REFERENCE ARCHITECTURE

Red Hat OpenShift on vSphere with Portworx

Reference architecture on deploying Portworx®
on OpenShift running on vSphere.

Contents

Executive Summary

Modern applications are built using containers and orchestrated by Kubernetes, but they still need a layer of persistence. Red Hat OpenShift, the industry's leading hybrid cloud application platform powered by Kubernetes, brings together tested and trusted services to reduce the friction of developing, modernizing, deploying, running, and managing applications. OpenShift delivers a consistent experience across public cloud, on-premise, hybrid cloud, or edge architecture.

To run stateful applications on Red Hat OpenShift, organizations need a robust data services platform like Portworx®. Portworx provides features like replication and high availability, security and encryption, capacity management, disaster recovery, and data protection to Red Hat OpenShift deployments. Instead of spending resources architecting and managing a custom Kubernetes storage layer, organizations can accelerate their modernization journeys by adopting a solution like Red Hat OpenShift with Portworx.

About This Document

This Portworx reference architecture contains a validated architecture and design model to deploy Portworx on OpenShift running on vSphere. The document is intended for Kubernetes Administrators and Cloud Architects who are familiar with Portworx.

The audience must be familiar with basic OpenShift concepts, like MachineSet and MachineAutoscaler, and vSphere concepts, like Datastore and Datastore Cluster.

The document has three main technical areas as described below:

- **Planning and architecture overview:** This section presents the high-level architecture overview on how Portworx will be deployed on OpenShift. It discusses the requirements related to OpenShift MachineSets for storage and storageless nodes and also vSphere Datastores configuration recommendations.
- **Design considerations:** This section provides more detailed requirements and recommendations that must be considered during the design phase. It covers OpenShift requirements, networking, capacity planning, high availability, resource and performance considerations, security and monitoring. At the end of the section a recommended Portworx installation template is presented.
- **Operations considerations:** This section covers “day 2” best practices after Portworx is deployed. It discusses how to validate the Portwox installation, how to scale Portworx, backup and recovery techniques and best practices on how to upgrade Portworx and OpenShift. A section discussing how to check Portworx logs is also available.

Value Proposition

Benefits of Portworx

Traditional storage solutions provide a simple container storage interface (CSI) driver connector to handle stateful applications in Kubernetes environments. A CSI connector has several limitations and doesn't provide a robust solution for high availability.

Unlike these solutions with CSI drivers, Portworx accelerates time to revenue, delivers data resiliency, and agility at enterprise scale for Kubernetes storage and databases—leading to a boost in platform engineering productivity.

Portworx storage services provide elastic scalability, industry-leading availability, and self-service access to storage for Kubernetes environments. Integrated storage management includes rule-based automation, thin-provisioning allocation and flexibility for multi-cloud, hybrid-cloud and on-premises.

Benefits of Running Portworx on OpenShift with vSphere

As part of digital transformation efforts, organizations are modernizing their applications and infrastructure by adopting containers and Kubernetes for their applications and leveraging a solution like Red Hat OpenShift for their infrastructure. Red Hat OpenShift allows organizations to take advantage of full-stack automated operations, a consistent experience across all environments, and self-service provisioning for developers that lets teams work together to move ideas from development to production.

Portworx adds a robust, secure, highly available, and scalable data management layer to Red Hat OpenShift so applications can consume storage in an easy way.

Target Use Cases

This document provides guidelines and best practices to deploy Portworx on OpenShift.

After deploying Portworx using the guidelines in this document, OpenShift users can deploy any type of stateful application in Portworx. The scope of this document does not include any specific recommendations for particular applications, but it is meant to be a stable deployment suitable for any application that requires storage.

Planning and Architecture Overview

Reference Architecture High Level Design

The diagram below shows a high level design of the Portworx reference architecture deployed on OpenShift running on vSphere.

By implementing this design, a Platform Engineering team can automate the provisioning of a well defined architecture following best practices that includes high availability, operations management, observability, business continuity, performance, and security.

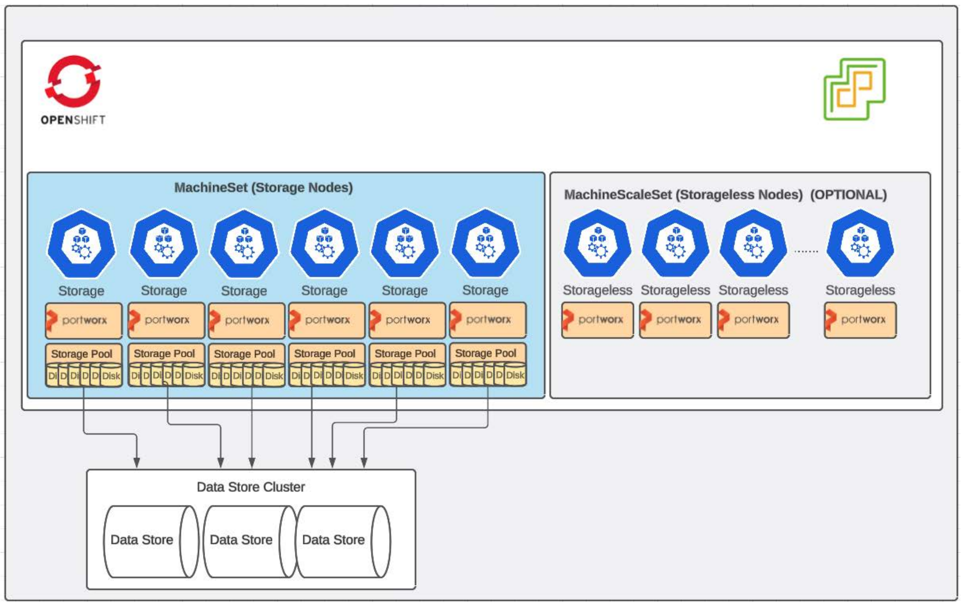


FIGURE 1 High level architecture overview

Portworx requires a minimum of three storage nodes in the cluster, but for a production environment this reference architecture recommends an initial cluster size with six storage nodes. Although a Portworx cluster with three storage nodes may work well in some environments, there are some advantages on using six storage nodes in the initial deployment:

- **Cluster capacity:** If a cluster with three storage nodes loses one, it loses one-third of its capacity, increasing the load on the remaining two nodes until the lost node recovers. In a six-node cluster, losing one node affects only one-sixth of its capacity, allowing the remaining five nodes to more evenly distribute the load.
- **I/O load distribution:** A Portworx cluster with three storage nodes may face more frequent I/O latencies during peak times. Conversely, a six-node cluster can distribute I/O requests more effectively, reducing the risk of I/O latencies.

Based on the recommendations above, it is important to prepare OpenShift and vSphere before deploying Portworx. Consider the following points:

Storage Nodes

A MachineSet with six storage nodes is the minimum requirement (less than six storage nodes is not recommended for production environments, unless the load is very minimal):

- It should have at least six nodes.
- If the OpenShift cluster spans multiple availability zones, equally distribute these storage nodes amongst them. If you have multiple availability zones, at least three zones are recommended.
- At install time, Portworx will set the configuration parameter "MaxStorageNodesPerZone" to put an upper limit on the maximum number of storage nodes it creates. Based on how many storage nodes you need to update this field in the StorageCluster spec.
- It cannot have an associated MachineAutoscaler (can scale horizontally, i.e. add more nodes if needed to increase storage capacity, but cannot shrink).
- Use the label "portworx.io/node-type:storage" on all nodes, so Portworx can automatically provision storage for new nodes
- Run hyperconverged applications on these nodes, i.e. applications that need to achieve high performance benchmarks.
- Each node must have a minimum of 8 CPU cores and 16 GB RAM (check the Resource considerations section for more details on CPU and memory requirements)

Optional Storageless Nodes

An optional MachineSet for storageless nodes:

- There is no minimum number of nodes.
- Can have an associated MachineAutoscaler.
- Use the label "portworx.io/node-type:storageless" on all nodes, so Portworx can automatically add nodes as storageless.
- Can be used in a very dynamic compute environment, where the number of compute nodes can elastically increase or decrease based on workload demand.
- Applications running on these nodes will access storage available in the storage nodes via the cluster network.
- Critical to closely monitoring performance in this scenario, since growing number of storageless nodes can overload the storage nodes and affect application performance.
- Each node must have a minimum of eight CPU cores and 16 GB RAM (check the Resource considerations section for more details on CPU and memory requirements).

Data Stores

In the case of vSphere, along with planning for cluster and node capacity, you must also size your datastores accordingly. The combination of vSphere datastore sizes and Portworx pool sizes must be selected initially such that future growth of pool resizing is easily satisfied. A Portworx pool can be expanded using two mechanisms:

- By adding a new disk (vmdk) to the pool
- By resizing all the existing disks from the pool

Portworx advises resizing existing disks in a pool rather than adding new ones when increasing pool size. Resizing is quick and immediate as it doesn't require data redistribution, unlike adding new disks, which is an I/O intensive process that takes time proportional to the amount of data needing redistribution.

Below are the guidelines on planning and provisioning vSphere Datastores for Portworx to tailor to the resize disk recommendation:

- Provision multiple and large sized datastores which can be expanded in the future. This reduces the additional work of managing VMFS datastore sizing.
- Having larger datastores avoids limitations on the maximum size of disk Portworx can create on them.
- Choosing fewer datastores of larger sizes is preferred over multiple datastores of smaller sizes. For example, it is better to have six datastores of 16TiB than 12 datastores of 8TiB.
- Expanding existing datastores is preferred over adding new datastores, to avoid the need for performing a Storage DRS and rebalancing of VMDKs.
- Portworx recommends limiting datastore usage to a single Portworx cluster. This helps in predicting the right starting datastore size. Do not share datastores across multiple Portworx clusters.

Storage Pools

Portworx recommends starting with a single pool per node with six drives in the pool.

- This can be achieved by having six different cloud drive spec entries in the StorageCluster spec where size and type of all the disks is the same. The example below will create a 3TiB storage pool per node where the pool has six disks.

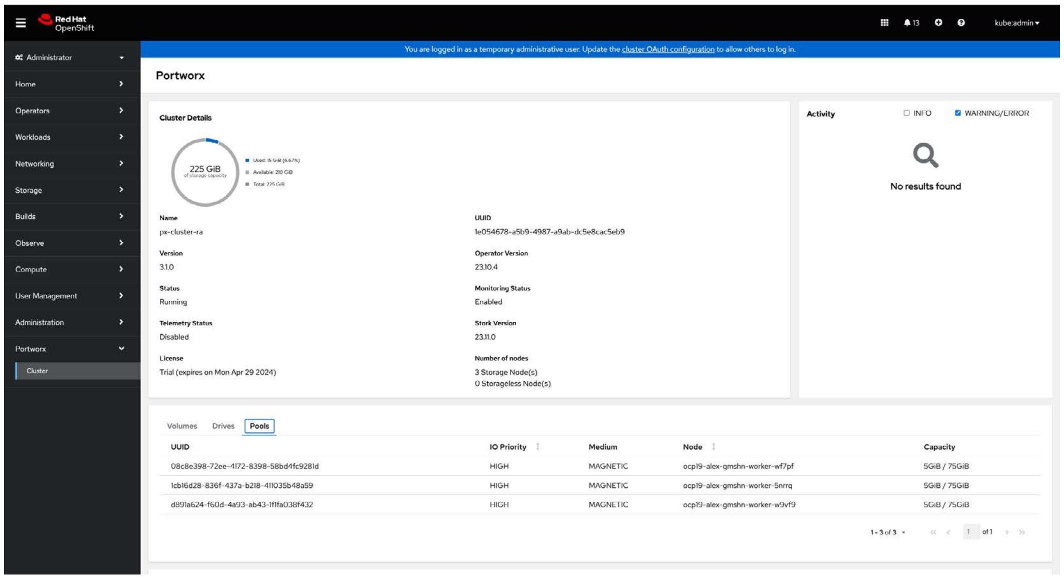
```
cloudStorage:
    deviceSpecs:
    - type=thin,size=500
    - type=thin,size=500
    - type=thin,size=500
    - type=thin,size=500
    - type=thin,size=500
    - type=thin,size=500
  journalDevice: type=thin,size=3
```

- Dividing capacity across multiple drives in the pool will allow expanding the pool with just a drive resize in future instead of adding disks to the pool.
- With multiple disks in a single pool, while expanding a pool a smaller incremental resize is required on the VMDKs which are spread out across datastores.
- If a datastore hosts a large single disk, chances are that the datastore won't have enough space to resize that single disk to satisfy the pool expansion requirement.
- However, when there are multiple disks in a pool, chances are that the datastore has the room for this smaller incremental resize required on the disk.
- Adding a disk to the pool is a more expensive and time consuming operation than resizing existing disks as addition leads to data movement.

Integrating Portworx on OpenShift on vSphere

Portworx is a complete Data Platform for Kubernetes and it is fully integrated with OpenShift. All features that Portworx provides to Kubernetes platforms are also available for OpenShift clusters. In addition, Portworx provides a plugin for OpenShift platforms that seamlessly integrates the Portworx Console UI with OpenShift Console.

The Portworx Console provides an overview of the Portworx cluster and allows some management tasks done directly in the UI. It also adds additional panels with Portworx volume details in the PVC and PV UIs available in the OpenShift console.



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Portworx

Cluster Details

Name: px-cluster-ra	UUID: 1e054678-a5b9-4987-a9ab-dc5e8cac5eb9
Version: 3.1.0	Operator Version: 23.10.4
Status: Running	Monitoring Status: Enabled
Telemetry Status: Disabled	Stork Version: 23.11.0
License: Trial (expires on Mon Apr 29 2024)	Number of nodes: 3 Storage Node(s), 0 Storageless Node(s)

UUID	IO Priority	Medium	Node	Capacity
08c8e398-72ee-4172-8398-58bd4fc9281d	HIGH	MAGNETIC	ocp19-alex-gmshn-worker-wf7pf	5GiB / 75GiB
1cb16d28-836f-437a-b218-411035b48a59	HIGH	MAGNETIC	ocp19-alex-gmshn-worker-5nrrq	5GiB / 75GiB
d891a624-f60d-4a93-ab43-1f1fa038f432	HIGH	MAGNETIC	ocp19-alex-gmshn-worker-w9vf9	5GiB / 75GiB

1 - 3 of 3 · 1 of 1

FIGURE 2 Portworx cluster dashboard in the OpenShift Console

OpenShift 4.12 or earlier is required to enable the Portworx Console plugin.

The plugin can be enabled during Portworx operator deployment or upgrade directly in the OpenShift UI.

Design Considerations

OpenShift Container Platform

The minimal version of OpenShift required for this reference architecture is 4.12 or newer.

If you're using an older version of OpenShift, the following features will not be available:

- Portworx plugin integration with OpenShift web console
- Portworx monitoring integration with OpenShift Prometheus

Networking

Portworx recommends a network bandwidth of 10Gbps, with a minimum requirement of 1Gbps with latency less than 10ms.

By default, OpenShift opens all ports among worker nodes. However, if your network includes specific firewalls, please ensure that the ports listed in the table below are open to allow node-to-node communication.:

Port	Protocol	Description
17001	TCP	PX Management
17002	TCP and UDP	PX node to node gossip communication
17003	TCP	PX store data port
17004	TCP	PX namespace
17005	TCP	PX control plane server
17006	TCP	PX data plane server
17018	TCP	Kvdb monitor port
17009	TCP	PX node to node communication
17010	TCP	PX namespace driver
17011	TCP	PX diagnostic service
17014	TCP	Watchdog server
17015	TCP	PX etcd peer-to-peer
17016	TCP	PX etcd client service
17017	TCP	PX SDK
17018	TCP	PX gRPC SDK gateway
17019	TCP	PX health monitor

TABLE 1 Portworx ports requirements

Additionally, the telemetry services have specific network requirements as outlined below:

Port	Protocol	Description
17021	TCP	Telemetry log uploader
20002	TCP	Telemetry phone home

TABLE 2 Portworx telemetry ports requirement

Telemetry also requires the following domains to be reachable by the worker nodes:

- https://logs-01.loggly.com
- register.cloud-support.purestorage.com:443
- rest.cloud-support.purestorage.com:443

Storage

This section provides guidelines on capacity planning for your Portworx cluster.

It covers three aspects of the capacity planning:

- Initial cluster capacity
- Storage node capacity sizing
- vSphere Datastore sizing

Initial Cluster Capacity

The following factors should be considered when creating the initial capacity planning:

- Number of volumes (PVCs) in the cluster
- Average size of volumes
- Number of nodes
- Replication factor (Portworx recommends a replication factor of two or three)

Below are two examples on how to calculate the initial cluster capacity:

Volume Size	Volumes	Replication	Cluster Size (1.3 x Repl x Volumes x Size)
50GiB	30	3	5.85 TiB
100GiB	50	2	13 TiB

TABLE 3 Calculating the initial cluster capacity.

The initial cluster size is calculated by multiplying the average volume size, number of volumes, and replication factor plus a 30% buffer for local snapshots.

Storage Node Capacity Sizing

Once you have the cluster size, you can calculate the size of each storage node with this equation:

Cluster size / Number of storage nodes = node capacity + min(10%, 100 Gib) for pool recovery in case the pool becomes full

Following the example above gives:

Cluster Size	Number of Storage Nodes	Node Capacity	Number of Datastores
5.85 TiB	6	1.075 TiB	3 (2.2 TiB each)
13 TiB	6	2.26 TiB	3 (4.6 TiB each)

TABLE 4 Calculating Node capacity sizing

vSphere Datastore Sizing

Finally, you can plan the number of vSphere Datastores in your environment. One key point to consider is that vSphere limits a datastore capacity to 64TB, and Portworx recommends multiple datastores rather than a single large one.

Creating multiple datastores gives more room to expand these datastores in the future when the capacity needs increase.

Expanding existing datastores is preferred over adding new datastores, to avoid the need for performing a Storage DRS and rebalancing of VMDKs.

In the first example above you can provide three datastores, each one initially with a 2.2 TiB size, while in the second example you can provide three datastores with 4.6 TiB size.

Implementing High Availability

To implement a highly available cluster, you can take advantage of Portworx topology awareness feature. In an OpenShift running on vSphere environment, Portworx can automatically identify topology labels, specifically topology.kubernetes.io/region and topology.kubernetes.io/zone.

Ensure that your OpenShift cluster has added topology labels on each node, the recommendation for the six storage nodes is to have all them in the same region and two nodes in each zone. A zone in this case must be an isolated entity and if it fails it does not affect other zones.

In this scenario Portworx will automatically place volume replicas in separate zones and if one zone becomes unavailable the application can run in a different zone where another replica of the same volume resides.

Deployment Model

This reference architecture uses a deployment model where applications can run both on the storage and storageless nodes.

As described in the reference architecture high level design section this document recommends two separate OpenShift MachineSets for storage and storageless nodes.

The MachineSet for the storage nodes provides a static set of nodes for hyper converged applications. This MachineSet can be scaled out if more storage nodes are needed, but cannot be scaled back, i.e. cannot automatically remove storage nodes from the Portworx cluster.

On the other hand, with OpenShift MachineAutoscaler, it is easier to manage Portworx storageless nodes in this deployment model. The optional MachineSet for the storageless nodes can have an associated MachineAutoscaler to automatically scale those nodes based on the cluster resources consumption.

For details on how to configure MachineSets and MachineAutoscalers please refer to the OpenShift documentation here.

Resource Considerations

When designing the cluster for a specific workload, make a note of the expected number of volumes in the cluster to be used at any given time, their average throughput or IOPS requirement, their HA level, and if snapshots are going to be used or not. Sum up these IOPS per volume, multiplied by their HA level, and add a snapshot and fragmentation overhead of 1.4. This will give you the approximate backend IOPS requirement, which should be less than the sum of the IOPS per pool across all the nodes in the cluster.

For vSphere, each pool could be aggregated out of many Vmdks across datastores, the total IOPS should be a sum of the IOPS per VMDK.

The table shows two examples using sample numbers for illustrative purposes:

Number of Volumes	Repl Factor	Average IOPS	Overhead	Total IOPS
240	2	200	1.4	134,400
300	3	200	1.4	252,000

TABLE 5 Total IOPs

The minimum requirements for each Portworx node are eight CPUs and 16GB RAM, but depending on the expected I/O load on the system, we recommend providing enough RAM and CPU resources in a high performance setting. For high performance systems, we recommend at least 32 cores and 32 GB ram where Portworx is running along with the applications on the same virtual machine.

Performance Considerations

Portworx is set up with a default configuration to optimize cluster performance, but certain situations may benefit from additional parameters. This section outlines scenarios where adding these parameters can boost the overall performance of the Portworx cluster.

- **Journal device:** A dedicated journal device is recommended in all cases, to enable it you can add the following in the StorageCluster yaml:

```
kind: StorageCluster
…
 cloudStorage:
    …
    journalDeviceSpec: type=lazyzeroedthick,size=3
…
```

When using a journal device you can also consider using the auto_journal I/O profile which can improve performance for volumes with replica 1. See this article for more details and use cases for this profile.

- **Storage class auto profile:** Portworx can auto-detect the optimal IO profile for an application when the storage class sets the io_profile to auto. In this case the replication factor must be set to two or three. Here is an example of a storage class using this parameter:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: px-storage-class
provisioner: pxd.portworx.com
allowVolumeExpansion: true
parameters:
  repl: "2"
  priority_io: "high"
  io_profile: auto
```

- **Runtime options:** Portworx consumes minimal CPU and memory from the VMs, by default using up to eight vCPUs and 8GB of memory. If your nodes have a high number of CPUs (greater than 32 vCPU) and large memory (greater than 64GB), it is recommended to use the rt_opts_conf_high runtime option to allow Portworx to use more CPU threads and memory, thus improving performance. Below is a snippet of the StorageCluster yaml with this parameter:

```
apiVersion: core.libopenstorage.org/v1
kind: StorageCluster
metadata:
  name: px-cluster
  namespace: portworx
spec:
   image: portworx/oci-monitor:3.1.0
      ...
   runtimeOptions:
          rt_opts_conf_high: "1"
  …
```

- **Nodiscard option:** Some applications, like Kafka and Elastic, perform a large number of discard/delete operations. This can affect the overall performance of the Portworx cluster and the nodiscard parameter is recommended. When the nodiscard parameter is used, the Portworx volume is mounted with the nodiscard option. This means that while data is deleted from the filesystem, it isn't immediately removed from the block device. Therefore, a filesystem trim operation must be run periodically to clear the deleted data from the block device.

The example below has a storage class definition with this option:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: px-storage-class
provisioner: pxd.portworx.com
allowVolumeExpansion: true
parameters:
  repl: "2"
  nodiscard: "true"
  io_profile: auto
```

Additionally, when using the nodiscard option, Portworx recommends configuring autofstrim in the cluster to periodically delete unused data blocks. For instructions on enabling this feature, refer to the Maintain volumes using Filesystem Trim article in the Portworx documentation.

Large Clusters Considerations

For large clusters (between 100 to 200 nodes) Portworx recommends additional configuration:

- Configuration of Portworx kvdb nodes: dedicated (reserved) 32 vCPU and 48GB memory

Add the following labels on the nodes reserved for kvdb nodes
(minimum 3 nodes, but at least 5 recommended for high availability):

```
# oc label nodes <node1> <node2> … <node6> px/metadata-node=true
```

- Increase number of sessions in vCenter

Allows more connections to be opened to the vCenter to enable better handling of the large number of nodes talking to the vCenter. Before making the changes ensure the vCenter Server has enough memory and CPU. The recommendation is to use at least 16 CPU and 32GB RAM.
vCenter vapi-endpoint config changes:

- maxSessionCount = 6000
- maxSessionsPerUser = 3000

Details on how to make these changes can be found the VMWare article below:

https://knowledge.broadcom.com/external/article/312158/vcenter-cloud-account-datacollection-fai.html

- Increase quorum timeout parameter

```
# pxctl cluster options update --runtime-options "quorum_timeout_in_seconds=3600"
```

Important note: For clusters larger than 200 nodes please consult your Portworx Sales team for additional guidance.

Security

Securing your Portworx cluster involves two key areas:

- **Authorization**: This protects Portworx volumes from unauthorized access.
- **Encryption**: This secures the data within the volumes by encrypting it.

Authorization

With authorization, Portworx adds an extra layer of security for the Portworx volumes. Using well-known industry standards, it protects the volumes from unauthorized access by adding a role-based access control (RBAC) mechanism. Only authenticated and authorized users can access the volumes. (By default Portworx creates a user token for the 'kubernetes' user when security is enabled.)

To enable authorization in Portworx, add the spec.security.enabled:true stanza in the StorageCluster yaml:

```
kind: StorageCluster
…
 spec:
 …
  security:
    enabled: true
…
```

Once enabled, only Kubernetes users will be able to access the Portworx volumes if PVCs are created using storage classes with the authentication token (see example here). By default 'guest access' is allowed if no authentication token is included in the storage class, to disable 'guest access' see the documentation here.

More details and advanced configuration for multi-tenant clusters can be found in this Portworx documentation link.

Encryption

Portworx recommends protecting your volumes with encryption. All encrypted volumes use a passphrase for protection and are encrypted both at rest and in transit.

Portworx supports several secret stores, such as Hashicorp Vault and Vault Transit, Kubernetes Secrets, and most of the public cloud provider secret stores. The complete list of supported secret store management can be found in the Secret Store Management article of the Portworx documentation.

For this reference architecture Portworx uses the Hashicorp Vault secret store.

Before deploying Portworx, you must configure your Openshift cluster to access the Vault server using the Vault kubernetes authentication method. Follow the instructions in the Vault article of the Portworx documentation to complete this configuration.

Then when deploying Portworx, you can add Vault as the secret store provider:

```
kind: StorageCluster
…
 spec:
  …
  secretsProvider: vault
  …
```

After Portworx is successfully deployed, you must define a cluster-wide passphrase for Portworx to encrypt the volumes.

Follow the steps in the Encrypting Kubernetes PVCs with Vault article of the Portworx documentation to create the passphrase and start using encrypted volumes with Portworx. This document recommends using at least the cluster-wide encryption feature provided by Portworx. For environments requiring additional security, such as multi-tenant clusters, refer to the documentation in the link above, which details more advanced per-volume encryption options.

Monitoring

Prometheus

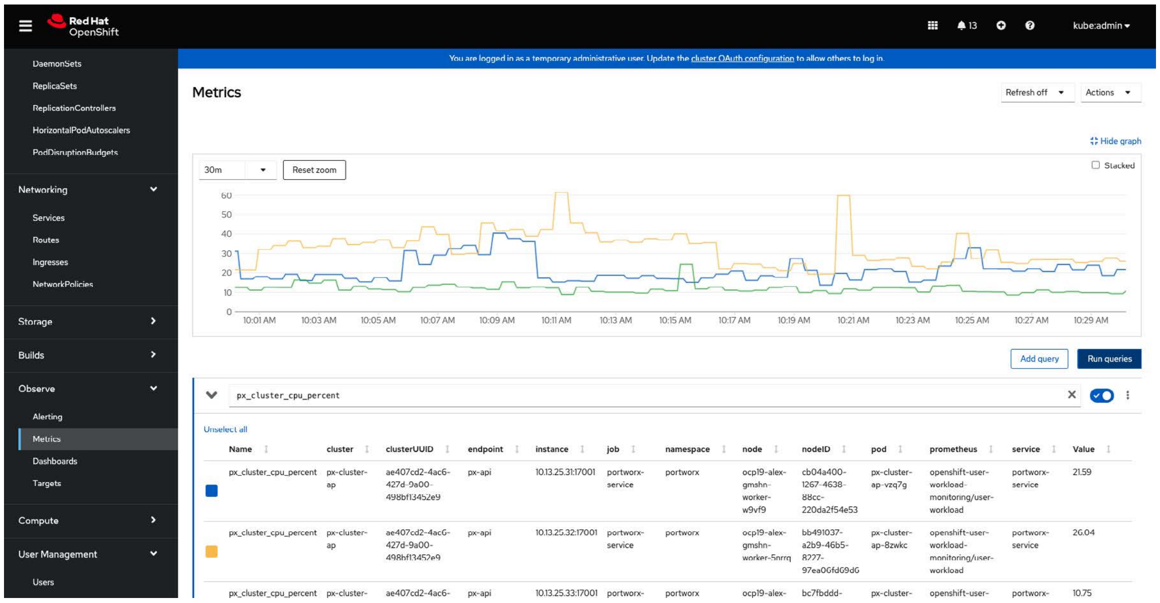
Portworx seamlessly integrates with the Openshift prometheus and provides several key metrics that can be used to monitor the health and performance of the Portworx cluster.

Before deploying Portworx in the OpenShift cluster, ensure that monitoring for user-defined projects is enabled. Follow the steps provided in the Create a Monitoring ConfigMap article of the Portworx documentation for guidance.

You can find a list of Portworx metrics in the Portworx Metrics Reference article of the Portworx documentation.

To query Portworx metrics in the OpenShift web console:

1. Find the Observe dropdown, then select the Metrics dashboard.
2. On the Metrics page, enter any Portworx metric, all of which start with 'px_'. For example, to check CPU usage on the Portworx cluster, use the metric 'px_cluster_cpu_percent'.



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

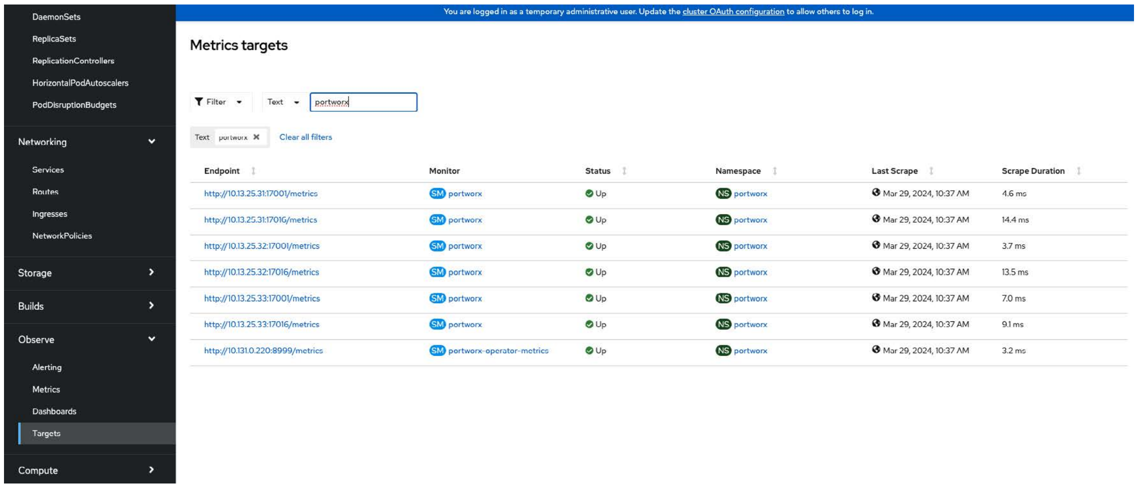
px_cluster_cpu_percent

	Name	cluster	clusterUUID	endpoint	instance	job	namespace	node	nodeID	pod	prometheus	service	Value
🟦	px_cluster_cpu_percent	px-cluster-ap	ae407cd2-4ac6-427d-9a00-498bf13452e9	px-api	10.13.25.31:17001	portworx-service	portworx	ocp19-alex-gmshn-worker-w9vf9	cb04a400-1267-4638-88cc-220da2f54e53	px-cluster-ap-vzq7g	openshift-user-workload-monitoring/user-workload	portworx-service	21.59
🟨	px_cluster_cpu_percent	px-cluster-ap	ae407cd2-4ac6-427d-9a00-498bf13452e9	px-api	10.13.25.32:17001	portworx-service	portworx	ocp19-alex-gmshn-worker-5nrrq	bb491037-a2b9-46b5-8227-97ea06fd69d6	px-cluster-ap-8zwkc	openshift-user-workload-monitoring/user-workload	portworx-service	26.04
	px_cluster_cpu_percent	px-cluster-	ae407cd2-4ac6-	px-api	10.13.25.33:17001	portworx-	portworx	ocp19-alex-	bc7fbddd-	px-cluster-	openshift-user-	portworx-	10.75

FIGURE 3 Portworx metrics

You can also check status of Portworx metrics targets:

1. Find the Observe dropdown, then select the Targets dashboard
2. Filter for “Portworx” in the text filter:



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Metrics targets

Text: portworx

Endpoint	Monitor	Status	Namespace	Last Scrape	Scrape Duration
http://10.13.25.31:17001/metrics	SM portworx	Up	NS portworx	Mar 29, 2024, 10:37 AM	4.6 ms
http://10.13.25.31:17016/metrics	SM portworx	Up	NS portworx	Mar 29, 2024, 10:37 AM	14.4 ms
http://10.13.25.32:17001/metrics	SM portworx	Up	NS portworx	Mar 29, 2024, 10:37 AM	3.7 ms
http://10.13.25.32:17016/metrics	SM portworx	Up	NS portworx	Mar 29, 2024, 10:37 AM	13.5 ms
http://10.13.25.33:17001/metrics	SM portworx	Up	NS portworx	Mar 29, 2024, 10:37 AM	7.0 ms
http://10.13.25.33:17016/metrics	SM portworx	Up	NS portworx	Mar 29, 2024, 10:37 AM	9.1 ms
http://10.131.0.220:8999/metrics	SM portworx-operator-metrics	Up	NS portworx	Mar 29, 2024, 10:37 AM	3.2 ms

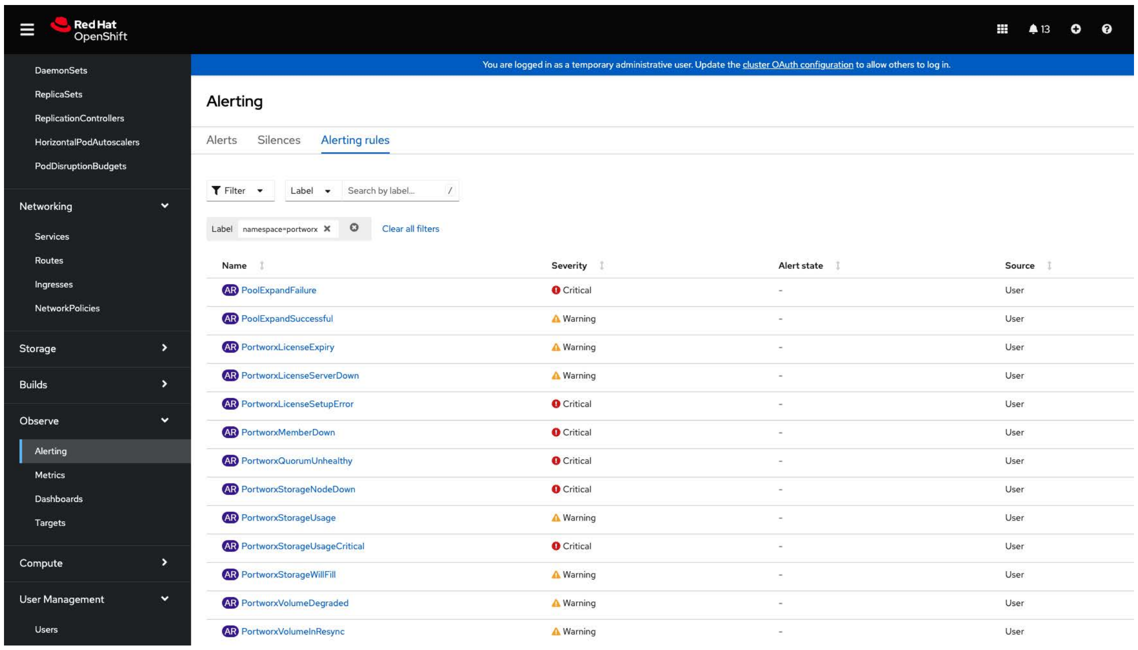
FIGURE 4 Portworx metrics targets

AlertManager

Similar to Prometheus metrics, you can set up user-defined alerts in OpenShift to receive standard Prometheus alerts provided by Portworx. To enable these alerts in your OpenShift cluster, follow the instructions in the Enabling Alert Routing for User-Defined Projects article of the OpenShift documentation.

After enabling user-defined alerts, access the Portworx alert rules via the OpenShift web console:

1. Under the Observe, select the Alerting dashboard
2. On the Alerting dashboard, select the 'Alerting rules' tab, and apply the 'namespace=portworx' filter to view the specific rules.



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Alerting

Label: namespace=portworx

Name	Severity	Alert state	Source
AR PoolExpandFailure	Critical	-	User
AR PoolExpandSuccessful	Warning	-	User
AR PortworxLicenseExpiry	Warning	-	User
AR PortworxLicenseServerDown	Warning	-	User
AR PortworxLicenseSetupError	Critical	-	User
AR PortworxMemberDown	Critical	-	User
AR PortworxQuorumUnhealthy	Critical	-	User
AR PortworxStorageNodeDown	Critical	-	User
AR PortworxStorageUsage	Warning	-	User
AR PortworxStorageUsageCritical	Critical	-	User
AR PortworxStorageWillFill	Warning	-	User
AR PortworxVolumeDegraded	Warning	-	User
AR PortworxVolumeInResync	Warning	-	User

FIGURE 5 Portworx Alerts

Note the following:

- Any fired (active) alerts will be displayed in the "Alerts" tab.
- You can see details on Portworx Prometheus rules by running the following command:

```
$ oc -n portworx get prometheusrules portworx -o yaml
```

Grafana Dashboards

Portworx provides five out-of-the-box Grafana dashboards to help monitor its status and performance. To deploy these dashboards in your own Grafana instance on OpenShift, follow the steps outlined in the Configure the OpenShift monitoring solution article of the Portworx documentation. If Grafana is not already installed on your OpenShift cluster, refer to the Grafana documentation for installation and configuration instructions.

Here is a list of Portworx Grafana dashboards:

- Internal KVDB (ETCD)
- Portworx Cluster
- Portworx Nodes
- Portworx Volumes
- Portworx Performance

Installation Methods and Tooling

Before installing Portworx on OpenShift, complete the prerequisite steps outlined in the Install Portworx on OpenShift on vSphere section of the Portworx documentation. Here is a summary:

- **Enable user workload monitoring in the Openshift cluster:** The latest releases of Openshift don't allow applications to deploy their own Prometheus instance, so you must enable user workload monitoring in the Openshift cluster to monitor Portworx.
- **Install the Portworx operator:** Portworx recommends installing the Portworx operator via the OpenShift web console for ease and convenience. Alternatively, you can deploy the operator using the oc command line tool if you prefer a command-line approach.
- **Create the px-vsphere secrets:** Portworx dynamically creates and manages vSphere disks using the VMware API. To enable this, you must create a Kubernetes secret with a username and password. Ensure that the user credentials meet the minimum requirements specified in the Portworx documentation.
- **Create a StorageCluster template from central.portworx.com:** Portworx recommends using its StorageCluster spec generator to create the initial StorageCluster spec template. Once generated, you can incorporate this template into an existing CI/CD pipeline as needed.

Below is a snapshot of a StorageCluster YAML file, created using the Portworx spec generator in accordance with the guidelines provided in this document. Some annotations have been omitted for simplicity:

```
kind: StorageCluster
apiVersion: core.libopenstorage.org/v1
metadata:
  name: px-cluster-dcdb0ef5-f652-41a6-ab70-1032c8992278
  namespace: portworx
  annotations:
     portworx.io/is-openshift: “true”
spec:
  image: portworx/oci-monitor:3.1.0
  imagePullPolicy: Always
  security:
    enabled: true
  kvdb:
    internal: true
  cloudStorage:
    deviceSpecs:
    - type=lazyzeroedthick,size=150
    - type=lazyzeroedthick,size=150
    - type=lazyzeroedthick,size=150
    - type=lazyzeroedthick,size=150
    - type=lazyzeroedthick,size=150
    - type=lazyzeroedthick,size=150
    journalDeviceSpec: type=lazyzeroedthick,size=3
  secretsProvider: vault
  startPort: 17001
  stork:
    enabled: true
    args:
      webhook-controller: “true”
  autopilot:
    enabled: true
  runtimeOptions:
    default-io-profile: “6”
  csi:
    enabled: true
  monitoring:
    telemetry:
      enabled: true
    prometheus:
      exportMetrics: true
  env:
  - name: VSPHERE_INSECURE
    value: “true”
  - name: VSPHERE_USER
    valueFrom:
      secretKeyRef:
        name: px-vsphere-secret
        key: VSPHERE_USER
  - name: VSPHERE_PASSWORD
    valueFrom:
      secretKeyRef:
        name: px-vsphere-secret
        key: VSPHERE_PASSWORD
  - name: VSPHERE_VCENTER
    value: “<vcenter_endpoint>”
  - name: VSPHERE_VCENTER_PORT
    value: “443”
  - name: VSPHERE_DATASTORE_PREFIX
    value: “<vcenter_datastore_prefix>”
  - name: VSPHERE_INSTALL_MODE
    value: “shared”
```

Note the following details on the specification above:

- cloudStorage.deviceSpecs: Contains 6 disks in the storage pool as discussed in the "Reference Architecture High Level Design" section
- cloudStorage.journalDeviceSpec: Create a 3GB journal device per best practices recommendation
- security: enabled: Enable RBAC authorization for Portworx volumes
- secretsProvider: vault: Use Vault for encryption and security
- vSphere username and password:

Read username and password to access vSphere APi from px-vsphere-secret

- vSphere details: Define details for vCenter URL and port, Datastore prefix

Operational Considerations

Post Installation Validation

After deploying Portworx, you can perform the following steps to ensure that all components are functioning correctly:

- Verify if all pods are running
- Verify the Portworx cluster status
- Verify the Portworx internal KVDB status
- Verify the Portworx cluster provisioning status (more details on the pools status)

To ensure your Portworx installation is successful, consult the Verify your Portworx Installation article in the Portworx documentation for detailed command instructions.

Once you confirm that Portworx is installed correctly, you can proceed to create your first persistent volume claim (PVC). For step-by-step instructions, visit the Create Your First PVC article in the Portworx documentation.

Scaling Portworx

There are several reasons to scale Portworx in your environment. Depending on these reasons, various methods can be employed to effectively scale your deployment.

Adding Storage

Adding more storage on current Portworx nodes is commonly referred to as vertical scaling up the cluster. Portworx recommends you use the Portworx Autopilot feature to accomplish this task.

You can create Autopilot rules to automatically increase the size of Portworx storage pools. The following types of Autopilot rules help on managing storage pools:

- Expand every Portworx storage pool in your cluster
- Expand Portworx storage pools

Adding Storage Nodes

In certain cases, you may need to add more storage nodes to your cluster. This task is commonly referred to as horizontal scaling out the cluster.

Portworx recommends adding a new node in your cluster if one of the current nodes reaches a consistent 80% IOPS or 80% CPU utilization. Monitoring the latency on the pools is also important and high latency can also indicate a need for a new node.

To scale out a cluster you can simply increase the size of the storage node MachineSet in Openshift. Portworx will automatically be deployed in the new node that is created.

Adding Compute (Storageless) Nodes

As mentioned in the "Reference Architecture High Level Design" section, if you plan to run stateful applications in compute nodes, i.e. nodes without storage, Portworx recommends you create a separate MachineSet in your Openshift cluster and automatically add the label portworx.io/node-type: storageless.

Storageless nodes can be created and removed as needed without impacting the overall status of the Portworx cluster.

Portworx automatically removes a storageless node from the cluster 20 minutes after the VM is deleted.

Backup and Recovery

Portworx recommends PX-Backup for backup and recovery of your cluster. PX-Backup is a complete Kubernetes backup solution fully integrated with Portworx. It is Kubernetes-aware, i.e. understands all Kubernetes resources like statefulsets, secrets, configmap, PVC, etc. and Portworx volumes, so you can have granular backups and restores if needed.

Portworx recommends creating different backup schedule policies for each namespace in your cluster and space out those schedules throughout the day. This minimizes backups to interfere and compete with regular I/O loads in the cluster.

For more details on PX-Backup please check its documentation in this link.

Upgrading Portworx

Pre-upgrade Checks

Before upgrading Portworx, it is important to check if the current deployment is healthy:

- Ensure all pods in the portworx namespace are running:

```
$ oc -n portworx get pods
NAME                                         READY    STATUS          RESTARTS        AGE
autopilot-d678b9c-zc5xv                      1/1      Running         0               7d3h
portworx-api-4g2vq                           2/2      Running         0               28m
portworx-api-4qvw7                           2/2      Running         0               29m
portworx-api-pr29q                           2/2      Running         0               31m
portworx-kvdb-9qp5w                          1/1      Running         0               7d4h
portworx-kvdb-p9lsn                          1/1      Running         0               7d4h
portworx-kvdb-ptxqg                          1/1      Running         0               7d4h
portworx-operator-84db5687c5-8cqzd           1/1      Running         0               29h
portworx-pvc-controller-cdf6967bd-2tfc9      1/1      Running         0               7d4h
portworx-pvc-controller-cdf6967bd-jj6m7      1/1      Running         0               7d4h
portworx-pvc-controller-cdf6967bd-qhmfm      1/1      Running         0               7d4h
px-cluster-ra-g5vsz                          1/1      Running         0               31m
px-cluster-ra-rjddq                          1/1      Running         0               29m
px-cluster-ra-wpzll                          1/1      Running         0               28m
px-csi-ext-866b9d8db7-4shtc                  4/4      Running         0               7d3h
px-csi-ext-866b9d8db7-hhqd8                  4/4      Running         0               7d3h
px-csi-ext-866b9d8db7-pzn6s                  4/4      Running         0               7d3h
px-plugin-56876bd449-926fm                   1/1      Running         0               7d4h
px-plugin-56876bd449-lmjbt                   1/1      Running         0               7d4h
px-plugin-proxy-545b555f5-qnfcc              1/1      Running         0               7d3h
px-telemetry-phonehome-22gg5                 2/2      Running         0               31m
px-telemetry-phonehome-mmzjh                 2/2      Running         0               31m
px-telemetry-phonehome-q4fpv                 2/2      Running         0               31m
Px-telemetry-registration-6c668c58b5         2/2      Running         0               31m
stork-795f64b46b-4sqhv                       1/1      Running         0               7d3h
stork-795f64b46b-9cgmc                       1/1      Running         0               7d3h
stork-795f64b46b-qqxck                       1/1      Running         0               7d3h
stork-scheduler-fcffdb957-fcxmg              1/1      Running         0               7d4h
stork-scheduler-fcffdb957-mlc62              1/1      Running         0               7d4h
stork-scheduler-fcffdb957-t7qh5              1/1      Running         0               7d4h
```

- Ensure all pods are in 1/1 state, i.e. running and ready. If any pod is not in this state please fix the pod(s) before starting the upgrade. Check the "Troubleshooting" section in this document or contact Portworx support for further assistance.Ensure all Openshift nodes are ready:

```
$ oc get nodes
NAME                            STATUS    ROLES    AGE     VERSION
ocp19-alex-gmshn-master-0      Ready     master   231d    v1.25.11+1485cc9
ocp19-alex-gmshn-master-1      Ready     master   231d    v1.25.11+1485cc9
ocp19-alex-gmshn-master-2      Ready     master   231d    v1.25.11+1485cc9
ocp19-alex-gmshn-worker-5nrrq  Ready     worker   231d    v1.25.11+1485cc9
ocp19-alex-gmshn-worker-w9vf9  Ready     worker   231d    v1.25.11+1485cc9
ocp19-alex-gmshn-worker-wf7pf  Ready     worker   231d    v1.25.11+1485cc9
```

Ensure that all nodes are in Ready state before starting the Portwox upgrade. If any node is not in the 'Ready' state, please address the issue before proceeding. For assistance, check OpenShift documentation or contact OpenShift support.

- Ensure all Portworx nodes are up and running. You can run the command below to check Portworx status:

```
$ PX_POD=$(oc get pods -l name=portworx -n portworx -o jsonpath='{.items[0].metadata.name}')
$ oc -n portworx exec $PX_POD -- /opt/pwx/bin/pxctl status
Status: PX is operational
Telemetry: Healthy
Metering: Disabled or Unhealthy
License: Trial (expires in 31 days)
Node ID: cf09a959-556a-4519-a83e-44daa72f4d84
IP: 10.13.25.32
Local Storage Pool: 1 pool
POOL    IO_PRIORITY     RAID_LEVEL      USABLE  USED    STATUS  ZONE            REGION
0       HIGH            raid0           75 GiB  5.0 GiB Online  px-cluster      us-west
Local Storage Devices: 1 device
Device  Path            Media Type              Size            Last-Scan
0:1     /dev/sdb STORAGE_MEDIUM_MAGNETIC  75 GiB 29 Mar 24 18:59 UTC
        total                   -                       75 GiB
        Cache Devices:
         * No cache devices
        Kvdb Device:
        Device Path     Size
        /dev/sdc 32 GiB
         * Internal kvdb on this node is using this dedicated kvdb device to store its data.
Cluster Summary
        Cluster ID: px-cluster-ra
        Cluster UUID: 1e054678-a5b9-4987-a9ab-dc5e8cac5eb9
        Scheduler: kubernetes
        Nodes: 3 node(s) with storage (3 online)
        IP              ID                                      SchedulerNodeName               Auth
StorageNode     Used    Capacity Status  StorageStatus   Version         Kernel                          OS
        10.13.25.32     cf09a959-556a-4519-a83e-44daa72f4d84    ocp19-alex-gmshn-worker-5nrrq   Disabled Yes
        5.0 GiB  75 GiB         Online  Up (This node)  2.13.7-1305f5a  4.18.0-372.59.1.el8_6.x86_64    Red Hat
Enterprise Linux CoreOS 412.86.202307040956-0 (Ootpa)
        10.13.25.33     895a249c-0ab6-4778-94a8-3c1d82924a2c    ocp19-alex-gmshn-worker-wf7pf   Disabled Yes
        5.0 GiB  75 GiB         Online  Up              2.13.7-1305f5a  4.18.0-372.59.1.el8_6.x86_64    Red Hat
Enterprise Linux CoreOS 412.86.202307040956-0 (Ootpa)
        10.13.25.31     81797d3d-f3b1-422b-bc89-50df2f3b5c39    ocp19-alex-gmshn-worker-w9vf9   Disabled Yes
        5.0 GiB  75 GiB         Online  Up              2.13.7-1305f5a  4.18.0-372.59.1.el8_6.x86_64    Red Hat
Enterprise Linux CoreOS 412.86.202307040956-0 (Ootpa)
Global Storage Pool
        Total Used      :  15 GiB
        Total Capacity  :  225 GiB
```

Similar to previous steps, ensure all Portworx nodes are online and errors or warnings are not displayed in the command above. If you encounter any errors, check the “Troubleshooting” section in this document or contact Portworx support for further assistance.

- Ensure all Portworx KVDB instances are running by running the command below:

```
$ PX_POD=$(oc get pods -l name=portworx -n portworx -o jsonpath='{.items[0].metadata.name}')
$ oc -n portworx exec $PX_POD -- /opt/pwx/bin/pxctl sv kvdb members
Kvdb Cluster Members:
ID                                      PEER URLs                               CLIENT URLs
LEADER  HEALTHY  DBSIZE
81797d3d-f3b1-422b-bc89-50df2f3b5c39     [http://portworx-2.internal.kvdb:17015]    [http://10.13.25.31:17016] false
true     300 KiB
cf09a959-556a-4519-a83e-44daa72f4d84     [http://portworx-3.internal.kvdb:17015]    [http://10.13.25.32:17016] false
true     296 KiB
895a249c-0ab6-4778-94a8-3c1d82924a2c     [http://portworx-1.internal.kvdb:17015]    [http://10.13.25.33:17016] true
true     312 KiB
```

Your output must:

- Show three KVDB members.
- Members must be all healthy.
- One of the members must be the leader.

If you encounter any errors, check the "Troubleshooting" section in this document or contact Portworx support for further assistance.

Operator Upgrade

After all prerequisites above have been completed the first component to upgrade is the Portworx operator.

By default, OpenShift automatically upgrades the operator when a new version is released, but if you disable automatic upgrades then you have to manually upgrade the operator to the latest version. You can do that in the OpenShift web console:

1. Under the Operators dropdown, select the Installed Operators dashboard.
2. Select the Portworx project and approve the upgrade to the latest version.
3. Run the following command to confirm Portwox operator is running after the upgrade:

```
> oc -n portworx get pod -l name=portworx-operator
NAME                                 READY   STATUS    RESTARTS    AGE
portworx-operator-84db5687c5-mw57d   1/1     Running   0           2d11h
```

Portworx Upgrade

Once you've upgraded the Operator, you're ready to begin upgrading Portworx and all its associated components. Portworx utilizes a rolling upgrade approach, upgrading one node at a time. It will only proceed to the next node after the previous one has been successfully upgraded.
To upgrade Portworx, edit the StorageCluster resource and update the Portworx image. For instance, to upgrade to release 3.1.0, modify the image entry as follows:

image: portworx/oci-monitor:3.1.0

Besides Portworx, other components are automatically upgraded as well to the versions compatible with the Portworx release you are upgrading to, like Autopilot, Stork and CSI.

For more detailed instructions on upgrading Portworx, visit the Upgrade Portworx using the Operator article of the Portworx documentation.

Upgrading OpenShift

Before upgrading OpenShift you must check if the new version of OpenShift is compatible with Portworx and will work properly. In some cases you may need to upgrade Portworx first before upgrading your OpenShift cluster.

To ensure the smooth operation of your OpenShift cluster with Portworx during and after an OpenShift upgrade please follow these best practices:

- If the new OpenShift version has a new kernel then make sure the current deployed version of Portworx is compatible with this new kernel version.
- If the new OpenShift version has a new version of kubernetes then make sure the currently deployed version of Portworx is compatible with it
- Make sure the deployed version of Portworx is compatible with the new version of OpenShift you are planning to use
- Make sure Portworx cluster is healthy
- Make sure Portworx internal KVDB is healthy and all 3 instances of KVDB are up and running. Portworx internal KVDB has an associated PodDisruptionBudget (PDB) that requires at least 2 KVDB pods be up and running, so you need all 3 KVDB pods running before starting the OpenShift upgrade, otherwise a node draining operation could fail and block the upgrade

If Portworx is not compatible with any of the points above, you must first upgrade Portworx to a supported version and then proceed with the OpenShift upgrade.

Logging and Monitoring

All Portworx pods will generate logs that can be viewed or retrieved using standard OpenShift command lines (oc logs). If necessary (or required by Portworx support personnel) you can increase log levels by updating the StorageCluster for each component, for example to enable debug level for the Stork component you can add the spec.stork.args.verbose: true stanza to the StorageCluster resource:

```
…
  stork:
    args:
      verbose: true
      webhook-controller: "true"
    enabled: true
…
```

To enable debug level in the Portwox container, add the PX_LOGLEVEL=debug environment variable to the StorageCluster specification:

```
...
 env:
  - name: PX_LOGLEVEL
    value: debug
...
```

To collect more details for troubleshooting Portworx, you can generate the Portworx diags by running the following command a specific node:

```
$ pxctl sv diags -a
```

This will generate a tar.gz file that can be sent to Portworx support for investigation on issues. If you have Telemetry enabled the diags file is automatically to Pure1.

To enable Telemetry follow the instructions in the documentation link here.

Application Considerations

Application HA

Portworx ensures data availability through storage replication, allowing an application to access its data from a replica node should the original node become unavailable. However, if the node hosting an application pod fails, the application will experience downtime until Kubernetes moves this pod to another healthy node within the cluster. Thus, while storage remains accessible, the application itself may be temporarily offline. Depending on your requirements and the nature of your application, you may opt for one of the following strategies:

1. **HA Mode**: For applications that support HA mode and require zero downtime, it is advisable to run multiple replicas of the application pods. This configuration ensures continuous service availability, as the failure of a node hosting one of the pods will lead the other replicas to seamlessly continue handling traffic.
2. **Non-HA mode**: For applications that can withstand temporary downtime during Kubernetes' failover process or those that do not support HA mode, deploying a single replica of the application pod is sufficient.

Portworx Images

To get list of Portworx images, you can point your browser to an URL similar to this:

https://install.portworx.com/3.1/images\?kbver\=1.25.11

The example above will display images for the latest PX version 3.1 and kubernetes version 1.25.11.

You can also use the curl command, for example:

```
# curl  https://install.portworx.com/3.1/images\?kbver\=1.25.11
docker.io/portworx/px-enterprise:3.1.1
docker.io/portworx/oci-monitor:3.1.1
docker.io/openstorage/stork:23.11.1
docker.io/openstorage/cmdexecutor:23.11.1
docker.io/portworx/autopilot:1.3.14
docker.io/purestorage/ccm-go:1.2.2
docker.io/purestorage/realtime-metrics:1.0.23
docker.io/purestorage/telemetry-envoy:1.1.11
docker.io/purestorage/log-upload:px-1.1.8
docker.io/portworx/px-operator:23.10.5
registry.k8s.io/pause:3.1
registry.k8s.io/kube-controller-manager-amd64:v1.25.11
registry.k8s.io/kube-scheduler-amd64:v1.25.11
docker.io/portworx/portworx-dynamic-plugin:1.1.0
docker.io/nginxinc/nginx-unprivileged:1.25
registry.k8s.io/kube-scheduler-amd64:v1.21.4
registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.9.0
registry.k8s.io/sig-storage/csi-provisioner:v3.6.1
docker.io/openstorage/csi-attacher:v1.2.1-1
registry.k8s.io/sig-storage/csi-resizer:v1.9.1
registry.k8s.io/sig-storage/csi-snapshotter:v6.3.1
registry.k8s.io/sig-storage/snapshot-controller:v6.3.1
quay.io/prometheus/prometheus:v2.48.1
quay.io/prometheus-operator/prometheus-operator:v0.70.0
quay.io/prometheus-operator/prometheus-config-reloader:v0.70.0
quay.io/prometheus/alertmanager:v0.26.0
```

You can find more details on the list of images and how you can upload those images to a local repository in the Install Portworx on bare metal air-gapped Kubernetes cluster article of the Portworx documentation.

Monitoring During the Installation

Once Portworx is deployed, you can follow the progress by checking status of the pods:

```
# oc -n portworx get pods
NAME                                    READY   STATUS              RESTARTS       AGE
autopilot-588c9dd998-wjk6t              0/1     ContainerCreating   0              4s
portworx-api-jffgp                      0/2     ContainerCreating   0              3s
portworx-api-mnj9n                      1/2     Running             0              3s
portworx-api-nfmz6                      0/2     ContainerCreating   0              3s
portworx-operator-84db5687c5-mw57d      1/1     Running             2 (7d21h ago)  17d
portworx-pvc-controller-cdf6967bd-7k5cf 1/1     Running             0              6s
portworx-pvc-controller-cdf6967bd-8xlh4 1/1     Running             0              6s
portworx-pvc-controller-cdf6967bd-zjj9n 1/1     Running             0              6s
px-csi-ext-866b9d8db7-2fxc9             0/4     Pending             0              3s
px-csi-ext-866b9d8db7-6n2qj             0/4     Pending             0              3s
px-csi-ext-866b9d8db7-znp9c             0/4     Pending             0              3s
px-ocp-ra-2t2v4                         0/1     ContainerCreating   0              3s
px-ocp-ra-7lgxb                         0/1     ContainerCreating   0              3s
px-ocp-ra-9xkjm                         0/1     ContainerCreating   0              3s
px-plugin-56876bd449-c7dz8              1/1     Running             0              6s
px-plugin-56876bd449-g9t8w              1/1     Running             0              6s
px-plugin-proxy-545b555f5-65kmf         0/1     ContainerCreating   0              6s
stork-694758bd55-f46bj                  0/1     ContainerCreating   0              7s
stork-694758bd55-r5x7l                  0/1     ContainerCreating   0              6s
stork-694758bd55-xqh7r                  0/1     ContainerCreating   0              6s
stork-scheduler-fcffdb957-bpkj7         1/1     Running             0              6s
stork-scheduler-fcffdb957-trbmw         1/1     Running             0              6s
stork-scheduler-fcffdb957-w8zrs         1/1     Running             0              6s
```

Another monitoring option is to get the status of the storagenodes resources:

```
# oc -n portworx get storagenodes
NAME                             ID     STATUS         VERSION    AGE
ocp19-alex-gmshn-worker-5nrrq           Initializing              114s
ocp19-alex-gmshn-worker-w9vf9           Initializing              114s
ocp19-alex-gmshn-worker-wf7pf           Initializing              114s
```

At the end of deployment, all pods must be running and in the `Ready` state, if any problems happen please check the troubleshooting section below.

Validating Post Installation: Checking Cluster Operators, Cluster Version and Nodes

After deploying Portworx you can follow the steps in this link to verify the installation and create your PVC with a Portwox storage class.

Troubleshooting Commands

To troubleshoot installation issues you can check the logs from the pods that may be failing and look for some errors. For example to troubleshoot a specific Portworx pod from a cluster deployed with the name `px-ocp-ra` you can run this command:

```
# oc -n portworx logs px-ocp-ra-2t2v4
```

Other helpful commands:

- Describe a pod to check for any errors in the Events section. The example below shows that the `stork` image pull is failing:

```
# oc -n portworx describe pod stork-694758bd55-r5x7l
Events:
  Type     Reason          Age                   From               Message
  ----     ------          ----                  ----               -------
  Normal   Scheduled       14m                   default-scheduler  Successfully assigned portworx/stork-694758bd55-
r5x7l to ocp19-alex-gmshn-worker-wf7pf
  Normal   AddedInterface  14m                   multus             Add eth0 [10.128.2.166/23] from openshift-sdn
 Warning  Failed          14m                   kubelet            Failed to pull image "docker.io/openstorage/
stork:23.11.1": rpc error: code = Unknown desc = writing blob: storing blob to file "/var/tmp/storage63576889/3":
happened during read: read tcp [2620:125:9006:1324:bb4c:9a1:72cf:488b]:55724->[2606:4700::6810:62d7]:443: read:
connection reset by peer
  Normal   Pulling         12m (x4 over 14m)     kubelet            Pulling image "docker.io/openstorage/
stork:23.11.1"
  Warning  Failed          12m (x4 over 14m)     kubelet            Error: ErrImagePull
  Warning  Failed          11m (x6 over 14m)     kubelet            Error: ImagePullBackOff
  Normal   BackOff         4m29s (x35 over 14m)  kubelet            Back-off pulling image "docker.io/openstorage/
stork:23.11.1"
```

- Retrieve logs from the Portworx operator pod:

```
# oc -n portworx logs -l name=portworx-operator --tail=9999
```

For more information on how to troubleshoot Portworx, refer to the Troubleshooting section of the Portworx Documentation.

Details on how to contact Portworx support are available here.

Legal Notices and Attributions

purestorage.com

800.379.PURE

portworx by Pure Storage