



Pod density comparison: VMware vSphere with Tanzu vs. a bare-metal approach using Red Hat OpenShift 4.7

The solutions we tested

We deployed two on-premises cloud platforms with Kubernetes integrations: VMware® vSphere® 7 Update 2 (7U2) with VMware Tanzu™ and bare-metal Red Hat® OpenShift® 4.7. Our goal was to quantify the maximum number of pods that each solution supported. Due to limitations of the kubelet, Kubernetes API, and other platform components, each Kubernetes node can support only a maximum number of pods regardless of the underlying physical resources. In a bare-metal environment, each node runs on a dedicated physical server. Virtualized environments provide a workaround to those limitations, however, by supporting more nodes and pods on the same physical hardware. Virtual environments can thus offer a pod density advantage for Kubernetes compared to bare-metal solutions.

We configured the worker hosts on each cloud platform on five HPE ProLiant DL380 Gen10 servers with identical processors, memory, and storage configurations. We kept the number of hosts performing workloads in each deployment consistent across both platforms to represent a company deploying either cloud platform onto a defined number of servers. Each solution had different requirements for management, storage, and worker hosts.

For the vSphere 7U2 with Tanzu deployment, we configured all five hosts as a complete vSphere cluster with Tanzu Kubernetes capabilities. For the bare-metal Red Hat OpenShift deployment, we configured three hosts (i.e., servers) as management hosts and five hosts as worker hosts, for a total of eight hosts.

Our test approach

Given the identical hardware in our two five-host clusters, one might expect that they would support similar numbers of pods. To explore this, we conducted pod density testing using a minimal web-service workload that supported a basic website. Each pod in the workload ran a replica of the website, and a standard Kubernetes load balancer split the traffic evenly across all pods. We started with a small number of pods and gradually increased the number until we reached the point where increasing pod count further would cause performance deterioration or cluster instability.

What we found

The vSphere with Tanzu environment supported 13,700 pods, 6.3 times the 2,150 pods that the bare-metal Red Hat OpenShift 4.7 cluster supported.

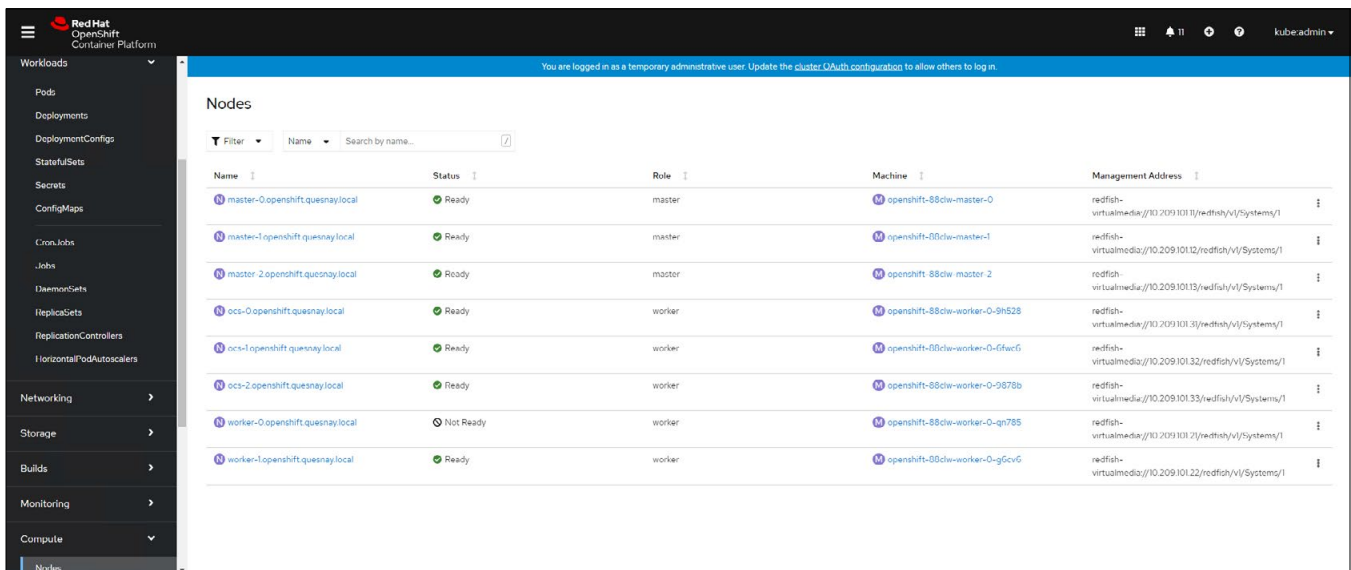
Red Hat OpenShift solution

According to Red Hat OpenShift documentation, the platform supports up to 250 pods per node, or physical host in our case. However, Red Hat provides additional documentation on scaling to 500 pods per node (or physical host in our case). We set out to test this higher limit and see whether our five-host bare-metal cluster could support 2,500 pods. To do so, we modified default parameters as follows to allow the cluster to achieve the theoretical max pod count:

- kubeletConfig
 - maxPods: 520 #default is 250
 - kubeAPIBurst: 200 #default is 100
 - kubeAPIQPS: 100 #default is 50

We also modified the install.yaml configuration by changing the hostPrefix networking setting to 22 (from a default of 23).

For the Red Hat OpenShift solution, we stopped our pod count increase when we began observing instability in the physical hosts operating as worker nodes. We found that when we exceeded 2,150 of our workload pods (430 workload pods per host), individual worker node hosts would unpredictably go offline, and pod counts would drop accordingly. In Figure 1, the status for the second-to-last node is "Not Ready," illustrating the behavior we observed.



Name	Status	Role	Machine	Management Address
master-0.openshift.queensay.local	Ready	master	openshift-88clw-master-0	redfish-virtualmedia://10.209.101.11/redfish/v1/Systems/1
master-1.openshift.queensay.local	Ready	master	openshift-88clw-master-1	redfish-virtualmedia://10.209.101.12/redfish/v1/Systems/1
master-2.openshift.queensay.local	Ready	master	openshift-88clw-master-2	redfish-virtualmedia://10.209.101.13/redfish/v1/Systems/1
ocs-0.openshift.queensay.local	Ready	worker	openshift-88clw-worker-0-9h528	redfish-virtualmedia://10.209.101.31/redfish/v1/Systems/1
ocs-1.openshift.queensay.local	Ready	worker	openshift-88clw-worker-0-6fac6	redfish-virtualmedia://10.209.101.32/redfish/v1/Systems/1
ocs-2.openshift.queensay.local	Ready	worker	openshift-88clw-worker-0-9878b	redfish-virtualmedia://10.209.101.33/redfish/v1/Systems/1
worker-0.openshift.queensay.local	Not Ready	worker	openshift-88clw-worker-0-qn785	redfish-virtualmedia://10.209.101.7/redfish/v1/Systems/1
worker-1.openshift.queensay.local	Ready	worker	openshift-88clw-worker-0-g5cv6	redfish-virtualmedia://10.209.101.22/redfish/v1/Systems/1

Figure 1: Red Hat OpenShift solution node dashboard. Source: Principled Technologies.

Figure 2 shows 2,150 stable pods.

```

Waiting for deployment 'hello-kubernetes' rollout to finish: 2078 of 2100 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2085 of 2100 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2091 of 2100 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2094 of 2100 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2095 of 2100 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2097 of 2100 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2098 of 2100 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2099 of 2100 updated replicas are available...
deployment 'hello-kubernetes' successfully rolled out
[kni@provisioner ~]$ oc rollout status deployment/hello-kubernetes -n hello
Waiting for deployment 'hello-kubernetes' rollout to finish: 2100 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2101 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2102 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2103 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2104 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2105 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2106 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2107 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2108 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2109 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2110 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2111 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2112 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2113 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2114 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2115 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2116 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2117 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2118 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2119 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2120 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2121 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2122 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2123 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2124 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2125 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2126 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2127 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2128 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2129 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2130 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2131 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2132 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2133 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2134 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2135 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2136 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2137 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2138 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2139 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2140 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2141 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2142 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2143 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2144 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2145 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2146 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2147 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2148 of 2150 updated replicas are available...
Waiting for deployment 'hello-kubernetes' rollout to finish: 2149 of 2150 updated replicas are available...
deployment 'hello-kubernetes' successfully rolled out
[kni@provisioner ~]$ oc get deployments -n hello
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
hello-kubernetes                     2150/2150  2150          2150         35m
[kni@provisioner ~]$

```

Figure 2: Red Hat OpenShift environment showing 2,150 stable pods. Source: Principled Technologies.

We did not observe any specific software or hardware bottleneck or error message to identify and resolve the source of the host instability we experienced. All our further attempts to deploy a pod count beyond 2,150 ultimately resulted in unpredictable host instability for one or more hosts operating as worker hosts. We found that the most reliable way to bring the offline worker hosts back was to delete all pods, reboot the offline host, and redeploy the pods at a maximum of 2,150 for stable cluster performance.

As Figure 3 shows, the cluster utilization metrics did not reveal bottlenecks in any areas. (Note: The dramatic dip in the Pod count chart reflects a previous test that failed.)

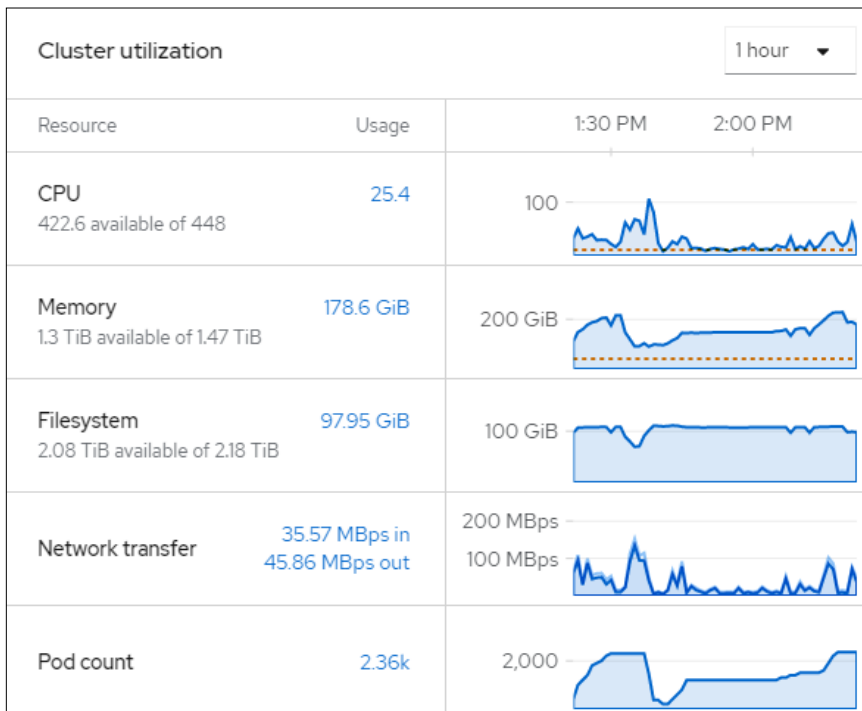


Figure 3: Red Hat OpenShift environment utilization screen. Source: Principled Technologies.

VMware vSphere with Tanzu solution

In contrast to the Red Hat OpenShift solution, we observed no node instability with VMware vSphere with Tanzu. On this platform, we were able to steadily increase the worker node VMs and the number of pods, which allowed us to scale the number of pods to 13,700, the point at which we reached 95 percent memory utilization. Figure 4 shows the 13,700 stable pods, and Figure 5 shows the memory utilization. We determined this to be the maximum because continuing to increase pod counts would have caused performance deterioration across the cluster due to memory paging because of overcommitted memory resources.

```

user@tkg-client:~$ kubectl scale deployment hello-kubernetes --replicas=13000
deployment.apps/hello-kubernetes scaled
user@tkg-client:~$ kubectl rollout status deployment/hello-kubernetes
Waiting for deployment "hello-kubernetes" rollout to finish: 13122 out of 13000 new replicas have been updated...
Waiting for deployment "hello-kubernetes" rollout to finish: 12988 out of 13000 new replicas have been updated...
Waiting for deployment "hello-kubernetes" rollout to finish: 12989 out of 13000 updated replicas are available...
Waiting for deployment "hello-kubernetes" rollout to finish: 12997 of 13000 updated replicas are available...
deployment.apps/hello-kubernetes successfully rolled out
user@tkg-client:~$ kubectl scale deployment hello-kubernetes --replicas=13500
deployment.apps/hello-kubernetes scaled
user@tkg-client:~$ kubectl rollout status deployment/hello-kubernetes
Waiting for deployment "hello-kubernetes" rollout to finish: 13122 out of 13500 new replicas have been updated...
Waiting for deployment "hello-kubernetes" rollout to finish: 13337 out of 13500 new replicas have been updated...
Waiting for deployment "hello-kubernetes" rollout to finish: 13344 out of 13500 new replicas have been updated...
Waiting for deployment "hello-kubernetes" rollout to finish: 13471 out of 13500 new replicas have been updated...
Waiting for deployment "hello-kubernetes" rollout to finish: 13438 of 13500 updated replicas are available...
Waiting for deployment "hello-kubernetes" rollout to finish: 13439 of 13500 updated replicas are available...
Waiting for deployment "hello-kubernetes" rollout to finish: 13467 of 13500 updated replicas are available...
Waiting for deployment "hello-kubernetes" rollout to finish: 13499 of 13500 updated replicas are available...
deployment.apps/hello-kubernetes successfully rolled out
user@tkg-client:~$ kubectl scale deployment hello-kubernetes --replicas=13700
deployment.apps/hello-kubernetes scaled
user@tkg-client:~$ kubectl rollout status deployment/hello-kubernetes
Waiting for deployment "hello-kubernetes" rollout to finish: 13388 out of 13700 new replicas have been updated...
Waiting for deployment "hello-kubernetes" rollout to finish: 13588 out of 13700 new replicas have been updated...
Waiting for deployment "hello-kubernetes" rollout to finish: 13662 out of 13700 new replicas have been updated...
Waiting for deployment "hello-kubernetes" rollout to finish: 13691 of 13700 updated replicas are available...
Waiting for deployment "hello-kubernetes" rollout to finish: 13693 of 13700 updated replicas are available...
Waiting for deployment "hello-kubernetes" rollout to finish: 13699 of 13700 updated replicas are available...
deployment.apps/hello-kubernetes successfully rolled out
user@tkg-client:~$ kubectl get services
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
hello-kubernetes    LoadBalancer  172.17.0.258     172.31.254.71    80:32194/TCP     62m
kubernetes           ClusterIP   172.17.0.1      <none>           443/TCP          90m
supervisor           ClusterIP   None            <none>           <none>           90m
user@tkg-client:~$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
hello-kubernetes    13700/13700  13700        13700      62m
user@tkg-client:~$ kubectl get pods > full-pod-list.txt

```

Figure 4: VMware vSphere with a Tanzu cluster showing 13,700 stable pods. Source: Principled Technologies.

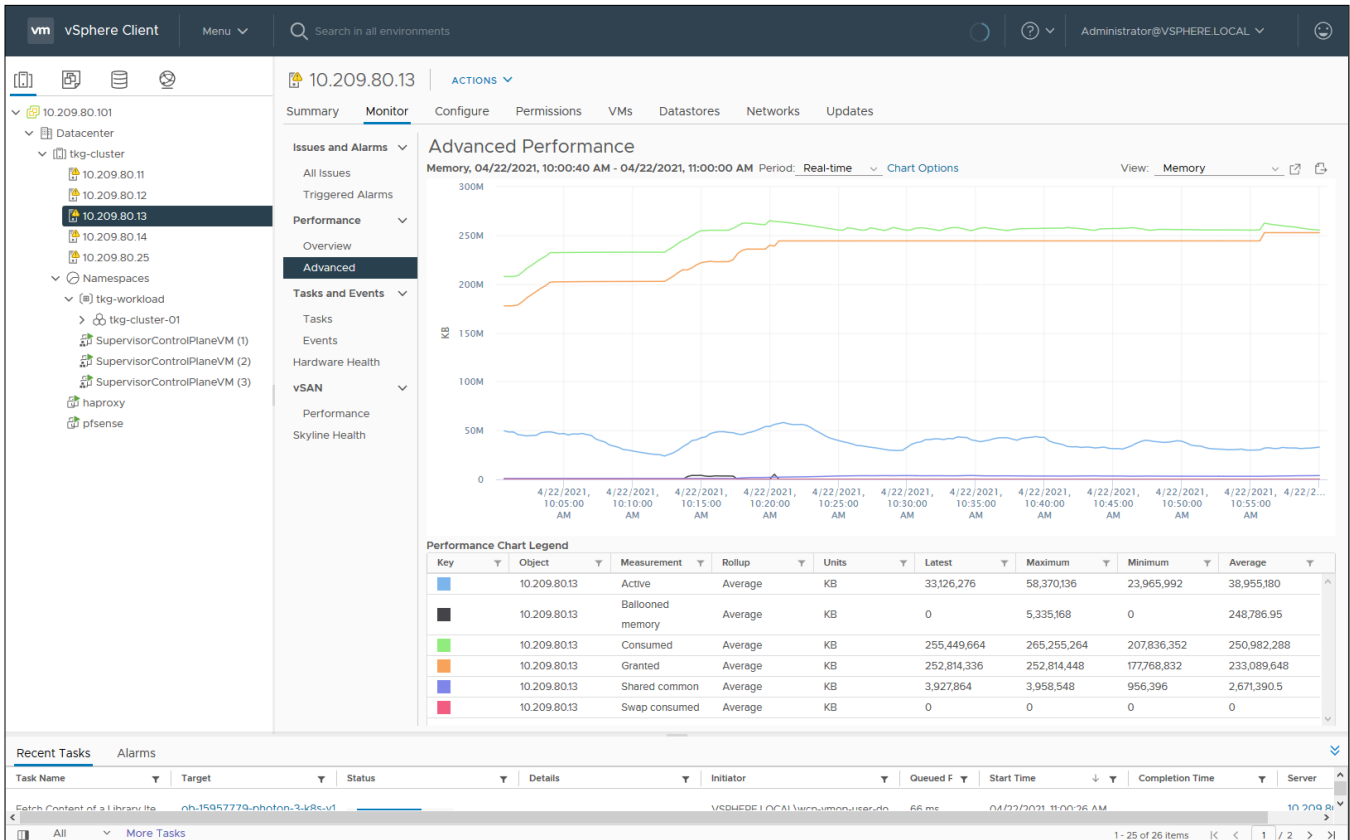


Figure 5: VMware vSphere with Tanzu cluster memory utilization screen. Source: Principled Technologies.

Figure 6 illustrates the difference we observed. The 13,700 pods that the vSphere with Tanzu cluster supported was 6.3 times the 2,150 pods the bare-metal Red Hat OpenShift cluster could handle in our testing. The vSphere with Tanzu cluster's maximum pod count of 13,700 is 5.4 times the theoretical maximum 2,500 pods that Red Hat OpenShift should support per Red Hat documentation. We ran more pods in our virtualized vSphere environment because it supported more worker nodes than the Red Hat bare-metal solution.

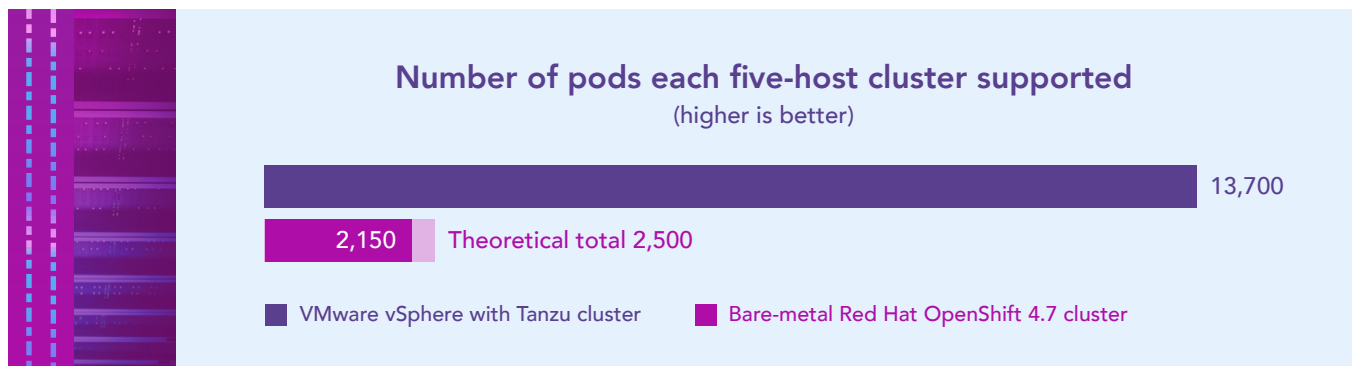


Figure 6: The number of pods each cluster supported in our testing. Source: Principled Technologies.

Clearly, in situations where it is necessary to run a large number of simple web app workloads, the vSphere with Tanzu cluster made much more efficient use of CPU and memory resources across the worker nodes, giving it a substantial advantage over the bare-metal Red Hat OpenShift cluster. In addition, the vSphere with Tanzu cluster did not require additional management hosts, while the Red Hat OpenShift cluster required three additional control plane hosts.

The greater efficiency of the vSphere with Tanzu environment on this workload could translate to significant CapEx and OpEx savings because you would theoretically need an environment five to six times the size to run the same number of pods in a bare-metal Red Hat environment. The greater pod density that the vSphere with Tanzu environment supports could allow platform administration teams to work with their infrastructure peers to reduce hardware spend.

Appendix A – System configuration information

Table 1: System configuration information for the test clusters. Note that each cluster comprised five servers.

System configuration information	vSphere with Tanzu cluster systems	Bare-metal Red Hat OpenShift cluster systems
Server	HPE ProLiant DL380 Gen10	HPE ProLiant DL380 Gen10
BIOS name and version	U30 v2.34	U30 v2.34
Operating system name and version/build number	VMware ESXi™ 7.0.2	Red Hat Enterprise Linux® CoreOS 4.18 47.83.202106252242-0
Date of last OS updates/patches applied	4/1/2021	4/1/2021
Power management policy	Dynamic Power Savings Mode	Dynamic Power Savings Mode
Processor		
Number of processors	2	2
Vendor and model	Intel® Xeon® Gold 5120 CPU	Intel Xeon Gold 5120 CPU
Core count (per processor)	14	14
Core frequency (GHz)	2.2	2.2
Stepping	-	M0
Memory module(s)		
Total memory in system (GB)	256	256
Number of memory modules	4	4
Vendor and model	Hynix® 809085-091	Hynix 809085-091
Size (GB)	64	64
Type	PC4-19200T	PC4-19200T
Speed (MHz)	2,400	2,400
Speed running in the server (MHz)	2,400	2,400
Storage controller 1		
Vendor and model	HPE Smart Array P408i-a SR Gen10	HPE Smart Array P408i-a SR Gen10
Firmware version	1.04	1.04
Local storage (capacity)		
Number of drives	2	2
Drive vendor and model	Intel S4500 SSDSC2KG96	Intel S4500 SSDSC2KG96
Drive size (GB)	960	960
Drive information (speed, interface, type)	6Gbps, SATA, SSD	6Gbps, SATA, SSD
Local storage (cache)		
Number of drives	2	2
Drive vendor and model	Intel S3700 SSDSC2BA40	Intel S3700 SSDSC2BA40
Drive size (GB)	400	400
Drive information (speed, interface, type)	6Gbps, SATA, SSD	6Gbps, SATA, SSD

System configuration information	vSphere with Tanzu cluster systems	Bare-metal Red Hat OpenShift cluster systems
Network adapter		
Vendor and model	HPE 331i	HPE 331i
Number and type of ports	4x 1Gb Ethernet	4x 1Gb Ethernet
Firmware version	20.6.41	20.6.41
Network adapter 2		
Vendor and model	Mellanox® ConnectX-3 Ethernet Adapter	Mellanox ConnectX-3 Ethernet Adapter
Number and type of ports	2x 10GbE SFP	2x 10GbE SFP
Firmware version	02.40.70.04	02.40.70.04
Cooling fans		
Vendor and model	HPE PFR0612XHE	HPE PFR0612XHE
Number of cooling fans	6	6
Power supplies		
Vendor and model	HPE 865408-B21	HPE 865408-B21
Number of power supplies	2	2
Wattage of each (W)	500	500

Appendix B – How we tested

Deploying VMware vSphere 7U2 with Tanzu

Installing ESXi

1. Boot into the ESXi install ISO.
2. In the installation welcome screen, press Enter to continue.
3. In the EULA, press F11 to accept and continue.
4. Select the correct drive, and press Enter to continue.
5. Select the keyboard layout, and press Enter to continue.
6. Enter a root password, and press Enter to continue.
7. In the confirm install screen, press F11 to install.
8. Repeat steps 1 through 7 for the remaining four hosts.

Configuring ESXi

1. Log into the ESXi server console.
2. Select Configure Management Network.
3. Select IPv4 Configuration.
4. In IPv4 Configuration, select a static IPv4 address, enter the appropriate IP address, and press Enter.
5. Select DNS Configuration.
6. In DNS Configuration, enter your Primary DNS server and fully qualified domain name, and press Enter.
7. To exit the management network configuration, press Esc.
8. When prompted, press Y to apply changes and restart the management network.
9. Select Troubleshooting Options.
10. In Troubleshooting Mode Options, enable ESXi Shell and SSH, and press Esc to confirm.
11. Repeat steps 1 through 10 for the remaining four hosts.

Enabling NTP on ESXi

1. Log into the ESXi server GUI.
2. Click Manage.
3. Click Time & date.
4. Click Edit NTP Settings.
5. In Edit time configuration, select Use Network Time Protocol (enable NTP client), select Start and stop with host, add your NTP server, and click Save.
6. Repeat steps 1 through 5 for the remaining four hosts.

Installing the VMware vCenter Server Appliance

1. Download the VMware vCenter® installation ISO from the VMware support portal at <https://my.vmware.com>.
2. Mount the image on your local system, and browse to the vcsa-ui-installer folder. Expand the folder for your OS, and launch the installer if it doesn't automatically begin.
3. Once the vCenter Server Installer wizard opens, click install.
4. To begin installation of the new vCenter server appliance, click Next.
5. Check the box to accept the license agreement, and click Next.
6. Enter the IP address of a temporary ESXi host. Provide the root password, and click Next.
7. To accept the SHA1 thumbprint of the server's certificate, click Yes.
8. Accept the VM name, and provide and confirm the root password for the VCSA. Click Next.
9. Set the size for environment you're planning to deploy, check Thin Disk, and click Next.
10. Select the datastore to install on, accept the datastore defaults, and click Next.
11. Enter the FQDN, IP address information, and DNS servers you want to use for the vCenter server appliance. Click Next.
12. To begin deployment, click Finish.
13. When Stage 1 has completed, click Close. To confirm, click Yes.
14. Open a browser window, and connect to [https://\[vcenter.fqdn:5480/](https://[vcenter.fqdn:5480/).
15. On the Getting Started - vCenter Server page, click Set up.
16. Enter the root password, and click Log in.
17. Click Next.

18. Enable SSH access, and click Next.
19. To confirm the changes, click OK.
20. Enter `vsphere.local` for the Single Sign-On domain name, enter a password for the administrator account, confirm it, and click Next.
21. Click Next.
22. Click Finish.

Creating a cluster in VMware vSphere

1. Open a browser and enter the address of the vCenter server you deployed ([https://\[vcenter.FQDN\]/ui](https://[vcenter.FQDN]/ui)).
2. Select the vCenter server in the left panel, right-click, and select New Datacenter.
3. Provide a name for the new data center, and click OK.
4. Select the data center you just created, right-click, and select New Cluster.
5. Give a name to the cluster, and enable VMware vSphere® DRS, HA, and VMware vSAN™. Click OK.
6. In the cluster configuration panel, under Add hosts, click Add.
7. Check the box for Use the same credentials for all hosts. Enter the IP Address and root credentials for the first host, and the IP addresses of all remaining hosts. Click Next.
8. Check the box beside Hostname/IP Address to select all hosts, and click Ok.
9. Click Next.
10. Click Finish.

Configuring Tanzu

The steps below contain the methodology for setting up the pre-requisites and deploying Tanzu in vSphere. These instructions assume you have at least a five-host vSphere cluster running ESXi and vCenter, backed by vSAN storage. These instructions also assume you have configured upstream networking switches to accommodate the VLANs and routing for your environment.

Tanzu prerequisites: Defining networks and IP addressing

In this section, we define the networks and addresses we used in this deployment.

Management network

This network can interact with all vSphere components and ESXi hosts. We used our VM network on a standard (not Distributed) vSwitch. We used a 10.209.0.0/16 network scope with a gateway of 10.209.0.1. We did not assign VLAN tagging for this network. (All ports were “untagged” or access.)

vMotion network

We defined the VMware vSphere vMotion® Network port group on a distributed vSwitch, which Tanzu Kubernetes deployment requires. We used a 192.168.12.0/24 network scope for the vMotion network. We assigned this network as VLAN 1612.

Storage network

We defined the Storage Network port group on a Distributed vSwitch, which Tanzu Kubernetes deployment requires. We used a 192.168.13.0/24 network scope for the storage network. We assigned this network as VLAN 1613.

Workload network

We isolated this network behind a NAT gateway and used private addresses for all connectivity. We defined the Workload Network port group on a Distributed vSwitch, which Tanzu Kubernetes deployment requires. We used a 192.168.0.0/24 network scope for the workload network with a NAT gateway of 192.168.1.1. We assigned the network as VLAN 1614.

Required network addresses

We used the following network addresses for our configuration. Your environment may use a different address scheme as long as it meets the requirements listed under each address section.

- HAProxy management address: 10.209.201.200/16, gateway 10.209.0.1
- HAProxy Workload address: 192.168.1.2/24, gateway 192.168.1.1
- Cluster management addresses 10.209.201.201 - 10.209.201.205
 - The Supervisor Control Plane VMs use these addresses to manage the namespace.
 - These must be five contiguous IP addresses.
 - ♦ You will only provide the starting IP of 10.209.201.201, which the shared cluster IP use.
 - ♦ The next three addresses will be individual VM IP addresses.
 - ♦ One will be for upgrade purposes.
- Worker VM addresses: 192.168.1.64/26 (192.168.1.65 - 192.168.1.126)
 - The workload network uses these addresses to provision deployed services, and the Supervisor Control Plane VMs use them to provide connectivity to those services.
 - In this case, the CIDR notation defines a scope of addresses. When inputting the address range, exclude the first (.64) and last (.127) addresses
- Load Balancer addresses: 192.168.1.240/29 (192.168.1.240 - 192.168.1.247)
 - The HAProxy management address and must not overlap any other network or VM (especially gateways) because the HAProxy will bound and respond to all the addresses.
 - In this case, the CIDR notation includes the first and last addresses. You will need to provide all eight addresses to the wizard.

Configuring a cluster through Quickstart

1. Select Menu → Hosts and Clusters.
2. Click the TKG Cluster.
3. In the right pane, the Quickstart view should automatically appear. Click Configure.
4. In Distributed switches, accept the defaults, add your host adapters, and click Next.
5. In vMotion traffic, complete the fields as follows, and click Next:
 - Use VLAN: 1612
 - Protocol: IPv4
 - IP Type: Static
 - Host #1 IP: 192.168.12.11
 - Host #1 subnet: 255.255.255.0
 - Check the autofill checkbox.
6. In Storage traffic, complete the fields as follows:
 - Use VLAN: 1613
 - Protocol: IPv4
 - IP Type: Static
 - Host #1 IP: 192.168.13.11
 - Host #1 subnet: 255.255.255.0
 - Check the autofill checkbox
7. In Advanced options, leave defaults, and click Next.
8. In Claim disks, ensure the correct disks are selected, and click Next.
9. In Create fault domains, accept the defaults, and click Next.
10. In Ready to complete, click Finish to create the cluster.

Adding the Tanzu Kubernetes Grid Workload Network port group to the distributed switch

1. Right-click the DvSwitch you created during the cluster Quickstart wizard, and select Distributed Port Group → New Distributed Port Group.
2. Give the name `Workload Network`, and click Next.
3. Change the VLAN type to VLAN, set the VLAN ID to VLAN 1614, and click Next.
4. Click Finish.

Creating a DevOps user

Create a dedicated vSphere user account to access Kubernetes namespaces.

1. From the vSphere client, click Home → Administration.
2. In the left panel, click Users and Groups.
3. In the right panel, click Users, select the vsphere.local domain, and click Add.
4. Provide a username and password, and click Add.
5. For simplicity, we added the DevOps user to a group with Administrator privileges. Click Groups, and select the Administrators group. Click Edit.
6. Under Add Members, search for the DevOps user you just created, and add them to the administrators group. Click Save.

Creating the HAProxy content library

Create a content library for housing the HAProxy appliance source image.

1. Click the following link to download the vSphere compatible HAProxy OVF(v0.1.8): <https://github.com/haproxytech/vmware-haproxy>.
2. From the vSphere client, in the left menu pane, click Content Libraries.
3. In the Content Libraries panel on the right, click Create.
4. Name the content library `HAProxy-cl` and click Next.
5. Accept the default, and click Next.
6. Choose the storage location for the content library, and click Next.
7. Review, and click Finish.
8. Click the newly created HAProxy-cl content library.
9. In the upper portion of the right-side panel for HAProxy-cl, click the actions pull-down menu, and select Import Item.
10. Change the selection to local file, and click the upload files button.
11. Browse to the location of the OVF file you downloaded in step 1, and click Open.
12. Click Import.

Creating the TKG content library

Create a Tanzu Kubernetes Grid (TKG) content library for Kubernetes-specific images.

1. From the vSphere client, in the left menu pane, click Content Libraries.
2. In the Content Libraries panel on the right, click Create.
3. Name the content library `TKG-cl`, and click Next.
4. Select Subscribed content library, and use <https://wp-content.vmware.com/v2/latest/lib.json> for the subscription URL. Click Next .
5. To verify, click Yes.
6. Choose the storage location for the content library, and click Next.
7. Review, and click Finish.

Installing a pfSense gateway

1. Download the most recent release of pfSense from: <https://www.pfsense.org/download/>.
2. Extract the pfSense ISO.
3. Select Menu → Hosts and Clusters.
4. Right-click your TKG cluster, and select New Virtual Machine.
5. In Select a creation type, select Create a new virtual machine, and click Next.
6. In Select a name and folder, name your pfSense VM, and click Next.
7. In Select a compute resource, select your TKG cluster, and click Next.
8. In Select storage, select your vSAN datastore, and click Next.
9. In Select compatibility, accept defaults, and click Next.
10. In Select a guest OS, change the OS family to Linux and the OS version to Debian GNU/Linux 10 (64-bit), and click Next.

11. In Customize hardware, change the following:
 - CPU: 2
 - Memory: 8 GB
 - Hard Disk: 32 GB
 - SCSI controller: LSI Logic Parallel
 - Network 1: Management Network
 - Network 2: Workload Network
12. Click Next.
13. Click Complete.
14. Right-click your newly created VM, and select Power On.
15. Right-click the now-on VM, and select Open Remote Console.
16. Select Removable Devices → CD/DVD drive 1 → Connect to Disk Image File (iso).
17. Select the pfSense ISO you downloaded.
18. After the VM boots off of the pfSense ISO, accept the license agreement.
19. Select Install.
20. Select Continue with the default keymap.
21. Select Auto (UFS) BIOS.
22. Select No.
23. Select Reboot.
24. When prompted to set up VLANs, select No.
25. For the WAN interface, select vmx0.
26. For the LAN interface, select vmx1.
27. To proceed, press Y.
28. In a web browser, navigate to the vmx1 IP Address of the pfSense VM (it should be 192.168.1.1).
29. Select to ignore the security warning.
30. Sign into the console (default username admin/pfsense).
31. In the welcome screen, click Next.
32. Click Next.
33. Enter your DNS server, and click Next.
34. Enter your time zone, and click Next.
35. In Configure WAN interface, leave defaults, and click Next.
36. In Configure LAN interface, level defaults, and click Next.
37. In Set Admin WebGUI Password, enter the password you want for the admin user, and click Next.
38. To apply the pfSense changes, click Reload.
39. To finish the pfSense installation, click Finish.

Creating storage tag

Create a storage tag for use in storage policies to control virtual machine placement.

1. From the vSphere client, select Menu → Storage.
2. From the left pane, select the storage you want to tag for Tanzu.
3. Under the Summary tab, locate the Tags panel, and click Assign.
4. Click Add Tag.
5. Name the tag `Tanzu`. Click Create New Category.
6. Give the category name `Tanzu Storage`. Clear all object types except Datastore, and click Create.
7. Use the Category pull-down menu to select Tanzu Storage, and click Create.
8. Check the box beside the newly created tag, and click Assign.

Creating VM storage policies

1. From the vSphere client, click Menu → Policies and Profiles.
2. On the left panel, click VM Storage Policies.
3. Click Create.
4. Create a new VM Storage policy named `tkg-cl`, and click Next.
5. Check the box for Enable tag based placement rules, and click Next.
6. Use the Tag Category pulldown menu to select the Tanzu Storage policy you created. Click Browse Tags.
7. Click the Tanzu checkbox, and click OK.
8. Click Next.
9. Review the compatible storage to make sure your storage target is marked as compatible, and click Next.
10. Click Finish.

Deploying the HAProxy load balancer

Deploy the load balancer supported for Tanzu Kubernetes with vSphere

1. From the vSphere client, click Menu → Content Libraries.
2. Click the HAproxy-cl library.
3. In the left panel, click OVF & OVA Templates, and right-click the HAProxy template that appears in the panel below, and select New VM from This Template...
4. Provide a simple name (we used `HAProxy`), and select the Datacenter and/or folder you want to deploy to. Click Next.
5. Select the cluster or compute resource where you want to deploy the HAProxy VM, and click Next.
6. Review details, and click Next.
7. Check the box to accept all license agreements, and click Next.
8. Accept the default configuration, and click Next.
9. Select the target storage for the VM, and click Next.
10. Select VM Network for the Management network, and choose a network for the workload network. Choose the same network for the Frontend network, and click Next.
11. Customize the template using the following:
 - Appliance configuration section
 - For the root password, we used Password1!
 - Check the box for Permit Root Login.
 - Leave the TLS CA blank.
 - Network configuration section
 - We left the default, `haproxy.local`.
 - Configure a local DNS server.
 - For management IP, we used `10.209.201.200/16`.
 - For management gateway, we used `10.209.0.1`.
 - NOTE: The description asks for the workload network gateway address. You should enter the management gateway address instead.
 - For Workload IP, we used `192.168.1.2/24`.
 - For Workload gateway, we used `192.168.1.1`.
 - Load Balancing section
 - For load balancer IP ranges, we used `192.168.1.240/29`.
 - Accept the default management port.
 - For HAProxy User ID, we used `admin`.
 - For the HAProxy password, we used Password1!
12. Click Next.
13. Review the summary, and click Finish.
The deployment will take a few minutes to completely deploy and configure.
14. Power on the HAProxy VM.

Adding licenses

1. Select Menu → Administration.
2. In the left pane, select Licenses.
3. In Licenses, click Add New Licenses.
4. In Enter license keys, enter the license keys for your deployment, and click Next.
5. In Edit license names, rename the licenses if desired, and click Next.
6. In Ready to complete, click Finish.

Configuring Workload Management

Configure the vSphere environment for Kubernetes.

1. From the vSphere client, click Menu → Workload Management.
2. Click Get Started.
3. Review the messages and warnings regarding supported configurations, and click Next.
4. Select the Cluster on which you want to enable workload management, and click Next.
5. Choose the capacity for the control plane VMs (we chose Small). Click Next.
6. Choose the storage policy you wish to use for the control plane nodes (we chose tkg-clusters). Click Next.
7. Configure the Load Balancer section with the following:
 - Name: `haproxy`
 - Type: `HA proxy`
 - Data plane API Addresses: `10.209.201.200:5556`
 - User name: `admin`
 - Password: `Password1!`
 - IP Address Ranges for Virtual Servers: `192.168.1.240-192.168.1.247`
 - Server Certificate Authority: [copy and pasted from the instructions in step 10 below]
8. Open an SSH session to the HAProxy management address and connect using root and Password1!
9. Type the following: `cat /etc/haproxy/ca.crt`
10. Copy the entire output (including the first and last lines) and paste the contents into the Server Certificate Authority box) in the last line under step 7 above.
11. Close the SSH session.
12. Click Next.
13. Configure Workload Management with the following:
 - Network: `VM Network`
 - Starting IP Address: `10.209.201.201`
 - Subnet Mask: `255.255.0.0`
 - Gateway: `10.209.0.1`
 - DNS Server: `10.41.0.10`
 - NTP Server: `10.40.0.1`
14. Click Next.
15. Configure Workload Network with the following:
 - Leave the default for Services addresses.
 - DNS Servers: `10.41.0.10`
 - Under Workload Network, click Add.
 - Accept default for network-1.
 - Port Group: `Workload Network`
 - Gateway: `192.168.1.1`
 - Subnet: `255.255.255.0`
 - IP Address Ranges: `192.168.1.65-192.168.1.126`
16. Click Save.
17. For TKG Configuration, beside Add Content Library, click Add.
18. Select the TKG-cl library, and click OK.
19. Click Next.
20. Click Finish.

The workload management cluster will deploy and configure. You may see apparent errors during configuration, but vCenter will resolve them upon successful completion.

Configuring namespaces

Configure the Kubernetes namespace for services deployment.

1. In Workload Management, click Namespaces.
2. Click Create Namespace.
3. Select the target cluster, and provide a name. We used `tanzu-ns`. Click Create.
4. The new namespace is created. Click the Permissions tab, and click Add.
5. Choose vSphere.local for the identity source. Search for the DevOps user you created. Select the “can edit” role. Click OK.
6. Click the Storage tab.
7. In the Storage Policies section, click Edit.
8. Select the tkg-clusters policy, and click OK. The environment is ready for connection and deployment of containers.

Deploying Red Hat OpenShift 4.7

Configuring DNS

1. Log into your domain controller.
2. Press the Start key, type DNS and press Return.
3. In DNS manager, navigate to your domain’s forward lookup zone.
4. Right-click your forward lookup zone, and select New Host (A or AAAA).
5. In the New Host screen, make sure that Create associated pointer (PTR) record is selected, then input the host name and IP address, and click Add Host.
6. Click OK to acknowledge the creation of the host.
7. Repeat steps 5 and 6 for the following hosts:

api.openshift	10.209.60.201
api-int.openshift	10.209.60.201
*.apps.openshift	10.209.60.202
bootstrap.openshift	10.209.60.10
master-0.openshift	10.209.60.11
master-1.openshift	10.209.60.12
master-2.openshift	10.209.60.13
worker-0.openshift	10.209.60.21
worker-1.openshift	10.209.60.22
worker-2.openshift	10.209.60.23
worker-3.openshift	10.209.60.24
worker-4.openshift	10.209.60.25

Installing the Red Hat OpenShift provisioner

We installed our OpenShift installer on a physical host with a network connection to the infrastructure network.

Installing RHEL

1. Connect a RHEL 8.4 installation disk image to the host.
2. Boot the host off the RHEL installation disk.
3. When choosing what to boot, select Install Red Hat Enterprise Linux 8.4.0
4. Once the installer loads, click Next to move past the start screen.
5. In the Installation Summary screen, click Installation Destination.
6. In the Installation Destination screen, accept the default, and click Done.
7. In the Installation Summary screen, click KDUMP.
8. In the KDUMP screen, uncheck Enable kdump, and click Done.
9. In the Installation Summary screen, click Software Selection.
10. Select Minimal Install, and click Done.
11. In the Installation Summary screen, click Network & Host Name.
12. In Network & Host Name, choose your hostname, turn your Ethernet connection on, and click Configure.
13. In Editing [your Ethernet connection], click Connect automatically with priority: 0, and click Save.
14. Click Done.
15. In the Installation Summary screen, click Begin Installation.
16. In the Configuration screen, click Root Password.
17. In Root Password, type your root password, and click Done.
18. When the installation is complete, click Reboot to reboot into the OS.

Creating an SSH private key

1. Log into the OpenShift installer
2. Create a non-root user account with root access for OpenShift:

```
useradd kni
passwd kni
echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
chmod 0440 /etc/sudoers.d/kni
```
3. To create an SSH key for your new user that you will use for communication between your cluster hosts, type the following command:

```
su - kni -c "ssh-keygen -t ed25519 -f /home/kni/.ssh/id_rsa -N ''"
```

When configuring your OpenShift `install_config.yaml` file, you will need to use the information in `/home/kni/.ssh/id_rsa.pub` to fill in the `sshKey` value.

Installing the OpenShift installer

1. Log into the kni user on the OpenShift provisioner.
2. Disable the firewall service:

```
sudo systemctl stop firewalld
sudo systemctl disable firewalld
```
3. Register your RHEL installation to Red Hat to allow your system to install updates.
4. Install the prerequisites for the OpenShift installation:

```
sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool
```
5. Modify and start the libvirtd service:

```
sudo usermod --append --groups libvirt kni
sudo systemctl enable libvirtd --now
```
6. Create the default storage pool:

```
sudo virsh pool-define-as --name default --type dir --target /var/lib/libvirt/images
sudo virsh pool-start default
sudo virsh pool-autostart default
```
7. Copy the Pull Secret file to the OpenShift install machine.

8. Set the environmental variables that will grab the OpenShift install files:

```
export VERSION=latest-4.7
export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/
  release.txt | grep 'Pull From: quay.io' | awk -F ' ' '{print $3}')
export cmd=openshift-baremetal-install
export pullsecret_file=~/.pull-secret.txt
export pullsecret_file=~/.pull-secret
export extract_dir=$(pwd)
```

9. Download and unpack the OpenShift file:

```
curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/openshift-client-linux.
  tar.gz | tar zxvf - oc
sudo cp oc /usr/local/bin
```

10. Using the OpenShift executable, download and extract the OpenShift installer, then move it to the bin directory:

```
oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to "${extract_dir}"
  ${RELEASE_IMAGE}
sudo cp openshift-baremetal-install /usr/local/bin
```

Running an OpenShift install using installer-provided infrastructure

1. Create an install-config.yaml for your testbed. We have provided an example below:

```
apiVersion: v1
baseDomain: [the domain you're using]
metadata:
  name: openshift
networking:
  machineCIDR: [a routable CIDR pool that can reach the Internet]
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 5
controlPlane:
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    apiVIP: 10.209.60.201
    ingressVIP: 10.209.60.202
    provisioningNetwork: "Disabled"
    bootstrapProvisioningIP: 10.209.60.10
    hosts:
      - name: master-0
        role: master
        bmc:
          address: redfish-virtualmedia://10.209.101.11/redfish/v1/Systems/1
          username: Administrator
          password: Password1
          disableCertificateVerification: True
          bootMACAddress: [the MAC address of the NIC you're planning to use]
hardwareProfile: default
  rootDeviceHints:
    model: "LOGICAL"
    minSizeGigabytes: 300
  - name: master-1
    role: master
```

```

bmc:
  address: redfish-virtualmedia://10.209.101.12/redfish/v1/Systems/1
  username: Administrator
  password: Password1
  disableCertificateVerification: True
bootMACAddress: [the MAC address of the NIC you're planning to use]
hardwareProfile: default
rootDeviceHints:
  model: "LOGICAL"
  minSizeGigabytes: 300
- name: master-2
  role: master
  bmc:
    address: redfish-virtualmedia://10.209.101.13/redfish/v1/Systems/1
    username: Administrator
    password: Password1
    disableCertificateVerification: True
bootMACAddress: [the MAC address of the NIC you're planning to use]
hardwareProfile: default
rootDeviceHints:
  model: "LOGICAL"
  minSizeGigabytes: 300
- name: worker-0
  role: worker
  bmc:
    address: redfish-virtualmedia://10.209.101.21/redfish/v1/Systems/1
    username: Administrator
    password: Password1
    disableCertificateVerification: True
bootMACAddress: [the MAC address of the NIC you're planning to use]
rootDeviceHints:
  model: "LOGICAL"
  minSizeGigabytes: 300
- name: worker-1
  role: worker
  bmc:
    address: redfish-virtualmedia://10.209.101.22/redfish/v1/Systems/1
    username: Administrator
    password: Password1
    disableCertificateVerification: True
bootMACAddress: [the MAC address of the NIC you're planning to use]
rootDeviceHints:
  model: "LOGICAL"
  minSizeGigabytes: 300
- name: worker-2
  role: worker
  bmc:
    address: redfish-virtualmedia://10.209.101.23/redfish/v1/Systems/1
    username: Administrator
    password: Password1
    disableCertificateVerification: True
bootMACAddress: [the MAC address of the NIC you're planning to use]
rootDeviceHints:
  model: "LOGICAL"
  minSizeGigabytes: 300
- name: worker-3

```

```

    role: worker
    bmc:
      address: redfish-virtualmedia://10.209.101.24/redfish/v1/Systems/1
      username: Administrator
      password: Password1
      disableCertificateVerification: True
    bootMACAddress: [the MAC address of the NIC you're planning to use]
    rootDeviceHints:
      model: "LOGICAL"
      minSizeGigabytes: 300
- name: worker-4
  role: worker
  bmc:
    address: redfish-virtualmedia://10.209.101.25/redfish/v1/Systems/1
    username: Administrator
    password: Password1
    disableCertificateVerification: True
  bootMACAddress: [the MAC address of the NIC you're planning to use]
  rootDeviceHints:
    model: "LOGICAL"
    minSizeGigabytes: 300
pullSecret: '[your pull secret text]'
sshKey: '[your public SSH key]'

```

2. Create an install directory:

```
mkdir clusterconfig
```

3. Copy install-config.yaml into your install directory:

```
cp install-config.yaml clusterconfig/
```

4. Run the following command to generate the Kubernetes manifests for your OpenShift cluster:

```
openshift-baremetal-install --dir=clusterconfig create manifests
```

5. Create the Ignition files:

```
openshift-baremetal-install --dir=clusterconfig create cluster
```

Creating the Local Storage operator

1. Log into your OpenShift Container Platform cluster.
2. Click Operators → OperatorHub.
3. In the search box, type `Local Storage`
4. When the Local Storage operator appears, click it.
5. In the right side of the screen, click Install.
6. In the Create Operator Subscription screen, accept all defaults, and click Subscribe.
The installation of the operator is complete when the Installed Operators status page lists Local Storage as Succeeded.

Creating the OpenShift Container Storage operator

1. Click Operators → OperatorHub.
2. In the search box, type `OpenShift Container Storage`
3. When the OpenShift Container Storage operator appears, click it.
4. In the right side of the screen, click Install.
5. In the Create Operator Subscription screen, accept all defaults, and click Subscribe.

Creating an OpenShift Container Storage cluster

1. Click Operators → Installed Operators.
2. Click on OpenShift Container Storage.
3. In OpenShift Container Storage, under the Storage Cluster option, select Create Instance.
4. In the Create Instance screen, make sure Internal-Attached devices is selected, select All Nodes, and click Next.
5. In the Create Storage Class screen, enter the Volume Set Name, and click Next.
6. When a pop-up warns you that this cannot be undone, click Yes to continue.
7. In the Set Storage and nodes screen, wait for the Storage Class to finish creating, and click Next.
8. In the Security configuration screen, leave Enable encryption unchecked, and click Next.
9. In the Review screen, click Create to kick off the OCS cluster creation.

Installing and running the pod-density test with VMware vSphere with Tanzu

1. Deploy the Tanzu Kubernetes Grid cluster from your Workload Management namespace. The specification file we used, new-tkg-cluster.yaml, is in the Files we used in our testing section of this document.

```
kubectl apply -f new-tkg-cluster.yaml
```

2. After the cluster is created (it may take some time), log into the cluster by first logging out of the Workload Management namespace context, and logging into the context of the new cluster you created.

```
kubectl vsphere logout
```

```
kubectl vsphere login --insecure-skip-tls-verify --vsphere-username devops@vsphere.local  
--server=https://10.209.201.201 --tanzu-kubernetes-cluster-namespace tkg-workload --tanzu-  
kubernetes-cluster-name tkg-cluster-01
```

3. Deploy the application, which is based on this example: <https://github.com/paulbouwer/hello-kubernetes>. The specification file, hello-kubernetes.yaml, is in the Files we used in our testing section of this document.

```
kubectl apply -f hello-kubernetes.yaml
```

4. To set the number of pods, scale the replicas. To limit the stress on cluster resources, we added replicas in batches of 100 until we reached the final number. For example, to scale to 300 pods, we added pods in three steps:

```
kubectl scale deployment hello-kubernetes --replicas=100  
# wait until all pods have started  
kubectl scale deployment hello-kubernetes --replicas=200  
# wait until all pods have started  
kubectl scale deployment hello-kubernetes --replicas=300  
# wait until all pods have started
```

5. Get the external IP address for the service:

```
kubectl get svc hello-kubernetes
```

6. To perform the test, execute this script that sends an unending set of queries to the load balancer's IP address, which you found in step 5. For example, if the IP address was 10.233.59.17, the script would be:

```
while : ; do  
  curl -s --connect-timeout 10 http://10.233.59.17 -w \  
  '\n%{time_total} hello-kubernetes- 0000\n'  
done | grep hello-kubernetes- | tee pod-density-results.txt
```

7. To end the test, type Control+C. The results file contains the name of the pod that received and responded to the query and the total time needed to complete the request.

Installing and running the pod-density test with Red Hat OpenShift

1. Deploy the application, which is based on this example: <https://github.com/paulbouwer/hello-kubernetes>. The specification file, hello-kubernetes.yaml, is in the Files we used in our testing section of this document.

```
oc apply -f hello-kubernetes.yaml
```

2. To set the number of pods, scale the replicas. To limit the stress on cluster resources, we added replicas in batches of 100 until we reached the final number. For example, to scale to 300 pods, we added pods in three steps:

```
oc scale deployment hello-kubernetes --replicas=100 -n hello
# wait until all pods have started
oc scale deployment hello-kubernetes --replicas=200 -n hello
# wait until all pods have started
oc scale deployment hello-kubernetes --replicas=300 -n hello
# wait until all pods have started
```

3. Get the external IP address for the service:

```
kubectl get svc hello-kubernetes -n hello
```

4. To perform the test, execute this script that sends an unending set of queries to the load balancer's IP address, which you found in step 3. For example, if the IP address was 10.233.59.17, the script would be:

```
while : ; do
  curl -s --connect-timeout 10 http://10.233.59.17 -w \
    '\n%{time_total} hello-kubernetes- 0000\n'
done | grep hello-kubernetes- | tee pod-density-results.txt
```

5. To end the test, type Control+C. The results file contains the name of the pod that received and responded to the query and the total time needed to complete the request.

Files we used in our testing

new-tkg-cluster.yaml

We used the new-tkg-cluster.yaml file only for the VMware vSphere with Tanzu environment.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkg-cluster-01
  namespace: tkg-workload
spec:
  distribution:
    version: 1.19.7+vmware.1-tkg.1.fc82c41
  topology:
    controlPlane:
      count: 3
      class: guaranteed-medium
      storageClass: tkg-storage-policy
    workers:
      count: 130
      class: best-effort-medium
      storageClass: tkg-storage-policy
  settings:
    storage:
      defaultClass: tkg-storage-policy
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["172.17.0.0/16"]
      pods:
        cidrBlocks: ["172.18.0.0/16"]
```

hello-kubernetes.yaml

We used this same hello-kubernetes.yaml file for both environments for identical workloads.

```
apiVersion: v1
kind: Service
metadata:
  name: hello-kubernetes
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: hello-kubernetes
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-kubernetes
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-kubernetes
  template:
    metadata:
      labels:
        app: hello-kubernetes
    spec:
      containers:
      - name: hello-kubernetes
        image: paulbouwer/hello-kubernetes:1.9
        ports:
        - containerPort: 8080
```

This project was commissioned by VMware.



Facts matter.®

Principled Technologies is a registered trademark of Principled Technologies, Inc. All other product names are the trademarks of their respective owners.

DISCLAIMER OF WARRANTIES; LIMITATION OF LIABILITY:

Principled Technologies, Inc. has made reasonable efforts to ensure the accuracy and validity of its testing, however, Principled Technologies, Inc. specifically disclaims any warranty, expressed or implied, relating to the test results and analysis, their accuracy, completeness or quality, including any implied warranty of fitness for any particular purpose. All persons or entities relying on the results of any testing do so at their own risk, and agree that Principled Technologies, Inc., its employees and its subcontractors shall have no liability whatsoever from any claim of loss or damage on account of any alleged error or defect in any testing procedure or result.

In no event shall Principled Technologies, Inc. be liable for indirect, special, incidental, or consequential damages in connection with its testing, even if advised of the possibility of such damages. In no event shall Principled Technologies, Inc.'s liability, including for direct damages, exceed the amounts paid in connection with Principled Technologies, Inc.'s testing. Customer's sole and exclusive remedies are as set forth herein.