Dremio Software

Disaster Recovery - Principles and Best Practices

Introduction

Disaster recovery is fundamental for any business to deliver a reliable data platform. Data platforms are critical since they provide the ability for decision support, innovation, retrospective analysis, optimization, etc. It is therefore essential that even in the rare cases when disasters occur (that are sometimes beyond our control), platforms remain stable and operational, and more fundamentally, the business experiences minimal disruption.

A platform such as Dremio is a core component in any enterprise data strategy, where many services, applications and (or) users depend on it daily, whether that be providing timely and accurate insights to support decision-making processes or just delivering the ability to perform ad hoc/what-if analysis based on past or current events. It is therefore paramount that during any such disaster, Dremio can recover promptly to mitigate and minimize the impact on the business.

This article provides an understanding of the core concepts of disaster recovery, how these translate to Dremio capabilities and best practices for delivering an effective disaster recovery solution for your data platform.

Disaster Recovery Overview

Disaster Recovery (or DR) often involves a set of policies, principles, tools and procedures that enable the recovery and continuation of service following a disaster, whether human-induced or otherwise. This is typically a program funded by IT leadership that often goes through continuous improvement cycles to ensure that it meets the ongoing needs of the business and the changing data landscape. The program is designed to implement an effective strategy and a recovery plan that is regularly tested and reviewed to ensure it aligns with the evolving enterprise data strategy.

DR is organization-centric; that is, the plan should be specific to an organization and it should be based on two key metrics, these are:

1. Recovery Time Objective (RTO) is the duration and service level within which the service availability is restored after a disaster to avoid the consequences of the break in continuity.
2. Recovery Point Objective (RPO) explains how much time can elapse upon an outage before the quantity of data lost exceeds an acceptable threshold, often called a tolerance. Another way of thinking about this is how old data can be *after* the application/service has recovered.

Both targets are measured in time, with lower numbers representing less downtime and less data loss. As depicted in Figure 1.0, the smaller the window between the RPO and the disaster, the less data is lost. Similarly, the smaller the window between the RTO and the disaster, the less downtime is experienced. Typically, an organization would want to minimize both windows, but in reality, this might not be necessary. Reducing these windows can mean they spend a lot of time, resources, effort, and money on lowering these metrics, but in reality, they achieve minimal benefits. Not to mention the additional complexity involved in implementing and maintaining these solutions. Therefore, RTO and RPO must be application/service-specific, realistic values and evaluated with cost and complexity in mind.

RPO vs RTO

How much data can you afford to recreate or lose?

Event/Disaster

How quickly must you recover? What is the cost of downtime?

Time

Data Loss

How far to go back?

Downtime

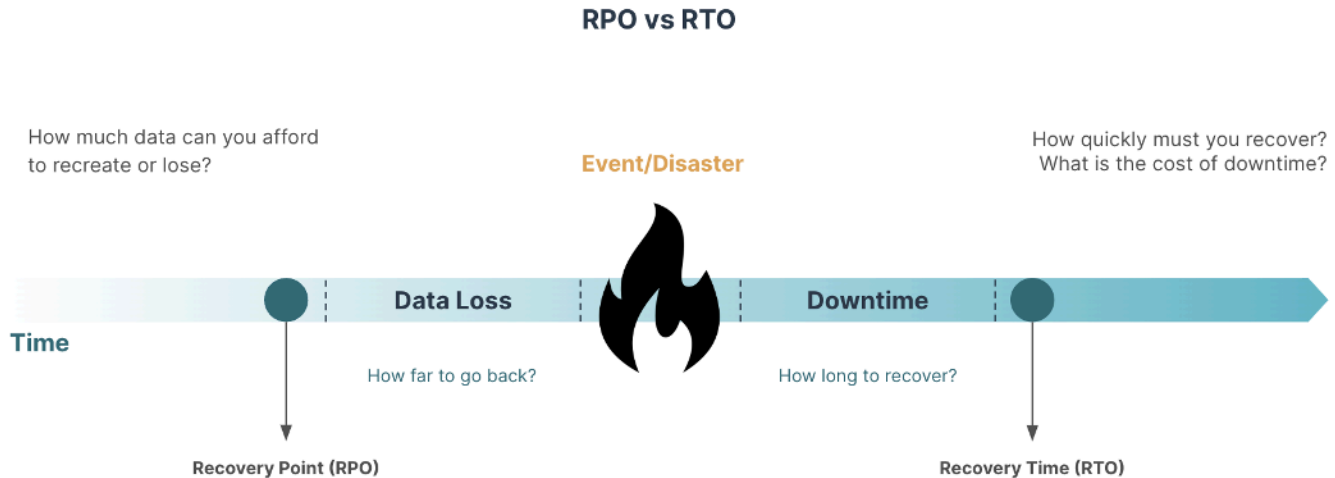How long to recover?

Recovery Point (RPO)

Recovery Time (RTO)

Figure 1.0 RTO vs RPO

In the context of Dremio, we can define RPO as the maximum amount of time allowed for jobs to complete, for example reflection refreshes, metadata refreshes etc, which can sometimes be as important as the data itself.

> **ℹ️ NOTE**
> This could be especially useful when we shift left, moving away from ETL towards EL+T, leveraging the Dremio semantic layer to perform transformations on your data.

Although Dremio does not store your data, it can be responsible for creating/managing data inside your object storage; an example would be DML and DDL operations for Apache Iceberg. Furthermore, in some cases, we might have to factor in concerns about where a region goes down and where the S3 bucket that hosts your data is located. In this case, we should consider something similar to cross-region replication (see CRR for details).

A high-level workflow for DR inside Dremio would play out as follows:
1. A failure occurs in your primary production environment, whereby the coordinator is unreachable.
2. You investigate what might be causing this by performing troubleshooting across your infrastructure and Dremio logs.
3. If you conclude that this issue cannot be resolved inside your primary deployment in time with your RTO/RPOs, you can failover to a secondary deployment.
4. Before you begin failover, you confirm that the same issue does not reside in the secondary environment.
5. You begin Failover to your secondary environment.

a. Stop all jobs/workflows that are working inside your primary environment.
b. Start the recovery procedure of your secondary environment. This involves restoring the latest backup from your primary environment.
c. After successfully restoring and testing the secondary environment, you can declare this as your new primary and production workloads can be routed to it, allowing users to log on as usual and run their jobs.

6. Once the issue is resolved in your primary environment, you can restore to this or keep Dremio where it is. If you fail back to this environment, you can follow a similar process to the failover. However, this time, you should back up the whole environment to extract anything missing between the last backup and when the old environment became ready to restore.

Disaster Recovery Configurations

There are two main configurations of disaster recovery, these are:
1. **Active-active** - two or more environments are deployed in multiple sites but work simultaneously. During a disaster impacting the primary environment, workloads will be diverted to the secondary environment immediately, yielding no downtime.
2. **Active-passive** - two or more environments are deployed in multiple sites but are not operating simultaneously; one is kept idle (or shut down). In the case of a prolonged outage, the passive environment will be activated. Often, this required restoring backups taken from the primary site.

Whichever configuration is chosen as the correct one for your business, you should consider the length of disruption, the effort required to failover and, if relevant, the effort to restore the primary environment. The following graphic shows the different strategies within each of the configurations mentioned.

> **ℹ️ NOTE**
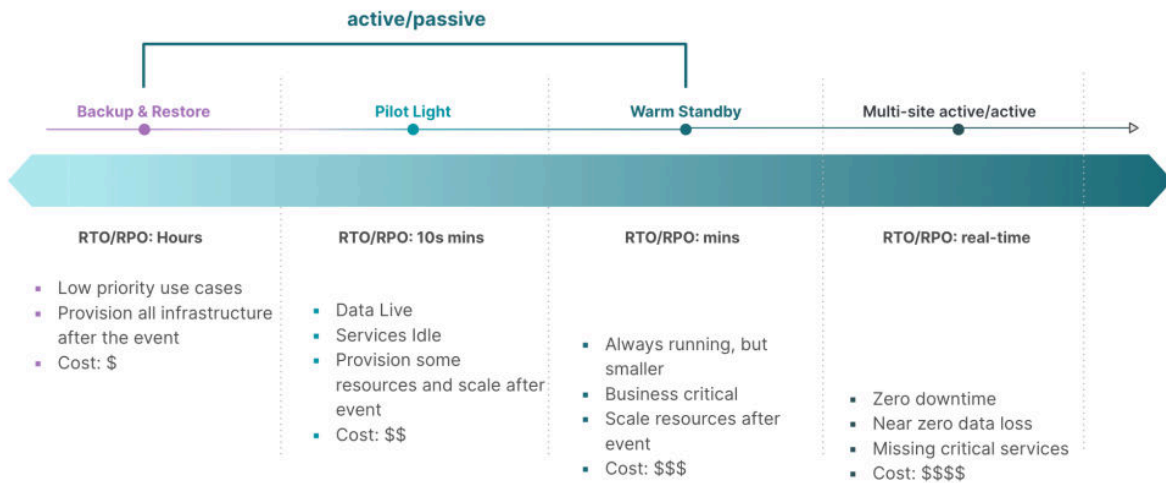> Dremio does not currently support Active-Active in Software or Cloud.

Figure 2.0 - DR Strategies

Active-passive

Active-passive is typically the most common method of implementing disaster recovery. It comprises many strategies, each increasing in complexity and cost, starting with the basic backup & restore and moving to warm standby. In the context of Dremio, the strategies can be defined as:

- Backup & restore - the basic level of disaster recovery that any organization should consider. It provides the ability to recover based on the latest backup, where the backup could be at a chosen periodicity, e.g. daily. Typically, this would involve creating a Continuous Delivery (CD) process that automates infrastructure deployments and performs a full system restore. The backup can be used to restore the primary artefacts of Dremio, such as metadata, view definitions, sources, spaces, RBAC etc.
- Pilot-light - the data is live and available, but the services are idle. What this means is that the data sources are available and up-to-date (or almost up-to-date). However, the engines are unavailable, so they cannot serve queries immediately.
- Warm standby - Dremio can serve queries immediately, although this is at a reduced capacity. This means it cannot support production workloads immediately, but this can be achieved by scaling up the engines.

> **ℹ NOTE**
> Warm standby can be achieved with Dremio Cloud using Arctic, where Dremio objects are stored inside an Arctic Catalog (please refer to Arctic for details). This Catalog can be used by multiple environments (within the same region), and any environment with access to this catalog will have immediate access to all its Tables and Views.

Implementing DR Strategies

To support customers on their DR strategy, Dremio has created several tools that can be used to help mitigate the risks associated with disasters. This section demonstrates how these tools can help customers across different deployment methodologies.

Before we go any further, we would like to introduce those tools.

1. **Dremio Admin** - this tool allows you to perform a backup and restore of your Dremio environment; it takes a full system backup that includes metadata, uploaded files, sources, Views and Tables definitions, users, roles and many more. The tool does not backup cache, downloaded files and query results.
2. **Dremio Cloner** - this tool was developed by Dremio Professional Services to enable the export and import of Dremio artefacts (the artefacts do not include passwords or uploaded files). It can synchronize Dremio's internal artefacts and objects, such as sources, tables, views, roles and groups across environments.

> **ℹ NOTE**
> As of version 24.3, Dremio Cloner cannot replicate User Defined Functions (UDFs) or row and column security policies.

Dremio has multiple deployment methods; the following table provides an overview of Dremio support for disaster recovery strategies across these models. In summary, backup & restore are provided out of the box, however if you need to implement pilot-light or warm standby, then the costs and complexity of the environment will increase (e.g. additional people, infrastructure and processes required to fulfil the expectations for the strategy).

	Backup & Restore	Pilot-light	Warm standby
VM's	✓	$	$$$
Hadoop	✓	$	$$$
Kubernetes	✓	✓	✓
Cloud	✓	✓	✓

Table 1.0 - Dremio support for DR Strategy

Disaster Recovery for Dremio Software

This section covers how we can implement disaster recovery within Dremio software, including Kubernetes, VMs, and YARN deployment methods. The remainder of this document will focus on how we can deliver disaster recovery for each software deployment method. These are split into two separate categories, Kubernetes and Non-Kubernetes.

In the case of Dremio Cloud, disaster recovery is covered as part of our SaaS offering on the Dremio Cloud Control Plane, which Dremio manages. For the data plane (customer's cloud subscription), customers should configure multiple subnets across multiple availability zones (AZs) to mitigate AZ failure in AWS. In Azure, VNets support multiple AZs as standard. The project store has tolerance to AZ availability.

Backup & Restore for Dremio Software

As a minimum, any DR process for an organization should cover backup & restore. This provides the essential ability for an organization to recover Dremio from the latest backup in case of a failure. It gives the basic assurance that Dremio can become operational and that the risk to business continuity can be minimized after an event.

For this approach, we make the following assumptions about your environment:

- You have deployed a secondary Kubernetes cluster that is configured in a different environment and this cluster is using the same Dremio version as your primary cluster.

> ℹ️ NOTE
> You can only restore from a backup using the same Dremio version used to create the backup.

Backup

This section describes the method for successfully performing a backup of your Dremio environment. Before we begin, Dremio must be online.

The following commands are only required for Kubernetes deployments.

Kubernetes - Dremio Software

1. Connect to the master-coordinator pod using the exec command as per below.

   ```
   kubectl exec -it dremio-master-0 -- bash
   ```

2. On the master-coordinator pod, perform a backup using the following command in the bash shell. See [Backup Dremio](#) for more details.

   ```
   /opt/dremio/bin/dremio-admin backup -d <BACKUP_PATH>
   ```

3. Copy the backup files outside the pod using the following command.

   ```
   kubectl cp dremio-master-0:<BACKUP_PATH> /dremio-backup/
   ```

The following commands are only required for non-Kubernetes deployments: Standalone, Docker, VMs and YARN.

The following must be confirmed before continuing:

- The Dremio daemon is running
- The backup path is writable by the user running the Dremio daemon.

Non-Kubernetes - Dremio Software

1. Connect to the master-coordinator node of your cluster.

2. On the master-coordinator node, perform a backup using the following command in the bash shell. See [Backup Dremio](#) for more details.

   ```
   /opt/dremio/bin/dremio-admin backup \
   -u <DREMIO_ADMIN_USER> -p <DREMIO_ADMIN_PASS> -d <BACKUP_PATH>
   ```

Restore

This section describes the steps to restore a Dremio environment successfully. Before we begin, please do the following:

- On the master-coordinator node, create a copy of <DREMIO_LOCAL_DATA_PATH>. Check the default location for your deployment method.
- On the master-coordinator node, delete the contents of <DREMIO_LOCAL_DATA_PATH> and then create an empty directory called db that is readable and writable by both the user running the restore and the dremio user.

The following commands are only required for Kubernetes deployments.

> **Kubernetes - Dremio Software**
>
> 1. Connect to the master-coordinator pod using the exec command as per below.
>
>    ```
>    helm upgrade --wait <chart release name> . --set DremioAdmin=true
>    ```
>
> 2. Copy the backup files created in the backup to your Dremio Admin pod.
>
>    ```
>    kubectl cp /dremio-backup/ dremio-admin: <BACKUP_PATH>
>    ```
>
> 3. Connect to the Dremio Admin pod using the following exec command.
>
>    ```
>    kubectl exec -it dremio-admin -- bash
>    ```
>
> 4. Perform a restore of Dremio using the following command, where BACKUP_PATH is the location of your backup files.
>
>    ```
>    /opt/dremio/bin/dremio-admin restore -d <BACKUP_PATH>
>    ```
>
> 5. Delete the admin pod and clean up.
>
>    ```
>    helm upgrade --wait <chart release name> . --set DremioAdmin=false
>    ```
>
> 6. After performing the offline command, you can restart your Dremio cluster.

The following commands are only required for non-Kubernetes deployments.

> ### Non-Kubernetes - Dremio Software
>
> 1. Shutdown Dremio service on all nodes of your cluster using [Start/Shutdown](#).
> 2. Connect to the master coordinator node of your cluster.
> 3. Run the following command located in `<DREMIO_HOME>/bin/`
>
>    ```
>    ./dremio-admin restore -d <BACKUP_FOLDER_PATH>
>    ```
>
> 4. Look for the confirmation message as per below.
>
>    ```
>    ...
>    Restored from backup at /tmp/dremio_backup_2017-02-23_18.25, dremio
>    tables 14, uploaded files 1
>    ```
>
> 5. Validate the restore completed by performing some functional and UAT tests against your sources from different clients, including the Dremio UI, JDBC etc.

Pilot-light for Dremio Software

As discussed previously, the pilot-light strategy is primarily designed to keep the lights on or the platform operational. If an event occurs in your primary environment, this strategy can be used as a failover. It is worth noting that queries can only be run against the Preview engine; if you want to run large queries or even operational workloads, you must start up and scale out the other engines.

For this approach, we make the following assumptions about your environment:

- You have deployed a secondary Dremio cluster configured in a different environment using the same Dremio version as your primary cluster.
- Dremio engines and Workload Management (WLM) have been configured to replicate the primary cluster.
- dremio-cloner has been downloaded and is ready to use.

> **ℹ NOTE**
> Dremio will only be able to accept queries when engines have been started and scaled accordingly.

The core principles of the pilot-light strategy with Dremio are:

- Dremio will have the latest possible state (based on the latest backup).
- Dremio artefacts, including users, roles, sources, views and tables, will be synchronized between the primary and secondary production environments.
- Dremio cannot accept queries immediately; engines must be started.

The first element of the Pilot-light strategy is to ensure that the latest backup is retrieved, and the secondary environment is based on this. This can be achieved by following Backup & Restore. Once we have the latest backup, we must ensure that Dremio is fully synchronized with the primary environment. To do this, we will use a combination of dremio-admin and dremio-cloner. The following steps will guide you on how to synchronize the two environments. In summary, we will extract artefacts and objects from your primary environment and import them into your secondary.

1. Periodically, based on your RPO requirements, dremio-admin will be used to perform a backup of the Dremio KV store, which contains Dremio metadata, including view definitions, source settings, space, RBAC, UDF's and local user accounts.
2. Copy the backup to your secondary environment. Do this step as frequently as the backup is performed on the primary environment.
3. Stop the secondary environment and clear the KV store directory.

4. Restore the secondary environment based on the backup following the steps in [Restore](#).

5. Start the coordinator inside your secondary environment.

6. Configure the dremio-cloner "GET" function based on your requirements. Details of this can be found [here](#). Using the dremio-cloner "GET" command, we can extract Dremio artefacts/objects and output these into a JSON file for later use. The following command can be used to run the export.

```
python dremio_cloner.py [get_config_file.json]
```

7. Once we have successfully exported the objects, we can use the "PUT" command to import the objects into your secondary environment. Details of the configuration for this can be found [here](#). It is worth noting that at this stage, you can import all or select specific objects that you want to, this can be done using the filters configured in the JSON config file. Once you are happy with the configuration, you can run the import using the following command.

```
python dremio_cloner.py [put_config_file.json]
```

8. Validate that the objects have been successfully migrated between the environments.

This approach should be automated and incorporated within your existing CI/CD processes to ensure that the two environments are synchronized regularly, e.g. hourly.

Warm Standby for Dremio Software

The warm standby approach is similar to pilot-light, but the environment is ready to accept queries, albeit limited. In the case of an event/disaster with a prolonged outage in the primary environment, this strategy can be used as a fail-over to accept some workloads and prevent further disruptions to your users. Sometimes, a region might become unavailable, so organizations might need to move operations to a different site/region.

This warm standby environment can be used as a fail-over to accept workloads based on the criticality to the business, which could be determined based on how important they are for core operations in your organization; having some functional workloads is considerably better than having none at all. However, you must scale up the engines to match your primary environment to accept production workloads. This will come at an additional cost but for a limited time window. In summary, warm standby means you only need to scale the engines and not build the environment from scratch. Therefore, it will take considerably less time to get up and running again. There is a cost/availability tradeoff that you will need to consider.

For this approach, we make the following assumptions about your environment:

- You have deployed a secondary warm standby Kubernetes cluster configured in a different environment using the same Dremio version as your primary cluster.
- Dremio engines and Workload Management (WLM) have been configured to replicate the primary cluster.

> ℹ️ NOTE
> This approach will not be able to support production workloads immediately; it will only be able to support a subset. If you require production workloads, you must scale up the engines accordingly.

The core principles of the warm standby approach are very similar to pilot-light, but they allow the execution of small queries immediately. See below.

1. Dremio will have the latest possible state (based on the latest backup).
2. Dremio artefacts, including users, roles, sources, views and tables, will be synchronised between the primary and secondary production environments.
3. Dremio will be able to accept queries immediately but not production workloads. To enable this, you will need to scale up the engines accordingly.

To deliver a warm standby DR strategy, we must follow the same process described in Pilot-light. However, we must prepare the engines to ensure they can accept queries

immediately. The following discusses how to achieve this for different deployment methods.

The following is only relevant for Kubernetes deployments.

Kubernetes - Dremio Software

1. Provision the nodes that will act as executors for the engines based on your requirements (or simply increase the size of your node pool if using Cloud Kubernetes, e.g. EKS, AKS or GKE deployments).
2. Configure the engines within the helm chart. See below for an example configuration inside your values.yaml file.

```
executor:
  # CPU & Memory
  # Memory allocated to each executor, expressed in MB.
  # CPU allocated to each executor, expressed in CPU cores.
  cpu: 15
  memory: 122800
  engines: ["default", "bi", "datascience", "ad-hoc", "reflections"]
  count: 3
```

The following is only relevant for non-Kubernetes deployments.

Standalone/VMs - Dremio Software

1. Provision the physical nodes/VMs that will act as executors for Dremio.
2. Download, install and configure Dremio on the nodes following dremio-deployment.

The following is only relevant for YARN deployments.

YARN - Dremio Software

1. Allocate the nodes for the engines in YARN.
2. Download and install Dremio on the nodes following yarn-deployment.
3. Set up the engines using the Dremio UI. Please refer to configure-engines for details.

References

- https://aws.amazon.com/blogs/mt/establishing-rpo-and-rto-targets-for-cloud-applications/
- https://aws.amazon.com/blogs/architecture/disaster-recovery-dr-architecture-on-aws-part-i-strategies-for-recovery-in-the-cloud/
- https://docs.oracle.com/en/solutions/oci-pilot-light-dr/index.html#GUID-3C1F7B6B-0195-4166-A38C-8B7AD53F0B79
- https://docs.dremio.com/software/advanced-administration/restore/#options
- https://docs.dremio.com/software/deployment/azure-aks/aks-admin/#backup
- https://github.com/deane-dremio/dremio-cloner