How to deploy an App into RH OpenShift on IBM zSystems to scale automatically

Wilhelm Mild
IBM Executive IT Architect
Containerization, RH OpenShift on IBM zSystems & LinuxONE.
IBM R&D Lab Boeblingen, Germany

Red Hat OpenShift and positioning

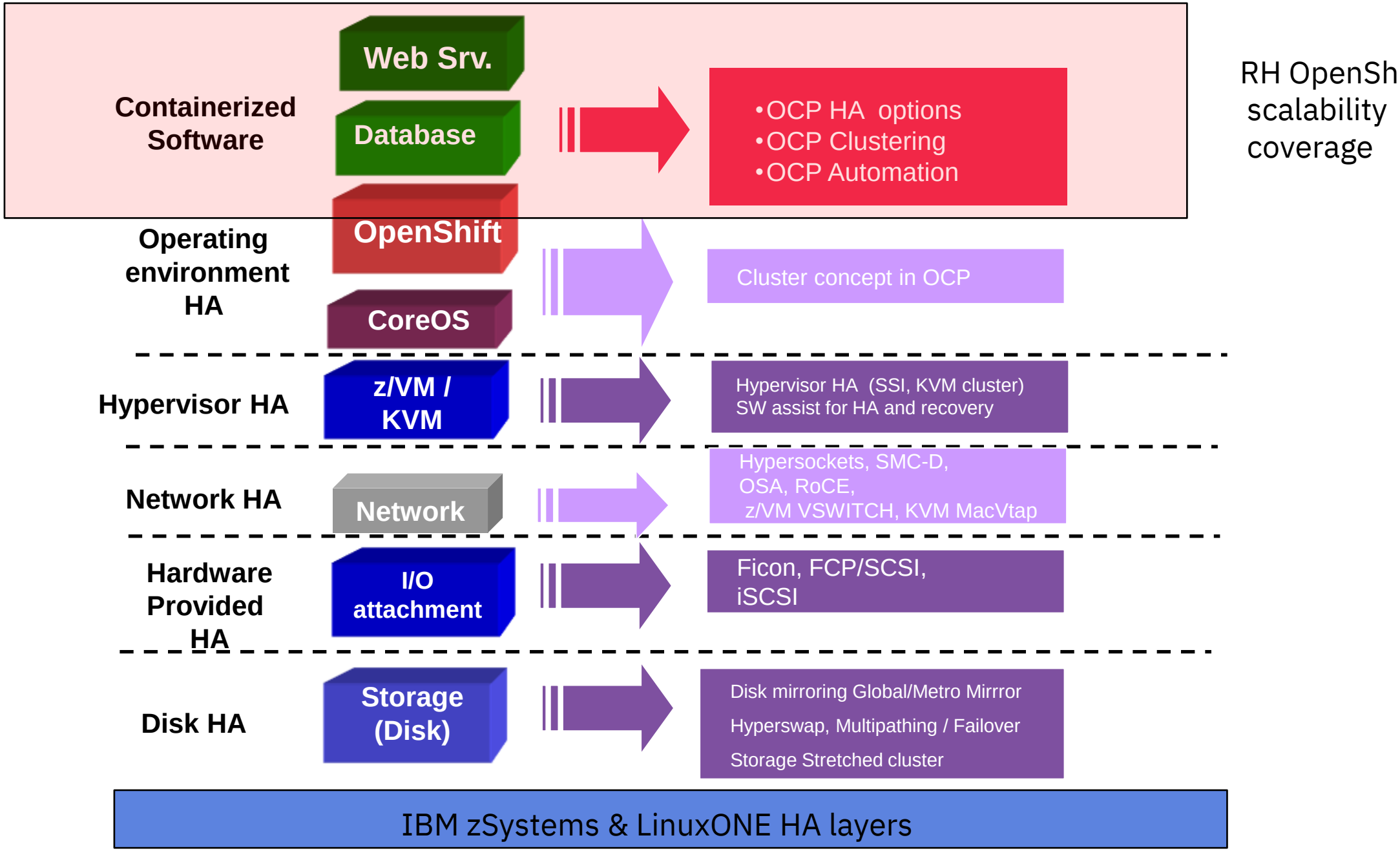
OpenShift is designed as an **application runtime environment** based on **containers** with **build-in capabilities for services** to provide **availability and scalability** based on **unreliable HW**.

OpenShift expects unreliable hardware and adds resilience by orchestrating containers and pods

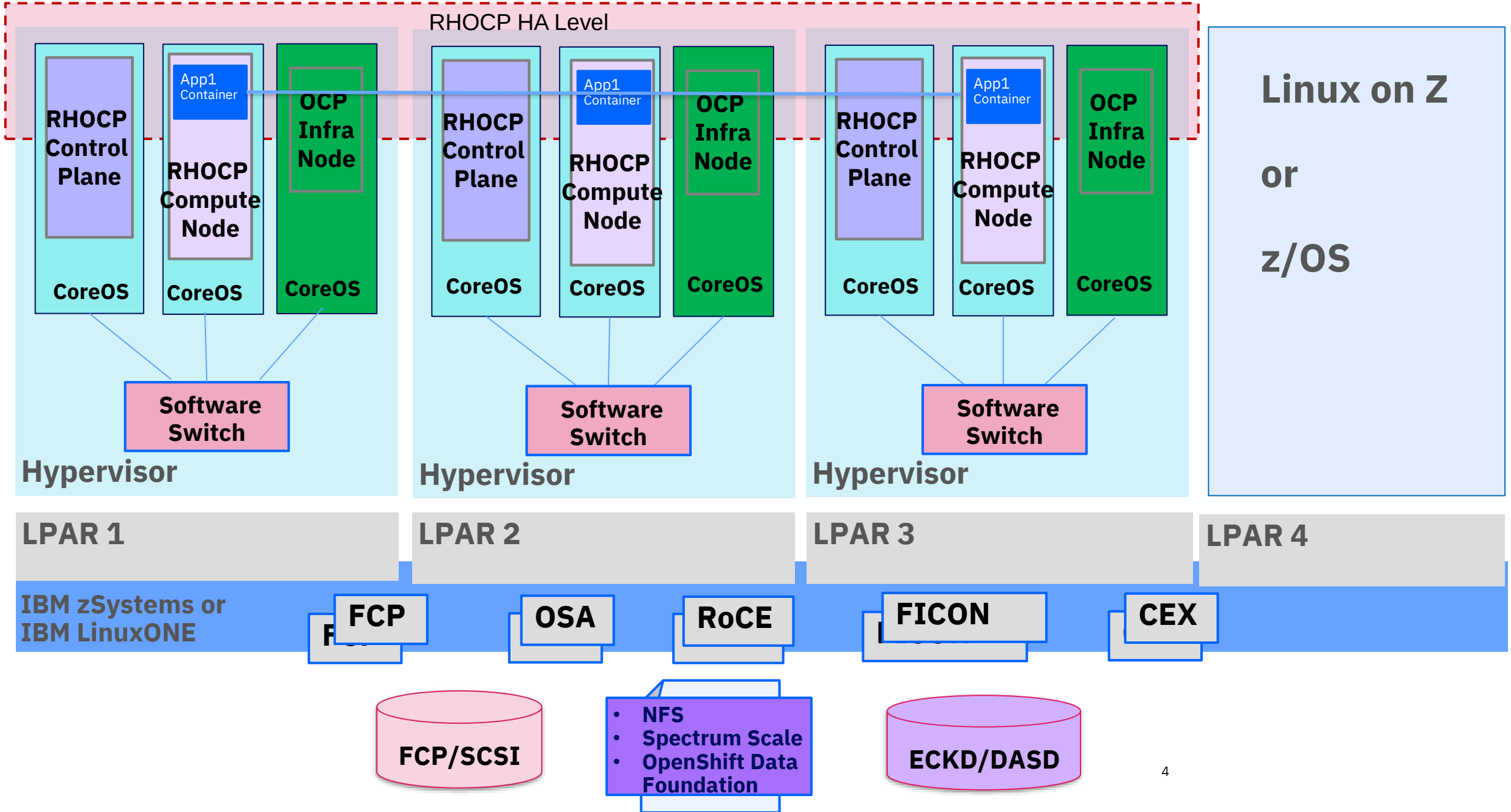
Containers assume strongly simplified life-cycle

→ **orchestration** is extremely easy and scalable (e.g. restart, create more instances, move between servers)

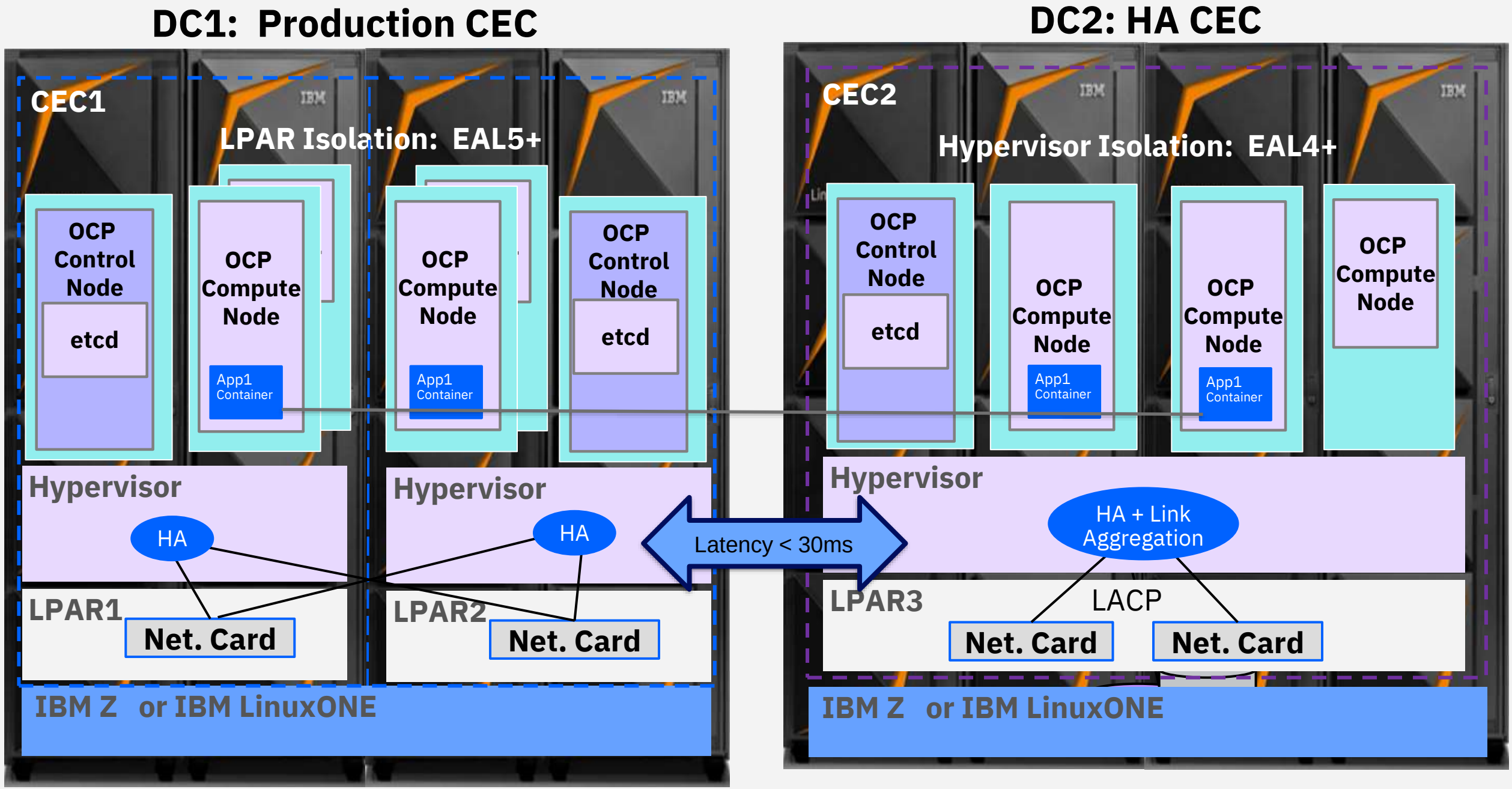
Layered Components of HA with Openshift on IBM zSystems



Red Hat OpenShift – **application** placement



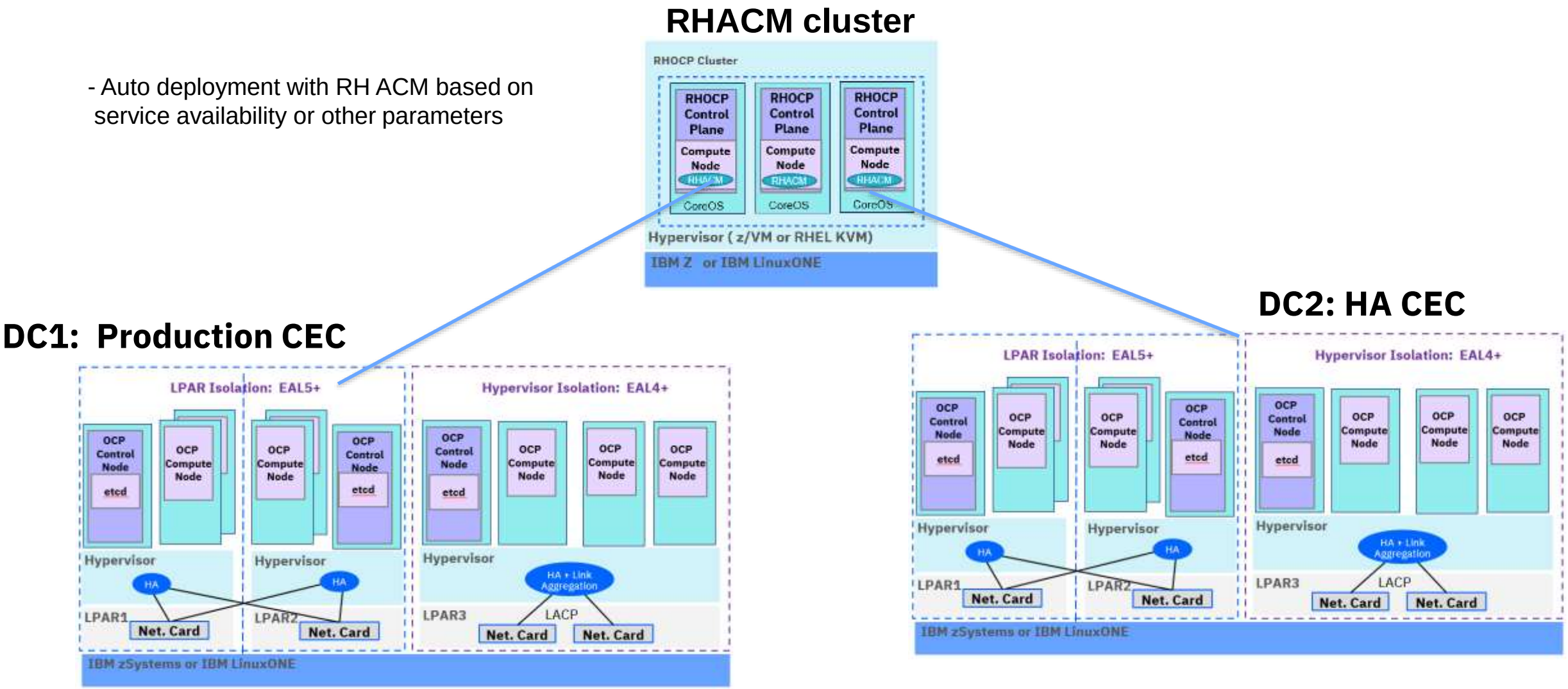
(2) Services/Apps HA Architectural pattern for Red Hat OpenShift



(3) Services/Apps HA Architectural pattern for Red Hat OpenShift

Red Hat Advanced Cluster Manager (RHACM) as Workload distributor / balancer

- Auto deployment with RH ACM based on service availability or other parameters



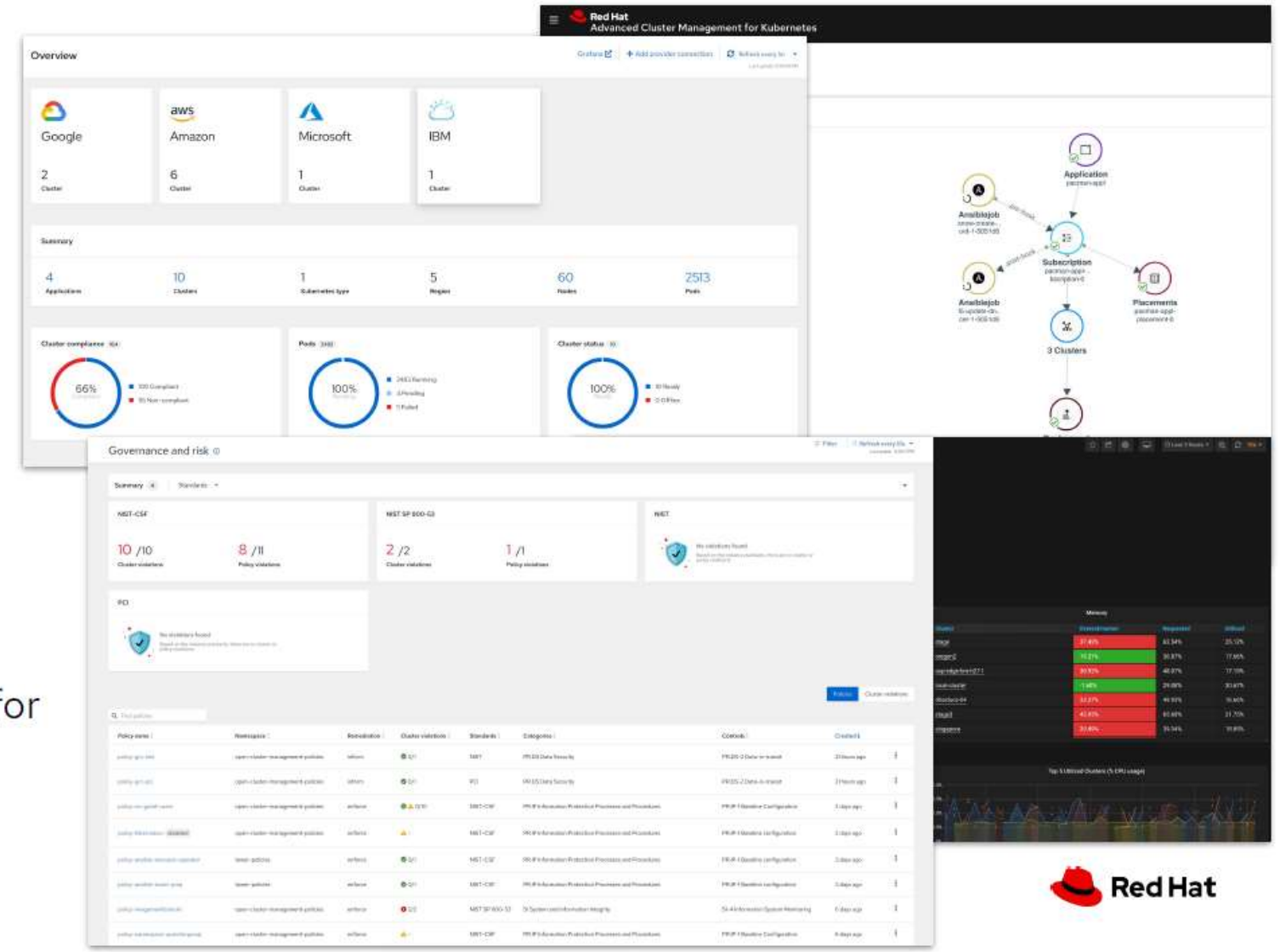
Red Hat Advanced Cluster Manager Overview

Multicluster lifecycle management

Policy driven governance, risk, and compliance

Advanced application lifecycle management

Multicluster observability for health and optimization



Red Hat

RH OpenShift – application scaling configuration options

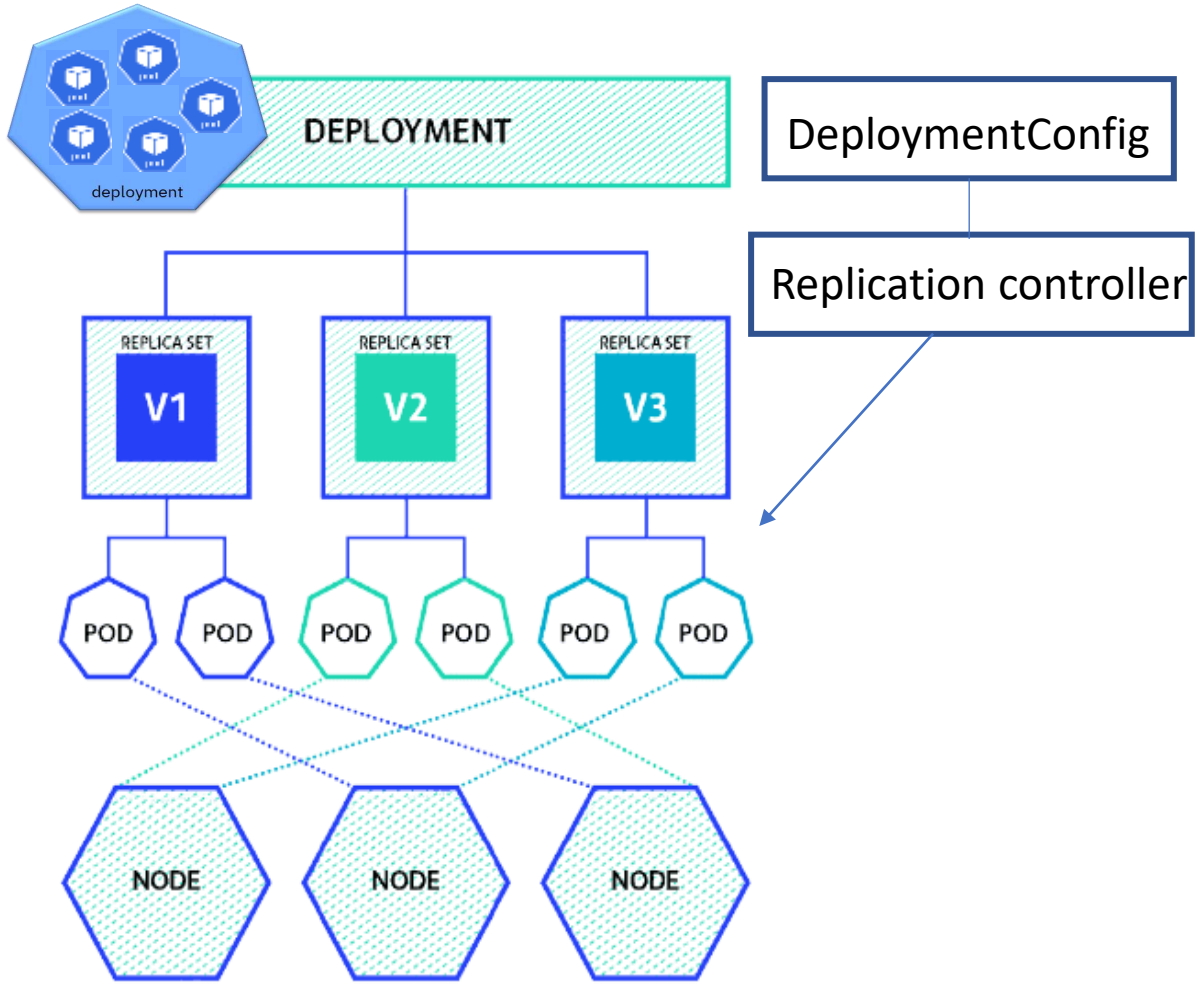
- **Scalability for Container** workloads – can be configured via:
  - **ReplicaSet -** ensures that a specified number of pod replicas are running at any given time
  - **HPA – Horizontal Pod Autoscaler** - adjusts the number of replicas of an application. It scales (increase/decrease) **replica set** of applications in the Pods according to the CPU and Memory utilization of the pods.
  - **VPA – Vertical Pod Autoscaler** - adjusts the resource requests and limits of a container.
    It scales cpu and memory resources for pods and can update the resource limits and requests based on the metrics it learns.
  - **Pod placement scheduler** - OpenShift allows to change how pods are placed on compute nodes using several placement profiles (Default scheduler profile, LowNodeUtilization, HighNodeUtilization, NoScoring) profile

https://docs.openshift.com/container-platform/4.13/nodes/index.html

RH OpenShift - Understanding Deployment and DeploymentConfig objects

RH OpenShift Container Platform leverages the Kubernetes concept of a pod, which is one or more containers deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed.

- One or more pods, represent an instance of a particular version of an application.
- **Deployments** and **Deployment configs** are enabled by the use of native Kubernetes API objects **ReplicaSet** and **replication controllers** respectively, as their building blocks.
- Users do not have to manipulate **replication controllers**, **replica sets**, or pods owned by **DeploymentConfig** objects or **Deployments**. The deployment systems ensure changes are propagated appropriately.



https://docs.openshift.com/container-platform/4.13/applications/deployments/what-deployments-are.html#what-deployments-are

RH OpenShift - deployment configuration options

- **Understanding Deployment and DeploymentConfig objects**
  - A **replication controller** ensures that a specified number of replicas of a pod are running at all times. If pods exit or are deleted, the replication controller acts to instantiate up to the defined number. Likewise, if there are more running than desired, it deletes as many as necessary to match the defined amount.
  - A **replication controller configuration** consists of:
    - The **number of replicas** desired, which can be adjusted at run time.
    - A **Pod definition** to use when creating a replicated pod.
    - A **selector** for identifying managed pods.
  - A **selector** is **a set of labels** assigned to the pods that are managed by the replication controller. These labels are included in the Pod definition that the replication controller instantiates. The replication controller uses the selector to determine how many instances of the pod are already running in order to adjust as needed.

  - **The replication controller does not perform auto-scaling based on load or traffic**, as it does not track either. Rather, this requires its replica count to be adjusted by an external auto-scaler.

https://docs.openshift.com/container-platform/4.13/applications/deployments/what-deployments-are.html#what-deployments-are

RH OpenShift – deployment configuration options

- **Understanding Deployment and DeploymentConfig objects**

  - Similar to a replication controller, a **ReplicaSet** is a native Kubernetes API object that ensures a specified number of pod replicas are running at any given time.
  - The **difference** between **a replica set** and **a replication controller** is:
    - a **replica set** supports set-based selector requirements whereas
    - a **replication controller** only supports equality-based selector requirements

Note:

- **Deployments** manage their replica sets automatically, provide declarative updates to pods, and do not have to manually manage the replica sets that they create.
- **Replica sets** can be used independently, but are **used by deployments** to orchestrate pod creation, deletion, and updates.

- **The ReplicaSet does not perform auto-scaling based on load or traffic**, as it does not track either. Rather, this requires its replica count to be adjusted by an external auto-scaler.

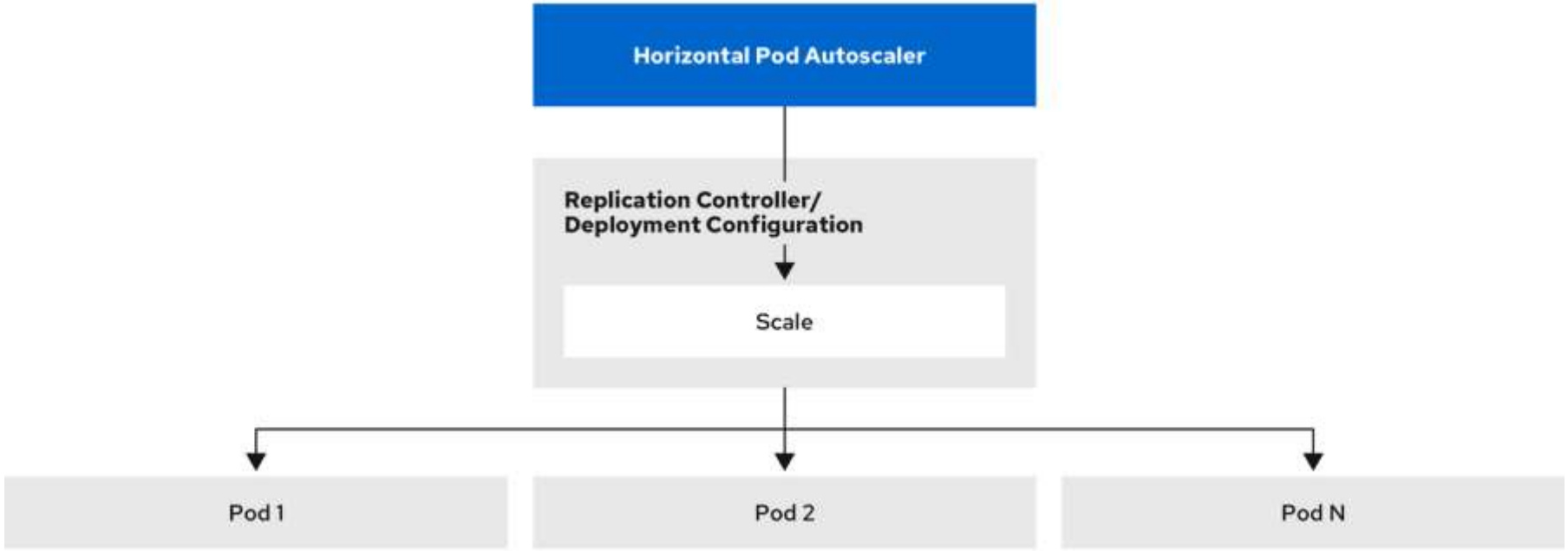
https://docs.openshift.com/container-platform/4.13/applications/deployments/what-deployments-are.html#what-deployments-are

RH OpenShift – **Horizontal** Pod Autoscaler (HPA)

- controls how OpenShift Container Platform should automatically increase or decrease the scale of a replication controller or deployment configuration, based on metrics collected from the pods that belong to that replication controller or deployment configuration.

You can create an HPA for any Deployment, DeploymentConfig, ReplicaSet, ReplicationController.

Note:
It is recommended to use a Deployment object or ReplicaSet object unless you need a specific feature or behavior provided by other objects.



RH OpenShift – **Horizontal** Pod Autoscaler (HPA)

- Can be configured using Web UI & CLI
- Can add scaling policies
- Can add stabilization window
- Notifies cluster autoscaler

- **Metrics for configuration**
- CPU Utilization based on percentage
- CPU Utilization based on specific value
- Memory utilization based on percentage
- Memory utilization based on specific value
- Scaling Policy for HPA based on CPU Utilization
- Scaling Policy for HPA based on Memory Utilization

To use horizontal pod autoscalers, the cluster administrator must have properly configured cluster metrics.

RH OpenShift – **Horizontal** Pod Autoscaler (HPA)

- During horizontal pod autoscaling, there might be a **rapid scaling of events** without a time gap. Configure the cool down period to prevent frequent replica fluctuations, by configuring the stabilizationWindowSeconds field.

- The stabilization window is used to restrict the fluctuation of replicas count when the metrics used for scaling keep fluctuating.

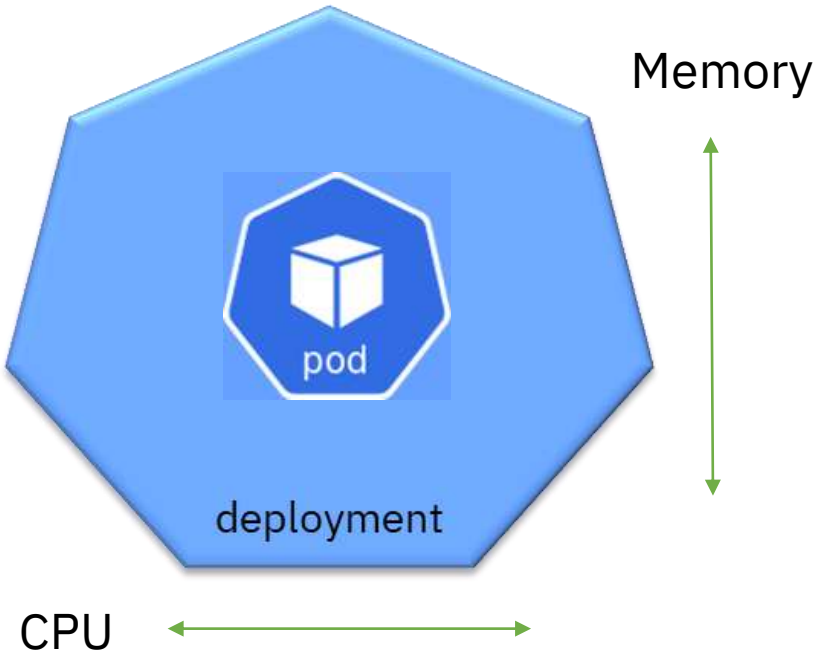
- The autoscaling algorithm uses this window to infer a previous desired state and avoid unwanted changes to workload scale.

More details docs:
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.13/html/nodes/working-with-pods#nodes-pods-vpa

https://cloud.redhat.com/blog/horizontal-pod-autoscaling-of-quarkus-application-based-on-memory-utilization

What is the **Vertical** Pod Autoscaler (VPA)

- **What is VPA**
- It scales cpu and memory resources for containers in pods and can update the resource limits and requests based on the metrics it learns
- Available as an operator
- Can be configured using CLI only
- Can be enabled for recommendation only



https://docs.openshift.com/container-platform/4.13/nodes/pods/nodes-pods-vertical-autoscaler.html

Using VPA Pod recommender

- You can use your own **recommender** to autoscale based on your own algorithms.
- If you do not specify an **alternative recommender**, OpenShift Container Platform uses the **default recommender**, which suggests CPU and memory requests based on historical usage.
- Because there is no universal recommendation policy that applies to all types of workloads, you might want to create and deploy different recommenders for specific workloads.

Note:

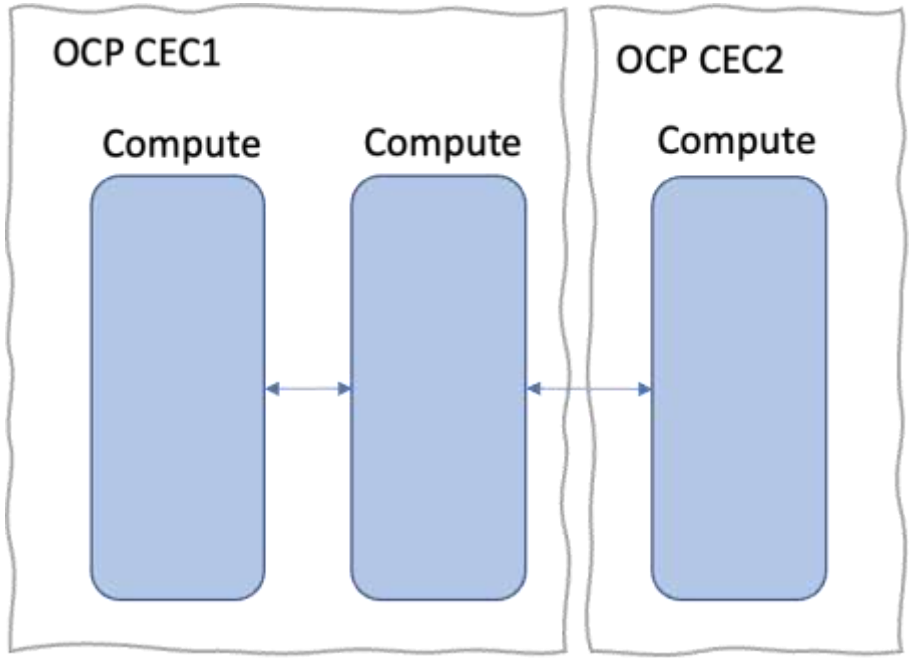
Using the default recommender with the usage behaviors might result in significant over-provisioning and Out of Memory (OOM) which can kill your applications.
The VPA recommender monitors resource utilization and computes target values.
Looks at the metric history, OOM events, and the VPA deployment spec and suggests fair requests.
The limits are raised/lowered based on the limits-requests proportion defined.

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.13/html/nodes/working-with-pods#nodes-pods-vertical-autoscaler-custom_nodes-pods-vertical-autoscaler

Scheduling Pods: How and where to schedule application pods How to use and optimize ?

Value: Scheduling options to improve workload placement and management

Application pods

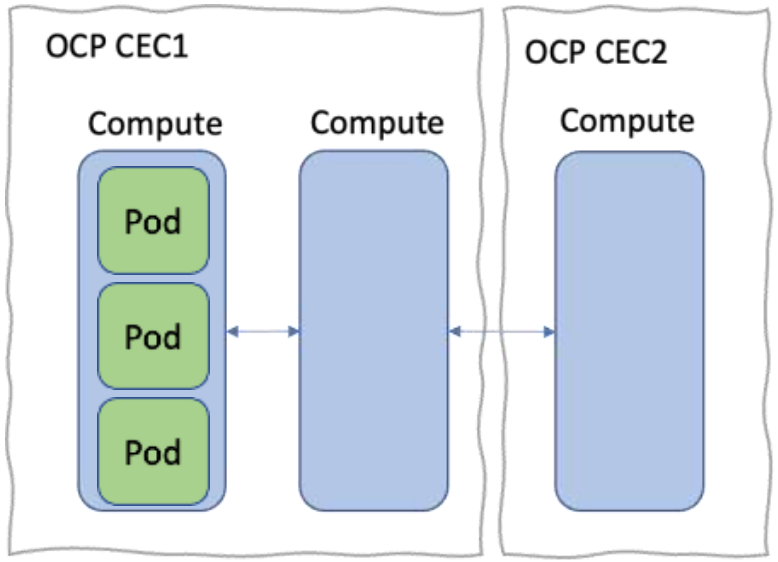


https://access.redhat.com/documentation/en-us/openshift_container_platform/4.13/html/nodes/working-with-pods#nodes-pods-priority

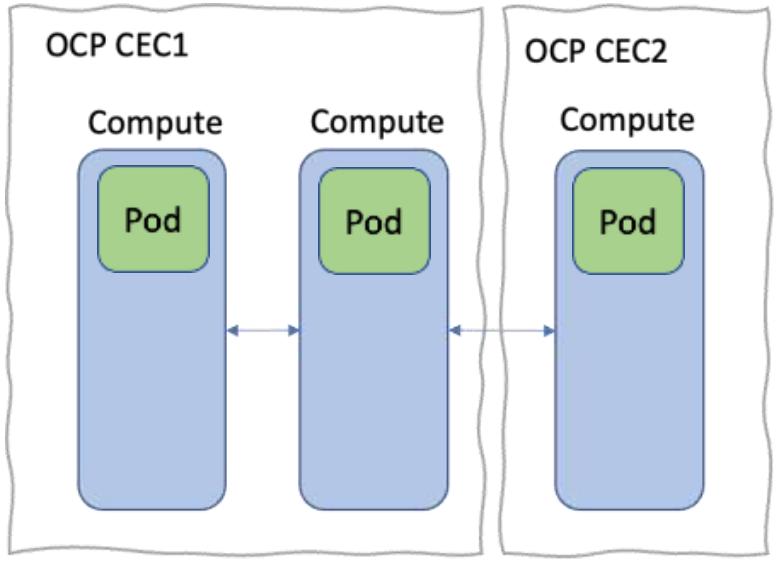
How and where to schedule the pods

Value: Scheduling options to improve workload placement and management

1.



2.



- Considers trade-off between **filling up** compute nodes vs. **spread pods** across the workers
- Take care when you use highly vCore over-provisioned setups
- Selected scheduler strategy must be in sync with your over-provisioning/setup strategy

Pod placement scheduler

Value: Scheduling options to improve workload placement and management

- OpenShift allows to **change how pods are placed on compute nodes using several placement profiles**.
- There are four **scheduling profiles** you can specify to control how pods are scheduled onto nodes
  1. **Default scheduler profile**
  2. **LowNodeUtilization scheduler profile**
  3. **HighNodeUtilization scheduler profile**
  4. **NoScoring scheduler profile**

➢ Can be used to decide how compute nodes are "filled up" with load regarding CPU/memory consumption of pods

- It is required to define ‘pod requirements’ for each pod deployed
- Scheduler uses defined pod requirements but does not consider real utilization

Pod placement scheduler: Profiles

1. **Default Scheduler**: Default Scheduler deploys pods in a three-step operation:
   - Filters the nodes by running each node through the list of filter functions called *predicates*, or *filters*.
   - Prioritizes the filtered list of nodes by passing each node through a series of *priority*, or *scoring*, functions that assign it a score between 0 - 10, with 0 indicating a bad fit and 10 indicating a good fit to host the pod.
   - **Selects the best fit node by sorting the nodes** by scores and selecting the node base on highest scores.
2. **LowNodeUtilization** Scheduler Profile
   - This profile attempts to **spread pods evenly across nodes to get low resource usage per node**.
3. **HighNodeUtilization** Scheduler Profile
   - This profile attempts to place as many pods as possible on to as few nodes as possible.
   - This **minimizes node count and has high resource usage per node.**
4. **NoScoring** Scheduler Profile
   - This is a low-latency profile that strives for the quickest scheduling cycle by disabling all score plug-ins.
   - This might sacrifice better scheduling decisions for faster ones

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.13/html/nodes/controlling-pod-placement-onto-nodes-scheduling

Conclusion: Scaling an application

1) define characteristics for pod(s) for the application

2) define the scalability characteristics (e.g. Replicaset, VPA, HPA)

3) define the placement characteristics (scheduling profiles)

4) decide for pod placement

Questions?

Wilhelm Mild

IBM Executive IT Architect

The Open Group Open Certified Distinguished Architect — IT Architecture Chief/Lead IT Architect

IBM

IBM Deutschland Research & Development GmbH
Schönaicher Strasse 220
71032 Böblingen, Germany

Office: +49 (0)7031-16-3796
wilhelm.mild@de.ibm.com

Notices and disclaimers