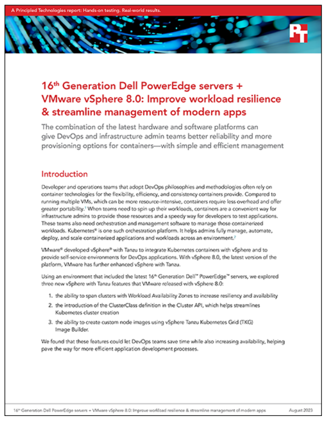

A Principled Technologies report: Hands-on testing. Real-world results.

16th Generation Dell PowerEdge servers + VMware vSphere 8.0: Improve workload resilience & streamline management of modern apps

The combination of the latest hardware and software platforms can give DevOps and infrastructure admin teams better reliability and more provisioning options for containers—with simple and efficient management

Introduction

Developer and operations teams that adopt DevOps philosophies and methodologies often rely on container technologies for the flexibility, efficiency, and consistency containers provide. Compared to running multiple VMs, which can be more resource-intensive, containers require less overhead and offer greater portability.[1] When teams need to spin up their workloads, containers are a convenient way for infrastructure admins to provide those resources and a speedy way for developers to test applications. These teams also need orchestration and management software to manage those containerized workloads. Kubernetes® is one such orchestration platform. It helps admins fully manage, automate, deploy, and scale containerized applications and workloads across an environment.[2]

VMware® developed vSphere® with Tanzu to integrate Kubernetes containers with vSphere and to provide self-service environments for DevOps applications. With vSphere 8.0, the latest version of the platform, VMware has further enhanced vSphere with Tanzu.

Using an environment that included the latest 16th Generation Dell™ PowerEdge™ servers, we explored three new vSphere with Tanzu features that VMware released with vSphere 8.0:

1. the ability to span clusters with Workload Availability Zones to increase resiliency and availability
2. the introduction of the ClusterClass definition in the Cluster API, which helps streamlines Kubernetes cluster creation
3. the ability to create custom node images using vSphere Tanzu Kubernetes Grid (TKG) Image Builder.

We found that these features could let DevOps teams save time while also increasing availability, helping pave the way for more efficient application development processes.

16th Generation Dell PowerEdge servers + VMware vSphere 8.0: Improve workload resilience & streamline management of modern apps August 2023

The science behind the report:

16th Generation Dell PowerEdge servers + VMware vSphere 8.0: Improve workload resilience & streamline management of modern apps

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report 16th Generation Dell PowerEdge servers + VMware vSphere 8.0: Improve workload resilience & streamline management of modern apps.

We concluded our hands-on testing on June 8, 2023. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on April 28, 2023 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

System configuration information

Table 1: Detailed information on the systems we tested.

System configuration information	3x Dell™ PowerEdge™ R6625	3x Dell PowerEdge R6625	3x Dell PowerEdge R7525
BIOS name and version	Dell 1.1.3	Dell 1.1.3	Dell 2.10.2
Operating system name and version/build number	VMware® ESXi™, 8.0.1, 21495797	VMware ESXi, 8.0.1, 21495797	VMware ESXi, 8.0.1, 21495797
Date of last OS updates/ patches applied	04/28/2023	04/28/2023	04/28/2023
System profile settings	Performance	Performance	Performance
Processor			
Number of processors	2	2	2
Vendor and model	AMD EPYC™ 9554	AMD EPYC 9554	AMD EPYC 7513
Core count (per processor)	64	64	32
Core frequency (GHz)	3.1	3.1	2.6
Stepping	Model 17 Stepping 1	Model 17 Stepping 1	Model 1 Stepping 1

System configuration information	3x Dell™ PowerEdge™ R6625	3x Dell PowerEdge R6625	3x Dell PowerEdge R7525
Memory module(s)			
Total memory in system (GB)	128	128	512
Number of memory modules	8	8	16
Vendor and model	Hynix HMCG78MEBRA107N	Hynix HMCG78MEBRA107N	Hynix HMAA4GR7CJR8N-XN
Size (GB)	16	16	32
Type	PC5-38400	PC5-38400	PC4-25600R
Speed (MHz)	4,800	4,800	3,200
Speed running in the server (MHz)	4,800	4,800	3,200
Storage controller			
Vendor and model	PERC H755 Front	PERC H755 Front	N/A
Cache size (GB)	8	8	N/A
Firmware version	52.14.0-3901	52.14.0-3901	N/A
Driver version	7.722.02.00	7.722.02.00	N/A
Local storage (VMware vSAN™)			
Number of drives	4	4	4
Drive vendor and model	Dell Ent NVMe® v2 AGN MU U.2	Dell Ent NVMe v2 AGN MU U.2	Dell Ent NVMe v2 AGN MU U.2
Drive size (GB)	3,200	6,400	3,200
Drive information (speed, interface, type)	NVMe Gen4, SSD, 2.5	NVMe Gen4, SSD, 2.5	NVMe Gen3, SSD, 2.5
Local storage (OS)			
Number of drives	2	2	2
Drive vendor and model	Seagate ST600MM0069	Seagate ST600MM0069 / Toshiba AL15SEB060NY	Micron MTFDDAV240TDU / Intel SSDSCKKB240G8R
Drive size (GB)	600	600	240
Drive information (speed, interface, type)	12Gbps, SAS, HDD, 2.5	12Gbps, SAS, HDD, 2.5	6Gbps, SATA, SSD, 2.5
Network adapter			
Vendor and model	Broadcom® BCM57508	Broadcom BCM57508	Broadcom Adv. Dual 25Gb Ethernet
Number and type of ports	2x 100Gb	2x 100Gb	2x 25Gb
Driver version	21.85.21.91	21.85.21.91	21.80.16.92
Cooling fans			
Vendor and model	Dell Gold	Dell Gold	Dell Standard / Gold
Number of cooling fans	16	16	6 / 12
Power supplies			
Vendor and model	Dell, 1400W, RDNT, ARTESYN	Dell, 1400W, RDNT, ARTESYN	Dell, 1400W, RDNT, ARTESYN
Number of power supplies	2	2	1 / 2
Wattage of each (W)	1,400	1,400	1,400

How we tested

Setting up the VMware vCenter environment

We installed the VMware vSphere® 8.0.1 release of the Dell-customized VMware ESXi image on six Dell PowerEdge R6625 servers and three Dell PowerEdge R7525 servers We deployed the latest version of VMware vCenter® and pfSense on a separate infrastructure server. We added all the servers to the vCenter in a vSphere Datacenter named tkg-datacenter. We created three vSAN-enabled vSphere clusters: two clusters comprising three PowerEdge R6625 servers each and one cluster comprising three PowerEdge R7525 servers.

In vCenter, we configured VMware vSphere Distributed Switch™ networking for vSphere with Tanzu and deployed and configured HAProxy on the infrastructure server to act as a load balancer for the vSphere with Tanzu Supervisor. We then deployed a zonal Supervisor across three vSphere Zones corresponding to the three host clusters using Workload Management. The following sections describe the steps we took to configure the test environment and run the test.

Configuring infrastructure host networking

Configure networking on the infrastructure host.

1. Log into the infrastructure host, and under the Navigator menu, click Networking.
2. In the right panel, click Virtual switches.
3. Click Add standard virtual switch.
4. Provide a name (such as "private") for the vSwitch. Set the appropriate MTU for the upstream switch, and select the 25Gb uplink. Click Add.
5. Click VMkernel NICs.
6. Click Add VMkernel NIC.
7. Leave the selection for New port group. Provide a name for the port group (we used "management"), select the vSwitch ("private") you created in step 4, set the VLAN ID (we used VLAN 1000), and set the appropriate MTU for the upstream switch.
8. Change DHCP to Static, and expand the IPv4 settings. Provide a static IP address and Subnet mask.
9. Check the box for Management, and click Create.
10. Click Port groups.
11. Click Add port group.
12. Provide a name for the new port group (we used "management-vm"), set the VLAN ID (we used VLAN 1000), and select the vSwitch (private) you created step 4. Click Add.
13. Click Add port group.
14. Provide a name for the new port group (we used "workload"), set the VLAN ID (we used VLAN 4000), and select the vSwitch ("private") you created in step 4. Click Add.

Deploying and configuring pfSense virtual appliance for the private network

1. Download the most recent release of pfSense from: https://www.pfSense.org/download/.
2. Extract the pfSense ISO.
3. In the vSphere client, from the top of the left panel, select Hosts and Clusters.
4. Right-click your infrastructure cluster or host, 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 infrastructure cluster or host, and click Next.
8. In Select storage, select your target 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: public
    - Network 2: private
12. Click Next, and click Complete.
13. Right-click your newly created VM, and select Power On.
14. Right-click the VM you turned on, and select Open Remote Console.

15. Select Removable Devices → CD/DVD drive 1 → Connect to Disk Image File (iso).
16. Select the pfSense ISO you downloaded.
17. After the VM boots off of the pfSense ISO, accept the license agreement.
18. Select Install.
19. Select Continue with the default keymap.
20. Select Auto (UFS) BIOS.
21. Select No.
22. Select Reboot.
23. When prompted to set up VLANs, select no.
24. For the WAN interface, select vmx0.
25. For the LAN interface, select vmx1.
26. To proceed, press Y.
27. In a web browser, navigate to the vmx1 IP address of the pfSense VM.
28. Select to ignore the security warning.
29. Sign into the console (default username admin/pfSense)
30. On the welcome screen, click Next.
31. Click Next.
32. Enter your DNS server, and click Next.
33. Enter your timezone, and click Next.
34. In Configure WAN interface, leave defaults, and click Next.
35. In Configure LAN interface, use the pull-down menu to select Static IPv4.
36. Under Static IPv4 Configuration, enter the address you want to use as the gateway address for the workload subnet (we used 172.16.0.1).
37. On the right, in the pull-down menu, select the correct number of bits for the subnet mask (we used /16), and click Next.
38. In Set Admin WebGUI Password, enter the password you want for the admin user, and click Next.
39. To apply the pfSense changes, click Reload.
40. To finish the pfSense installation, click Finish.
41. Log in using the administrative credentials.
42. Select Services → DHCP Server.
43. Check the box for Enable DHCP Server on LAN interface, and enter the range of IP addresses for use in the DHCP scope.
44. In the DNS Servers entry, enter the DNS server provided on the public side interface.
45. At the bottom of the page, click Save.
46. To restart the service, click the icon in the top right.

Installing the vCenter Server Appliance (VCSA)

1. Download the VMware vCenter from the VMware support portal: 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. When 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/.
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. For the Single Sign-On domain name, enter `tkg-test.lan`. Enter a password for the administrator account, confirm it, and click Next.
21. Click Next.
22. Click Finish.

Configuring a vSAN-enabled cluster through Quickstart

1. Select Menu → Hosts and Clusters.
2. Click the cluster you want to configure.
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: 2000
    - Protocol: IPv4
    - IP Type: Static
    - Host #1 IP: 192.168.2.x
    - Host #1 subnet: 255.255.255.0
    - Check the autofill checkbox
6. In Storage traffic, complete the fields as follows:
    - Use VLAN: 3000
    - Protocol: IPv4
    - IP Type: Static
    - Host #1 IP: 192.168.3.x
    - 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.
11. Repeat steps 2 through 10 for each cluster you want to configure.

Adding the Tanzu Kubernetes Grid workload network port group to the distributed switch

1. Right-click the DvSwitch the system created during the cluster Quickstart wizard, and select Distributed Port Group → New Distributed Port Group.
2. Name it "Workload Network," and click Next.
3. Change the VLAN type to VLAN, set the VLAN ID to VLAN 4000, and click Next.
4. Click Finish.

Deploying and configuring pfSense virtual appliance for workload network

1. In the vSphere client, in the left pane, select Hosts and Clusters.
2. Right-click your infrastructure cluster or host, and select New Virtual Machine.
3. In Select a creation type, select Create a new virtual machine, and click Next.
4. In Select a name and folder, name your pfSense VM, and click Next.
5. In Select a compute resource, select your infrastructure cluster or host, and click Next.
6. In Select storage, select your target datastore, and click Next.
7. In Select compatibility, accept defaults, and click Next.
8. 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.
9. In Customize hardware, change the following:
    - CPU: 2
    - Memory: 8 GB
    - Hard Disk: 32 GB
    - SCSI controller: LSI Logic Parallel
    - Network 1: private
    - Network 2: workload
10. Click Next, and click Complete.
11. Right-click your newly created VM, and select Power On.

12. Right-click the VM you turned on, and select Open Remote Console.
13. Select Removable Devices → CD/DVD drive 1 → Connect to Disk Image File (iso).
14. Select the pfSense ISO you downloaded.
15. After the VM boots from the pfSense ISO, accept the license agreement.
16. Select Install.
17. Select Continue with the default keymap.
18. Select Auto (UFS) BIOS.
19. Select No.
20. Select Reboot.
21. When prompted to set up VLANs, select no.
22. For the WAN interface, select private.
23. For the LAN interface, select workload.
24. To proceed, press Y.
25. In a web browser, navigate to the IP address of the pfSense VM.
26. Select to ignore the security warning.
27. Sign into the console (default username admin/pfSense).
28. In the welcome screen, click Next.
29. Click Next.
30. Enter your DNS server, and click Next.
31. Enter your timezone, and click Next.
32. In Configure WAN interface, leave defaults, and click Next.
33. In Configure LAN interface, use the pull-down menu to select Static IPv4.
34. Under Static IPv4 Configuration, enter the address you want to use as the gateway address for the workload subnet (we used 192.16.0.1).
35. In the right pull-down menu, select the correct number of bits for the subnet mask (we used /24), and click Next.
36. In Set Admin WebGUI Password, enter the password you want for the admin user, and click Next.
37. To apply the pfSense changes, click Reload.
38. To finish the pfSense installation, click Finish.
39. Log in using the administrative credentials.
40. Select Services → DHCP Server.
41. Check the box for Enable DHCP Server on LAN interface, and enter the range of IP Addresses for use in the DHCP scope.
42. In the DNS Servers entry, enter the DNS server provided on the public side interface, and at the bottom of the page, click Save.
43. To restart the service, in the top right, click the icon.

Creating vSphere zones

Create a high-availability zone to protect against cluster-level failure.

1. Log into vCenter, and from the Home drop-down menu, click Infrastructure.
2. In the left panel, select the vCenter.
3. In the right panel, click Configure.
4. In the list, select vSphere Zones.
5. Click Add New vSphere Zone.
6. Provide a name for the vSphere zone. Click Next.
7. Select from the list of available vSphere clusters that have not been added to any other zone. Click Finish.
8. Repeat steps 1 through 7 until all vSphere clusters are in a zone.

Creating a DevOps user

Create a dedicated vSphere user account to access Kubernetes namespaces.

1. Log into vCenter, and from the Home drop-down menu, click Administration.
2. In the left panel, click Users and Groups.
3. In the right panel, select tkg-test.lan from the Domain drop-down.
4. Click Add and provide a username and password, and click Add.
5. Click Groups and select the Administrators group, and 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

1. Download the vSphere compatible HAProxy OVF from https://github.com/haproxytech/vmware-haproxy.
2. Log into vCenter, and in the Home drop-down menu, click Content Libraries.
3. In the right panel, click Create.
4. Name the content library "HAproxy-cl," and click Next.
5. Accept the default, and click Next.
6. Click Next.
7. Choose the storage location, and click Next.
8. Review, and click Finish.
9. In the left pane, click the newly created HAproxy-cl content library.
10. Click Actions, and select Import Item.
11. Change the selection to local file, and click Upload Files.
12. Browse to the location of the HAProxy OVF you downloaded in step 1, and click Open.
13. Click Import.

Deploying and configuring HAProxy virtual appliance

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. 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 for I 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 172.16.0.2/16.
        - For management gateway, we used 172.16.0.1.
        - NOTE: The description asks for the workload network gateway address. You should enter the management gateway address instead.
        - For Workload IP, we used 192.168.0.2/24.
        - For Workload gateway, we used 192.168.0.1.
    - Load Balancing Section
        - For load balancer IP ranges, we used 192.168.0.240/29.
        - 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.

Creating the subscribed TKG content library

1. Log into vCenter, and in the Home drop-down menu, click Content Libraries.
2. In the right panel, 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 as the subscription URL. Click Next.
5. Click Yes to verify.
6. Choose the storage location, and click Next.
7. Review, and click Finish.

Creating the local TKG content library

1. Log into vCenter, and in the Home drop-down menu, click Content Libraries.
2. In the right panel, click Create.
3. Name the content library "TKR-local," and click next.
4. Select Local content library. Click Next.
5. Click Next.
6. Choose the storage location, and click Next.
7. Review, and click Finish.

Creating a virtual machine storage tag

1. Log into vCenter, and in the left pane, click Storage.
2. From the left pane, select the storage you want to tag for Tanzu (in this case, one of the clusters' vSAN storage).
3. Under the Summary pane on the right, under the Tags panel, click Assign.
4. Click Add Tag.
5. Name the tag "Tanzu."
6. Click Create New Category.
7. Name the category "Tanzu Storage." Clear all object types except Datastore, and click Create.
8. Use the category drop-down menu to select Tanzu Storage, and click Create.
9. Check the box beside the newly created tag, and click Assign.
10. Navigate to another of the untagged vSAN cluster datastores, and assign the storage tag to it by clicking Assign, selecting the Tanzu tag, and clicking Assign.
11. Repeat step 10 for the third vSAN cluster datastore.

Creating a VM storage policy

1. Log into vCenter, and from the Home drop-down menu, click Policies and Profiles.
2. In the left panel, click VM Storage Policies.
3. Click Create.
4. Create a new VM storage policy named "tkg-storage," and click Next.
5. Check the box for Enable tag based placement rules, and click Next.
6. Use the Tag Category drop-down to select the Tanzu Storage policy you created previously. Click Browse Tags.
7. Click the Tanzu checkbox, and click OK.
8. Click Next twice.
9. Review the compatible storage to make sure your storage target is compatible, and click Next.
10. Click Finish.

Configuring Workload Management and Deploy Supervisors

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 you want to enable Workload Management on, 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: 172.16.0.2:5556
    - User name: admin
    - Password: Password1!
    - IP Address Ranges for Virtual Servers: 192.168.0.240-192.168.0.247
    - Server Certificate Authority: [copy and paste from the instructions in steps 8 through 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 step 7 above.
11. Close the SSH session.
12. Click Next.
13. Configure Workload Management with the following:
    - Network: DSwitch-PRIV
    - Starting IP Address: 172.16.0.201
    - Subnet Mask: 255.255.0.0
    - Gateway: 172.16.0.1
    - DNS Server: [use the DNS provided by the public side adapter on the private pfSense appliance]
    - NTP Server: 0.us.pool.ntp.org
14. Click Next.
15. Configure Workload Network with the following:
    - Leave the default for Services addresses.
    - DNS Servers: [use the DNS provided by the public side adapter on the private pfSense appliance]
    - Under Workload Network, click Add.
    - Accept default for network-1.
    - Port Group: Workload Network.
    - Gateway: 192.168.0.1.
    - Subnet: 255.255.255.0
    - IP Address Ranges: 192.168.0.65-192.168.0.200
16. Click Save.
17. For TKG Configuration, use the following:
    - Beside Add Content Library, click Add.
    - Select the TKG-cl library, and click OK.
    - Click Next.
    - Click Finish. The workload management cluster will deploy and configure. You may see apparent errors during configuration—these will resolve upon successful completion.

Creating a namespace for Kubernetes

1. Log into vCenter, and from the Home drop-down menu, click Workload Management.
2. Click Namespaces, and click New Namespace.
3. Select the supervisor you previously deployed, and provide a name. We used "tanzu-ns."
4. Click Create.
5. Click the Permissions tab, and click Add.
6. Select tkg-test.lan as the identity source, and search for the DevOps user you created previously.
7. Select the can edit role, and click OK.
8. Click the Storage tab.
9. In the Storage Policies section, click Edit.
10. Select the tkg-storage policy, and click OK.
11. Click the Summary tab.
12. Under VM Service, click Manage VM Classes.
13. Check the boxes beside the desired VM classes to make them available to the namespace.
14. Click OK.
15. Click Manage Content Libraries.
16. Check the boxes beside the TKG-cl and TKR-local content libraries you created previously.
17. Click OK.

Creating a bootstrap Ubuntu 22.04 VM for vSphere with Tanzu

1. Log into vCenter, and in the left pane, click Storage.
2. Select datastore1, and click Files.
3. Click Upload Files, and upload the Ubuntu 22.04 ISO.
4. From the drop-down menu, click Inventory.
5. Right-click a cluster, and click New Virtual Machine.
6. Click Next.
7. Enter a name for the VM, and click Next.
8. Click Next.
9. Select the cluster's vSAN storage, and click Next.
10. Click Next.
11. From the Guest OS Family drop-down, select Linux. From the Guest OS drop-down, select Ubuntu Linux (64-bit).
12. Click Next.
13. Assign the VM 2 vCPUs, 16GB of memory, and a 40GB hard disk.
14. From the New CD/DVD Drive drop-down, select Datastore ISO File, and select the Ubuntu ISO you uploaded previously. Ensure Connect At Power On is checked.
15. Click Add New Device, click Network Adapter, and from the drop-down, select the workload network. Click Next.
16. Click Finish.
17. Power on the VM, and click Launch Remote Console.
18. Click Install Ubuntu.
19. Click Continue twice.
20. Select Minimal Installation, and click Continue.
21. Click Install Now.
22. Click Continue.
23. Select the appropriate time zone by clicking the map, and click Continue.
24. Fill out the name, computer name, username, and password fields, and click Continue.
25. After the install process completes, click Restart Now.
26. Press Enter.
27. Enter your password, and click Sign In.
28. To install updates when prompted by the Software Updater, click Install Now.
29. Click OK.

Installing Docker on the bootstrap VM

We followed the steps at https://docs.docker.com/engine/install/ubuntu/ to install Docker on the Ubuntu boostrap VM:

1. Open a terminal on the VM.
2. Install prerequisite packages:

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
```

3. Add Docker's GPG key:

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/
keyrings/docker.gpg
```

4. Set up the Docker repository:

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://
download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Install Docker:

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Installing kubectl and the vSphere for kubectl plugin

1. Open a browser on the boostrap VM, and navigate to the IP of one of the three SupervisorControlPlane VMs. In our environment, the first control plane VM was at 192.168.101.66.
2. To bypass the certificate warning, click Advanced, and click Accept the Risk and Continue.
3. Click Download CLI Plugin Linux.
4. Open the Files app, and navigate to the Downloads folder.
5. Right-click vsphere-plugin.zip, and select Extract Here.
6. Open a terminal, and navigate to the vsphere-plugin binary you extracted in the previous step:

```
cd Downloads/vsphere-plugin/bin
```

7. Make the kubectl and kubectl-vsphere binaries executable, and add them to PATH:

```
sudo mv kubectl-vsphere /usr/local/bin
sudo mv kubectl /usr/local/bin
chmod +x /usr/local/bin/kubectl-vsphere
chmod +x /usr/local/bin/kubectl
```

Provisioning a workload cluster across three workload availability zones with ClusterClass

1. Open a terminal on the bootstrap VM, and log into the namespace:

```
kubectl vsphere login --server=192.168.0.194 --vsphere-username=devops@tkg-test.lan --tanzu-
kubernetes-cluster-namespace tanzu-ns -insecure-skip-tls-verify
```

2. Ensure the context is set to the Supervisor namespace:

```
kubectl config use-context tanzu-ns
```

3. Prepare a YAML file called cluster-clusterclass-zoned.yaml with the following contents:

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: cluster-clusterclass-zoned
  namespace: tanzu-ns
spec:
  clusterNetwork:
    services:
      cidrBlocks: [“198.51.100.0/12”]
    pods:
      cidrBlocks: [“192.0.2.0/16”]
    serviceDomain: “cluster.local”
  topology:
    class: tanzukubernetescluster
    version: v1.23.8+vmware.2-tkg.2-zshippable
    controlPlane:
      replicas: 3
    workers:
      #muliple node pools are used
      machineDeployments:
        - class: node-pool
          name: node-pool-1
          replicas: 3
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: zone-1
        - class: node-pool
          name: node-pool-2
          replicas: 3
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: zone-2
        - class: node-pool
          name: node-pool-3
          replicas: 3
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: zone-3
    variables:
      - name: vmClass
        value: guaranteed-medium
      - name: storageClass
        value: tkg-storage
```

4. Apply the cluster object to create the cluster:

```
kubectl apply -f cluster-clusterclass-zoned.yaml
```

5. Give the cluster a minute or two to deploy, and verify that it was successfully provisioned:

```
kubectl get clusters
```

Provisioning a workload cluster using Calico CNI with ClusterClass

1. Open a terminal on the bootstrap VM, and log into the namespace:

```
kubectl vsphere login --server=192.168.0.194 --vsphere-username=devops@tkg-test.lan --tanzu-
kubernetes-cluster-namespace tanzu-ns -insecure-skip-tls-verify
```

2. Ensure the context is set to the Supervisor namespace:

```
kubectl config use-context tanzu-ns
```

3. Prepare a YAML file called cluster-calico.yaml with the following contents:

```
---

apiVersion: cni.tanzu.vmware.com/v1alpha1

kind: CalicoConfig

metadata:

  name: cluster-calico

spec:

  calico:

    config:

      vethMTU: 0

---

apiVersion: run.tanzu.vmware.com/v1alpha3

kind: ClusterBootstrap

metadata:

  annotations:

    tkg.tanzu.vmware.com/add-missing-fields-from-tkr: v1.23.8---vmware.2-tkg.2-zshippable

  name: cluster-calico

spec:

  cni:

    refName: calico.tanzu.vmware.com.3.22.1+vmware.1-tkg.2-zshippable

    valuesFrom:

      providerRef:

        apiGroup: cni.tanzu.vmware.com

        kind: CalicoConfig

        name: cluster-calico

---

apiVersion: cluster.x-k8s.io/v1beta1

kind: Cluster

metadata:

  name: cluster-calico
```

```
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: tanzukubernetescluster
    version: v1.23.8---vmware.2-tkg.2-zshippable
    controlPlane:
      replicas: 3
    workers:
      machineDeployments:
        - class: node-pool
          name: node-pool-1
          replicas: 3
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: zone-1
        - class: node-pool
          name: node-pool-2
          replicas: 3
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: zone-2
        - class: node-pool
          name: node-pool-3
          replicas: 3
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: zone-3
    variables:
      - name: vmClass
        value: guaranteed-medium
      - name: storageClass
        value: tkg-storage
```

4. Apply the cluster object to create the cluster:

```
kubectl apply -f cluster-calico.yaml
```

5. Give the cluster a minute or two to deploy, and verify that it was successfully provisioned:

```
kubectl get clusters
```

Deploying NGINX to a workload cluster to test vSphere cluster-level high availability with workload availability zones

1. Open a terminal on the Ubuntu bootstrap VM, and log in to the namespace and cluster context of the cluster-clusterclass-zoned workload cluster you deployed previously:

```
kubectl vsphere login --server=192.168.0.194 --vsphere-username=devops@tkg-test.lan --tanzu-
kubernetes-cluster-name cluster-clusterclass-zoned --tanzu-kubernetes-cluster-namespace tanzu-ns –
insecure-skip-tls-verify
```

2. Ensure the context is set to the workload cluster namespace:

```
kubectl config use-context cluster-clusterclass-zoned
```

3. Add a pod security policy role bind to grant your authenticated user privileges to administer the cluster (the deployment will fail without this):

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --clusterrole=psp:vmware-
system-privileged --group=system:authenticated
```

4. Prepare a YAML file called NGINX-lbsvc.yaml with the following contents:

```
kind: Service
apiVersion: v1
metadata:
  name: srvclb-ngnx
spec:
  selector:
    app: hello
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: loadbalancer
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
        tier: frontend
    spec:
      containers:
      - name: NGINX
        image: "NGINXdemos/hello"
```

5. Apply the object to create the service and deployment:

```
kubectl apply -f NGINX-lbsvc.yaml
```

6. Ensure the LoadBalancer service was deployed successfully and has been assigned an external IP:

```
kubectl get services
```

- Your output should look similar to this:

```
NAME           TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
kubernetes     ClusterIP      198.48.0.1      <none>          443/TCP        24m
srvclb-ngnx    LoadBalancer   198.60.28.198   192.168.0.196   80:30322/TCP   22s
supervisor     ClusterIP      None            <none>          6443/TCP       23m
```

7. Confirm that NGINX is running by navigating to the external IP of the LoadBalancer service in a browser. If successful, you should see an NGINX landing page with the server address, name, and date listed.
8. Confirm that the three replica pods are running in separate node pools:

```
kubectl describe pods | grep Node
```

- Your output should look similar to this:

```
Node:          cluster-clusterclass-zoned-node-pool-2-cdltj-7d986757c9-rj4vm/192.168.0.104
Node-Selectors:               <none>
Node:          cluster-clusterclass-zoned-node-pool-3-gzkhd-755d7f7dcd-sv7qd/192.168.0.107
Node-Selectors:               <none>
Node:          cluster-clusterclass-zoned-node-pool-1-4tp4b-5799c867b5-4r47w/192.168.0.111
Node-Selectors:               <none>
```

9. Log into vCenter, and shut down the three hosts in the first cluster by right-clicking the hosts, and selecting Power → Power Off.
10. Once the hosts in the first cluster are powered off, in a new browser window, navigate to the LoadBalancer service external IP address, and confirm that the NGINX landing page is still accessible.
11. Shut down the three hosts in the second cluster by right-clicking the hosts, and selecting Power → Power Off.
12. Once the hosts in the second cluster are powered off, in a new browser window, navigate to the LoadBalancer service external IP address, and confirm that the NGINX landing page is still accessible.
13. After confirming that the NGINX landing page is still accessible with two of the three workload availability zone clusters disabled, power all of the hosts back on.
14. Once the hosts are powered back on, confirm they have reconnected to vCenter and that the cluster control plane and worker node VMs have come back up automatically.
15. Confirm that the replica pods have redeployed in the two node pools that you disabled in steps 9 through 12 (there should be three pods with status "Running"):

```
kubectl get pods
```

Building a custom Ubuntu node OS image with VMware Tanzu Kubernetes Image Builder and using it to provision a workload cluster

1. Open a terminal on the Ubuntu bootstrap VM.
2. Clone the git repository for the Tanzu Kubernetes Grid Image Builder:

```
git clone https://github.com/vmware-tanzu/vsphere-tanzu-kubernetes-grid-image-builder.git
```

3. Add the local user to the Docker group to run Docker commands without superuser privileges:

```
sudo usermod -aG docker $(whoami)
```

4. Restart the VM, and open a new terminal to apply the change.
5. Install JQ:

```
sudo apt install -y jq
```

6. Install make:

```
sudo apt install make
```

7. From the repository folder, modify the vsphere.j2 configuration file:

```
nano packer-variables/vsphere.j2
```

8. Populate vsphere.j2 with your vCenter environment details, and save. We show our example below:

```
{
    {# vCenter server IP or FQDN #}
    "vcenter_server":"172.16.42.99",
    {# vCenter username #}
    "username":"administrator@tkg-test.lan",
    {# vCenter user password #}
    "password":"Password1!",
    {# Datacenter name where packer creates the VM for customization #}
    "datacenter":"infra",
    {# Datastore name for the VM #}
    "datastore":"datastore2",
    {# [Optional] Folder name #}
    "folder":"",
    {# Cluster name where packer creates the VM for customization #}
    "cluster": "",
    "host": "172.16.42.100",
    {# Packer VM network #}
    "network": "management-vm",
    {# To use insecure connection with vCenter  #}
    "insecure_connection": "true",
    {# TO create a clone of the Packer VM after customization#}
    "linked_clone": "true",
    {# To create a snapshot of the Packer VM after customization #}
    "create_snapshot": "true"
}
```

9. Modify the default-args.j2 configuration file to configure extra package(s) to install:

```
nano packer-variables/default-args.j2
```

10. Add the desired package(s) on the image by adding them to the “extra_debs” line:

```
    {% elif os_type == "ubuntu-2004-efi" %}
    "extra_debs": "unzip iptables-persistent nfs-common",
    "boot_disable_ipv6": "1"
    {% endif %}
```

- In our example, we added `curl` to the list of packages to install:

```
"extra_debs": "unzip iptables-persistent nfs-common curl",
```

11. Run the artifacts container for the desired version of Kubernetes from those listed by the `make list-versions` command:

```
make run-artifacts-container KUBERNETES_VERSION=v1.24.9+vmware.1
```

12. Run the image builder, where ARTIFACT_CONTAINER_IP is the address of the Linux bootstrap VM hosting the container and IMAGE_ARTIFACTS_PATH is the local directory to store the image OVA:

```
make build-node-image OS_TARGET=ubuntu-2004-efi KUBERNETES_VERSION=v1.24.9+vmware.1 TKR_SUFFIX=byoi
ARTIFACTS_CONTAINER_IP=172.16.222.45 IMAGE_ARTIFACTS_PATH=/home/ptuser/image ARTIFACTS_
CONTAINER_PORT=8081
```

13. Once the image builds successfully, retrieve the OVA from the location specified above, and log into vCenter.
14. Click the menu icon, and navigate to Content Libraries.
15. Click the TKR-local content library you created previously.
16. Click Actions → Import Item.
17. Select Local File, and click Upload Files.
18. Select the OVA you just created, and click Open.
19. Click Import.
20. In your Linux terminal, log into the Supervisor namespace:

```
kubectl vsphere login --server=192.168.0.194 --vsphere-username devops@tkg-test.lan --tanzu-
kubernetes-cluster-namespace tanzu-ns --insecure-skip-tls-verify
```

21. Ensure the namespace context is selected:

```
kubectl config use-context tanzu-ns
```

22. Confirm that the custom image you created is listed as a Tanzu Kubernetes Release:

```
kubectl get tkr
```

23. Create a YAML file with the following contents, and name it cluster-byoi.yaml:

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: cluster-byoi
  namespace: tanzu-ns
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: tanzukubernetescluster
    version: v1.24.9---vmware.1-byoi
    controlPlane:
      replicas: 3
      metadata:
        annotations:
          run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
    workers:
      #muliple node pools are used
      machineDeployments:
        - class: node-pool
          name: node-pool-1
          replicas: 3
          metadata:
            annotations:
              run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: zone-1
        - class: node-pool
          name: node-pool-2
          replicas: 3
          metadata:
            annotations:
              run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: zone-2
        - class: node-pool
          name: node-pool-3
          replicas: 3
          metadata:
            annotations:
              run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: zone-3
    variables:
      - name: vmClass
        value: guaranteed-medium
      - name: storageClass
        value: tkg-storage
```

24. Create the cluster using the custom node image:

```
kubectl apply -f cluster-byoi.yaml
```

25. Once the cluster is provisioned, use the following command to list running VMs:

```
kubectl get virtualmachines
```

26. Note the name of any of the cluster-byoi nodes from the above output, and use it to find its IP address:

```
kubectl describe virtualmachine cluster-byoi-node-pool-1-ftm94-74548c689d-6wlr2
```

27. Retrieve the SSH password secret for the cluster nodes:

```
kubectl get secrets cluster-byoi-ssh-password -o yaml
```

28. Decode the secret to get the cluster node SSH password:

```
echo NTBodCtIV2VQSVBpaGRGK1VGS2ZCNmc3emNzOG1DMlplTU56bmVRdlByTT0= | base64 --decode
```

29. SSH into the cluster node using the IP address and password from the steps above:

```
ssh vmware-system-user@192.168.0.129
```

30. Confirm that curl is pre-installed on the image:

```
curl -V
```

Read the report at https://facts.pt/hTYf0D6 ▶

This project was commissioned by Dell Technologies.

Principled Technologies®

Facts matter.®