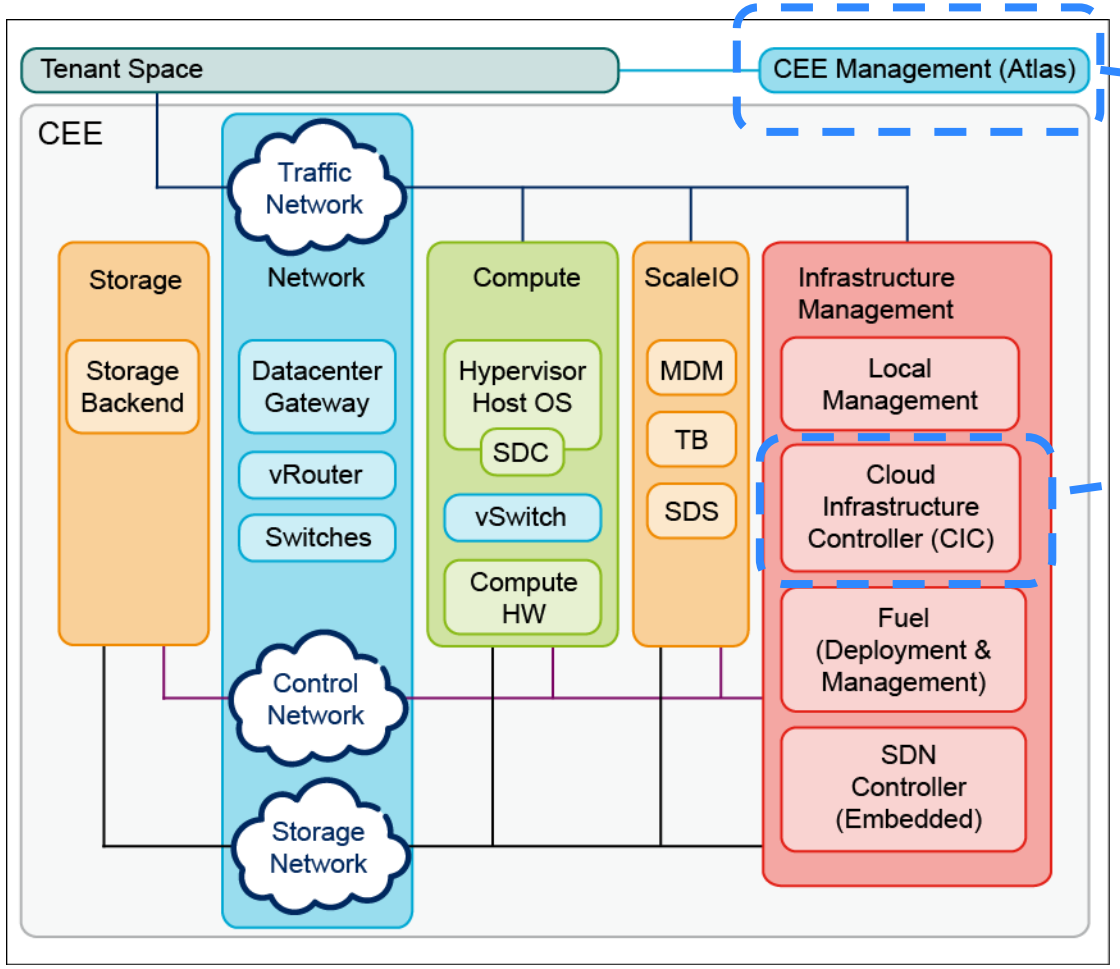
July 3, 2017
IEEE CQR - ERT

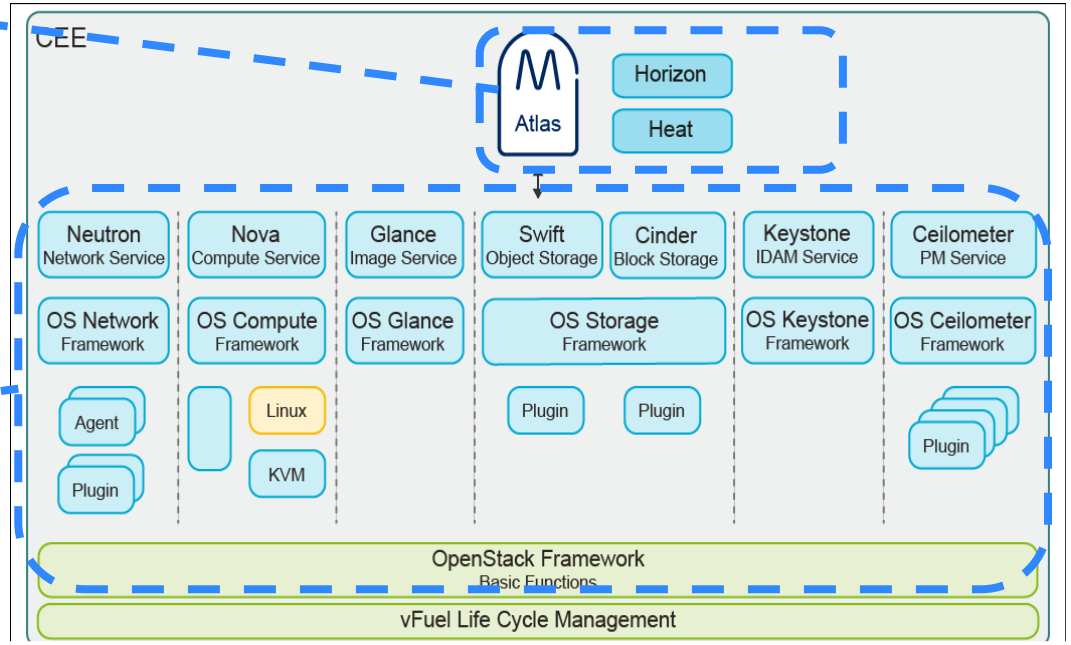
ABSTRACT

›Especially with stateful telecom virtual network functions (VNFs), OpenStack “out of the box” does not meet reliability requirements for those applications running in a Cloud. Improvements to detection and triggering actions for infrastructure faults combined the recovery mechanisms for applications running in a Cloud have provided the service reliability needed.

›In Ericsson, “Continuous Monitoring – High Availability” (CM-HA) uses a 'recreation' automatic recovery action to ensure that the configured VM availability level is met. Collaborating in OPNFV Project Doctor, Ericsson along with various groups are aligning on a solution to upstream to OpenStack. This presentation will cover additions to work with OpenStack components to deliver high availability for VNFs as well as the summarize the current work in OPNFV.

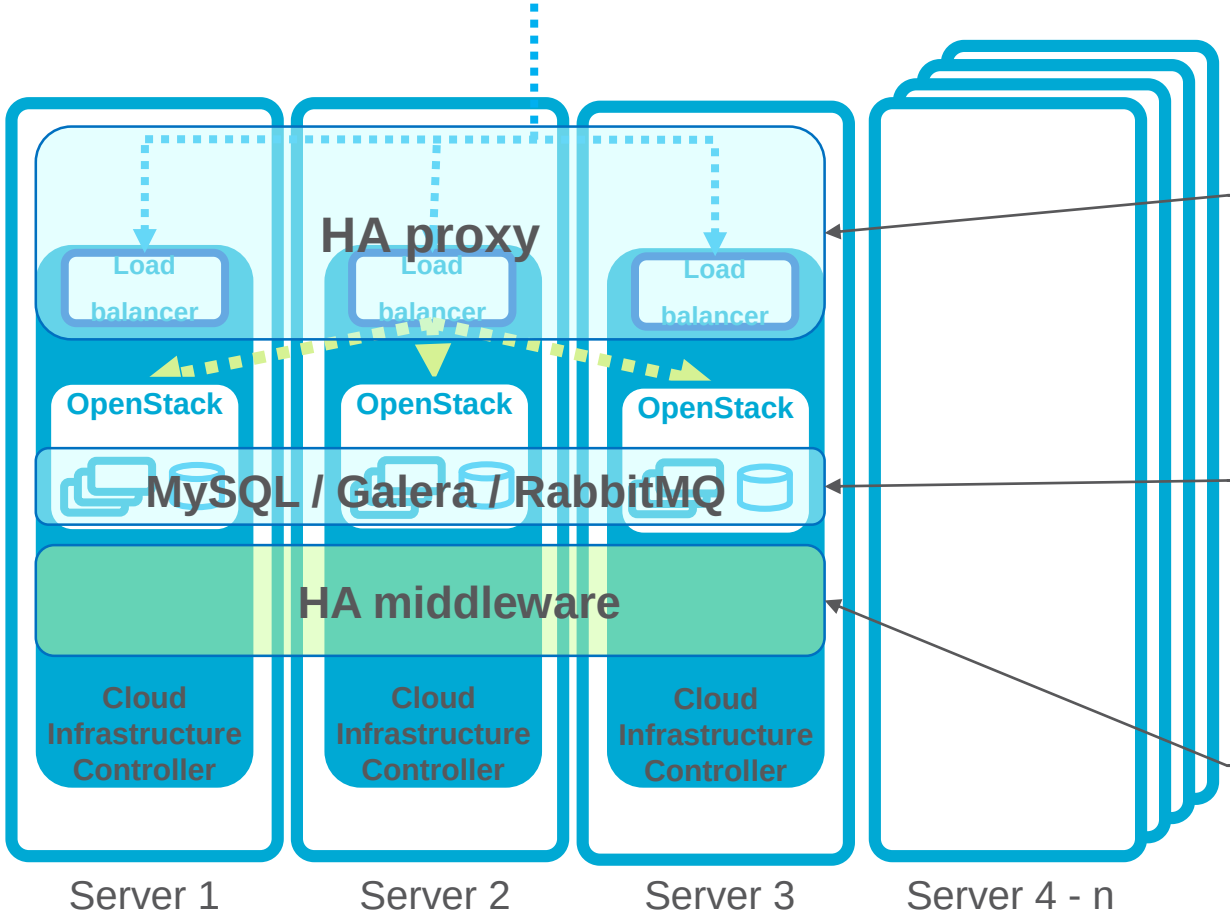
CEE ARCHITECTURE





Ericsson's CEE is based on OpenStack and includes a number of extensions

RESILIENT CLOUD INFRASTRUCTURE CONTROLLER CIC (3X)



Can sustain single HW faults

Load balancing functions using **HA Proxy** nodes ensures robust and scalable access to the OpenStack functions.

High-available and consistent configuration and state storage is provided by **MySQL** and **Galera**. **RabbitMQ** ensures robust and scalable transaction handling by mirroring message queues.

High availability middleware ensures service availability of the controller functions in the event of errors that affect parts of the CIC cluster.

RESILIENT CLOUD INFRASTRUCTURE CONTROLLER (3X)

- **HA middleware** embraces several functions. It contains the following functions
  - **Corosync** uses Totem protocol, which is an implementation of Virtual Synchrony protocol. It uses it in order to provide connectivity between cluster nodes, decide if cluster is quorate to provide services, to provide data layer for services that want to use features of Virtual Synchrony.
  - **Pacemaker** is the cluster resource manager used to manage Neutron resources, HAProxy, virtual IP addresses and MySQL Galera cluster. It is done by use of Open Cluster Framework (see http://linux-ha.org/wiki/OCF_Resource_Agents) agent scripts which are deployed in order to start/stop/monitor Neutron services, to manage HAProxy, virtual IP addresses and MySQL replication
  - **Compute Monitoring – High Availability (CM-HA)** The CM-HA is a daemon process which monitors ECS resources. Periodically checks the status of Compute Nodes and CICs. When a failing is detected it sends alarm, and if it is possible (The VMs evacuation policy permits) then evacuates the VMs. **This is an Ericsson enhancement**
  - **Ubuntu watchdogs** are set to detect hanging system processes. See http://manpages.ubuntu.com/manpages/lucid/man8/watchdog.8.html
  - **Upstart** is an event-based replacement for the /sbin/init daemon which handles starting of tasks and services during boot, stopping them during shutdown and supervising them while the system is running. **Ericsson has extended the configuration to enable proper supervision of all services on the CICs**

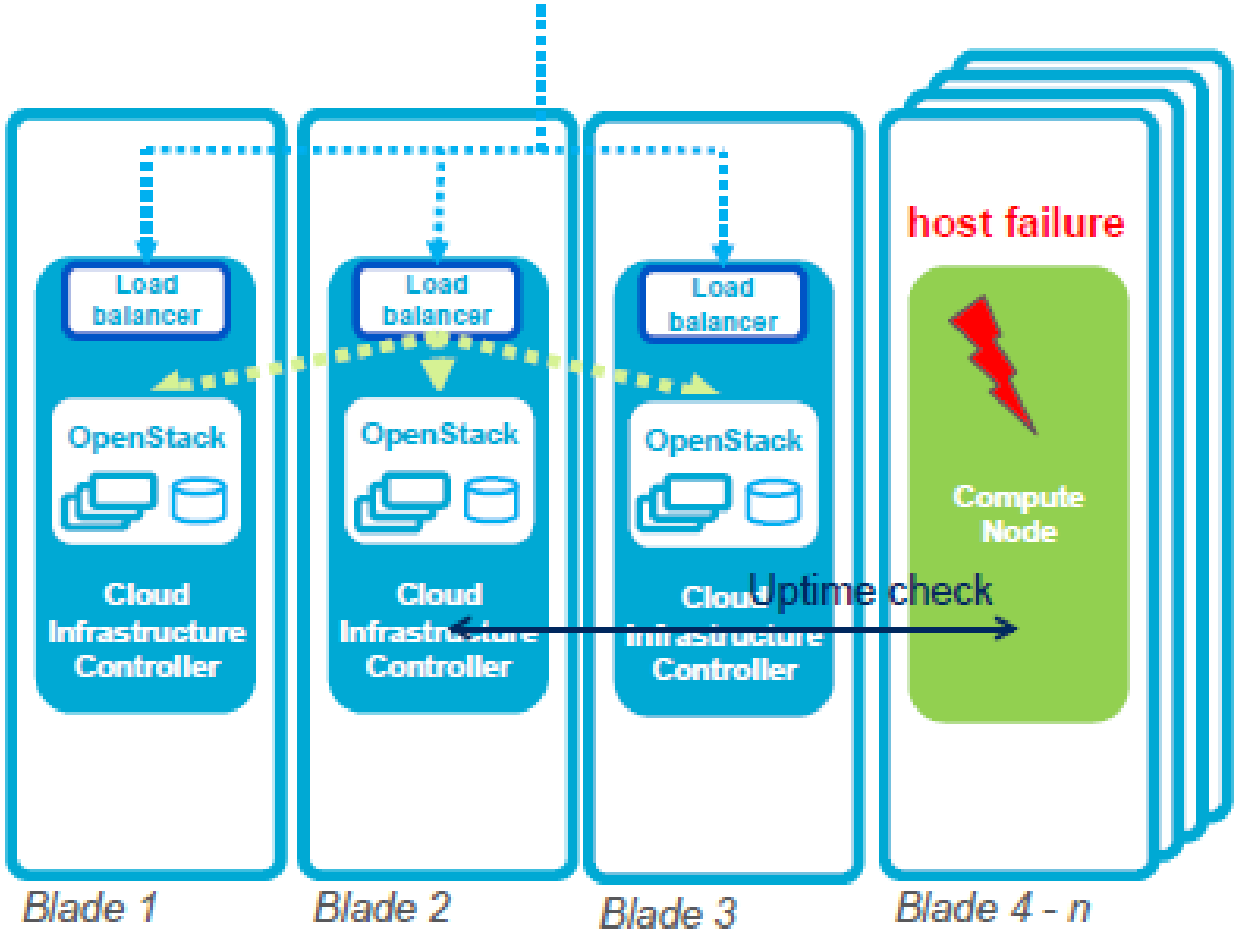
WHAT IS CM-HA?

- CM-HA ***detects*** problems by continously monitoring resources
- CM-HA ***corrects*** problems by informing operators and evacuating problematic resources
- CM-HA ***recovers*** the system by restarting services according to individual policies after failure

- CM-HA is very low impact
  - One system component on the control node
  - Uses existing system features for monitoring

- CM-HA is NOT a backup-restore solution

CMHA, COMPUTE MONITORING HIGH AVAILABILITY



One of the CICs monitors periodically (default every 5 seconds) all compute nodes. In case of CIC failure another CIC takes over the monitoring.

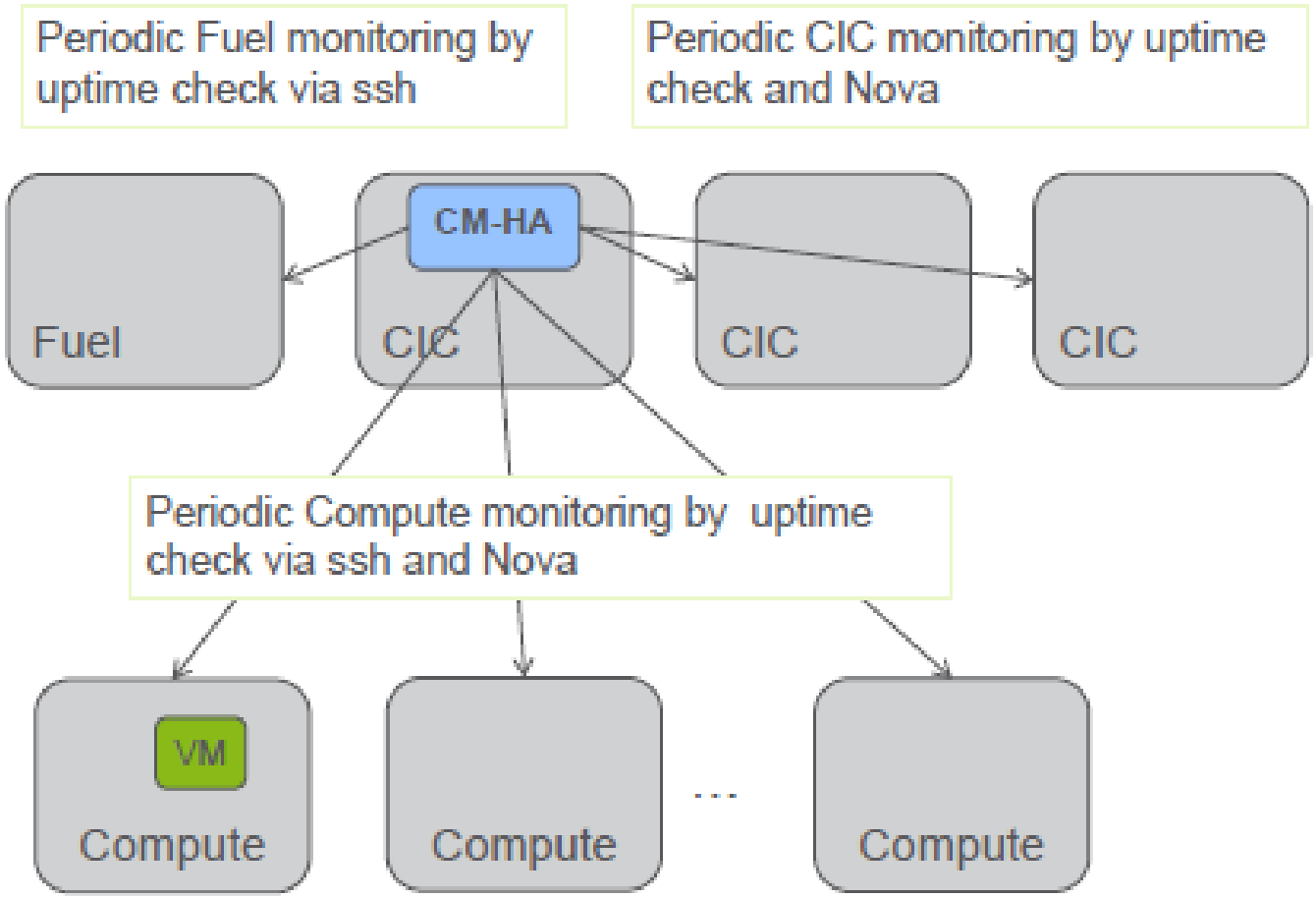
In case a compute node unavailability is detected it orchestrates automatic recovery of (tenant) VMs by evacuation and restart/resume actions.

VM High Availability policies per VM to specify VM behavior at recovery actions by CM-HA

Fencing of compute nodes assures that no duplication of VM happen even in unlikely case of control network failure (where the monitoring happens).

CM-HA is being aligned with OPNFV Doctor activities. Current differences (healing engine) will be upstreamed to become fully aligned.

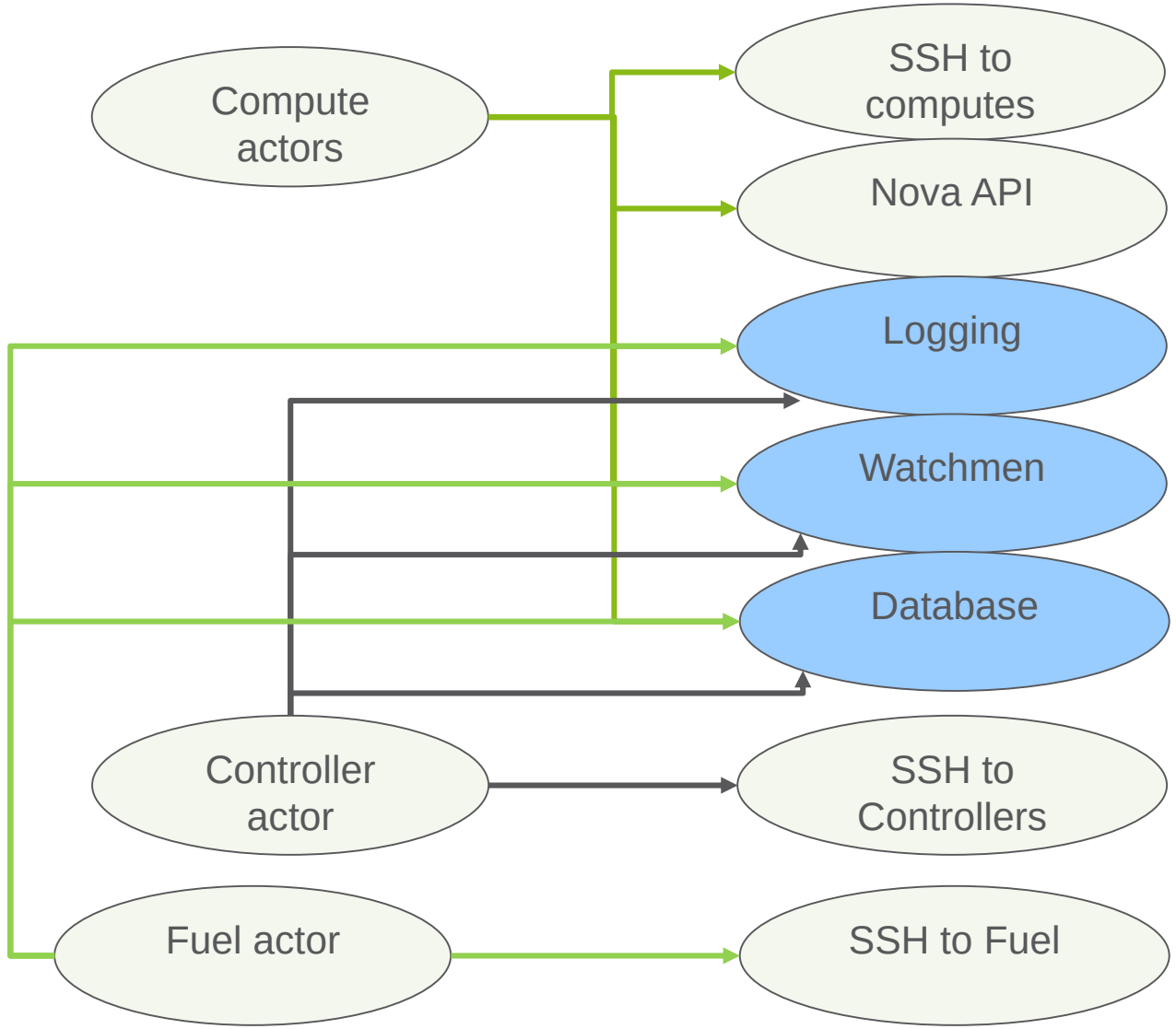
CM-HA (COMPUTE MONITORING – HIGH AVAILABILITY) - CIC, FUEL AND COMPUTE HOST MONITORING



- One of the CICs **periodically monitors** all compute hosts.
- CM-HA is running active-standby-standby in a CIC
- CM-HA monitors hosts via the Control Network.
- In case compute node unavailability is detected it orchestrates **automatic recovery** of (tenant) VMs by evacuation and restart/resume actions. Alarm VM unavailable issued for VMs running on a failed compute host
- In case of CIC or Fuel Node is unavailable **alarms and recovery action** are initiated

CM-HA ACTORS

- On startup, CM-HA launches distinct processes on components which are called **actors**. These processes are physically running in the controller node as threads. No code is installed on any other part of the system on behalf of CM-HA
- Currently there are three types of Actors:
  - 1. Controller (CiC) Actor: Checks the cluster status of CICs with "crm-node -l" command. CIC Actor connects to CIC to get uptime via SSH. Compares that with the stored value in order to detect if a CIC has been rebooted.
    2. Compute Actors: Checks nova service availability and node connectivity.
    In case of error it starts VM evacuation, it just observes evacuation policy settings of the VM. For evacuation it uses Nova Forcemove API.
    Compute Actors also connects to Compute Nodes to get uptime.
    The uptime check is implemented to detect Compute Node failures during CM-HA downtime (e.g. all controllers are rebooted).
    3. Fuel actor: Check the connectivity to Fuel over SSH
- All actors are logging into a file or use RSyslog. All actors sends alarms, and uses a mysql database called 'cmha'.



MONITORING

- Nodes monitored by CM-HA are internally represented as state machines. CM-HA uses a range of checks to identify state changes of the network nodes, these changes cause state transitions of the corresponding state machines. State transitions are linked with certain actions, such as sending alerts or performing VM evacuations. CICs, Compute nodes and Fuel are represented using the same base state machine, the differences are in how state transitions are triggered and what kinds of actions they are associated with.
- State transitions are governed by a function which returns a single boolean value, indicating if the node is online or not. There are four states:
  - **Unknown** - this is the initial state. If the node in this state is reported online, it immediately transitions to available state. If the node is consecutively reported offline for a predefined number of times, it transitions to unavailable state.
  - **Available** - the logic for transition from this state is the same as for unknown, except when the node is reported online, there is no state transition. If the node is consecutively reported offline for a predefined number of times, it transitions to unavailable state.
  - **Unavailable** - a temporary state between available and failed. In this state, if the node is reported online, it immediately transitions to available state. If the node is continuously reported offline for a predefined period of time, it transitions to failed state.
  - **Failed** - a final state for the node which is continuously reported offline. If the node is reported online, it immediately transitions to available state.
- CM-HA keeps all variables and timings associated with the state machines stored persistently in the system database. Thus, CM-HA does not rely on locally kept state for anything important to its task. As a result, if the CM-HA daemon dies and is restarted on the same, or a different node, it will resume its operation without loosing any information from the previous run (Note: this functionality is missing for VMs during migration and restore).

COMPUTE ACTOR

- This actor is monitoring the Compute Nodes. If a failing is detected, CMHA powers off the compute host, then it starts to evacuate VMs. After evacuation process CMHA try to power on the host. In case of successful node recovery, CMHA recovering VMs with redefine and reboot commands. This actor also sends Compute Host Failed alarm and Compute Host Restarted alert.
- StateMachine of Compute Actor
  - When the Compute actor starts it compares current compute uptime with previously stored one. CM-HA can detect that there was a compute node reboot while it was stopped. In this case it restarts VMs on the mentioned node.
  - When the Compute actor enters **Available** state, "Compute Host Restarted" alert is issued, stop ongoing evacuations(EvacuationHelper thread) if it still running, and recover VMs.
  - When the Compute actor enters **Unavailable** state, store active instances for later recovery action, and start EvacuationHelper thread.
  - When the Compute actor enters **Failed** state, "Compute Host Failed" alarm is raised.
  - When the Compute actor leaves **Failed** state, "Compute Host Failed" alarm is ceased.
- Recovery action
  - Only managed VMs will be recovered. Firstly it redefines the VM, then it tries to restarts the VM multiple times if this VM was running previously.

CONTROLLER ACTOR

- Controller state transitions are governed by an availability test.
- There are two metrics used by the test:
  - Node uptime query using SSH
  - Corosync cluster membership.
- If the uptime query is successful, and the cluster membership is confirmed, the controller node is presumed online.
- StateMachine of CIC Actor State Transitions
  - When CIC actor enters **Available** state, "CIC Restarted" alert is issued.
  - When CIC actor enters **Failed** state, "CIC Failed" alarm is raised.
  - When CIC actor leaves **Failed** state, "CIC Failed" alarm is ceased.
  - When the CMHA daemon starts, it will check if all previously known CIC nodes have restarted. If this is true, "Complete CIC Service Restarted" alert is issued.

FUEL ACTOR

- Fuel state transitions are controlled by just one metric: uptime query via SSH.
  - When Fuel actor enters **Available** state, "Fuel Restarted" alert is issued.
  - When Fuel actor enters **Failed** state, "Fuel Failed" alarm is raised.
  - When Fuel actor leaves **Failed** state, "Fuel Failed" alarm is ceased.

EVACUATION HELPER THREAD

- Before evacuation it powers off the host, then starts evacuation. After evacuation it tries to power on the host, finally the thread stops.
- It sends “VM Unavailable” alarm for every VM hosted on the failed compute.
- CMHA detects evacuation policy of the VM. If ha-offline policy is set it tries to evacuate VMs from the failed node.
- Evacuation will be done in the order of the VMs creation time.
- It waits 30sec for every VM evacuation, if it is not finished then it starts a new evacuation.
- The maximum number of parallel evacuation can be configured.
- Any unfinished evacuations will be checked again in the next round (ThreadActor logic call step in every 5 sec).
- For evacuation the Nova Forcemove API is used.
  - If there are persisted scheduler hints defined for the VM forcemove tries to apply them.
  - If the scheduler cannot find a new host for the VM, because of the hints, the VM evacuation will fail.
  - In case of evacuation failure it will send “VM Evacuation Failed” alarm for that VM.
  - If “NoEvacuation” was set then VM Evacuation Failed alarm is sent.

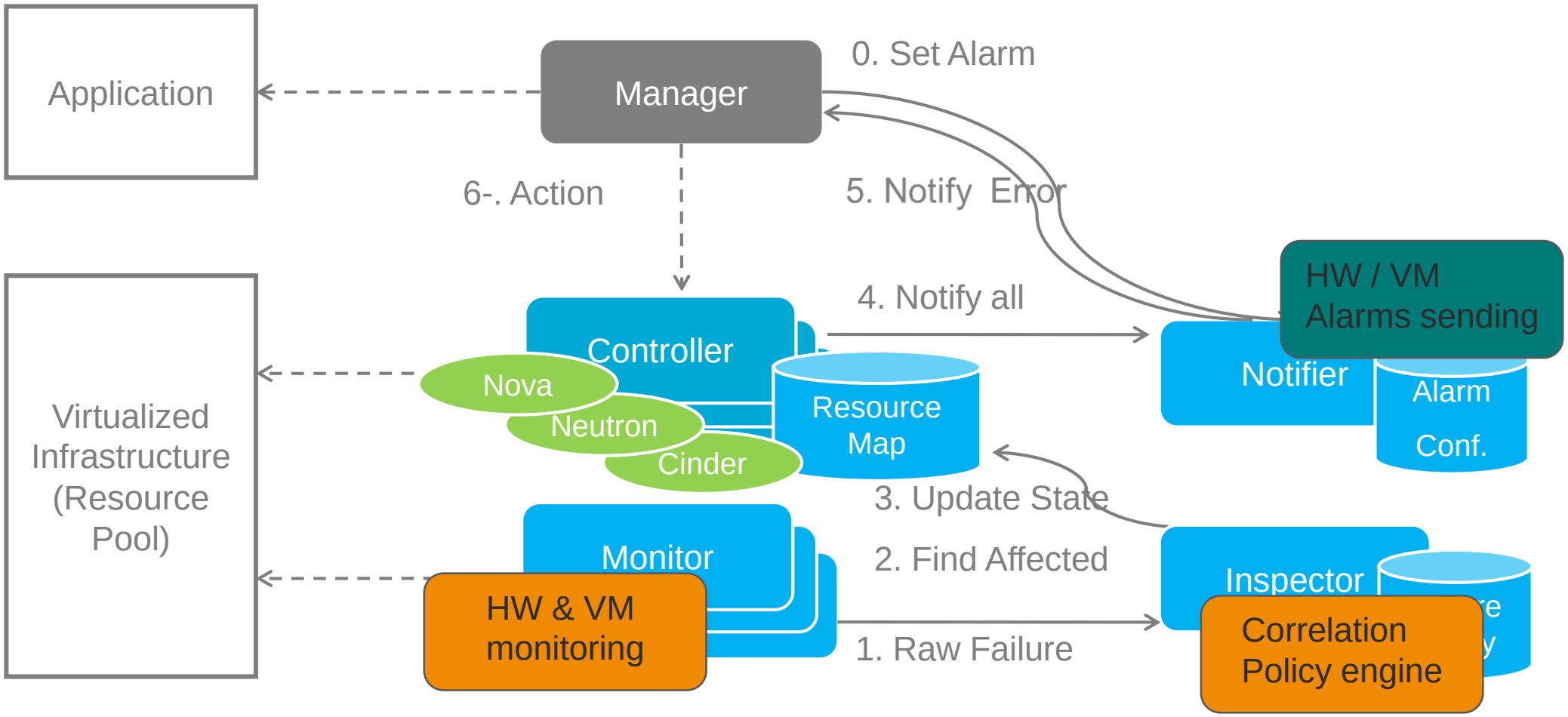
THE ALTERNATIVES

- There are alternatives being worked in the open source community.
- The prominent contenders being:
  - OPNFV Doctor
  - OpenStack Freezer
  - OpenStack Masakari

OPNFV DOCTOR

- Well documented and fully accepted as a Openstack solution.
- Limited use-case: detection & notification only
- Framework – not finished component

DOCTOR – CMHA MAPPING



ALIGNMENT WITH OPNFV DOCTOR

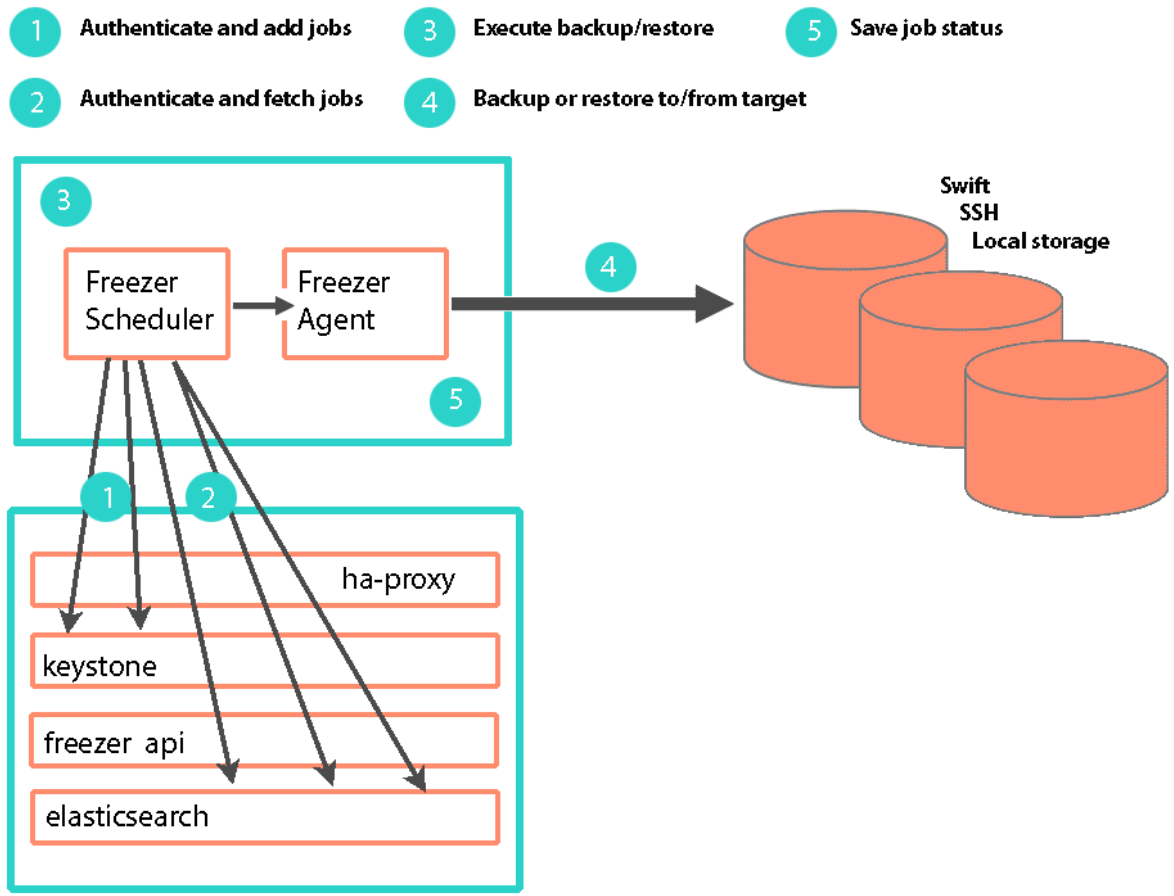
- Ericsson is taking part in the upstream implementation of OPNFV Doctor project
  - Service status notification implementation in Nova
  - Monasca contribution
- To align with OPNFV doctor we've made an exit plan for CMHA
  - Use standardized open source components as much as possible
  - Upstream the remaining part ("healing engine")

OPENSTACK FREEZER

- Freezer is a distributed backup restore and disaster recovery as a service platform.
- Designed to be multi OS (Linux, Windows, OSX, *BSD)
- Large scope (block based backups, file based incremental backups, point-in-time actions, jobs synchronization etc)
- Uses physical agents and schedulers on monitored nodes

FREEZER ARCHITECTURE

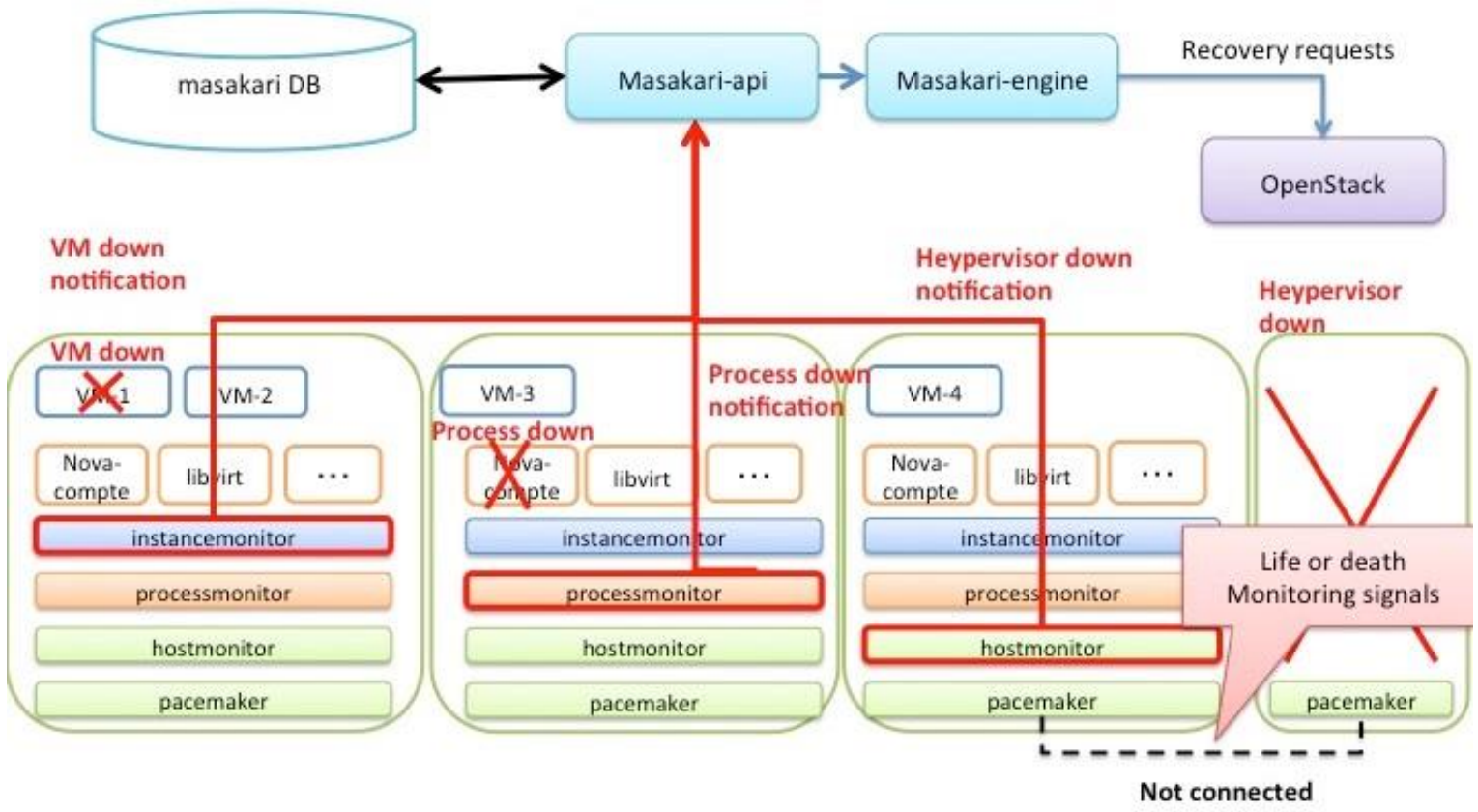
Backup/recovery architecture



OPENSTACK MASAKARI

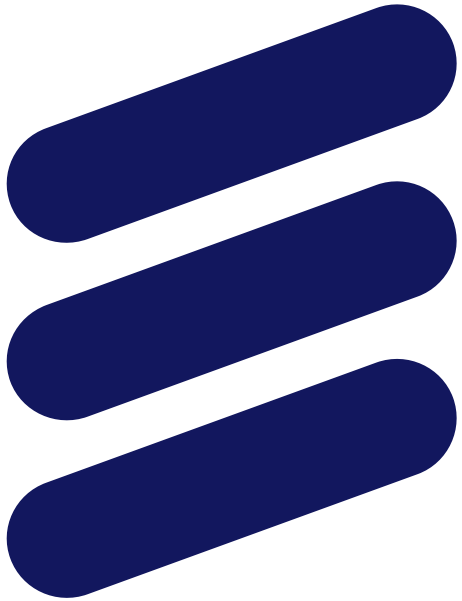
- Architecture close to CM-HA
- Provides Virtual Machine High Availability (VMHA) service for OpenStack clouds by automatically recovering the KVM-based Virtual Machine (VM)s from failure events such as VM process down, provisioning process down, and nova-compute host failure.
- Also provides API service to manage and control the automated rescue mechanism.

MASAKARI ARCHITECTURE



CM-HA CONCLUSIONS

- OpenStack can be hardened create high availability for VNFs
- Basic principles (detect, correct, and recover) apply
  - Need to leverage OpenStack components
- Open source community is now responding



ERICSSON