Reference Architecture

REFERENCE ARCHITECTURE DEPLOYING PORTWORX PX-ENTERPRISE™ ON MESOSPHERE DC/OS

portworx | MESOSPHERE

EXECUTIVE SUMMARY

Mesosphere DC/OS makes it easy to build and run modern distributed applications in production at scale, by pooling resources across an entire datacenter or cloud. With the DC/OS platform, you get a public cloud like experience, from an open partner ecosystem, on any infrastructure.

Running stateful applications, such as Cassandra, ElasticSearch, Kafka, Spark and Hadoop, requires special care to ensure persistence, high availability (HA), and security in a completely automated way that works for any service or infrastructure.

Portworx PX-Enterprise, the container data services platform which scales up to one thousand nodes per cluster and is used in production by DC/OS users such as GE Digital, solves the operational and data management problems enterprises encounter when running stateful applications on DC/OS.

Mesosphere and Portworx jointly created this reference architecture that describes how to deploy data-intensive infrastructure applications with Mesosphere DC/OS and Portworx PX-Enterprise.

INTRODUCTION: THE BENEFITS AND CHALLENGES OF MODERN CONTAINERIZED APPS

Thanks to Docker, enterprises have adopted container technology at an explosive growth rate in the past decade. Containers provide a process runtime and application packaging format, which have concrete benefits in terms of speed to launch, density and portability. However, the combination of containers and microservices yield the biggest benefits. These outsize benefits explain why enterprises, often conservative when it comes to technologies that require large-scale platform refactoring, are leading the charge to adopt containers, with well-known companies such as GE and Capital One publicly discussing their move to container-based architectures.

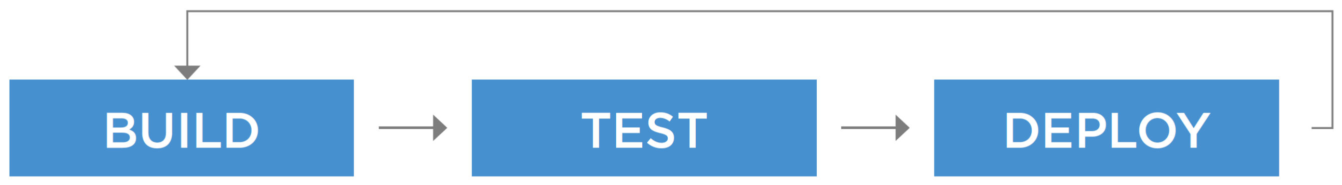
Benefits of container-based microservices

Container-based microservices provide two main benefits to enterprises: improved agility and application resilience.

Agility

The big don't eat the small anymore. The fast eat the slow. Software-powered innovation is transforming every major industry. If a team can speed up the build-test-deploy cycle faster than their competitors, they can capture a larger share of the market by better responding to changing conditions.

Speeding up this cycle makes you more competitive



Container-based microservices improve agility by breaking an application into multiple smaller parts that can be independently built, tested and deployed without affecting any other part of the application.

Container-based microservices stand in contrast to what are often called "monolithic" applications. Often, in a monolithic application, improvement to one part of the code requires changes to another. This tight coupling of features into a single codebase leads to infrequent and high-risk software releases. Enterprises that release new versions of software only quarterly or annually open themselves up to disruption by more nimble competitors.

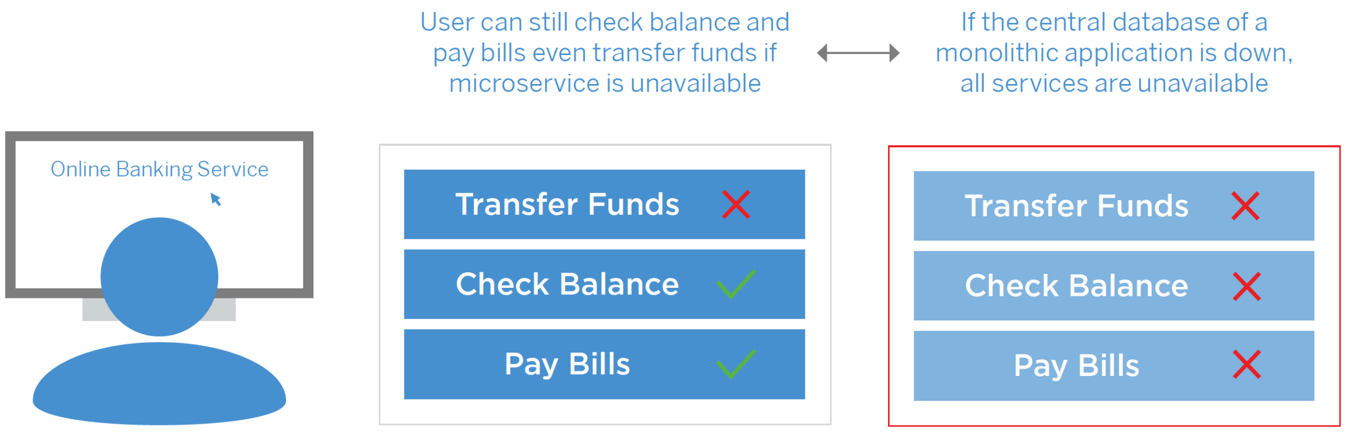
Container-based microservices also improve agility by putting a premium on automation. Automating a task means that it can be done faster and more frequently without increasing the risk of human error. Automation also lets you concentrate on automating other tasks that are currently manual and error-prone.

Resilience

Container-based microservices are faster to build, test, and deploy, but are they of higher quality? The evidence from enterprises would suggest yes.

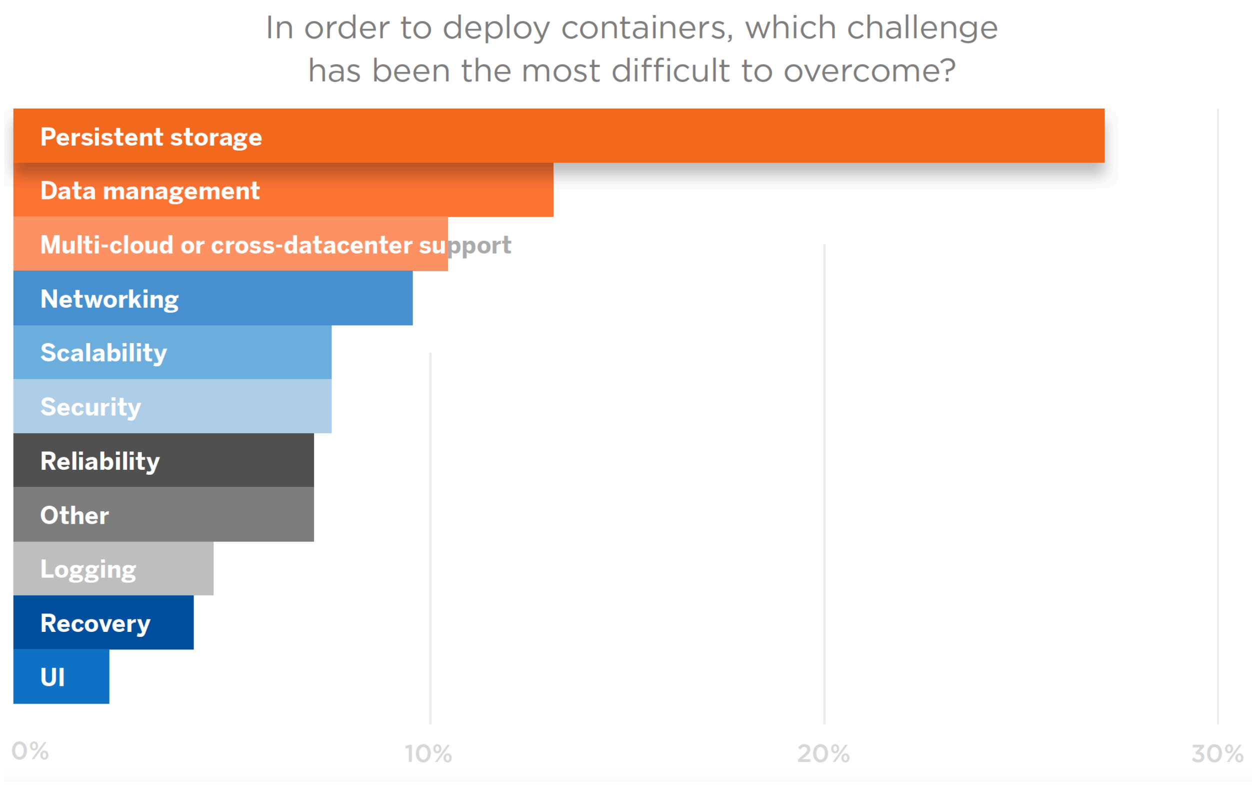
The reason is that because microservices are "loosely coupled," a failure in one part of the system is less likely to affect another. For example, if an online banking service built using microservices is having a problem with its 'transfer funds' function, a user can still check account balances or pay bills online because each of the individual features is its own microservice, complete with its own database. While the user might experience a degraded experience, the service is still useful as a composition of the functioning parts.

Contrast this microservices-based resilience with a monolithic banking application, where the inability to access a single Oracle database causes the user to be unable to check account balances, transfer funds, or pay a bill.


User can still check balance and pay bills even transfer funds if microservice is unavailable
If the central database of a monolithic application is down, all services are unavailable
Online Banking Service
Transfer Funds
Check Balance
Pay Bills
Transfer Funds
Check Balance
Pay Bills

The challenge of stateful containers

The benefits of container-based microservices are clear. However, innovative enterprises who took the microservices path have run into problems converting their databases, queues, and key-value stores to run inside containers. In fact, a recent industry survey indicates that persistent storage was the #1 barrier to production deployments of containers..



This section examines some of the general barriers to overcome when running stateful applications in containers, with special emphasis on particular issues that users of DC/OS can encounter.

Persistence

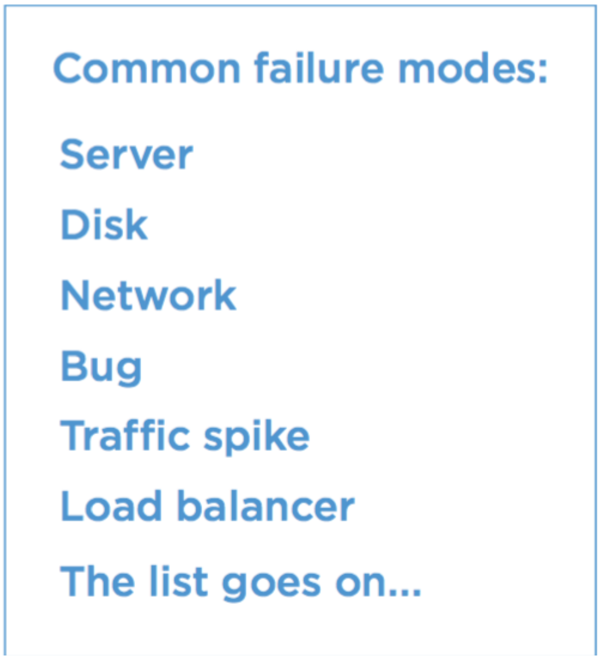
The first problem with stateful containers is that native Docker doesn't provide a persistence layer that enables a containerized database to survive host failure. While a container can mount a local disk as an external volume, if the host crashes or if the container gets rescheduled on a different host, then none of that data is accessible.

Data Durability

Stateful containers require reliable storage solutions that are not only container and scheduler-aware but offer mission critical data durability that protect valuable customer data 24x7x365 with no data-loss or data-availability issues.

High availability

While HA has always been a requirement for certain databases, it is increasingly becoming a requirement for all. Infrequent software releases typically require DevOps teams to issue maintenance notices to inform their internal and external customers about service disruptions during migrations or updates. Such disruptions are unacceptable when software updates are daily or weekly. Databases must be available during updates, deployments, high traffic events, and a variety of host, network and disk failures.



Common failure modes:

- Server
- Disk
- Network
- Bug
- Traffic spike
- Load balancer
- The list goes on...

Traditional solutions for HA aren't a fit for modern container-based microservices. Using custom hardware to mitigate failure is expensive (and anti-DevOps). Application-layer replication can be slow and tank your performance especially as the cluster rebuilds itself.

The lack of a persistence layer compounds the issues related to ensuring HA for stateful containers.

Security

Security is important for all parts of an application, but given the sensitivity around data, especially for regulated industries, it is of particular importance for stateful containers. Enterprises moving to stateful containers must explore encryption and access controls.

For encryption, it is important that data in motion is encrypted using SSL or other protocols, and that data at rest is encrypted with a key that only the customer can access.

For access controls, it is important to ensure that only containers with sufficient privileges can access certain data volumes. Access control is critical for organizations building multi-purpose clouds or PaaS infrastructure where different customer groups share the same underlying elastic infrastructure but want strict data governance to ensure that only applications and users with the correct privileges can access their data.

Scheduler-based automation

Another challenge is scheduler-based automation of data services. While it is easy to theoretically create a system to persist and secure data using custom hardware solutions or highly controlled manual processes, these practices offset the agility gains provided by microservices. To be agile, DevOps teams can't wait days or weeks for storage to be provisioned so they can deploy their application. They need stateful containers that are as easy to deploy and manage as the stateless parts of their app. A container manager, such as DC/OS, provides the necessary data provisioning and management automation.

Any database, any infrastructure

Not only must enterprise teams solves persistence, HA, security and data automation for one database running in one environment, they must solve those challenges for many databases in many environments.

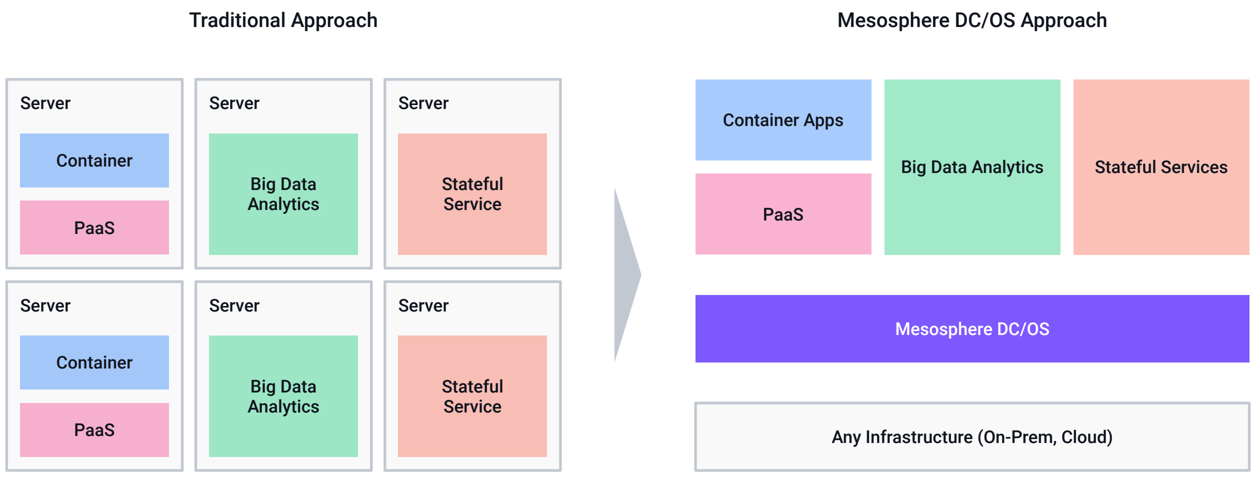
You can no longer run everything on an Oracle database managed by a dedicated team of DBAs. Now, DevOps teams must consistently manage a wide mix of SQL and NoSQL databases as well as other stateful services such as Jenkins, GitLab, or WordPress. Additionally, DevOps must handle those databases and stateful services in many different environments: AWS, Azure, Google, VMware and bare metal.

INTRODUCTION: MESOSPHERE DC/OS ENTERPRISE

Mesosphere makes modern enterprise apps easy to build, run, and scale with DC/OS, a data center-scale platform that elastically runs the full modern app: containerized microservices and stateful data services.

The Datacenter Operating System model

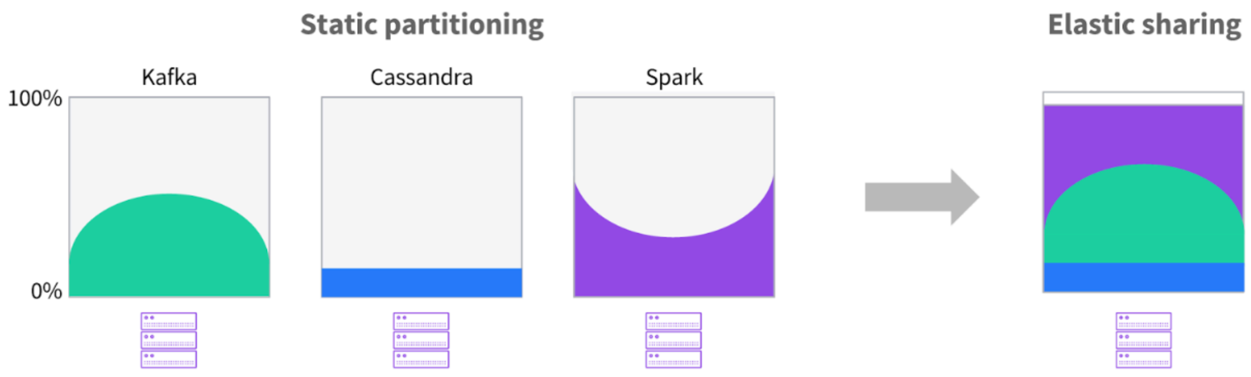
The Datacenter Operating System (DC/OS) model was conceived with the disruptive idea that running datacenter-scale services should not be more complex than using a single computer. DC/OS aggregates primitive infrastructure services so that both the developer and the operator work with a single form factor: the logical datacenter. In addition to providing elastic scalability for distributed systems, DC/OS ensures high availability and fault tolerance of services.



Elastic Data Infrastructure with Mesosphere DC/OS

The core of DC/OS is the Apache Mesos distributed systems kernel. Its power comes from the two-level scheduling that enables distributed systems to be pooled and share datacenter resources. Mesos provides the core primitives for distributed systems, such as resource allocation, isolation, and quota management.

Operators in a DC/OS model do not spend time managing individual machines (physical or virtual), dramatically reducing time and effort. A key benefit for operators is the ability to run at high levels of utilization, even as demand from multiple distributed services changes over time.



Elastic Resource Sharing Example

Special consideration for DC/OS environments

DC/OS offers two approaches to persistent storage: local persistent volumes and external persistent volumes. At a high-level, local persistent volumes involve using the storage available on a host that tasks run on, while external persistent volumes involve network-attached storage like Amazon EBS or an on-premises SAN. Both of these approaches have important limitations documented on the Mesosphere DC/OS website and outlined below. For full details, see the DC/OS docs on local and external volumes.

Using Portworx PX-Enterprise on DC/OS addresses each of these limitations.

Using local volumes

Tasks are pinned to a single host

Once a task has run on a server with a local volume, it is "pinned" to that server and you can't reschedule it to run onto a different server, even if the initial host becomes unavailable.

Resource requirements are fixed at start time

Just as a task run on a host with a local volume is forced to always run on that host, the initial resource requirements of a task using a local volume are fixed. That means that if a database is running out of space, resizing the volume first requires that you stop the application, take down the task, spin up a new host with a larger disk, transfer the data, and bring the task back up.

External management for replication and backups

Because a given task is pinned to a host and if that hosts dies, all data related to that task is lost, teams must think about replication and backups to protect any mission-critical infrastructure. DC/OS is a great solution for scheduling and managing containers in a large scale. DC/OS combined with the PX-Enterprise scale-out container-granular data services, including snapshots and CloudSnap™ off-site backup, enables you to protect all your data, all the time in any cloud or on-premises infrastructure.

External volumes

One of the ways to get external volumes support today is through connector plugins to legacy storage products. But these connector products have several pitfalls as documented in the following sections and on the DC/OS website.

Only one task per volume

External volumes accessed via a connector plugin can be used by only one task at a time. This prevents applications from accessing their data from multiple hosts. The connector products prevent customers from realizing the true agility, elasticity and scale that could be unleashed by containerizing the application.

Cross-AZ limitations

Since connector plugins do not provide storage, they might not be able to ensure cross-AZ availability. Security of container data volumes as cloud-based block devices and some on-premises SANs are not available across AZs or datacenters.

Inconsistent launch times

Similarly, since connectors are not full fledged storage services stacks, they are highly dependent on the underlying capabilities of the storage system. For EBS or Google Persistent Disk, simply starting a task with an volume mounted can take two minutes or more. Similar launch times can affect on-premises SANs.

Low density of stateful containers per host

There are some other limitations of connector-based external volumes that are important to understand. First, while thousands of containers can run per host, there are limitations to how many block devices can mount to a host due to the networking limitations in the Linux kernel. For example, when using a connector plugin, you can mount a maximum of 40 EBS volumes per EC2 instance before risking boot failure. So while simple, the model of one EBS volume per container volume can severely limit the density of containers that can be achieved per host. If reducing infrastructure costs or increasing density is a desired attribute for an application, a connector-based approach might not be a good fit.

Slow block device mounts

Additionally, the mount operation of a physical block device to a container host can be a slow operation, taking up to 2 minutes, and frequently failing altogether, requiring a host reboot. As a result of the time it takes to unmount and remount a block device, failover operations, such as those supported by connector plugins, do not function well as an HA mechanism.

Volumes at run time can't be dynamically provisioned

Connector plugins do not support the in-line volume specification which allows DC/OS platform users to launch containers that use storage without having to log a ticket and wait for someone to manually create a volume for their app.

Can't scale up stateful apps

One of the biggest benefits of a container platform is the ability to easily add more instances to a running application, enabling you to smoothly keep up with incoming user requests. This is known as app scaling and the concept is supported from with the DC/OS UI. An operator can simply increase the number of instances from 1 to 2, for example. If the app was using a volume exposed via a connector, however, that request would be blocked by the platform since the use of a connector volume limits the number of tasks an app can have to 1. An operator can "scale down" to 0 to effectively pause the app and then scale back up to 1 but can not go past that when using a connector.

Being able to scale apps is important, for example, for running Consul on Marathon. Other examples include things like DiskCache and EHCache which would also like to be able to scale volumes with compute when running on Marathon.

INTRODUCTION: PORTWORX PX-ENTERPRISE

Portworx PX-Enterprise is a software-defined storage solution purpose built for DevOps. PX-Enterprise, designed to be an integral part of a Mesosphere DC/OS deployment, automates the deployment and operations of data services at scale. With PX-Enterprise, an operator can manage any database or stateful service on any infrastructure using any container scheduler. PX-Enterprise offers a single data management layer for all stateful services, no matter where they run.

PX-Enterprise is a clustered block storage solution that provides a cloud-native data layer from which containerized stateful applications programmatically consume storage services directly through schedulers such as DC/OS. The PX-Enterprise software is delivered as a container that gets installed on DC/OS agents that run stateful applications.

Portworx technology:

- Provides virtual, container-granular data volumes to applications running in containers and enables dynamic volume management at a container granular level
- Is scheduler aware, providing data persistence and HA across multiple nodes, cloud instances, regions, datacenters, or even clouds.
- Is application aware. Applications such as Cassandra are deployed as a set of containers, and PX-Enterprise is aware of the entire stack. Data placement and management is done at an application level to ensure performance and availability.
- Manages physical storage that is directly attached to servers, whether cloud volumes such as AWS EBS, or hardware arrays such as an on-premises SAN.

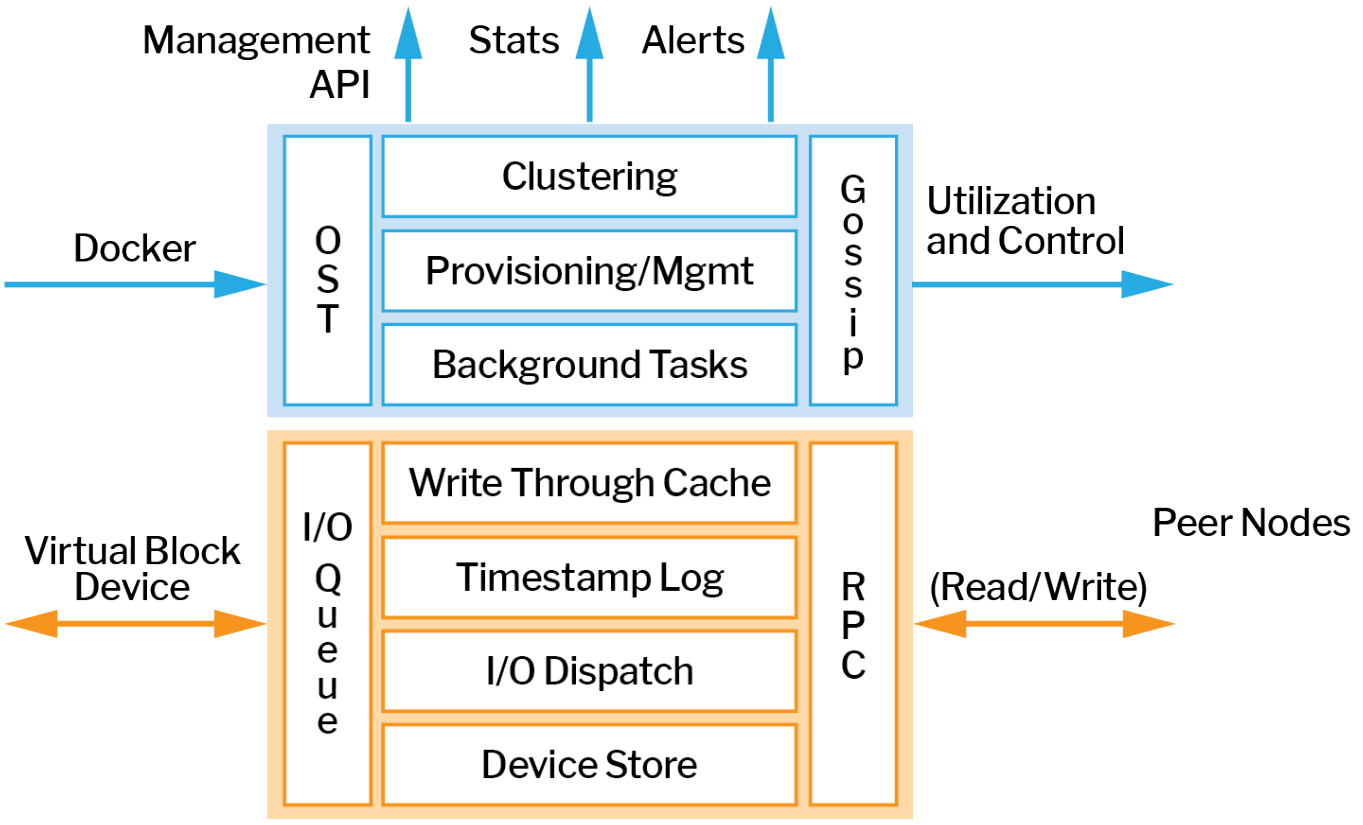
- Provides programmatic control on available storage resources. You can create and directly consume volumes and other stateful services by using the DC/OS tool chain.
- Is radically simple. You deploy PX-Enterprise just like any other container. PX-Enterprise integrates with DC/OS natively and can scale to thousands of nodes with no application disruption

In a nutshell, PX-Enterprise eliminates all the pain points in deploying stateful containerized applications and enables you to build and scale applications without worrying about storage infrastructure.

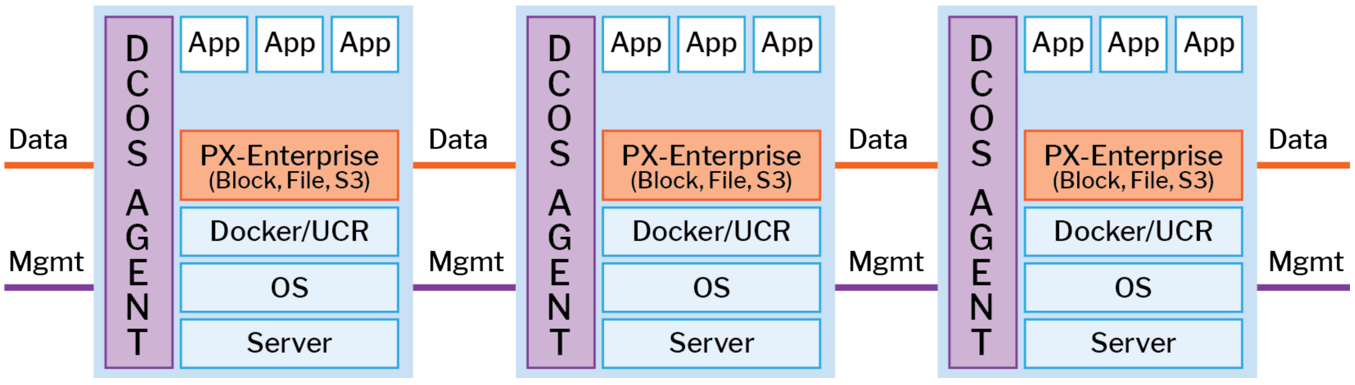
PX-Enterprise is available for free for up to 3 nodes and 1 TB of storage. To learn more, visit https://portworx.com/products/features/.

PX-ENTERPRISE ARCHITECTURE AND PRODUCT FEATURES

PX-Enterprise is built from the ground up as a distributed scale-out block storage layer that can scale to thousands of nodes. Gossip handles node discovery and inventory management. PX-Enterprise maintains the state in a distributed key-value store.



Each container gets a separate block device and separate queues that enable container granular data services and fine-grained quality of service.



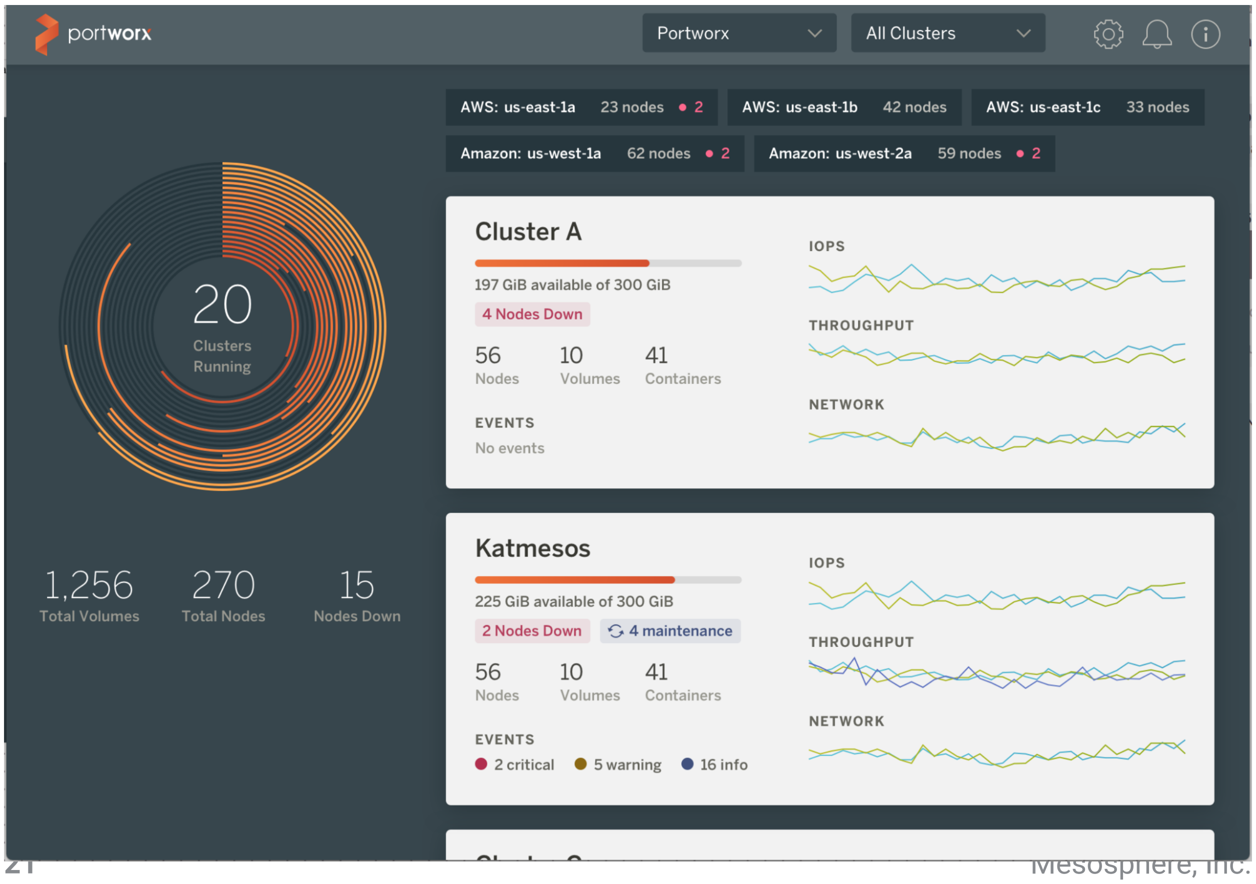
PX-Enterprise features:

- Supports a unified scale-out data layer that provides block, file and s3 interfaces to the applications running on the container host.
- Has a distributed control and data plane. There is no need for dedicated metadata servers. This unique architecture enables PX-Enterprise to scale to large number of nodes with no impact to application performance.
- Supports highly-available data volumes with two or three replicas that can be deterministically placed in a quorum number of nodes. All PX-Enterprise volumes are thinly provisioned and can be expanded on demand with no disruption to the applications.
- Supports up to 16K volumes in a cluster and 64 snapshots per volume. You can create the snapshots manually or schedule them for hourly, daily, weekly or monthly policies.
- Supports shared volumes which enable all containers running on all hosts to access the same block device namespace. This enables shared workflows like Jenkins or other CI/CDs to easily share data between hosts.
- Supports advanced data protection features such as snapshots, multi-cloud backup to AWS S3, Azure Blob, Google Cloud Storage and any object storage solution that supports the Amazon S3 interface.

- Supports data-at-rest and data-in-flight encryption, with seamless integration with key management systems such as Hashicorp Vault, AWS KMS, and Docker Secrets. The encryption is at the granularity of a single container, scaling to hundreds of thousands of containers over a large number of nodes either on-premises or in the cloud.
- Supports the native Docker volume driver and integrates with Docker Volume Driver Isolator (DVDI) through the DVDCLI. This enables an application container running as a Mesos Container to leverage PX-Enterprise data services natively in the DC/OS environment

Is managed through pxctl CLI (at /opt/pwx/bin on each DC/OS agent that the PX-Enterprise container is running), the Lighthouse GUI, and a REST API. You can also invoke the PX-Enterprise CLI from the DC/OS CLI by installing the PX-Enterprise CLI as an add-on to the DC/OS CLI.

The REST API is documented here: http://api.openstorage.org/openstorage/index.html



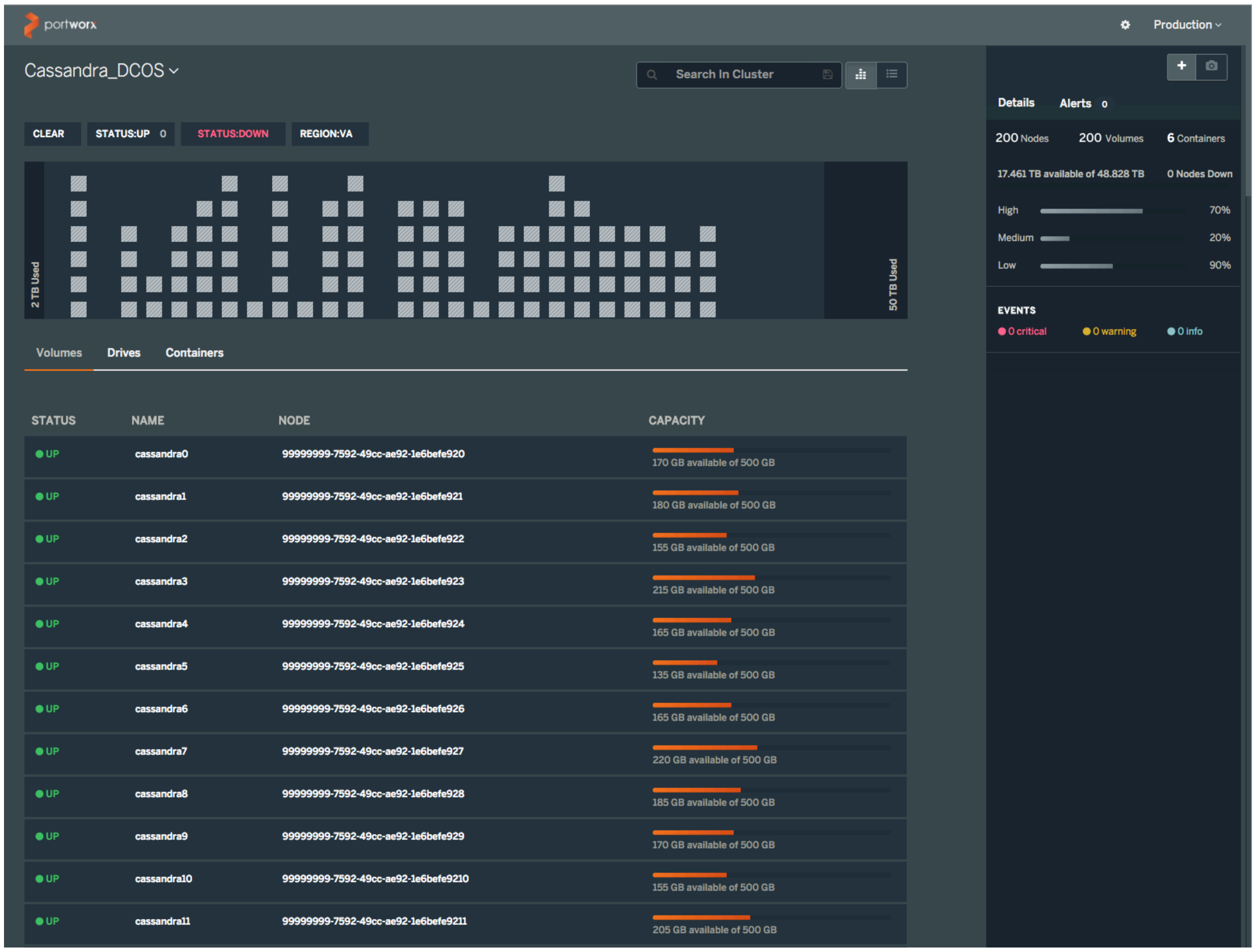
AWS: us-east-1a 23 nodes ● 2
AWS: us-east-1b 42 nodes
AWS: us-east-1c 33 nodes
Amazon: us-west-1a 62 nodes ● 2
Amazon: us-west-2a 59 nodes ● 2

20 Clusters Running

1,256 Total Volumes
270 Total Nodes
15 Nodes Down

Cluster A
197 GiB available of 300 GiB
4 Nodes Down
56 Nodes
10 Volumes
41 Containers
EVENTS
No events

Katmesos
225 GiB available of 300 GiB
2 Nodes Down
4 maintenance
56 Nodes
10 Volumes
41 Containers
EVENTS
● 2 critical ● 5 warning ● 16 info



200 Nodes 200 Volumes 6 Containers

17.461 TB available of 48.828 TB 0 Nodes Down

High	70%
Medium	20%
Low	90%

EVENTS

● 0 critical ● 0 warning ● 0 info

STATUS	NAME	NODE	CAPACITY
● UP	cassandra0	99999999-7592-49cc-ae92-1e6befe920	170 GB available of 500 GB
● UP	cassandra1	99999999-7592-49cc-ae92-1e6befe921	180 GB available of 500 GB
● UP	cassandra2	99999999-7592-49cc-ae92-1e6befe922	155 GB available of 500 GB
● UP	cassandra3	99999999-7592-49cc-ae92-1e6befe923	215 GB available of 500 GB
● UP	cassandra4	99999999-7592-49cc-ae92-1e6befe924	165 GB available of 500 GB
● UP	cassandra5	99999999-7592-49cc-ae92-1e6befe925	135 GB available of 500 GB
● UP	cassandra6	99999999-7592-49cc-ae92-1e6befe926	165 GB available of 500 GB
● UP	cassandra7	99999999-7592-49cc-ae92-1e6befe927	220 GB available of 500 GB
● UP	cassandra8	99999999-7592-49cc-ae92-1e6befe928	185 GB available of 500 GB
● UP	cassandra9	99999999-7592-49cc-ae92-1e6befe929	170 GB available of 500 GB
● UP	cassandra10	99999999-7592-49cc-ae92-1e6befe9210	155 GB available of 500 GB
● UP	cassandra11	99999999-7592-49cc-ae92-1e6befe9211	205 GB available of 500 GB

Basic operational procedure

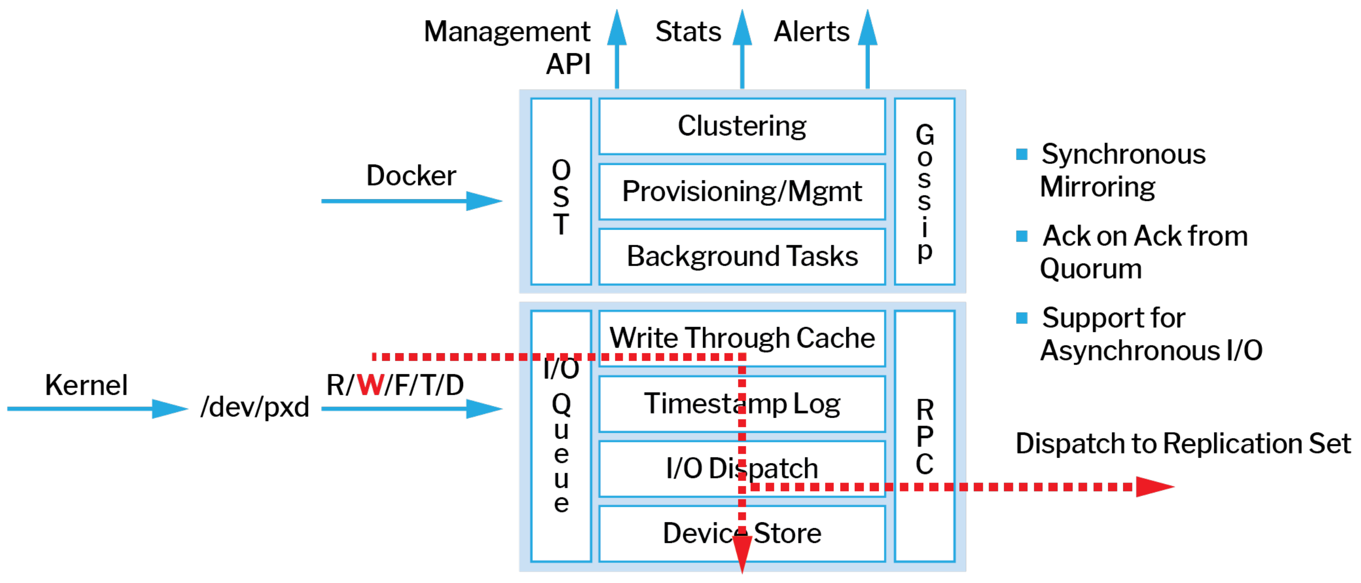
This section documents the basic control path for how a Docker volume controller creates, attaches, uses, and deletes PX-Enterprise volumes. A Docker volume controller uses a subset of the libopenstorage API documented here: http://api.openstorage.org/openstorage/index.html PX-Enterprise is a distributed block volume solution. Each volume is exposed to a Linux host as a local block device. Normally, the output of `lsblk` is:

```
root@ip-172-31-19-148:/home/ubuntu# lsblk
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  128G  0 disk
└─xvda1 202:1    0  128G  0 part /

When a container uses a volume, additional virtual devices
are shown as:
root@ip-172-31-19-148:/home/ubuntu# lsblk
NAME                      MAJ:MIN RM  SIZE RO TYPE
MOUNTPOINT
xvda                      202:0    0  128G  0 disk
└─xvda1                   202:1    0  128G  0 part /
pxd/pxd888797473879456559 251:1    0    1G  0 disk /var/
```

A PX-Enterprise virtual device is then bind-mounted into the container's persistent storage path. Therefore, all IO from a container appears as if it is actually going to a local volume. It is important to understand that PX-Enterprise is not just a connector to externally attached storage devices, but rather a virtual local block device from which it then provides the rest of the SDS functionality.

IO Path



When a PX-Enterprise volume is exposed at the host, it appears as a local block device. This means that as long as the host is alive, the volume is attached to that node.

PX-Enterprise uses the Gossip and RAFT protocols to maintain which nodes are alive and not. Any volume can be attached on any host as long as it is exclusive. The PX-Enterprise control plane decides which nodes are in the cluster and which are out (or dead).

PX-ENTERPRISE AND DC/OS

Portworx and Mesosphere built a joint reference architecture to help customers deploy any data-intensive application on a DC/OS cluster.

PX-Enterprise provides data services for both Mesos containers and Docker containers.

PX-Enterprise is available as a DC/OS Universe package that you can download and install in a single click on all available DC/OS Agents.

Follow the steps in the next section to install PX-Enterprise from the DC/OS Universe.

Installing PX-Enterprise as a DC/OS service

The Portworx DC/OS framework deploys Portworx PX-Enterprise as DC/OS service. This framework:

- Deploys PX-Enterprise on all available DC/OS agents.
- Deploys Lighthouse, a rich GUI that helps manage PX-Enterprise.
- Integrates PX-Enterprise as a DC/OS service that you can monitor by using the DC/OS GUI.

You can also use PX-Enterprise to provision volumes on DC/OS using the Docker Volume Driver Interface (DVDI).

Minimum requirements for running PX-Enterprise:

- Linux kernel 3.10 or greater
- Docker 1.12 or greater

- Minimum resources per server:
  - 4 CPU cores
  - 4 GB RAM
- Recommended resources per server:
  - 12 CPU cores
  - 16 GB RAM
  - 128 GB Storage
  - 2 x 10 Gb Ethernet NIC
- Maximum nodes per cluster:
  - Up to 1000 nodes for the Enterprise License
- Network ports:
  - Open ports 9001 - 9004 for internal network traffic between nodes running PX-Enterprise.
  - Ensure that the ports are open in the local firewall of each node.
  - Bond (or aggregate) the network ports so that any failure in the NICs are tolerated.

You can deploy PX-Enterprise on any server or cloud instance that meets the minimum requirements above. PX-Enterprise does not have any requirements for a meta-data node and consumes a very small amount of host resources so it can run in a converged manner closer to the applications, delivering bare-metal performance with low overhead.

Follow this link to learn how to deploy PX-Enterprise with a single-click on your DC/OS cluster via the DC/OS Universe package: https://docs.portworx.com/scheduler/mesosphere-dcos/install.html.

Best practices for deploying PX-Enterprise on DC/OS

PX-Enterprise deploys as a container in the DC/OS task agents.

- Meet the recommended requirements, outlined here, before deploying PX-Enterprise.
- Ensure that there are raw disks available that do not have any pre-existing filesystems on them so PX-Enterprise can include them.
- For best data protection, create PX-Enterprise Volumes with a minimum replication factor of two, so there are two copies for each block of data. Use this command:

```
sudo /opt/pwx/bin/pxctl volume create dcosvol --size=10 —
repl=2
```

You can create PX-Enterprise volumes and pass all the options for those volumes by using the DC/OS Universe package.

- PX-Enterprise can place replica nodes in different racks if the rack information is passed along. PX-Enterprise parses the rack information from the 'racks' variable http://mesos.readthedocs.io/en/0.22.2/attributes-resources/. Or, use the --rack option when creating a volume with the pxctl volume create commands. For information, see CLI Reference–Volume.

PX-Enterprise configuration through DVDCLI

PX-Enterprise supports Docker Volume Plugins. Mesosphere DC/OS can provision storage to any container that supports Docker Volume Plugins through the Docker Volume Driver Isolator and Docker Volume Driver CLI.

For example, someone with an administrator role can provision a PX-Enterprise volume with the following command

```
sudo /opt/mesosphere/bin/dvdcli create  --
volumedriver="pxd" —volumename="testdvdclivol"

INFO[0001] testdvdclivol
```

That command creates a volume "testdvdclivol" in the underlying PX-Enterprise cluster. A volume list command using the pxctl CLI displays the following.

```
sudo /opt/pwx/bin/pxctl volume list
ID    NAME   SIZE  HA   SHARED     ENCRYPTED  IO_PRIORITY
      SCALE  STATUS
355705327512613299 testdvdclivol 1 GiB 1     no    no
      LOW        0     up - detached
```

PX-Enterprise and DC/OS external persistent volumes

You can mount PX-Enterprise volumes as external persistent volumes for DC/OS that applications running in the DC/OS agents can consume. These frameworks enable you to mount PX-Enterprise volumes as persistent volumes in the DC/OS agents and enable applications to consume them as block devices. For information, see: https://docs.portworx.com/scheduler/mesosphere-dcos/portworx_volumes.html

```
{
  "id": "/mysql",
  "instances": 1,
  "cpus": 0.5,
  "mem": 256,
  "backoffSeconds": 1,
  "backoffFactor": 1.15,
  "maxLaunchDelaySeconds": 3600,
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "mysql:5.7.12",
      "network": "BRIDGE",
      "portMappings": [
        {
          "containerPort": 3306,
          "hostPort": 32000,
          "servicePort": 10002,
          "protocol": "tcp",
          "name": "default"
        }
      ],
      "privileged": false,
      "parameters": [
        {
          "key": "volume-driver",
          "value": "pxd"
        },
        {
          "key": "volume",
          "value": "size=100,repl=2,name=mysql_vol:/var/lib/mysql"
        }
      ],
      "forcePullImage": false
    }
  },
  "upgradeStrategy": {
    "minimumHealthCapacity": 1,
    "maximumOverCapacity": 1
  },
  "unreachableStrategy": {
    "inactiveAfterSeconds": 300,
    "expungeAfterSeconds": 600
  },
  "killSelection": "YOUNGEST_FIRST",
  "requirePorts": true,
  "env": {
    "MYSQL_ROOT_PASSWORD": "password"
  }
}
```

DATA INTENSIVE APPS WITH PX-ENTERPRISE ON DC/OS

MySQL

MySQL is a popular Open Source Relational Database Management System (RDBMS). It is one of the most common stateful apps to run in containers.

MySQL using Portworx Volumes on DC/OS

You can launch a MySQL instance using marathon on DC/OS and ask it to use PX-Enterprise volumes for its persistent data. You can also pass in additional options for the volume like size and replication factor when starting the service. If a volume with the provided name does not exist PX-Enterprise will create it with the specified options. So you don't need to pre-provision volumes when starting a service.

Here is an example on how to start MySQL with a 100GB PX-Enterprise volume with a replication factor of 2. The volume will be mounted under "/var/lib/mysql" once the container starts up.

MySQL failover handling with DC/OS and PX-Enterprise

Since PX-Enterprise volumes can replicate across multiple nodes, there is no “stickiness” required for tasks. If a PX-Enterprise volume is replicated to multiple nodes, the service which uses it can launch on any of the nodes in the PX-Enterprise cluster.

If a node running a task (which uses a PX-Enterprise volume with a replication factor greater than 1) crashes, the same task can be bought online on any other node in the PX-Enterprise cluster. This is not possible with local volumes because the same data is not available on any other node.

Mesosphere and Portworx have developed dcos-commons frameworks that greatly simplify deployment of data intensive apps and provide external volume support to these apps running in containers on a DC/OS platform. This section describes how you can easily deploy Cassandra on DC/OS, with PX-Enterprise as the container data layer.

Apache Cassandra

Apache Cassandra is a free and open-source highly distributed NoSQL database management software designed to handle large amounts of data across large number of commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters and availability zones, on different cloud providers, with asynchronous masterless replication allowing low latency operations for all clients.

Cassandra is used at companies known for their large scale operations, such as Facebook, CERN, and Instagram. As a result, Cassandra is responsible for some of humanity's most awesome accomplishments.

PX-ENTERPRISE AND DC/OS WITH CASSANDRA

Cassandra with PX-Enterprise DC/OS Cassandra service

The Mesosphere DC/OS Cassandra service makes deploying and scaling containerized Cassandra clusters effortlessly easy, eliminating the many complexities of deploying a Cassandra cluster. For more information about the DC/OS Cassandra service, see https://docs.mesosphere.com/service-docs/cassandra/.

PX-Enterprise enables customers running data intensive applications such as Cassandra to get container granular data services and get the most out of their container infrastructure.

As an example of how external storage works with Cassandra, here is how Portworx's cassandra-px framework allows for easy deployment of external persistent volumes in DC/OS for Cassandra.

Before proceeding further, ensure that PX-Enterprise is installed on the DCOS cluster, as documented in the previous sections.

To deploy Cassandra with PX-Enterprise volumes on DC/OS:

1. Run the following command to install the cassandra-px framework::

```
$ dcos package install --yes cassandra-px
```

You can also install the package by clicking the **Install** button on the WebUI next to the service and then clicking **Install Package**.

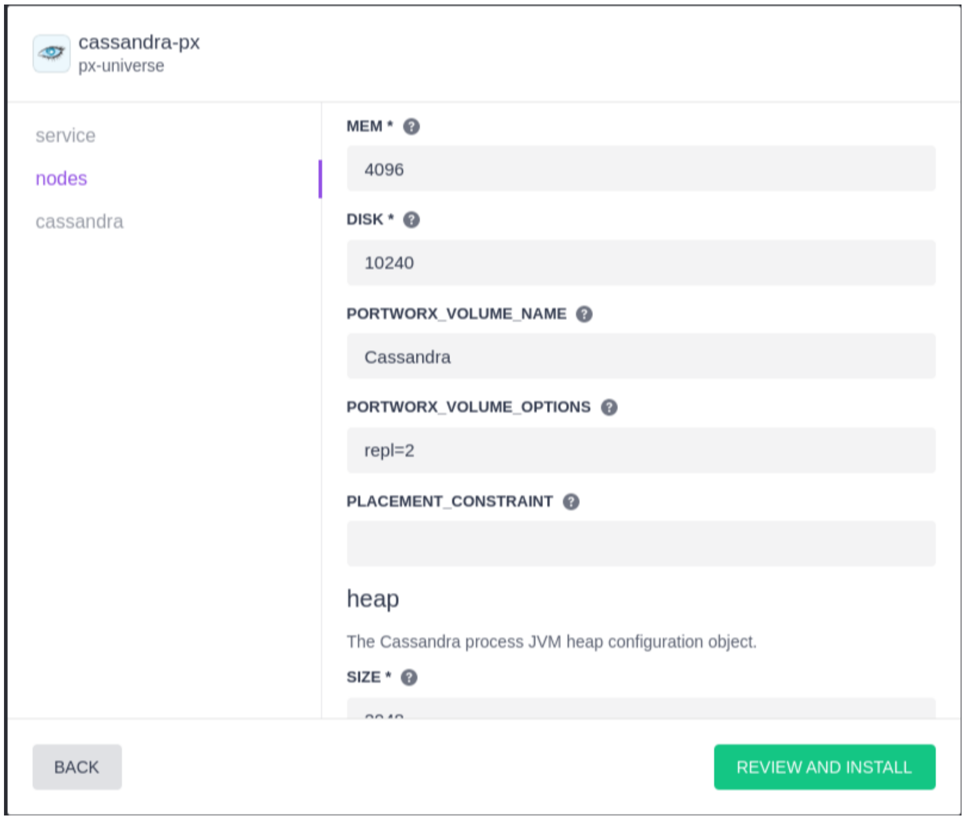
2. After the above command executes, the cassandra-px service is available as a package in DC/OS Universe as shown in the next section.

INSTALLATIONS

Advanced install

To install the example Casandra service with more advanced options, click the **Install** button next to the package on the DC/OS UI in the Universe section and then click **Advanced Installation**.

An advanced install includes options to change the service name, volume name, and volume size, and provide any other options that you need to pass to the Docker volume driver. This example creates a volume with a replication factor of 2, with the repl=2 key/value pair in the PORTWORX_VOLUME_OPTIONS field. There are also options to configure other Cassandra related parameters, including the number of Cassandra nodes, as shown in the following screen sample.



MEM *
4096

DISK *
10240

PORTWORX_VOLUME_NAME
Cassandra

PORTWORX_VOLUME_OPTIONS
repl=2

PLACEMENT_CONSTRAINT

heap

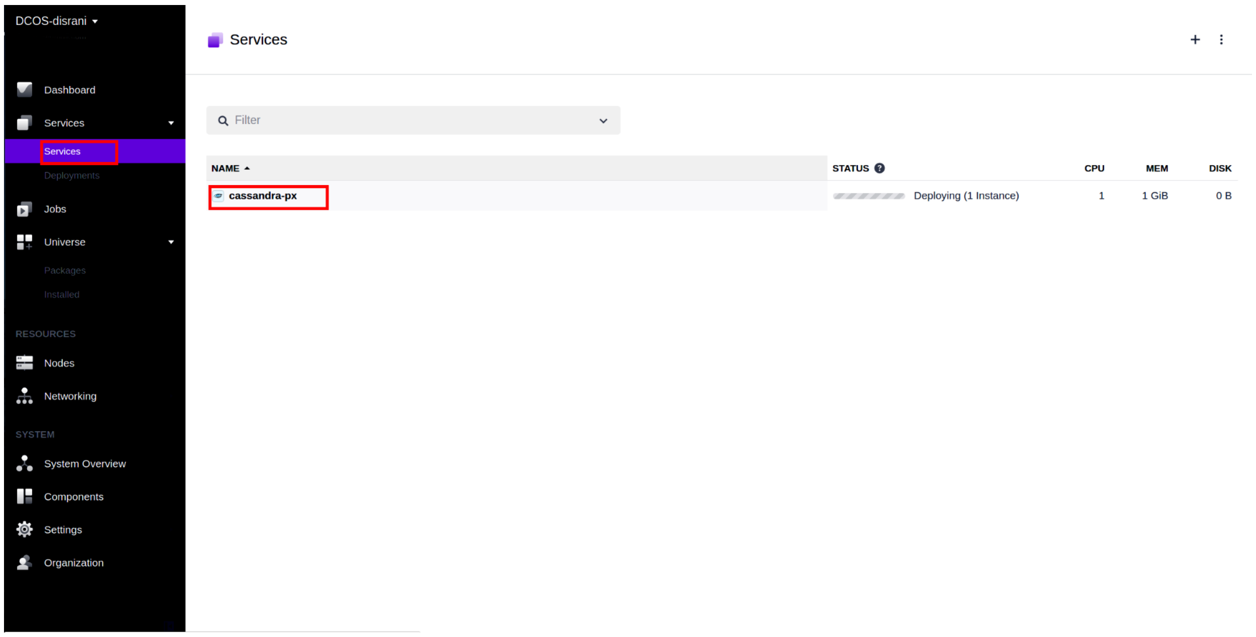
The Cassandra process JVM heap configuration object.

SIZE *

To start the service installation, click **Review and Install** and then click **Install.**

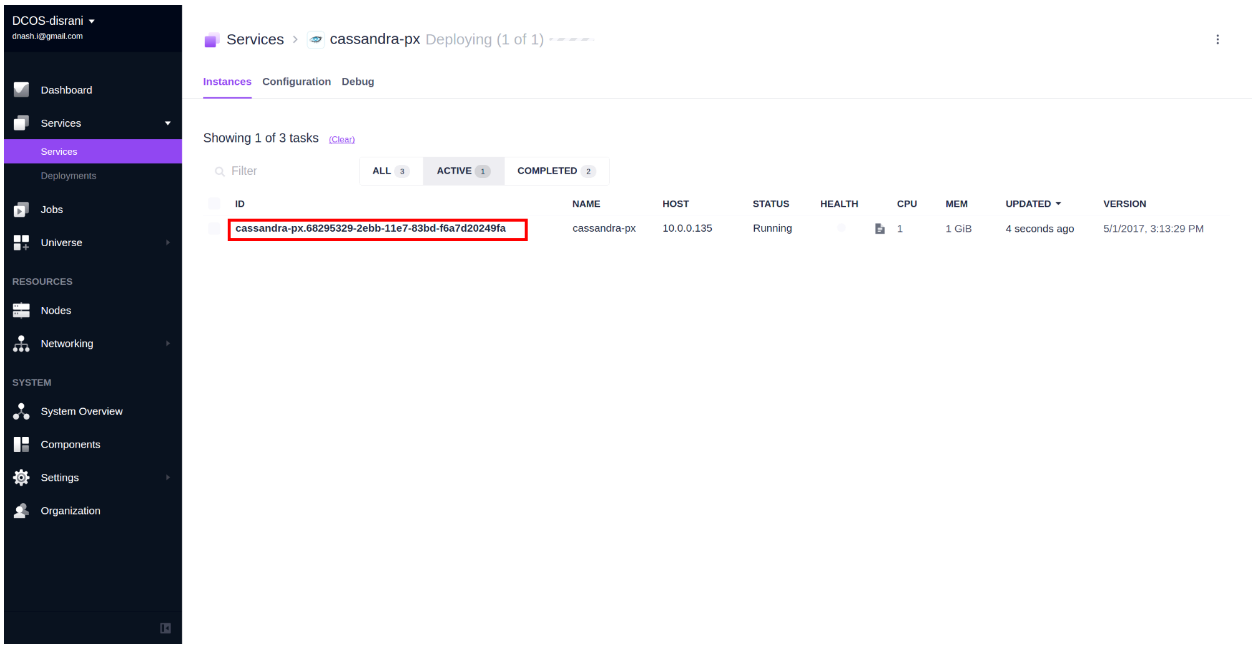
Install status

Once the install starts, you can monitor it on the Services page.



NAME	STATUS	CPU	MEM	DISK
cassandra-px	Deploying (1 Instance)	1	1 GiB	0 B

To check the status of the service deployment in the nodes, click the Cassandra service. There will be one service for the scheduler and one service each for the Cassandra nodes.



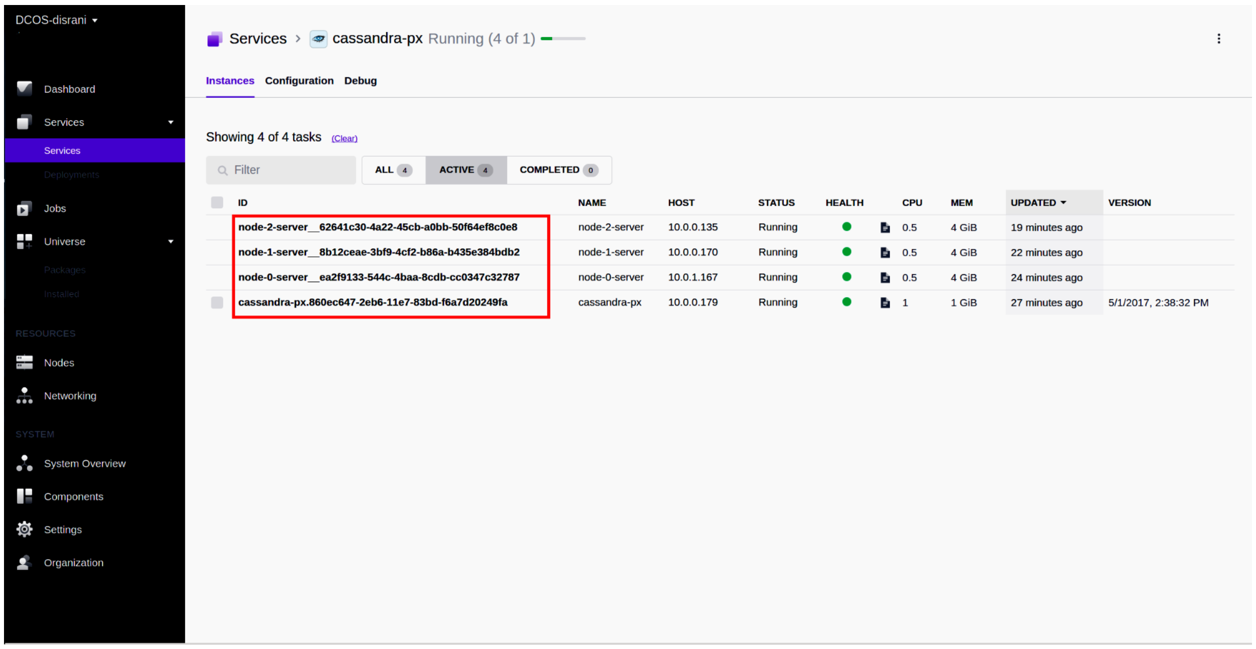
Services > cassandra-px Deploying (1 of 1)

Showing 1 of 3 tasks (Clear)

ALL 3 | ACTIVE 1 | COMPLETED 2

ID	NAME	HOST	STATUS	HEALTH		CPU	MEM	UPDATED	VERSION
cassandra-px.68295329-2ebb-11e7-83bd-f6a7d20249fa	cassandra-px	10.0.0.135	Running			1	1 GiB	4 seconds ago	5/1/2017, 3:13:29 PM

When the Scheduler service and the Cassandra nodes are in Running (green) status, the Cassandra cluster is ready to start.



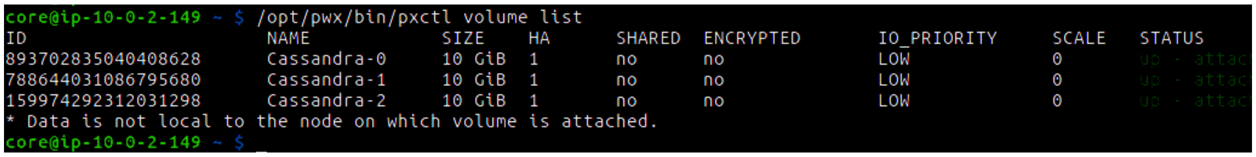
Services > cassandra-px Running (4 of 1)

Showing 4 of 4 tasks (Clear)

ALL 4 | ACTIVE 4 | COMPLETED 0

ID	NAME	HOST	STATUS	HEALTH		CPU	MEM	UPDATED	VERSION
node-2-server__62641c30-4a22-45cb-a0bb-50f64ef8c0e8	node-2-server	10.0.0.135	Running	●		0.5	4 GiB	19 minutes ago	
node-1-server__8b12ceae-3bf9-4cf2-b86a-b435e384bdb2	node-1-server	10.0.0.170	Running	●		0.5	4 GiB	22 minutes ago	
node-0-server__ea2f9133-544c-4baa-8cdb-cc0347c32787	node-0-server	10.0.1.167	Running	●		0.5	4 GiB	24 minutes ago	
cassandra-px.860ec647-2eb6-11e7-83bd-f6a7d20249fa	cassandra-px	10.0.0.179	Running	●		1	1 GiB	27 minutes ago	5/1/2017, 2:38:32 PM

Checking the PX-Enterprise cluster shows multiple volumes that were automatically created using the options provided during install, one for each node of the Cassandra cluster. In this case, a three-node Cassandra cluster is created. Note that the framework created Cassandra-0, Cassandra-1, and Cassandra-2 volumes automatically for each Cassandra node.



```
core@ip-10-0-2-149 ~ $ /opt/pwx/bin/pxctl volume list
ID                      NAME            SIZE    HA      SHARED  ENCRYPTED       IO_PRIORITY     SCALE   STATUS
893702835040408628      Cassandra-0     10 GiB  1       no      no              LOW             0       up - attac
788644031086795680      Cassandra-1     10 GiB  1       no      no              LOW             0       up - attac
159974292312031298      Cassandra-2     10 GiB  1       no      no              LOW             0       up - attac
* Data is not local to the node on which volume is attached.
core@ip-10-0-2-149 ~ $ _
```

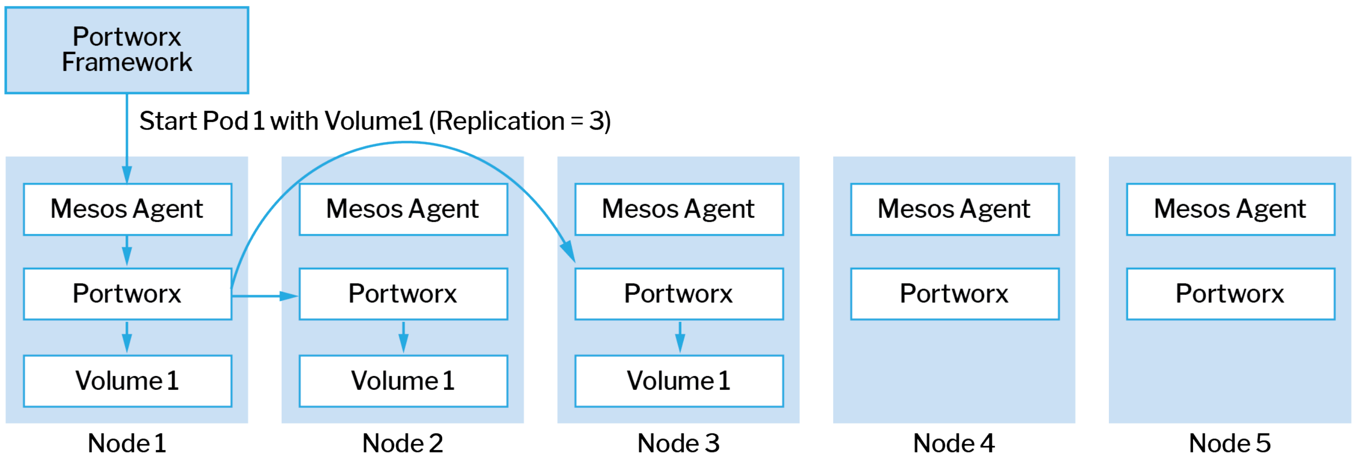
Running the the “dcos service” command shows the cassandra-px service in ACTIVE state with 3 running tasks, one for each Cassandra node.

```
$ dcos service
NAME                                HOST                        ACTIVE  TASKS
 CPU     MEM     DISK   ID
cassandra-px                     10.0.0.179                      True     3
   1.5   12288.0   0.0   5c6438b2-1f63-4c23-b62a-ad0a7d354a91-0115
marathon                         10.0.4.21                       True     1
   1.0    1024.0   0.0   01d86b9c-ca2c-4c3c-9d9f-d3a3ef3e3911-0001
metronome                        10.0.4.21                       True     0
   0.0     0.0     0.0   01d86b9c-ca2c-4c3c-9d9f-d3a3ef3e3911-0000
```

PX-Enterprise data convergence for Cassandra volumes

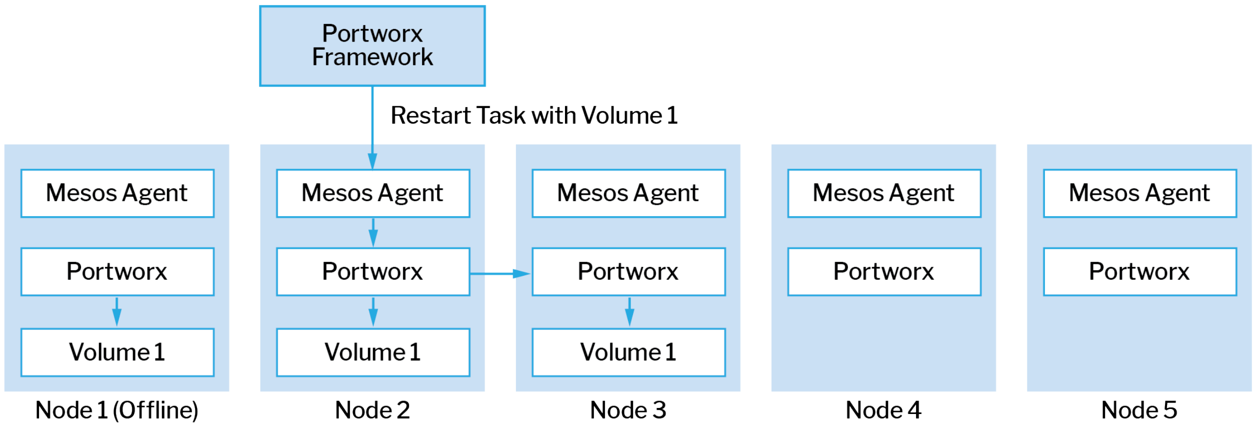
PX-Enterprise supports data convergence by enabling Cassandra instances to always run locally on the node where data is located.

When the Cassandra service is first launched, it creates the required PX-Enterprise volumes. These volumes are created with data local to the node where they are first launched.



Volume Creation on First Launch of Pod

On subsequent launches of the same pod, the framework queries PX-Enterprise to determine where the data for the volume resides and then uses that data to determine where to launch the pod. This query can be made to any node in the PX-Enterprise cluster, since all nodes have information about the location of volumes.



Restart Task on Any Node Where Volume Is Local

Scaling the number of nodes

Before scaling the number of nodes, ensure that there are available resources and nodes in the cluster and that all nodes have PX-Enterprise installed.

Scaling the Cassandra cluster is quick and easy. Go to the Cassandra service page, click on the three dots on the top-right corner of the page, select **nodes**, scroll down, and then increase the nodes parameter.

Click **Review and Run** and then click **Run Service**. The service scheduler restarts with the updated node count and creates more Cassandra nodes.

Cassandra node failover handling with DC/OS and PX-Enterprise

Because PX-Enterprise volumes can replicate across multiple nodes, there is no “stickiness” required for tasks. If a PX-Enterprise volume is replicated to multiple nodes, the task which uses it can launch on any of the nodes in the PX-Enterprise cluster. PX-Enterprise makes an effort to start a failing task on a node which has data local to it, to provide convergence.

If a node running a task (which uses a PX-Enterprise volume with a replication factor greater than 1) crashes, the same task can be bought online on any other node in the PX-Enterprise cluster. This is possible because the framework determines whether the resources (CPU, memory, ports and PX-Enterprise Volumes) have any stickiness to the node that crashed. This was not possible with local volumes because the same data could not be available on any other node.

For the Cassandra service this failover handling has a major advantage in that a new node does not have to be repaired from scratch with all the application data of the old node. The new node comes online with the identity of the old node and only needs to repair the data for the time that it was offline from the Cassandra cluster. This same advantage is seen in other applications using this framework.

MORE RESOURCES FOR PX-ENTERPRISE ON DC/OS

Portworx on DC/OS overview: https://docs.portworx.com/scheduler/mesosphere-dcos/

Install Portworx on DC/OS: https://docs.portworx.com/scheduler/mesosphere-dcos/install.html

CONCLUSION

By combining Mesosphere DC/OS and Portworx PX-Enterprise, enterprises are able to deploy and run data-intensive infrastructure applications in a fully automated, DevOps friendly manner. DC/OS makes it easy to build and run modern distributed applications in production at scale, by pooling resources across an entire datacenter or cloud. By adding Portworx PX-Enterprise to DC/OS, enterprises are able to solve the operational and data management problems they encounter when running stateful applications on DC/OS. We hope this reference architecture has given you the information and tools you need to try it yourself.

MESOSPHERE