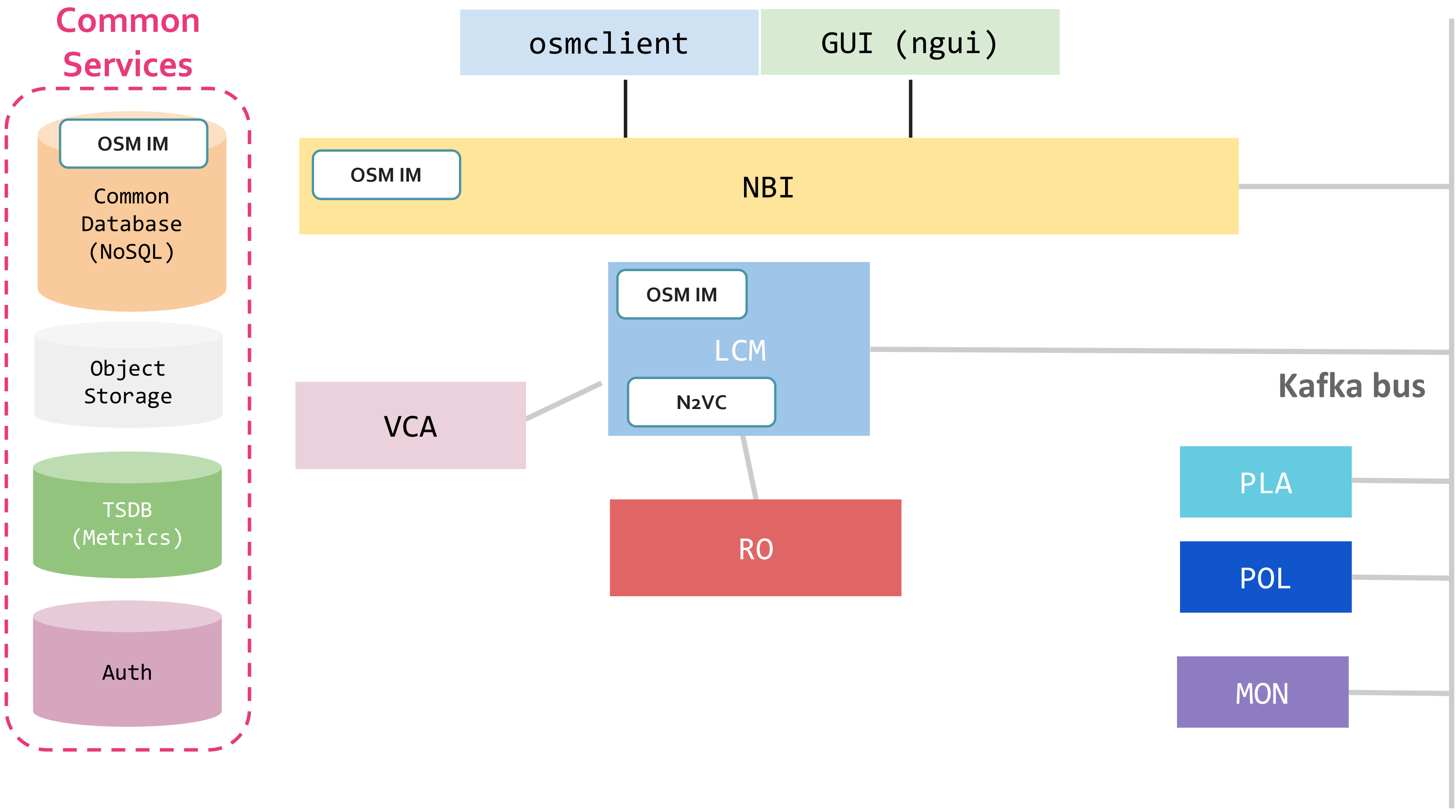
Open Source MANO

OSM architecture

Gerardo García de Blas (TSC Chair, Telefónica)

OSM architecture - Whiteboard

OSM architecture



1 **Unified Northbound Interface (SOL005-based), decoupled from LCM**

2 **E2E orchestration through Life Cycle Manager (LCM)**

3 **Message bus for async communications**

4 **Common DB, auth and object storage**

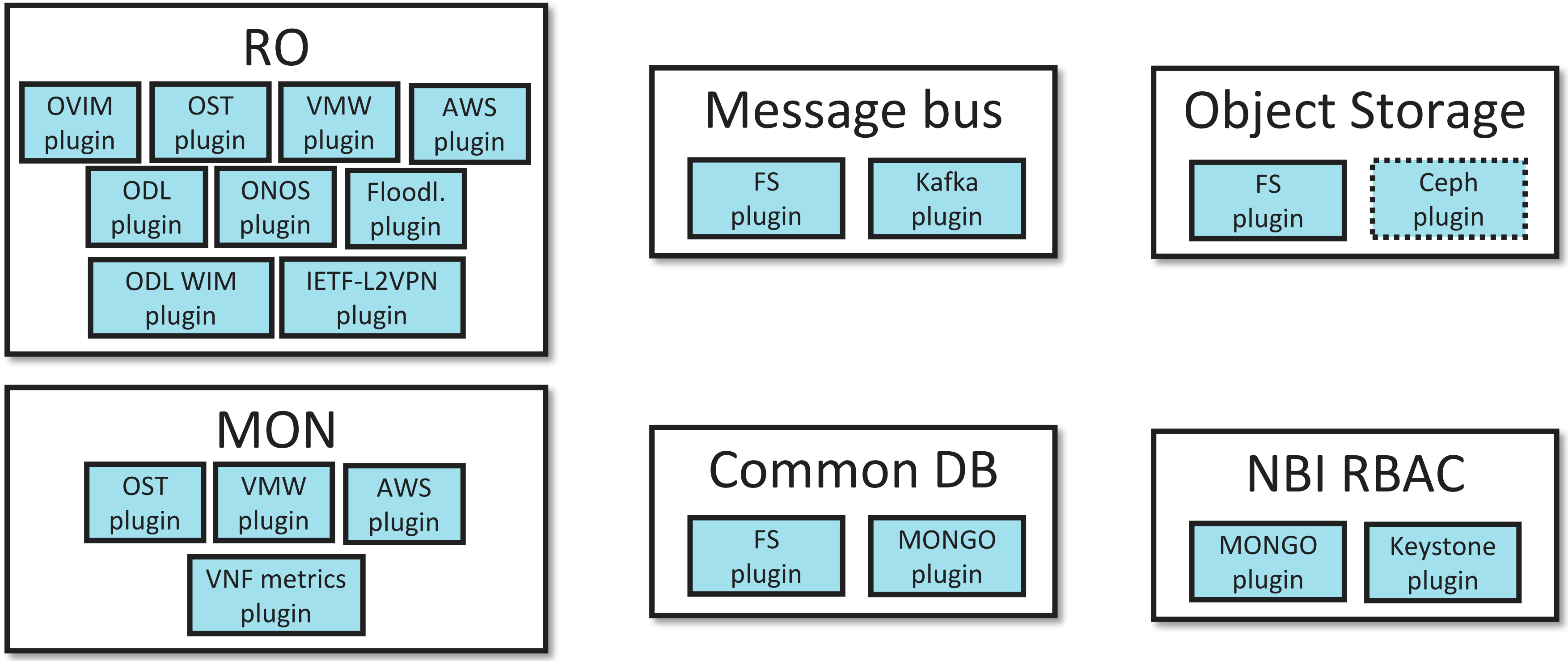
5 **Integrated components for policy, fault, performance management and placement**

6 **Complete control through CLI and UI**

7 **Execution environments for VNF configuration**

Plugin model

Open Source MANO



RO

OVIM plugin

OST plugin

VMW plugin

AWS plugin

ODL plugin

ONOS plugin

Floodl. plugin

ODL WIM plugin

IETF-L2VPN plugin

Message bus

FS plugin

Kafka plugin

Object Storage

FS plugin

Ceph plugin

MON

OST plugin

VMW plugin

AWS plugin

VNF metrics plugin

Common DB

FS plugin

MONGO plugin

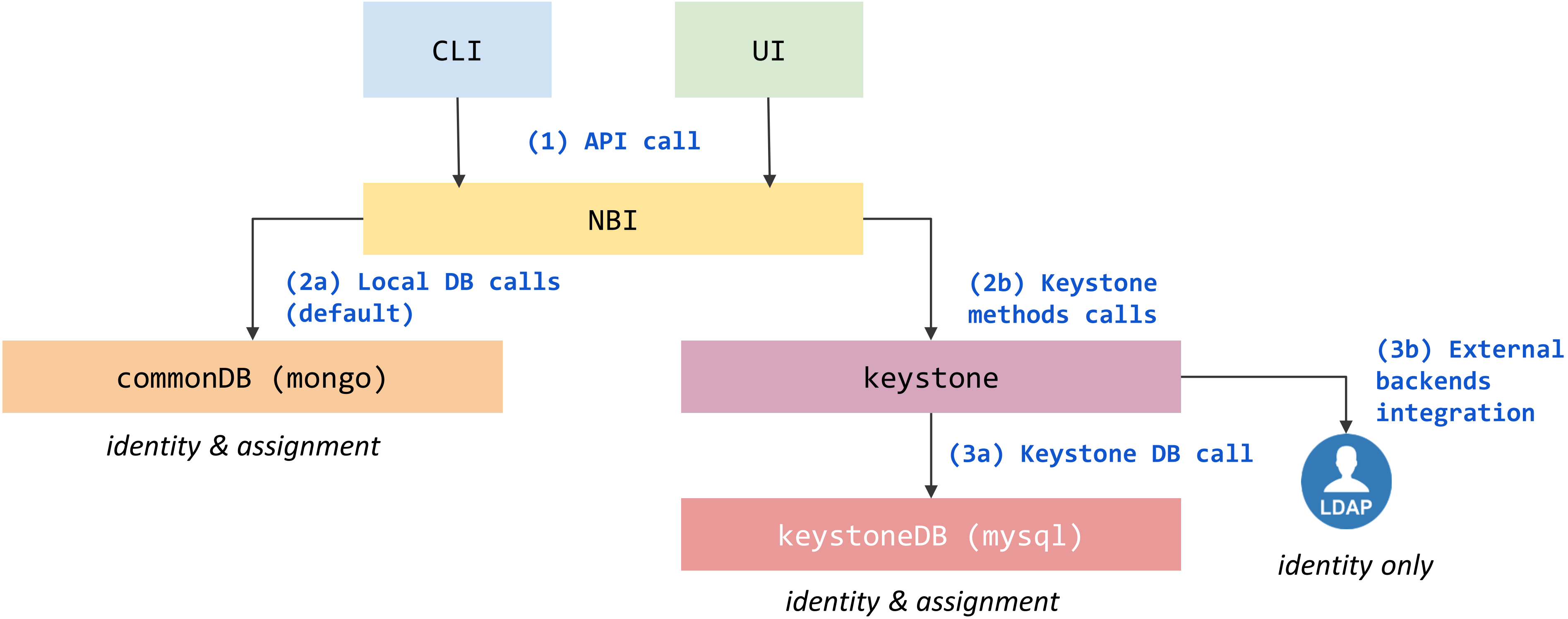
NBI RBAC

MONGO plugin

Keystone plugin

Identity & Assigment Operations

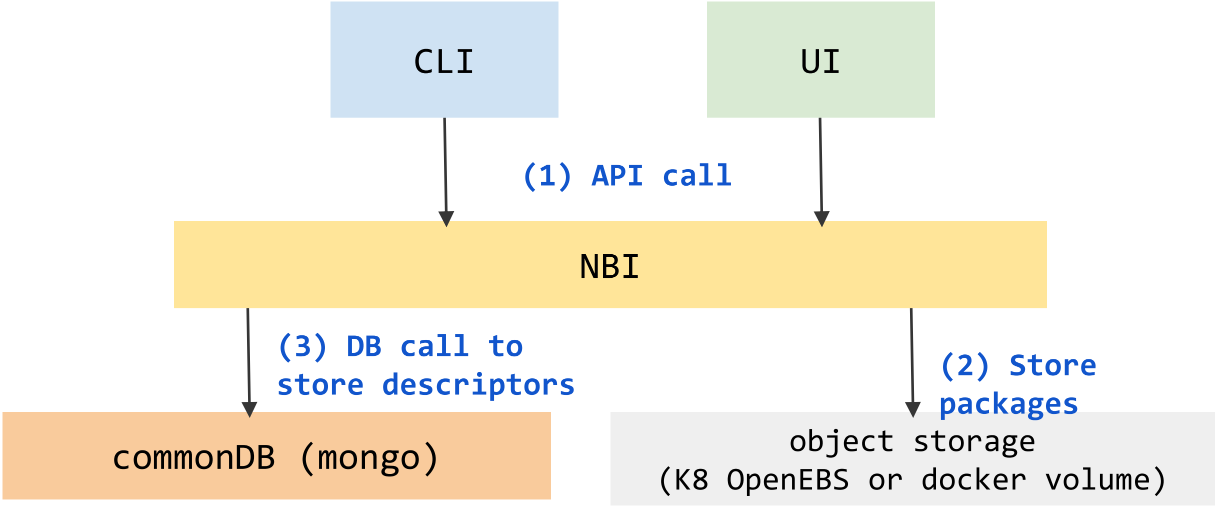
When dealing with the creation, modification or deletion of users, projects and roles, the interacting components vary according to the selected backend.



Uploading packages

When reading, uploading, modifying and deleting a Network Slice Template, Network Service Package or VNF Package, the following components interact.

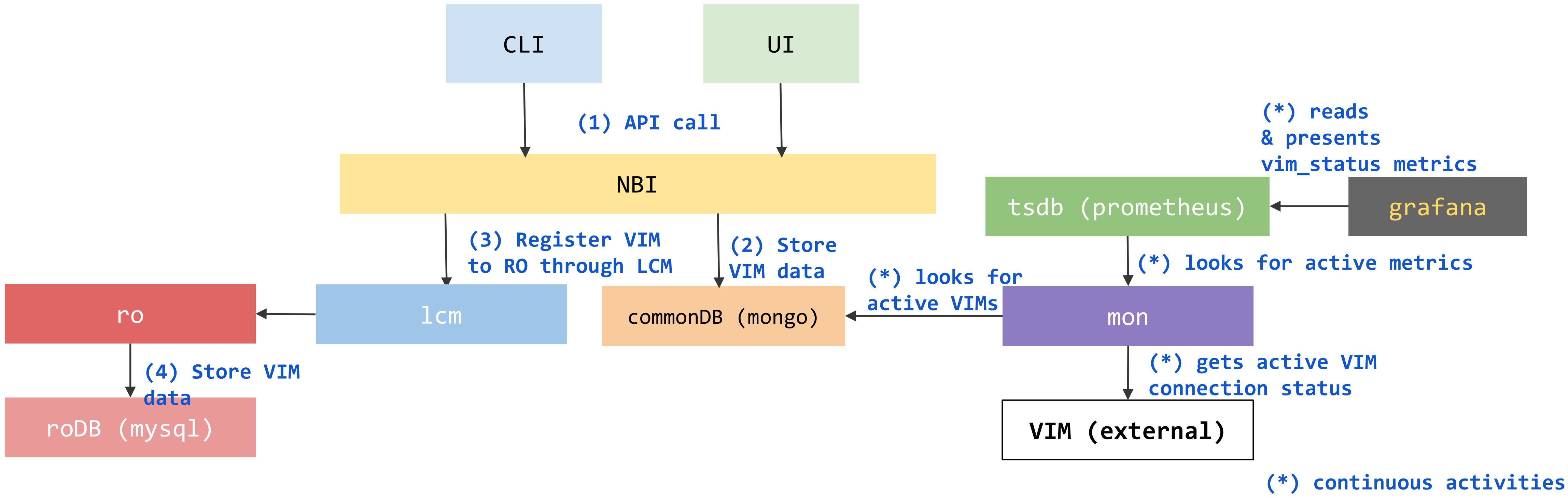
CLI Example: `osm nfpkg-create myvnfpackage.tar.gz`



Adding VIM/SDNC

When registering new VIMs or SDN Controllers, the following components interact.

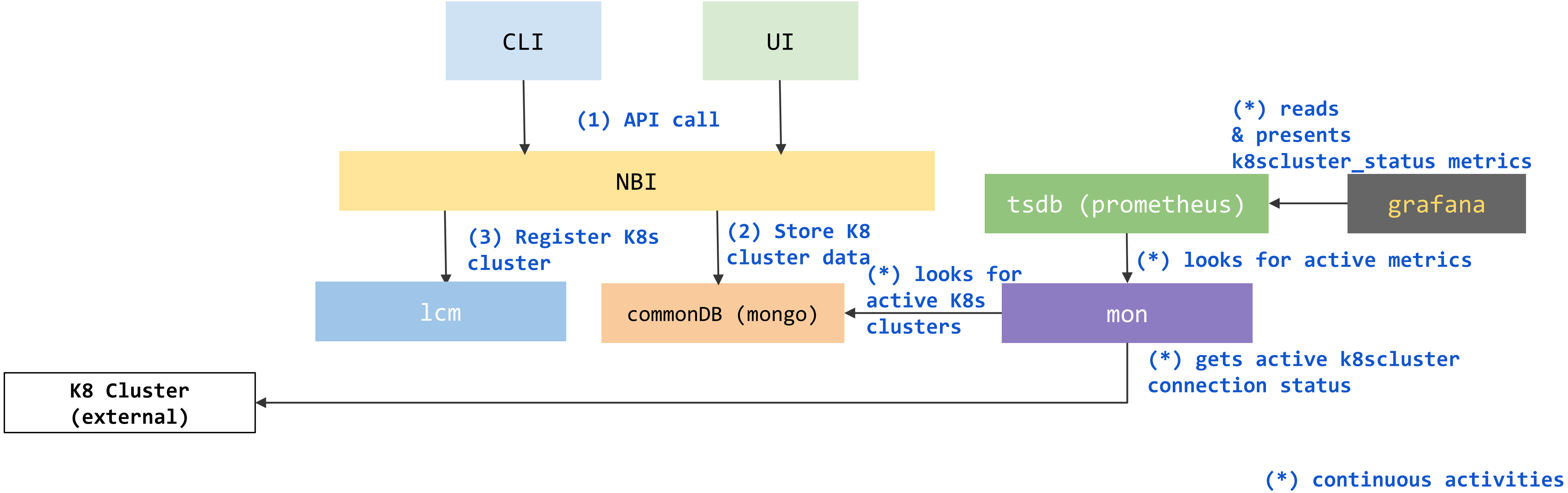
CLI Example: `osm vim-create --name myVIM --user myuser --password myprecious --auth_url http://172.21.7.5:5000/v3 --tenant mytenant --account_type openstack`


CLI
UI
(1) API call
NBI
(3) Register VIM to RO through LCM
(2) Store VIM data
ro
lcm
commonDB (mongo)
(*) looks for active VIMs
mon
(4) Store VIM data
roDB (mysql)
tsdb (prometheus)
grafana
(*) reads & presents vim_status metrics
(*) looks for active metrics
(*) gets active VIM connection status
VIM (external)
(*) continuous activities

Adding a K8s Cluster

When registering new Kubernetes clusters, the following components interact.

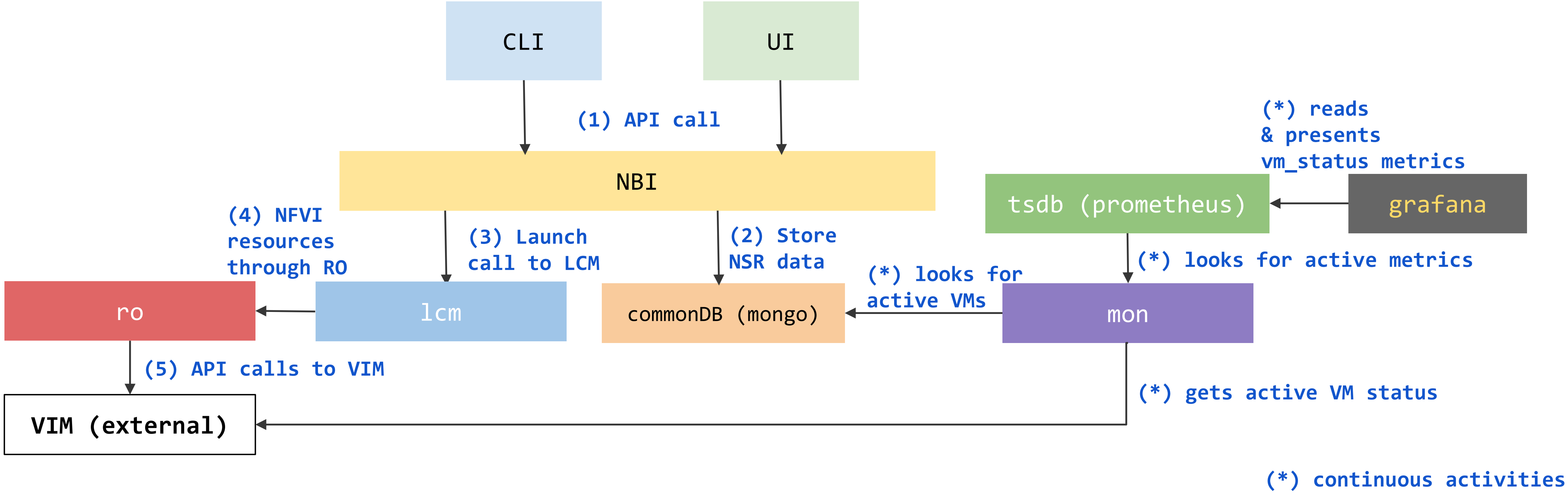
CLI Example: `osm k8scluster-add --creds myCredentials.yaml --version '1.15' --vim myVIM --description "My K8s cluster" --k8s-nets '{"net1": "myVIMnet"}' myK8Cluster`


CLI
UI
(1) API call
NBI
(3) Register K8s cluster
(2) Store K8 cluster data
lcm
commonDB (mongo)
(*) looks for active K8s clusters
mon
tsdb (prometheus)
grafana
(*) reads & presents k8scluster_status metrics
(*) looks for active metrics
(*) gets active k8scluster connection status
K8 Cluster (external)
(*) continuous activities

NS instantiation

When launching a new instance of a Network Service or Slice Instance (n x VNFs), the following components interact.

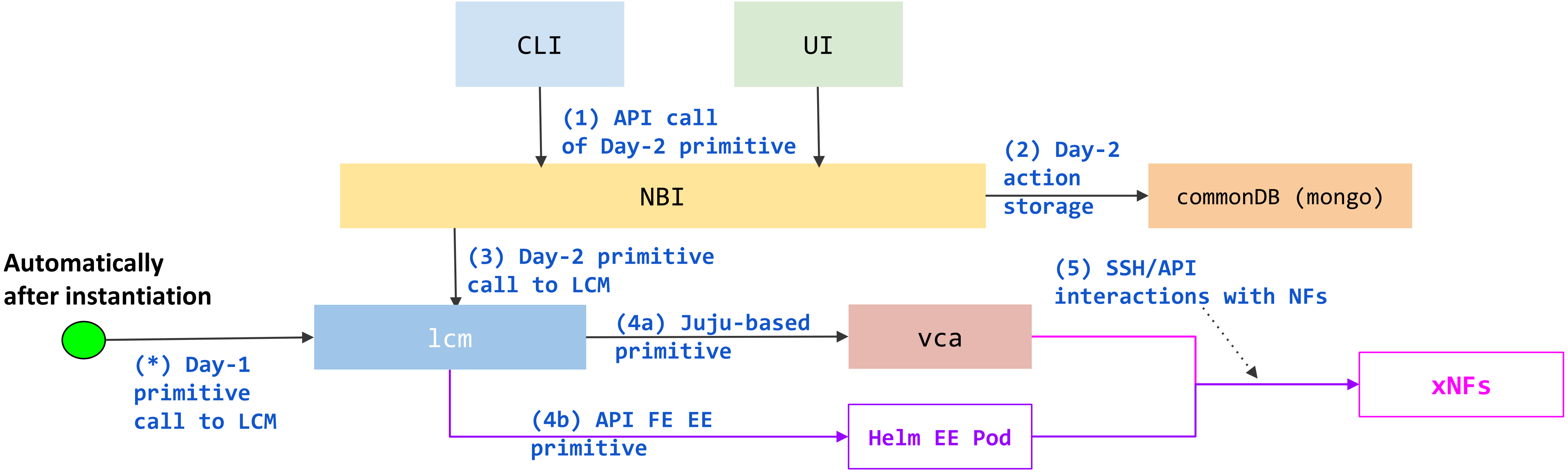
CLI Example: `osm ns-create --ns_name myNS --nsd_name myNSD --vim_account myVIM`



NS primitives

When launching a Day-2 primitive over a NS, the following components interact.

CLI Example: `osm ns-action myNS --vnf_name 1 --action_name myAction`


CLI
UI
(1) API call of Day-2 primitive
NBI
(2) Day-2 action storage
commonDB (mongo)
(3) Day-2 primitive call to LCM
Automatically after instantiation
(*) Day-1 primitive call to LCM
lcm
(4a) Juju-based primitive
vca
(4b) API FE EE primitive
Helm EE Pod
(5) SSH/API interactions with NFs
xNFs

Open Source MANO

Monitoring framework

What MON Collector Monitors?

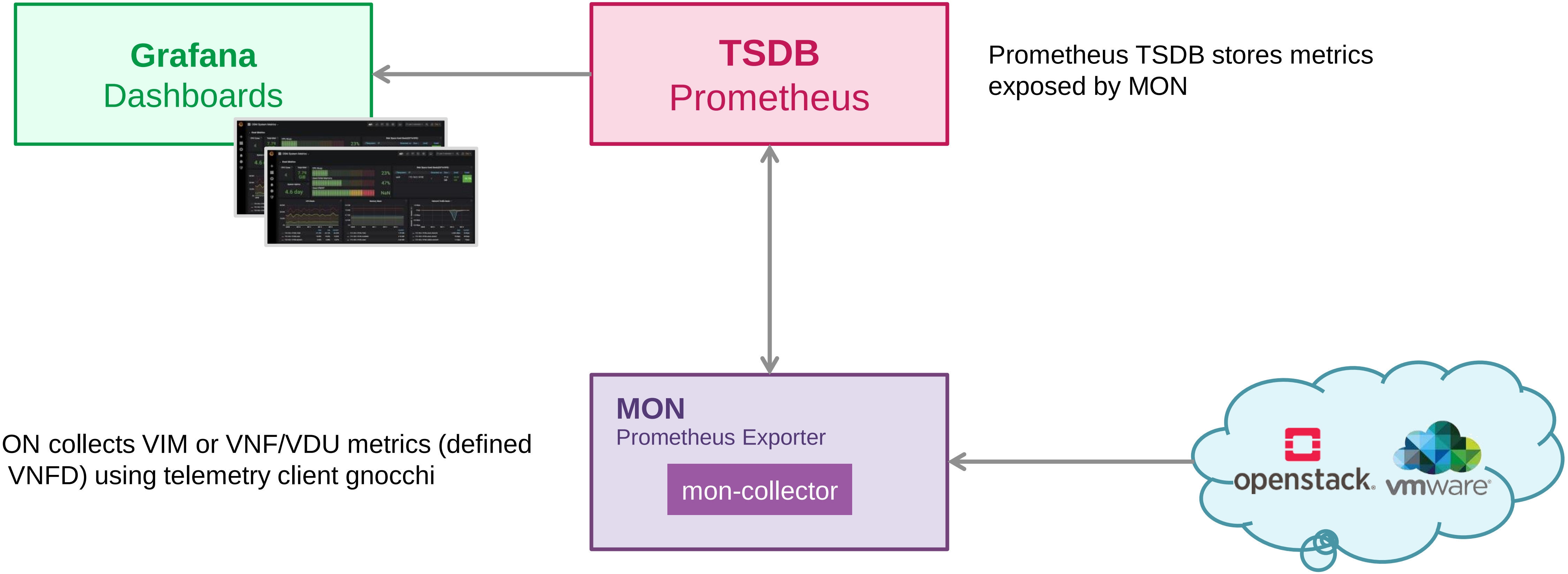
- VIM Metrics (Openstack & VMware)
- VNF/VDU Infrastructure Metrics (Openstack & VMware)
- VNF Metrics/Indicators (VCA)
- VNF Specific Metrics (using Prometheus Exporters)
- SDNC Metrics
- K8s Monitoring (Introduced in Release 11)

VIM & VNF/VDU Infrastructure Metrics Collection

- For infrastructure metrics to be collected, your VIM should support a telemetry system.
- From Rel 7.0 onwards, metric collection works with:
  - OpenStack **Ceilometer** and **Gnocchi** based telemetry services.
  - VMware vCD (vCloud Director) with **vROps** (vRealize Operations Manager)
- Openstack metrices collected are -
  - CPU utilization, memory utilization, disk read and write, packets sent and received, packets dropped etc.

VIM & VNF/VDU Infrastructure Metrics Collection

Provides analytics dashboards that shows metrics value from TSDB



Prometheus TSDB stores metrics exposed by MON

MON collects VIM or VNF/VDU metrics (defined in VNFD) using telemetry client gnocchi

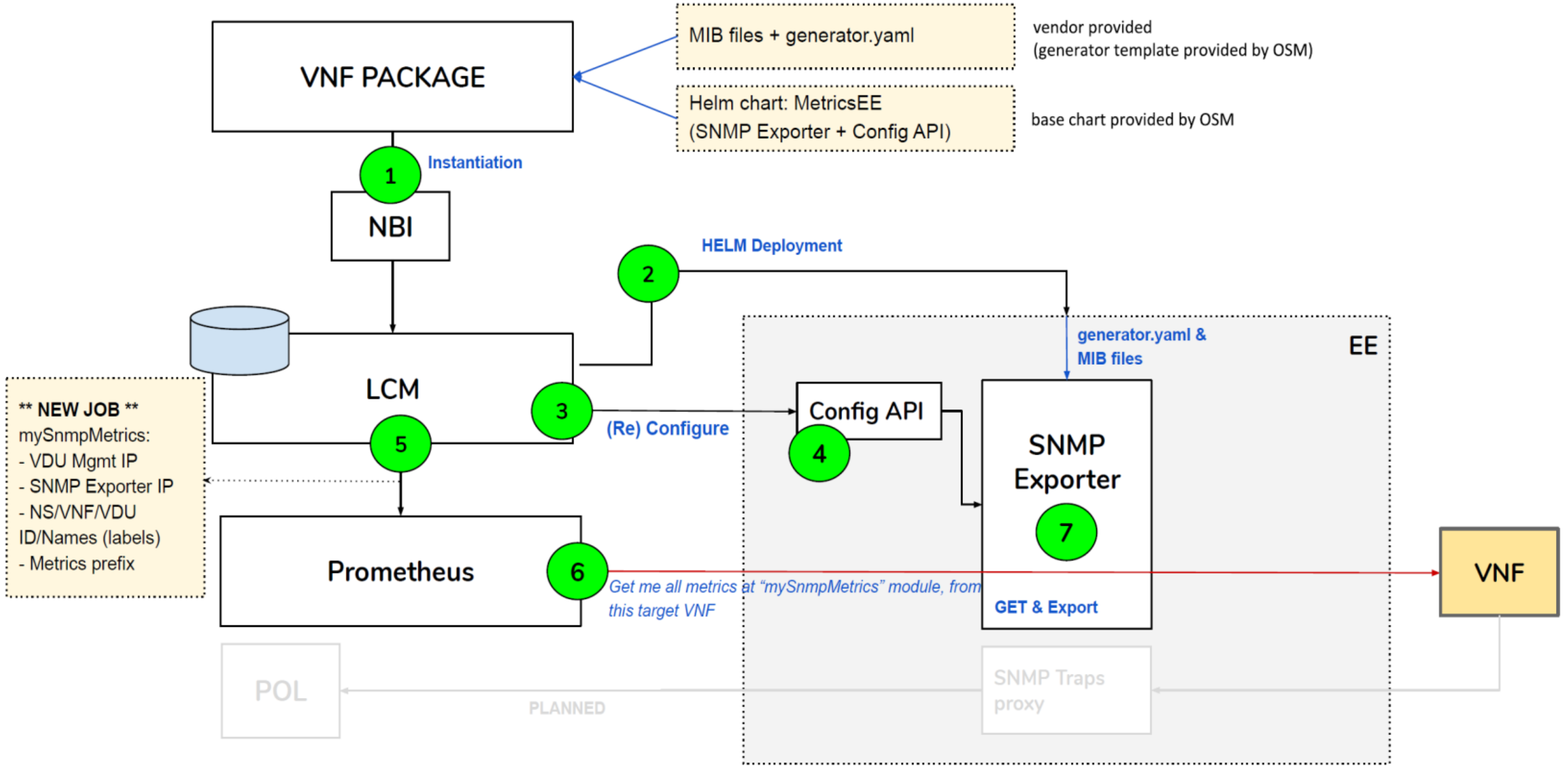
VNF Metrics/Indicators (VCA)

Open Source MANO

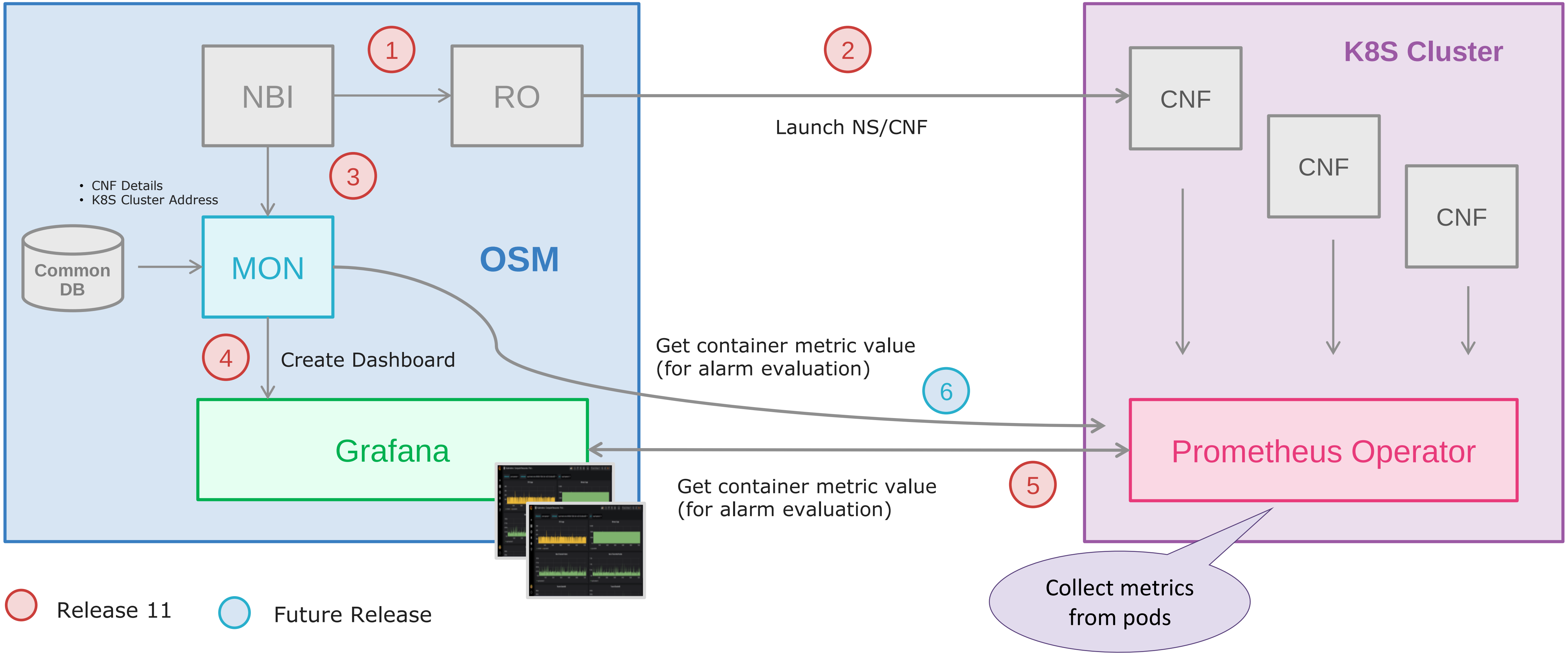
- Few VNF metrics can also be collected directly from VNFs using VCA (VNF Configuration & Abstraction) , through the Juju Metrics framework. Granularity is fixed to 5 minutes.
- Few example metrics that can be collected are:
  - Number of users
  - Load average
- Not a preferred approach for large scale metric collection as Juju is not designed for telemetry.

VNF Specific Metrics (SNMP Exporter)

VNF specific metrics collection is done by using **Prometheus Exporters** running as “**execution environments**”, which translate into PODs instantiated in the same K8s cluster where OSM runs. These PODs follow the VNF lifecycle and are dedicated to the collection of metrics. From Rel 8 OSM support has support for SNMP Exporters, to grab scalar provided by any SNMP MIB/OID.



K8s Monitoring



OSM

NBI

1

RO

2

Launch NS/CNF

K8S Cluster

CNF

CNF

CNF

3

- CNF Details
- K8S Cluster Address

Common DB

MON

4

Create Dashboard

Grafana

Get container metric value (for alarm evaluation)

6

Prometheus Operator

Get container metric value (for alarm evaluation)

5

Collect metrics from pods

○ Release 11

○ Future Release

Open Source MANO

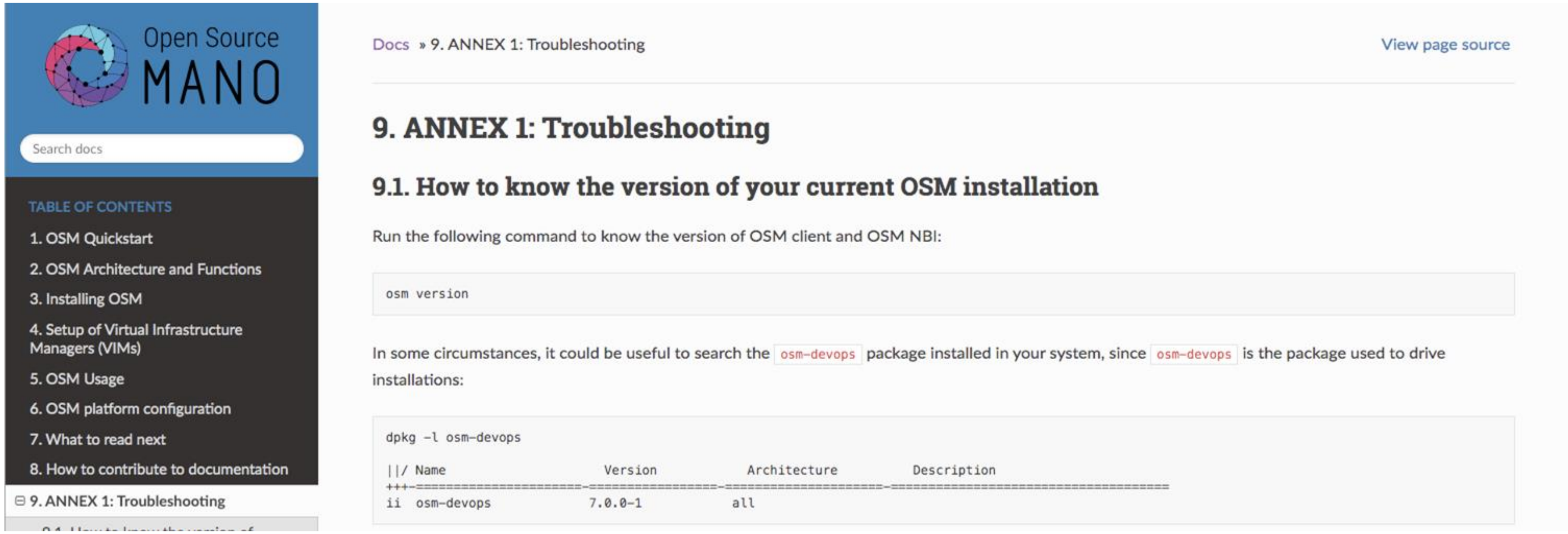
Troubleshooting

Troubleshooting OSM

A general approach for OSM Troubleshooting is to first look for error messages in “show” commands, as in:

```
osm ns-show [ns]
osm vim-show [vim]
```

Besides that, knowing which components interact for each operation, you can troubleshoot by looking at the logs of each component. All troubleshooting tips are being documented in the user guide, here: https://osm.etsi.org/docs/user-guide/09-troubleshooting.html



9. ANNEX 1: Troubleshooting

9.1. How to know the version of your current OSM installation

Run the following command to know the version of OSM client and OSM NBI:

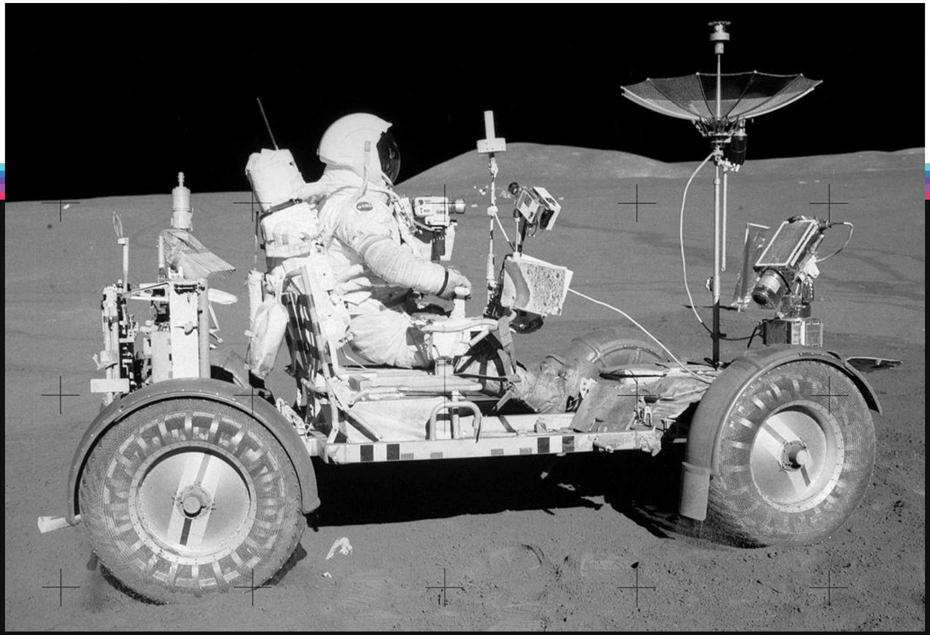
```
osm version
```

In some circumstances, it could be useful to search the `osm-devops` package installed in your system, since `osm-devops` is the package used to drive installations:

```
dpkg -l osm-devops

||/ Name                   Version           Architecture         Description
+++-======================-=================-====================-=====================================
ii  osm-devops             7.0.0-1           all
```

Open Source MANO



For more information:

osm.etsi.org
osm.etsi.org/wikipub
osm.etsi.org/docs/user-guide