The science behind the report:

Enjoy greater functionality and efficiency with VMware vSphere with VMware Tanzu

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 Enjoy greater functionality and efficiency with VMware vSphere with VMware Tanzu.

We concluded our hands-on testing on September 27, 2021. 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 August 15, 2021 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to http://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: Results of our disk usage testing

Solution	Setting	Disk usage in GB
VMware® vSphere® 7U2 with Tanzu™ cluster	Default setting of RAID 1 (two copies)	496.3
	Space-saving setting of RAID 5	380.0
Bare-metal Red Hat® OpenShift® 4.8 cluster	Default setting (three copies)	691.8
	Space-saving setting (two copies)	460.7

Table 2: Results of our memory usage testing

Solution	Memory usage in GB
VMware vSphere 7U2 with Tanzu cluster	415.01
Bare-metal Red Hat OpenShift 4.8 cluster	533.34

System configuration information

Table 3: Detailed information on the system we tested.

System configuration information	HPE DL380 Gen10
BIOS name and version	U30 v2.34
Non-default BIOS settings	VMware ESXi 7.0.2 Red Hat Enterprise Linux CoreOS 48.84.202109090400-0
Date of last OS updates/patches applied	4/8/2021
Power management policy	Dynamic Power Savings Mode
Processor	
Number of processors	2
Vendor and model	Intel® Xeon® Gold 5120 CPU
Core count (per processor)	14
Core frequency (GHz)	2.2
Stepping	M0
Memory module(s)	
Total memory in system (GB)	256
Number of memory modules	4
Vendor and model	Hynix® 809085-091
Size (GB)	64
Type	PC4-19200T
Speed (MHz)	2,400
Speed running in the server (MHz)	2,400
Storage controller	
Vendor and model	HPE Smart Array P408i-a SR Gen10
Cache size	N/A
Firmware version	3.53
Local storage (capacity)	
Number of drives	2
Drive vendor and model	Intel S4500 SSDSC2KG96
Drive size (GB)	960
Drive information (speed, interface, type)	6Gbps, SATA, SSD
Local storage (cache)	
Number of drives	2
Drive vendor and model	Intel S3700 SSDSC2BA40
Drive size (GB)	400
Drive information (speed, interface, type)	6Gbps, SATA, SSD

System configuration information	HPE DL380 Gen10
Network adapter	
Vendor and model	HPE 331i
Number and type of ports	4x 1Gb Ethernet
Firmware version	20.18.31
Network adapter	
Vendor and model	Mellanox® ConnectX-3 Ethernet Adapter
Number and type of ports	2x 10GbE SFP
Firmware version	02.40.70.04
Cooling fans	
Vendor and model	HPE PFR0612XHE
Number of cooling fans	6
Power supplies	
Vendor and model	HPE 865408-B21
Number of power supplies	2
Wattage of each (W)	500

How we tested

Deploying VMware vSphere 7U2 with Tanzu

Installing ESXi

1. Boot into the ESXi installation ISO.
2. At the installation welcome screen, press Enter.
3. At the EULA, to accept and continue, press F11.
4. Select the correct drive, and press Enter.
5. Select the keyboard layout, and press Enter.
6. Enter a root password, and press Enter.
7. In the confirm install screen, press F11.
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, to apply changes and restart the management network, press Y.
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. To accept the license agreement, check the box, 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:580/.
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).
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: 192.168.1.2/23, 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 provide only the starting IP of 10.209.201.201, which the shared cluster IP uses.
    - 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. At the Ready to complete screen, to create the cluster, click Finish.
11. In PowerCLI, enable vSAN disk space reclamation:

```
Get-Cluster -name vSAN*|set-VsanClusterConfiguration -GuestTrimUnmap:$true
```

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. To download the vSphere compatible HAProxy OVF(v0.1.8), click the following link: 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 HAproxy-cl content library you just created.
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 VM, which is now on, 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 192.168.0.1).
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 a storage tag

Create a storage tag to control virtual machine placement within storage policies.

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 `mirroring-tkg-policy`, and click Next.
5. Check the boxes for Enable tag based placement rules and Enable rules for vSAN storage, and click Next.
6. In the vSAN policy options, click the drop-down menu for Fault tolerance method, select RAID-1 (Mirroring), and click Next.
7. Use the Tag Category pull-down menu to select the Tanzu Storage policy you created. Click Browse Tags.
8. Click the Tanzu checkbox, and click OK.
9. Click Next.
10. Review the compatible storage to make sure your storage target is marked as compatible, and click Next.
11. Click Finish.
12. Click Create.
13. Create a new VM Storage policy named `parity-tkg-policy`, and click Next.
14. Check the boxes for Enable tag based placement rules and Enable rules for vSAN storage, and click Next.
15. In the vSAN policy options, click the drop-down menu for Fault tolerance method and select RAID-5/6 (Erasure Coding), and click Next.
16. Use the Tag Category pull-down menu to select the Tanzu Storage policy you created. Click Browse Tags.
17. Click the Tanzu checkbox, and click OK.
18. Click Next.
19. Review the compatible storage to make sure your storage target is marked as compatible, and click Next.
20. 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. To accept all license agreements, check the box, 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 192.168.0.2/23.
        - For Workload gateway, we used 192.168.0.1.
    - Load Balancing section
        - For load balancer IP ranges, we used 192.168.1.0/24.
        - 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 you wish, 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: `192.168.1.1-192.168.1.254`
   - Server Certificate Authority:
     - Open an SSH session to the HAProxy management address and connect using `root` and `Password1!`
     - Type the following: `cat /etc/haproxy/ca.crt`
     - Copy the entire output (including the first and last lines) and paste the contents into the Server Certificate Authority box.
     - Close the SSH session.
     - Click Next.
8. 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`
9. Click Next.
10. 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: `192.168.0.1`
    - Subnet: `255.255.254.0`
    - IP Address Ranges: `192.168.0.60-192.168.0.200`
11. Click Save.
12. For TKG Configuration, beside Add Content Library, click Add.
13. Select the TKG-cl library, and click OK.
14. Click Next.
15. 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 4.8

Configuring DNS

1. Log into your domain controller.
2. Press the Start key, type `DNS` and press Enter.
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. To acknowledge the creation of the host, click OK.
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
control-0.openshift	10.209.60.11
control-1.openshift	10.209.60.12
control-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, to move past the start screen, click Next.
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 the screen that allows you to edit 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-4.8
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
```

Installing OpenShift using installer-provided infrastructure

1. Create an install-config.yaml for your testbed. We have provided an example install-config.yaml in the Files we used in our testing section of this document. If you are using the install-config.yaml for the memory usage test, make sure to change the number of workers from five to three.
2. Create an install directory:

```
mkdir clusterconfig
```

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

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

4. 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 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.

Creating storage classes for OpenShift Container Storage

1. Click Storage→StorageClasses.
2. Click Create StorageClass.
3. In the StorageClass menu, choose the following options:
   - Name: 3-replica-class
   - Reclaim policy: Delete
   - Volume binding mode WaitForFirstConsumer
   - Provisioner: openshift-storage.rbd.csi.ceph.com
   - Storage Pool: Create New Pool
   - In Create BlockPool, choose the following options:
     - Pool name: 3-replica-pool
     - Data protection policy: 3-way Replication
     - Volume type: SSD
     - Click Create.
   - Storage Pool: 3-replica-pool

4. Click Create.
5. Click Create StorageClass.
6. In the StorageClass menu, choose the following options:
   - Name: 2-replica-class
   - Reclaim policy: Delete
   - Volume binding mode WaitForFirstConsumer
   - Provisioner: openshift-storage.rbd.csi.ceph.com
   - Storage Pool: Create New Pool
   - In Create BlockPool, choose the following options:
     - Pool name: 2-replica-pool
     - Data protection policy: 2-way Replication
     - Volume type: SSD
     - Click Create.
   - Storage Pool: 2-replica-pool
7. Click Create.

Enabling external IPs for OpenShift pods

1. In an OpenShift CLI, type the following to modify the OpenShift networking settings: `oc edit networks.config cluster`
2. Under `spec`, to enable externally accessible IPs, add the following lines:

```
externalIP:
  autoAssignCIDRs:
  - 10.209.254.0/24
  policy:
    allowedCIDRs:
    - 10.209.254.0/24
```

Figure 1 below shows the logical layout of the Redis cluster we used in our storage efficiency scenario.

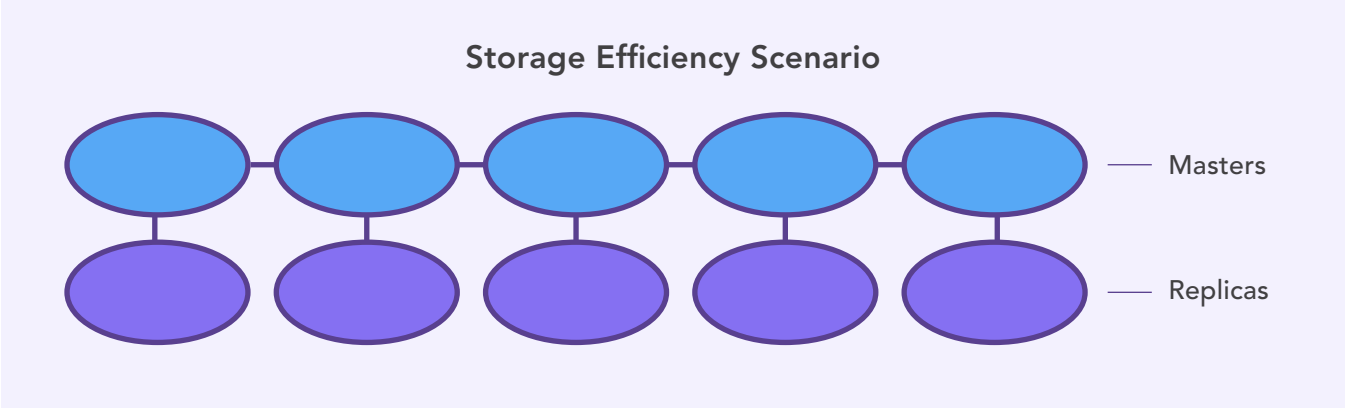

Figure 1: The logical layout of the Redis cluster used in our storage efficiency scenario. There are five Redis master nodes, each one backed by a single replica node, for a total of ten nodes. Source: Principled Technologies.

Installing and running the disk usage study on vSphere with Tanzu

1. Deploy a Tanzu Kubernetes Grid cluster to your Workload Management namespace. The specification file we used, scenario-2-cluster-raid1.yaml or scenario-2-cluster-raid5.yaml, is in the Files we used in our testing section of this document.

```
kubectl apply -f scenario-2-cluster.yaml
```

2. After the cluster has created (it may take some time), log into the cluster by first logging out of the Workload Management namespace context and logging in to 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. Add permissions to create the Redis cluster. The file we used, tkg-psp.yaml, is in the Files we used in our testing section of this document:

```
kubectl apply -f tkg-psp.yaml
```

4. Create the Redis namespace:

```
kubectl create ns redis
```

5. Using Helm, deploy Redis with the following specifications:

```
helm install redis-test bitnami/redis-cluster --set "cluster.nodes=10,persistence.
    size=100Gi,usePassword=false,cluster.externalAccess.enabled=true" -n redis
```

6. Get all the external IPs for Redis:

```
kubectl get svc -n redis
```

7. Attach the external IPs for Redis to their relevant pod:

```
helm upgrade --namespace redis redis-test --set "cluster.externalAccess.enabled=true,cluster.
    externalAccess.service.type=LoadBalancer,cluster.externalAccess.service.
    loadBalancerIP[0]=[IP0],cluster.externalAccess.service.loadBalancerIP[1]=[IP1],cluster.
    externalAccess.service.loadBalancerIP[2]=[IP2],cluster.externalAccess.service.
    loadBalancerIP[3]=[IP3],cluster.externalAccess.service.loadBalancerIP[4]=[IP4],cluster.
    externalAccess.service.loadBalancerIP[5]=[IP5],cluster.externalAccess.service.
    loadBalancerIP[6]=[IP6],cluster.externalAccess.service.loadBalancerIP[7]=[IP7],cluster.
    externalAccess.service.loadBalancerIP[8]=[IP8],cluster.externalAccess.service.
    loadBalancerIP[9]=[IP9],cluster.nodes=10,persistence.size=100Gi,usePassword=false"
    bitnami/redis-cluster
```

8. Use YCSB to create the dataset for the disk usage study:

```
~/ycsb-0.17.0/bin/ycsb.sh load redis -s -threads 20 -P ~/ycsb-0.17.0/workloads/workloada -p "redis.
    host=$SERVICE_IP" -p "redis.cluster=true" -p recordcount=50000000
```

9. After YCSB has finished creating the database, record the disk usage on your cluster.

Performing a disk trim

1. SSH into the first worker node.
2. Type `lsblk` to list the block volumes and note the volumes with the PVC formatting.
3. For each PVC volume, trim the unused disk space:

```
sudo fstrim [path to your PVC volume]
```

4. Repeat for each worker host.
5. Go back into the vSphere console and record the disk usage on your cluster.

Figure 2 on the following page shows the logical layout of the Redis cluster we used in our memory efficiency scenario

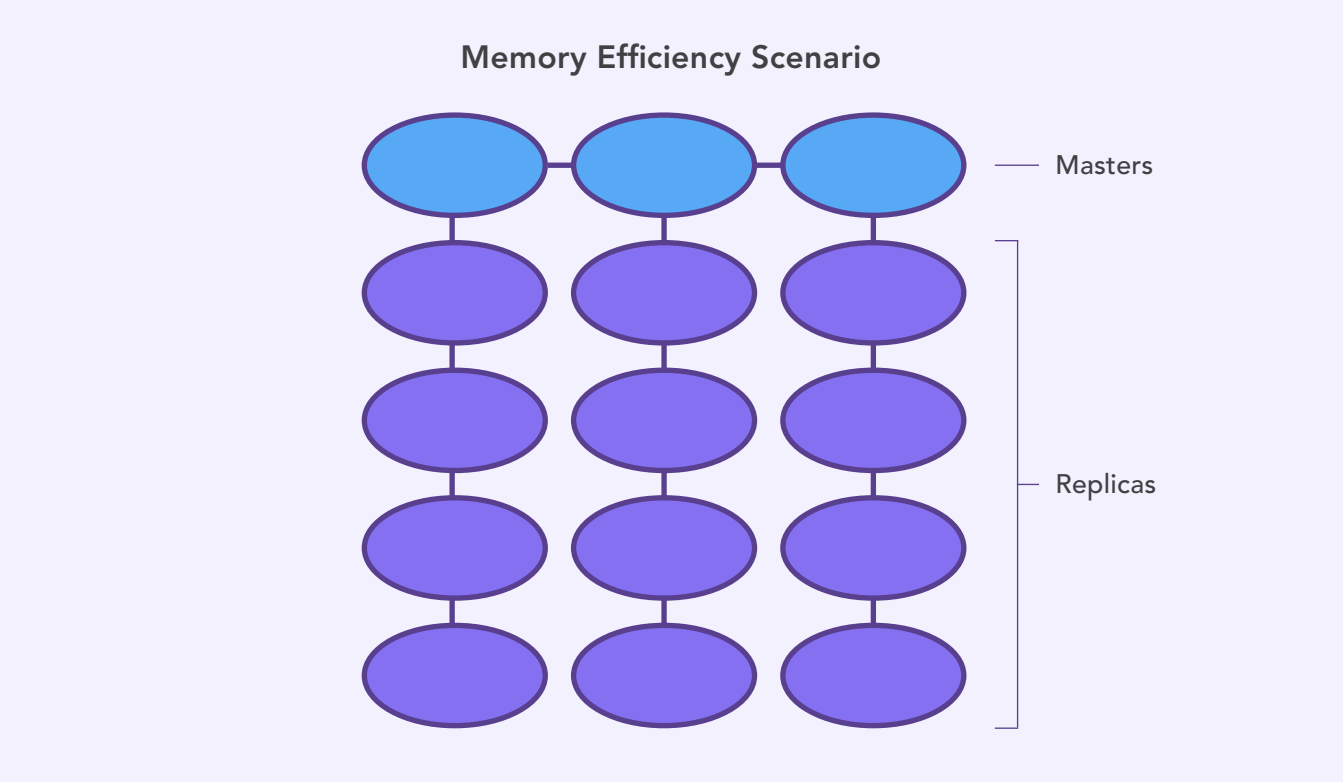

Figure 2: The logical layout of the Redis cluster used in our memory efficiency scenario. There are three Redis master nodes, each one backed by four replica nodes, for a total of fifteen nodes. Source: Principled Technologies.

Installing and running the memory usage study on vSphere with Tanzu

1. In the vCenter console, select Home → Workload Management.
2. In Workload Management, select the workload management namespace you created for Tanzu.
3. In the Virtual Machine Classes, select Create new Virtual Machine Class.
4. Create a new Virtual Machine class with 32 cores and 256GB of RAM. Label it `best-effort-8x-large-memory`
5. Attach the new Virtual Machine class to the workload management namespace.
6. Deploy a Tanzu Kubernetes Grid cluster to your Workload Management namespace. The specification file we used, scenario-3-cluster.yaml, is in the Files we used in our testing section of this document.

   ```
   kubectl apply -f scenario-3-cluster.yaml
   ```

7. After the cluster has created (it may take some time), log into the cluster by first logging out of the Workload Management namespace context and logging in to 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
   ```

8. To create the Redis cluster, add permissions. The file we used, tkg-psp.yaml, is in the Files we used in our testing section of this document:

   ```
   kubectl apply -f tkg-psp.yaml
   ```

9. Create the Redis namespace

   ```
   kubectl create ns redis
   ```

10. Using Helm, deploy Redis with the following specifications:

    ```
    helm install redis-test bitnami/redis-cluster --set "cluster.nodes=15,cluster.replicas=4,persistence.
        size=50Gi,usePassword=false,cluster.externalAccess.enabled=true" -n redis
    ```

11. Get all the external IPs for Redis:

    ```
    kubectl get svc -n redis
    ```

12. Attach the external IPs for Redis to their relevant pod:

```
helm upgrade --namespace redis redis-test --set "cluster.externalAccess.enabled=true,cluster.
    externalAccess.service.type=LoadBalancer,cluster.externalAccess.service.
    loadBalancerIP[0]=[IP0],cluster.externalAccess.service.loadBalancerIP[1]=[IP1],cluster.
    externalAccess.service.loadBalancerIP[2]=[IP2],cluster.externalAccess.service.
    loadBalancerIP[3]=[IP3],cluster.externalAccess.service.loadBalancerIP[4]=[IP4],cluster.
    externalAccess.service.loadBalancerIP[5]=[IP5],cluster.externalAccess.service.
    loadBalancerIP[6]=[IP6],cluster.externalAccess.service.loadBalancerIP[7]=[IP7],cluster.
    externalAccess.service.loadBalancerIP[8]=[IP8],cluster.externalAccess.service.
    loadBalancerIP[9]=[IP9],cluster.externalAccess.service.loadBalancerIP[10]=[IP10],cluster.
    externalAccess.service.loadBalancerIP[11]=[IP11],cluster.externalAccess.service.
    loadBalancerIP[12]=[IP12],cluster.externalAccess.service.loadBalancerIP[13]=[IP13],cluster.
    externalAccess.service.loadBalancerIP[14]=[IP14],cluster.nodes=15,cluster.replicas=4,persistence.
    size=50Gi,usePassword=false" bitnami/redis-cluster
```

13. Use YCSB to create the dataset for the memory usage study:

```
~/ycsb-0.17.0/bin/ycsb.sh load redis -s -threads 20 -P ~/ycsb-0.17.0/workloads/workloada -p "redis.
    host=$SERVICE_IP" -p "redis.cluster=true" -p recordcount=50000000
```

14. After YCSB dataset creation has filled the disks, record the memory usage of your cluster using the vCenter memory dashboard.

Installing and running the disk usage study on Red Hat OpenShift Container Platform

1. Create the Redis namespace:

```
oc create ns redis
```

2. To create the Redis cluster, add permissions:

```
oc adm policy add-scc-to-user anyuid system:serviceaccount:redis:default
```

3. Using Helm, deploy Redis with the following specifications:

```
helm install redis-test bitnami/redis-cluster --set "cluster.nodes=10,persistence.size=100Gi,
    global.storageClass=[2-replica-sc or 3-replica-sc],usePassword=false,cluster.externalAccess.
    enabled=true" -n redis
```

4. Get all the external IPs for Redis:

```
oc get svc -n redis
```

5. Attach the external IPs for Redis to their relevant pod:

```
helm upgrade --namespace redis redis-test --set "cluster.externalAccess.enabled=true,cluster.
    externalAccess.service.type=LoadBalancer,cluster.externalAccess.service.
    loadBalancerIP[0]=[IP0],cluster.externalAccess.service.loadBalancerIP[1]=[IP1],cluster.
    externalAccess.service.loadBalancerIP[2]=[IP2],cluster.externalAccess.service.
    loadBalancerIP[3]=[IP3],cluster.externalAccess.service.loadBalancerIP[4]=[IP4],cluster.
    externalAccess.service.loadBalancerIP[5]=[IP5],cluster.externalAccess.service.
    loadBalancerIP[6]=[IP6],cluster.externalAccess.service.loadBalancerIP[7]=[IP7],cluster.
    externalAccess.service.loadBalancerIP[8]=[IP8],cluster.externalAccess.service.
    loadBalancerIP[9]=[IP9],cluster.nodes=10,persistence.size=100Gi,global.storageClass=2-replica-
    sc,usePassword=false" bitnami/redis-cluster
```

6. Use YCSB to create the dataset for the disk usage study:

```
~/ycsb-0.17.0/bin/ycsb.sh load redis -s -threads 20 -P ~/ycsb-0.17.0/workloads/workloada -p "redis.
    host=$SERVICE_IP" -p "redis.cluster=true" -p recordcount=50000000
```

7. After YCSB has finished creating the database, record the disk usage on your cluster.

Performing a disk trim

1. SSH into the first worker node.
2. Type `lsblk` to list the block volumes and note the volumes with the PVC formatting.
3. For each PVC volume, trim the unused disk space:

```
sudo fstrim [path to your PVC volume]
```

4. Repeat for each worker host.
5. Go back to OpenShift and record the disk usage on your cluster.

Installing and running the memory usage study on Red Hat OpenShift Container Platform

1. Create the Redis namespace:

```
oc create ns redis
```

2. To create the Redis cluster, add permissions:

```
oc adm policy add-scc-to-user anyuid system:serviceaccount:redis:default
```

3. Using Helm, deploy Redis with the following specifications:

```
helm install redis-test bitnami/redis-cluster --set "cluster.nodes=15,cluster.replicas=4,persistence.
    size=50Gi,global.storageClass=[2-replica-sc or 3-replica-sc],usePassword=false,cluster.
    externalAccess.enabled=true" -n redis
```

4. Get all the external IPs for Redis:

```
oc get svc -n redis
```

5. Attach the external IPs for Redis to their relevant pod:

```
helm upgrade --namespace redis redis-test --set "cluster.externalAccess.enabled=true,cluster.
    externalAccess.service.type=LoadBalancer,cluster.externalAccess.service.
    loadBalancerIP[0]=[IP0],cluster.externalAccess.service.loadBalancerIP[1]=[IP1],cluster.
    externalAccess.service.loadBalancerIP[2]=[IP2],cluster.externalAccess.service.
    loadBalancerIP[3]=[IP3],cluster.externalAccess.service.loadBalancerIP[4]=[IP4],cluster.
    externalAccess.service.loadBalancerIP[5]=[IP5],cluster.externalAccess.service.
    loadBalancerIP[6]=[IP6],cluster.externalAccess.service.loadBalancerIP[7]=[IP7],cluster.
    externalAccess.service.loadBalancerIP[8]=[IP8],cluster.externalAccess.service.
    loadBalancerIP[9]=[IP9],cluster.externalAccess.service.loadBalancerIP[10]=[IP10],cluster.
    externalAccess.service.loadBalancerIP[11]=[IP11],cluster.externalAccess.service.
    loadBalancerIP[12]=[IP12],cluster.externalAccess.service.loadBalancerIP[13]=[IP13],cluster.
    externalAccess.service.loadBalancerIP[14]=[IP14],cluster.nodes=15,cluster.replicas=4,persistence.
    size=50Gi,global.storageClass=2-replica-sc,usePassword=false" bitnami/redis-cluster
```

6. Use YCSB to create the dataset for the memory usage study:

```
~/ycsb-0.17.0/bin/ycsb.sh load redis -s -threads 20 -P ~/ycsb-0.17.0/workloads/workloada -p "redis.
    host=$SERVICE_IP" -p "redis.cluster=true" -p recordcount=50000000
```

7. After YCSB dataset creation has filled the disks, record the memory usage of your cluster using the OpenShift Grafana dashboard

Files we used in our testing

install-config.yaml

```
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: [Admin Password]
              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: [Admin Password]
    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: [Admin Password]
    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: [Admin Password]
    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: [Admin Password]
    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: [Admin Password]
    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: [Admin Password]
    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: [Admin Password]
            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]'
```

scenario-2-cluster-raid1.yaml

```
apiVersion: run.tanzu.vmware.com/v1alpha1
    kind: TanzuKubernetesCluster
    metadata:
      name: tkg-cluster-01
      namespace: tkg-workload
    spec:
      distribution:
        version: 1.20.9+vmware.1-tkg.1.a4cee5b
      topology:
        controlPlane:
          count: 3
          class: guaranteed-medium
          storageClass: tkg-raid1-storage-policy
        workers:
          count: 10
          class: best-effort-4xlarge
          storageClass: tkg-raid1-storage-policy
      settings:
        storage:
          defaultClass: tkg-raid1-storage-policy
        network:
          cni:
            name: antrea
          services:
            cidrBlocks: ["172.17.0.0/16"]
          pods:
            cidrBlocks: ["172.18.0.0/16"]
```

scenario-2-cluster-raid5.yaml

```
apiVersion: run.tanzu.vmware.com/v1alpha1
    kind: TanzuKubernetesCluster
    metadata:
      name: tkg-cluster-01
      namespace: tkg-workload
    spec:
      distribution:
        version: 1.20.9+vmware.1-tkg.1.a4cee5b
      topology:
        controlPlane:
          count: 3
          class: guaranteed-medium
          storageClass: tkg-raid5-storage-policy
        workers:
          count: 10
          class: best-effort-4xlarge
          storageClass: tkg-raid5-storage-policy
      settings:
        storage:
          defaultClass: tkg-raid5-storage-policy
        network:
          cni:
            name: antrea
          services:
            cidrBlocks: ["172.17.0.0/16"]
          pods:
            cidrBlocks: ["172.18.0.0/16"]
```

scenario-3-cluster.yaml

```
apiVersion: run.tanzu.vmware.com/v1alpha1
    kind: TanzuKubernetesCluster
    metadata:
      name: tkg-cluster-01
      namespace: tkg-workload
    spec:
      distribution:
        version: 1.20.9+vmware.1-tkg.1.a4cee5b
      topology:
        controlPlane:
          count: 3
          class: guaranteed-medium
          storageClass: tkg-storage-policy
        workers:
          count: 3
          class: best-effort-8xlarge-memory
          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"]
```

tkg-psp.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
    kind: ClusterRole
    metadata:
      name: psp:privileged
    rules:
    - apiGroups: ['policy']
      resources: ['podsecuritypolicies']
      verbs:     ['use']
      resourceNames:
      - vmware-system-privileged
    ---
    apiVersion: rbac.authorization.k8s.io/v1
    kind: ClusterRoleBinding
    metadata:
      name: all:psp:privileged
    roleRef:
      kind: ClusterRole
      name: psp:privileged
      apiGroup: rbac.authorization.k8s.io
    subjects:
    - kind: Group
      name: system:serviceaccounts
      apiGroup: rbac.authorization.k8s.io
```

Read the report at https://facts.pt/yZoY2XZ ▶

This project was commissioned by VMware.