***Chapter 5*: Understanding Oracle Cloud Infrastructure Storage Options**

Data in your overall cloud computing system is the most critical part of your application. **Oracle Cloud Infrastructure** (**OCI**) storage options give you multiple options and flexibility to store your digital data in a logical representation of the storage pools. OCI's physical storage options span across multiple backend storage devices and their categories are also very different from their use cases. Whether you want super-performing local data storage within your compute box or you want highly scalable and performant storage, OCI has it all covered. In fact, OCI also offers very low-cost storage for backing up your application data. OCI has vast storage options to choose from for your various needs, providing Block Volume, Local NVMe, **File Storage Service** (**FSS**), Object Storage, and Archive Storage storage options.

OCI's Block Volume service lets you attach a volume to an instance, move that across to a different instance, and also change the performance characteristics of it—for example, changing it to extreme performance. You can choose local NVMe for higher **input/output operations per second** (**IOPS**) performance where data locality is a requirement. OCI's FSS provides durable, scalable, secure, and enterprise-grade network filesystems, whereas you can choose Object Storage for fast, immediate, and frequent access to data as objects, and Archive Storage for rare access to data that is stored there, such as backup data. However, by nature, Archive Storage is set up to prevent data being stored for a long period of time.

By the end of this chapter, you should be able to choose between the Block Volume, Object Storage, File Storage, and Archive Storage options.

In this chapter, we're going to cover the following main topics:

* OCI Block Volume
* OCI File Storage
* OCI Object Storage

**OCI Block Volume**

To understand the service, let's begin by understanding what a block volume is. A **block volume** is a type of block device that is used as data storage. The OCI Block Volume service uses **Internet Small Computer Systems Interface** (**iSCSI**) to deliver features and performance. OCI Block Volume has been carefully designed for the security and durability of data and lets you create block volumes and attach them to your compute instance. The OCI Block Volume service delivers a simple, scalable block volume service that fulfills all your workload performance needs. You can treat this volume as your regular hard drive once you attach it to an instance. The Block Volume service utilizes industry-leading highest performance **Non-Volatile Memory Express** (**NVMe**) drives and is offered over the network using the standard iSCSI protocol.

The Block Volume service serves both the boot volume (for the **operating system** (**OS**) disk) and the block volume (applications data). As the name suggests, it lets you store your application data independently and beyond the lifespan of compute instances. This means that even if you shut down or terminate your instance, your block storage data remains.

OCI Block Volume storage uses a fixed-size block for consistent performance. As iSCSI works on the network layer, the OCI Block Volume storage service also uses the same network backend (OCI **virtual cloud network**, or **VCN**) that an instance uses for connectivity. Not only that—OCI also offers the **paravirtualized** attachment of block volumes.

However, there are some restrictions. Not all instances can use paravirtualized block volumes. To use these, your instances must be provisioned using Oracle-provided images. Because OCI Block Volume storage operates at a raw-storage level, it performs consistently. OCI Block Volume storage performance is linear with the size of the storage, which means that, as the size of the storage grows, so does the level of IOPS that you can get out of it. It's so flexible that you can create a volume, attach it to an instance, and also move those block volumes whenever you need to. This total flexibility enables you to meet your applications' storage requirements.

These are typical use case scenarios for **Block Volume**:

* When you want persistent and durable storage for your application data
* If you want to expand an instance's local storage
* When you want to scale your instances

In the following diagram, you can get a high-level view of how an instance gets connected to a block storage and gets the boot volume and block volume for data from the same storage backend:

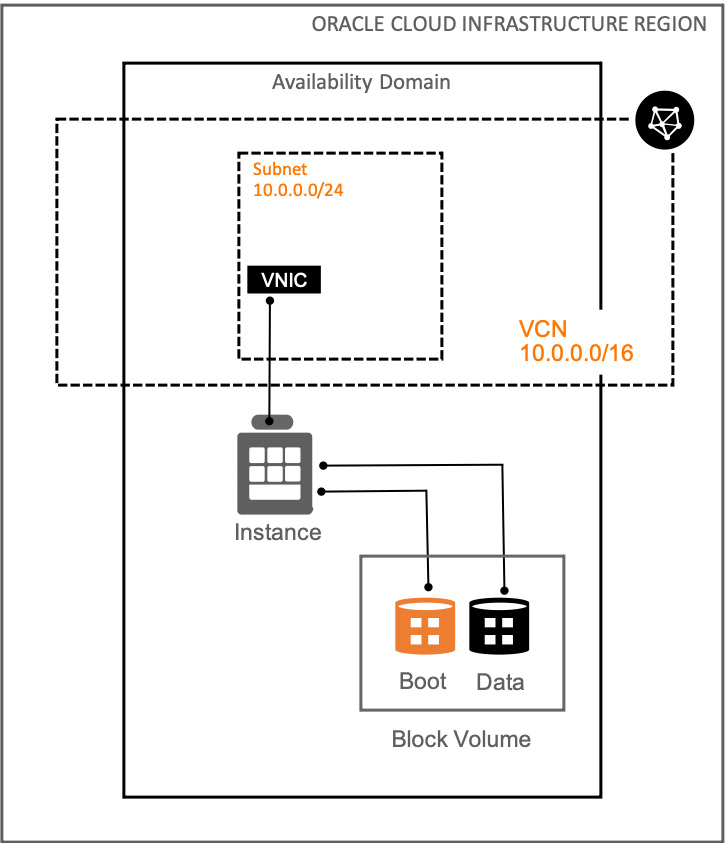


Figure 5.1 – Block storage overview

The Block Volume service is configurable from 50 **gigabytes** (**GB**) to 32 **terabytes** (**TB**), which is a huge range when you see it from an application's data requirement perspective. However, not only that: it also provides 32-volume attachment per instance. That means you can have 1 **petabyte** (**PB**) (32 GB x 32 volume) of block volume attached to an instance.

From a performance perspective, it is linear to the size of the storage. However, that is also dependent on an instance's shape because the shape decides the network speed of the instance. You can achieve 60 IOPS per GB of block volume for a **Balanced Performance** profile, at a maximum of 25,000 IOPS for **Balanced Performance** and 35,000 IOPS for **Higher Performance**. From a throughput perspective, you can achieve 480 **kilobytes per second** (**KBps**) per GB of block volume at a maximum of 320 **megabytes per second** (**MBps**) for a **Balanced Performance** profile. For a **Higher Performance** profile, this is 600 KBps per GB of block volume at a maximum of 480 MBps.

It is durable as well because OCI keeps multiple replicas across multiple storage server backends in an **availability domain** (**AD**). You also have an option to encrypt the volume at transit or at rest.

As you can imagine, the storage servers are hosted in a particular AD and connected to the AD local VCN, thus you can only access a volume from an instance hosted in the same AD and not from another AD.

**Creating a block volume**

So far, you have seen an overview of block storage and its use cases and performance criteria. In this subsection, we will go through creating a block volume and then attach it to an instance.

Let's create a block volume and connect to an instance, as follows:

1. Sign in to the OCI console.
2. Open the **Navigation** menu. Under **Core Infrastructure**, go to **Block Storage** and click **Block Volumes**.
3. Click **Create Block Volume**.
4. Provide the volume information, as follows:

a. **Name**—Provide a name for the volume.

b. **Compartment**—Choose a compartment where it is going to be created.

c. **AD**—Select the AD where it should be provisioned.

d. **Volume Size and Performance**—Either chose the **Default** size (which is 1,024 GB, and you can see the calculated performance numbers) or choose **Custom** and provide the size that you want. The benefit of choosing **Custom** is that you can choose **Higher Performance** with the same size of storage to satisfy your application requirement. You can see a sample output in the following screenshot:

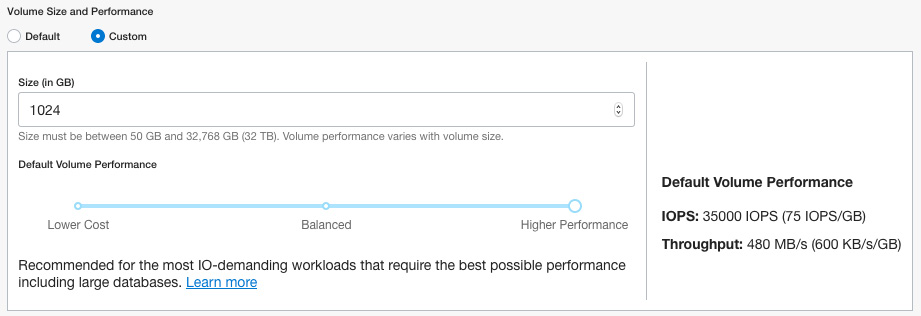


Figure 5.2 – Custom storage options for Higher Performance

e. **Backup Policy**—You can choose a backup policy from the list. We will discuss volume backups in detail in our upcoming section.

f. **Encryption**—You can either choose to encrypt this volume using Oracle-managed keys or customer-managed keys.

1. Click on **Create Block Volume**. You can see the overall workflow in the following screenshot:

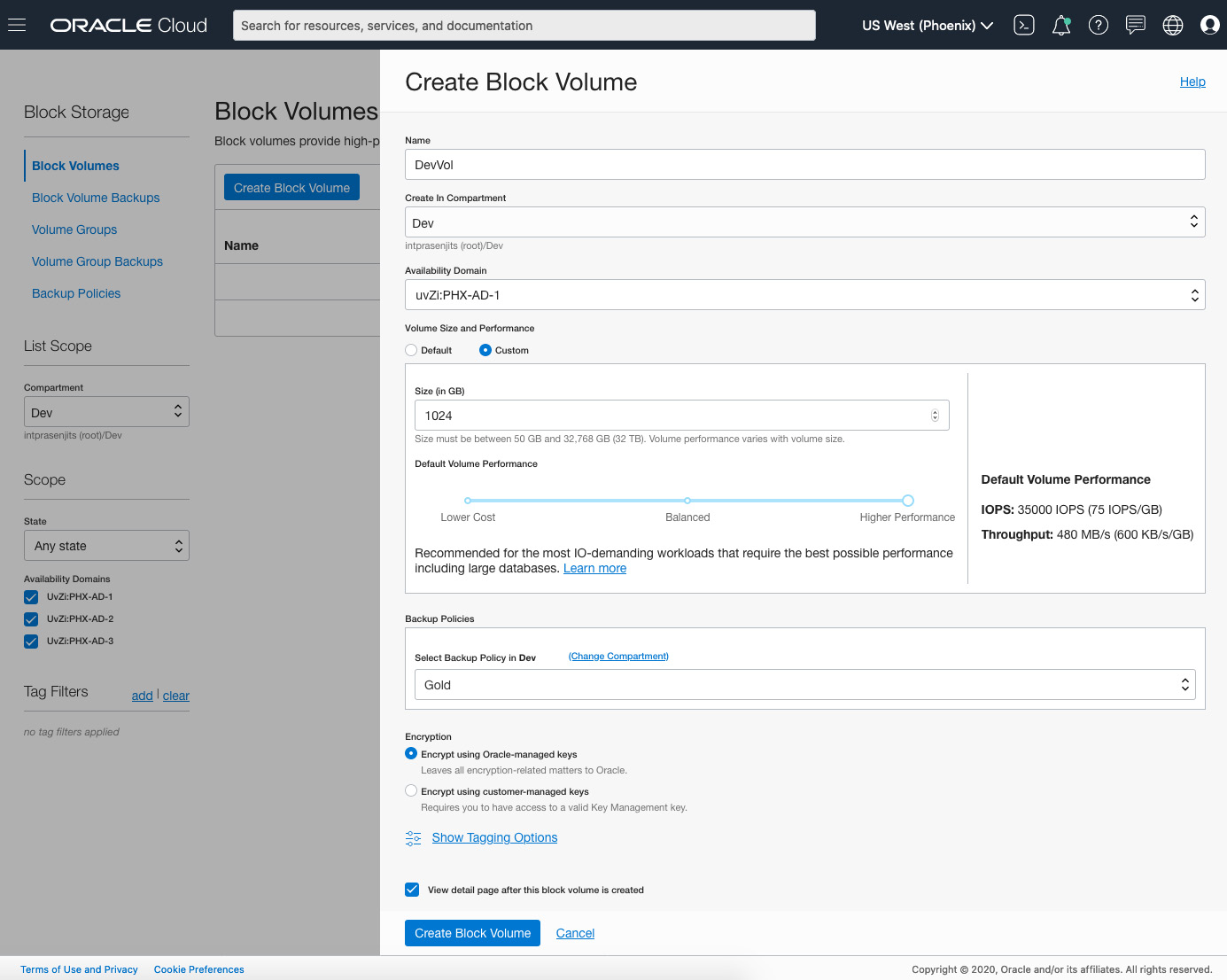


Figure 5.3 – Block volume creation workflow

So, you have now created a volume. Let's see how you can resize it if you have to.

**Resizing a block volume**

OCI also provides an online and offline volume resize feature through which you can enhance the performance of the block volume, or you can let OCI choose the performance based on your workload requirement and save money on it.

Let's perform an online resize of a block volume, as follows:

1. Sign in to the OCI console.
2. Open the **Navigation** menu. Under **Core Infrastructure**, go to **Block Storage** and click **Block Volumes**.
3. Click the **Actions** menu of the block volume that you want to resize and click on **Edit**.
4. On the **Edit** screen, you can increase the size of the volume and also choose the **Default Volume Performance** type.
5. Set **Auto-tune Performance** to **On**. This will help you to lower the cost when you detach the volume from an instance. It will retain this setting when you reattach it.
6. Click on **Save Changes**. You can see a sample output in the following screenshot:

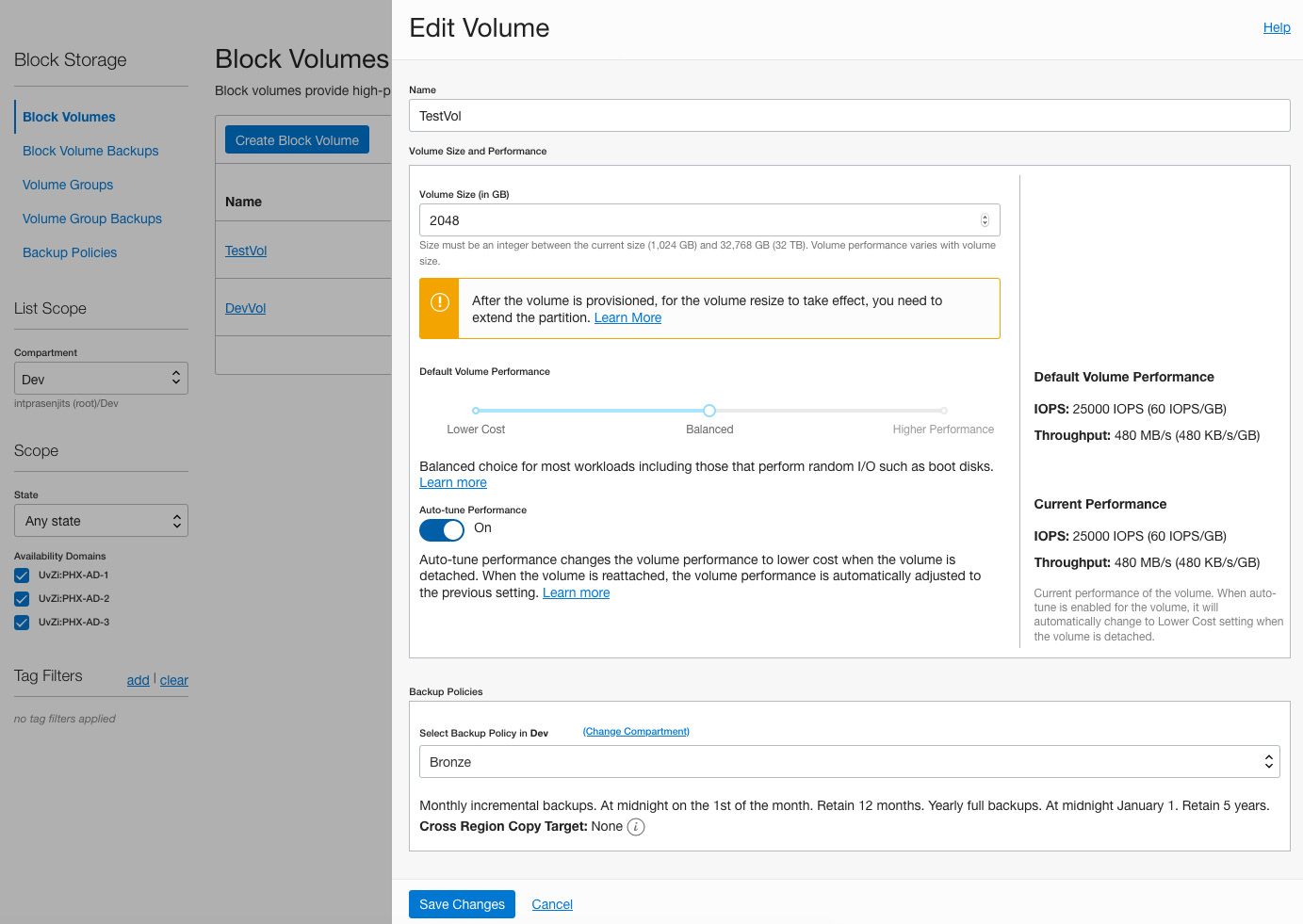


Figure 5.4 – Block volume creation workflow

You can see how you can use this resizing feature to keep a balance between performance and money for the block volume. Let's now see how you can attach this volume to an instance.

**Attaching a block volume to an instance**

There are two ways to attach a block volume to an instance. Either you can go to the **Instance Details** page and connect an existing block volume or you can go to the **Attached Instances** screen under the **Block Volume Details** page and do it. Let's go through the second option, as follows:

1. From the **Block Volume Details** page, go to the **Attached Instances** page. Click on **Attach to Instance**.
2. Depending on the type of the image to create the instance, select either **iSCSI** or **Paravirtualized**. We will choose **iSCSI** as that will walk you through the process of running iSCSI commands to mount the volume within the OS.
3. Choose an access type. In this case, we will show you the **Read/Write** method. If you want to create a cluster filesystem where you can simultaneously read and write data from multiple instances, then choose **Read/Write Shareable**. For a witness kind of a disk, you can choose a volume of **Read-only Shareable**. We will discuss this later in the chapter.
4. Choose the **Select Instance** option and select an instance from the dropdown.
5. You can optionally check the **Require CHAP credentials** option. We are not going to do this for this exercise.
6. Choose a device from the **Device Name** dropdown. If you select a provided device path, it becomes persistent between instant reboots. This is very important when you create a partition and filesystem and mount it inside the guest OS.
7. Click on **Attach**. You can see a sample output in the following screenshot:

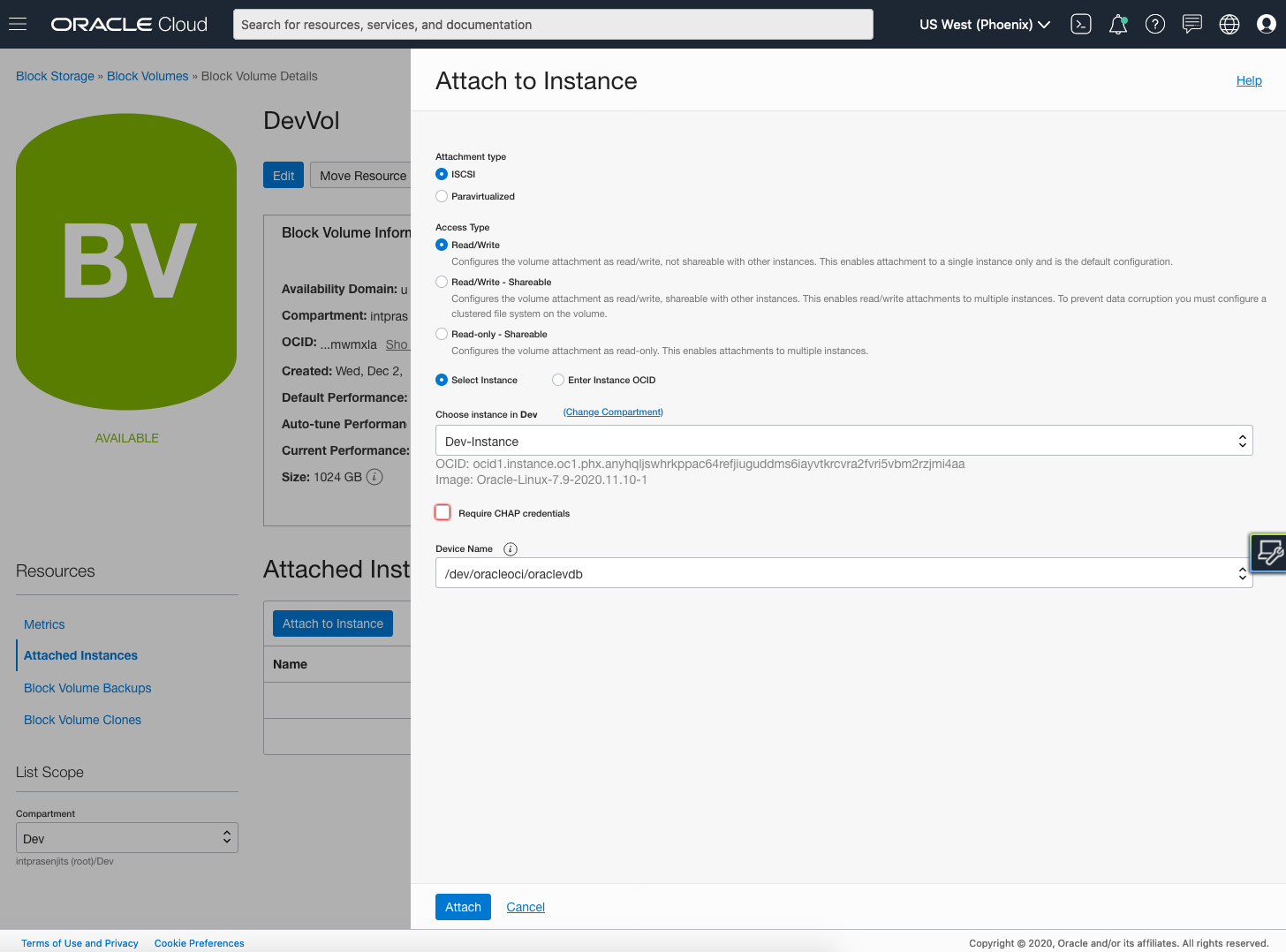


Figure 5.5 – Block volume attachment to instance

1. Once it is attached, you need to log in to the instance separately, using **Secure Shell** (**SSH**).
2. Copy the iSCSI connect command to connect the guest OS to the block volume. To do that, click the actions icon (three dots) next to the volume you're interested in, and then click **iSCSI Commands and Information**.
3. Copy the commands from the **Commands** section and paste it into a terminal window where you are logged in to the Linux instance.
4. This way, you will have the volume attached to the instance. But that's not all—you need to create a partition and filesystem and mount it within the guest OS as well. To do that, follow the standard method of creating a partition, create a filesystem, and mount it on the guest OS. You can see an example output in the following screenshot:

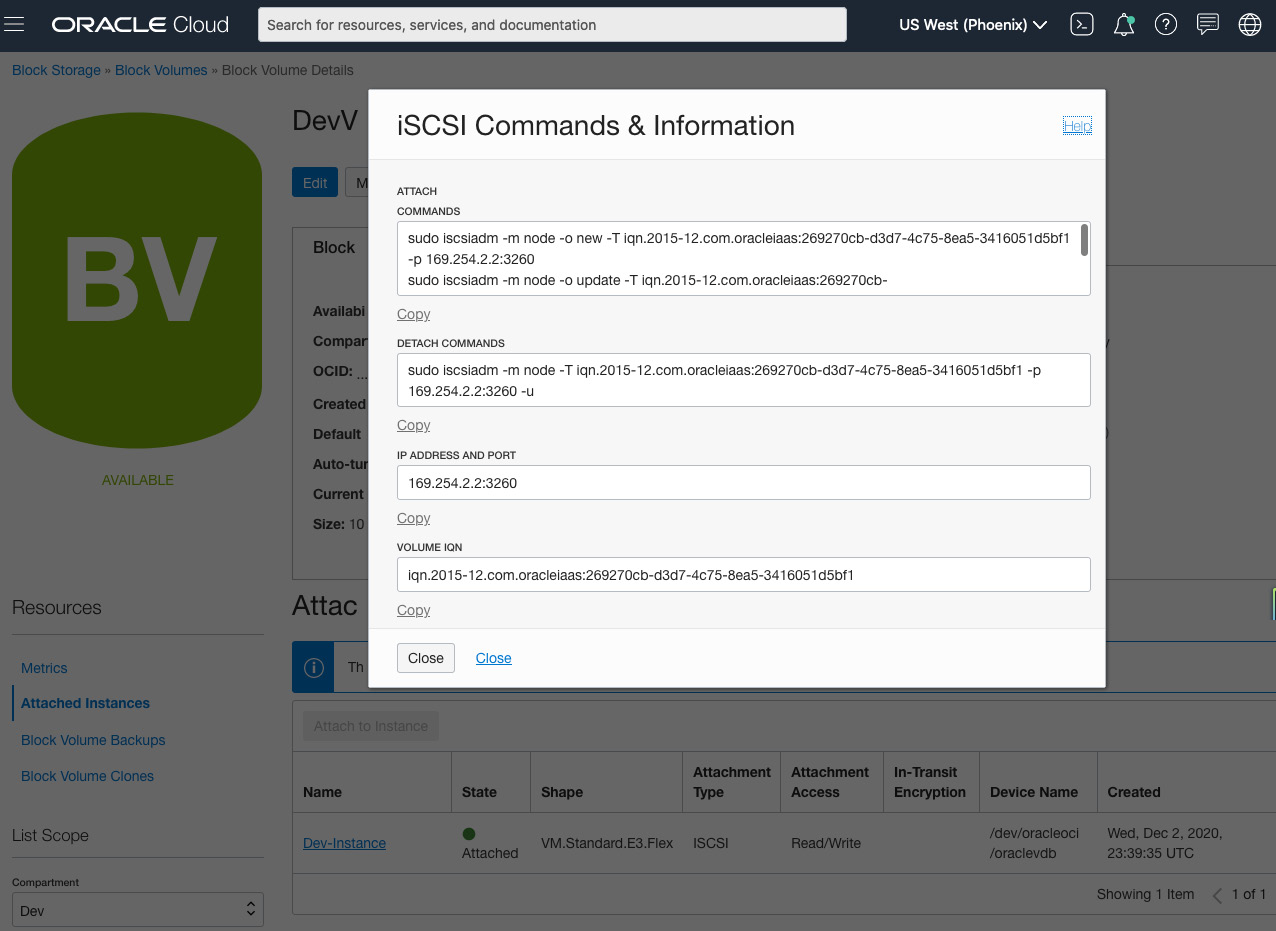


Figure 5.6 – iSCSI commands and information

So, you can see how easy it is to create a block volume, attach it to an instance, and then mount it within the filesystem. In the next section, let's talk about block volume backup operations.

**Backing up and restoring a block volume**

Block volumes allow you to take snapshots and point-in-time backups at any time. An interesting thing about volume backups is that they are differential and automatically managed, so a number of volume snapshot backups will chain to form the most recent backup, saving you space, time, and cost. The backup service manages these incremental backups for users. Backups go into the backup section of the storage service, which is based on Object Storage and gets encrypted as well. You can also copy the backup of a volume from one region to another.

When the time comes to restore, these backup/snapshots are automatically maintained so that any restorations will match the volume state at the time of the last snapshot. Users don't have to keep track of those incremental backups. Block volumes can be restored to the same or a different AD and reattached to the original instance or a new one.

You can see a logical representation of the backup operations of a block volume in the following diagram:

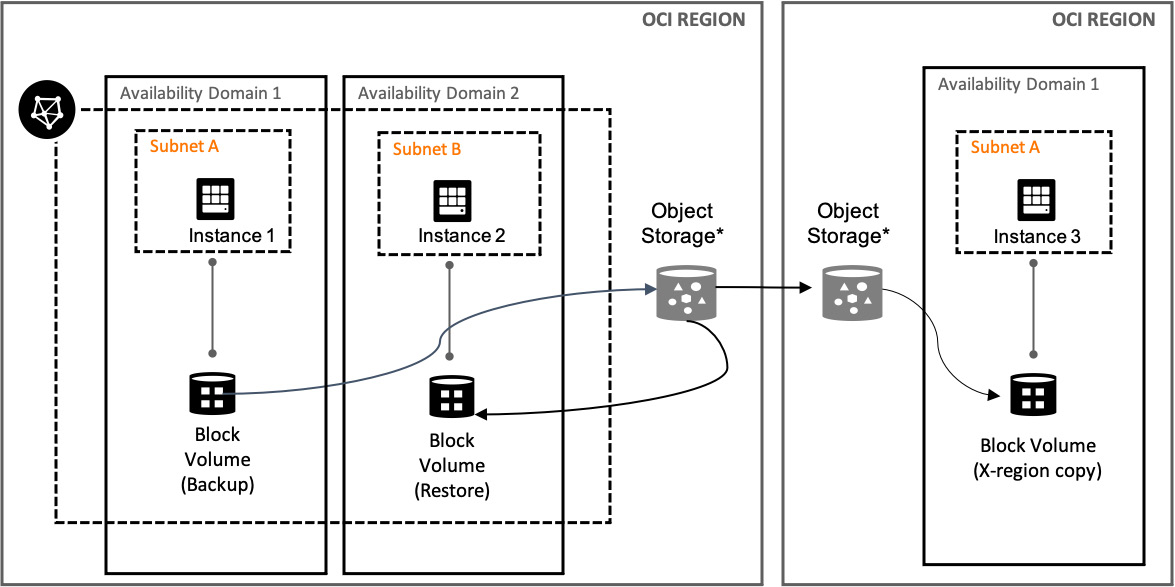


Figure 5.7 – Block volume backup operations

The preceding diagram shows how **Instance 1** is connected to **Block Volume (Backup)** and getting backed up to the Object Storage bucket, and it's getting used to restore it to **Instance 2**. Also, note how the **Object Storage** bucket is getting replicated to a different region so that you can have a **disaster recovery** (**DR**) copy of this data in a different geographical region.

You can either initiate a backup manually or you can assign a policy that defines a set backup schedule.

There are two backup types (manual configuration) available in the Block Volume service, outlined as follows:

* **Incremental**: Only the changes since the last backup will be backed up.
* **Full**: As the name suggests, this backup includes all changes.

You can choose to create a custom type of backup policy and let the policy drive the replication, schedule, and so on. OCI provides one more level of backup and restoration, and that is using a pre-defined policy.

There are three backup policies, outlined as follows:

* **Bronze**: Monthly incremental backups, whereby your data is retained for 12 months. For a full yearly backup, it is retained for 5 years.
* **Silver**: Weekly incremental backups, whereby your data is retained for 4 weeks. It includes capabilities from the Bronze category.
* **Gold**: Daily incremental backups, whereby your data is retained for 7 days. It includes capabilities from the Silver and Bronze categories.

Apart from schedule- and policy-based backups, you can always choose to get a manual backup done. Here is how it should be done:

1. Sign in to the OCI console.
2. Open the **Navigation** menu. Under **Core Infrastructure**, go to **Block Storage** and click **Block Volumes**.
3. Click on the **Actions** menu of the volume that you want to take a backup of and click on **Create Manual Backup**. You can see a sample output in the following screenshot:

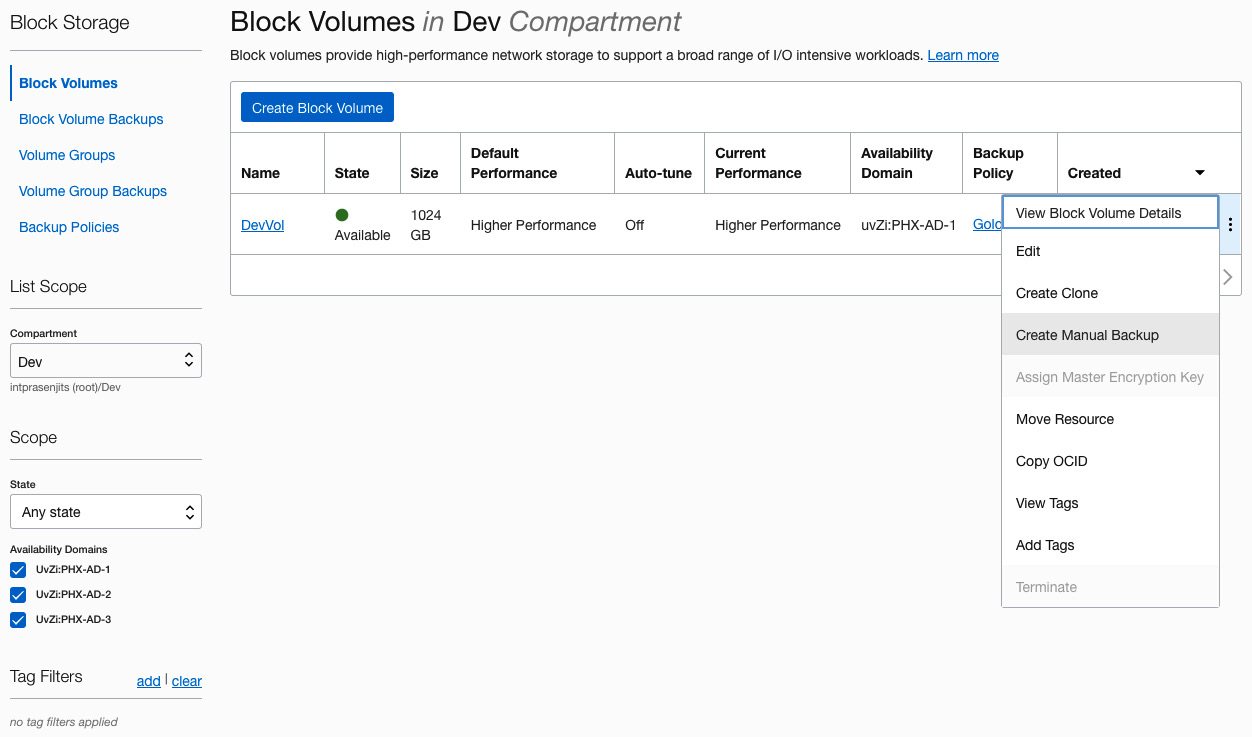


Figure 5.8 – Block volume manual clone

1. Provide a name for the backup and choose a type from the **Backup Type** dropdown. This can either be **Full Backup** or **Incremental Backup**.
2. Click on **Create Block Volume Backup**. You can see a sample output in the following screenshot:

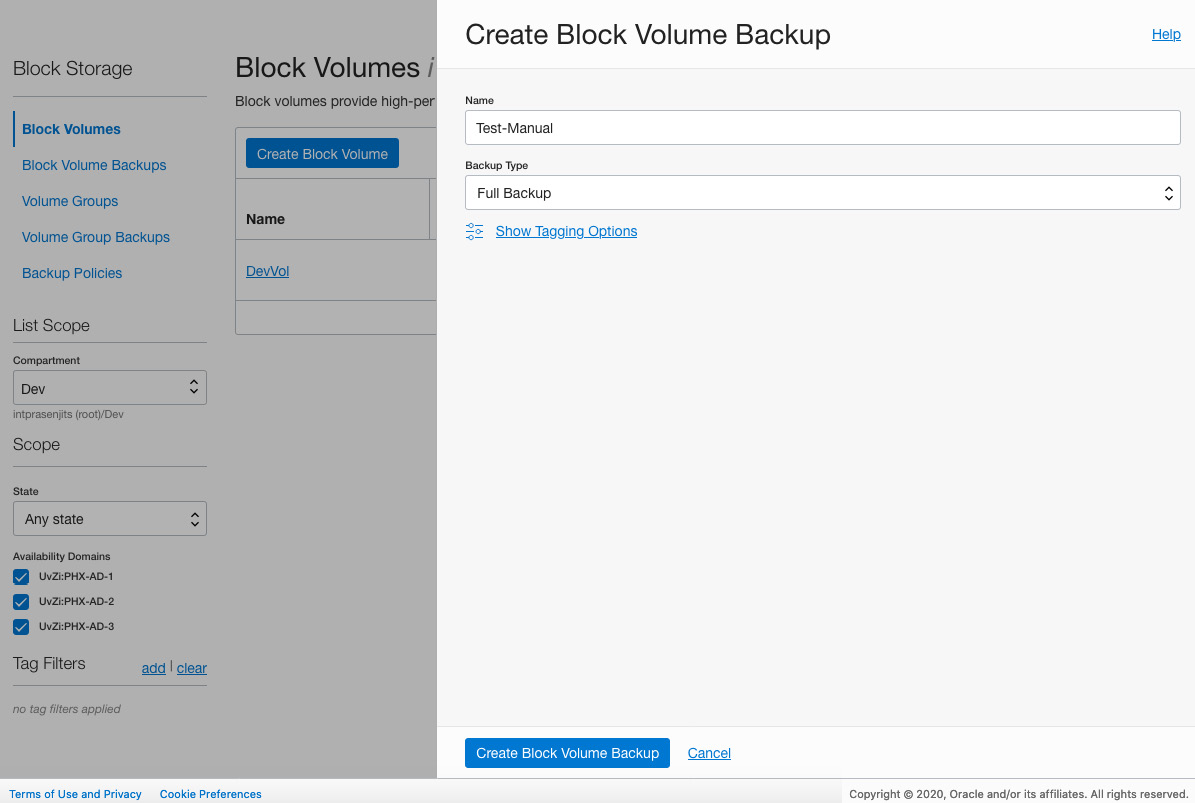


Figure 5.9 – Block volume manual clone workflow

So, you can see you have a choice when protecting your block volume data—either you can let OCI take a backup automatically using the policy and schedule or you can manually take a backup as well. In the next section, we will talk about another way of having a backup copy of your block volume.

**Cloning a block volume**

The Block Volume service also allows you to clone a block volume, whereby a cloning operation makes a copy of an existing block volume. It's a point-in-time direct disk-to-disk deep-copy process. When you initiate a clone operation, it just creates a cloned volume and initiates a data copy operation in the background. Once the copy is done, no data will be copied again from the source.

You are allowed to make a clone of a volume in the same region and same AD as the source volume. However, if you have adequate permission, then you can create a clone of the source volume and place it in a different compartment.

The best thing about doing a cloning operation on a block volume is that you can increase the size of the volume and also change the performance characteristics of the cloned volume. Let's take a clone of a block volume, as follows:

1. Sign in to the OCI console.
2. Open the **Navigation** menu. Under **Core Infrastructure**, go to **Block Storage** and click **Block Volumes**.
3. Click on the **Actions** menu of the volume you want to take a backup of and click on **Create Clone**.
4. Provide a name and choose a compartment in which you want this clone to be created.
5. For the **Volume Size and Performance** setting, you can either choose the **Default** option, which will keep the same characteristics as the source volume, or you can click on **Custom** and choose a volume size and performance.
6. For this example, click on **Custom** and change any of the characteristics of the volume (size or performance).
7. Select an **Encryption** type and click on **Create Clone**. You can see a sample output in the following screenshot:

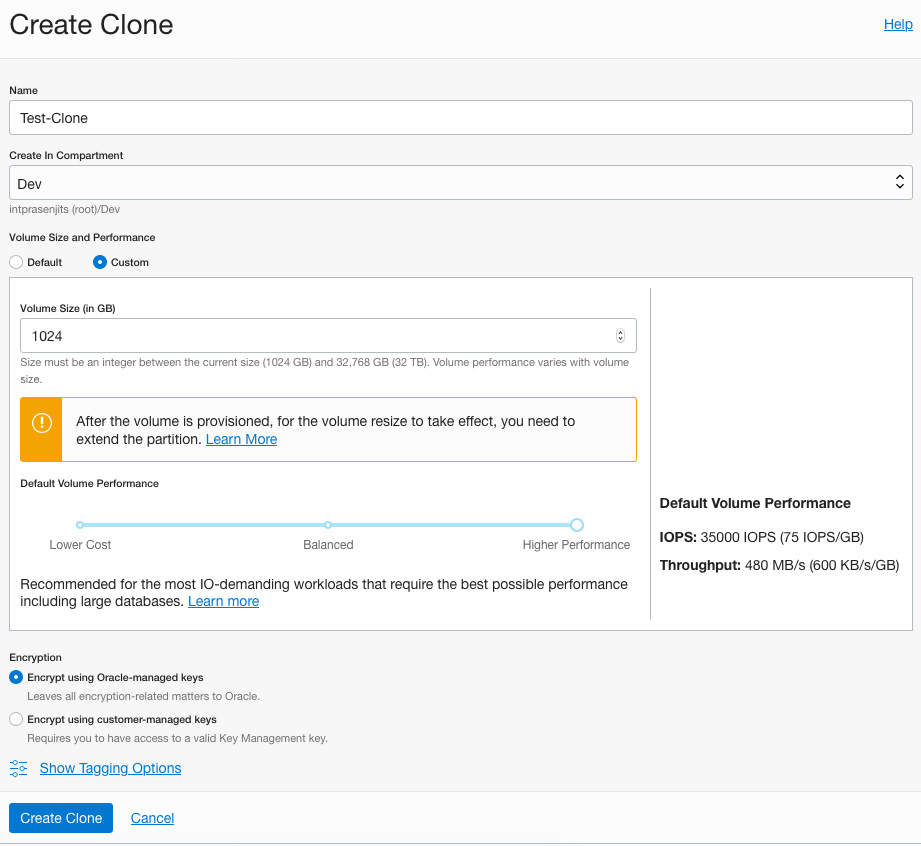


Figure 5.10 – Creating a clone of a block volume

So, you can see that by cloning, you can create a bigger volume, change the performance characteristics, and at the same time keep a backup of the volume as well. In the next section, we will discuss how you can group these volumes together to perform the operations we have discussed so far.

**Volume groups**

The OCI Block Volume service lets you group together block and boot volumes from multiple compartments across multiple compute instances in a volume group. Volume groups are essential when you create volume group backups and clones.

We can manually trigger a full or incremental backup of all the volumes in a volume group, leveraging a coordinated snapshot across all the volumes. Volume groups help to manage the life cycle of enterprise applications because these enterprise applications require multiple volumes across multiple compute instances to function effectively.

You can see a logical representation of volume groups in the following diagram:

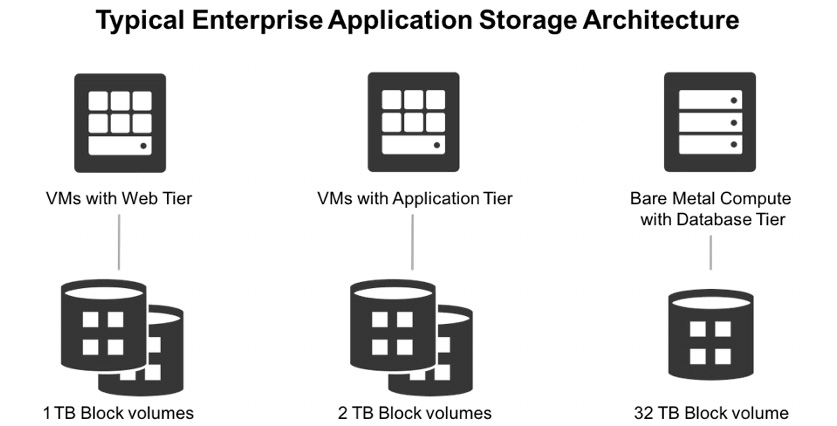


Figure 5.11 – Volume groups overview

The preceding diagram shows the grouping of volumes that are getting tied up with a set of **virtual machines** (**VMs**).

Let's quickly create a volume group from the OCI console, as follows:

1. Sign in to the OCI console.
2. Open the **Navigation** menu. Under **Core Infrastructure**, go to **Block Storage** and click **Block Volumes**.
3. Click on the **Volume Groups** section and click on **Create Volume Group**.
4. Provide a name and choose a compartment in which you want this volume group to be created.
5. Choose an AD from where you want to pick up the volumes.
6. In the **Volumes** section, make selections from the **Compartment** and **Volume** dropdowns. You can click on **+ Volume** to add multiple of them.
7. Click on **Create Volume Group**. You can see a sample output in the following screenshot:

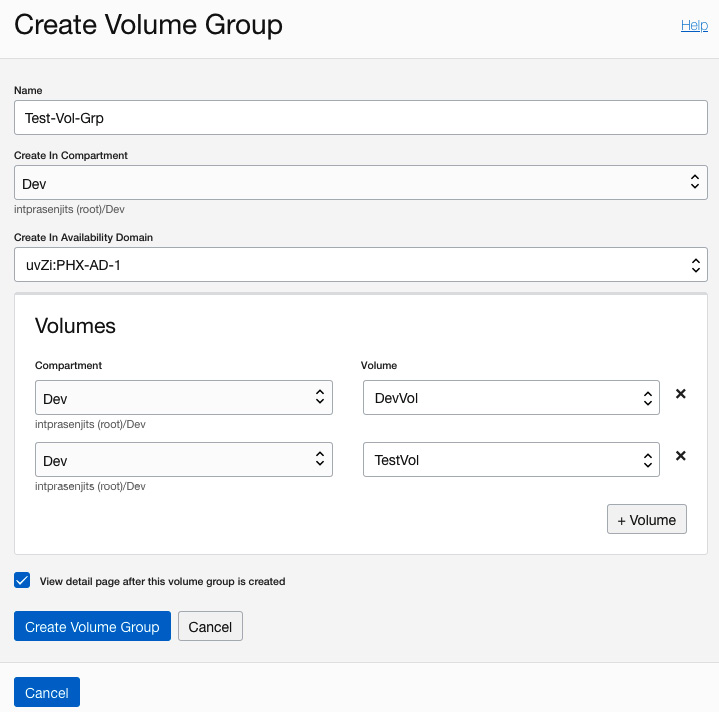


Figure 5.12 – Volume group creation

In this section, you have seen how to group volumes together to perform a backup of the volumes together.

**Volume groups for coordinate backups**

A volume group is used when you want to create a one-time backup of all the volumes, and as a result it generates a volume group backup. You can then use this volume group backup to create a new volume group, and to do so you need to restore all the data for volumes that are in the volume group.

From a volume group, a manual backup can be created. This can be either a full or an incremental backup (a full backup is required to have happened before an incremental backup can be triggered). When backing up a volume group, an aggregated state for the backup is available to enable applications to query the progress of the backup, in particular to figure out when the coordinated backups have been completed.

Restoring a backup of a volume group will create a new volume group containing the restored volumes. A restore operation can be made to a different AD than the source volume group, but this must be in the same region.

You can see an overview of this in the following diagram:

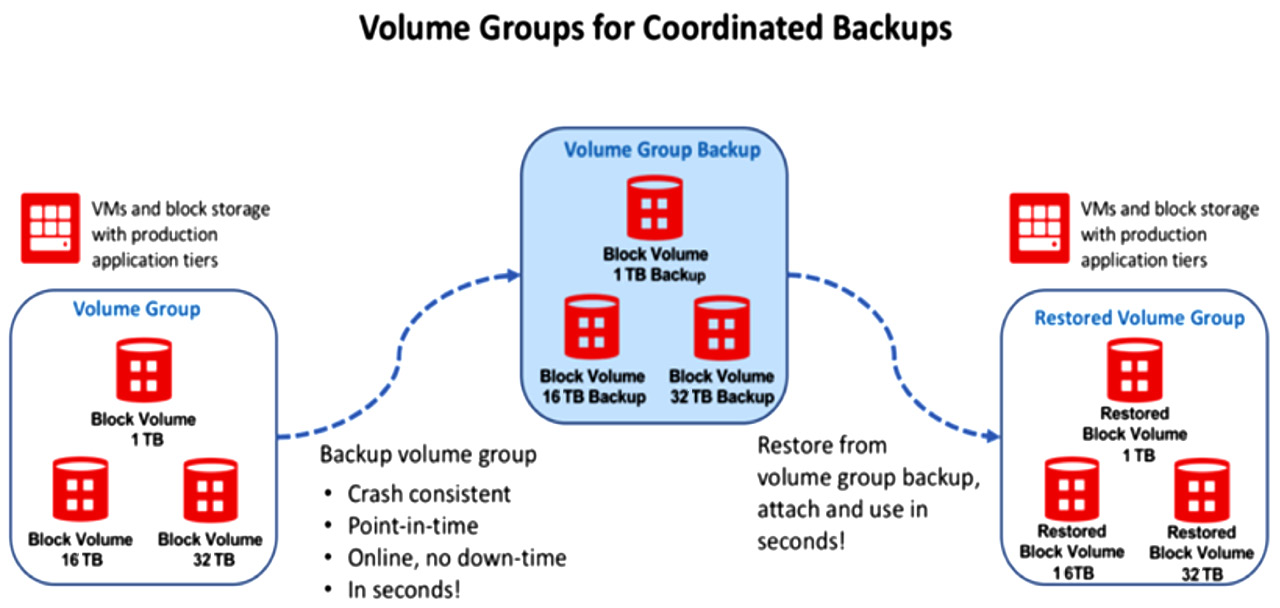


Figure 5.13 – Volume group coordinated backup

In the preceding diagram, you can see how a volume group is getting backed up and then restored to a different volume group.

We have discussed backup options, so let's now discuss cloning options.

**Volume groups for coordinate clones**

A coordinate clone is a deep disk-to-disk copy of an entire volume group. This creates a new completely isolated volume group and a new set of volumes that are also completely isolated from their corresponding source volumes.

There is no delay in a cloning operation, which means it is immediate, and right after initiating a cloning operation, you can access the cloned volume group and cloned volumes within that. While you access the volume groups and clone volumes, your data gets copied in the background. The time it takes to finish the actual data copy depends on the amount of data that you are copying across—for example, copying 1 TB of data can take up to 15 minutes.

You can see an overview of this in the following diagram:

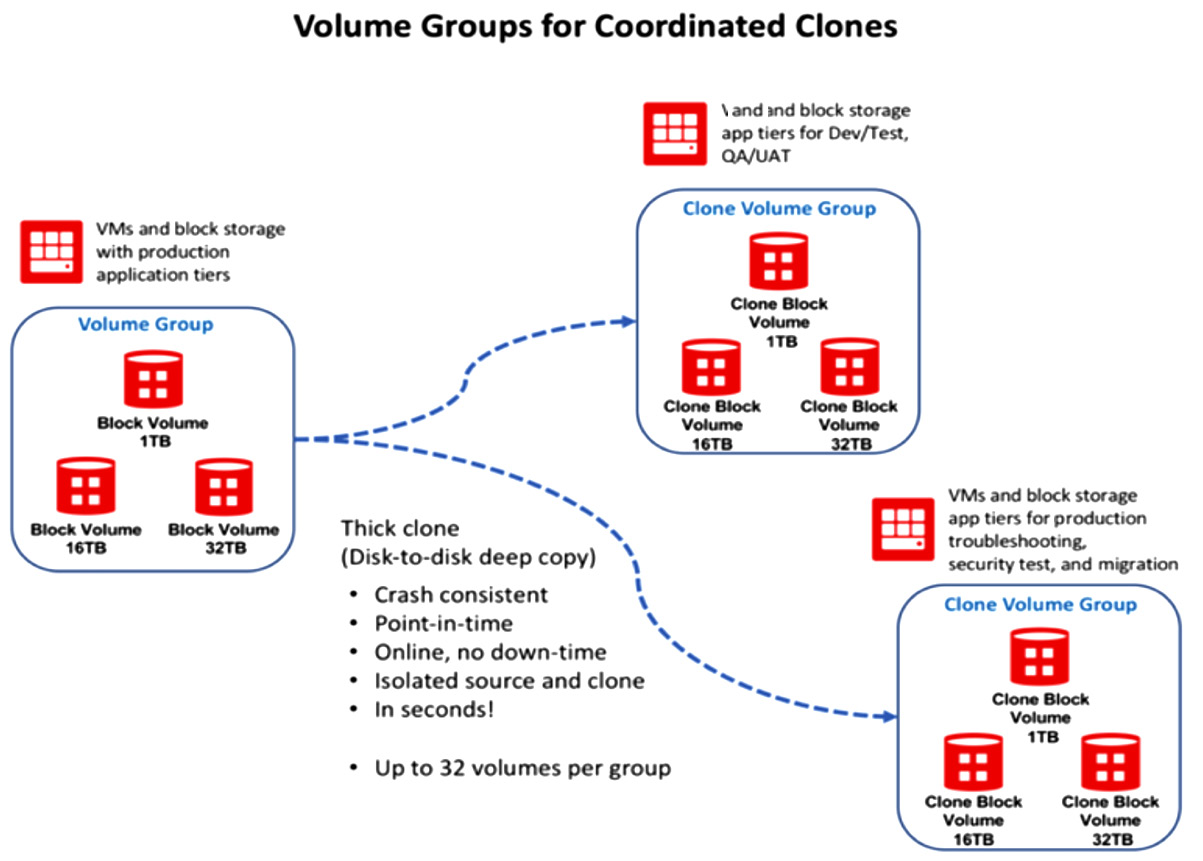


Figure 5.14 – Volume group cloning operation

The preceding diagram shows the coordinated clone operations of a volume group.

In the next section, you will see how you can attach one block volume to multiple instances.

**Block Volume operations – shared multi-attach**

The OCI Block Volume service has the capability to let you attach one single block volume to multiple instances. You can attach a volume in either **R/O** or **R/W** mode. The main use case of a shared volume is a cluster filesystem.

*IMPORTANT NOTE*

*One thing that you have to keep in mind is that the Block Volume service does not provide the capability for concurrent writes on this block device, so you need to install a cluster filesystem on top of the block volume.*

You can see a logical representation of a shared multi-attach volume in the following diagram:

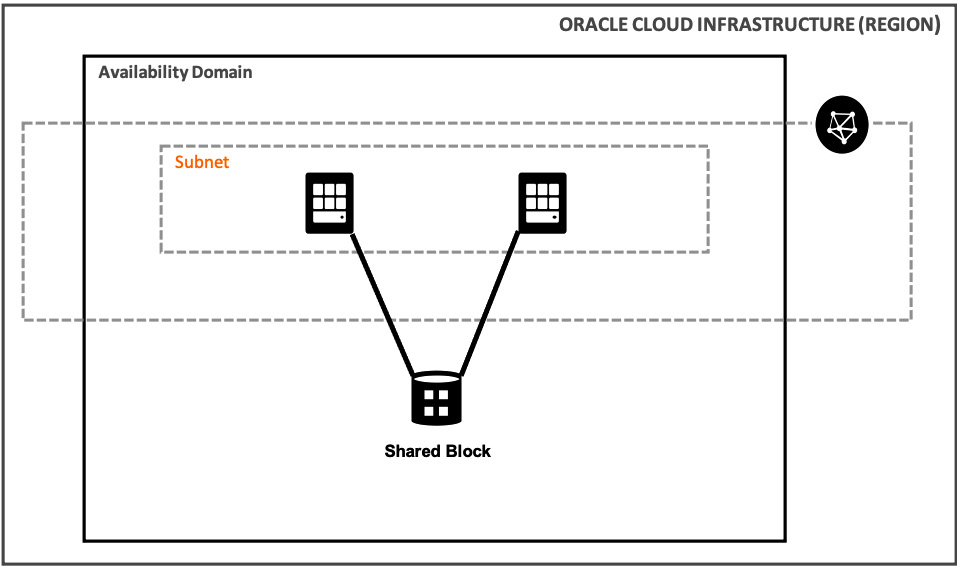


Figure 5.15 – Shared block volume attachment

In the preceding diagram, you can see that two different instances are connected to the same shared block volume in a single AD.

So, not only can you have performance and capacity, but you can also have other options, such as a cluster-aware application running on the OCI block volume too.

**OCI File Storage Service**

The OCI FSS is an enterprise-grade network filesystem, and at the same time, it is durable, scalable, and secure as well. As it's a network file storage service, it utilizes the same VCN backbone to serve, so you can use any instance type, be it bare-metal or VM, or even containers as well. If you implement VCN peering or a FastConnect or **Internet Protocol Security** (**IPsec**) VPN, then you can even access the filesystem from outside of the host VCN.

It's so massively scalable that you can even connect thousands of compute instances to the filesystem. It uses the **Network File System version 3.0** (**NFS v3.0**) protocol, and for the locking mechanism, it uses **Network Lock Manager** (**NLM**). For the best reliability, OCI uses five-way replication of the filesystem and stores it in different fault domains. For data protection, it uses snapshots; you can have 10,000 snapshots per filesystem. For security, it employs 128-bit data-at-rest encryption for all filesystems and metadata.

These are the main use cases of File Storage:

* Oracle applications lift and shift
* General-purpose filesystem
* Big data and analytics
* **High-performance computing** (**HPC**) scale-out applications
* Test/dev databases
* Microservices and containers

**Creating a filesystem**

A filesystem in the OCI FSS is a primary resource for storing files. To access your filesystems, you need a new or an existing mount target. You can have 100 filesystems per mount target. You can see an overview of a filesystem in the following diagram:

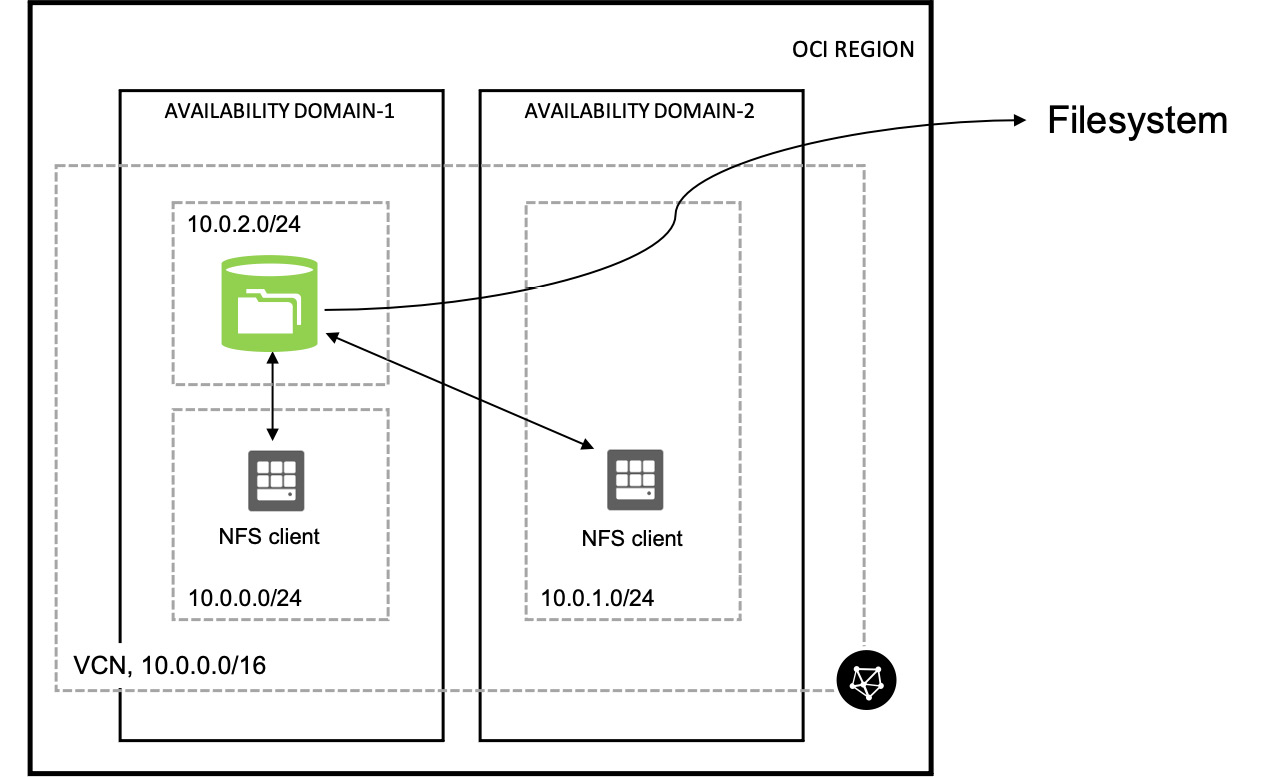


Figure 5.16 – Overview of FSS filesystem

To have access to this filesystem, you need mount points. Mount targets are NFS endpoints that are connected to your choice of VCN. These mount targets have an IP address and **Domain Name System** (**DNS**) name that you can use in your mount command.

Each mount target requires three private IP addresses in the subnet; you can't use a **/30** subnet or smaller subnets for the mount targets. Out of those three, two of the IP addresses are used during mount target creation, and the third IP is used for **high availability** (**HA**).

However, you need to be careful placing NFS clients and mount targets in the same subnet as that can cause IP conflicts, as users are not shown which private IPs are used for the mount targets. To avoid this, place the FSS mount target in its own subnet, where it can consume IPs as it needs.

Before we proceed further and discuss filesystem security, let's first create a filesystem and a mount target, as follows:

1. Sign in to the OCI console.
2. Open the **Navigation** menu. Under **Core Infrastructure**, go to **File Storage** and click **File Systems**.
3. Click on **Create File System**.
4. By default, OCI will provide all the required information—for example, name of the filesystem, encryption type, and export path—and will also create a mount target along with this.
5. However, you can click on the **Edit Details** link to edit each section. You can see a sample output in the following screenshot:

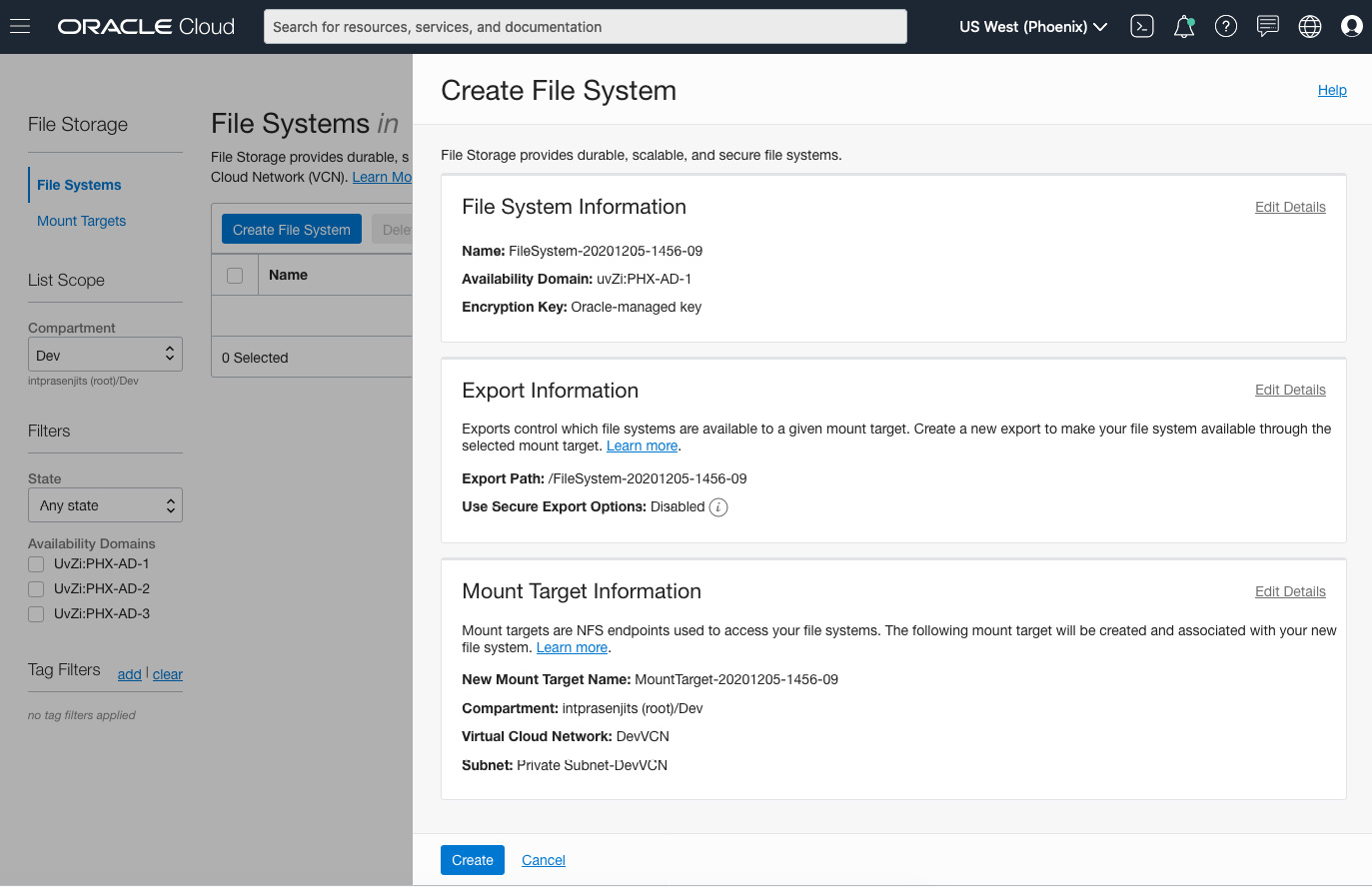


Figure 5.17 – Creating a filesystem

1. Click on **Create**.

The preceding steps will let you create a filesystem but it is not mounted to any instance, and the VCN security list is also not prepared for it to be mounted. Let's do that now, as follows:

1. Sign in to the OCI console.
2. Open the **Navigation** menu. Under **Core Infrastructure**, go to **File Storage** and click **File Systems**.
3. Click on **File System** to go to the **Filesystem details** page.
4. From the **Resources** section, click on **Exports**.
5. Click on the export that was created while creating the filesystem.
6. Click on **Mount Commands** and you will see the **Mount Commands** details screen. You can see an example screenshot here:

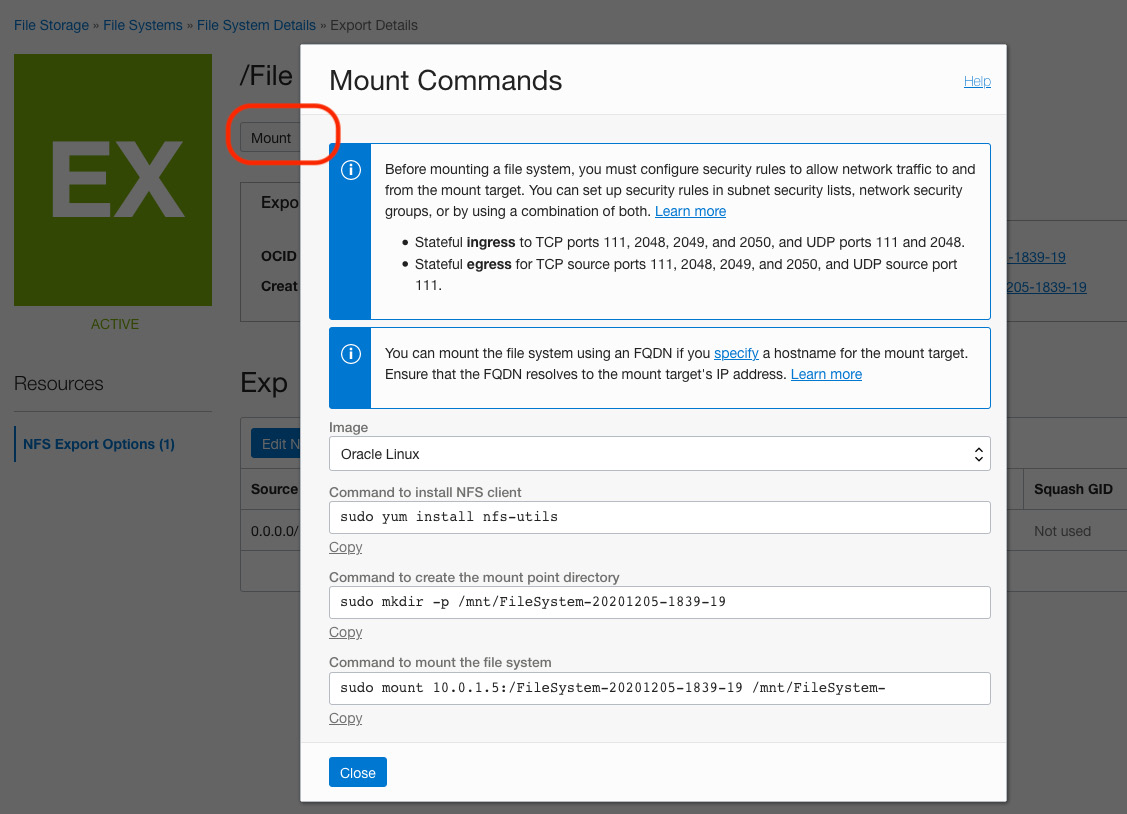


Figure 5.18 – FSS mount command screen

1. From here, you can see that you have to create a couple of security rules in your subnet to allow this mount target to be connected to the instances.
2. Open up the security list according to the given ports and traffic direction.
3. Choose the OS image and copy and run the commands inside the guest OS to mount the filesystem.

**Filesystem security**

A security list can be used as a virtual firewall to prevent NFS clients from mounting an FSS mount target (even in the same subnet). FSS needs these ports to be opened:

* Stateful ingress **Transmission Control Protocol** (**TCP**) ports **111**, **2048**-**2050**
* Stateful ingress **User Datagram Protocol** (**UDP**) ports **111** and **2048**
* Stateful egress TCP ports **111**, **2048**-**2050**
* Stateful egress UDP port **111**

However, using a security list to block unwanted traffic to a filesystem is not ideal because you can either allow access or block whole access to the mount target. Therefore, this will be applicable on all filesystems associated with it.

In a multi-tenant environment, you should use an NFS export option to limit how a client is able to connect to a filesystem.

By default, an NFS export option will allow full access for all NFS clients. You can see an example of that configuration in the following screenshot:

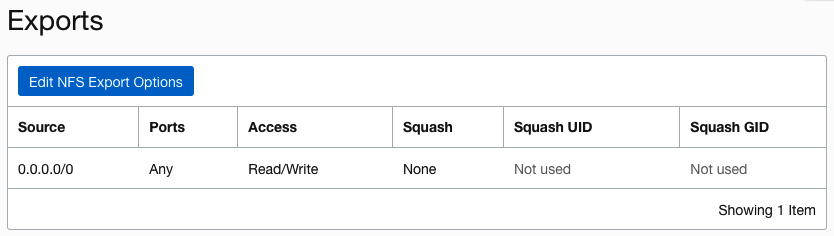


Figure 5.19 – NFS export option

Let's look at an example where the following applies:

* **Client X**, assigned to **10.0.0.0/24**, requires **Read/Write** access to **filesystem A**, but not **filesystem B**.
* **Client Y**, assigned to **10.0.1.0/24**, requires **Read** access to **filesystem B**, but no access to **filesystem A**.
* Both filesystems **A** and **B** are associated with a single mount target.

You can see an example of the preceding scenario in the following diagram:

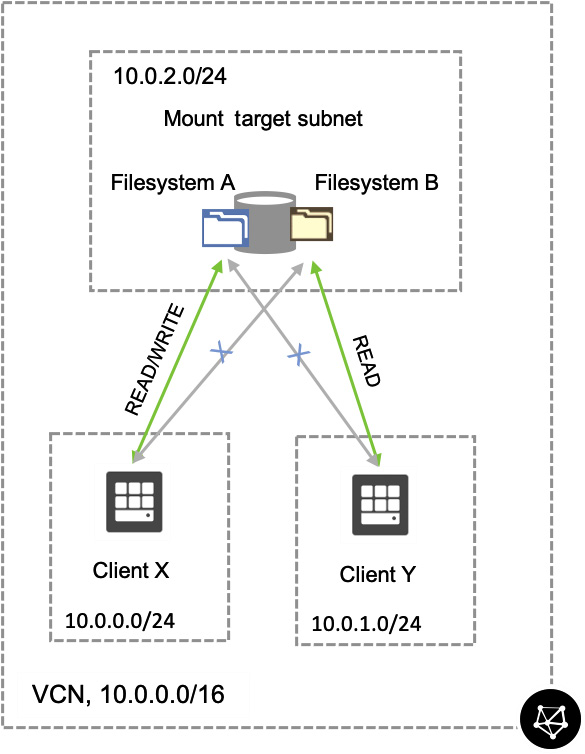


Figure 5.20 – Export option rule

You need to run the following commands to make it happen. To do that, you can use OCI Cloud Shell:

oci fs export update --export-id <FS\_A\_export\_ID> --export-options '[{"source":"10.0.0.0/24 ","require-privileged-source-port":"true","access":"READ\_WRITE","identity-squash":"NONE","anonymous-uid":"65534","anonymous-gid":"65534"}]'

You have just allowed an **export** option from a particular subnet; let's do that for the other one as well, as follows:

oci fs export update --export-id <FS\_B\_export\_ID> --export-options '[{"source":"10.0.1.0/24 ","require-privileged-source-port":"true","access":"READ\_ONLY","identity-squash":"NONE","anonymous-uid":"65534","anonymous-gid":"65534"}]'

The preceding commands will update **NFS Export Options** to allow specific clients to use only specific mount targets.

With this, we have concluded our discussion on the OCI FSS. In the next section, we will go through OCI Object Storage.

**OCI Object Storage**

In our earlier section, we discussed high-performance, durable, secure, and scalable storage solutions, but OCI has lot more to offer. OCI offers "hot" storage that is frequently accessed and "cold" storage that is less frequently accessed.

These are two different storage tiers that are performant as well, outlined as follows:

* **Object Storage**: This tier of storage is what you need when you require fast, immediate, and frequent access to data.
* **Archive Storage**: This tier of storage is what you need when you don't require frequent access to data, but the retention period for this type of data is long.

The OCI Object Storage service is an internet-scale, high-performance storage platform. If you think about storing unlimited bytes of unstructured data, such as images, media files, logs, and backups, then this is an ideal solution.

Data in object storage is managed using an **application programming interface** (**API**) and standard **HyperText Transfer Protocol** (**HTTP**) verbs. This service is regional. OCI Object Storage not only provides object storage but also provides an archive storage solution at a low cost for less frequently accessed data. You can either put the objects (your data) onto the object storage and have it as public-facing or you can put the data onto private access. If it is put onto **Private access** mode, you can use an **OCI service gateway** to access this data from the OCI resources. You can create a replication policy and have the objects copied over to a different region. You can also give pre-authenticated access to either a whole bucket or a specific object. For larger files, you can use a multipart upload as well.

There are three resources in the OCI Object Storage service, outlined as follows:

* **Object**: Objects are nothing but data, regardless of content type. Each object is composed of data and metadata.
* **Bucket**: A bucket is what you store your objects in. You have to store all of your objects (that is, your data) in a bucket.
* **Namespace**: This is a top-level container and refers to all buckets and objects. Each tenancy has a unique namespace that spans across all compartments and regions. You cannot customize, change, or request a namespace name change.

Let's create an object storage bucket and upload a file (as an object) to the bucket, as follows:

1. Sign in to the OCI console.
2. Open the **Navigation** menu. Under **Core Infrastructure**, go to **Object Storage** and click **Object Storage**.
3. Click on **Create Bucket**.
4. Provide a name for the bucket, and then choose a storage tier from either **Standard** or **Archive**. You can check **Emit Object Events**, which is needed when you run automation based on object upload. You can also choose **Enable Object Versioning**—choose an **Encryption** method and click on **Create Bucket**. You can see a sample output in the following screenshot:

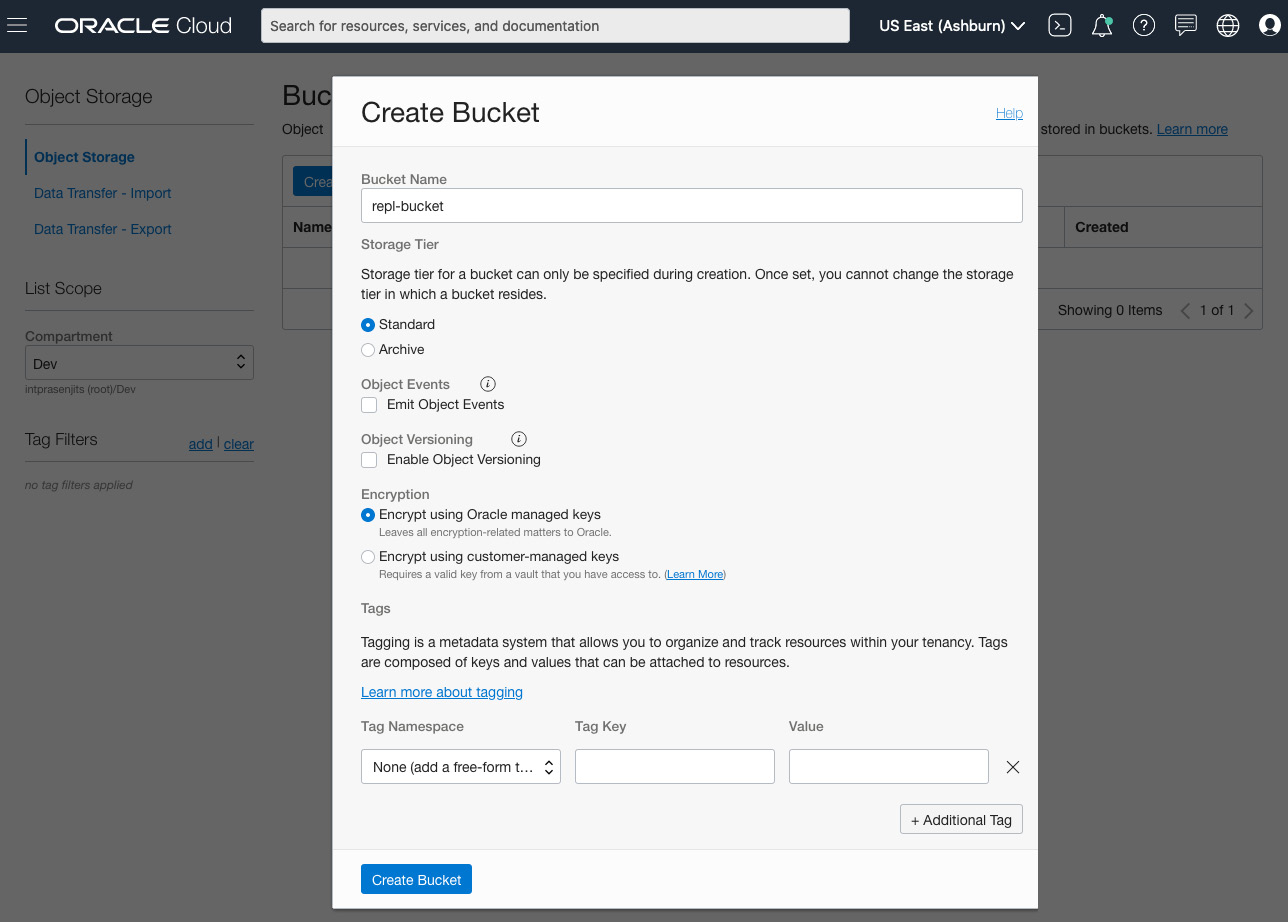


Figure 5.21 – Creating an Object Storage bucket

You have just created a bucket. Now, it is time to upload some objects to it, as follows:

1. From the **Object Storage Bucket Details** page, select **Objects** under the **Resources** section.
2. Click on **Upload**.
3. Select a file or drag a file onto the **Choose Files from your Computer** section.
4. Click on **Upload**. You can see a sample screenshot of this operation here:

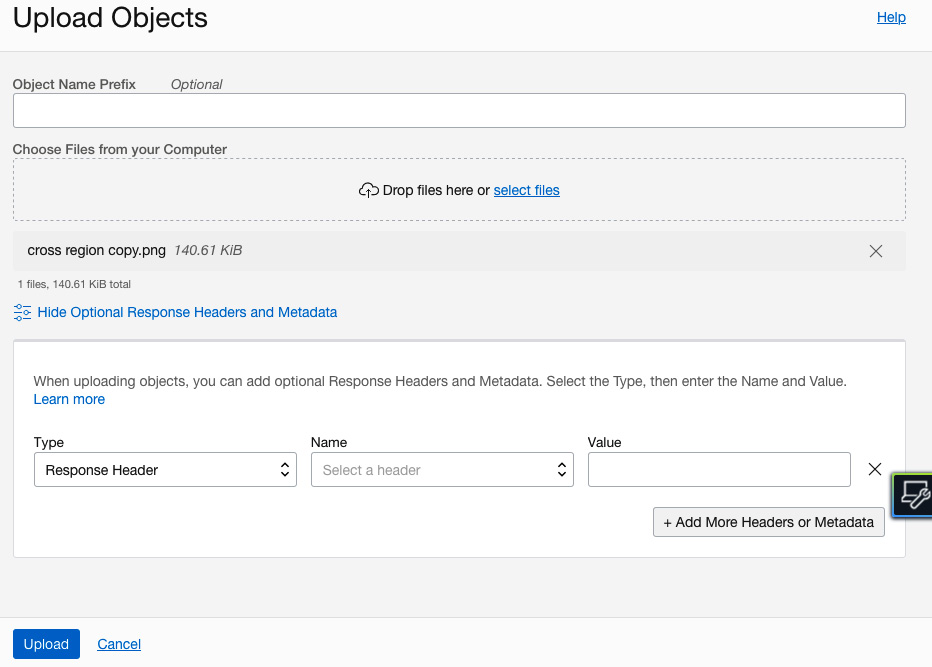


Figure 5.22 – Uploading objects

So, you have just created a bucket and uploaded some objects, but you did not provide access to it. So, let's see what we can do about that.

**Pre-authenticated requests**

The OCI Object Storage service provides **pre-authenticated requests** (**PARs**). PARs help you to access either an object or a bucket without having your own credentials to access it. You can either create a PAR for a whole bucket or you can choose individual objects to grant access as well.

Let's provide access to the just-created object, as follows:

1. From the **Object Storage Bucket Details** page, select **Objects** under the **Resources** section.
2. Click on the **Actions** menu of the object you want to create a PAR for.
3. Click on **Create Pre-Authenticated Request**.
4. Provide a name for the PAR. By default, it will show this as an **Object** PAR, but you can choose **Bucket** as well.
5. Choose an access type: either read, write or read, or both read and write.
6. Choose an expiration time. By default, it will give you 7 days' access to this object. You can see a sample output in the following screenshot:

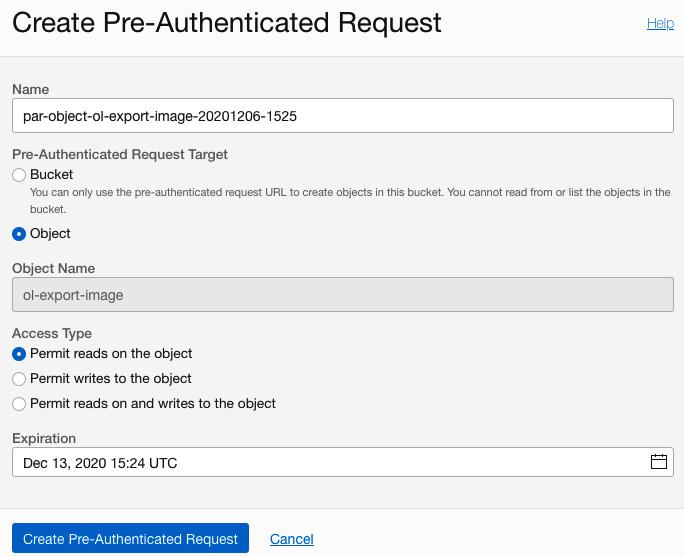


Figure 5.23 – Creating a PAR

1. Click on **Create Pre-Authenticated Request**.

This way, you can have public access to any object for a specified period of time.

**Cross-region copy**

OCI Object Storage provides you with the ability to copy objects across regions using a replication policy. You have an option to choose between either copying objects to another bucket but in the same region or to another bucket but in a different region. However, you cannot copy objects from Archive Storage.

Let's perform a cross-region copy of the objects, as follows:

1. From the **Object Storage Bucket Details** page, select **Replication Policy** under the **Resources** section.
2. Click on **Create Policy**.
3. Provide a name for the policy.
4. Choose a **destination region**.
5. Choose a **destination bucket** in the selected compartment from the destination region. You can see a sample screenshot of this operation here:

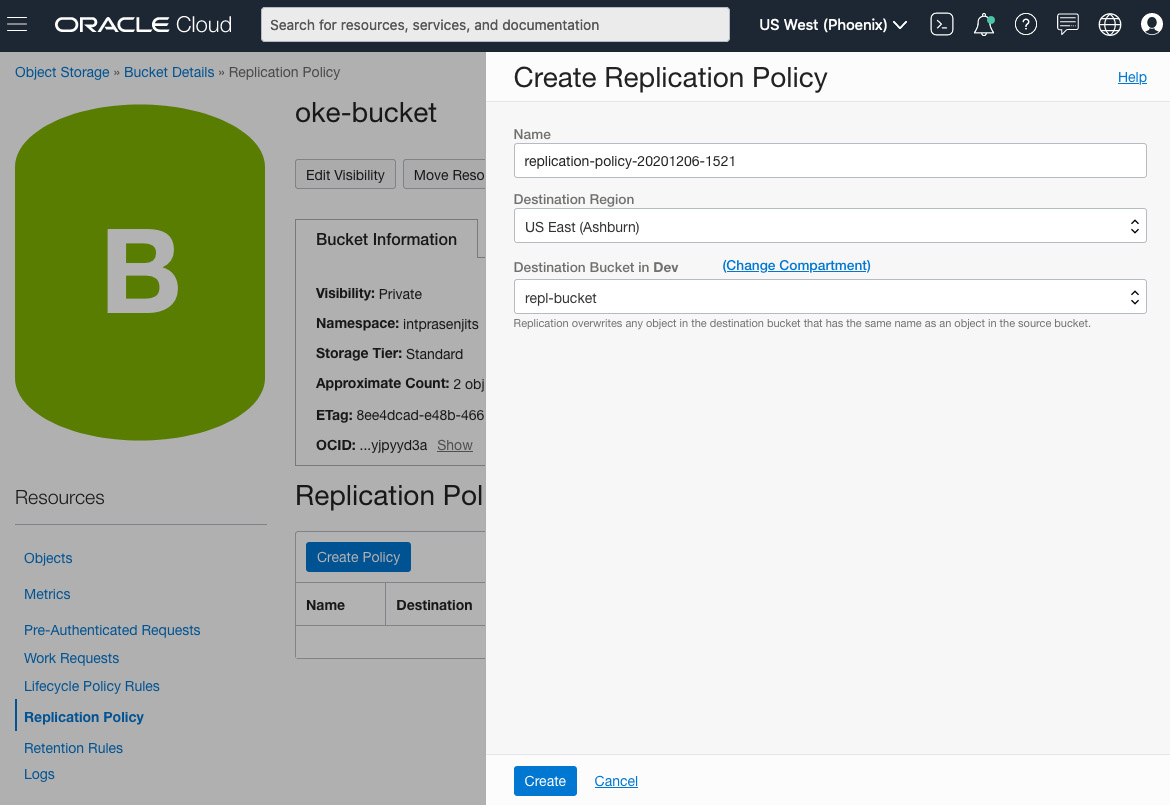


Figure 5.24 – Cross-region copy of the objects

1. Click on **Create**.

**Multipart upload**

A multipart upload is basically a parallel operation through which your objects will be uploaded to a bucket in parts so that the time to upload large objects can be reduced. There are four steps to a multipart upload, outlined as follows:

1. **Create object parts**—A multipart upload is performed when you upload objects larger than 100 **megabytes** (**MB**). At a maximum, an individual part of the data can be of 50 GB in size, or it can be as small as 10 MB. In this phase, the multipart upload will assign part numbers from 1 to 10,000 to the object.
2. **Initiate an upload**—Once the multipart upload engine creates a chunk, then a **REpresentational State Transfer** (**REST**) API called **CreateMultipartUpload** will be called to initiate a multipart upload.
3. **Upload object parts**—After the initialization, this engine initiates an **UploadPart** request for each object part upload. The best part of this is that if there are any hiccups such as network errors, then you can restart the failed upload of that individual part instead of the entire upload.
4. **Commit the upload**—Lastly, when every part is uploaded, this engine will complete the multipart upload by committing it, adding a bullet on the checksum, and so on.

**Summary**

In this chapter, we have learned the various different options for storing your application data in OCI. OCI not only provides best-in-class storage options but also offers various flavors of storage to have every aspect of your application's storage requirements satisfied. So, you can either choose high-performance block storage or super-elastic file storage to keep your application data very local to your instances. You can also choose the Object Storage tier for your unstructured data or the Archive Storage tier to keep your backup costs under control.

In the next chapter, you will see which database choices you have.