The science behind the report:

Run more Kubernetes pods and applications on VMware Cloud Foundation with VMware Kubernetes Service

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 Run more Kubernetes pods and applications on VMware Cloud Foundation with VMware Kubernetes Service.

We concluded our hands-on testing on March 3, 2026. 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 February 15, 2026 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 https://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 testing.

Measurement	VMware® Cloud Foundation™ (VCF) 9.0 with VMware vSphere® Kubernetes Service (VKS) 3.6 (four nodes)	Red Hat® OpenShift® 4.21 (four nodes)	Times improvement with VCF and VKS
Pods supported (higher is better)	42,000	7,400	5.6
Average latency for pods to reach a Ready state (seconds, lower is better)	35.71	175.58	4.9
99th percentile latency for pods to reach a Ready state (seconds, lower is better)	178	4,009	22.5

System configuration information

Table 2: Detailed information on the systems we tested.

System configuration information	Dell™ PowerEdge™ R640	
	VMware Cloud Foundation 9.0 (four nodes)	Red Hat OpenShift 4.21 (four nodes)
BIOS name and version	Dell 2.22.2	Dell 2.22.2
Non-default BIOS settings	None	None
Operating system name and version/build number	VMware ESXi 9.0.1.0, 24957456	OpenShift Container Platform 4.21
Date of last OS updates/patches applied	15 February 2026	15 February 2026
Power management policy	Performance	Performance
Processor		
Number of processors	2	2
Vendor and model	Intel® Xeon® Platinum 8260	Intel® Xeon® Platinum 8260
Core count (per processor)	24	24
Core frequency (GHz)	2.4	2.4
Stepping	7	7
Memory		
Total memory in system (GB)	768	768
Number of memory modules	24	24
Vendor and model	Hynix HMA84GR7CJR4N-WM	Hynix HMA84GR7CJR4N-WM
Size (GB)	32	32
Type	DDR4	DDR4
Speed (MHz)	2,933	2,933
Speed running in the server (MHz)	2,933	2,933
Storage controller 1		
Vendor and model	Dell BOSS-S1	Dell BOSS-S1
Firmware version	2.5.13 3024	2.5.13 3024
Local storage for OS		
Number of drives	1	1
Drive vendor and model	Intel SSDSCKKB240G8R	Intel SSDSCKKB240G8R
Drive size (GB)	240	240
Drive information (speed, interface, type)	6Gbps, M2, SSD	6Gbps, M2, SSD
Storage controller 2		
Vendor and model	Dell HBA330 Mini (Embedded)	Dell HBA330 Mini (Embedded)
Firmware version	16.17.01 00	16.17.01 00

System configuration information	Dell™ PowerEdge™ R640	
	VMware Cloud Foundation 9.0 (four nodes)	Red Hat OpenShift 4.21 (four nodes)
Local storage for distributed storage		
Number of drives	4	4
Drive vendor and model	Dell Express Flash NVMe™ P4500	Dell Express Flash NVMe™ P4500
Drive size (GB)	4	4
Drive information (speed, interface, type)	8GT/s, PCIe 3.0, NVMe 1.2	8GT/s, PCIe 3.0, NVMe 1.2
Network adapter 1 (VMs, management)		
Vendor and model	Mellanox ConnectX-4 LX 25GbE SFP	Mellanox ConnectX-4 LX 25GbE SFP
Number and type of ports	2x 25GbE	2x 25GbE
Driver version	14.32.20.04	14.32.20.04
Network adapter 2 (application and storage)		
Vendor and model	Mellanox ConnectX-5 Ex 100GbE QSFP	Mellanox ConnectX-5 Ex 100GbE QSFP
Number and type of ports	2x 100GbE	2x 100GbE
Driver version	16.35.30.06	16.35.30.06
Cooling fans		
Vendor and model	Dell	Dell
Number of cooling fans	8	8
Power supplies		
Vendor and model	Dell 0CMPGM	Dell 0CMPGM
Number of power supplies	2	2
Wattage of each (W)	1,100	1,100

How we tested

Deploying and configuring the testbed

Broadcom provided us with two four-node Dell PowerEdge R640 clusters with identical hardware to use as the server infrastructure for our testing. Broadcom fully set up and configured the VMware Cloud Foundation (VCF) 9.0 cluster. The cluster included VMware vSAN with four P4500 disks per node and cluster networking with one 10Gb connection and one 25Gb connection in a virtual distributed switch with port groups for vMotion, vSAN, VM, and management traffic. Broadcom also enabled vSphere Kubernetes Service (VKS) 3.6 on the VCF cluster with a maximum of 300 pods per node.

Broadcom also configured the Red Hat OpenShift 4.21 cluster installation process. We verified that Broadcom made the appropriate modifications for high pod density at cluster creation (e.g., the `maxPods` setting was 5,000 per node). This cluster contained a shared storage pool across all four nodes using OpenShift Data Foundation and the Local Storage operator, so we did not modify the infrastructure.

Testing with kube-burner

Configuring VKS for kube-burner testing

Before we could deploy kube-burner on VKS, we needed to add and connect to worker nodes to handle the load. Our steps consisted of running the following scripts:

1. Create and use a context for the new VKS cluster:

```
vcf context create pt --endpoint [your supervisor IP address] --insecure-skip-tls-verify -u [a
vSphere username with permissions to make changes]
vcf context use pt
```

2. Create the new VKS cluster:

```
kubectl apply -f kubeburner-vks.yaml
```

3. Connect to the VKS cluster:

```
vcf context create vks --endpoint [your supervisor IP address] --insecure-skip-tls-verify -u
[a vSphere username with permissions to make changes] --workload-cluster-namespace=kubeburner
--workload-cluster-name=kubeburner-vks
vcf context use vks:kubeburner-vks
```

Installing kube-burner on the client VM

We used a client VM to run kube-burner. This VM could access both the VMware and Red Hat deployments. We used the following scripts to install and configure kube-burner:

1. On the client VM, download kube-burner using their recommended process:

```
curl -Ls https://raw.githubusercontent.com/kube-burner/kube-burner/refs/heads/main/hack/
install.sh | sh
```

2. Extract the kubeburner compressed files:

```
tar -xzvf kube-burner-V2.2.2-linux-x86_64.tar.gz
```

3. Navigate to the kubelet-density-heavy workload folder:

```
cd kube-burner/examples/workloads/kubelet-density-heavy
```

4. Edit the kubelet-density-heavy.yml file for the following changes (we include an example config file in the appendix):

   - Change the jobIterations to your target number.
   - Change namespacedIterations to true
   - Change iterationsPerNamespace to 250

Running kube-burner

1. Connect to the client VM via SSH.
2. In the client, navigate to the kubelet-density-heavy directory:

```
cd kube-burner/examples/workloads/kubelet-density-heavy
```

3. Run the kube-burner test:

```
../../../kube-burner init -c kubelet-density-heavy.yml
```

   a. If kube-burner successfully runs the test, it will display results at the end. Otherwise, it will timeout after four hours and inform you that the test could not run successfully.

4. Before kicking off a new test, delete the previously created test pods:

```
For ns in `kubectl get namespace | grep kubelet-density | awk '{print $1}'`; do kubectl delete
namespace $ns; done
```

Appendix: Scripts we used

pt-pod-density.yaml

```
apiVersion: cluster.x-k8s.io/v1beta2
kind: Cluster
metadata:
  name: pdcluster1
  namespace: pod-density
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["10.96.0.0/12"]
    pods:
      cidrBlocks: ["192.168.0.0/15"]
    serviceDomain: "cluster.local"
  topology:
    classRef:
      name: builtin-generic-v3.6.0
    version: v1.35.0---vmware.2-vkr.4
    controlPlane:
      replicas: 3
      metadata:
        annotations:
          run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu,os-version=24.04
    workers:
      machineDeployments:
        - class: node-pool
          metadata:
            annotations:
              run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu,os-version=24.04
          name: node-pool-1
          replicas: 300
          variables:
            overrides:
            - name: vmClass
              value: best-effort-medium
            - name: volumes
              value:
                - name: containerd
                  mountPath: /var/lib/containerd
                  storageClass: vsan-esa-default-policy-raid5
                  capacity: 40Gi
    variables:
      - name: vmClass
        value: cpvm
      - name: storageClass
        value: vsan-esa-default-policy-raid5
      - name: vsphereOptions
```

```
        value:
          persistentVolumes:
            defaultStorageClass: vsan-esa-default-policy-raid5
      - name: kubernetes
        value:
          security:
            podSecurityStandard:
              enforce: baseline
          kubeletConfiguration:
            maxPods: 200
```

kubelet-density-heavy.yml

```
---
global:
  gc: false
  measurements:
    - name: podLatency

jobs:
  - name: kubelet-density-heavy
    jobIterations: 3800
    qps: 10
    burst: 10
    namespacedIterations: true
    iterationsPerNamespace: 250
    namespace: kubelet-density-heavy
    waitWhenFinished: true
    preLoadImages: false
    podWait: false
    objects:

      - objectTemplate: templates/postgres-deployment.yml
        replicas: 1

      - objectTemplate: templates/app-deployment.yml
        replicas: 1
        inputVars:
          readinessPeriod: 10

      - objectTemplate: templates/postgres-service.yml
        replicas: 1
```

This project was commissioned by Broadcom.

Read the report ▶

Primary contributors

Tech: Nathan C.

Writing: Nathan P.

Design: Jacqueline H.

PM: Claire A.

How we created this report

A PT team, which includes the contributors we've listed and others, created this report and performed the technical work behind it. We used AI to do some research and edit the report.

Principled Technologies®

Facts matter.®