WHITE PAPER–NOVEMBER 2019

# Best Practices for Red Hat OpenShift on the VMware SDDC

vmware

## Table of Contents





vmware

## Executive Summary

In this paper, we outline the best practices gained from helping our customers to run Red Hat OpenShift on the VMware Software-Defined Data Center (SDDC). To aid in the readability of the document, we have created an example company called Example.com as a use case for applying these best practices.

In the past three years, Example.com had some debates between virtual machines or bare metal servers to power their ambitious Red Hat OpenShift (OCP) project. This debate was primarily triggered by the fact Example.com had two different groups, one on the application side advocating OpenShift on bare metal servers, and the other on the cloud infrastructure side advocating the use of the VMware SDDC.

Once the VMware technical specialist team was engaged with Example.com teams, many of the concerns that the application teams had were alleviated as they learnt about such features as live migration (vMotion), high-availability (vSphere HA), smart placement (DRS), and a refined ability to manage physical layer through vSphere virtualization.

Just like containers can create an encapsulation construct to abstract the lifecycle of an application process to make it easier to manage, in much the same way, vSphere virtualization can create a software construct to abstract physical compute resources to be able to manipulate them easily. This made the debate tip in favor of the VMware SDDC, and then VMware technical specialists moved to sizing, performance, and scalability, all of which drove Example.com to success. Just the ability to quickly reinstate OpenShift master node with vSphere HA, after a hardware failure, was a significant factor for the resiliency of the environment.

Example.com quickly realized that VMware bigger value proposition besides around operational capabilities was enabling them to run a vast diversity of workloads (cloud-native workloads and traditional workloads) in production, on a single platform such as the VMware SDDC. The VMware SDDC acts as a lightweight infrastructure software fabric for the datacenter across multiple areas: physical servers (vSphere), networking (NSX-T, NSX-SM, SD-WAN), storage (vSAN), multi-cloud (VCF and all six biggest public cloud providers run VMware), and containers (VMware Enterprise PKS, VMware Tanzu and Project Pacific).

The biggest lesson was that many of the assumptions against the VMware SDDC by the Example.com's application teams were due to the knowledge gap between them and the Example.com's cloud infrastructure group. Consequently, they worked together to form the Application Platforms Architecture group to foster collaboration and knowledge sharing across multiple disciplines, resulting in a new persona called Application Platform Architect (read more on Application Platform Architecture here) .

After deploying many production OCP clusters, with thousands of users, Example.com found that **relying on a VMware SDDC was one of the best choices that they have made for enterprise containers** for several business, functional, technical, and operational reasons as outlined below:

### Business and Functional Benefits

1-) The adoption of containers alone is a significant disruption to both application developers and operations teams as a new paradigm in how to develop, deploy, and operate mission-critical workloads. As such, it was critical that a trusted virtualization and automation layer would be adopted to minimize operational risks and complexities.

2-) A mature virtual infrastructure layer for container-based application platforms brings efficiencies such as reuse of compute resources and skills. Mostly, Example.com was able to leverage the existing vSphere team to deliver fully configured OpenShift platforms. This allowed for the development to entirely focus on application business logic and not be concerned with details of the infrastructure associated with OCP.

## Technical and Operational Benefits

1-) Kubernetes (or Red Hat OpenShift) is still unfamiliar territory to many. It is complex to install and to operate on a large scale with hundreds of users running thousands of Kubernetes pods. In a non-virtualized environment (OCP on bare metal) Example.com had lost two OCP master nodes on the same rack, and the process of rebuilding a master node from scratch was daunting, causing damage of credibility with the user base. They found that vSphere virtualization provided a protection layer from hardware malfunction with much faster recovery compared to kubernetes alone (*kubernetes gives automatic recovery, not high availability*. See vSphere High Availability and DRS and Implementing Availability Zones with OpenShift ).

2-) vSphere virtualization acts as another layer of resource management, opening many more possibilities for business to extract the most benefit by optimizing hardware resources. This paper explains Example.com's journey on leveraging virtualization to fine-tune the application node size, its performance and multiple reservation grades of OCP clusters (see vSphere Logical View ).

3-) It is wishful thinking to assume that all OCP users can master the knobs of kubernetes pod resource scheduling and prioritization to achieve the most optimized utilization of the compute resources on the kubernetes level. Virtualization is key to reduce the total cost of ownership for the kubernetes cluster, which allows for the onboarding more users on larger OCP clusters (See vSphere Utilization Metrics and Over-allocation ).

4-) Unfortunately, OCP (or any kubernetes) clusters are not "set-it-and-forget-it". Kubernetes clusters require Day 2 tasks such as monitoring, backup, patching, and migration. VMware vSphere streamlines these tasks by reducing error-prone procedures and avoids extra hardware costs (See Day 2 – Operational Procedures and Cluster Patching and Migrations ).

5-) OCP clusters running on a VMware SDDC integrate further strategically with enterprise ecosystem when compared to bare metal deployments. Two examples of these integrations are NSX-T for load balancing and micro-segmentation and VMware volumes and vSAN for container data persistency (see sections Security and Networking with NSX and vSphere as Cloud Provider to enable VMDK for Container Data Persistency).

6-) Kubernetes runs faster on the VMware SDDC if compared to bare metal. While it may sound a bit counter-intuitive, this is because of the optimized VMware vSphere virtualization scheduler. See Application Node Sizes.

## Overview of this Paper

**Red Hat OpenShift Container Platform (OCP)** is an enterprise kubernetes distribution targeted at corporations of multiple segments and sizes. OpenShift is a critical workload platform, and without a virtualization layer, users and workloads can suffer every time there is a hardware event that will reduce OCP compute capacity and possibly lead to OCP cluster outages.

Over the last 20 years, VMware has mastered running some of the most critical workloads in large enterprises [1], ensuring that a hardware failure do not impact operations. VMware has firmly established itself in the enterprise datacenter, first with its vSphere compute virtualization technology and then extending to vSAN storage virtualization and NSX network virtualization, in addition to robust management with vRealize, which are the key components of a VMware SDDC. Today, VMware helps customers build modern applications, ensures customers run and manage kubernetes.

This paper outlines the rationale of why corporations such as Example.com can benefit from integrating a VMware SDDC and OpenShift 3.11 and 4.1 [2] to support an exponential adoption of container workloads with clusters scaled to run 10,000+ kubernetes pods with real-world production environments.

Components such as VMware HA and DRS are detailed in this paper for the optimal configuration of OpenShift and its resiliency.

Other sections of this paper detail the role of VMware on Day 2 operational procedures on OpenShift and lessons learned on deploying OpenShift on the enterprise-level scale. This paper is a complement of Reference Architecture "Deploying and Managing OpenShift 3.11 on VMware".

**OpenShift 4.1:** OpenShift 4.1 was released in June/2019 with substantial changes compared to OpenShift 3.11. Since this paper is a best practice of vSphere for OpenShift, the content applies for OpenShift 4 unless noted. For more information on OCP4, please see Appendix A: OpenShift 4 .

The adoption of OpenShift has impacted multiple groups in Example.com in the way they develop, deploy, and manage applications. Multiple personas have been part of this journey, but for simplicity, this paper focuses on the following actors:

- **Application Platform Architect**: A multi-disciplinary professional across infrastructure (VMware), PaaS (Platform as a Service), cloud, software development, DevOps, and Site Reliability Engineer (SRE) practices. It is a technical lead responsible for designing and architecting OpenShift clusters and practices. Bridges the teams of Application Developers and Operations.
- **Application Developers**: Individuals (and groups) responsible for developing and managing the lifecycle of Example.com's applications. Usually in communication with business units to code their business logic under business priority and to provide time-to-market. They are the OpenShift end users.
- **Operations**: The team responsible for managing and supporting the IaaS (VMware) and PaaS (OpenShift). This paper refers to them as a single group, but in reality, Operations can be multiple teams (one for vSphere, one for OCP clusters, one per business unit, and so on).

[1] https://blogs.vmware.com/vsphere/tag/business-critical-apps .

[2] OpenShift 4.1 was released in June/2019 with significant changes when compared to 3.11. Most of this paper is applicable to OCP 4.1 unless noted.

The examples and considerations in this document provide guidance only and do not represent strict design requirements, as varying application requirements might result in many valid configuration possibilities.

## Summary of OCP 3.11 Components

Before explaining the best practices of vSphere with Red Hat OpenShift, this section reviews at high-level the components of the OCP 3.11 cluster. For additional information on core kubernetes and the mechanics of how an OCP works, refer to kubernetes components and OCP. See Figure 1 for the very high-level diagram of the type of nodes:

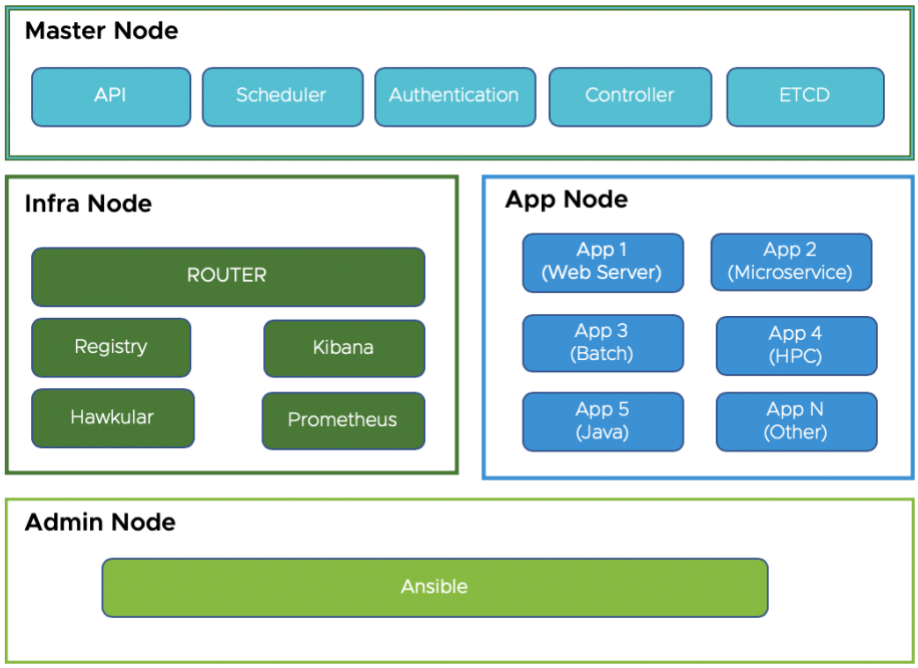


*Figure 1: Types of Node on an OCP Cluster*

**Master Nodes:** Hosts that contain the kubernetes control plane components such as the master-api server (API/Authentication) and master-controller (Controller and Scheduler). Additionally, the same nodes run the etcd (DataStore). The masters are only accessible over API, and they are also known as "control plane nodes". It is recommended to have at least *three master nodes* for HA purposes. The etcd's Raft distributed consensus algorithm requires that the number of OCP master nodes are in odd numbers (1,3,5, and so on.) [3] . VMware vSphere makes it possible to enforce the full redundancy of master nodes with as little as two bare metal servers.

**Infrastructure Nodes:** Hosts that contain the router pods, responsible for managing and redirecting incoming application traffic from the users to the application nodes. Infrastructure nodes perform this task through route objects, and these nodes are known to provide the Routing Layer. It is recommended to deploy *at least three nodes*

[3] https://raft.github.io/ .

to avoid application traffic outages and provide traffic distribution between them. vSphere HA restarts VMs in case of hardware failures and consequently protects users from application time-outs or complete disruption of the application traffic. Optionally, infrastructure nodes can run other OCP ancillary components such as Docker Registry, ElasticSearch, Hawkular, and Prometheus. Technically, they are like any Application Node: the difference is the embedded automation on OpenShift 3.11 to configure native OCP pods to run only on these nodes.

**Application Nodes:** Hosts that run the actual application containers. The applications running on kubernetes pods are accessible by service objects with application nodes providing the Service Layer. The size and number of nodes will depend on the application footprint.

**Admin Node (a.k.a. Bastion Host or Ansible Host):** Host that manages the entire cluster by Ansible. Usually one node.

This paper addresses the number and size of these nodes in the next sections.

**OpenShift 4.1:** OpenShift 4.1 uses the term "Control Plane" nodes and "Worker" nodes interchangeably with "Master" Nodes and "Application" Nodes, respectively. On OCP4 clusters, we recommend assigning some "Worker" nodes to act as "Infrastructure" Nodes. Please refer to the Appendix on OpenShift4 for significant similarities and differences between OpenShift 3.11 and OpenShift 4.

## OCP Conceptual View

Figure 2 has a conceptual view of a scaled, secured and highly available OCP cluster running on vSphere.

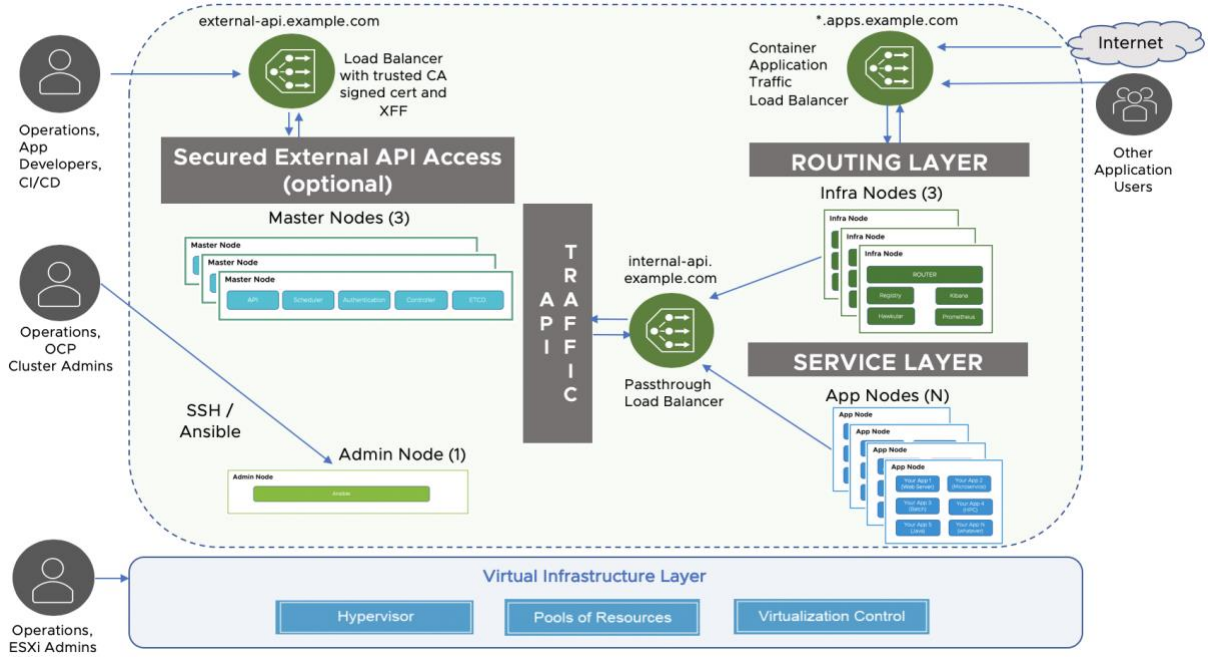


*Figure 2: Conceptual view of OpenShift on vSphere*

For traffic management, security and micro-segmentation, Example.com decided on having three load balancers per OCP cluster:

Two separate load balancers (LB) for OCP API traffic. One load balancer dedicated to external access (users, CI/CD, operations) using a DNS alias such as ***external-api.example.com***. The same load balancer has the Example.com's signed cert for security. There is a second load balancer with SSL passthrough for internal API traffic (exclusively between masters to all other OCP nodes) using the DNS alias ***internal-api.example.com***. The segregation of the two load balancers adds the ability for extra security (API signed certs, micro-segmentation, and audit).

One dedicated load balancer for application traffic with a wild card DNS pointing to it (i.e., ****.apps.example.com***)

For load balancing configuration, Example.com opted to have all be performed by NSX Edge load balancers to achieve network performance and to reduce the operational overhead of configuring load balancer software and Virtual IPs (See Appendix A for NSX-V/NSX-T integration). The same three-load balancer design and functionality can be achieved with any load balancer of choice, such as ha-proxy.

From a networking perspective, Red Hat recommends placing the master nodes on the same network or on "an uncongested, low latency LAN communication" to protect etcd health from unnecessary leader elections and failed heartbeats. The master nodes and application nodes must not be on the same network or under the same vSphere cluster. In the diagram, the entire OCP cluster runs entirely on the Virtual Infrastructure layer (the hypervisor, pool of resources, and vSphere virtualization control).

For extra security, some corporations can opt to run etcd on separate non-master nodes, but this design is out of the scope for this paper.

## vSphere Logical View

There are multiple possible configurations on how to run OCP clusters in vSphere. Some of the pertinent factors are OCP cluster size, vSphere HA configuration, reservation grade of OCP cluster, and so on.

The reservation grade of OCP Clusters on vSphere is based on the amount of resources (Memory and vCPUs) guaranteed using Resource Allocation Reservation to the Application Nodes VMs powering each OCP cluster.

| Reservation Grade | OCP Application Node VMs Resource Allocation Reservation |
|---|---|
| Platinum | 100% Reservation of the size of the VM. |
| Gold | 50% Reservation of the size of the VM. |
| Silver | Reservation set to 0 |

*Table 1: Reservation Grade of OpenShift Clusters based on Resource Allocation Reservation*

The initial adoption of OpenShift (or any kubernetes) usually comes with a certain level of uncertainty with little requirements and a limited budget from the business. It is best to leverage the existing internal cloud for initial OCP deployments to reduce initial investments on the OCP platform. On "Day 1" Example.com did not need a large amount of computing for their kubernetes environments. However, the adoption of kubernetes can be remarkably rapid, requiring the scalability of the OCP clusters to absorb the demand. This scenario is a testament to why a VMware SDDC is strategic and relevant in any environment: with multiple businesses competing for compute resources to accommodate the growing user base and the number of OCP clusters, resource prioritization comes with various reservation grades of OCP cluster and respective chargebacks.

With this in mind, this paper documents two possible scenarios of OCP clusters running on vSphere cluster:

- Scenario #1: Shared vSphere cluster to Support Multiple OCP Clusters with Different Reservation Grades ("Platinum", "Gold", "Silver")
- Scenario #2: Dedicated vSphere cluster to Support Single "Platinum" OCP Cluster.

**ATTENTION:** The master, infrastructure, and admin VMs must be always set to 100% Reservation regardless the type of reservation grade of OCP cluster.

For illustration purposes, we assume the following on both scenarios:

- Specs of vSphere host with 1 TB RAM and dual-socket, 28-core per socket. A total of 56 physical CPUs with hyperthreading enabled yielding 112 logical CPUs per host.
- vSphere HA configured with Cluster Resource Percentage as Admission Control leveraging the automatic calculation of "Host failures cluster tolerates" to "1" . More about Admission Control on section Admission Control Algorithms.
- OCP Master/Infra/Admin node VMs running on a separate Management vSphere cluster.

NOTE: VMware Validated Design (VVD) practices of a Standard SDDC recommends separating VMs on different vSphere clusters according to the type of processing nature: computing, management, and networking. Following the VVD, OCP application nodes VMs should be on vSphere ***"Compute Pod"*** clusters, OCP management nodes (master, infrastructure, and admin) VMs should be on vSphere ***"Management Pod"*** cluster and Load Balancers should be on vSphere ***"Edge Pod"*** cluster.

In both scenarios, (Scenario #1: Shared vSphere cluster to Support Multiple OCP Clusters and Scenario #2: Dedicated vSphere cluster to Support Single OCP Cluster), this paper uses the recommended design of separating different types of VMs (compute and management VMs) to run on respective vSphere clusters (compute and management clusters). For more information on VVD, see "Virtual Infrastructure Layer in Standard SDDC" section in VMware Validated Design Reference Architecture 5.1.

### Scenario #1: Shared vSphere cluster to Support Multiple OCP Clusters

Figure 3 depicts the logical view of a shared vSphere cluster with 32 vSphere hosts powering multiple OCP clusters and non-OCP VMs. This shared vSphere cluster can accommodate multiple OCP clusters with mixed resource reservation grades with or without Memory Overcommitment and CPU overcommitment. The Resource Allocation Reservation guarantees resources for "Platinum" and "Gold" OCP Clusters.

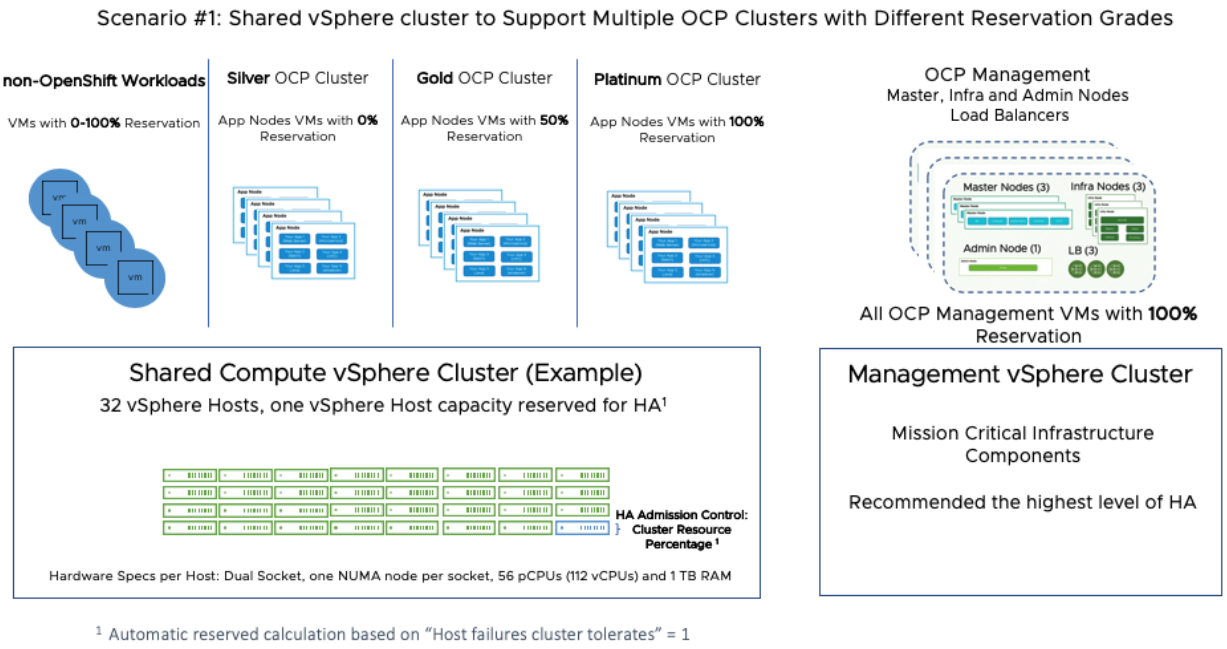


*Figure 3: Example of a shared vSphere Cluster supporting multiple OCP Clusters with Different Reservation Grades*

Application Platform Architects must coordinate with Operations (vSphere cluster admins) on the correct configuration of OCP application node VMs to honor the OCP reservation grades. The shared vSphere cluster configuration is very common in the initial phases of the adoption of OpenShift to transition to the highest critical workloads. The OCP management nodes (master, infrastructure, and admin) must always have full reservations with vSphere HA (see the section vSphere High Availability and DRS ). Optionally, each OCP cluster can have a Resource Pool.

### Scenario #2: Dedicated vSphere cluster to Support Single OCP Cluster

Figure 4 provides an example of a single vSphere cluster with 16 vSphere hosts, with one host dedicated to HA. Each host has 1 TB RAM and 56 physical CPUs with hyperthreading enabled yielding 112 logical CPUs per host. The net capacity of 1680 logical CPUs and 15 TB powers application node VMs on a single Platinum OCP cluster.

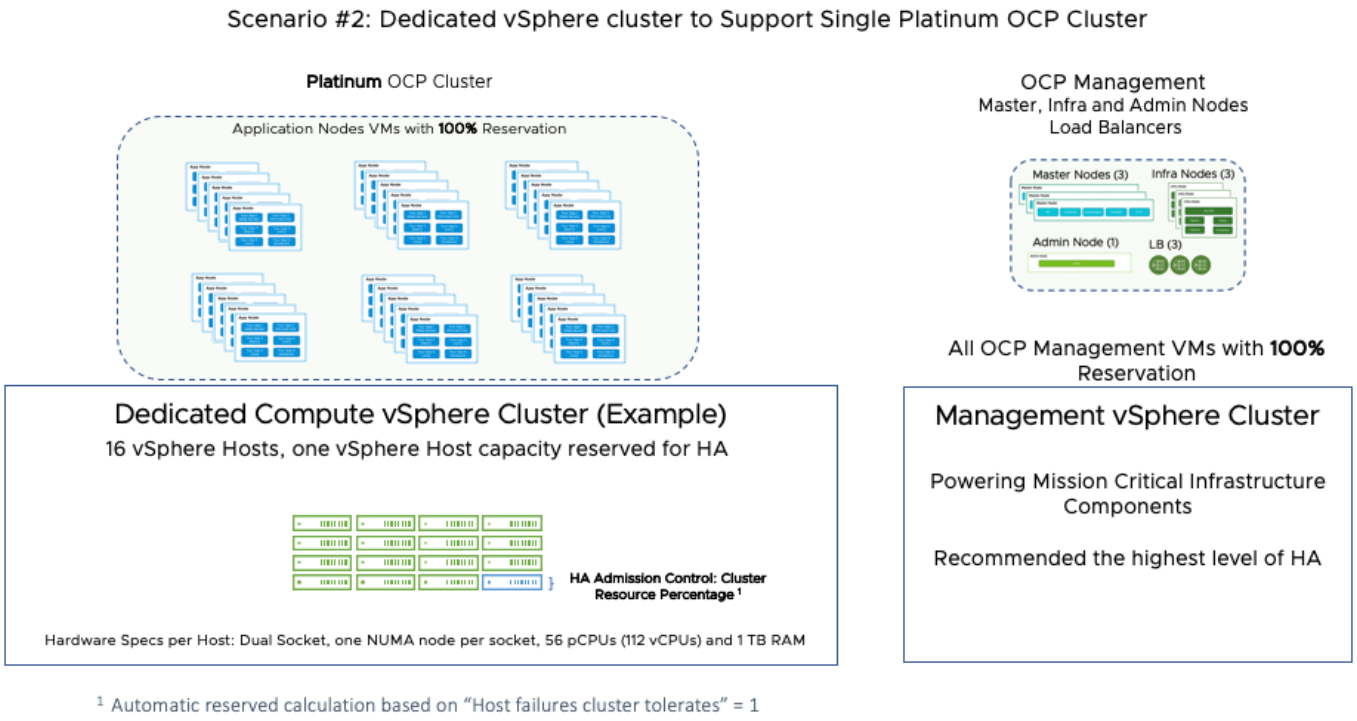


*Figure 4: An example of a vSphere Cluster to Support a Single Platinum OCP Cluster*

All application nodes VMs must be created with full reservation with enough capacity to be powered up all the time to assure OCP Platinum reservation grade.

NOTE: Always take into consideration some capacity headroom for the vSphere cluster to avoid 100% resource allocation. Later in this paper, there is a section on Scenario #2 as an example to calculate sizes of application nodes VMs by reserving 15% of the compute resources as headroom for vSphere to operate (see Detailed Example of Calculation of Application Node Size section). The actual vSphere overhead is substantially less, but for stringent production level measures, Example.com used 15% as ample headroom to absorb workload spikes in normal daily operations.

### Scenario Comparison and Recommendation

Scenario #2 (Dedicated vSphere cluster) can be seen as a natural evolution from Scenario #1 (Shared vSphere cluster), and Scenario #2 works best to support mission-critical workloads on OCP clusters.

Please see Table 2 below comparing the two scenarios:

| | Shared vSphere Cluster for multiple workloads (OCP and non-OCP) (Scenario #1) | Dedicated vSphere Cluster for a Single OCP Cluster (Scenario #2) |
|---|---|---|
| Allocation of Capacity | Not fully tailored for the OCP clusters, application node VMs sizes depend on the availability of shared vSphere cluster. | Tailored for a single OCP cluster with sizes of application node VMs dependent on container workloads. |
| Chargeback and License Costs | More complex to calculate. Based on the number of VMs running OCP workload versus non-OCP workload. Might require integration with vCenter. | Simpler to calculate. Entire vSphere cluster allocated for the same software stack with the same license costs. |
| Security | Less secure with potentially mixed workloads that involve development, UAT (User and Acceptance Testing), and production. NSX Micro-segmentation is needed for complete control. | More secure and less complicated to setup using Macro-Segmentation since the isolation of workloads of development, UAT (User and Acceptance Testing), and production. |
| Change Control | Less strict with different workloads and reservation grade. | Stricter with well-defined reservation grade. |
| VM Performance for High-Performance workloads | Non-guaranteed and non-deterministic for Silver and Gold reservation grades. | Guaranteed and deterministic. |
| Total Cost of Ownership (TCO) | Possibly lower. The cost of the vSphere cluster is shared with multiple users and businesses, with more tolerance of oversubscription of resources. | Possibly higher. The cost of the vSphere cluster is likely allocated to single-tenant (cost center or business) with a possible minimum tolerance of oversubscription of resources. |

*Table 2: Comparison of the Shared vSphere Cluster versus Dedicated vSphere Cluster*

**FLEXIBILITY:** Example.com used scenario #2 for production clusters and scenario #1 for development and UAT clusters.

## Optimized Resource Management

When Example.com saw a proliferation of OCP clusters to accommodate the rapid adoption from multiple businesses, it was paramount for the Application Platform Architect to determine standard OCP management nodes (master, infrastructure and admin) sizes while customizing OCP application node sizing to maintain optimal utilization of hardware and consequently lower TCO of OCP clusters.

vSphere provides fine granular control of the underlying compute resources for each OCP component with node re-sizing as per workload demand without causing any user impact (combining VM live migration and flexibility of increase/decrease nodes, clusters, and so on).

Regardless of the reservation grade of the OCP cluster, it was determined the OCP management nodes (master, infrastructure, and admin) VMs to have a standard size to sufficiently accommodate 5,000 to 10,000 kubernetes pods (scalable on Day 2, if necessary). Application node VM sizing required more elaboration (it is case-by-case, depending on application container workload), and the Application Platform Architects determined the calculation of Application Node sizes mostly based on CPU architecture, as you can see on the following sections.

## Master, Infrastructure, and Admin Nodes Sizes

Red Hat recommendations on the minimum sizing for master, infrastructure, and admin nodes can be found here.

Figure 5 shows the specs of nodes that Example.com fine-tuned to support 5,000 to 10,000 kubernetes pods per OCP cluster:

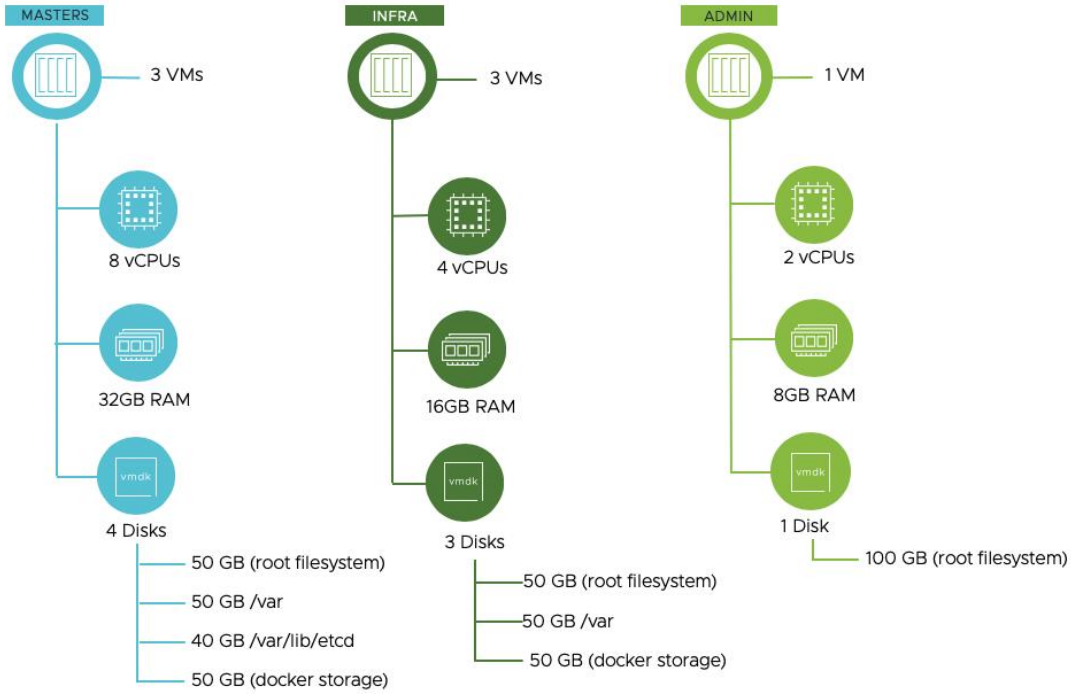


*Figure 5: OCP Master/Infra/Admin Node Sizing Recommendations for a 5,000-10,000 pod cluster*

SCALABILITY: Application Platform Architects can always leverage vSphere on Day 2 and further scale the sizes of the Master/Infra/Admin nodes to support growing OCP clusters.

## Application Node Sizes

Application node sizing calculation – *both horizontally (number of nodes) and vertically (nodes sizing)* - requires more attention than management nodes since their sizing is done case-by-case for each enterprise.

It is essential to understand the type of application footprints on the OpenShift cluster. Applications with homogenous requirements, that is, with the same kind of middleware, consumption, and nature (jobs, microservice, and so on), can drive the sizing of application nodes tailored to a desirable number of kubernetes pods per application node. If the footprint of the application is very heterogeneous, for example, from small microservices to large memory cache databases, it is better to have large application nodes and have the kubernetes scheduler to manage the different size of kubernetes pods.

For the maximum size of application nodes, take into consideration the CPU architecture – such as Non-Uniform Memory Architecture (NUMA).

Figure 6 illustrates the minimum and maximum recommended sizes of an application node.

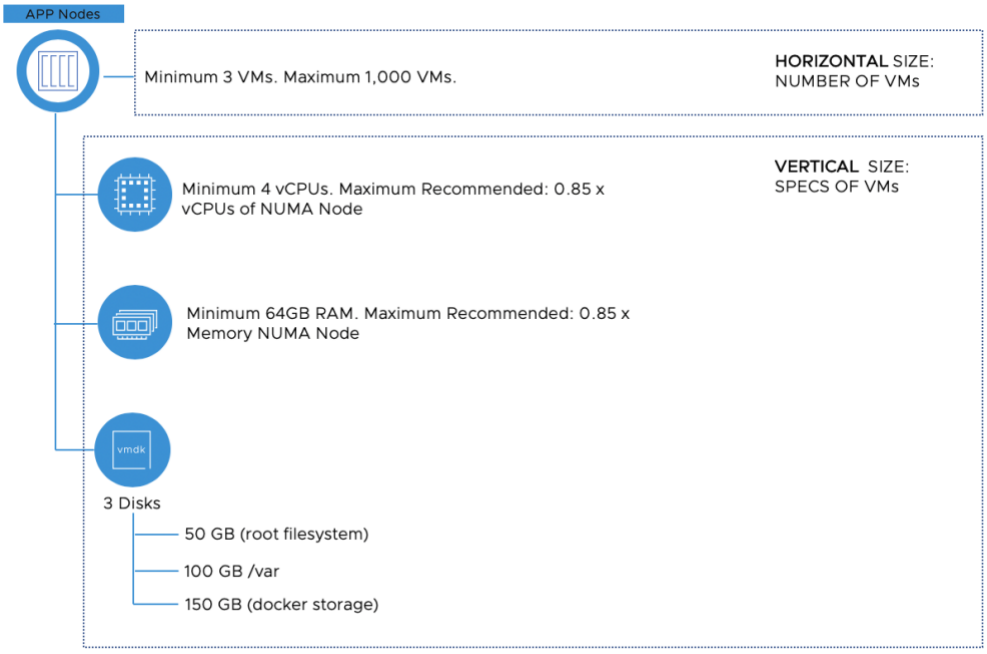


*Figure 6: Minimum and Maximum specs of OCP Application Node.*

**PERFORMANCE:** When compared to generic Linux running on bare metal, vSphere does a better job at scheduling the Application Node VMs and their pods to run on the right CPUs. vSphere provides better localization, and it dramatically reduces the number of remote memory accesses. Kubernetes pods running memory-bound workloads show superior performance when deployed on VMware SDDC, as seen on the performance testing with Project Pacific.

### Simple Method of Sizing Application Nodes

There are multiple ways to calculate the optimal sizing of application node VMs.

A simple method of calculation is opting for the vertical size of the application node VM to get the memory size and vCPUs of the NUMA node with the headroom for vSphere.

You can arrive at values for the memory and vCPU by applying the following formula [4]:

| | |
|---|---|
| Memory Size | **(0.85 * Total RAM on Host) / Number of Sockets** |
| vCPU (*) | **(0.85 * Total Logical Processors on Host) / Number of Sockets** |

(*) Hyperthreaded vCPUs (please see Appendix section on Hyperthreading)

**HEADROOM:** The factor of 0.85 is a practical approximation for the reserved capacity for ample vSphere headroom of 15%, tested for mission-critical applications with deterministic performance such as trading systems. [5] (*) If users of Platinum OCP clusters report vCPU slowness (which will depend on application footprint), check utilization vSphere metrics, and if necessary, revisit the formula to increase the vSphere overhead.

The horizontal size of Application Node VMs is the total number of NUMA nodes of the vSphere cluster discounting capacity of the reserved failover.

For illustration purposes, Example.com's calculated the size of application nodes based on Scenario #2: Dedicated vSphere cluster to Support Single OCP Cluster would be:

- **Vertical Size.** Example.com's bare metal server is an Intel dual-socket with one NUMA node per socket. Each socket had 28 physical CPUs (or 56 logical CPUs with hyper-threading) and 512 GB. Calculation is (112 logical processors | 1 TB RAM) *0.85 / 2 = **48 vCPUs** and **436 GB RAM.**
- **Horizontal Size.** Example.com has 16 vSphere Hosts per vSphere cluster. Discounting one vSphere host for HA ("Host failures cluster tolerates" to "1"), there is the net capacity of 15 vSphere hosts. Calculation is 15 * 2 NUMA nodes per Host = 30 NUMA nodes = **30 Application Nodes VMs.**

This simple method naturally drives to fairly large application node VMs in terms of vertical size. Application Platform Architects can make initial assumptions on Day 1 and check USE metrics (Utilization, Saturation, Errors) metrics of Operating System, vSphere, and OCP layers to fine-tune the sizing of application nodes on Day 2, and reduce the vertical size while increasing the horizontal size.

See section " Detailed Example of Calculation of Application Node Size " which other factors such as kubernetes pod footprint are taken into consideration in the sizing of the application node VMs.

[4] "How to Design and Tune Virtualized Trading Platforms", Minute 26 - Emad Benjamin, VMworld 2016 .

## Over-allocation on OCP for Non-Production Workloads

Some businesses opt for over-committing compute capacity on non-production OCP clusters (such as development clusters) to schedule more pods and onboard users using the same compute capacity.

Commonly, users ask which is the best way of computing capacity over-allocation for OpenShift clusters: vSphere over-committing (VM level, that is, larger application node VMs on the same physical infrastructure) versus kubernetes scheduling over-allocation.

vSphere can give the options to corporations to perform over-allocation at a hypervisor layer on application nodes, transparently, and in conjunction with kubernetes scheduler. However, teams need to exercise caution on this combination, mainly when OCP admins and VMware admins belong to different groups and are unaware of each layer's utilization. Use both layers only under close supervision and monitoring. For more information about such monitoring, see the section vSphere Utilization Metrics and Over-allocation.

Start with kubernetes as the first layer of over-allocation and leverage OCP metrics (executing the command "**oc describe nodes**") to see specific metrics of kubernetes allocation (that is, the requests and limits metrics). Only after analysis, determine whether the second level of over-allocation through hypervisor is required for the fine-tuning of the utilization efficiency.

**OVER-ALLOCATION:** Example.com chooses to over-commit compute resources (without over subscribing available physical compute resources) primarily on the OCP (kubernetes) layer because the business users (application developers and managers) have control and permissions on OCP scheduler (such as default requests and limits) rather than on vSphere cluster.

See the section vSphere Utilization Metrics for an example of how Example.com used both kubernetes and vSphere metrics to decide for over-allocation on both layers.

## vSphere High Availability and DRS

vSphere HA and DRS are paramount to give resource optimization and availability to OpenShift clusters.

### vSphere High Availability (HA)

Kubernetes alone cannot completely handle the resiliency and recovery needs of all mission-critical cloud-native applications. **Kubernetes gives automatic recovery and not high availability** [5].

vSphere HA offers these additional benefits for kubernetes clusters and cloud-native applications:

Protection and quick recovery for the master VMs, especially to etcd health.

In case of an application node failure, vSphere HA acts faster (VMs are restarted before kubernetes pods are evacuated) and more transparent (kubernetes pods continue to run on the same application node after the restart, avoid re-mounting and connectivity on a different node) when compared with the default OCP pod timeout mechanism (*5 minutes*). Stateful applications with persistent mounted volumes (VMFS, NFS) can take even longer (up *to 10 minutes*) [5] to recover while awaiting the re-attachment of volumes.

**AVOID OUTAGES:** It is paramount to configure the restart behavior priority of OCP nodes via Restart Priority. In the event of complete (planned or unplanned) maintenance of vSphere cluster, admin and master nodes should be preferably powered on first, followed by the infrastructure nodes, and finally the all application node VMs. Example.com has learned from experience that when infrastructure node VMs start slightly before the master node VMs, they result in an inconsistent OCP cluster state. For the shutdown sequence, it is recommended that app nodes shut down first, followed by infrastructure nodes, master nodes, and finally, the admin node.

Additionally, vSphere HA utilizes heavily tested heart beating and partitioning detection mechanisms to monitor hardware, network, VM, and even forecasting failures with vSphere 6.7 proactive HA (such as one out of two power supplies had failed on the host and then proactively vMotion OCP nodes to a healthy metal server).

Lastly, in case there are Platinum, Gold, and Silver OCP clusters on the same vSphere cluster, VMs of Platinum OCP clusters should have higher restart priority than VMs of Gold and Silver OCP clusters. This way, the Platinum application nodes will be powered up first to guarantee more uptime and resource priority.

[5] "The Magic of Kubernetes Self-Healing Capabilities" (minute 26), Saad Ali, Kubecon 2019 .

### Admission Control Algorithms

Quoting from vSphere Availability Guide, "vSphere HA uses admission control to ensure that sufficient resources are reserved for the virtual machine."

There are three Admission Control algorithms: Cluster Resource Percentage, Slot Size, and Failover Hosts.

Regardless of which algorithm used, HA Admission Control provides flexibility and resiliency for OCP clusters when compared deployments on pure bare metal.

The most used HA Admission Control algorithm is per percentage basis. In this case, on vSphere 6.5 and later versions, it is recommended to leverage the automatic calculation of percentage on using "Host failures cluster tolerates" to a number of hosts (our scenarios set it to "1").

We recommend the book "VMware vSphere 6.7 Clustering Deep Dive" for more details on Admission Control. Please note there is no Admission Control for storage: all storage capacity tasks must be done manually as of this writing.

### vSphere DRS

vSphere DRS is crucial for a balanced distribution of computing resources and initial placement of OCP node VMs.

Example.com has decided to enable DRS with full automation and migration threshold set to "3" (across all reservation grades). With the master nodes and infrastructure nodes, this automation level should be overridden (set as "Partially automated") to avoid unnecessary migrations [6].

VM-VM anti-affinity rule must exist on the master and infrastructure nodes so that a single vSphere host failure does not impact the OCP cluster. Such a rule ensures that two master VMs are never on the same vSphere host.

### Implementing Availability Zones with OpenShift

In case the vSphere cluster runs across multiple availability zones (such as across racks or different metropolitan data centers using vMSC [7]), OCP must be integrated with DRS groups for OCP nodes zone segregation and application container scheduling.

**AVAILABILITY ZONES:** Multiple availability zones with DRS groups (also known as Cluster Rules) is an optional increment of Scenario #2 (Dedicated vSphere Cluster) to provide an extra layer of protection for the business continuity.

With multiple availability zones, vSphere hosts can belong to different DRS host groups (also known as Cluster Rules) [8], one host group per zone. Figure 7 depicts two zones (zone-a and zone-b) and two respective DRS host groups (with one HA failover host each), one per zone.

6 More details on Understanding vSphere DRS Performance Guide, DRS Aggression Levels section .

7 vSphere Metro Storage Cluster .

8 See vSphere Resource Management guide, section DRS, "Using DRS Affinity Rules" .

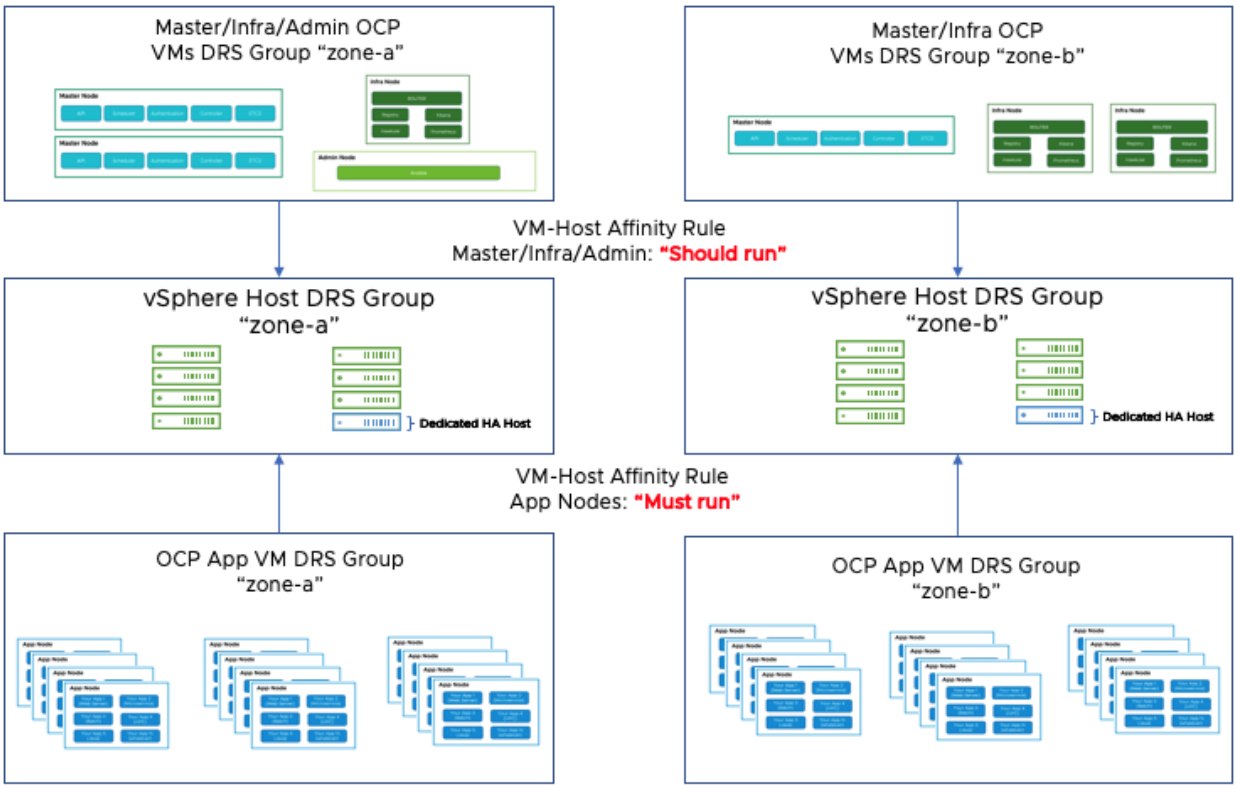


*Figure 7: DRS Configuration of OCP Nodes VMs Across Two Zones*

Remember that the OCP master, infrastructure, and admin node VMs are in odd numbers: master (3 VMs), infrastructure (3 VMs), and admin (1 VM).

In this example with two VM DRS Groups, the OCP master, infrastructure and admin must be divided asymmetrically between the two VMs DRS Groups:

- "zone-a" VMs DRS Group contains two OCP master node VMs, one OCP infrastructure node VM and the single OCP admin node VM.
- "zone-b" VMs DRS Group contains one OCP master node VM and two OCP infrastructure nodes VM.

A VM-Host Affinity Rule ***"Should Run"*** causes the master, infrastructure, and admin node VMs to run preferably in separate zones, but they may run in the same zone if necessary, in case another zone is unavailable.

In this example, OCP application node VMs must be created in even number and divided equally into two VM DRS Groups, one VM DRS group per zone. The VM-Host Affinity Rule ***"Must Run"*** enforces application node VMs always to be confined in each DRS Host Group despite any event (such as zone failure). The rule ensures that application node VMs from one DRS group are never be powered on another DRS group, hence providing isolation and protection to each availability zones.

Note the difference of behavior between "Should Run" (master, infrastructure, and admin node VMs) and "Must Run" (application node VMs). The latter will never have Application Node VMs being powered on a different zone.

On kubernetes v1.12 and later [9], the kubernetes scheduler can be integrated with vSphere and be aware of different DRS groups. This way, kubernetes scheduler can ensure to run copies of the same kubernetes pods across DRS groups and avoid a single point of failure.

OpenShift 3.11, which is based on kubernetes 1.11, does not have this integration "out-of-the-box" however, it can be done using manual steps leveraging the DRS host groups, kubernetes labels, and kubernetes pod anti-affinity rule.

Executing OCP command **`oc label node <application_node_VM_name> failure-domain.beta.kubernetes.io/zone=<Host DRS group>`,** the OCP cluster admin labels the application nodes with their respective Host DRS groups.

Example:

```
failure-domain.beta.kubernetes.io/zone=zone-a
```

**USER EDUCATION:** Application Platform Architects must publish documentation to educate the application developers to avoid singletons and leverage OCP scheduling awareness of application nodes labels using OpenShift 3.11 pod anti-affinity preferred rule.

[9] See Kubernetes github #64021 and Kubernetes 1.12 Change Log. OpenShift4 runs Kubernetes 1.13, and it supports vSphere tags with vSphere as cloud provider.

## Day 2 – Operational Procedures

Example.com, like any large corporation, has understood that there are multiple operational challenges in productizing and deploying OpenShift on a large scale. VMware vSphere alleviates some of the operational challenges of day 2, reducing the day-by-day OCP complexities and resulting in a more resilient enterprise deployment.

### vSphere Utilization Metrics and Over-allocation

Example.com has also realized that high utilization on kubernetes schedule resources does not translate to high utilization on a vSphere cluster.

vSphere cluster utilization metrics are critical in validating the effectiveness of OCP cluster utilization and driving higher utilization of hardware resources. With DRS and vSphere metrics, Example.com has decided for extra overallocation of vCPUs on non-production clusters (Silver reservation grade cluster).

*Table 3* describes this journey. Each row is a logical day with the sequence of events, and the arrows indicate relevant changes from the previous state.

The success criteria of this journey are the ability of onboarding more OpenShift users (and consequently scheduling more kubernetes pods) on the OCP development cluster (Silver reservation grade) without expenditure (that is, expanding the cluster with more hardware).

| Day | Events and Observations on non-Production OCP development Cluster (Silver reservation grade) | **Observed Metric and Success Criteria** Number of users on-boarded on the OCP development cluster (More is Better) | **Configured Parameter** Kubernetes Scheduler: CPU Commitment [10] (Overcommitment is more than 100%) | **Observed Metric** Kubernetes Scheduler: Utilization (CPU Requested) | **Configured Parameter** vSphere: vCPUs allocation on underlining hardware (Overallocation is more than 100%) | **Observed Metric** vSphere: CPU Cluster Utilization [11] |
|---|---|---|---|---|---|---|
| 1 | OCP development cluster opened for business. Rapid user adoption. | 100 | 100% | 20% | 85% | 5% |
| 2 | Users report they cannot schedule more pods. Kubernetes scheduler constrained on CPUs. | **250 ⇑** | 100% | **100% ⇑** Exhaustion | 85% | **15% ⇑** |
| 3 | To avoid a moratorium on user on-boarding, the business has decided to enable overcommitment on the kubernetes scheduler level. | **300 ⇑** | **200% ⇑** Configured Overcommitment | 100% | 85% | 15% |

[10] Overcommitment is the difference between Request and Limits. https://docs.openshift.com/container-platform/3.11/admin_guide/overcommit.html .

[11] Assume for this scenario the entire vSphere cluster is dedicated to this OCP development cluster.

| Day | Events and Observations on non-Production OCP development Cluster (Silver reservation grade) | **Observed Metric and Success Criteria**<br>Number of users on-boarded on the OCP development cluster<br>(More is Better) | **Configured Parameter**<br>Kubernetes Scheduler:<br>CPU Commitment [12]<br>(Overcommitment is more than 100%) | **Observed Metric**<br>Kubernetes Scheduler:<br>Utilization<br>(CPU Requested) | **Configured Parameter**<br>vSphere:<br>vCPUs allocation on underlining hardware<br>(Overallocation is more than 100%) | **Observed Metric**<br>vSphere: CPU Cluster Utilization [13] |
|---|---|---|---|---|---|---|
| 4 | Users reported kubernetes scheduler constrained again on CPUs, although vSphere cluster CPU is still underutilized *(25%)*. | **500 ⇑** | 200% | **200% ⇑**<br>Exhaustion | 85% | **25% ⇑** |
| 5 | The business has decided to over-allocate CPU on the vSphere level doubling the number of vCPUs on every application node. Users can now schedule their pods. vSphere cluster CPU utilization is still under an acceptable utilization (40%). | **750 ⇑** | 200% | **150% ⇓**<br>(Additional vCPUs on every node increased compute capacity to the kubernetes scheduler reducing the overallocation) | **170% ⇑**<br>Configured Overcommitment on vSphere | **40% ⇑** |

*Table 3: Journey of How vSphere Cluster Utilization Monitoring Has Driven Overallocation*

12 Overcommitment is the difference between Request and Limits. https://docs.openshift.com/container-platform/3.11/admin_guide/overcommit.html .

13 Assume for this scenario the entire vSphere cluster is dedicated to this OCP development cluster.

The over-commitment of CPUs dedicated to VMs (vSphere layer) on day 5 allowed another growth of the number of users on the OCP development cluster. It is evident in this scenario that kubernetes scheduling parameters from Day-1 to Day-5 have been sub-optimal (for more information about this learning, see the section The Importance of Defaults Requests and Limits).

After day 5, Application Platform Architects deployed fine-tuned kubernetes parameters (diminished default requests and limits, implemented reduction of replicas, and so on), but the business has decided to keep the over-allocation on vCPU both on kubernetes and vSphere for non-production clusters proving the value of resource management on both layers.

## OpenShift Backup

Red Hat recommends an environment wide backup of OCP cluster for disaster recovery, especially prior patching or a major configuration change (as a fallback procedure). You can complete this task almost entirely by performing regular operational procedures of snapshots of the master, infrastructure, and application node VMs.

Attention: You must back up etcd irrespective of any VMs snapshots. For more information, see the procedure as documented by Red Hat. Also, you can take the backup of OpenShift objects using VMware Project Velero.

## Master Node Recovery and Protection

There is always the chance of permanent loss of a master node. For more information about how to configure its recovery, see this OpenShift documentation. Application Platform Architects must create a runbook with procedures such as the master recovery, and hand these over to Operations. vSphere reduces the chances of such failures dramatically by providing giving HA protection.

**ATTENTION**: In case of two out of three master nodes are lost, the cluster becomes completely unavailable. Again, vSphere HA is another critical protection of the OCP cluster uptime.

## Cluster Patching and Migrations

### Patching

Updates of OCP 3.11 are released every month. These updates resolve security issues and must be executed across all VMs, especially on master nodes. Although patching is considered lower risk if compared to a major version upgrade, you must plan for a reliable fallback procedure in case of failure. vSphere comes to rescue with VMs snapshots and reduces the time spent in fallback procedures. If you are using VMDK mounted as Persistent Volumes, the snapshot of Application Nodes might not be possible (as of this writing not fully supported by vStorage API).

### Migration from OCP 3.11 to OCP 4.X

In specific scenarios, in-place upgrade (same cluster upgrade) of OCP clusters is not supported. This is the case for the migration from OCP 3.11 to the newly release OCP 4.1. The application workload can only be transferred from one OCP 3.11 cluster to another OCP 4.X cluster. Application Platform Architects refer to this type of migration as cluster-side migration or parallel cluster migration.

On bare-metal deployments, this migration procedure would necessarily demand hardware expenditure. Example.com plans to take advantage of running OCP clusters (Gold and Silver reservation grades) and use DRS and vMotion to redistribute utilization of the vSphere cluster and free up resources to instantiate the net-new OCP 4.1 clusters on the same underlining hardware.

The plan is:

1. Create OCP 4.X clusters. The minimum application node footprint (3 VMs) has no reservation, and the OCP 4.X management nodes have the full reservation.
2. Copy OCP objects from OCP 3.11 cluster to OCP 4.X cluster using Project Velero or a similar tool, and open OCP 4.X cluster to users to start migrating their workloads.
3. Monitor utilization of vSphere cluster and power on new OCP 4.X application node VMs if there are available resources on the vSphere cluster. Gradually power off OCP 3.11 application node VMs to release resources to OCP 4.X cluster. This might be a lengthy process, depending on the size and the number of users on-boarded on each OCP 3.11 cluster.
4. Platinum OCP 3.11 clusters most likely will require net-new compute capacity to migrate to another Platinum OCP 4.X cluster to honor the reservation grade of both clusters.

## Other Operational Costs of Hardware Maintenance

There is a multitude of tasks related to hardware configuration and maintenance (NIC teaming, disk replacement, firmware upgrades, and so on) that require more operational time and costs on a non-virtualized environment when compared to a virtualized environment. Hardware maintenance must never cause a microservice interruption, especially on kubernetes clusters that cannot afford a reduction in compute capacity.

## Other Lessons Learned on Enterprise OpenShift

This section documents other lessons learned from an enterprise deployment of Red Hat OpenShift. These lessons are not necessarily related to vSphere.

### OpenShift 3.11 Security Hardening

The Example.com security team requires all OCP clusters to be aligned to CIS benchmarks, and so, both OCP and Docker needed to have a layer of security on top of the Red Hat defaults.

Docker (configured in the ansible file)

Secure docker and increase log pool adding these parameters:

```
openshift_docker_options="--log-driver=json-file --log-opt max-size=250m --default-ulimit
nproc=10240:10240 --default-ulimit nofile=100000:100000"
```

etcd/Master (configured in the ansible file)

Eliminate weaker ciphers on the cluster adding these lines:

```
openshift_master_min_tls_version=VersionTLS12

openshift_node_min_tls_version=VersionTLS12

openshift_master_cipher_suites=['TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256',
'TLS_RSA_WITH_AES_256_CBC_SHA','TLS_RSA_WITH_AES_128_CBC_SHA']

openshift_node_cipher_suites=['TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256',
'TLS_RSA_WITH_AES_256_CBC_SHA','TLS_RSA_WITH_AES_128_CBC_SHA']

etcd_cipher_suites="TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_R
SA_WITH_AES_128_CBC_SHA"
```

The hardening configuration of OpenShift 4.X is still to be determined.

### The Importance of Defaults Requests and Limits

The values of default limits and requests are very critical for overall resource management and user experience. Example.com has had to tweak down kubernetes parameters multiple times during its journey to guarantee resources for their users (see vSphere Utilization Metrics and Over-allocation).

The mapping between vSphere vCPU and kubernetes Limits/Request is that 1 vCPU equals to 1,000 kubernetes millicores. This correlation is automatically performed on the kubernetes scheduler based on the detected number of CPUs that the Operating System exposes to the kubelet.

The initial default requests and limits have been very generous (based on assumed regular application footprint [14]): Requests were 500 millicores and 2 GB RAM, and limits were set to 1,250 millicores and 4 GB RAM.

14 See average assumption in Detailed Example of Calculation of Application Node Size .

The settings have caused several ripple effects:

- Kubernetes resources are exhausted very quickly, especially on development clusters (Silver reservation grade).
- Developers have taken the generous resources for granted and have not configured the requests and limits themselves on any cluster or environment. The practice of not explicitly set the requests and limits started on development, and it was followed to UAT and finally to cluster in production.
- Once kubernetes pods are in production on clusters, it is an operational challenge to negotiate with application developers to reduce requests and limits from the defaults so that Operations can retrieve the resources. We learned the following lessons:
    - Platinum clusters are great to guarantee performance, but it comes with a cost. Much of this cost can be mitigated assuming the initial default requests and limits are correct with start with.
    - Platinum clusters also can reduce the overall OCP clusters utilization because of existing tenants consuming resources, that they do not need and consequently preventing new tenants from being scheduled. Also, there was no incentive for users to reduce their OpenShift footprint since there were no chargebacks based on mechanisms on OCP utilization (projects).
- Example.com has run out of kubernetes CPU resources much faster than kubernetes' memory resources.

Example.com then tweaked down the default kubernetes pod requests to 250 millicores and 2 GB RAM, and default limits to 500 millicores and 4 GB RAM to all projects (namespaces) retroactively, even in production. The retrieved capacity was sufficient for a couple of months, but kubernetes resources have starved again. During another round of analysis, it was observed a significant number of idle containers running for months. They were consuming as little as 10 to 50 millicores each while locking 250 millicores for each copy of the pod.

Finally, Example.com purposely set the default request and limit to extremely low values on Silver OCP clusters to foster an enterprise practice of developers to overwrite the defaults. The third generation of values requests to 100 millicores and 1 GB RAM, and limits to 200 millicores and 2GB RAM for all projects on the development clusters. Production and UAT clusters were treated by case-by-case (in some instances, original requests and limits remained) with smaller parameters only for net-new projects.

### Scheduling Parameters: One Size Does Not Fit All

As seen in the last section, Example.com has realized quickly that "one size did not fit all". Every environment (development, UAT, or production) running on different OCP reservation grade types (Platinum, Gold, and Silver) requires different kubernetes scheduling parameters and quotas.

- Non-production (development) running on "Silver" OCP Cluster: Over-commitment, extremely lower default limits and request, QoS class Best Effort.
- UAT running on Gold OCP clusters: Project (namespaces) quotas have been more dynamic if compared to development and production. Multiple application teams have been running capacity tests demanding the entire cluster allocation over the weekends. Such tooling and procedures (to dynamically reallocate cluster resources) have been needed to Operations for capacity management (tooling and procedures part of the runbook created by Application Platform Architects).
- Production on Platinum OCP clusters: No over-commitment, with default requests and limits at the same value (to guarantee resources). Only Operations has been given the authorization of making changes on objects such as "limitranges".

It is recommended that any corporation define kubernetes scheduling policies and procedures on Day 0 for large, multi-tenant clusters and educate users about them.

### Firewall and OpenShift SDN

iptables has been the default OpenShift firewall since version 3.2. However, Red Hat has started to recommend using firewalld on OpenShift 3.5 and later. Firewalld has shown better results for Example.com compared to iptables. After migrating to firewalld, Example.com eliminated intermittent issues on pod-to-pod network communications. To enable firewalld, you can add the line "**`os_firewall_use_firewalld=True`**" to the ansible host file.

Example.com has opted to use ovs-networkpolicy on OCP 3.11 clusters without any issues (technical or operational). One of the challenges, however, it is the deployment of micro-segmentation at scale to dozens of OCP clusters and thousands of VMs. In this case, NSX-T integration is critical. See section Security and Networking with NSX.

### Container Logging: SLA or no SLA

One of the biggest challenges of Example.com has been to manage the logs of the application containers. The default OCP logging mechanism is not flexible for Example.com's requirements: log aggregation across multiple clusters, global and custom dashboards, extra metadata fields such as charge-back codes, cluster name, and so on.

As a result, Example.com set up its ElasticSearch infrastructure to support the exponential growth of container adoption. In a few months, the customary rate of logging per application node has grown to 6-8GB/hour with peaks of 11GB/hour, causing many capacity problems.

Upon some investigation, Example.com found multiple challenges on extensive usage of OpenShift logging:

- Multiple developers were leveraging container stdout to dump data structures and objects (besides standard application logs).
- A single noisy pod (with excessive log output) can impact and peg the entire logging subsystem for hours (that is, starting multiple copies of pods in debug mode).
- Data truncation happened due to docker and fluentd log rotation. To alleviate the number of occurrences of the truncation, you can increase the docker log size (that is, 250 MB per file) and fluentd log size (that is, 10 GB per file).
- Limit of 16 KB per line (docker 1.13.1).
- Absence of a solution to enforce log shaping, log pacing, and log quota per pod or project. (namespace).

Due to all these variables, Example.com has been unable to provide any Service Level Agreement (SLA) on the application container logging subsystem, even for Platinum reservation grade clusters. This is still a working in progress item.

## Conclusion and Call to Action

This paper has documented the journey of Example.com why and how any corporation can leverage vSphere to deploy Red Hat OpenShift successfully.

Relying on a VMware SDDC is necessary for any company of any size that is pursuing to run containers to support mission-critical applications.

At VMware, we believe that a fine-tuned kubernetes deployment is a critical building block of what matters: How well you have abstracted the application platform nature of your enterprise workloads and run them successfully. We encourage the audience to read the VMware Office of CTO Application Position Paper for more details.

Finally, read how VMware helps customers build modern applications and ensures customers run and manage kubernetes through strategic initiatives of VMware Tanzu and Project Pacific.

If you have any questions or feedback about this paper, please reach out to your VMware Technical Account Manager or octo-openshift@vmware.com.

## Appendix A: OpenShift 4

OpenShift 4 was released in June/2019 and it introduces multiple changes from OpenShift 3.11.

Table 4 below shows the major differences between the two versions:

| | OpenShift 3 (OCP3) | OpenShift 4 (OCP4) |
|---|---|---|
| Operating System | RHEL7 | RH Core OS |
| Default Container Runtime | Docker 1.13.1 | CRI-O 1.13 [15] |
| OCP Cluster Installation Method | OCP Ansible + Connected (Internet) or Disconnected | Ignition + Machine Config Operator + Connected (Required Direct Internet Connectivity – no Proxy) |
| OCP Cluster Custom/Post-Install Steps | User provided Ansible / Scripts | Cluster Operator SDKs |
| Management of Nodes / System Admin Tasks (i.e. Management of Operating System Files) | Ansible / Scripts / Configuration Management tools (Puppet/Chef/etc.) | MachineSets |

*Table 4: Differences between OpenShift 3 and OpenShift 4*

[15] Found only CRI-O Documentation on OpenShift 3.11 Manuals .

Table 5 below points notable similarities and differences of the sections used of this paper regarding OpenShift 4:

| Section | OpenShift 4 (OCP4) |
|---|---|
| Summary of OCP 3.11 Components | Infrastructure nodes are not created in the OCP4 installation process. However, there are so many "out-of-the-box" components on OCP4 (40+ projects, 150+ pods) that you can configure three worker nodes as infrastructure nodes yourself as a post-installation step as Red Hat documentation here.<br>Master nodes have an additional role of "Machine Controllers".<br>On OCP4, there is one more type of node: bootstrap node. This node is used only during installation, and it can be deleted afterward. |
| OCP Conceptual View | OCP3 and OCP4 have the same conceptual view. The only minor conceptual difference is Ansible is no longer a hard requirement on the admin node. Application Platform Architects and Operations can still configure and use Ansible separately if they want. |
| Master, Infrastructure, and Admin Nodes Sizes and<br>Application Node Sizes | Despite some lower minimum requirements for OCP4, we recommend using the VM sizes documented on these sections for the OpenShift 4 clusters. |
| OpenShift 3.11 Security Hardening | To be determined for OCP4. Some hardening parameters are to be similar, but with CRI-O and MachineSets, the method of hardening of container runtime is different on OCP4. |

*Table 5: Sections with Relevant Notes about OpenShift 4*

## Appendix B: VMware Fling "OpenShift on Cloud Automation Services"

VMware Flings are apps and tools built by VMware engineers and community "that are intended to be explored"[16]. In August/2019, VMware Office of CTO released code to allow the community to deploy OpenShift 3.11 clusters using VMware Cloud Automation Services. The fling follows the recommended components and settings as per sections OpenShift OCP Conceptual View, Master, Infrastructure, and Admin Nodes Sizes and OpenShift 3.11 Security Hardening .

The fling (Figure 8) is downloadable at https://labs.vmware.com/flings/enterprise-openshift-as-a-service-on-cloud-automation-services .

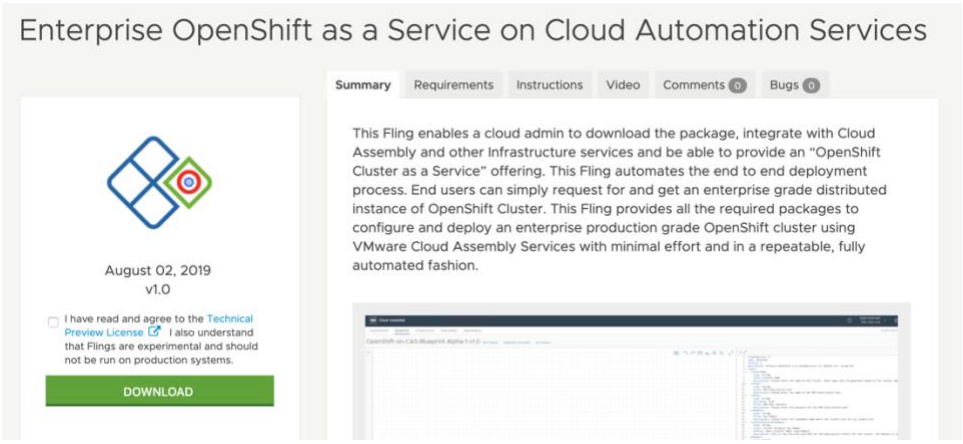


*Figure 8: Screenshot of the Fling OpenShift as a Service with CAS*

As of this writing, VMware Office of CTO is working on the fling to support OpenShift 4.

16 https://labs.vmware.com/flings

# Other Appendices

## Audience

This paper is intended to OpenShift engineers and architects, SREs, VMware practitioners, Engineer Managers, and decision-makers looking for guidance in reducing TCO when integrating vSphere and OpenShift.

Information security officers looking for overall security best practices on architecting OpenShift.

Ultimately, this paper is targeted on what VMware understands as the rise of a new persona in cloud-native applications: Application Platform Architect [17], which is an intersection of three disciplines of development, infrastructure, and production deployments.

## In the Scope and Out of the Scope of This Document

*In the scope* of this document is the rationale of rationale best practices and high-level design of enterprise OCP 3.11 on vSphere 6.5/6.7.

*Out of the scope* of this document is the rationale (or documentation) of:

- Containers versus non-container deployments.
- vSphere versus KVM.
- Red Hat OpenShift capabilities versus other kubernetes distros.
- "how-to" and "step-by-step" install and integration of OCP and VMware. For this, please refer to VMware Office of CTO Blog entry on Red Hat and VMware SDDC.
- Detailed information of OCP components and internals. For this, please refer to Red Hat OCP 3.11 documentation.
- Very detailed differences between OCP 3.11 and OCP 4.X.

## Hyperthreading

Too often, hyperthreading is disabled, which would potentially cause additional CPU cycles not to be utilized. Turning hyperthreading on does not mean you are getting the double of performance of the physical core. However, by turning hyperthreading on, you allow multiple vCPUs to be available to various workloads whenever they need the CPU cycles. By doing this, you are allowing the maximum ability for various kubernetes pods to be scheduled by multiple physical cores by requesting multiple vCPUs.

vSphere makes conscious CPU management decisions regarding mapping vCPUs to physical cores. An example is a VM with four virtual CPUs. Each vCPU will be mapped to a different physical core and not to two logical threads that are part of the same physical core. [18]

Example.com received tremendous benefits from turning on hyperthreading and the ability to allocate multiple vCPUs to various kubernetes pods as opposed to limiting it to a mapping of a single vCPU (or 1.3 vCPU) to one physical CPU.

Always enable hyperthreading and distribute and consume all the vCPUs for the OCP cluster, even if the entire OCP cluster is expected to be dedicated to single-threaded High-Performance workloads.

17 https://octo.vmware.com/vmware-octo-application-platforms-position-paper/

18 Reference: Hyper-Threading Section on SQL Server on VMware Best Practices Guide .

It is worth mentioning that public cloud services commonly use hyperthreaded (vCPU) charged as a CPU on almost all types of instances. The user needs to choose particular instance types to access the real physical CPU instead of the vCPU.

Additionally, kubernetes understands that one CPU is a hyper-threaded CPU on a bare-metal server.

Example.com took a similar approach of on vCPU of vSphere clusters adopting the following convention:

**1 physical CPU (bare metal) = 2 Logical Processors (vSphere Host) = 2 vCPUs (VM) = 2,000 kubernetes millicores**

For further discussion on for hyperthreading behavior, refer to vSphere 6.7 performance best practices - "ESXi systems manage processor time intelligently to guarantee that load is spread smoothly across all physical cores in the system." (page 22).

Another source of information is The Value of Running Kubernetes on vSphere, a VMworld 2018 presentation.

### Security and Networking with NSX

Having Red Hat OpenShift running on vSphere opens more avenues of integration to container deployments to other enterprise components.

This is the case of all OCP clusters on vSphere leveraging NSX Load Balancers for the OCP master and infra nodes load balancers. The NSX load balancer appliances reduce the operational costs and complexity of maintaining at the three load balancers per cluster[19].

Additionally, the NSX-T SDN can be integrated with OCP 3.11 clusters to achieve SDDC benefits such as complete micro-segmentation and advanced load balancing. For such, NSX-T Container Plug-In (NCP) must be installed on Day 0, which is along with the OCP cluster installation. NCP support is not possible on Day 1 (OCP clusters cannot be converted to NCP once they are running).

### vSphere as Cloud Provider to enable VMDK for Container Data Persistency

Red Hat OpenShift running on vSphere gives the opportunity to further integration of containers on other SDDC subsystems such as vSAN.

VMware vSAN allows storage resources attached directly to vSphere hosts to be used for distributed storage and accessed by multiple vSphere hosts.

OpenShift supports VMware vSAN (VMDK disks) as one of its providers of persistent storage for containers. The requirements are documented here.

At very high-level, the requirements are:

- OCP node VMs on a VM folder.
- OCP node VM names must follow these rules: cannot use capital letters, cannot start with numbers, cannot have special characters other than dashes.
- A vCenter user with roles to create and delete VMs and disks (see required roles). This user's credentials must be added to the ansible host file for OCP installation.

The integration of OCP 3.11 and VMDK can be done on Day 0 or Day 1 (at the installation of OCP or after the installation of OCP) as well. More references and details on this implementation and step-by-step on "Deploying and Managing Openshift 3.11 on a VMware Software-Defined Data Center" in Section 2.4.

[19] See OCP Conceptual View .

As of this writing, the combination of Proactive HA and vSAN is not supported.

### Detailed Example of Calculation of Application Node Size

In this section, we document a more elaborated step-by-step method illustrating how Example.com calculated the size of application nodes on a Platinum reservation grade cluster based on Scenario #2: Dedicated vSphere cluster to Support Single OCP Cluster. This is one example of how to calculate the sizing of application nodes, not necessarily the best way for all scenarios and every company.

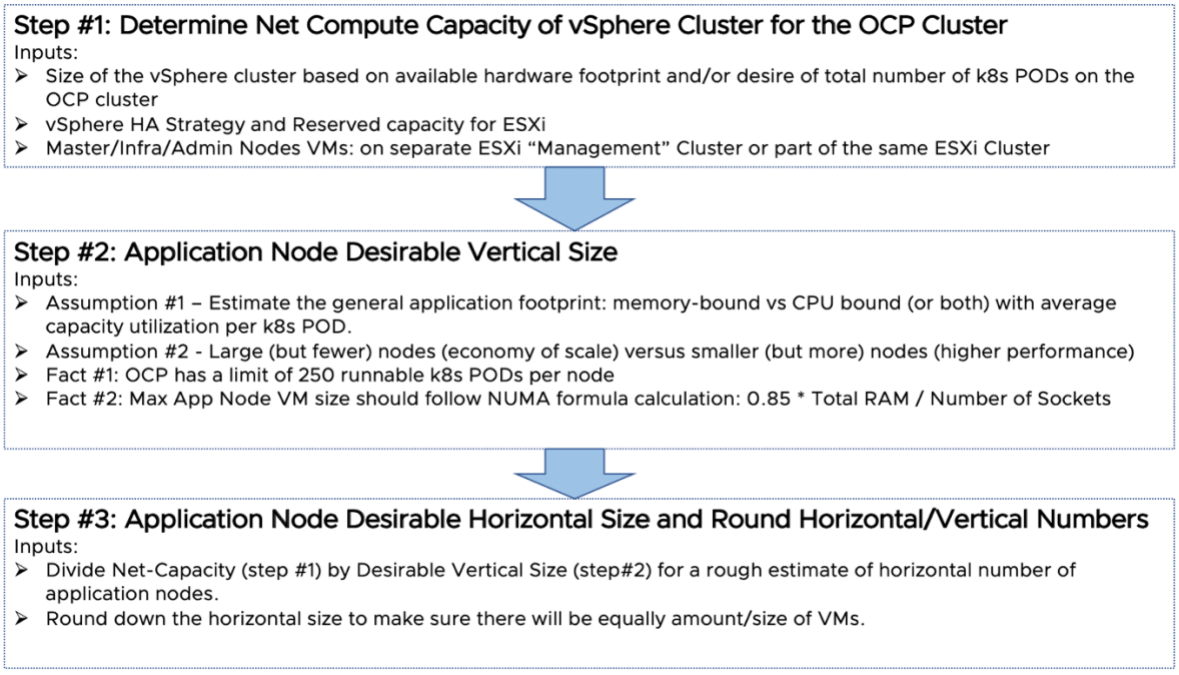


*Figure 9: An Example of How to Determine the Application Node Sizing - Step-by-Step*

*Step 1 – Determine the Net Compute Capacity of OCP Cluster*

The first step was to determine the net compute capacity for application nodes. There were four general parameters:

- Available hardware footprint on the vSphere cluster. It can be corporate standards (servers per rack, type of bare metal, etc.) or a business decision (total desirable kubernetes pods that OCP cluster would accommodate). For Example.com, the minimum size of any vSphere cluster was 16 vSphere hosts.
- vSphere HA strategy. There are multiple options. In this scenario, it reserves one vSphere host for a 16-Host vSphere cluster (through Cluster Resource Percentage).
- Reserved capacity for vSphere cluster headroom (which uses the same factor of the NUMA node formula, 0.85 or 15%).
- Master, infrastructure, and admin node VMs. Run these VMs on the same vSphere cluster or a different vSphere cluster (management cluster).

NOTE: The combined requirement of master, infrastructure, and admin node VMs is 46 vCPU and 152GB (see previous sections). For our example calculation, we will assume these VMs are under on a different vSphere management cluster.

The formula for the net capacity for application nodes is:

**(Total Number of vSphere Hosts on the vSphere cluster – Number of vSphere Hosts Reserved for HA) * (logical CPUs and RAM of each vSphere Host) * (Headroom Reserve) = Net Capacity**

Based on Scenario #2: Dedicated vSphere cluster to Support Single OCP Cluster, the vSphere cluster has 16 vSphere hosts. We are reserving the equivalent of one vSphere host for HA.

Example.com bare metal server is an Intel dual-socket with one NUMA node per socket. Total capacity is 56 physical CPUs (28 physical CPUs per NUMA node) and 1 TB RAM (512 GB RAM per NUMA node).

As a result, each vSphere host has 112 logical CPUs (hyperthreading enabled) and 1 TB RAM.

The final calculation of the net capacity for the application nodes VMs is:

(16 – 1) * (112 logical CPUs and 1TB RAM) * 0.85 = **1,428 logical CPUs and 12.75 TB RAM.**

*Step 2 – Application Node Optimal Vertical Size*

It is paramount to collect assumptions and facts about the environment to arrive at the optimal vertical size of application nodes.

*Assumption #1:* General application footprint and consumption. The kubernetes pods can be CPU-intensive (that is, performance-critical), memory-intensive (that is, cache apps), or a combination of both (that is, java apps). For our calculation, we estimated the kubernetes pods were balanced between CPU-bound and memory-bound with an average consumption of **500 millicores and 2GB per pod** (this estimation was arrived from observing typical tomcat application process running on a non-containerized environment).

*Assumption #2:* Preference of node sizes. Larger and fewer application nodes VMs translated into less operational costs for Example.com In the past, they had a VM sprawl (by sprawl, we refer to act of creating numerous VMs that are not fully utilized and that, otherwise, can be consolidated) that caused extensive administration overhead (inventory, patching, chargebacks, etc.).

*Fact #1:* OCP 3.11 has a limit of 250 pods per application node [20] .

*Fact #2:* The maximum memory size of Application Node VM must be less than NUMA node, preferably following the NUMA formula calculation that includes the reservation of the headroom for vSphere (which is 0.85, see Simple Method of Sizing Application Nodes ).

For our memory calculation, each application node VM vertical max size could be potentially up to 512 GB of RAM (multiplication of Assumption #1 and Fact #1) down to 435GB (Fact #2) to accommodate the largest application nodes VMs (Assumption #2) to run as many kubernetes sup to the limit of OpenShift.

In terms of vCPU, application node VMs must be created with a number of vCPUs equal to or less than the number of logical processors in each physical NUMA node.[21] One can apply the same memory NUMA formula for the vCPUs, with 0.85 factor for overhead: **`(0.85 * Total Logical Processors on Host) / Number of Sockets`**.

On our example, the optimal and theoretical vertical size for Example.com application node VM is ***48 vCPUs*** and ***436 GB RAM.***

[20] https://docs.openshift.com/container-platform/3.11/scaling_performance/cluster_limits.html .

[21] "Performance Best Practices for VMware vSphere 6.7", page 22 .

*Step 3 – Application Node Optimal Horizontal Size*

On the *horizontal* size is the number of application nodes from a rounded division of the net compute capacity (calculated in step 1) by the desirable vertical size (calculated in step 2).

Dividing the net compute capacity (see Step 1: 1,428 logical CPUs and 12.75 TB RAM) by the max vertical size (48 vCPUs and 435 GB RAM) yields in rounded 30 application nodes.

The final estimated application node size is ***30 application nodes with 48 vCPUs and 435 GB RAM*** (with a ratio of 1 vCPU per 8.85 GB RAM).

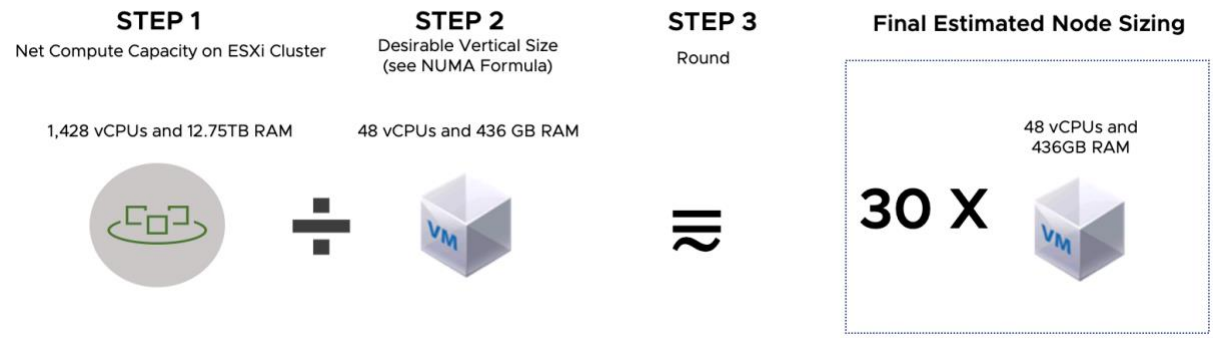


*Figure 10: Final Calculation of Application Node VM Sizing*

As mentioned previously, this is an example of a calculation of the sizing of application nodes used on Example.com Application Platform Architects can make their formulas based on other constraints, business requirements, initial assumptions, and so on.

## List of Figures



## List of Tables

## Acknowledgements

This document was authored by the following people:

- Rafael Brito
- Emad Benjamin
- Michael Gasch

The authors would like to thank the following people for their feedback:

- Sajal Debnath
- Moises Navarro
- Cormac Hogan
- Hal Rosenberg
- Frank Denneman
- Michael Patton
- Carlisia Campos
- Niranjan Jahagirdar
- Duncan Epping
- Eric Railine
- Kit Colbert
- Paul Fazzone

## Legal Notice

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com.


vmware®