HPE Ezmeral Container Platform

## Planning Overview

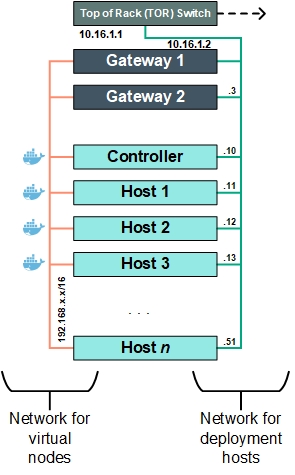
This article provides a high-level overview of the items to consider when planning an HPE Ezmeral Container Platform deployment. These items include:

### **Networking Considerations**

It describes the differences of using a routable (public) network for the virtual nodes (Docker containers) versus using a non-routable (private) network where Gateway hosts provide proxy access to the virtual nodes through port mappings.

### **Networks and Subnets**

HPE Ezmeral Container Platform operates on two networks, as shown here:

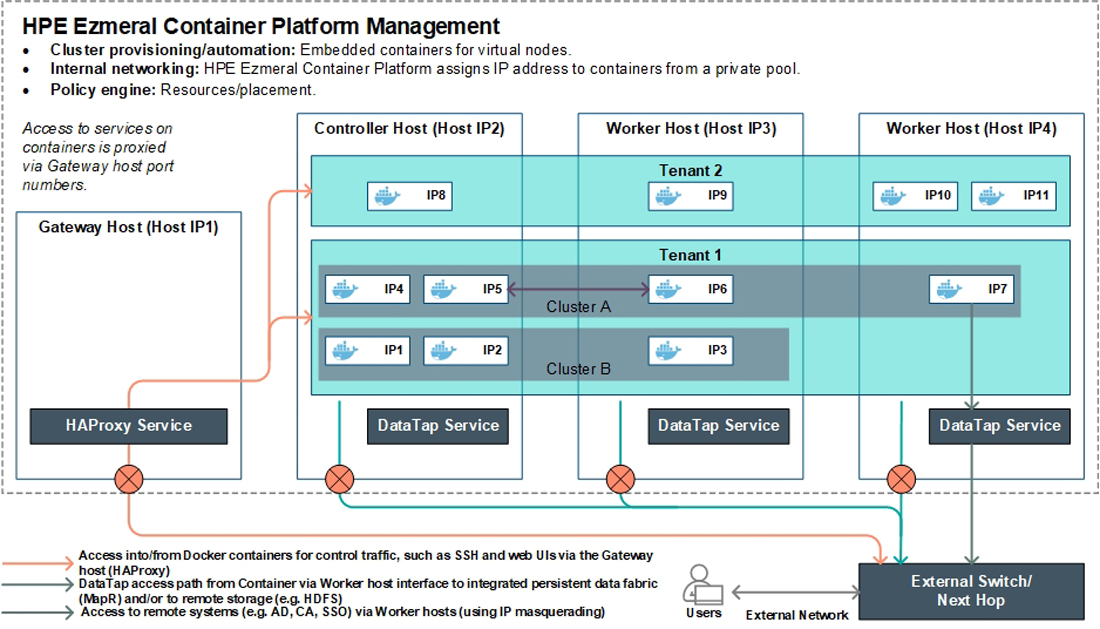


The two networks are laid out as follows:

* **Network for the Controller, Worker, and Gateway hosts:** This network must be both routable (public) and part of the organization's network that is managed by that organization's IT department.
* **Network for virtual nodes (Docker containers):** HPE Ezmeral Container Platform creates and manages this network, which can be either public (routable) or private (non-routable). For Kubernetes, Canal is used as the Container Network (CN) Network Provider. The container network is typically private (non-routable) instead of public (routable).

#### Private (Non-Routable) Virtual Node Network

Private, non-routable virtual node networks keep the virtual node IP addresses private and hidden within the private network. As described in Gateway Hosts and Load Balancing, the Gateway host(s) proxy service endpoint ports. IP masquerading replaces the IP addresses of outgoing and incoming packets when the containers are in a private, non-routable network, as shown here.



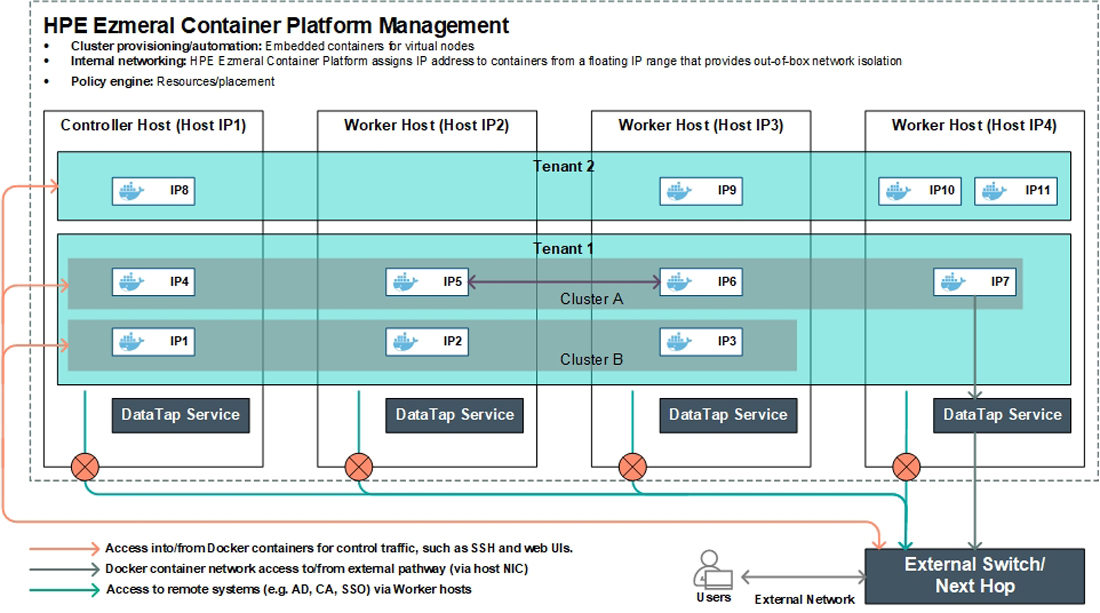
HPE Ezmeral Container Platform is deployed on a set of hosts. Each host has an IP address and FQDN such as **Host IP1**, **Host IP2**, etc. Hosts are typically deployed as one or more rack(s) of servers that are connected to an external switch for access to other subnets in the organization (such as end user network etc.).

HPE Ezmeral Container Platform provisions clusters of embedded, fully-managed Docker containers. Each cluster spins up within a tenant and receives distinct assigned IP addresses and FQDNs from a user-provided IP range, which appear in the diagram above as **IP1**, **IP2**, etc. Kubernetes supports network isolation via network policies.

End-user access to services in the containers (such as SSH or web applications) is routed through a Gateway host that runs the HAProxy service. This access is purely for control traffic. All other traffic, including access to remote HDFS or other enterprise systems such as Active Directory (AD), MIT KDC (Kerberos provider), SSO (Identity providers), and Certificate Authority (CA), is performed via the host network interface masquerading, as opposed to the Gateway host port proxying.

#### Public (Routable) Virtual Node Network

**Note:** Most deployments will not use routable container networks; however, this feature is supported if desired.



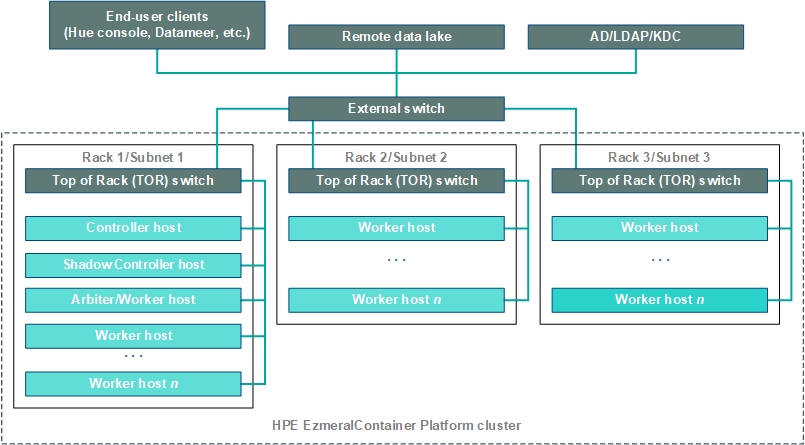
The above diagram depicts a deployment where a routable IP range is used for containers. Unlike private non-routable virtual node network configuration, all of the hosts must be in the same subnet. All the key functionality is identical to the recommended approach for a non-routable private IP range. This approach allows the Docker containers to directly access the external network via the network interface connector (NIC) on the hosts where they reside.

### Multiple Subnets

HPE Ezmeral Container Platform can be deployed across multiple subnets as long as it is configured to use private non-routable virtual. This diagram displays a sample deployment using multiple subnets.

When multiple subnets are used:

* The hosts may be located on-premises and/or in a public cloud. For example, hosts can reside on multiple racks and/or can be virtual machines residing on cloud-based services (such as AWS, Azure, or GCP).
* If the deployment includes cloud-based hosts, then the container network must be private and non-routable.
* If the Controller and Worker hosts are on different subnets, then the path MTU settings must be the same for both subnets.
* If the Controller and Worker hosts are on a single subnet and the Gateway host(s) are on the different subnet, then the subnet with the Gateway host(s) may use an MTU setting that is lower than or equal to the MTU setting on the other subnet with no further action needed. If the MTU of the Gateway host is larger, then it must be at least 1,000 bytes larger than the MTU setting of the other subnet.
* The subnet(s) used by Gateway hosts may have different path MTU settings, subject to the above.
* If the Controller and Shadow Controller are on one subnet, the Worker hosts on a second subnet, and the Gateway hosts on a third subnet, then each of the hosts must have the same path MTU setting.



When setting up a Kubernetes cluster, you will provide a CIDR IP range that is typically in a private (non-routable) IP range. Both private (non-routable) and public (routable) container networks are supported. Here are some areas of consideration when deciding whether to use a private or public container network.

|  |  |  |
| --- | --- | --- |
|  | ****Benefits**** | ****Considerations**** |
| **Private, non-routable container network** | No dependency on network team, such as for getting any additional subnets beyond the base set of hosts that each have an IP address and FQDN.  Create and use an arbitrarily large private IP range with no need for planning. A single /16 private subnet will support ~16000 containers. It is not normally possible for networking teams to provide such a large single range; they will probably provide multiple different subnets that will add administrative overhead for both the Platform Administrator and the network team.  Private IP ranges make security compliance significantly easier because it simplifies controlling the specific container service endpoints that need to be exposed to the end user network.  Many organizations treat any virtual machine or container with a routable IP address similarly to a physical machine. Deleting these hosts requires approval and signoff. This goes against the BDaaS user experience where users want to shrink and delete clusters at will.  Hosts can span multiple subnets, as described in Multiple Subnets. | Each Gateway host (or common FQDN) can support up to 40,000 service endpoints. You may reserve one or more port range(s) from 10,000 to 50,000. Ports reserved for HPE Ezmeral Container Platform may not be used for any other purpose.  Only control traffic (e.g. Console Hue, SSH, scp, etc.) goes through the Gateway host. Data I/O does not go through the Gateway host.  If you add new services to an existing virtual cluster, then you need to follow the procedure described in [Adding a Cluster Service](https://docs.containerplatform.hpe.com/53/reference/epic/common-cluster-operations/cluster-services/Adding_a_Cluster_Service.html) to expose the newly-added services to users.  If the port number changes for a service endpoint on an existing cluster, then the existing HTTP link to that service will no longer work. You will need to manually enter the new port number in the URL address. |
| **Public, routable container network** | All service endpoints are directly accessible using the IP address of the virtual node. | Make sure that the pool of reserved IP addresses is large enough to support both current and projected future needs.  All hosts must be in the same subnet. |

The cluster networking uses a VXLAN overlay network. The underlying physical network must be implemented with this in mind. Hewlett Packard Enterprise recommends the following best practices:

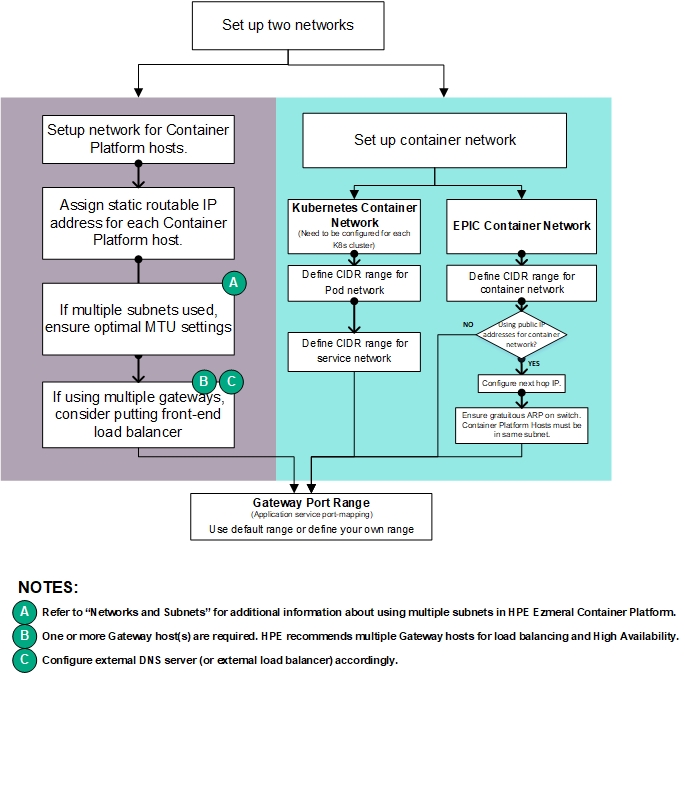
* Deploying 25Gbps Ethernet adapters on all hosts.
* Using NIC cards that support hardware packetization offload.
* Use the largest common MTU size possible for all the hosts on the same VLAN.
* Use LACP.

In a large complex scaled distributed environment, factors such as pod scheduling characteristics across these physical hosts and application design can impact your network operation.

## **Network Planning**

Once you understand how public and private networks are used for virtual nodes, you can use the flowchart below to determine which type of network makes the most sense for your deployment.

This diagram provides a workflow for making decisions about how to configure the virtual node (container) and host networks. This information does not apply to Kubernetes hosts and/or pod networking.



**Note:** Most deployments will not use routable container networks; however, this feature is supported if desired.

## **Storage**

This article describes the available storage schemas and the key advantages and considerations of each schema and the various storage usages and how datasets are made available to the containerized clusters.

### Container Local Data Storage

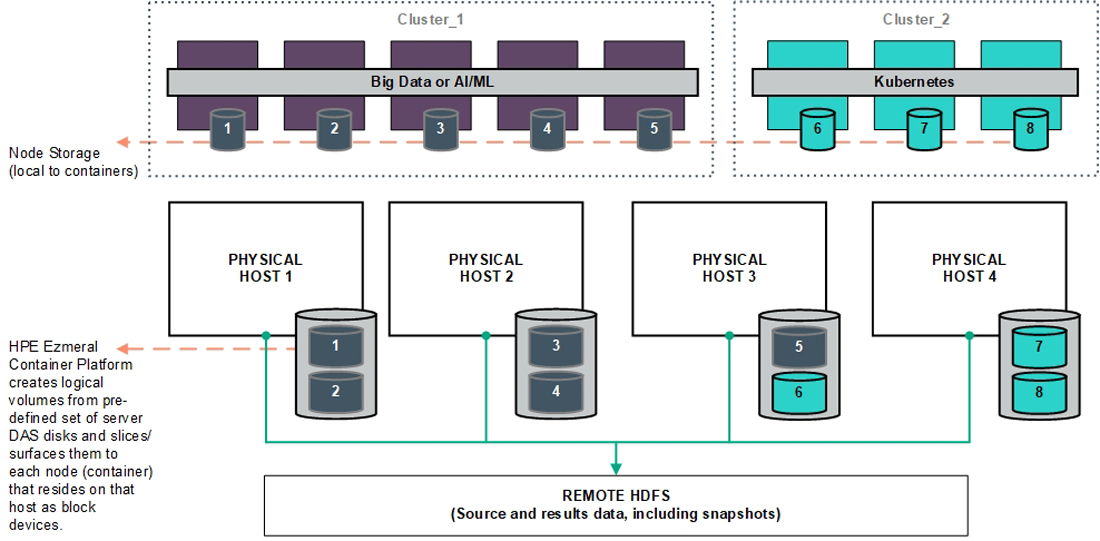
HDFS is provisioned within the Docker containers that comprise a virtual Hadoop cluster when that cluster is created. The underlying storage for the HDFS data nodes in the containers resides on local disks in the physical servers hosting those containers. The deployment refers to the set of local disks as node storage. When using HDFS storage in a virtual cluster, the data does not persist beyond the life of the virtual cluster.

### Ephemeral Storage

Ephemeral storage is built from the local storage in each host. It is used for the disk volumes that back the local storage for each virtual node. Installing a host reserves a subset of the local disks on that host for node storage. Physical Linux® volumes are created on those disks and then used to create a Linux volume group. A Linux logical volume is then created from this Linux volume group. This Linux logical volume is assigned to the Linux Docker subsystem, which in turn uses portions of the logical volume to the containers running on that host for use as local storage within those containers.

### Node Storage

Node storage (referred to as *ephemeral storage* in Kubernetes clusters) is built from the local storage in each host and is used for the disk volumes that back the local storage for each virtual node. Using SEDs (Self-Encrypting Drives) will ensure that any data written to node storage is encrypted on write and decrypted on read by the OS. A tenant can optionally be assigned a quota for how much storage the nodes in that tenant can consume.



Virtual nodes/containers running on public cloud VMs (such as AWS EC2) utilize storage within the instance (such as AWS Elastic Block Storage, or EBS) as node storage.

### Persistent Storage

Deploying a persistent data fabric is supported on the local disks within the hosts. This local storage can serve as either HDFS storage or as persistent volumes for Kubernetes clusters. Persistent volumes for Kubernetes stateful clusters are seamlessly available either from the native persistent data fabric or Nimble Storage using the storage interface driver (CSI) that is deployed during cluster creation.

### Tenant/Project Storage

**Note:** This article uses the term "tenant" to refer to tenants and AI/ML projects in both Kubernetes and legacy EPIC.

Tenant storage is an optional storage location that is shared by all nodes within a given tenant. The Platform Administrator configures tenant storage during installation and can change it at any time thereafter. Tenant storage can be configured to use either a local HPE Ezmeral Data Fabric installation (configured on the host storage) or a remote HDFS or NFS system. Alternatively, you can create a tenant without dedicated storage.

**Note:** If all tenants are created using the same tenant storage service settings, then no tenant can access the storage space of any other tenant.

When a new tenant is created, that tenant automatically receives a DataTap called **TenantStorage** that points at a unique directory within the tenant storage space. This DataTap can be used in the same manner as other DataTaps, but it cannot be edited or deleted. This does not apply if tenant storage has not been defined (meaning that you selected **None** for Tenant Storage during installation. The **TenantStorage** DataTap points at the top-level directory that a tenant can access within the Tenant Storage service. The Tenant Administrator can create or edit additional DataTaps that point at or below that directory.

If the tenant storage is based on a local HDFS, then the Platform Administrator can specify a storage quota for each tenant. The HDFS back-end is used to enforce this quota, meaning that the quota applies to storage operations that originate from either the DataTap browser or the nodes within that tenant.

Root tenant storage folders are created under the deployment global tenant storage root. For example, given a global tenant storage root of /a/b, the tenant-specific tenant storage root directories will be /a/bb/1 for Tenant 1 and /a/b/2 for Tenant 2.

You may create DataTaps that point to any subdirectory within the global tenant storage root, so long as that location cannot access another tenant's tenant storage root directory, nor the global tenant storage root. For example:

You could create a DataTap in Tenant 1 that points to /a/b/SharedStorage, because that directory is not part of any existing tenant's Tenant Storage root.

You will also be able, as Tenant 2, to create another DataTap that points to /a/b/SharedStorage, thereby allowing data sharing between Tenant 1 and Tenant 2.

**Note:** Tenant 2 cannot create a DataTap to the /a/b/1/SharedStorage directory, because the /a/b/1 directory is the root tenant storage directory for Tenant 1.

Users who have a Tenant Administrator role may view and modify detailed DataTap information. Members may only view general DataTap information and are unable to create, edit, or remove a DataTap.

**Note:** Data conflicts may occur if more than one DataTap points to a location being used by multiple jobs at once.

### HPE Ezmeral Data Fabric

Disks located across the hosts provide an integrated scale-out, edge-ready persistent HPE Ezmeral Data Fabric. This data fabric effectively handles the diversity of data types, data access, and ecosystem tools needed to manage data as an enterprise resource regardless of the underlying infrastructure or location.

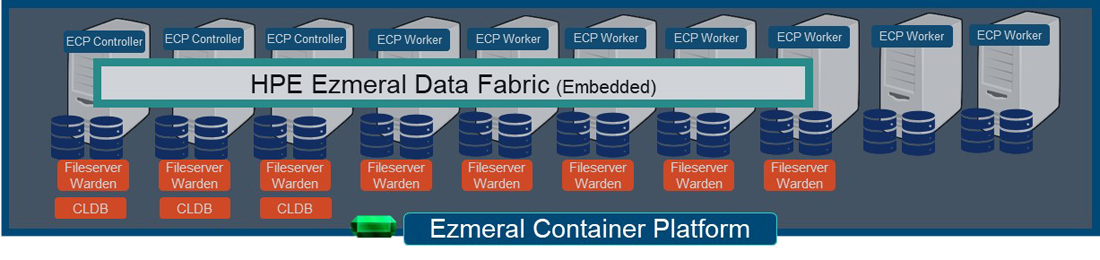
A deployment of HPE Ezmeral Container Platform must use an HPE Ezmeral Data Fabric for persistent storage. The standard way to do this is via HPE Ezmeral Data Fabric on Kubernetes.



HPE Ezmeral Container Platform allows you to create a Data Fabric as an embedded deployment, also called "embedded Data Fabric." If you configure tenant storage to connect to an embedded Data Fabric, then you will not be able to connect tenant storage to HPE Ezmeral Data Fabric on Kubernetes.

A deployment of HPE Ezmeral Container Platform can include HPE Ezmeral Data Fabric on Kubernetes, or an embedded Data Fabric, but not both.

**Note:** HPE highly recommends configuring your deployment to use HPE Ezmeral Data Fabric on Kubernetes instead of an embedded HPE Ezmeral Data Fabric.



### Compute and Storage Separation

Getting the maximum flexibility from a container-based solution requires being able to independently scale compute and storage resources. It is also essential to be able to support the persistence of Big Data datasets beyond the lifespan of a Big Data compute cluster. The DataTap and IOBoost technologies allow virtual clusters to access remote data regardless of location or format.

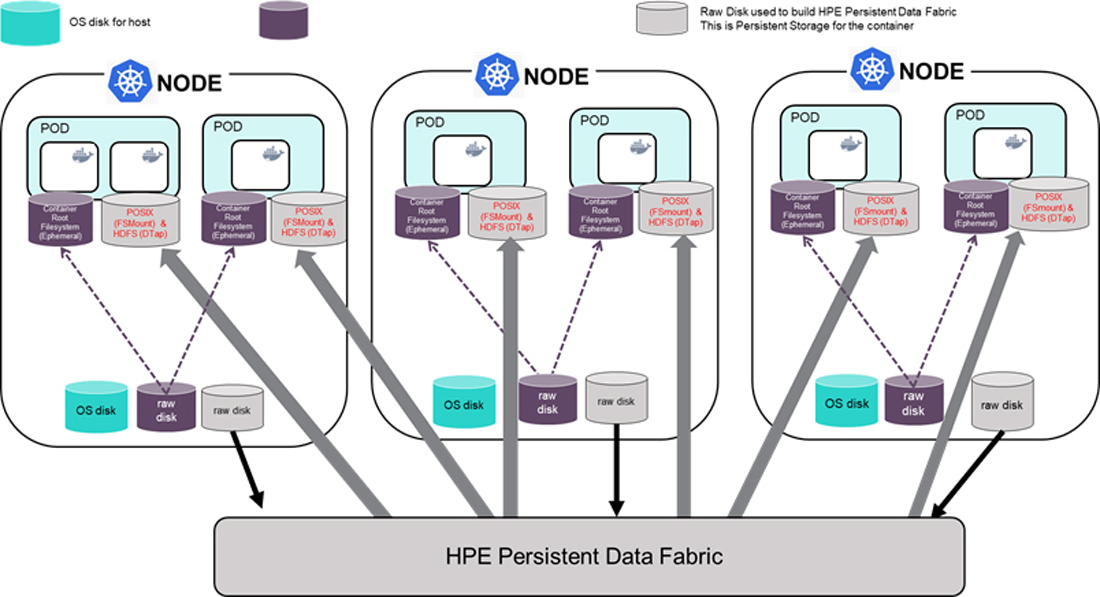
A DataTap creates a logical data lake overlay that allows access to shared data in the enterprise storage devices. This allows users to run Big Data and ML/DL jobs using the existing enterprise storage without needing to make time-consuming copies or transfers of data to local disks. IOBoost augments DataTap's flexibility by adding an application-aware data caching and tiering server to ensure high-speed remote data delivery.

This persistent storage can also serve as filesystem mount storage (FS mounts). The filesystem mount feature allows automatically adding mounts to virtual nodes/containers, thereby allowing virtual nodes/containers to directly access POSIX data as if they were local directories. You can use this feature to provide common files across all of the virtual nodes/containers in a given tenant, such as a common configuration file that will be used by all of the virtual nodes/containers in the Marketing tenant. This eliminates the need to manually copy common files to individual virtual nodes/containers.

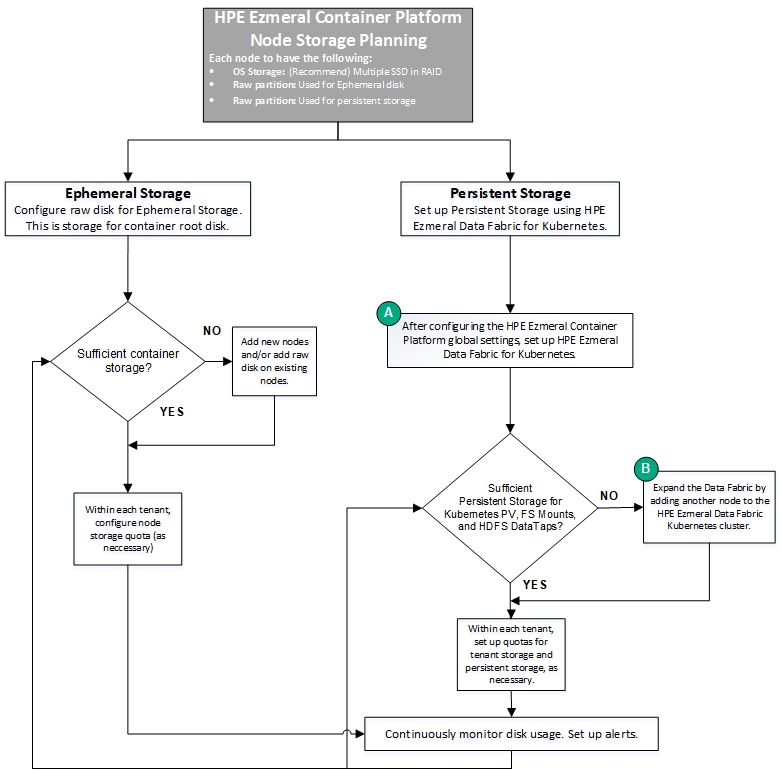
All applications running in containers can natively access data across the HPE Persistent Storage fabric via both DataTaps and FS mounts. Persistent volumes are seamlessly available across clusters from this persistent data fabric.

### Operating System Storage

For all host types, the recommended storage for the operating system is two 960 GB SSD's in a RAID 1 configuration. See [Host Requirements](https://docs.containerplatform.hpe.com/53/reference/system-requirements/general/Host_Requirements.html) for detailed storage requirements and recommendations.



### Storage Planning Flowchart



### About HPE Ezmeral Data Fabric on Kubernetes

A typical Kubernetes environment may have pods frequently coming and going. Large Kubernetes environments, such as in a public cloud, may handle pools of systems where new hosts are added to support pod and cluster placement. In HPE Ezmeral Container Platform, a Data Fabric cluster is a Kubernetes Custom Resource that functions as a storage cluster that provides access to PVCs, tenant storage, shares, and other storage needs. In a Data Fabric cluster:

The hosts (called nodes) commit considerable disk resources that may include NVMe and enterprise-class SSDs.

* The Data Fabric cluster may only need to come up on a few nodes.
* Pods are unlikely to be deleted frequently;
* The Data Fabric CR must account for host resource profiles to guarantee core pod availability.

HPE Ezmeral Container Platform includes native support for HPE Ezmeral Data Fabric. This avoids many manual steps and allows you to create Data Fabric clusters in a manner similar to that used for creating Compute Kubernetes clusters (see [Creating a New Data Fabric Cluster](https://docs.containerplatform.hpe.com/52/reference/hpe-ezmeral-data-fabric-admini/Creating_a_New_Data_Fabric_Cluster.html) and [Creating a New Kubernetes Cluster](https://docs.containerplatform.hpe.com/52/reference/kubernetes/kubernetes-administrator/clusters/Creating_a_New_Kubernetes_Cluster.html)). Each Data Fabric cluster resides on nodes. See [Kubernetes Worker Installation Overview](https://docs.containerplatform.hpe.com/52/reference/kubernetes/kubernetes-administrator/kubernetes-hosts/installing-kubernetes-hosts/Kubernetes_Worker_Installation_Overview.html) and [Kubernetes Data Fabric Node Installation Overview](https://docs.containerplatform.hpe.com/52/reference/kubernetes/kubernetes-administrator/kubernetes-hosts/installing-kubernetes-hosts/Kubernetes_Data_Fabric_Node_Installation_Overview.html).

#### Features

HPE Ezmeral Container Platform automates the following functionality for a Data Fabric backed by a Data Fabric cluster:

* Pre-checking nodes before tagging them for use with Data Fabric clusters.
* Checking for sufficient resources to being up core and service pods when creating a Data Fabric cluster.
* Bootstrapping software installation, namespace creation, and other functions.
* Automatic Data Fabric CR creation based on scanning node system information and resource profiles. This CR helps determine how many CLDB, ZK, and MFS pods can be created and ensure proportional resource requests relative to node resources or grouped disk profiles. HPE Ezmeral Container Platform updates the standard “template' Data Fabric CR at cluster creation time. Users may view/download the Data Fabric CR after cluster creation.
* Auto-registration of Tenant Storage/PVCs, along with clean-up functionality to allow deregistration if needed for another Data Fabric cluster.
* Data Fabric clusters automatically become the default storageclass for Compute Kubernetes clusters.
* Gateway hosts expose HPE Ezmeral Data Fabric services such as MCS, Kibana, and Grafana via clickable links in the web interface.
* User-settable configuration parameters allow fine-tuning a cluster to suit specific needs. See [User-Configurable Data Fabric Cluster Parameters](https://docs.containerplatform.hpe.com/52/reference/hpe-ezmeral-data-fabric-admini/UserConfigurable_Data_Fabric_Cluster_Parameters.html).
* Data Fabric clusters can be expanded by adding additional nodes, as described in [expanding a Data Fabric Cluster](https://docs.containerplatform.hpe.com/52/reference/hpe-ezmeral-data-fabric-admini/Expanding_a_Data_Fabric_Cluster.html). The original cluster size and the number and composition of new node determine whether CLDB, ZK, and/or MFS pods will be added. Once expanded, a Data Fabric cluster cannot be shrunk.
* HPE Ezmeral Data Fabric packages can be started automatically when creating a Kubernetes cluster in HPE Ezmeral Container Platform. The user can also select Compute packages to install by clicking the available option(s) during cluster creation.
* The POSIX client type (“Basic” or “Extended”) can be specified on a per-node basis.

#### Limitations

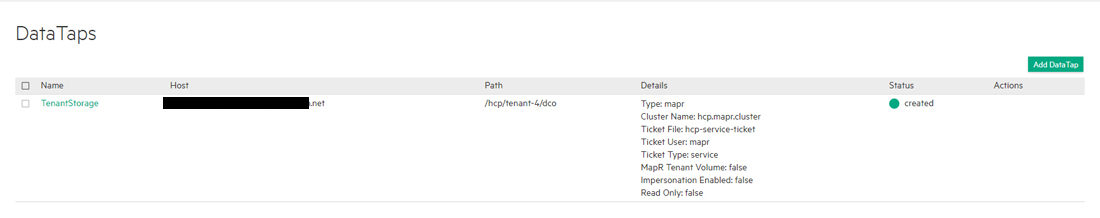
The following limitations apply to Data Fabric clusters:

* Only one Data Fabric cluster can be created. This one Data Fabric cluster therefore registers the Tenant Storage and Share for all Kubernetes tenants.
* Migrating from an integrated/embedded form of HPE Ezmeral Data Fabric (versions 5.1.1. and below) to a Data Fabric cluster (versions 5.2 and above) requires manual steps. See [Manually Creating/Editing a Data Fabric Cluster](https://docs.containerplatform.hpe.com/52/reference/hpe-ezmeral-data-fabric-admini/manual---advanced-functionalit/Manually_CreatingEditing_a_Data_Fabric_Cluster.html).

## About DataTaps

DataTaps expand access to shared data by specifying a named path to a specified storage resource. Applications running within virtual clusters that can use the HDFS filesystem protocols can then access paths within that resource using that name. This allows you to run jobs using your existing data systems without the need to make time-consuming copies or transfers of your data. Tenant/Project Administrator users can quickly and easily build, edit, and remove DataTaps using the **DataTaps** screen, as described below. Tenant Member users can access DataTaps by name.

Selecting **DataTaps** in the main menu opens the **DataTaps** screen. The information and functions on this screen will vary depending on your role. For Tenant Administrators, the **DataTaps** screen for Tenant Administrators appears as shown in the following image.



Each DataTap requires the following properties to be configured, depending on the type of storage being connected to (MapR, HFDS, HDFS with Kerberos, or NFS):

**Name:** A unique name for each DataTap. This name may contain letters (A-Z or a-z), digits (0-9), and hyphens (-), but may not contain spaces. You can use the name of a valid DataTap to compose DataTap URIs that you pass to applications as arguments. Each such URI maps to some path on the storage system that the DataTap points to. The path indicated by a URI might or might not exist at the time you start a job, depending on what the application wants to do with that path. Sometimes the path must indicate a directory or file that already exists, because the application intends to use it as input. Sometimes, the path must not currently exist, because the application expects to create it. The semantics of these paths are entirely application- dependent, and are identical to their behavior when running the application on a physical Hadoop or Spark platform.

**Description:** Brief description of the DataTap, such as the type of data or the purpose of the DataTap.

**Type:** Type of file system used by the shared storage resource associated with the DataTap (**MAPR**, **HDFS**, or **NFS**). This is completely transparent to the end job or other process using the DataTap.

The following fields depend on the DataTap type:

* MapR
* HDFS or NFS

## Using a DataTap

The storage pointed to by a DataTap can be accessed via a URI that includes the name of the DataTap.

A DataTap points to the top of the “path” configured for the given DataTap. The URI has the following form:

***dtap://datatap\_name/***

In this example, datatap\_name is the name of the DataTap that you wish to use. You can access files and directories further in the hierarchy by appending path components to the URI:

***dtap://datatap\_name/some\_subdirectory/another\_subdirectory/some\_file***

For example, the URI **dtap://mydatatapr/home/mydirectory** means that the data is located within the **/home/mydirectory** directory in the storage that the DataTap named **mydatatap** points to.

DataTaps exist on a per-tenant basis. This means that a DataTap created for Tenant A cannot be used by Tenant B. You may, however, create a DataTap for Tenant B with the exact same properties as its counterpart for Tenant A, thus allowing both tenants to access the same storage resource. Further, multiple jobs within a tenant may use a given DataTap simultaneously. While such sharing can be useful, be aware that the same cautions and restrictions apply to these use cases as for other types of shared storage: multiple jobs modifying files at the same location may lead to file access errors and/or unexpected job results.

Users who have a Tenant Administrator role may view and modify detailed DataTap information. Members may only view general DataTap information and are unable to create, edit, or remove a DataTap.

**CAUTION:** DATA CONFLICTS MAY OCCUR IF MORE THAN ONE DATATAP POINTS TO A LOCATION BEING USED BY MULTIPLE JOBS AT ONCE.

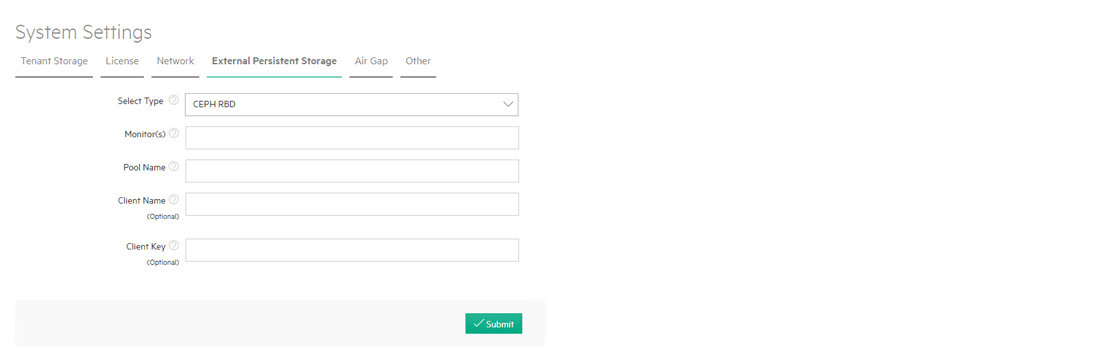
**CAUTION:** EDITING OR DELETING A DATATAP WHILE IT IS BEING USED BY ONE OR MORE RUNNING JOB(S) MAY CAUSE ERRORS IN THE AFFECTED JOB(S).

## Stateful Container Migration

You may specify an external persistent storage pool for tenants and/or AI/ML projects using the **Persistent Storage** tab of the **System Settings** screen in Application Persistent Storage Tab.

The Application Persistent Storage tab enables the Platform Administrator to manage the external persistent storage pool used when migrating containers between hosts in HPE Ezmeral Container Platform.

The **Application Persistent Storage** tab (formerly the **External Persistent Storage** tab) of the **Systems Settings** screen enables Platform Administrators to connect to an external storage pool that will be used to store crucial container folders and enable migrating containers between hosts. You may create, expand, and shrink this external storage resource just like any other storage resource.



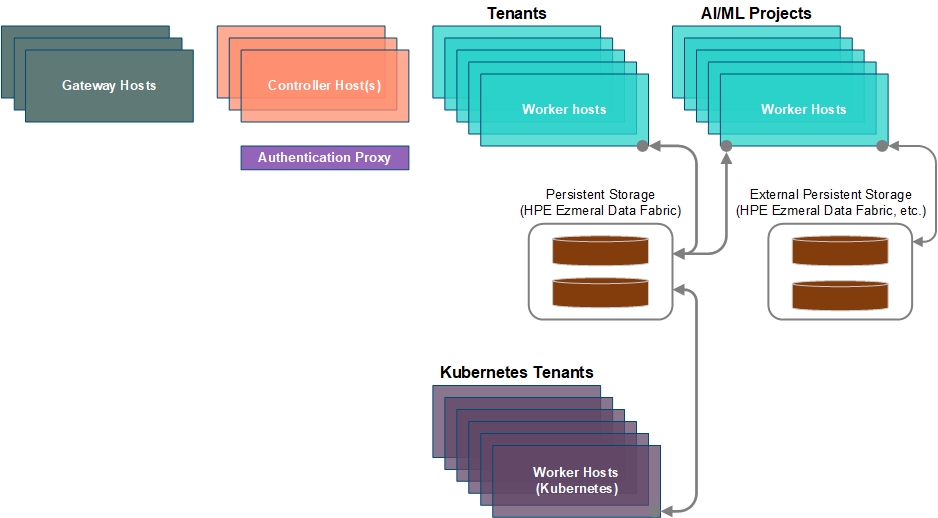
To map an external storage pool for use as persistent storage:

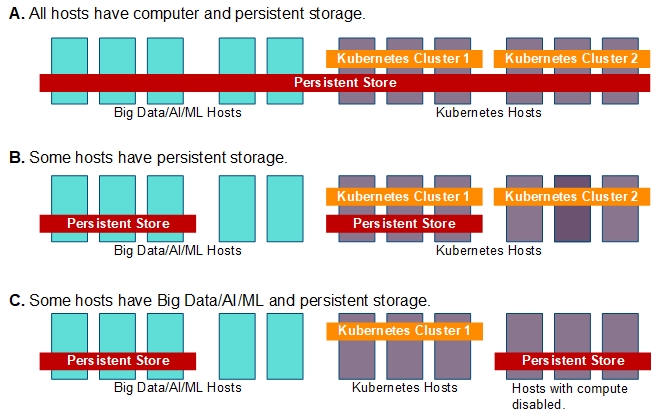
Use the **Select Type** pull-down menu to select the filesystem used by the external resource, and then enter the appropriate parameters based on your selection. The available options are:

* **None:**  Select this option to disable persistent storage.
* **Local MapR:**  This option is only available for on-premises deployments.
* **CEPH RBD:**  This option is only available for on-premises deployments.
* **NFS:**  This option is only available for on-premises deployments.
* **ScaleIO:**  This option is only available for on-premises deployments.

**CAUTION:** Changing the persistent storage pool settings will cause containers that are currently using persistent storage to become ineligible for migration.

Persistent storage exists either on hyper-converged resources or on a remote storage resource that is pointed to but not managed by HPE Ezmeral Container Platform.



Persistent storage can be implemented across some or all hosts, as shown here:  


You can create, expand, and shrink storage capacity just as you would any other resource. This feature allows you to migrate containers between hosts by (default) preserving the following critical container folders for ongoing use:

* /usr
* /opt
* /var
* /etc
* /home

The contents of other folders can be preserved during container migration by specifying their names in the metadata JSON file when creating a new application image generated using the App Workbench.

### Use Cases

Big Data applications such as Hadoop and Spark offer robust high availability capabilities; however, some enterprise customers have operational requirements that require the ability to move virtual nodes/containers from one host to another. These operational requirements include:

* A host crashing and the containers that were running on that host must be redeployed on other working hosts with minimal downtime and no additional configuration required.
* One or more host(s) needs to be replaced for maintenance and/or as part of a server refresh cycle. In this scenario, the containers running on those hosts must be seamlessly moved to other hosts that are not being replaced, with minimal downtime to the applications running in the containers.
* Resolving a condition where there is an inability to meet the SLA for an application running in a containerized cluster (e.g. Spark or Hadoop) due to poor performance (CPU, network, or storage). This is a resource contention (bottleneck) condition that requires a re-balancing of virtual nodes/containers onto hosts with more available resources.

### Enabling Container Migration

Once you have enabled persistent storage, the next step is to create one or more flavor(s) that include at least 20GB of persistent storage.

A virtual node flavor is the number of vCPU cores, RAM capacity, root disk storage capacity, persistent storage capacity (if any), and number of GPUs (if any) required by a virtual node/container. For example, if the flavor **small** specifies a single vCPU core, 3GB of RAM, 30GB disk, and two GPUs, then every virtual node created with the **small** flavor will be allocated those resources, subject to the rules described as follows:-

#### Virtual Cores, RAM, Storage, and GPU Devices

**Note:** This article uses the term "tenant" to refer to both tenants and projects.

**Virtual CPU (vCPU) cores are modeled as follows:**

* The license specifies the maximum number of CPU cores that can be surfaced by the set of on-premises and/or public cloud hosts in a given HPE Ezmeral Container Platform deployment. Starting with Container Platform version 3.8, the use or non-use of CPU hyperthreads does not impact the license and vCPU count.
* The number of available vCPU cores is the number of physical CPU cores multiplied by the CPU allocation ratio specified by the Platform Administrator. For example, if the hosts have 40 physical CPU cores and the Platform Administrator specifies a CPU allocation ratio of 3, then a total of 120 vCPU cores will be displayed as available. You can allocate an unlimited number of vCPU cores to each tenant or project. The collective core usage for all nodes (containers) within a tenant/project will be constrained by either the tenant's assigned quota or the available cores in the system, whichever limit is reached first. The tenant quotas and the CPU allocation ratio act together to prevent tenant members from overloading the system's CPU resources.
* When two or more nodes are assigned to the same host, they contend for the same physical CPU resources of that host. CPU resources are allocated to such nodes in a ratio determined by their vCPU core count. For example, a node with 8 cores will receive twice as much CPU time as a node with 4 cores.
* The Platform Administrator can also specify a Quality of Service (QOS) multiplier for each tenant or project. In the case of CPU resource contention, the node vCPU count is multiplied by the tenant/project QOS multiplier when determining the physical CPU time that will be allotted to each container running within a given tenant or project. For example, a node with 8 vCPU cores in a tenant or project with a QOS multiplier of 1 will receive the same physical CPU time as a node with 4 vCPU cores in a tenant or project with a QOS multiplier of 2. The QOS multiplier is used to describe relative tenant/project priorities when CPU resource contention occurs; it does not affect the overall cap on CPU load established by the CPU allocation ratio and tenant/project quotas.

**RAM is modeled as follows:**

* The total amount of available RAM is equal to the amount of unreserved RAM. Unreserved RAM is the amount of RAM remaining after reserving some memory in each host for platform services. For example, if your deployment consists of four hosts that each have 128GB of physical RAM with 110GB of unreserved RAM, the total amount of RAM available to share among tenants or projects will be 440GB.
* You may allocate an unlimited amount of RAM to each tenant/project. The collective RAM usage for all nodes within a tenant or project will be constrained by either the tenant's or project's assigned quota or the available RAM in the system, whichever limit is reached first.

**Storage is modeled as follows:**

* Root disk storage space is allocated from the disk(s) on each Worker host that are assigned as Node Storage disks when adding the Worker to the platform. Each node consumes Node Storage space equivalent to its root disk size on the Worker host where that node is placed.
* If desired, you may specify 20GB or more of persistent storage space to facilitate container migration. When this option is selected:
  + The persistent storage resource is configured using the web interface. See [Application Persistent Storage Tab](https://docs.containerplatform.hpe.com/52/reference/global-settings/settings/Application_Persistent_Storage_Tab.html).
  + When creating containers using a flavor that includes a persistent storage amount, the persistent storage resource must have an amount of free space equal to the **Persistent Storage Size (GB)** value times the number of container(s) that are being created using that flavor. See [Virtual Node Flavors](https://docs.containerplatform.hpe.com/52/reference/epic/key-epic-concepts/Virtual_Node_Flavors.html) and [Creating a New Flavor](https://docs.containerplatform.hpe.com/52/reference/epic/administering-a-tenant-or-proj/managing-virtual-node-flavors/Creating_a_New_Flavor.html).

If compatible GPU devices are present, then they are modeled as follows:

* HPE Ezmeral Container Platform must be running either:
  + Version 7.x of the RHEL/CentOS operating system
  + Version 15 of the SUSE operating system
* You must install the NVIDIA drivers on the hosts before deploying HPE Ezmeral Container Platform, as described in [GPU Driver Installation](https://docs.containerplatform.hpe.com/52/reference/deploying-the-platform/GPU_Driver_Installation.html).
* The total number of available GPU resources is equal to the number of physical GPU devices. For example, if your deployment consists of four hosts that each have 8 physical GPU devices, then there will be a total of 32 GPU devices available to share among tenants and/or projects.
* You may allocate an unlimited number of GPU resources to each tenant or project. The collective GPU resource usage for all nodes within a tenant or project will be constrained by either the tenant's or project's assigned quota or the available GPU devices in the system, whichever limit is reached first.
* GPU devices are expensive resources, and their usage is maximized as follows:
  + If a virtual node does not require GPU devices, then HPE Ezmeral Container Platform attempts to place that node on a host that does not have any GPU devices installed.
  + If a virtual node/container does require GPU resources, then HPE Ezmeral Container Platform attempts to place that container in such a way as to maximize GPU resource utilization on a given host and to reduce/eliminate wasted resources.
  + HPE Ezmeral Container Platform does not have the concept of a virtual GPU. This means that a container deployed on one host cannot access the GPU resources of another host. Containers are limited to accessing GPUs only on the host where they are deployed.
  + HPE Ezmeral Container Platform does not allow sharing the same GPU device between multiple containers simultaneously. Once a GPU device is allocated to a given container, that container has exclusive access to that GPU.

Default values will appear in the various quota fields when you are creating a tenant/project. These default values will be 25% of the total system resources for most fields. The exception to this rule is the quota for GPU devices where the default value is 0. When configuring each resource quota, the web interface displays the total available amount of that resource for comparison. You may edit these quota values or delete a value and leave the field blank to indicate that the tenant does not have a quota defined for that resource.

Assigning a quota of resources to a tenant does not reserve those resources for that tenant when that tenant is idle (not running one or more clusters). This means that a tenant may not actually be able to acquire system resources up to the limit of its configured quota.

* You may assign a quota for any amount of resources to any tenant(s) regardless of the actual number of available system resources. A deployment where the total amount of configured tenant resource quotas exceeds the current amount of system resources is called over-provisioning. Over-provisioning occurs when one or more of the following conditions are met:
  + You have a tenant which has resource quotas that either exceed the system resources or are undefined. This tenant will only be able to obtain the amount of resources that are actually available. This arrangement is typically a convenience to make sure that the tenant is always able to fully utilize the platform, even if you add more hosts in the future.
  + You have multiple tenants where none have overly large or undefined quotas, but where the sum of their quotas exceeds the resources currently available. In this case, you are not expecting all tenants to attempt to use all their resource quotas simultaneously. Still, you have given each tenant the ability to claim more than its “fair share” of resources when these extra resources are available. In this case, you must balance the need for occasional bursts of usage that may exceed quota resources against the need to restrict how much a “greedy” tenant can consume. A larger quota gives more freedom for burst consumption of unused resources while also expanding the potential for one tenant to prevent other tenants from fully utilizing their quotas.

**Note:** Over-provisioning is useful in certain situations; however, avoiding over-provisioning prevents potential resource conflicts by ensuring that all tenants are guaranteed to be able to obtain their configured quota of virtual CPU cores, RAM, and GPU devices.

A default set of flavors (such as **Small**, **Medium**, and **Large**) is created during installation.

**Note:** Flavors are specific to Big Data tenants and AI/ML projects. They do not apply to Kubernetes). A flavor that is created or edited in one tenant will not affect a flavor with the same name in another tenant.

The Tenant Administrator should create flavors with virtual hardware specifications appropriate to the clusters that tenant members will create. Application characteristics will guide these choices, particularly the minimum virtual hardware requirements per node. Using nodes with excessively large specifications will waste resources (and count toward a tenant's quota). It is therefore important to define a range of flavor choices that closely match user requirements.

The Tenant Administrator may freely edit or delete these flavors. When editing or deleting a flavor:

* If you edit or delete an existing flavor, then all nodes using that flavor will continue using the flavor as specified before the change or deletion.
* You may delete all of the defined flavors defined; however, you will be unable to create any new clusters until you create at least one flavor.
* You may specify a root disk size when creating or editing a flavor. This size overrides the default root disk size specified by the **App Store** image. Specifying a root disk size that is smaller than the minimum size indicated by a given image will prevent you from being able to instantiate that image on a cluster that uses that flavor. Creating a larger root disk size may slow down cluster creation.

**Note:** Consider using DataTaps where possible for optimal performance.

**Note:** Containers created using a flavor that does not have persistent storage enabled will not benefit from this feature, even if you later edit the flavor to enable persistent storage.

**Note:** You may create a flavor that specifies persistent storage even if no persistent storage has been defined in the **Persistent Storage** tab; however, an error will be returned if you attempt to use this flavor before enabling persistent storage.

**Note:** The persistent storage resource must have enough free capacity to accommodate the sum of all tenant persistent storage quotas or to accommodate the amount of persistent storage specified in all applicable flavor(s) times the number of containers that use the flavor(s), whichever is greater. Further, if you specify a per-tenant persistent storage quota, then that quota must be large enough to accommodate the flavor-defined persistent storage times the number of containers using the applicable flavor(s).

There are two ways to use persistent storage to migrate a container:

* **Worker Vacate (EPIC hosts only):** If an EPIC Worker host goes down and some or all of the containers on that host use persistent storage, then you can click the **Worker Vacate** button (moving dolly) for the desired host in the **EPIC Hosts Installation**screen (see [The EPIC Hosts Installation Screen](https://docs.containerplatform.hpe.com/52/reference/epic/epic-administration/hosts/The_EPIC_Hosts_Installation_Screen.html)) to perform a Worker vacate function. All jobs running on the affected container(s) will end, but the containers themselves will be recovered as follows:
  + No new containers will be placed on the affected host.
  + The protected containers are removed from the affected host.
  + Containers automatically migrate to one or more new host(s), provided that there are sufficient available resources, including any applicable placement constraints, as described in [About Tags](https://docs.containerplatform.hpe.com/52/reference/kubernetes/kubernetes-administrator/kubernetes-hosts/tags/About_Tags.html) and [Tenant/Project Tags and Constraints](https://docs.containerplatform.hpe.com/52/reference/epic/epic-administration/managing-tenants-and-projects/TenantProject_Tags_and_Constraints.html).
* **Node Migration:** This use case applies to a scenario where the hosts are functioning properly but are overburdened. In this case, the Tenant Administrator or Platform Administrator can add new hosts as described in [EPIC Worker Installation Overview](https://docs.containerplatform.hpe.com/52/reference/epic/epic-administration/hosts/installing-worker-hosts/EPIC_Worker_Installation_Overview.html). and [Kubernetes Worker Installation Overview](https://docs.containerplatform.hpe.com/52/reference/kubernetes/kubernetes-administrator/kubernetes-hosts/installing-kubernetes-hosts/Kubernetes_Worker_Installation_Overview.html). Containers can then be migrated to the new host(s) on a container-by-container basis, as described in [Viewing and Migrating Virtual Nodes](https://docs.containerplatform.hpe.com/52/reference/epic/epic-administration/Viewing_and_Migrating_Virtual_Nodes.html). Placement constraints apply to this type of container migration as well.

The following storage systems are supported for persistent storage:

* CEPH RBD
* NFS
* ScaleIO
* Local MapR

Migrating a container/virtual node has the following effects:

* The cluster to which the container belongs will be impacted because the container will not be executing jobs during the migration process.
* Any jobs or ActionScripts running on a cluster with one or more migrating container(s) will be lost and must be run again after completing the migration.
* Any data residing in non-persistent storage directories of a container being migrated will be lost.
* Any external host(s) that have access to HPE Ezmeral Container Platform will be unable to access the affected container(s) until the migration process completes.
* Migrated containers maintain their configuration and IP addresses.

## Using the Pre-Check Script

The pre-check script performs a series of checks on the Controller host to determine whether it is ready to accept the installation. To use the script:

* Download the hpe-cp-prechecks-<version>.bin script, where <version> is the version number, such as 5.2.
  + **For RHEL/CentOS:** [RHEL/CentOS Pre-check script](https://bdk8s.s3.us-east-2.amazonaws.com/5.3/3031/hpe-cp-rhel-prechecks-5.3.bin)
  + **For SLES:** [SLES Pre-check script](https://bdk8s.s3.us-east-2.amazonaws.com/5.3/3031/hpe-cp-sles-prechecks-5.3.bin)
* If needed, copy the .bin file to a directory on the machine that will become the Controller host.
* Make the .bin file executable by executing the command chmod a+x hpe-cp-prechecks-<version>.bin.
* Run the executable binary using the format hpe-cp-prechecks-<version>.bin <options>, where <options> denotes the option(s) and/or parameter(s) you need to pass to the script.
* After running the script, see [Sample Pre-Check Output](https://docs.containerplatform.hpe.com/53/reference/deploying-the-platform/phase-2/Sample_PreCheck_Output.html) for a complete example of a successful pre-check as well as links to common errors and how to resolve them. In addition to the displayed output, the pre-check script generates several files that are described in [Pre-Check Generated Files](https://docs.containerplatform.hpe.com/53/reference/deploying-the-platform/phase-2/PreCheck_Generated_Files.html). If needed, remediate any issues and then re-run the pre-check script until all tests pass or until you have accounted for any warnings.

**CAUTION: DO NOT USE THE CONFIG FILE TO INSTALL THE CONTROLLER IF THE PRE-CHECK OUTPUT LISTS ONE OR MORE ERROR(S).**

Once you are satisfied with the results of the pre-check, you may proceed to install the Controller host using either of the following methods:

* Configuration file generated by the pre-check script, as described in Using the Pre-Check Config File. This file will be named /tmp/bd\_prechecks.conf.
* Manually, as described in [Standard Installation](https://docs.containerplatform.hpe.com/53/reference/deploying-the-platform/phase-3/step-1--cli/Standard_Installation.html).

**Note**: HPE recommends using the config file generated by the pre-check script when installing HPE Ezmeral Container Platform on the Controller host as this method will most likely be faster and easier than manual installation.

#### Examples

This section presents some examples of using the pre-check script.

* **Root/Agent:** This example pre-checks HPE Ezmeral Container Platform as the root user and includes the Worker agent because password-less SSH is not available in the environment.

***# root@localhost> /root/hpe-cp-prechecks-5.1.bin --worker-agent-install***

* **Root/non-Agent:** This example pre-checks HPE Ezmeral Container Platform as the root user. The environment does allow password-less SSH, and thus the --worker-agent-install option is *not* used.

***# root@localhost> /root/hpe-cp-prechecks-5.1.bin***

* **Root/Agent/SSL:** This example pre-checks HPE Ezmeral Container Platform as the root user and includes the Worker agent because password-less SSH is not available in the environment. It also provides SSL information to enable secure (https://) access to the Container Platform interface.

***# root@localhost> /root/hpe-cp-prechecks-5.1.bin --worker-agent-install --ssl-cert /root/bdhost.cert --ssl-priv-key /root/bdhost.pem***

* **Root/Non-Agent/SSL:** This example pre-checks HPE Ezmeral Container Platform as the root user. The environment does allow password-less SSH, and thus the --worker-agent-install option is *not* used. This example also provides SSL information to enable secure (https://) access to the Container Platform interface.

***# root@localhost> /home/epic/hpe-cp-prechecks-5.1.bin --ssl-cert /root/bdhost.cert --ssl-priv-key /root/bdhost.pem***

* **Non-root/Agent:** This example pre-checks HPE Ezmeral Container Platform as a non-root user and includes the Worker agent because password-less SSH is not available in the environment.

***# epic@localhost> /home/epic/hpe-cp-prechecks-5.1.bin --worker-agent-install***

* **Non-root/non-Agent:** This example pre-checks HPE Ezmeral Container Platform as a non-root user. The environment does allow password-less SSH, and thus the --worker-agent-install option is *not* used

***# epic@localhost> /home/epic/hpe-cp-prechecks-5.1.bin***

* **Non-root/Agent/SSL:** This example pre-checks HPE Ezmeral Container Platform as a non-root user and includes the Worker agent because password-less SSH is not available in the environment. It also provides SSL information to enable secure (https://) access to the web interface.

***# epic@localhost> /home/epic/hpe-cp-prechecks-5.1.bin --worker-agent-install --ssl-cert /home/epic/bdhost.cert --ssl-priv-key /home/epic/bdhost.pem***

* **Non-root/non-Agent/SSL:** This example pre-checks HPE Ezmeral Container Platform as a non-root user. The environment does allow password-less SSH, and thus the --worker-agent-install option is *not* used. This example also provides SSL information to enable secure (https://) access to the web interface.

***# epic@localhost> /home/epic/hpe-cp-prechecks-5.1.bin --ssl-cert /home/epic/bdhost.cert --ssl-priv-key /home/epic/bdhost.pem***

**Note**: Be sure to run both the precheck script and installer as the user who will be installing HPE Ezmeral Container Platform.

## Using the Pre-Check Config File

Using the config file generated by the pre-check script to install HPE Ezmeral Container Platform on the Controller host can be a useful option. This option bypasses the pre-check script or overrides pre-check values provided that all of the following conditions are met:

* You are aware of and have accounted for any warnings contained in the pre-check output.
* The pre-check output did not contain any errors.
* Nothing has changed about the Controller host, user, network, infrastructure, operating system, or configuration since the pre-check was successfully run.

**CAUTION: DO NOT USE THE CONFIG FILE TO INSTALL THE CONTROLLER IF THE PRE-CHECK OUTPUT LISTS ONE OR MORE ERROR(S).**

**Note**: Be sure to rerun the pre-check script with the appropriate option(s) if anything has changed since the last time it was successfully run.

To use the config file for the installation:

1. Log into the host that you will be using as the Controller host using either the root account or password or your assigned username and password.
2. If needed, copy the binary (.bin) to the host that you will use as the Controller host.
3. Make the .bin file executable by executing the command chmod a+x <hpe\_ezmeral>.bin

Where:

* <hpe\_ezmeral> is the full name of the .bin file.

1. Run the executable binary from the Linux console as the assigned user by typing ./<hpe\_ezmeral>.bin --prechecks-config-file <path> <additional\_options>, where:

* <hpe\_ezmeral> is the full name of the .bin file.
* <path> is the complete path to the config file generated by the pre-check script.
* <additional\_options> are any additional options you need to add, such as --default-password. See [Standard Installation](https://docs.containerplatform.hpe.com/53/reference/deploying-the-platform/phase-3/step-1--cli/Standard_Installation.html) for a list of options.

1. The installer checks the integrity of the bundle and then extracts the bundle contents.
2. The End User License Agreement (EULA) appears. Read through the EULA, pressing [SPACE] to page through the content. Once you have viewed the entire EULA, press [y] to accept it and continue the installation.
3. HPE Ezmeral Container Platform installs on the Controller host. A series of messages appear during the installation. The following message appears once the installation is complete:

Successfully installed HPE CP.

Please visit https://10.50.1.1 to configure the server.

[root@hostname-1 ~] .

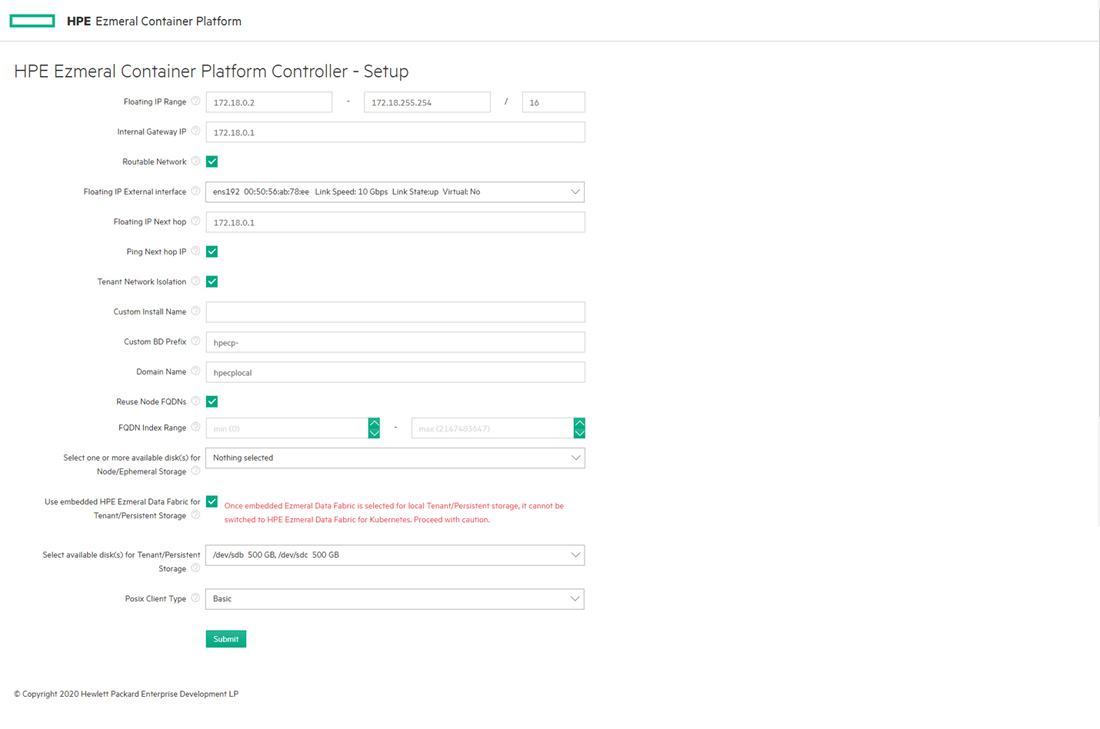
This concludes the first phase of the installation. Note the URL provided, as you will use this to continue configuration. Please proceed to Platform Controller Setup to continue the installation using the web interface.

## Platform Controller Setup

The next step of the installation process uses a Web browser to access the web interface. To do this:

1. Open a Web browser and navigate to the URL provided at the end of the command line installation process.

The HPE Ezmeral Container Platform Controller - Setup screen appears.



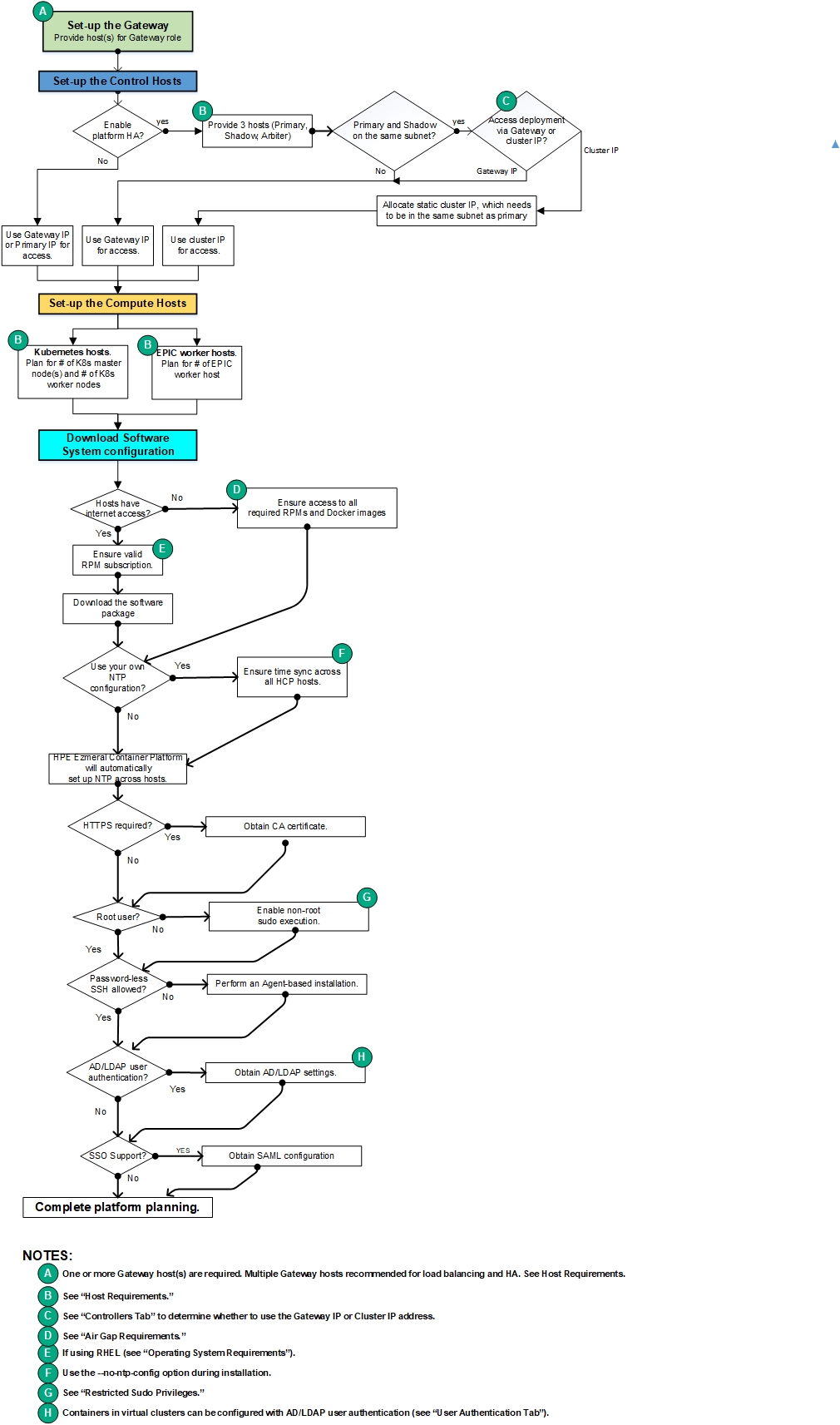
Use the **Select one or more available disk(s) for Node/Ephemeral Storage** list to select one or more disk(s) to use for ephemeral/node storage. In Kubernetes, this storage does not persist after the pod cease to exist. In virtual clusters, this storage does not persist after the virtual cluster is deleted but will remain when the Docker container is stopped.

Press either [CONTROL] (Windows/Linux) or [COMMAND] (MacOS) while clicking to select multiple disks. If you make a mistake, then either [CONTROL]-click or [COMMAND]-click the selected disk(s) that you want to remove.

See Storage. If the installer has detected one or more raw disks, then you can choose how to configure tenant/persistent storage:

* HPE recommends using HPE Ezmeral Data Fabric on Kubernetes for persistent storage. If you are planning to use HPE Ezmeral Data Fabric on Kubernetes, then do not specify any disks for tenant/persistent storage at this time, and **DO NOT SELECT** the **Use embedded HPE Ezmeral Data Fabric for Tenant/Persistent Storage** check box. HPE Ezmeral Container Platform does not support the use of both Embedded Data Fabric and HPE Ezmeral Data Fabric on Kubernetes in the same deployment.
* If you choose to use Embedded Data Fabric for tenant/persistent storage (not recommended), you can select the **Use embedded HPE Ezmeral Data Fabric for Tenant/Persistent Storage** check box, and then use the **Select one or more disk(s) for Tenant/Persistent Storage** pull-down menu to select disks. Press either [CONTROL] (Windows/Linux) or [COMMAND] (MacOS) while clicking to select multiple disks. If you make a mistake, then either [CONTROL]-click or [COMMAND]-click the selected disk(s) that you want to remove. After you select a disk for use by the Embedded Data Fabric and deploy the platform, you cannot later switch that disk for use by HPE Ezmeral Data Fabric on Kubernetes.

## Platform Resource Planning



## High Availability

Three levels of High Availability protection are available:

* **Platform High Availability**. This level of protection only applies to HPE Ezmeral Container Platform services.
* **Virtual Cluster High Availability**. This level of protection applies to all clusters.
* **Gateway Host High Availability**. This level of protection only applies to the Gateway host(s).

### Platform High Availability

The control plane supports platform-level High Availability that protects against the failure of the Controller host. Note that each Kubernetes cluster deployed by the control plane can be deployed as a highly available (HA) cluster by choosing multiple hosts as masters. (In this scenario, "multiple masters" means at least three hosts; because of quorum requirements, two master hosts are not sufficient.) The scope of this section is to focus on the HA of the control plane (the Controllers), which must include at least three hosts that conform to the requirements listed in Host Requirements in order to support this feature.

**Note:** HPE recommends enabling platform High Availability before adding a large number of Kubernetes or EPIC hosts. See Controllers Tab.

**Note:** You must either enable platform High Availability protection before creating any virtual clusters or delete all existing virtual clusters before enabling High Availability protection.

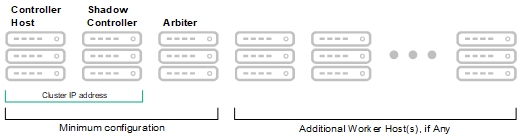
Platform-level High Availability requires designating two Worker hosts as the Shadow Controller and Arbiter, respectively. If the Controller host fails, failover to the Shadow Controller host occurs within approximately two or three minutes, and a warning appears at the top of the web interface screens. You might need to log back in if this occurs.

Each host has its own IP address. If the Controller host fails, attempting to access the Shadow Controller host using the same IP address will fail. Similarly, accessing the Shadow Controller host using that host's IP address will fail once the Controller host recovers. To avoid this problem, you must either:

Specify a cluster IP address that is bonded to the node acting as the Controller host, and then log into the web interface using that cluster IP address. You will automatically connect to the Controller host (under normal circumstances) or to the Shadow Controller host with a warning message (if the Controller host has failed and triggered the High Availability protection). In this case, the Primary Controller and Shadow Controller hosts must be on the same subnet.

When using a private (non-routable) virtual node network, the Primary Controller and Shadow Controller hosts need not be on the same subnet, and you may access the web interface by navigating to the IP address of any Gateway host without a port number. See Private (Non-Routable) Network and Accessing HPE Ezmeral Container Platform (non-SSO).

Platform-level High Availability protects against the failure of any one of the three hosts being used to provide this protection. The warning message appears if either the Shadow Controller or Arbiter host fails, even if the Controller host is functioning properly.



When platform High Availability is enabled:

* The Controller host manages HPE Ezmeral Container Platform, and the Shadow Controller and Arbiter hosts function as Worker hosts.
* If the Controller host fails, the Arbiter host switches management to the Shadow Controller host, and there is no protection against any failure of the Shadow Controller host(s).
* If the Arbiter host fails, the Controller host continues working, and there is no protection against any further failure of the Controller or Shadow Controller host(s).

The following actions occur when a failure of a High Availability host occurs:

* If the Controller host has failed, failover over to the Shadow Controller host occurs, and the deployment begins running in a degraded state. This process usually takes 2-3 minutes, during which you will not be able to log in to the web interface.
* If the Shadow Controller or Arbiter host fails, the Controller host continues running in a degraded state.
* A message appears in the upper right corner of the web interface warning you that the deployment is running in a degraded state. Use the **Service Status** tab of the Platform Administrator **Dashboard** (see [Dashboard - Platform Administrator](https://docs.containerplatform.hpe.com/53/reference/epic/epic-administration/Dashboard__Platform_Administrator.html)) to see which host has failed and which services are down.
* A root cause analysis of the host failure occurs, as do automated attempts to recover the failed host. If recovery is possible, the failed host will come back up and normal operation will resume.
* If the problem cannot be resolved, the affected host will be left in an error state and you will need to manually diagnose and repair the problem (if possible) and then reboot that host. If rebooting solves the problem, then the failed host will come back up and normal operation will resume. If this does not solve the problem, contact HPE Technical Support for assistance.

Enabling platform High Availability protection is a two-stage process: First, you must designate one Worker host as the Shadow Controller host in the **Controllers** tab of the **Controllers/Upgrades** screen. This is a one-time assignment; you cannot transfer the Shadow Controller role to another Worker host. Second, you must enable High Availability protection and then assign the Arbiter role to a third Worker host, also in the **Controller**tab of the **Controllers/Upgrades** screen.

High Availability cannot be enabled if any virtual clusters already exist. If you create one or more virtual clusters before deciding to enable High Availability, you should delete those clusters before proceeding with the High Availability. In general, you should implement High Availability immediately after installation, as described in Controllers Tab.

### Virtual Cluster High Availability

Some Big Data applications allow you to create a virtual cluster with High Availability protection. This is separate and distinct from HPE Ezmeral Container Platform High Availability described above, as follows:

* A cluster with High Availability enabled for that virtual cluster is still dependent on HPE Ezmeral Container Platform. If the Controller host fails, the virtual cluster will also fail.
* If the HPE Ezmeral Container Platform deployment has High Availability enabled and the master node of a virtual cluster fails, that virtual cluster will fail if High Availability for that virtual cluster was not enabled when that virtual cluster was created.

The following table displays the relationship between virtual cluster and HPE Ezmeral Container Platform High Availability protection under a variety of scenarios:

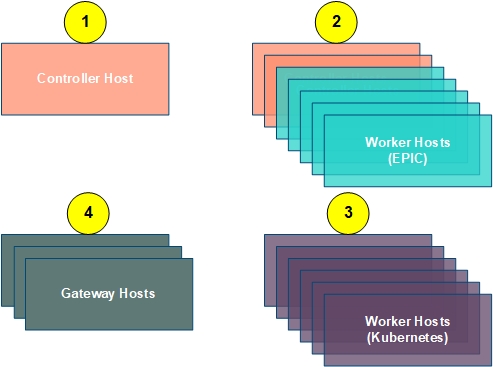
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****Failure Type**** | ****No HA**** | ****Cluster HA only**** | ****Container Platform HA only**** | ****Both HA**** |
| HPE Ezmeral Container Platform Controller host down; virtual cluster master node up. | X | X | protected | protected |
| HPE Ezmeral Container Platform Controller host up; virtual cluster master down. | X | protected | X | protected |
| HPE Ezmeral Container Platform Controller host up; other virtual cluster node down. | protected | protected | protected | protected |

### Gateway Host High Availability

You can add redundancy for Gateway hosts by mapping multiple Gateway host IP addresses to a single hostname. When this is done, then either the DNS server or an external load balancer will load-balance requests to the hostname among each of the Gateway hosts on a round-robin basis. This ensures that there is no single point of failure for the Gateway host.

## Controller, Gateway, and Worker Hosts

A host is either a physical server or a virtual server, located on your premises or in a public cloud, which is available to HPE Ezmeral Container Platform. Virtual nodes/containers and virtual clusters consisting of multiple virtual nodes/containers reside on the hosts.



The types of hosts are:

* **Controller (1):** The Controller host is the host where you initially install HPE Ezmeral Container Platform. This host controls the rest of the hosts in the deployment. It is located either on a physical server or virtual machine or on a public cloud instance. HPE Ezmeral Container Platform is supported on the operating systems listed below:

**OS Support**

HPE Ezmeral Container Platform (ECP) supports the following operating systems:

|  |  |  |  |
| --- | --- | --- | --- |
| ****ECP Version**** | ****CentOS Support**** | ****RHEL Support**** | ****SUSE Support**** |
| **5.3** | 7.8, 7.9, 8.1\*, 8.2\* | 7.8, 7.9, 8.1\*, 8.2\* | 15 SP2 |
| **5.2** | 7.7, 7.8, 7.9 | 7.7, 7.8, 7.9 | 15 SP2 |
| **5.1** | 7.6, 7.7, 7.8, 7.9 | 7.6, 7.7, 7.8, 7.9 | 15 SP1 |
| **5.0** | 7.6, 7.7, 7.8, 7.9 | 7.6, 7.7, 7.8, 7.9 | N/A |

You must have the operating system installed on each of the hosts that you will be using before beginning the installation process. If platform High Availability is enabled, then the deployment must include a Shadow (backup) Controller host and an Arbiter host.

* **Worker (2):** Worker hosts are under the direct control of the Controller host. Host resources are dynamically allocated to the virtual clusters and jobs within each tenant as needed, based on user settings and resource availability. This dynamic resource allocation means that HPE Ezmeral Container Platform achieves a much higher host utilization rate than traditional Hadoop, Spark, and AI/ML deployments. The Worker hosts are located either on your infrastructure (physical server or virtual machine) or on public cloud VMs (for cloud and hybrid deployments). Each Worker host can manage multiple containers/virtual nodes. For example, a virtual cluster that consists of four virtual nodes/containers could potentially be located on a single Worker host. Separate Worker hosts run Kubernetes and legacy EPIC container orchestrator clusters.

If platform High Availability is enabled, then two of these Worker hosts will have additional roles:

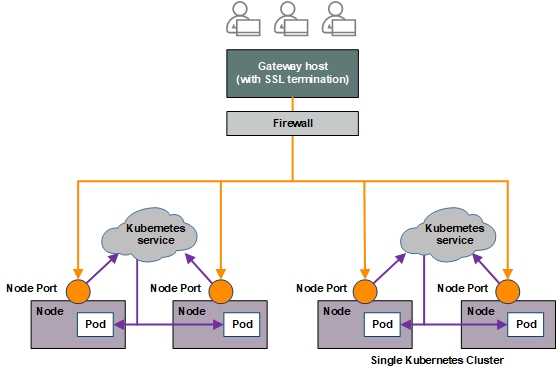
* + One Worker host will be designated as the Shadow Controller host. If the primary Controller host goes down, then this Worker host will assume the Controller functions.
  + Another Worker host will be designated as the Arbiter, which triggers the Shadow Controller host to assume the Controller role if the primary Controller host fails.
* **Kubernetes Hosts (3): Kubernetes hosts are under the direct control of the Controller host. These hosts can be grouped into one or more Kubernetes clusters. These clusters are then available as targets for native Kubernetes workloads spanning microservices, databases (e.g. MySQL), analytics processing (e.g. Spark on K8s) and AI/ML (e.g. KubeFlow). There must always be an odd number of Kubernetes Master Hosts, and any number of Kubernetes Worker hosts.**
* **Gateway (4):** These hosts are also required. The Gateway hosts map both the IP address of the Controller host and the private IP endpoints of services running on the virtual nodes/containers inside the virtual clusters to publicly-accessible IP addresses/ports.

## Gateway Hosts

A Gateway host is a required first-class role that is managed by HPE Ezmeral Container Platform in a manner similar to the Controller and Worker hosts.

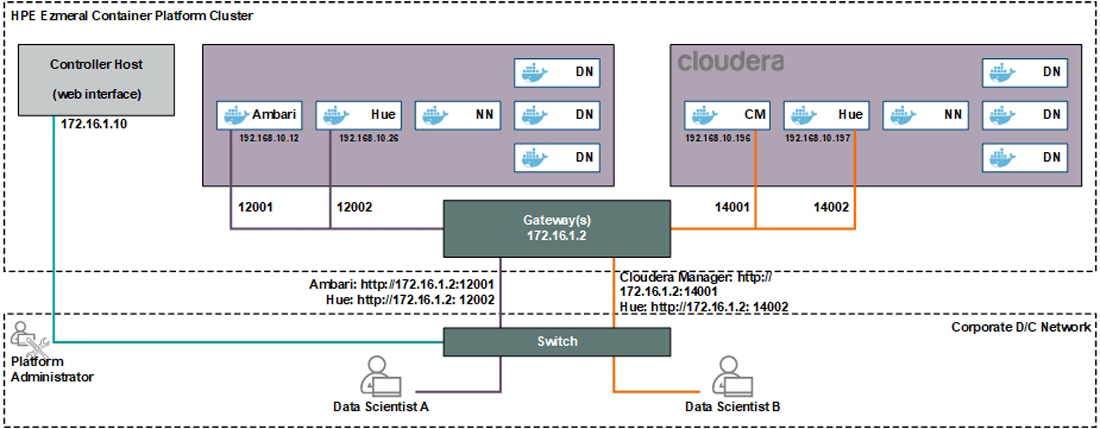
Gateway hosts allow for clear separation of network zones by providing the following:

* **Kubernetes orchestrator:** This diagram displays the logical architecture of Kubernetes clusters and Gateway hosts within a deployment:



* A simple, secure, and fully-managed control path for end users and admins alike to access Kubernetes API servers (such as when handling kubectl commands) as well as the service endpoints of multiple Kubernetes clusters. HPE Ezmeral Container Platform dynamically manages endpoints, including kubeconfig contents, as clusters and services are created, deleted, or updated.
* Automated load balancing for Kubernetes masters and services. Kube API traffic (via kubectl etc.) is load-balanced to both multi-master highly-available Kubernetes clusters and multi-replica NodePort services.
* SSL termination to container service access points.
* Interoperability with any ingress controller and NodePort service definition for maximum flexibility.
* **Legacy EPIC container-orchestrator:** One or more Gateway host(s) are required when the IP addresses used by the virtual nodes/containers/Kubernetes pods are private and non-routable, meaning that the virtual nodes cannot be accessed via the corporate network.

This diagram displays the logical architecture of non-Kubernetes clusters and Gateways hosts within a deployment:



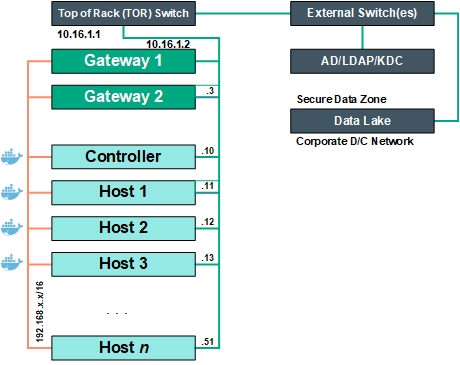
Unlike Worker hosts, Gateway hosts do not run virtual nodes/containers. Instead, they enable access to user-facing services such as Notebooks, Hue console, and/or SSH running on containers via an instance of the High Availability Proxy (HAProxy). You can configure multiple Gateway hosts with a common Fully Qualified Domain Name (FQDN) for round-robin load balancing and High Availability. You may also use a hardware load balancer in front of the multiple Gateway hosts, and can also configure one or more custom port range(s) between 10000 and 50000 for use as proxies.

All control traffic to the virtual nodes/Docker containers from end-user devices (browsers and command line), such as https, SSH, and/or AD/KDC, goes through the Gateway host(s), while all traffic from the virtual nodes/Docker containers is routed through the hosts on which those virtual nodes/containers reside.

Support for multiple subnets increases Gateway host flexibility. For example, you can use "small" virtual machines that meet all Gateway host requirements located on different racks or in different areas of your network, instead of having to place these hosts on the same rack as Worker hosts. This can help optimize resource usage.

## Physical Architecture

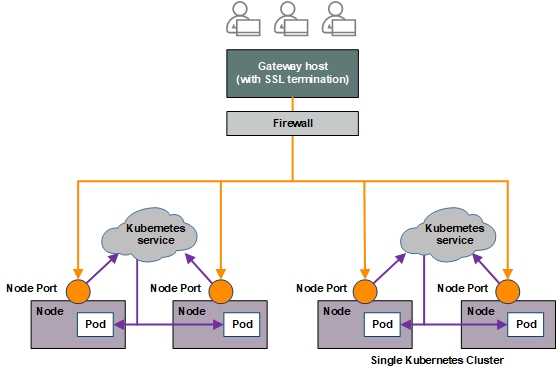
This diagram displays the physical architecture of a deployment with Gateway hosts.



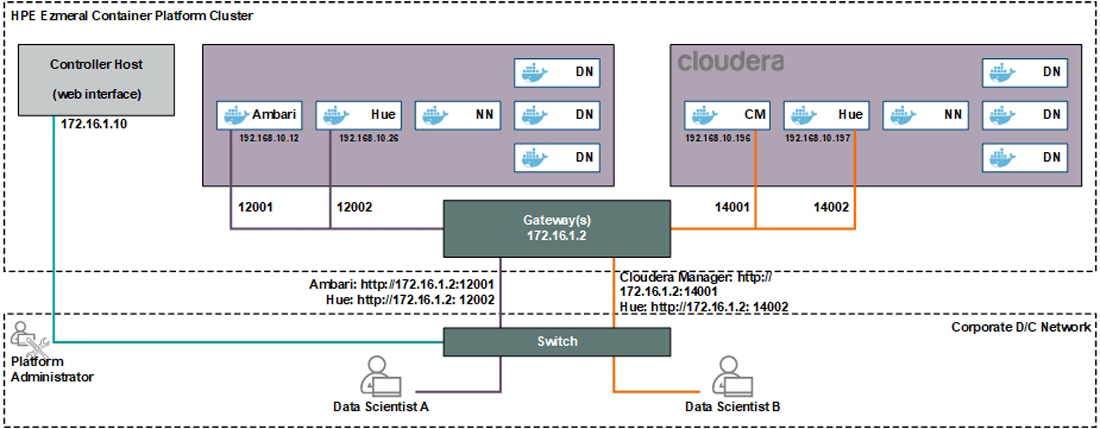
## Logical Architecture

The logical Gateway host architecture varies depending on whether or not one is using a Kubernetes cluster, as follows:

* This diagram displays the logical architecture of Kubernetes clusters and Gateway hosts within a deployment:



* This diagram displays the logical architecture of non-Kubernetes clusters and Gateways hosts within a deployment:



## HAproxy Service

Gateway hosts use the HAproxy service to enable load balancing and SSL termination. The Platform Administrator must install one or more host(s) as the Gateway hosts, as described in [Gateway Installation Tab](https://docs.containerplatform.hpe.com/52/reference/global-settings/gateway-lb/Gateway_Installation_Tab.html). Once the Gateway hosts are set up, the HPE Controller will automatically start using the Gateway proxy set for container port mappings.

* Service endpoints for a Kubernetes cluster on a deployment appear in the **Gateway Mappings** column of the table on the **Kubernetes Service Endpoints** screen. See [Endpoints Tab](https://docs.containerplatform.hpe.com/52/reference/kubernetes-applications/general/The_Kubernetes_Applications_Screen.html).
* Service endpoints for a non-Kubernetes cluster on a deployment appear on the **Node(s) Info** tab of the **Cluster Details** screen. See [Node(s) Info Tab](https://docs.containerplatform.hpe.com/52/reference/epic/working-with-big-data/virtual-clusters/Viewing_Cluster_Details.html).

## Basic Requirements/Limitations

The Platform Administrator must designate one or more Gateway hosts. These hosts only run the HAproxy service and cannot be included in container clusters. Gateway hosts must meet the following minimum requirements:

* Similar memory/CPU configuration as the minimal requirements for non-proxy (Compute) Worker hosts.
* 50GB or more of available disk space. There is no need for extra disks for node/system storage.
* Gateway hosts need not be on the same subnet as the Controller/Standby Controller, and any Worker host(s).
* Ports 10000-50000 on the Gateway hosts will be used for port mapping. The sysctl utility will be setup on the Gateway hosts to prevent misallocation of these port ranges. The HAproxy service will bind to ports in this range.
* The iptables service will be automatically disabled on the Gateway host(s) during the installation.
* One or more Gateway set(s) can be created for load sharing. All of the hosts in a Gateway set must have both an individual hostname and an externally-resolvable common hostname. The DNS server must be configured to do this. For example, the following two Gateway hosts have the common hostname sample-lab-proxy.enterprise.com:
  + 10.32.2.94 hostname-1.enterprise.com sample-lab-proxy.enterprise.com
  + 10.32.2.96 hostname-2.enterprise.com sample-lab-proxy.enterprise.com
* A Gateway host that belongs to a proxy set can be decommissioned provided the following conditions are met:
  + No active container port mappings are present.
  + Container port mappings are present and there is at least one other proxy node available in the set.
* A new Gateway host can be added to an existing set at any time. The new Gateway host will automatically be configured with the port instance mappings for that set.
* A new Gateway host can be added anytime; however, the newly added Gateway host will be used only for future cluster mappings.
* At the time of instance launch, all Gateway hosts in the set must be accessible to create mappings. If they are not available, then cluster creation will fail.
* Physical hosts must be on a routable network/standard corporate subnet, for both DataTap access and Kerberos access from the containers).

## Non-Routable Network Management

**Note:** This section only applies to non-Kubernetes container networks.

During installation, the Platform Administrator is able to specify whether to use a routable or non-routable container network range. This configuration can be changed post-installation, provided that no clusters have been launched. See [Network Tab](https://docs.containerplatform.hpe.com/52/reference/global-settings/settings/Network_Tab.html).

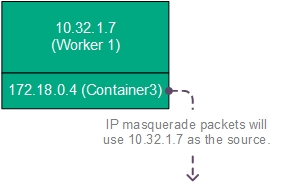
The default non-routable network range for the containers is 172.18.0.0/16. If this network is already in use, then the Platform Administrator must specify a different range.

By default, the Controller cannot communicate with the vAgent service for container orchestration, because the container network cannot be accessed through the directly-connected switch as vAgent runs within the container, which is on a different network. In order to allow the Controller to communicate with the vAgent service in a container, a route must be specifically configured. The Gateway for this route will be the host where the container is launched, as described below.

Communication between the Gateway hosts and containers occurs via the same VXLAN tunnels that are used for communications between containers. Traffic flowing between a host and the containers on that host uses a special VXLAN tunnel outside the normal 2-4094 range. Flow rules on the Gateway and external bridges identify this traffic and take appropriate action.

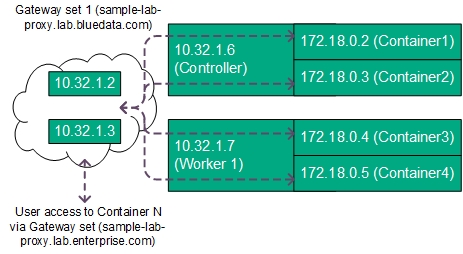
Every physical host must have an IP address in order to make use of the VXLAN tunnels. The bd\_public network on each host will be assigned a unique IP address (taken from the same /16 floating network range). Gateway hosts will also use Open VSwitch (OVS) and tunnels to proxy container traffic.

An IP masquerading rule is set up on each physical host to allow packets originating from the containers to reach any of the HPE Controller hosts (Primary, Shadow, or Arbiter) or an external Kerberos/AD server. This happens automatically after successfully launching a container, as shown in the following image.



## Proxy Mapping Management

Setting up one or more Gateway host(s) will allow users to access containers using a set of service endpoint proxy mappings, as shown here.



Each Gateway set can consist of one or more Gateway host(s). When a cluster launches, information is collected about the service endpoints configured for that cluster. These service endpoints will be those defined for services being deployed within that cluster, and each service endpoint will be mapped to a specific port regardless of whether that service uses HTTP, HTTPS, or TCP.

The Controller uses a scheduling algorithm to decide which Gateway set to use for creating port mappings. This is simply based on which Gateway set has fewer port mappings and whether or not all of the hosts in the Gateway set are accessible. If the Controller cannot find an available Gateway set, then cluster launch will fail with an error message returned to the user.

Once the Controller identifies an available Gateway set, it will allocate the necessary ports from the reserved range. A message will be sent to the service running on each Gateway host to create the appropriate proxy mappings. Mappings will be automatically deleted when a cluster is deleted, and the ports will be freed up for use by the next cluster.

Gateway hosts within a Gateway set have failover ability. If a Gateway host is down, the other host(s) will provide the port mappings for virtual clusters. However, all active hosts within a set must be available at the time of cluster creation. The user's DNS server will determine which specific host will receive the traffic within a single set. Typically, a round-robin configuration on the DNS server will serve this purpose.

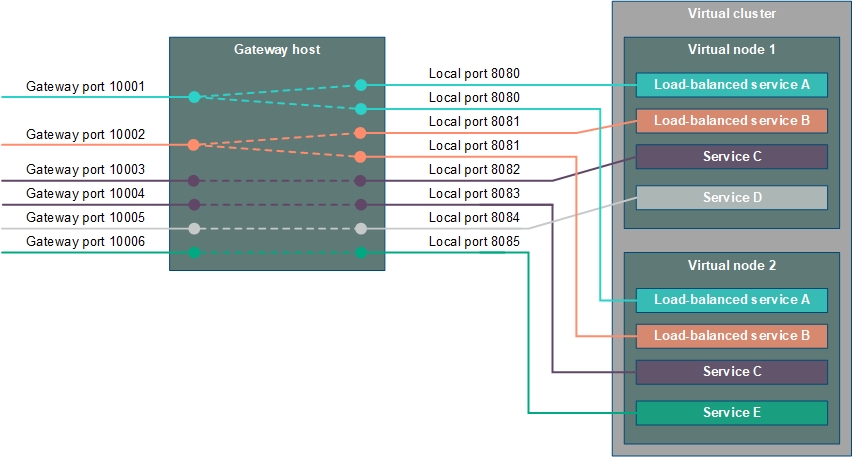
Multiple Gateway sets may be used to:

* Create a larger number of port mappings than allowed for a single set
* Load-balance traffic between two sets. See Load Balancing.

Mappings between sets are not shared. If an entire Gateway set is disabled, those mappings will no longer function.

## Load Balancing

Load-balanced access to services running on virtual nodes/containers within a cluster is supported as shown here:



This example shows a virtual cluster that consists of two virtual nodes/containers, where:

* Services A and B are running on both virtual nodes and are configured for load-balancing.
* Service C is running on both virtual nodes but is not configured for load-balancing. In this example, this service consumes two externally visible Gateway ports (10003 and 10004).
* Service D is only running on Virtual node 1.
* Service E is only running on Virtual node 2.

In this example:

* The Gateway host will assign all requests made to port 10001 to the instance of Service A that is running on either Virtual node 1 or Virtual node 2, on a round-robin basis. This means that if the previous request to port 10001 was routed to Virtual node 1, then the next request to that port will be routed to Virtual node 2, and vice-versa.
* The Gateway host will assign all requests made to port 10002 to the instance of Service B that is running on either Virtual node 1 or Virtual node 2, on a round-robin basis, as described above.
* Any request made to Gateway port 10003 will be routed to the instance of Service C that is running on Virtual node 1 because this service is not configured for load-balancing.
* Any request made to Gateway port 10004 will be routed to the instance of Service C that is running on Virtual node 2 because this service is not configured for load-balancing.

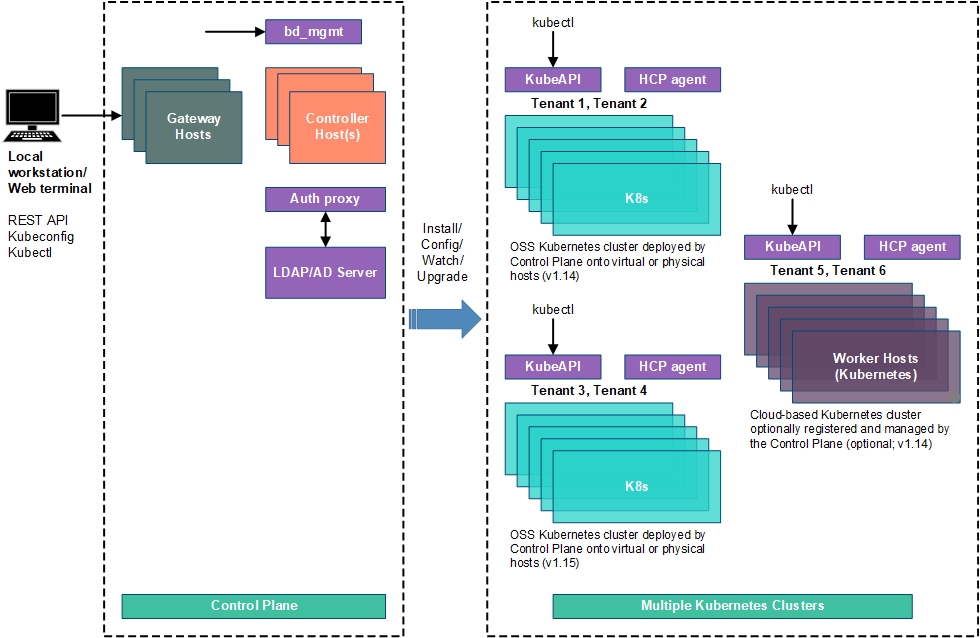
There are two ways to use load balancing:

**Automatic:** If at least one Gateway host is present, then you can create a virtual cluster using an App Store image that includes one or more service(s) that are flagged for load-balancing. The Gateway host will map all running instances of each load-balanced service to a single Gateway port, as described above. See [Gateway Hosts](https://docs.containerplatform.hpe.com/52/reference/universal-concepts/Gateway_Hosts.html) and [Creating a New Cluster](https://docs.containerplatform.hpe.com/52/reference/epic/working-with-big-data/virtual-clusters/Creating_A_New_Cluster.html).

**Manual:** If at least one Gateway host is present, then you can manually add a service that supports load balancing as described in [Adding a Cluster Service](https://docs.containerplatform.hpe.com/52/reference/epic/common-cluster-operations/cluster-services/Adding_a_Cluster_Service.html). No check occurs to determine whether a manually-added service actually supports load-balancing.

## Kubernetes Physical Architecture

This diagram depicts the physical Kubernetes architecture within HPE Ezmeral Container Platform.



Local workstations are used to:

* Access the web interface.
* Directly access service endpoints running on containers via the Gateway host(s) in the format <gateway\_ip>:<port>, where <gateway\_ip> is the IP address of a Gateway host and <port> is the mapped port of the service endpoint. For example, assume that a Kubernetes container is running a service endpoint that can be accessed remotely, and that the Gateway host has an IP address of 192.168.100.150. If the Gateway host has mapped the service endpoint running on the Kubernetes container to Port 12345, then you can access that endpoint by navigating to 192.169.100.150:12345. Gateway hosts can also offer load-balanced access to multiple instances of the same service endpoint within a cluster. See Gateway Hosts and Load Balancing.
* Access the REST API.
* Access Kubernetes clusters using Kubeconfig and Kubectl.

The Platform Control Plane consists of:

* Either one, three, or five Controller hosts. See Controller, Gateway, and Worker Hosts.
* The Controller host(s) authenticate users via the authentication proxy, using either the internal database or an LDAP/AD server. See User Authentication. The Authenticating Proxy consists of:
  + A server-side application that receives API requests from clients (usually from the kubectl tool) and (if they are properly authenticated) adds one or more groups to the request. The authenticating proxy then forwards the request to the kube-apiserver pod, and forwards any responses to the request back to the user.
  + A client-side kubectl plugin.

The Control Plane handles the installation, configuration, upgrade, and monitoring of Kubernetes hosts, clusters, and tenants.

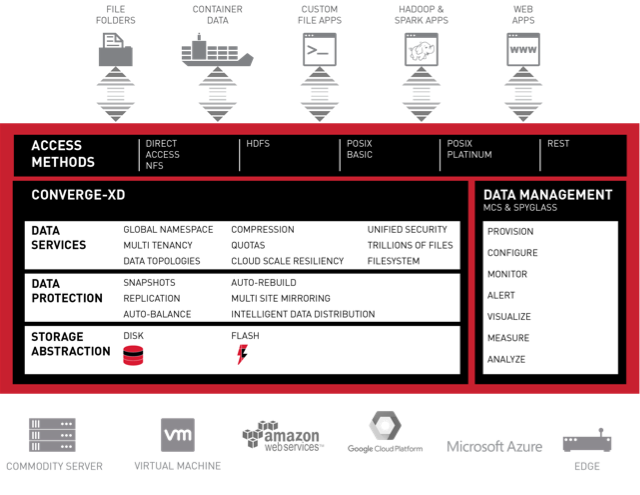
# HPE Ezmeral Data Fabric

## HPE Ezmeral Data Fabric XD Distributed Store

XD Cloud-Scale Data Store provides exabyte-scale data store for building intelligent applications with the HPE Ezmeral Data Fabric Converged Data Platform. XD includes all the functionality you need to manage large amounts of conventional data.

**Why XD?**

XD can be installed on SSD- and HDD-based servers. It includes the filesystem for data storage, data management, and data protection, support for mounting and accessing the clusters using NFS and the FUSE-based POSIX (basic, platinum, or PACC) clients, and support for accessing and managing data using HDFS APIs. The cluster can be managed using the Control System and monitored using HPE Ezmeral Data Fabric Monitoring (Spyglass initiative). XD is the only Cloud-Scale Data store that enables you to build a fabric of exabyte scale. XD supports trillions of files, 100s of 1000s of client nodes and can run on Edge Cluster, on-prem data centers and the public cloud.



## Filesystem

Discusses the features of the data-fabric filesystem, and provides a comparison with the Hadoop Distributed File System (HDFS).

The data-fabric filesystem provides a unified data solution for structured data (tables) and unstructured data (files).

The data-fabric filesystem is a random, read-write distributed filesystem that allows applications to concurrently read and write directly to disk. The Hadoop Distributed File System (HDFS), by contrast, has append-only writes and can only read from closed files. As HDFS is layered over the existing Linux filesystem, a large number of input/output (I/O) operations decrease the cluster’s performance. The data-fabric filesystem also eliminates the NameNode associated with cluster failure in other Hadoop distributions, and enables special features for data management, and high availability.

The storage system architecture used by data-fabric filesystem is written in C/C++ and prevents locking contention, eliminating performance impact from Java garbage collection.

The following table highlights some of the features of the data-fabric filesystem:

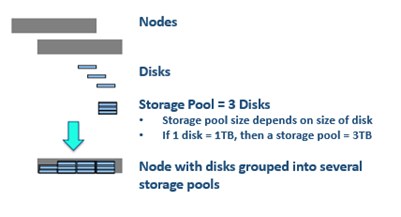
|  |  |
| --- | --- |
| Feature | Description |
| Storage pools | A group of disks to which the data-fabric filesystem writes data. |
| Containers | An abstract entity that stores files and directories in the data-fabric filesystem. A container always belongs to exactly one volume, and can hold namespace information, file chunks, or table chunks for that volume. |
| CLDB | A service that tracks the location of every container. |
| Volumes | A management entity that stores and organizes containers. Used to distribute metadata, set permissions on data in the cluster, and for data backup. A volume consists of a single name container, and a number of data containers. |
| Direct Access NFS | Enables applications to read and write data directly on to the cluster. |
| POSIX Clients | The loopbacknfs, and FUSE-based POSIX clients connect to one or more data-fabric clusters, and allow app servers, web servers, and applications to write data directly, and securely to the data-fabric cluster. |

## Storage Pool

*Describes what storage pools are.*

The filesystem storage architecture consists of multiple storage pools that reside on each node in a cluster. A storage pool is made up of one or more disks grouped by the data-fabric file system. The default number of disks in a storage pool is three. The containers that hold the data-fabric filesystem data are stored in, and replicated among the storage pools in the cluster.

The following image represents disks grouped together to create storage pools that reside on a node:



Write operations within a storage pool are striped across disks to improve write performance. Stripe width and depth are configurable with the disksetup script. As the data-fabric filesystem performs data replication, you do not need to configure RAID.

## Containers and the CLDB

*Describes what containers are, and the role of the Container Location Database (CLDB) in managing them.*

The data-fabric filesystem stores data in abstract entities called containers that reside on storage pools. Each storage pool can store many containers. Blocks enable full read-write access to the data-fabric filesystem, with efficient snapshots.

An application can write, append, or update more than once in the data-fabric filesystem, and can also read a file as it is being written. In other Hadoop distributions, an application can only write once, and the application cannot read a file as it is written.

On average, a container size is 10-30 GB. The default container size is 32GB. Large number of containers allow for greater scaling and allocation of space in parallel, without bottlenecks.

Described from the physical layer:

* Files are divided into chunks.
* The chunks are assigned to containers.
* The containers are written to storage pools, which are made up of disks on the nodes in the cluster.

The following table compares the data-fabric filesystem storage architecture to the HDFS storage architecture:

|  |  |  |
| --- | --- | --- |
| Storage Architecture | HDFS | Data Fabric Filesystem |
| Management layers | Files, directories and blocks, managed by NameNode. | Volume, which holds files and directories, made up of containers, which manage disk blocks and replication. |
| Size of file shard | 64MB block | 256MB chunk |
| Unit of replication | 64MB block | 32GB container |
| Unit of file allocation | 64MB block | 8KB block |

To preserve data, the data-fabric filesystem automatically replicates containers across various nodes on the cluster. Container replication creates multiple synchronized copies of the data across the cluster for failover. Container replication also helps localize operations, and ensures that read operations occur in parallel. When a disk or node failure brings a container’s replication levels below a specified replication level, the data-fabric filesystem automatically re-replicates the container elsewhere in the cluster until the desired replication level is achieved. A container only occupies disk space when an application writes to it.

The CLDB (Container Location Database) maintains information about the location of every container in the cluster, defines the container precedence in the replication chain, and organizes container content updates across the replication chain. It runs as a system of independent servers, only one of which is a master at any time.

The data-fabric filesystem and other services (such as NFS Gateway and POSIX) send heartbeat (HB) messages to the master CLDB. The CLDB is registered with ZooKeeper, and the master CLDB to ZooKeeper connection is kept alive by sending a probe message every few seconds. The CLDB service tracks the location of every container, and uses these HB messages to determine the state of all containers on that node. The CLDB actively participates in the failover of a node in the event of a node failure.

## Understanding Replication

*Describes how replication works, and how to configure the replication factor.*

Volumes are stored as pieces called containers that contain files, directories, and other data. By default, the maximum container size is 32 GB. The data-fabric administrator sets the maximum container size using the cldb.container.sizemb parameter (see the [config](https://docs.datafabric.hpe.com/62/ReferenceGuide/config.html) commands). Containers are replicated to protect data. Normally, each container has three copies stored on separate nodes to provide uninterrupted access to all data, even if a node fails.

For each volume, you can specify a desired and minimum data replication factor, and a desired and minimum namespace (name container) replication factor.

When enabled, the CLDB manages the namespace container replication separate from the data container replication. Use this capability when you have low volume replication, but want to have higher namespace replication.

**Note:** The namespace container parameters, nsreplication or nsminreplication, must be the same or larger than the equivalent data replication parameter, replication or minreplication.

### Data Replication

* The replication factor is the number of replicated copies that you need for normal operation and data protection. When the number of copies falls below the desired replication factor, but remains equal to or above the minimum replication factor, the CLDB actively creates additional copies of the container while trying to minimize the impact of making an additional copy of the container. Re-replication occurs after the timeout specified in the cldb.fs.mark.rereplicate.sec parameter (configurable using the [configuration API](https://docs.datafabric.hpe.com/62/ReferenceGuide/maprcli-REST-API-Syntax.html)). The minimum replication factor is 1 and the maximum is 6 (default: 3).

* The minimum replication factor is the smallest number of copies you need in order to adequately protect against data loss. When the replication factor falls below this minimum value, re-replication occurs aggressively if data is being actively written to the container. If the enforceminreplicationforio property is set to true, writes succeed only when the minimum replication factor requirements are met. If the enforceminreplicationforio property is set to true and the minimum number of copies are not available, the client is asked to retry. In the case of a:
  + Hard mount, the client might try for up to 10 minutes and then return an error
  + Soft mount, the client might return an error
* The minimum replication factor is 1 and the maximum is 6 (default: 2). In all cases, the minimum replication factor cannot be greater than the replication factor. When you increase the minimum replication factor, if the enforceminreplicationforio property (configurable at the volume level) is set to true, the requirement to maintain a minimum number of copies is not enforced during writes until new copies of all containers associated with the volume are created.

### Name Container Replication

* The namespace replication factor is the number of namespace container replicated copies that you need for normal operation and data protection. When the number of copies falls below the desired replication factor, but remains equal to or above the minimum replication factor, the CLDB actively creates additional copies of the container while trying to minimize the impact of making an additional copy of the container. Re-replication occurs after the timeout specified in the cldb.fs.mark.rereplicate.sec parameter (configurable using the [configuration API](https://docs.datafabric.hpe.com/62/ReferenceGuide/maprcli-REST-API-Syntax.html)). The minimum replication factor is 1 and the maximum is 6 (default: 3).
* The minimum namespace replication factor is the minimum number of namespace container replicated copies you want in order to adequately protect against data loss. When the replication factor falls below this minimum, re-replication occurs aggressively if data is being actively written to the container. If the enforcemineplicationforio property (configurable at the volume level) is set to true, writes succeed only when the minimum replication factor requirements are met. If this property is set to true and minimum number of copies are not available, the client is asked to retry. In the case of a:
  + Hard mount, the client tries for up to 10 minutes and then return an error
  + Soft mount, the client returns an error

The system does not wait for lost replicas to become available again. The minimum replication factor is 1 and the maximum is 6 (default: 2). In all cases, the minimum replication factor cannot be greater than the replication factor. When you increase the minimum replication factor, if the enforceminreplicationforio property is set to true, the presence of the minimum number of copies is not enforced during writes until new copies of all containers associated with the volume are created.

If any containers in the CLDB volume fall below the minimum replication factor, the cluster is inaccessible until aggressive re-replication restores the minimum level of replication. If a disk failure is detected, any data stored on the failed disk is re-replicated without regard to the timeout specified in the cldb.fs.mark.rereplicate.sec parameter.

If all copies of a container, which are neither under nor over replicated, are on the same rack, data-fabric automatically detects and distributes the copies, such that they are all not on the same rack, after 12 hours. If a container is under replicated and data-fabric is unable to find a different rack for the new copy, the creation of the copy is deferred. If another rack is unavailable for the new copy after 3 hours, data-fabric creates a copy of the container on the same rack and if this results in all copies of the container being on the same rack, data-fabric distributes the copies after 12 hours. Also, during replication, data-fabric tries to defer the scenarios where all copies end up on the same rack. As per deferring policy:

* If a container has copies less than the "minimum replication" but greater than 2 and if both copies end up on the same rack, then data-fabric tries to create the third copy on a different rack for up to 3 hours.
* If a container has copies more than the minimum but less than the desired and if all copies are on the same rack, then data-fabric tries to create the next copy on a different rack for up to 3 hours.

If you do not set the namespace (NS) replication and minimum namespace replication values explicitly, they assume the same values as (data) replication and minimum replication respectively. This means that all changes to (data) replication and minreplication parameters are also reflected in nsreplication and nsminreplication. If nsreplication or nsminreplication is modified or specified during creation, nsreplication and nsminreplication start assuming values different from replication and minreplication.

## Table Replication vs Mirroring - Understanding the Differences

This section describes the advantages of both Table Replication and Mirroring, to let you determine the best option for your use case.

**Advantages of Table Replication**

1. Table replication replicates each table update instantaneously, in seconds (subject to compute and network resources). Mirroring has a much larger RTO (recovery time objective), in minutes.
2. Table replication also transmits lesser data because it just transmits the actual physical rows and nothing else.
3. In table replication, both the end points are READ-WRITE masters with the option of two-way multi-master replication.
4. Table replication proceeds from Source Table > Destination Gateway(s) > Destination Table, which provides reasonable isolation between the two end point clusters. The source table talks only to the Destination Gateway(s).

When using mirroring, avoid placing table replication sources in the mirror volume. Doing so, creates problems if the mirror is broken and promoted.

For tables and streams, table replication is usually the right choice. However, there are exemptions where mirroring is the best choice.

**Advantages of Mirroring**

1. Since a volume mirror represents a moment in time, there is a higher probability of recovering from a volume than from multiple tables.
2. You can retain old states of a mirror. If you have deleted a bunch of data in your tables and table replication has replicated those changes, then you can recover your data from a mirror.
3. Mirrors are helpful during development. Create a read-write mirror and use for development. Revert it to the last mirrored state and start over. The point is that you can revert the entire volume to a known state, as needed.
4. Use local mirror(s) to increase read throughput.
5. You can use mirrors to obtain traceability and reproducibility during data operations such as machine learning. You can have separate mirrors for different clusters, and operations on one mirror do not affect the other.

## Understanding Topology

*Provides an overview of how to define cluster topology.*

The data-fabric software uses node topology to determine the location of replicated copies of data. Node topology describes the locations of nodes in a cluster. You can define the cluster topology by specifying a topology for each node in the cluster. Use topology to group nodes by rack or switch, to provide a hint as to how data should be replicated to protect against data loss or unavailability because of a switch or rack failure.

In a topology, data-fabric distributes container copies optimally among leaf nodes. For example, in a topology such as europe/uk/london/DC2/room4/row22, where row22 contains multiple racks such as row22/rack1, row22/rack2, row22/rack3, and so on, data-fabric tries to ensure that all copies of the container do not end up on the same rack (for example, rack1). By setting each leaf value to correspond to a physical rack, you can ensure that replicated data is distributed across racks to improve fault tolerance.

### Setting Up Node Topology

*Define node topologies for every node in the cluster.*

Define your cluster's topology by specifying a topology for each node in the cluster. You can use topology to group nodes by rack or switch, depending on how the physical cluster is arranged and how you want data-fabric to place replicated data.

Topology paths can be as simple or complex as needed to correspond to your cluster layout. In a simple cluster, each topology path might consist of the rack only (for example, /rack-1). In a deployment consisting of multiple large datacenters, each topology path can be much longer (for example, /europe/uk/london/datacenter2/room4/row22/rack5/). Data Fabric uses topology paths to spread out replicated copies of data, placing each copy on a separate path. By setting each path to correspond to a physical rack, you can ensure that replicated data is distributed across racks to improve fault tolerance.

### Setting Up Volume Topology

*Specifies how to use volume topology to place volumes on specific racks, nodes, or groups of nodes.*

After you define the node topology for the nodes in your cluster, you can use volume topology to place volumes on specific racks, nodes, or groups of nodes.

This section describes the process of setting up:

* Default volume topology
* Local volume topology
* Custom topology for local volume replicas

## Volumes, Snapshots, and Mirrors

*Describes what Snapshots and Mirrors are, and the advantages of using them for*[*replication*](https://docs.datafabric.hpe.com/62/AdministratorGuide/DetermineVolReplicationFactor.html)*.*

Volumes are a management entity that logically organize a cluster’s data. Since a container always belongs to exactly one volume, that container’s replicas all belong to the same volume as well. Volumes do not have a fixed size and they do not occupy disk space until the data-fabric filesystem writes data to a container within the volume. A large volume may contain anywhere from 50-100 million containers.

The CLI and REST API provide functionality for volume management. Typical use cases include volumes for specific users, projects, development, and production environments. For example, if an administrator needs to organize data for a special project, the administrator can create a specific volume for the project. The HPE Ezmeral Data Fabric filesystem organizes all containers that store the project data within the project volume. A cluster can have many volumes.

The HPE Ezmeral Data Fabric filesystem creates a name container for each volume. The name container stores the volume’s namespace and file chunk locations, along with inodes for the objects in the filesystem. The filesystem stores the metadata for files and directories in the name container, which is updated with each write operation.

The first 64KB of each file in a volume is written to the name container. Data beyond 64KB is written to data containers. Data containers are created only when the file or table data goes above 64KB. Each name or data container is associated with only one volume; volumes may have many associated data containers, but only one name container.

Local volumes are confined to one node, and are not replicated. Local volumes are part of the cluster’s global namespace, and are accessible on the path /var/mapr/local/<host>.

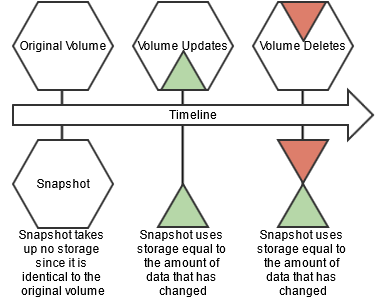
On a cluster with an Enterprise Edition or Enterprise Database Edition license, you can create a special type of volume called a mirror, a local or remote read-only copy of an entire volume. Mirrors are useful for load balancing or disaster recovery. You can also create a snapshot, an image of a volume at a specific point in time. Snapshots are useful for rollback to a known data set.

On a cluster, you can create a tenant share, or volume for tenant users. A tenant share is an isolated space where you can set different policies, quotas, and access privileges for specific users/hosts (referred to as tenants). This allows each tenant to own its own copy of storage space, users, data security, administration, and other such specifications

### Snapshots

Snapshots enable you to [roll back to a known good data set](https://docs.datafabric.hpe.com/62/AdministratorGuide/Snapshots-Restoring-new.html) and recover data always in case of data corruption or accidental deletions, without the help of storage administrators. A snapshot is a read-only image of a volume that provides point-in-time recovery. Snapshots only store changes to the data present in the volume, and as a result make extremely efficient use of the cluster’s disk resources. Snapshots preserve access to historical data, and protect the cluster from user and application errors. You can [create a snapshot manually](https://docs.datafabric.hpe.com/62/ClusterAdministration/data/volumes/CreateSnapshot.html), or automate the process with a schedule. Snapshots are stored in the .snapshots directory. You can always view snapshots from this directory.

The following image represents a mirror volume, and a snapshot created from a source volume:

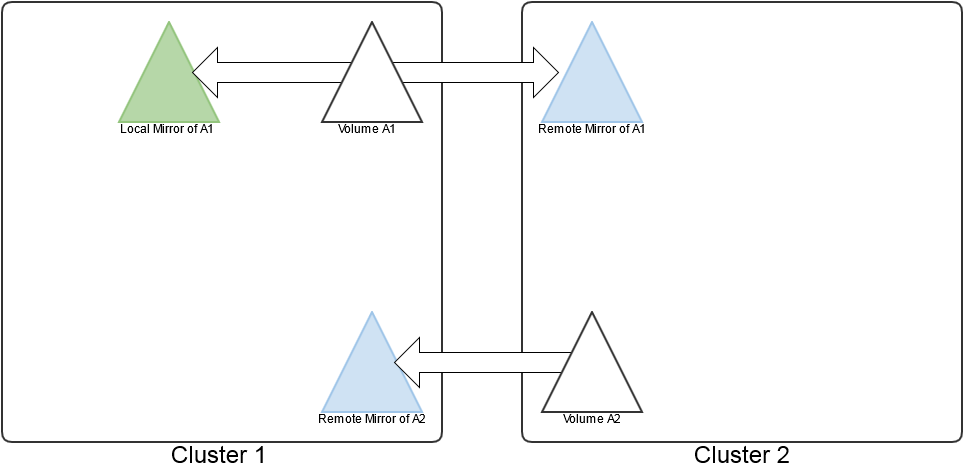


New write operations on a volume with a snapshot are redirected to preserve the original data. Snapshots only store the incremental changes in a volume’s data from the time the snapshot was created. The storage used by a volume's snapshots does not count against the volume's quota.

### Mirror Volumes

Data Fabric provides built-in mirroring to set recovery time objectives and to automatically mirror data for backup. You can create local or remote mirror volumes to mirror data between clusters, data centers, or between on-premise and public cloud infrastructures.

Mirror volumes are read-only copies of a source volume. You can control the schedule for mirror refreshes from the Control System or with the command-line tools. You can create local (on the same cluster) or remote (on a different cluster) mirror volumes from the Control System, or from the command line.



When you create a mirror volume, the HPE Ezmeral Data Fabric filesystem creates a temporary snapshot of the source volume. The mirroring process reads content from the snapshot into the mirror volume. The source volume remains available for read and write operations during the mirroring process. The initial mirroring operation copies the entire source volume. Subsequent mirroring operations only update the differences between the source volume and the mirror volume.

Mirror volumes can be promoted to read-write volumes. The main use case for this feature is to support disaster-recovery scenarios in which a read-only mirror needs to be promoted to a read-write volume so that it can become the primary volume for data storage. In addition, read-write volumes that were mirrored to other volumes can be made into mirrors (to establish a mirroring relationship in the other direction). You can also convert read-write volumes back to read-only mirrors.

## Types of Volumes

*Lists the various types of volumes.*

This glossary explains the different types of volumes.

|  |  |
| --- | --- |
| Term | Definition |
| NC Standard Volume | A non-convertible (NC) standard volume is a volume with read-write capabilities, created *before* data-fabric version 4.0.2. These volumes cannot be converted to standard mirror volumes. If this volume type is designated as a source volume when a mirror volume is created, the mirror volume will be a NC mirror volume.  A NC standard volume is designated as type 0 in the output of the volume info command. For example:  ***maprcli volume info -name oldrw***  lists  ***"mirrortype":0*** |
| Standard Volume | A standard volume is a read-write volume created as of data-fabric version 4.0.2. A standard volume can be converted from read-write to mirror (read-only). If a mirror is created from this type of volume, the mirror can be promoted to a read-write volume.  A standard volume is designated as type rw on the command line. For example:  ***maprcli volume create -name volA -path /testvol -type rw*** |
| NC Mirror Volume | A non-convertible read-only mirror volume is a volume created *before* data-fabric version 4.0.2. This volume type cannot be promoted to a read-write volume, and can only be created from a NC standard volume.  A NC mirror volume is designated as type 1 in the output of the volume info command. For example:  ***maprcli volume info -name oldmirror***  lists  ***"mirrortype":1*** |
| Standard Mirror | Standard mirror is a mirror volume that starts as a read-only volume, and can be promoted to a read-write volume.  A standard mirror volume is designated as type mirror on the command line and can only use a standard volume as its source. For example:  ***maprcli volume create -name volB -path /mirvol -type mirror -source volA*** |