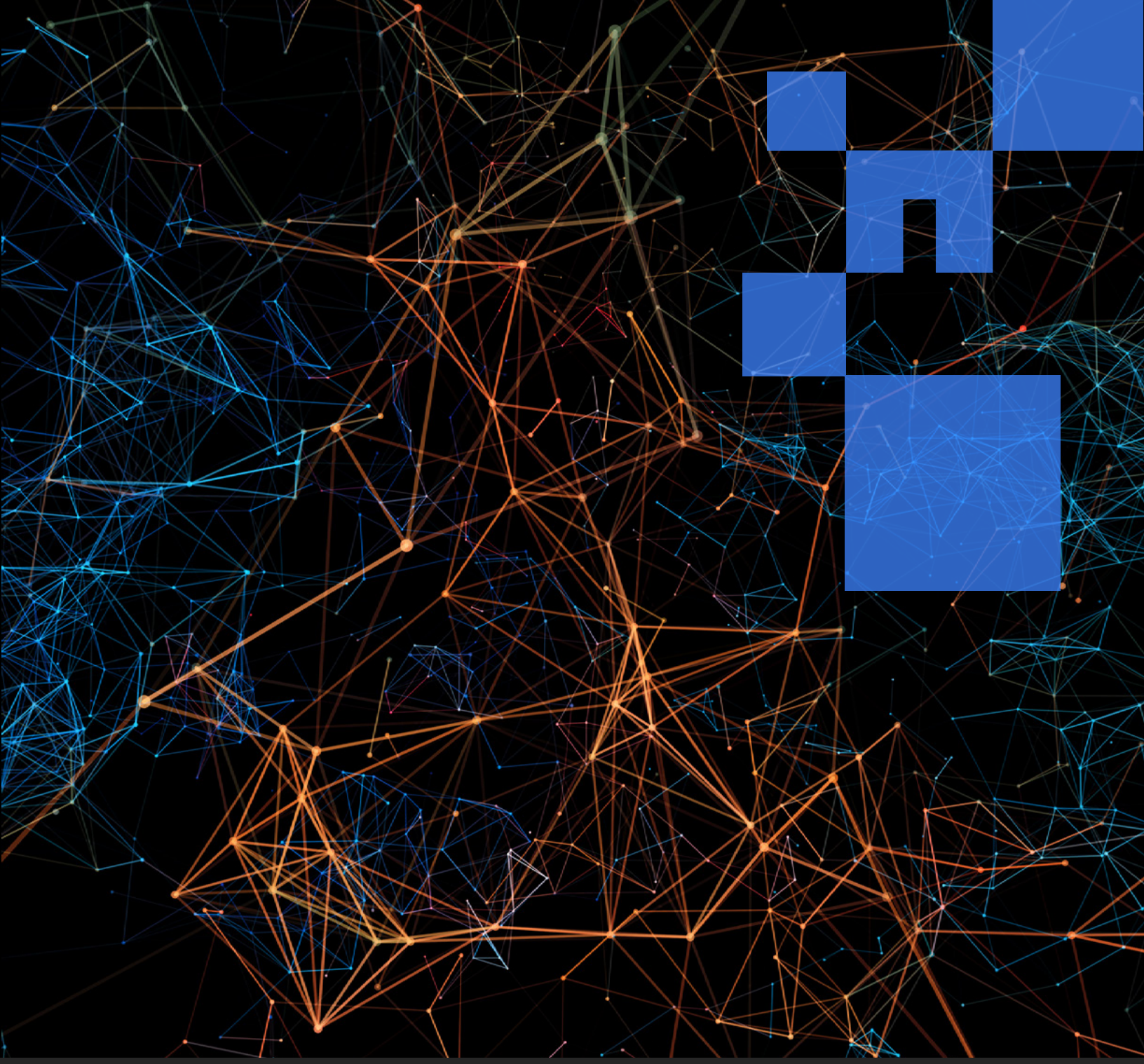
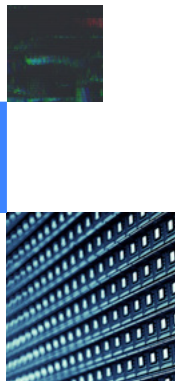


The NetApp Guide to Kubernetes:

Persistent Volumes, Dynamic Provisioning, Cloud Storage

NetApp®



Introduction

Kubernetes is today's most widely-used platform for container and microservices orchestration and provides the scalability and flexibility required for deploying enterprise applications and services.

Managing storage in a Kubernetes cluster with dynamic persistent storage provisioning massively reduces the manual administration required for allocating cloud storage to pods and containers.

In this ebook we'll discuss everything you need to know about Kubernetes, from the technology behind its use, how it works, and how NetApp's Trident provisioner can be used to dynamically provision persistent volumes in the cloud via Cloud Volumes ONTAP.

Table of Contents

NetApp

PART ONE

What Is Kubernetes?

TO UNDERSTAND KUBERNETES, YOU FIRST HAVE TO UNDERSTAND THE CORE COMPONENT OF KUBERNETES ARCHITECTURE: CONTAINERS.

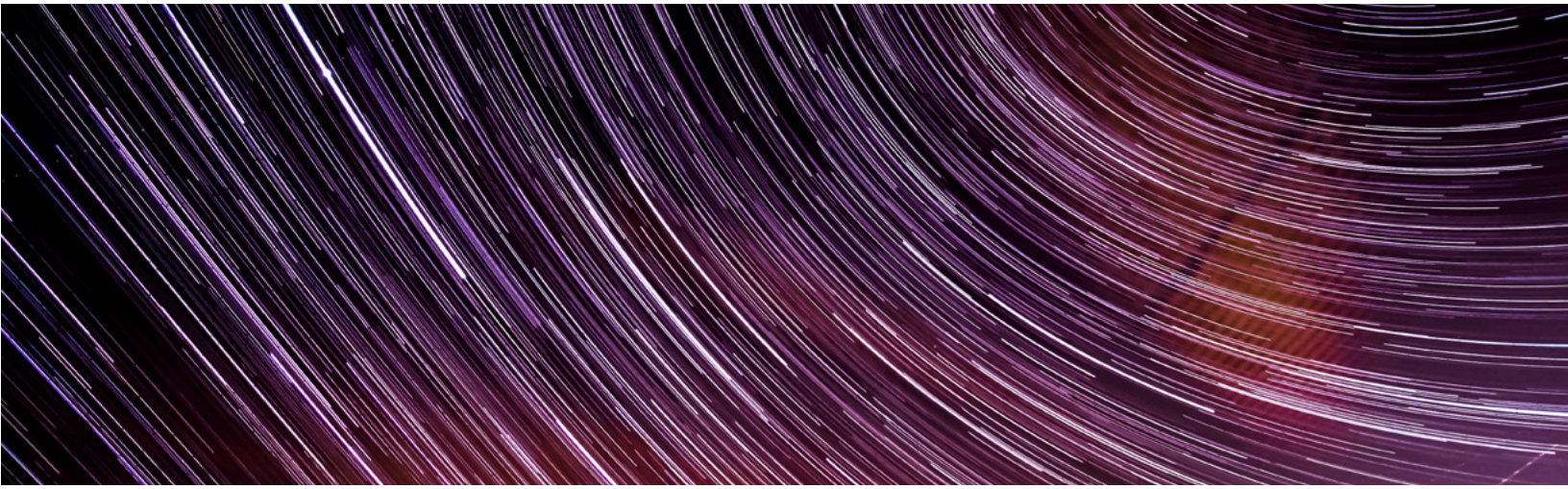
Containers are lightweight, independent units that software developers and DevOps engineers use to encapsulate applications. On deployment, a container provides process and file system separation in much the same way as a virtual machine, but with considerable improvements in server efficiency. That efficiency allows a much greater density of containers to be co-located on the same host.

LIGHTWEIGHT, INDEPENDENT	PROCESS AND SYSTEM SEPARATION	BETTER SERVER EFFICIENCY	CO-LOCATING

While container technology has been part of Unix-like operating systems since the turn of the century, it was only with the advent of Docker that containers really came into the mainstream.

Docker has succeeded by bringing both standardization to container runtimes, for example, through the Open Container Initiative, and by creating a complete container management system around the raw technology, simplifying the process of creating and deploying containers for end users. Docker, however, can only be used to execute a container on a single host machine. That's where Kubernetes stepped in.

Docker	vs	**Kubernetes**
Single host machine		Cluster of machines



NetApp

1.1 WHAT MAKES KUBERNETES SO IMPORTANT

KUBERNETES TAKES CONTAINER DEPLOYMENT TO A WHOLE NEW LEVEL BY PROVIDING A ROBUST SOLUTION FOR MANAGING CONTAINERS, AND CONTAINERIZED WORKLOADS ACROSS A CLUSTER OF MACHINES.

Kubernetes makes it possible to execute multiple instances of a container across a number of machines and achieve fault tolerance and horizontal scale-out at the same time. Kubernetes was created by Google after over a decade of using container orchestration internally to operate their public services. Google had been using containers for a long time and developed their own proprietary solutions for data center container deployment and scaling. Kubernetes builds on those solutions as an open source project, enabling the world-wide community of software developers to grow the platform.

And Kubernetes implementation is growing. A recent bi-annual survey of over 2000 IT professionals from North America and Europe by the Cloud Native Computing Foundation found that 75% of respondents were using containers in production today, with the remaining number planning to use them in the future. Kubernetes usage has remained very strong with 83%, up from 77%, using the platform and 58% using it in production.

NetApp

AT THE HEART OF KUBERNETES IS CONTAINERIZATION.

The ability to manage applications independently of infrastructure holds great value for cloud deployments. We can build out a cluster of machines in the cloud that provides the compute and storage resources for all of our applications, and then let Kubernetes ensure we get the best resource utilization. Kubernetes can also be configured to automatically scale the cluster up and down in response to changes in demand.

For deployed applications, Kubernetes offers many benefits, such as service discovery, load balancing, rolling updates, and much more. Kubernetes acts as an application server that is used to run all of the services, message queues, batch processes, database systems, caching services, etc. that make up an enterprise application deployment.

The flexibility of this service has driven Kubernetes introduction and adaptation across the cloud, with all major cloud vendors offering a native Kubernetes service, for example Amazon EKS, Google Kubernetes Engine, and Azure Kubernetes Service. Kubernetes is also the foundation of other container orchestration platforms, such as Red Hat OpenShift.

NetApp

IMPORTANT KUBERNETES TERMINOLOGY

In this section, we'll give you an introduction to Kubernetes terminology that describes the main moving parts that make up the service.

1 Cluster

The collective set of compute nodes and storage resources that form a Kubernetes environment. Each cluster has at least one master, which is responsible for overall management of the cluster, and a number of nodes on which containers will be scheduled to execute. Each node must have a container runtime installed, which is usually Docker, but may be an alternative, such as rkt.

2 Pod

In the Kubernetes architecture, a set of containers may be deployed and scaled together. This is achieved by using pods, which are the minimum unit of deployment in a Kubernetes cluster, and allow more than one container to share the same resources, such as IP address, file systems, etc.

3 Deployment

A deployment is used to control Kubernetes pod creation, updates, and scaling within the cluster, and is normally used for stateless applications. A stateless application does not depend on maintaining its own client session information, allowing any instance of the application to be equally capable of serving client requests.

4 Stateful Sets

For certain types of applications, such as database systems, it is crucial to maintain the relationship between pods and data storage volumes. Stateful sets provide an alternative model to Kubernetes Deployments, and give each pod a unique and durable identity.

5 Stateless Applications

Stateless applications do not keep a private record of client session information, which allows any running instance of the same application to process incoming requests. Applications deployed to Kubernetes or to containers in general are typically stateless, and so are easier to scale out horizontally across the cluster.

6 Services

When multiple, interchangeable pod replicas are active at the same time, clients need a simple way to find any active pod they can send requests to. Services solve this problem by acting as gateway to a set of pods, which may even exist in different Kubernetes cluster.

NetApp

The following terms relate to storage provisioning in a Kubernetes cluster:

7 Volume

A storage provisioned directly to a pod. Kubernetes supports a wide variety of volume types, including Amazon EBS, Azure Disk Storage, Google Persistent Disk, NFS, and many more. Volumes enable the containers within a pod to share information and are destroyed when their parent pod is deleted.

8 Persistent Volume

A volume that exists independently of any specific pod and with its own lifetime. Persistent volumes can be used to support stateful applications, such as database services, enabling all components of an enterprise solution to be deployed and managed by Kubernetes. Another major advantage of using persistent volumes is that they insulate the developers creating pods from the lower-level implementation details of the storage they are accessing. This allows a persistent volume to be provisioned using a local disk, for example, on a developer workstation, and using Cloud Volumes ONTAP in production.

9 Persistent Volume Claim

A claim for a persistent volume, which acts as the link between a pod and a persistent volume. A persistent volume claim is used by Kubernetes to search for suitable persistent volumes that can fulfil the request. This search is based on the size of the required storage, the access mode, a selector definition used to match against labels on the persistent volume, and, optionally, a storage class name.

10 Storage Class

Storage classes add a further level of abstraction to storage provisioning by allowing persistent volume claims to only specify the type of storage they require. For example, slow, fast, and shared may all be valid user-defined storage classes. Storage class also encapsulates the details of the provisioner to be used, the type of volume to create, as well as other provisioner-specific settings. This gives much greater control to the cluster over how the storage is provisioned, which is why storage classes are generally used with dynamic storage provisioning.

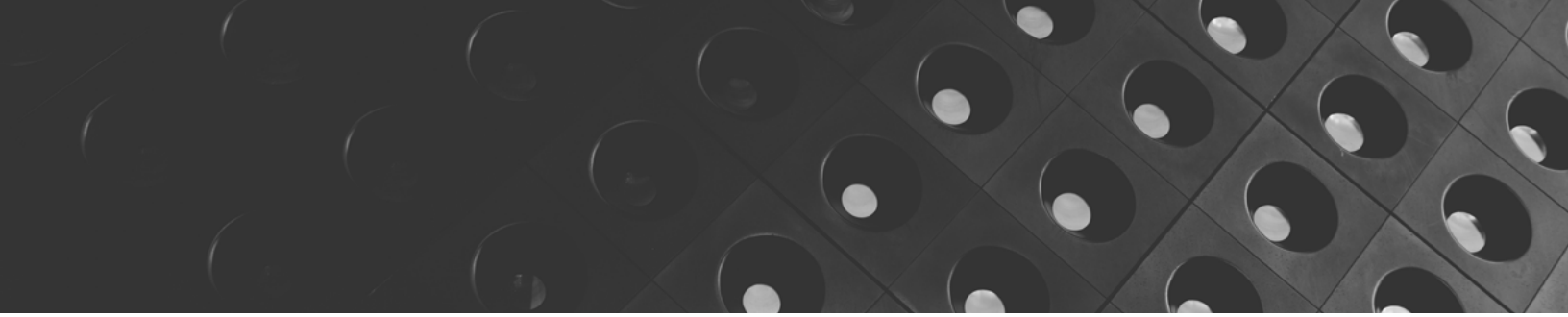
11 Dynamic Storage Provisioning

When Kubernetes administrators are required to manually set up persistent volumes ahead of time, this is known as static provisioning. By contrast, dynamic provisioning is used to automatically allocate persistent volumes based on the persistent volume claims that are received by the cluster. The type of storage to allocate is determined by the storage class specified in the claim.

12 Provisioner

When using dynamic storage provisioning, each storage class will stipulate the provisioner that should be used to create new persistent volumes. Internal provisioners are provided by Kubernetes for a wide range of storage options, such as Amazon EBS, however, it's also possible to specify external provisioners, such as NetApp Trident.

NetApp



1.2 PROVISIONING PERSISTENT VOLUMES FOR KUBERNETES

PERSISTENT VOLUMES PROVIDE STORAGE RESOURCES TO PODS IN THE SAME WAY THAT NODES PROVIDE COMPUTE.

Storage is required by containers and pods for a variety of purposes, from caching data to building stateful applications, such as database services. Kubernetes uses persistent volumes to create a separation between the application developers that simply require storage for their pods and the lower level details of how that storage is provisioned. This layer of abstraction allows the storage implementation to be controlled independently of requesting applications. That means that, for example, storage can be provisioned using local storage in a development cluster and with a more robust solution in production.

Each persistent volume is created by a provisioner that uses a plugin to interface with different types of backend storage, with support for Amazon EBS, Google Persistent Disk, and Azure Disk Storage among many others. The lifetime of a persistent volume is determined by its reclaim policy, which controls the action the cluster will take when a pod releases its ownership of the storage.

Volume: Storage provisioned directly to a pod that enable the containers within a pod to share information. Destroyed when their parent pod is deleted.

Persistent Volume: A volume that exists independently of any specific pod and with its own lifetime. Can be used to support stateful applications, such as database services, enabling all components of an enterprise solution to be deployed and managed by Kubernetes.

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv0003
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: "/mnt/data"
    server: 172.17.0.02
```

A Kubernetes manifest for a persistent volume

NetApp

KUBERNETES USES PERSISTENT VOLUME CLAIMS TO PROVIDE CLUSTERS WITH STORAGE.

This is how it works: Each pod uses a persistent volume claim to bind to storage from the cluster of a particular size, access mode, and volume mode. The claim can also use a selector to only match volumes with a specific set of labels. On receiving a claim, the cluster will search for an existing persistent volume with which to fulfill the request.

Persistent Volume Claim:
A claim for a persistent volume, which acts as the link between a pod and a persistent volume. Used by Kubernetes to search for suitable persistent volumes that can fulfill the request.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: thepub
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: bronze
```

A persistent volume claim that uses dynamic provisioning

NetApp

1.3 STATIC PROVISIONING VS. DYNAMIC PROVISIONING

THERE ARE TWO CHOICES FOR HOW PERSISTENT VOLUMES ARE PROVISIONED BY KUBERNETES: STATIC AND DYNAMIC.

Cluster administrators can pre-allocate persistent volumes for the cluster, known as static provisioning, however, this requires prior knowledge of storage requirements as a whole. Dynamic volume provisioning is an alternative model for managing storage provisioning in Kubernetes, and is used to automatically deploy persistent volumes based on the claims received by the cluster.

Static Provisioning

- Set amount of storage before use
- Controlled by administrators
- Pro: Storage spend is fixed
- Con: Can't scale storage if needed

VS

Dynamic Provisioning

- Created more organically
- Automatic creation process
- Pro: Able to provision volumes as needed
- Con: No set limit on storage spend

NetApp

USING DYNAMIC PROVISIONING IN A KUBERNETES CLUSTER REDUCES A LOT OF THE ADMINISTRATIVE OVERHEAD INVOLVED IN MANUALLY CREATING PERSISTENT VOLUMES.

Automatically allocating and deallocating persistent volumes in response to persistent volume claims can also help to reduce wasted storage that is allocated but never used. Persistent volume claims that are fulfilled dynamically make use of a storage class attribute to specify the type of storage required, which allows for data to be tiered by I/O performance profile.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: ontap-gold
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
```

An example Trident storage class definition

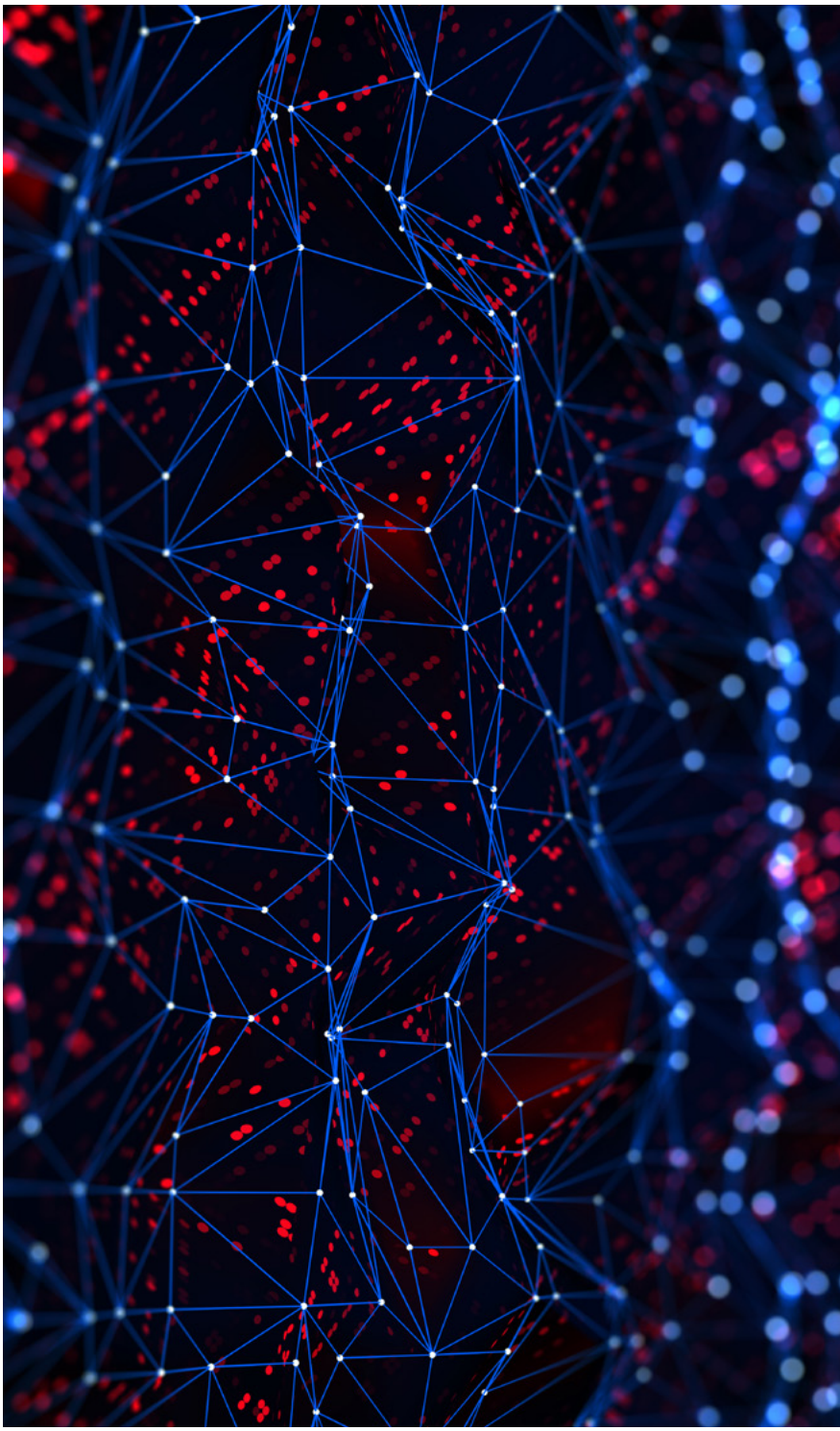
Storage Class:

Storage classes add a further level of abstraction to storage provisioning by allowing persistent volume claims to only specify the type of storage they require. Encapsulate the details of the provisioner to be used, the type of volume to create, etc. Generally used with dynamic storage provisioning.

NetApp

Kubernetes comes with a variety of internal provisioners that dynamically allocate storage using Amazon EBS, Google Persistent Disk, Azure Disk Storage, and many other storage platforms. However, that's where the support for storage ends: storage management features, such as creating and restoring backups, ensuring high availability, and provisioning test copies of persistent volumes are all out of scope for Kubernetes and must be catered for by cluster administrators. In a hybrid or multicloud environment, this can lead to a multitude of different solutions that will increase the complexity of managing storage across the enterprise.

What can you do to avoid that? In the next section we'll discuss NetApp's solution: NetApp Trident and Cloud Volumes ONTAP.



NetApp

1.4 MANAGING STATEFUL APPLICATIONS IN KUBERNETES

Stateful applications, such as database services and message brokers, record and manage the information generated within an enterprise platform. Though Kubernetes has always supported stateless applications—which are horizontally scalable due to the interchangeability of each pod—stateful applications require stronger guarantees for the storage they use.

Why the need for extra guarantees when it comes to stateful vs stateless? Whereas the storage used by stateless containerized applications can simply be re-initialized when a pod is rescheduled to different node in the cluster, stateful applications are recording business-critical information that must be preserved at all costs—that requires persistent storage with an independent lifetime.

Stateful Sets: For certain types of applications, such as database systems, it is crucial to maintain the relationship between pods and data storage volumes. Stateful sets provide an alternative model to Kubernetes deployments, and give each pod a unique and durable identity.

Stateless Applications: Stateless applications do not keep a private record of client session information, which allows any running instance of the same application to process incoming requests. Applications deployed to Kubernetes or to containers in general are typically stateless, and so are easier to scale out horizontally across the cluster.

NetApp

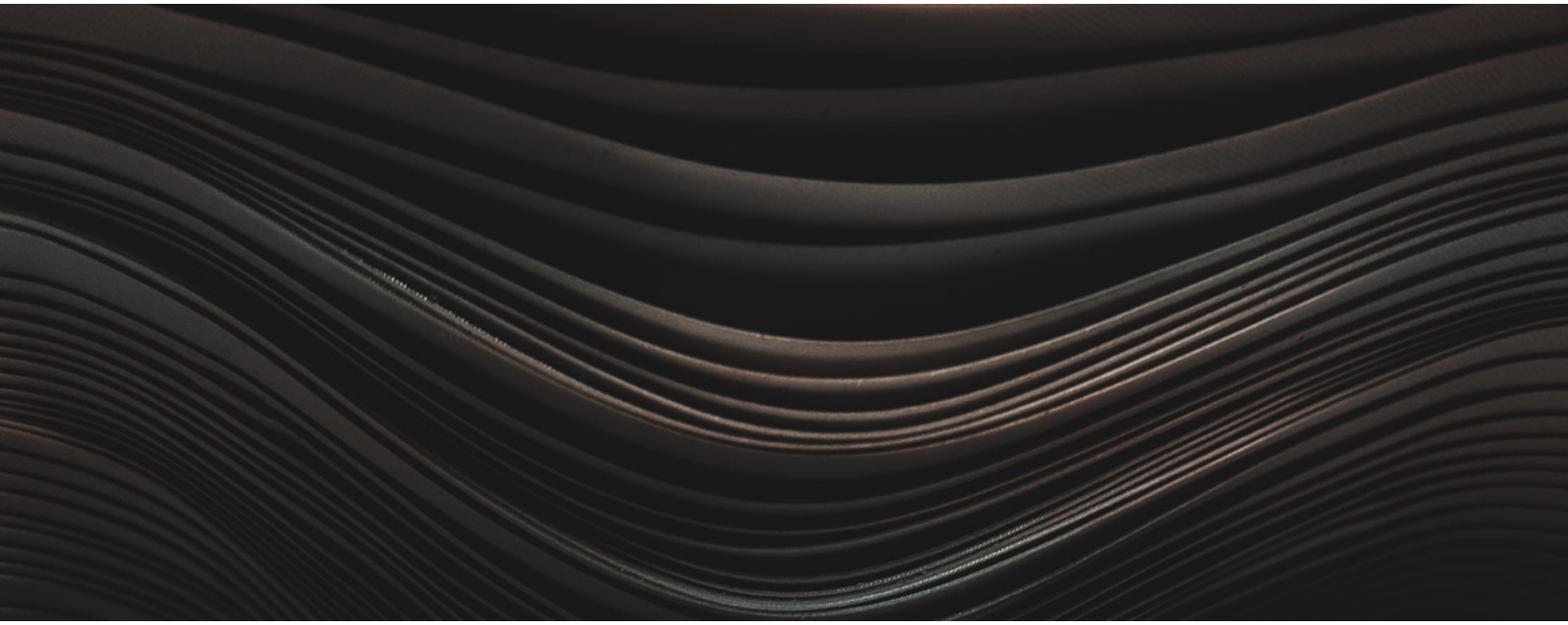
Kubernetes persistent volumes are used to create a layer of abstraction between pods and their provisioned storage, allowing each to be managed separately. Pods use a persistent volume claim as a request for storage that is matched to a persistent volume by the Kubernetes cluster.

Using a reclaim policy of "retain", a persistent volume will not be deleted or cleared down after the persistent volume claim is released. Because of this, in the event of a failure or the rescheduling of the pod onto a different node, the relationship between pod and storage can be re-established.

Take a look at this example use case of deploying MySQL with WordPress where persistent volumes were used in this way to deploy a database server.

In both static and dynamic provisioning scenarios, stateful applications have a very real need for reliable storage that is resilient against failure, and data protection features, such as backup and restore. In and of itself, these facilities are not provided by Kubernetes, but are instead delegated to the provisioner used and the storage backend. Users have to find outside solutions, such as Trident and Cloud Volumes ONTAP.

Trident provisions new persistent container storage that benefit from all of NetApp's data management capabilities. Using stateful sets simplifies the deployment of all stateful applications by automating much of the required provisioning and administrative activity. If a pod in a stateful set goes down, Kubernetes will automatically bind a new instantiation of the pod to the dynamically provisioned persistent volume it was previously using.



NetApp

PART TWO

NetApp Solutions and Benefits for Kubernetes

NETAPP TRIDENT AND CLOUD VOLUMES ONTAP WORK TOGETHER TO DYNAMICALLY PROVISION VOLUMES FOR KUBERNETES.

NetApp Trident is a dynamic storage provisioner for Kubernetes that fulfils persistent volume claims using storage managed by Cloud Volumes ONTAP leveraging Azure, AWS, or on-premises ONTAP appliances as the storage back end for Kubernetes persistent volumes. This enables Kubernetes clusters to take advantage of the power of NetApp storage management for persistent volume provisioning in on-premises, hybrid cloud, or multicloud environments.

Cloud Volumes ONTAP is deployable to either AWS or Azure and delivers:

1 High Availability

Cloud Volumes ONTAP HA provides storage failover capability that works across Availability Zones, and guarantees high availability, RPO=0, and RTO < 60 seconds.

2 Data Protection

NetApp Snapshots™ are used to create instant, space efficient, backups of a storage volume of any size that can be instantly restored. NetApp SnapMirror® technology uses snapshots as the basis for incrementally replicating a volume to another instance of Cloud Volumes ONTAP, which may reside in another region as a DR or backup secondary copy.

3 File Services

A flexible, scalable, performant, and secure solution for deploying file shares and file services with multiprotocol access for SMB / CIFS and NFS access for both Windows and Linux-based file workloads.

NetApp

4 Storage efficiencies

Volumes created using Cloud Volumes ONTAP are able to use built-in technologies such as thin provisioning, data deduplication, and data compression making it possible to reduce cloud storage footprint and operational costs by 50-70%.

5 Data Tiering

Cloud Volumes ONTAP can transparently tier cold data to Amazon S3 or Azure Blob to save on storage costs.

6 Storage cloning

Using NetApp FlexClone® technology, existing storage volumes can be instantly and space-efficiently cloned and made available for read/write use. Trident integrates with cloning to allow persistent volume claims to use a clone, rather than always provisioning new storage.

NetApp

2.1 DATA PROTECTION FOR PERSISTENT DATA STORAGE IN KUBERNETES APPLICATIONS

Enterprise workloads typically have a strong requirement for reliable data storage. Kubernetes persistent volumes can be provisioned using a variety of solutions. However, ensuring that the data is easy to backup and restore, always available, consistent, and durable in a Kubernetes workload DR (Disaster Recovery) situation or any other failure is the responsibility of end users and administrators.

Stateful applications in production environments, such as database services, require access to redundant and highly available data storage. Most stateless applications make use of stateful services in order to fulfill client requests, and therefore have an indirect dependency on robust data storage services as well. Kubernetes provides a lot of flexibility when it comes to persistent data storage provisioning, however, each solution uses its own specific mechanisms for protecting data, which may also have limitations.

A SOLUTION FOR PERSISTENT DATA NEEDS

Kubernetes caters for persistent data storage through persistent volumes, which have a life-cycle that is independent of any particular container and that can be provisioned using a diverse range of storage platforms.

How do different storage solutions protect that data differently? For example, persistent volumes can be provisioned using Amazon EBS, which provides some level of data redundancy within an Availability Zone; however, this is not sufficient protection for all organizations where end users are expected to build their own solutions to protect data across Availability Zones, or across regions. When it comes to Kubernetes workload DR requirements, investing in this type of data protection is not only mandatory for business continuity and regulatory requirements, it also pays huge dividends in the long run.

Another important requirement for protecting persistent data storage is the ability to create and restore backups. Examples of why you'd take regular backups include ensuring that previous versions of the data are available in case of user error and providing your deployment in Kubernetes security against malicious access, such as ransomware attacks. Due to the large size of production datasets, an efficient procedure is required not only to create backups, but also to restore them consistently.

NetApp

YOU CAN GET BETTER DATA PROTECTION USING CLOUD VOLUMES ONTAP.

Cloud Volumes ONTAP makes use of a number of features to better protect your data in Kubernetes

ONTAP Snapshots

can be used to instantly create space-efficient backups for Kubernetes storage of any size. Using NetApp SnapCenter®, application-aware snapshots can be created by temporarily freezing I/O write operations in order to guarantee that the data is in a consistent state before a snapshot is taken. Snapshots can also be instantly restored back to the original source volume or to a new volume. Users can even access the snapshot directly as a read-only view of the source data at the point in time the snapshot was created.

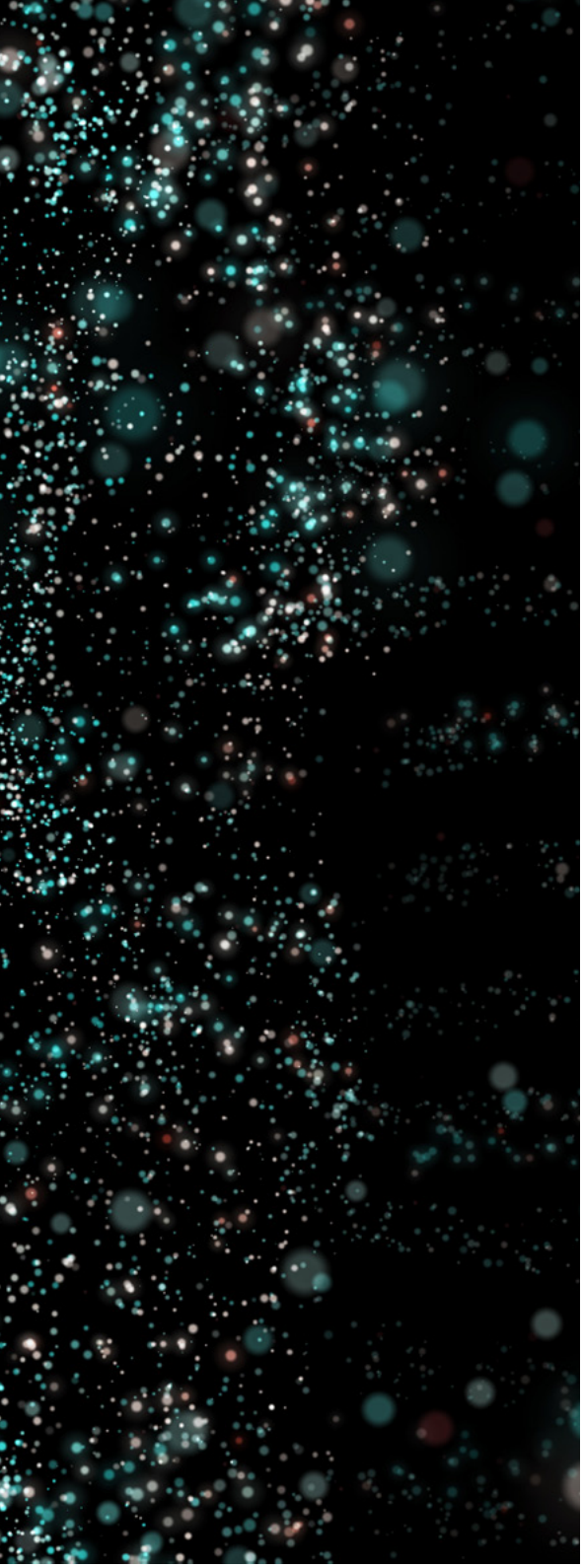
Cloud Volumes ONTAP HA

is a high availability solution that automatically mirrors storage volumes to a secondary instance of Cloud Volumes ONTAP, which may be placed in a different Availability Zone or Availability Set. These two instances can be deployed in either an active-active or active-passive configuration, with the ability to failover and failback between the nodes without affecting client applications that are actively using the storage. Should a planned or unplanned failover be required, Cloud Volumes ONTAP HA provides an RPO=0, i.e. zero data loss, and an RTO of less than 60 seconds.

Data replication

of a Kubernetes workload DR site in another region can be easily accomplished using NetApp SnapMirror®, which provides efficient, block-level data replication between ONTAP storage environments. After creating an initial baseline copy, all further synchronization occurs on an incremental basis, copying over only the data that has changed. Cloud Volumes ONTAP can also be used to failover storage to the destination volume, and efficiently re-synchronize in the reverse direction in order to failback. As well as mirroring storage for DR purposes, NetApp SnapMirror can also be used to create a repository for long-term backups and data archiving.

NetApp



2.2 STORAGE EFFICIENCY FOR IMPROVING PERSISTENT VOLUME STORAGE COSTS

In both static and dynamic provisioning, it can be a challenge to ensure storage efficiency, as application developers must prepare for storage usage peaks and can often overestimate the amount of storage they need. In a cloud environment, this leads to an unnecessary increase in cloud storage costs.

Many times, large allocations of storage made for an application are never used, and in other cases the data stored within a persistent volume is never compressed, leading to extra storage space being used up unnecessarily. These types of issues can be resolved using the storage efficiency technologies that are part of Cloud Volumes ONTAP, which the NetApp Trident provisioner makes available to Kubernetes.

There may also be situations where a large part of the production dataset is cold, or infrequently accessed, but which cannot be moved to more cost-effective storage without introducing a lot of complexity to the ways application services rely on that data. These application services require a uniform view of their data, with fast access to hot data and on-demand access to cold data. This can again be a difficult problem to solve, resulting in large allocations of high performance and costly storage.

Another scenario where transparently-applied storage efficiency would hold great value is where the data being stored contains a high degree of redundancy, which could be compressed in order to reduce the overall amount of storage required. Moving the onus for compression to the storage layer simplifies application software development and also helps to reduce the cloud storage footprint of legacy and third-party application services, where changes to the systems cannot be made.

Though storage efficiency is always important in the cloud, it can be even more so in Kubernetes environments due to the inherent scalability of containers. Spinning up new pods to deal with an increase in workload, or to provide greater redundancy and availability, also requires allocating new persistent volumes. The storage overhead for these persistent volumes can be brought under control through efficient data storage.

NetApp

YOU CAN REDUCE KUBERNETES STORAGE COSTS WITH CLOUD VOLUMES ONTAP STORAGE EFFICIENCIES.

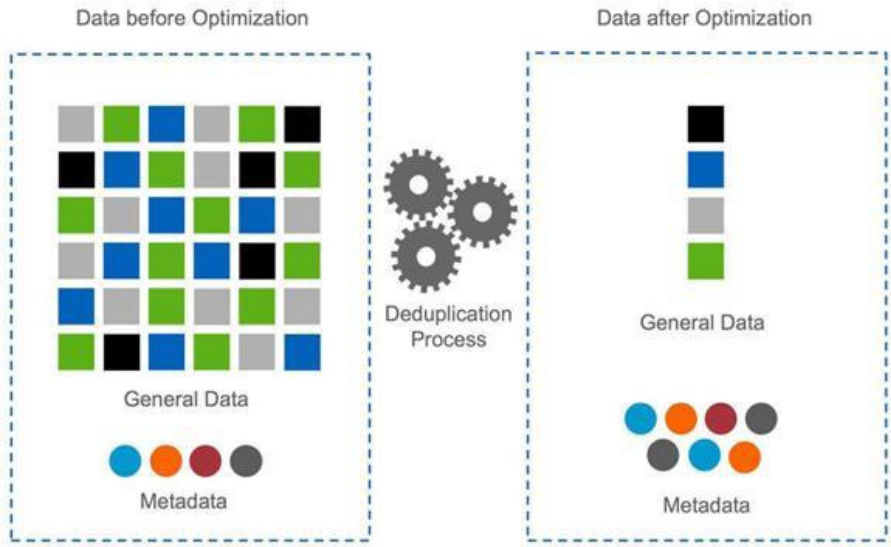
Trident benefits Kubernetes by making it possible to take advantage of NetApp's enterprise-grade storage efficiencies of Cloud Volumes ONTAP, providing Kubernetes advantages in terms of transparently reducing the storage space required for persistent volumes.

Thin Provisioning

Thin provisioning makes it possible to create persistent volumes that appear to pods as having the size they requested through a persistent volume claim, but without needing to allocate all of that storage in advance. Cloud Volumes ONTAP will automatically add storage capacity to the persistent volume as and when it is required, and also return back freed up storage to the common pool when data is deleted. This ensures that storage space is only allocated when it's actually needed, which reduce storage usage costs and drives up storage space utilization. Thin provisioning also makes it much easier to plan for the future storage requirements of the cluster.

Data Deduplication

Cloud Volumes ONTAP is able to transparently apply transformations, which help to reduce storage space usage, to the data it stores. Data deduplication collapses identical copies of a block into a single block, with reference pointers inserted into every place the block is used. This can dramatically reduce storage space requirements, with some customers reporting savings of up to 70%. A small amount of storage space is consumed in order to maintain the metadata required to support the block mappings.

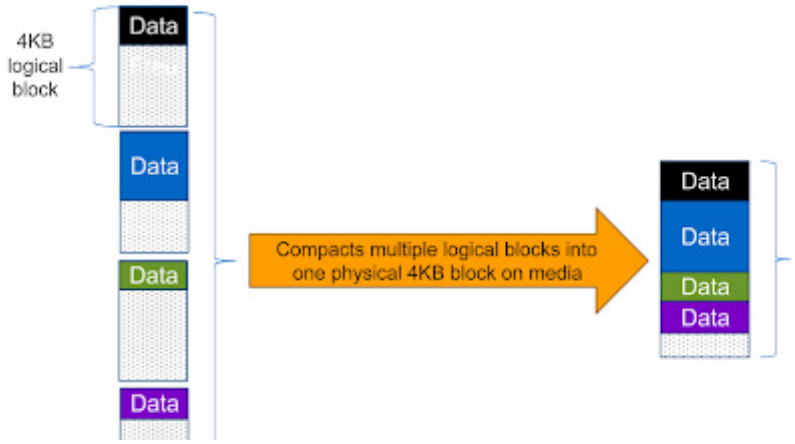


Data Compression

Data in an ONTAP storage volume can be transparently compressed without requiring any changes to client applications and services. This compression is applied to groups of consecutive blocks, as opposed to entire files, which makes reading and updating highly optimal. Data compression can also be used in conjunction with data deduplication.

Data Compaction

NetApp is constantly innovating and extending the capabilities of the ONTAP platform, and data compaction is one of the more recent features to be introduced. After applying inline data deduplication and compression, multiple blocks that are not completely filled are combined together, removing the unused of space that would have otherwise been left in each block.



Storage Tiering

A major advantage of using Cloud Volumes ONTAP is the ability to automatically balance data between a capacity storage tier for colder data and a performance tier for fast access. Depending on the cloud vendor being used, Cloud Volumes ONTAP will use Amazon S3 or Azure Blob for the capacity tier, which provides significant cost savings for large amounts of data that are infrequently accessed, but that must still be available on-demand. When the data is required, it is transparently moved to the performance tier, and will age back out to the capacity tier when it is no longer in active use.

NetApp

2.3 NFS FILE SHARES AND KUBERNETES

The NFS (Network File System) protocol is widely used in sharing files for enterprise environments, allowing many users to access the same files at the same time. This makes sharing data a lot easier. NFS can be used for a wide range of use cases, such as for creating data lakes for building analytics solutions, database services, data archives, etc. If you're using Kubernetes, NFS can be used with Kubernetes pods to manage persistent storage requirements and share data and files between containers and other pods.

There are several reasons why NFS shared storage is preferable for Kubernetes deployments. For starters, there's an advantage to NFS over iSCSI because managing a large number of individual iSCSI storage allocations can add to administrative overhead. All those block-level persistent volume allocations can make storage management more difficult, especially when Kubernetes persistent volumes are statically allocated. In these cases, storage utilization may be lower, as a persistent volume is used only for a single pod and that pod may not make use of all of its storage. Also, sharing data between pods can be difficult when using block-level persistent volumes.

Unless containers are deployed to the same pod, sharing data between containers can also be difficult. Deploying containers in this way, i.e. to the same pod, should only be done when it makes sense for the application, such as when the containers work together directly to fulfill some function. Putting containers into the same pod when they simply need to share data can lead to scalability problems as a pod will only be scheduled to a single node.

NFS solves these problems by allowing many hosts to mount the same file system at the same time, and for all hosts to access files concurrently. With NFS, users don't need to format the storage volume using an Operating System file system, such as ext4; the storage can simply be mounted and used straight away. This makes it much easier to attach storage to pods and reduce the administrative overhead of working with persistent storage. NFS storage volumes in Kubernetes can also be easily expanded without any client-side changes using Trident, which we'll discuss in more detail in the next section.

NetApp

2.4 CLONING KUBERNETES PERSISTENT VOLUMES WITH FLEXCLONE

Creating test copies of production data is very often a requirement for DevOps engineers implementing CI/CD pipelines or setting up staging clusters for pre-production testing. Automated software test suites will often mutate the data that they access, which means that a fresh copy of the data is required in order to repeat the testing, with test cycles sometimes executed hundreds of times. Ensuring faster TTM (Time To Market) and the delivery of high quality software relies upon developers working in parallel and executing tests as often as required.

Restoring data backups is not a scalable solution for creating test data sets: with the size of the source data involved it just takes too much time to perform a restore. This leads to reduced developer productivity when multiple, up-to-date copies of persistent volumes are required, which also have to be refreshed frequently. Using this approach also means that the amount of storage used to develop and support an application will be many times greater than the storage requirements of the production environment, causing a significant increase in cloud storage costs. That's not something any developer wants to get blamed for.

The time taken to perform a restore and the storage space it uses can both be very wasteful, as usually most of the restored data remains unaffected by the software testing and must be recreated simply to bring the data to a consistent point or to get an up-to-date copy. In contrast, not only is NetApp FlexClone able to instantly clone a source volume of any size, it also does so with zero storage penalty for any storage size required. NetApp Trident helps you take advantage of this for persistent volumes in a Kubernetes cluster.

In the third part of this ebook we'll walk you through the steps of cloning your persistent volumes in order to facilitate test environments and reduce your storage footprints.

NetApp

PART THREE

How to Use Trident and Cloud Volumes ONTAP with Kubernetes

In this section we'll see a few coding examples for how users can make Trident and Cloud Volumes ONTAP work with their deployments on Kubernetes.

3.1 USING CLOUD MANAGER FOR KUBERNETES DEPLOYMENT WITH NETAPP TRIDENT

Cloud Manager is the central platform from which you can deploy and manage instances of Cloud Volumes ONTAP for both large and small environments. The graphical, web-based user interface makes it easy to setup Cloud Volumes ONTAP storage services and organize them across multiple tenants for better overall manageability. On-premises systems, Azure deployments, and AWS deployments can all be controlled from a single dashboard, and NetApp SnapMirror replication relationships created between them with a simple drag-and-drop.

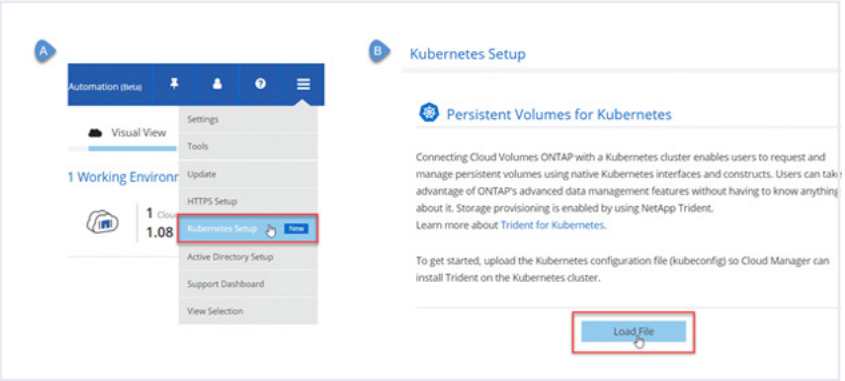
Cloud Manager allows us to deploy NetApp Trident to a Kubernetes cluster and then relate the cluster to Cloud Volumes ONTAP instances.

PREREQUISITES

Before you can start you should make sure that you have network access between Cloud Manager, the instances of Cloud Volumes ONTAP to be used, and the actual Kubernetes cluster. Cloud Manager will also require internet access in order to download the latest deployment packages for NetApp Trident.

DEPLOYING TRIDENT

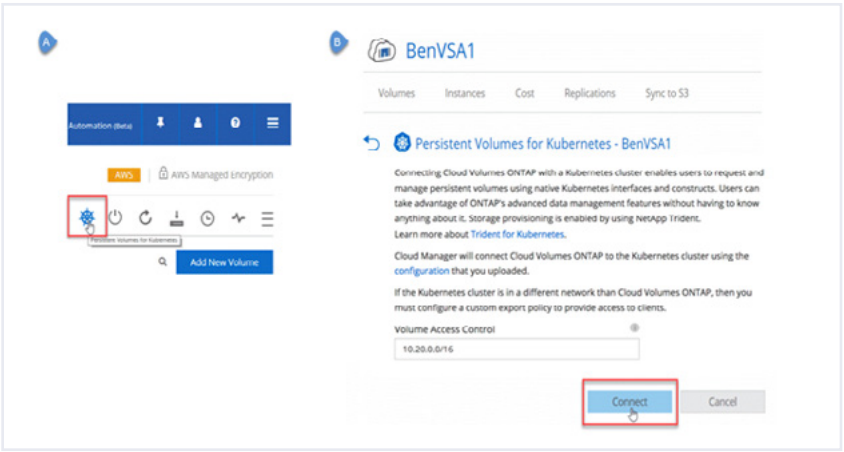
To deploy NetApp Trident, Cloud Manager must be provided with the kubeconfig file for the target Kubernetes cluster, which provides all the necessary details for making a connection and performing cluster operations. After uploading the file, Cloud Manager takes care of the rest of the deployment.



Setting up access to a Kubernetes cluster from Cloud Manager.

ASSOCIATING THE KUBERNETES CLUSTER WITH CLOUD VOLUMES ONTAP

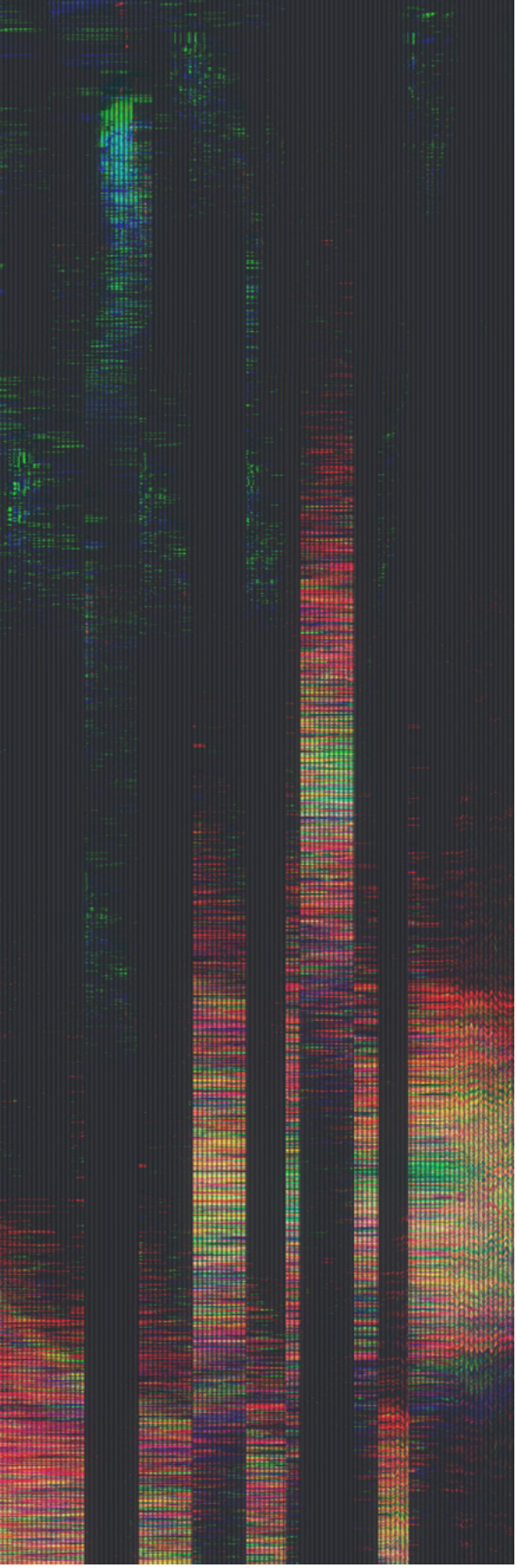
Now that we have NetApp Trident installed, our Kubernetes cluster is able to interface with Cloud Volumes ONTAP, however, we still need to associate the cluster with the specific instance of Cloud Volumes ONTAP that it should use. This is achieved by clicking the Kubernetes icon on the details page for that instance, and confirming that we wish to proceed. Cloud Manager will then complete the configuration process and allow us to start provisioning persistent storage for Kubernetes.



Associating a deployment of Cloud Volumes ONTAP with the Kubernetes cluster

When integrating with NFS Kubernetes deployments, Trident is also able to create the storage class and the NFS mount points. Other benefits include cloning Kubernetes persistent volumes with FlexClone, using storage efficiencies to cut persistent volume footprint and costs, storage tiering to Amazon S3 or Azure Blob, and data protection.

NetApp



3.2 CLONING PERSISTENT VOLUMES

Data cloning can help speed up TTM and reduce storage footprint and costs. Instead of always provisioning new persistent volumes in response to persistent volume claims, using Trident the user can clone existing volumes. To instruct Trident to create a clone, simply give it the name of the persistent volume claim that was used to provision the original persistent volume, which must also have been allocated through Trident. This information is provided through a metadata annotation on the persistent volume claim that is requesting the volume clone.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basicclone
  annotations:
    trident.netapp.io/cloneFromPVC: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic
```

In the background, Trident will find the ONTAP volume that was used to provision storage for the original persistent volume claim and then use NetApp FlexClone to perform a data cloning operation. Clones are created by using a snapshot on the parent volume. Both the clone and parent volumes are free to accept changes to their data, which are written to new blocks using a redirect-on-write mechanism that protects blocks locked by a snapshot from being overwritten. Using this technique, the clone requires negligible storage space when it is first created, and only grows with the changes that are made to it.

By using the default reclaim policy of delete, the clone will automatically be deleted when the pod using it is destroyed. This can be used to conveniently clean up any allocated storage at the end of a round of testing, for example. Trident also supports an optional annotation to split the clone from the parent volume it is associated with. This turns the clone into an independent copy of the data, which means the space efficiencies discussed previously will be lost, however, this can be more suitable in scenarios when the clone will undergo a lot of changes, or when the state of a volume is needed to provide seed data for a new requirement.

NetApp

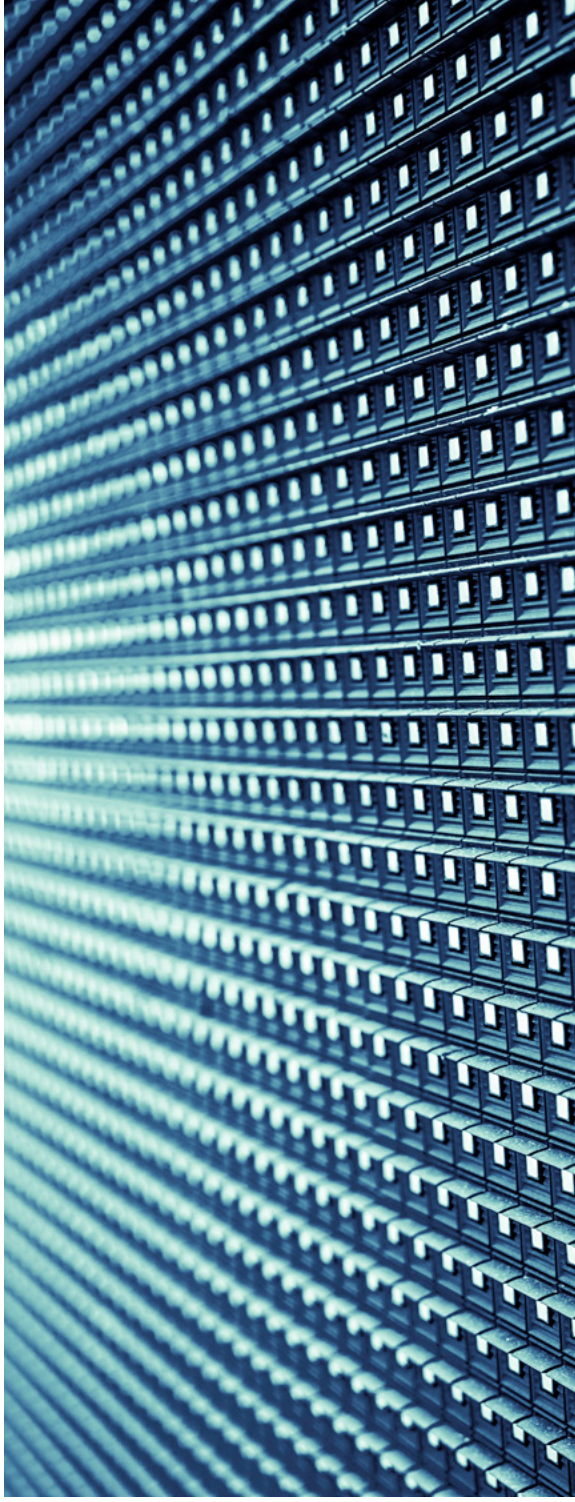
3.3 HOW TO SET UP MYSQL KUBERNETES DEPLOYMENTS WITH CLOUD VOLUMES ONTAP

MySQL is a popular flavor of SQL database. The following provides an example of deploying a single Kubernetes MySQL instance using NetApp Trident.

First, we will create a persistent volume claim for the storage we need. As Trident uses dynamic provisioning, we will specify a storage class, which must have been setup prior to executing this manifest. Each storage class defines the provisioner to be used, along with any other provisioner-specific settings that will determine how the storage is provisioned.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mysql-pvc
  annotations:
    trident.netapp.io/reclaimPolicy: "Retain"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: high-performance
```

When this claim is used by a pod, Trident will automatically create a 100 GiB high-performance storage volume in Cloud Volumes ONTAP to fulfil the request. This storage class may have been implemented to make use of Amazon EBS Provisioned IOPS disks, for example. A reclaim policy of "retain" has also been specified, which will prevent the persistent volume from being deleted if the pod releases the claim.



NetApp

We can now bind a pod to this storage by referencing it in our MySQL deployment manifest. As shown below, we simply need to reference our previously defined persistent volume claim and specify where the bound persistent volume should be mounted within our MySQL Kubernetes container.

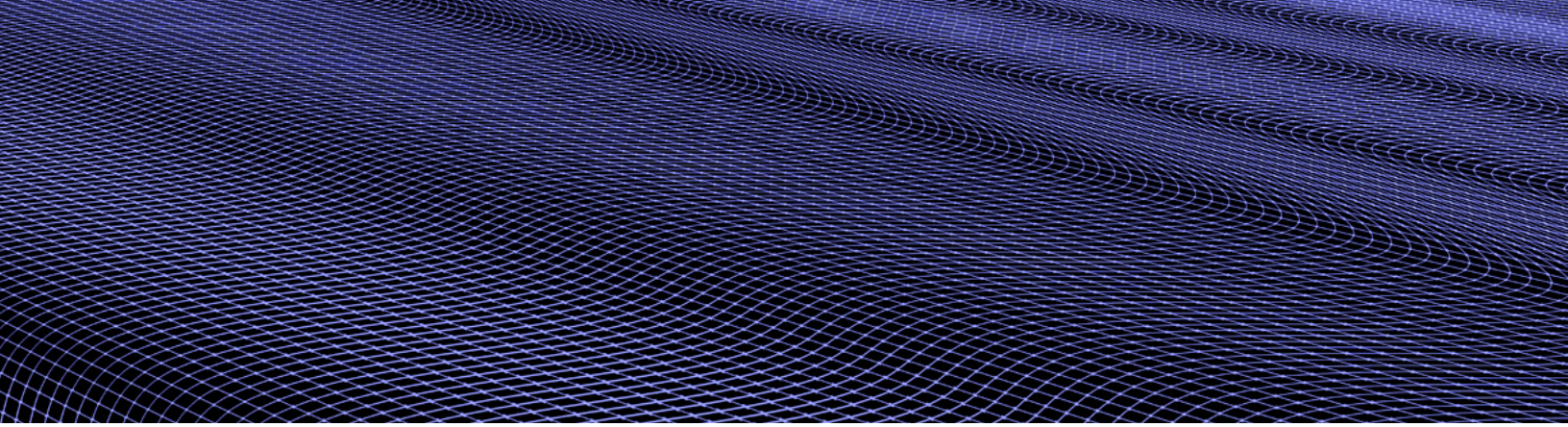
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-server
  labels:
    app: mysql
spec:
  selector:
    matchLabels:
      app: mysql
      tier: database
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
        tier: database
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-pv
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-pv
        persistentVolumeClaim:
          claimName: mysql-pvc
```


NetApp

We can additionally set up test instances of our Kubernetes MySQL database using Trident's integration with NetApp FlexClone. The following persistent volume claim manifest demonstrates how this can be done.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mysql-pvc-clone
  annotations:
    trident.netapp.io/cloneFromPVC: mysql-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: high-performance
```

We simply annotate our claim with the name of the persistent volume claim to clone from, and Trident takes care of the rest by automatically finding the source storage volume in Cloud Volumes ONTAP and performing a NetApp FlexClone operation. The default reclaim policy of delete will ensure that the clone volume is automatically deleted when it is no longer required.



3.4 HOW TO USE NFS FILE SERVICES WITH KUBERNETES

As a Kubernetes NFS provisioner, the benefits of using Trident include mounting persistent volumes as Read/Write Many, dynamically resizing NFS persistent volumes, and creating separate storage classes for different mount parameters and other requirements.

MOUNTING PERSISTENT VOLUMES AS READ/WRITE MANY

Persistent Volumes using NFS can be mounted as Read/Write Many, which allows multiple pods to mount the same volumes. Using native Kubernetes constructs means that storage can be allocated without the need for custom procedures or processes. Using the Read/Write Many access mode allows many persistent volumes claims to access the same persistent volume.

With block-level Kubernetes persistent volumes, Read/Write Once must be used, which means that there is a 1-to-1 relationship between persistent volume claims and persistent volumes.

Here's a persistent volume claim (PVC) for a Kubernetes NFS volume example:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nfs_share
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: silver
```


NetApp

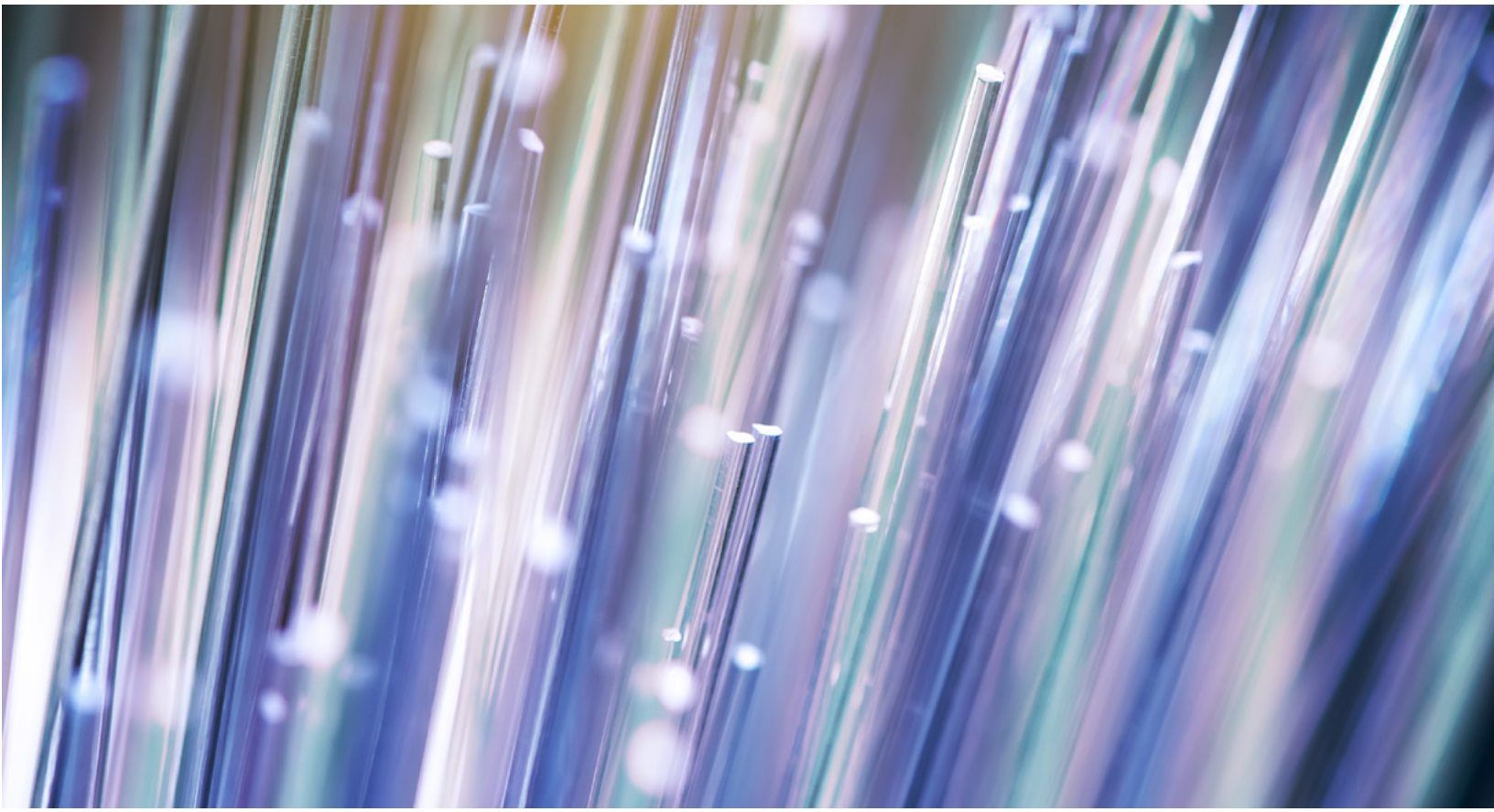
DYNAMICALLY RESIZING NFS PERSISTENT VOLUMES

New to provisioning in Kubernetes is the support for editing persistent volumes claims to request more storage space. With this new capability, persistent volumes can be expanded without needing to re-create the pod that uses the storage. In order for that to happen, the underlying storage provisioner must support resize. The commonly used provisioners, such as those for Amazon EBS or Azure Disk storage support this—except not when it comes to NFS. With Trident, NFS persistent volumes can be dynamically resized and all the back-end changes required taken care of. This makes Trident a major value add.

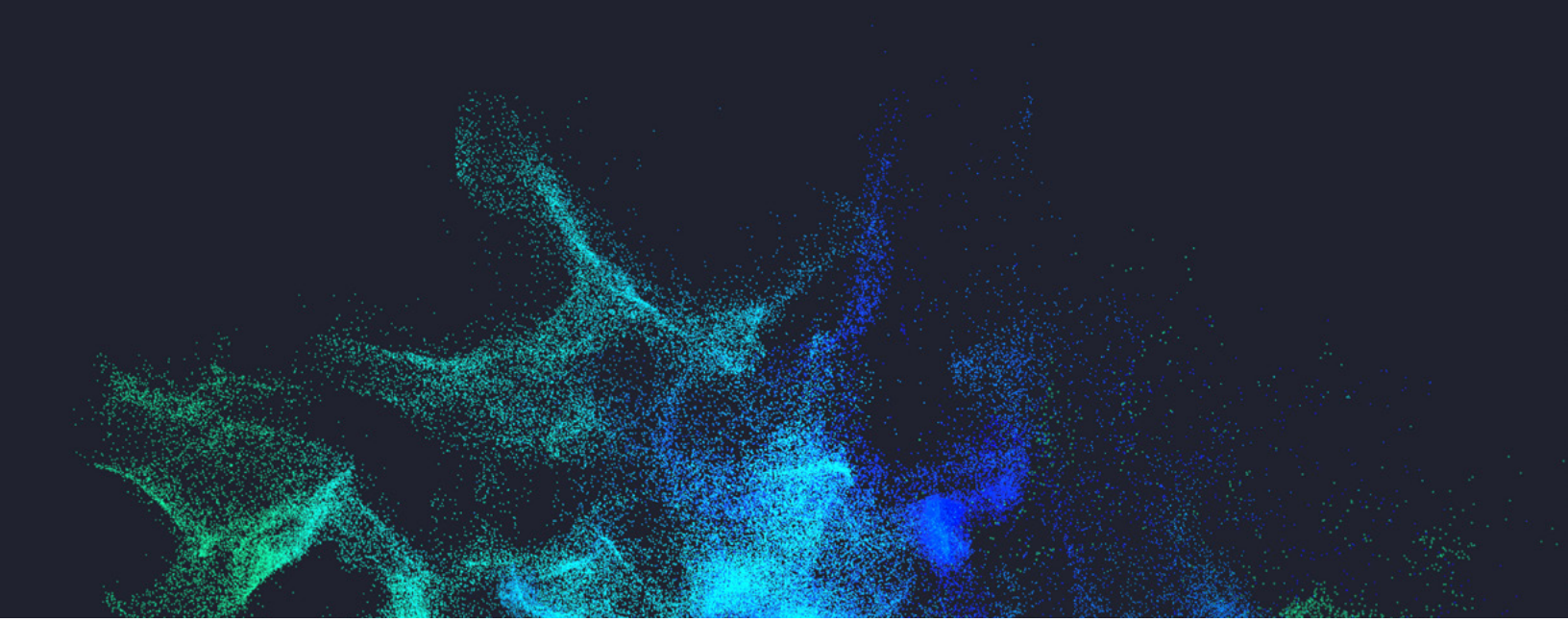
By extending the support for resizing persistent volumes to NFS, Trident is a major value add.

Block-level storage, such as Amazon EBS, needs to have the file system extended when the underlying storage is expanded. Kubernetes will take care of this, but it will also require that the pod using the storage to be restarted. That will cause downtime. With NFS, no file system expansion is required, and the pod can continue to work without interruption.

You can read more about resizing NFS volumes with Trident here.



NetApp



CREATING SEPARATE STORAGE CLASSES

Separate storage classes can be created for different mount parameters and other requirements, for example using separate storage classes for NFSv3 and NFSv4.

Storage classes allow users to control the type and configuration of the storage they are using when provisioning in Kubernetes. That extends to the Kubernetes dynamic provisioning NFS storage Trident provides.

Take this Kubernetes NFS storage class example. Here is the storage class manifest:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: netapp.io/trident
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

The `allowVolumeExpansion` attribute controls whether the resizing of persistent volumes created using this storage class is allowed or not.
The storage class can be used to control the storage pool, or NetApp aggregate used to provision persistent volumes, e.g. a storage class named `gold` may be used to configure high performance back-end storage, where as one called bronze may be configured for capacity storage.

NetApp

Conclusion

Kubernetes is today's most widely-used platform for container and microservices orchestration and provides the scalability and flexibility required for deploying enterprise applications and services. Managing storage in a Kubernetes cluster with dynamic storage provisioning massively reduces the manual administration required for allocating cloud storage to pods and containers.

Using NetApp Trident, Kubernetes storage requests are dynamically fulfilled by Cloud Volumes ONTAP, which similarly does for storage what Kubernetes does for containers.

Now that you've had this Kubernetes guide, find out for yourself how Cloud Volumes ONTAP transforms container storage management with a free 30-day free trial for AWS or for Azure.

NetApp

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

NA-287-0218