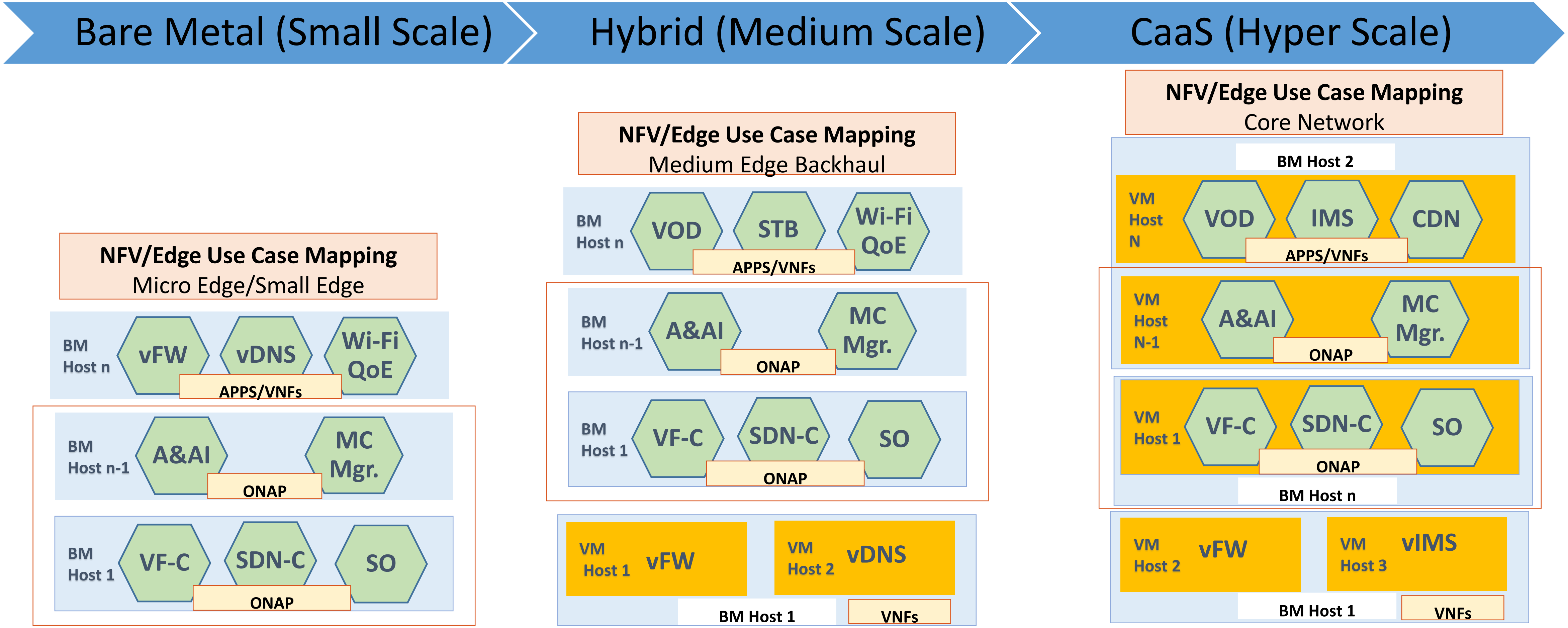
Multiple instances, instances change daily

Core Network
- Region DC
- IMS
- EPC Control Plane

Thousands of instances, instances changing constantly

Cited from KubeCon Austin 2017 ONAP Mini Summit

Cloud Native Deployment Options with ONAP (Examples)



Legend:
STB – Set Top BoX; EPG – Electronic Programming Guide; VOD – Video On Demand; IMS – IP Multi Media Subsystem;
BM – Bare Metal; VM – Virtual Machine; CaaS – Containers as a Service
Containerized ONAP components – A&AI, SDN-C etc.

Cited from KubeCon Austin 2017 ONAP Mini Summit

Agenda

- Cloud Native Workloads and Use Cases
- **Cloud Native Orchestration**
- MultiCloud Reference Architecture – Long-Term Vision
- Enabling Cloud Native Workloads Orchestration via MultiCloud
- Next Steps

Cloud Native Infrastructure and Workload Orchestration

- Infrastructure Orchestration in Cloud Native Environment
  - To create, heal, scale, and tear down container clusters
  - Lifecycle management of a container cluster (“infrastructure”)
- Workload Orchestration in Cloud Native Environment
  - To deploy, heal, scale, upgrade, and terminate containerized apps/VNFs (“workloads” in the form of “containers”) into container clusters
  - Lifecycle management of a container (“workload”)

Cloud Native Orchestration Overview

	Cloud Native “Workload” Orchestration		Hybrid Orchestration Cloudify
	K8S	**Docker Swarm**	
Multiple Hosts	√	√	**Infrastructure Orchestration** to create, heal, scale, and tear down container clusters through Docker Swarm BP and Kubernetes Cluster BP. **Service Orchestration** to deploy containerized apps/VNFs into container clusters through Docker Swarm Plugin and Kubernetes Plugin.
Placement Control	√	√	
Affinity / Anti-affinity	√	√	
Networking	√	√	
High Availability	√	√	
Scaling	√	Auto heal and manual scale; lack of pods	
Load Balancing	√	√	
Rolling Upgrades	√	Lack of pods	
Container Agnostic	√	Tie to Docker runtime	
Production Experience	√	Not much production deployment	
Community Support	50,000 commits / 1,200 contributors	3,000 commits / 120 contributors	
Scope	Container focused		More general
Applicability	Cloud native		Hybrid

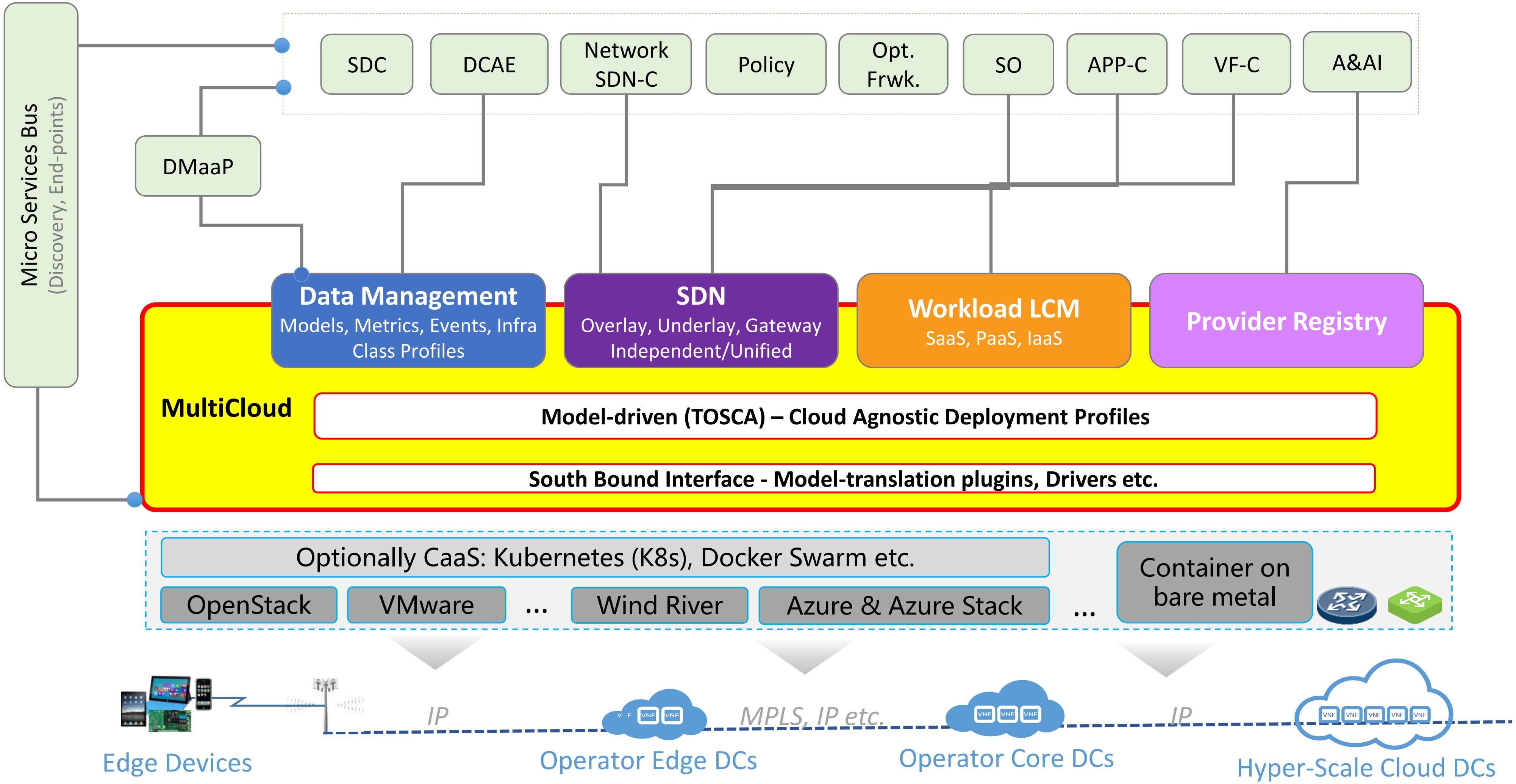
Workload Orchestration on Edge

- Challenges
  - Minimizing footprint required to place workloads on the edge
  - Autonomous workload management locally on the edge
- Considerations
  - Each edge platform is managed and operate independently or in a cluster
    - Fully capable local orchestrator is needed
    - The provisioning and orchestration of workload directly at the edge platform
      - even if the management plane between central orchestrator and edge platform is interrupted.
  - Global hierarchical orchestration

Agenda

- Cloud Native Workloads and Use Cases
- Cloud Native Orchestration
- **MultiCloud Reference Architecture – Long-Term Vision**
- Enabling Cloud Native Workloads Orchestration via MultiCloud
- Next Steps

MultiCloud Reference Architecture – Long-Term Vision

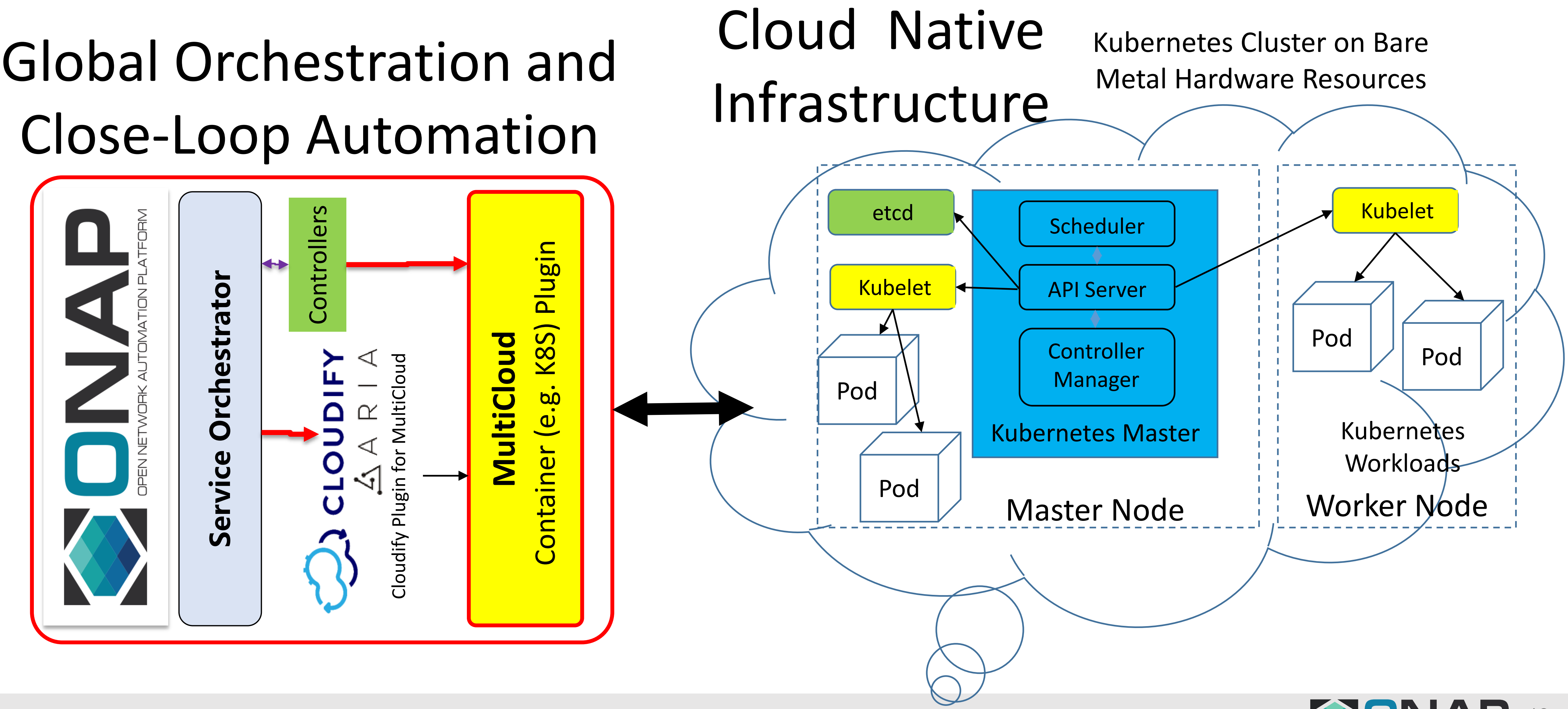


Adapted from ONAP Paris MC Workshop Introduction :
https://wiki.onap.org/download/attachments/11928197/ONAP-mc-intro.pdf?version=1&modificationDate=1506518564000&api=v2

Agenda

- Cloud Native Workloads and Use Cases
- Cloud Native Orchestration
- MultiCloud Reference Architecture – Long-Term Vision
- **Enabling Cloud Native Workloads Orchestration via MultiCloud**
- Next Steps

Enabling Cloud Native Workloads Orchestration via MultiCloud



Global Orchestration and Close-Loop Automation
Cloud Native Infrastructure
Kubernetes Cluster on Bare Metal Hardware Resources
ONAP
OPEN NETWORK AUTOMATION PLATFORM
Service Orchestrator
Controllers
CLOUDIFY
ARIA
Cloudify Plugin for MultiCloud
MultiCloud
Container (e.g. K8S) Plugin
etcd
Kubelet
Pod
Pod
Scheduler
API Server
Controller Manager
Kubernetes Master
Master Node
Kubelet
Pod
Pod
Kubernetes Workloads
Worker Node

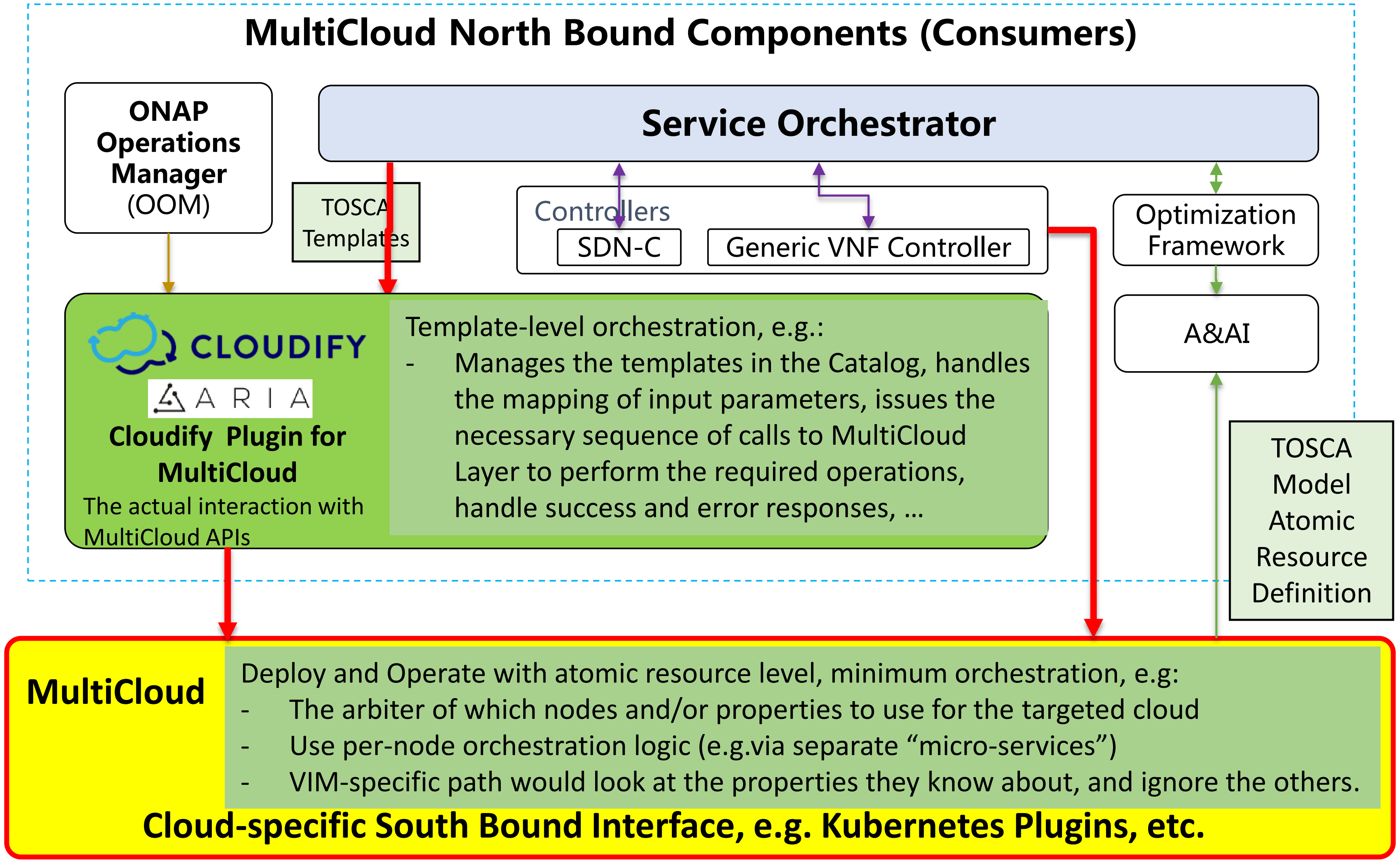
Enabling Cloud Native Workloads Orchestration via MultiCloud

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”



MultiCloud North Bound Components (Consumers)

The actual interaction with MultiCloud APIs

Template-level orchestration, 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, …

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

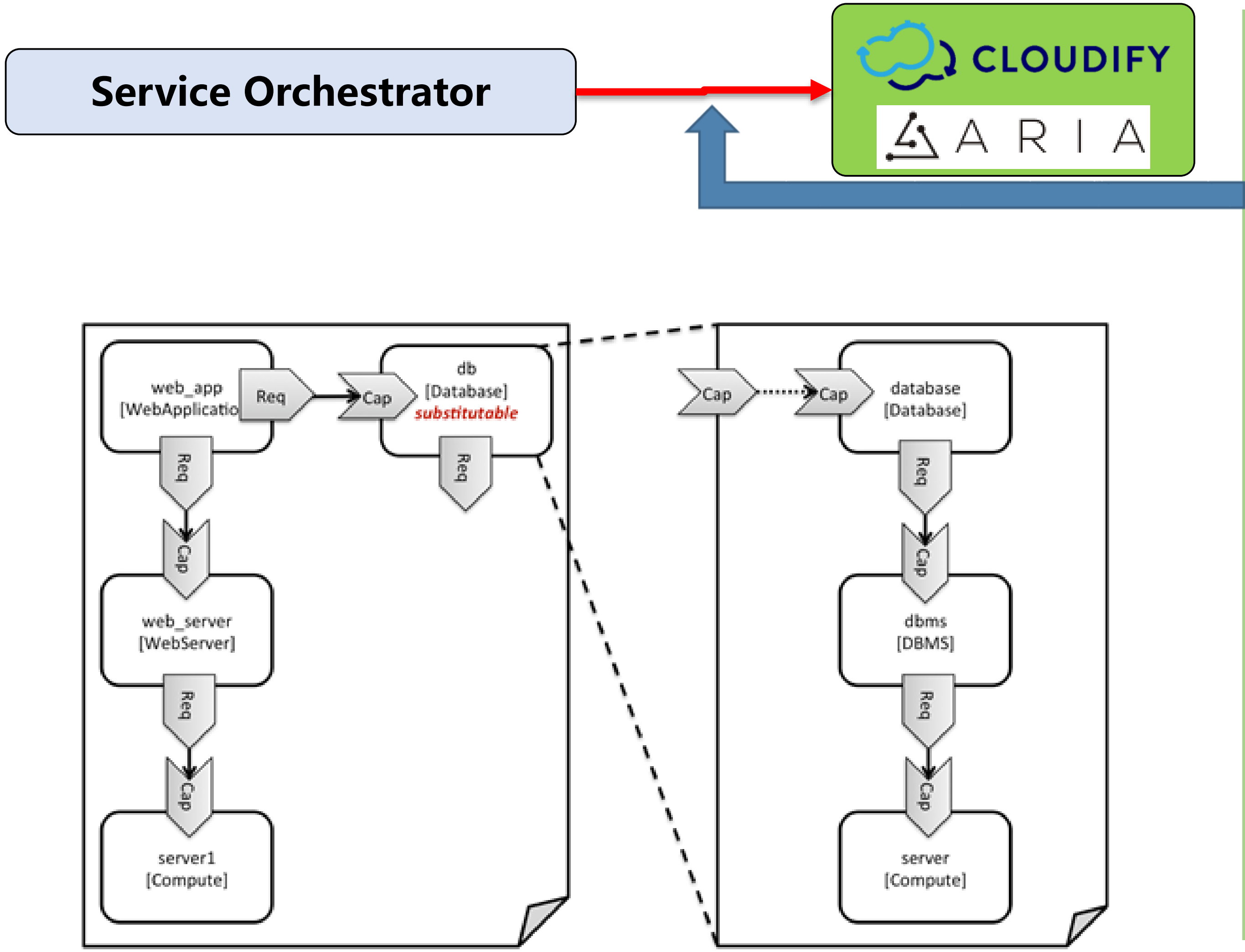
- The arbiter of which nodes and/or properties to use for the targeted cloud
- Use per-node orchestration logic (e.g.via separate “micro-services”)
- VIM-specific path would look at the properties they know about, and ignore the others.

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

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.

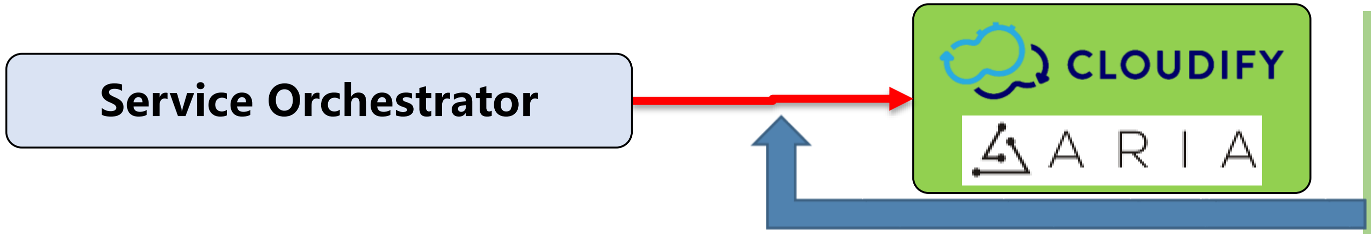
Enabling Cloud Native Workloads Orchestration via MultiCloud

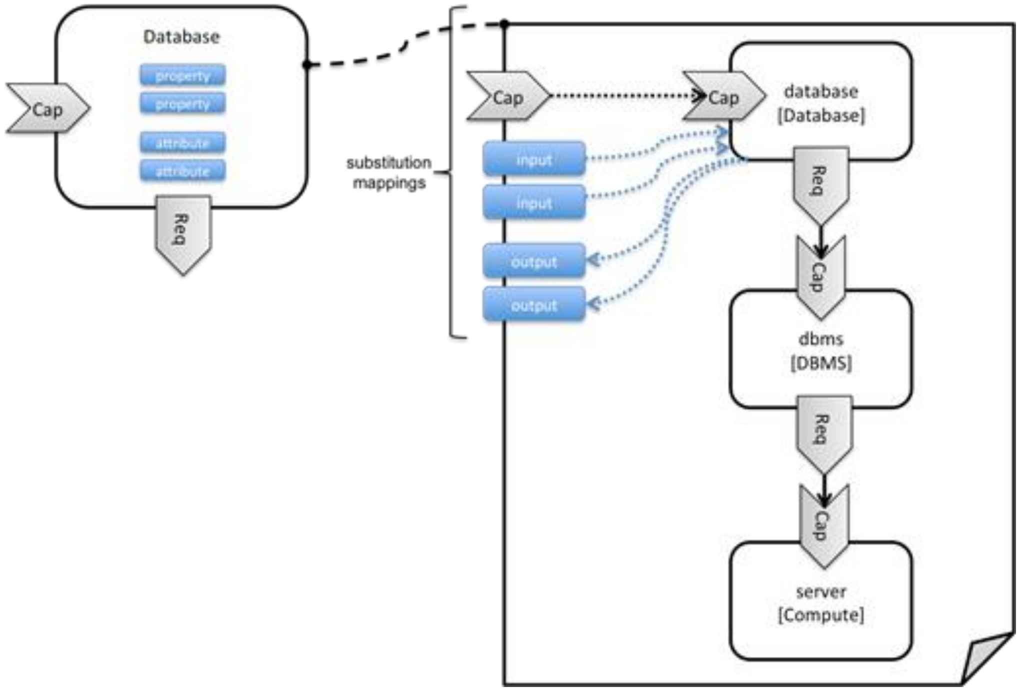


```
tosca_definitions_version: tosca_simple_yaml_1_0

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]. Copyright © OASIS Open 2017. All Rights Reserved.

Enabling Cloud Native Workloads Orchestration via MultiCloud

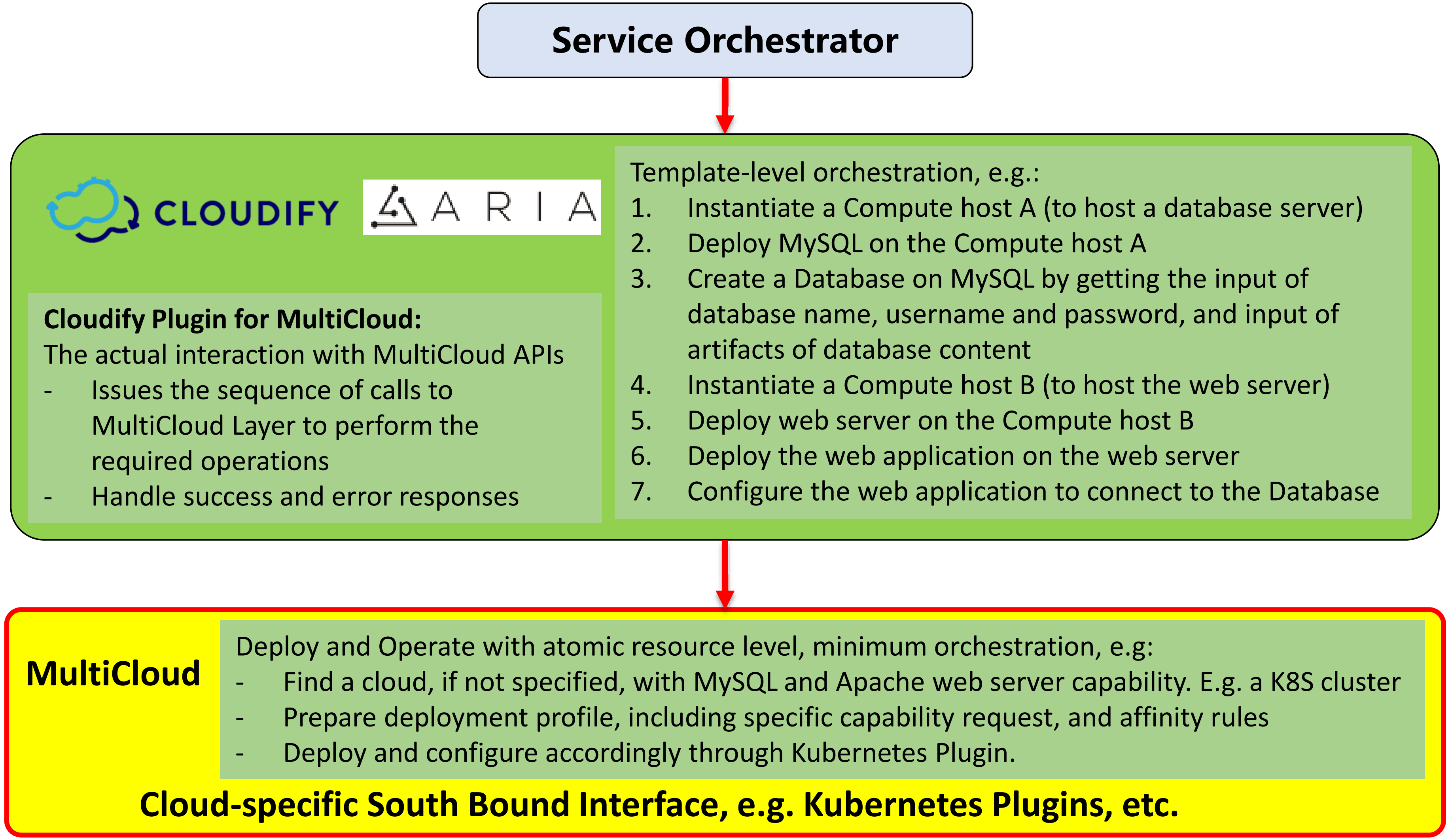




```
tosca_definitions_version: tosca_simple_yaml_1_0

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
```

Example cited from [TOSCA-Simple-Profile-YAML-v1.1]. Copyright © OASIS Open 2017. All Rights Reserved.

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.

Benefits

- Hierarchical orchestration fits the cloud native paradigm
  - Lightweight local orchestrator for autonomous workload management in edge
  - Global orchestration and closed-loop automation by ONAP for massive scalability and agility in multisite and multicloud
  - Particularly fits edge use cases
- Abstracting cloud-specific attributes away from the blueprints and service templates in global orchestration
  - Enable more flexible update of implementation of SBI/Plugin code within MultiCloud without impact on previously deployed blueprints and service templates
- Allow implementation of SBI/Plugin in any language within MultiCloud,
  - Enable reuse of existing SBI/Plugin within MultiCloud for current infrastructure

Continuing to Support HEAT

- Transparent to MultiCloud. SO need to handle it and encapsulate it in TOSCA appropriately. E.g:
  - When SO constructs service template, it can specify in the model structure with a custom node for OpenStack, and HEAT template within the custom node as the data structure.
  - MultiCloud handles the custom node, and sees it only fits OpenStack, so MultiCloud gives the deployment request to OpenStack, and use HEAT as the parameter.

Agenda

- Cloud Native Workloads and Use Cases
- Cloud Native Orchestration
- MultiCloud Reference Architecture – Long-Term Vision
- Enabling Cloud Native Workloads Orchestration via MultiCloud
- **Next Steps**

Next Steps

- Agree on the long term goal for ONAP to orchestrate workloads in cloud native infrastructure
- Agree on the approach to enabling cloud native workloads orchestration via MultiCloud
- Define tasks, timeline and targeted release

Backup Slides