With MapR version 5.0 being released recently, MapR customers got yet another powerful feature at no additional licensing costs: Auditing! In this two-folded blog post, I'll describe various use cases for auditing as well as instructions for how to deploy these cases in your MapR environment.

The auditing features in MapR let you log audit records of cluster administration operations as well as operations on directories, files, and tables. Besides the obvious use case of finding out who is doing what on your cluster, the auditing data also enables other use cases which I will outline in this blog post. In a subsequent blog post, I'll describe how to set up Apache Drill for querying audit logs.

The topics covered in this blog are related to the following five use cases, including the added value they deliver, as well as hands-on descriptions of how to deploy them. Let's start with summarizing the use cases:

1. Unauthorized cluster changes and data access
2. Complying with regulatory frameworks and legislation
3. Data usage heatmaps on cold, warm and hot data
4. Data access analytics and performance improvements
5. Data protection policies on data that matters using snapshots and mirroring

Introducing auditing

The powerful new auditing capabilities in MapR 5.0 allow you to log cluster usage and changes at various levels. First of all, any administrative tasks performed at the cluster can be logged, regardless of the methodology used to administer the cluster. Secondly, the access to data at volume, directory, file and table levels can be enabled to allow auditing of data access at a granular level.

By enabling auditing at the cluster administration level, every action performed via the MapR Control Service (MCS), MapR Command Line Interface (maprcli) and REST API gets logged. Also, every attempt at authentication to MCS, whether successful or unsuccessful, is logged.

Since all audit logging is written in the JSON file format, the power of Apache Drill (https://drill.apache.org/) allows you to query through this information using ANSI SQL, which means that you can use any of your favourite Business Intelligence tools to visualize the audit information. In a subsequent blog, I'll describe the setup of Apache Drill to query the audit logs.

In this blog, I'll explain how to enable the various levels of auditing. But first, let's take a look at the powerful use cases that the MapR auditing feature enables:

Auditing in MapR: 5 Business Use Cases

1. **Unauthorized cluster changes and data access**
   Enabling audit logging at the cluster level provides great value when it comes to who changes what from an administrative point of view (http://doc.mapr.com/display/MapR/Auditing+of+Activity+Related+to+Cluster+Administration) on the MapR cluster. Since it also logs both successful and unsuccessful login attempts on the MapR cluster, it provides insight into people trying to get unauthorized access to the cluster based on the amount of failed login attempts. In parallel, logging operations at the filesystem and table level (http://doc.mapr.com/display/MapR/Auditing+of+Filesystem+Operations+and+Table+Operations) allow cluster administrators to answer the following important questions:
   - Who touched customer records outside of business hours?
   - What actions did users take in the days before leaving the company?
   - What operations were performed without following change control?
   - Are users accessing sensitive files from protected or secured IP addresses?

2. **Complying with regulatory frameworks and legislation**
   The auditing information is very useful when your company needs to comply with strict regulations and legislations. In case of a compliance audit by an external independent accounting firm, the audit log files provide all necessary information on who did what on the cluster. This greatly reduces the time an external audit takes, and therefore saves your company valuable time and money. It also increases the level of your organization's maturity level while protecting your brand from reputation damage by unauthorized data access.
3. **Data usage heatmaps on most and least frequently used data**
   Since the data access audit logs are written in the JSON file format, it's easy to generate data usage heatmaps using your favourite Business Intelligence tool and Apache Drill. Creating these kinds of heatmaps on cold, warm and hot data allows you to advise your users on important and valuable data,as well as on rarely used and unexplored data.
4. **Data access analytics and performance improvements**
   The heatmaps I described above are also important information for cluster data layout planning. One use case can be that end users are experiencing a decrease in performance on a data volume. When creating the initial cluster layout, the administrator might have set a volume's replication level to the standard of three replicas (this means there are three cluster nodes able to serve the data in case of an end-user request). Although the MapR Container Location Database (CLDB) spread the load across the cluster to remove hotspots, the amount of data requests could be too high for the replication level set. Based on actual data usage as shown in the

heatmaps (either by end user, mapreduce jobs, Spark Streaming applications or others), it might be beneficial to increase the replication level to increase data locality, and have more nodes able to service this specific data volume. This will greatly improve end-user experience on data access performance.

5. **Data protection policies on data that matters using snapshots and mirroring**
   To make sure that your company's crucial data is safe, the heatmaps can be of great assistance when reviewing your snapshot and mirroring policies as currently deployed on the cluster. Data volumes that have many update and delete activities going on are candidates for a high frequency snapshot policy. This is to avoid disaster in case important data gets accidentally overwritten or deleted by end-user mistakes. Additionally, you should review the mirroring of these frequently used data volumes to make sure this data is available in a secondary location. That way, your company will be able to continue operating, even if you experience a disaster in your primary data center.

Enabling Auditing Use Cases on Your MapR Cluster

In this section, I will show you how to benefit from using the above use cases on your MapR cluster. First, we'll start with enabling audit logging at the MapR cluster level. After that, we will enable auditing at the volume, directory, file and table level to capture all data access. Please note that enabling auditing uses your cluster's capacity (both Linux storage as well as MapR filesystem storage) to write the audit logs. By using the 'retention' and 'maxSize' parameters, you can limit the maximum audit log files stored at disk.

First, let's check if the auditing feature is enabled by executing the following maprcli command, which should return "1":

```
maprcli config load -json | grep "mfs.feature.audit.support"

output example:
auditing enabled: "mfs.feature.audit.support":"1"
auditing disabled: "mfs.feature.audit.support":"0"
```

If you upgraded from previous versions of MapR to version 5.0, note that auditing is not yet enabled. You can enable auditing using the following command:

```
maprcli config save -values {"mfs.feature.audit.support":"1"}
```

Enabling auditing at cluster administration level

Let's start with enabling auditing at the cluster level to capture all administrative tasks performed at the cluster. Execute the following CLI command to enable cluster administration auditing:

```
maprcli audit cluster -enabled true
```

Immediately after enabling cluster level auditing, cluster management activities, including both MCS and MapR CLI commands, are written to the following log files:

```
/opt/mapr/logs/initaudit.log.json
/opt/mapr/logs/cldbaudit.log.json
/opt/mapr/mapr-cli-audit-log/audit.log.json
```

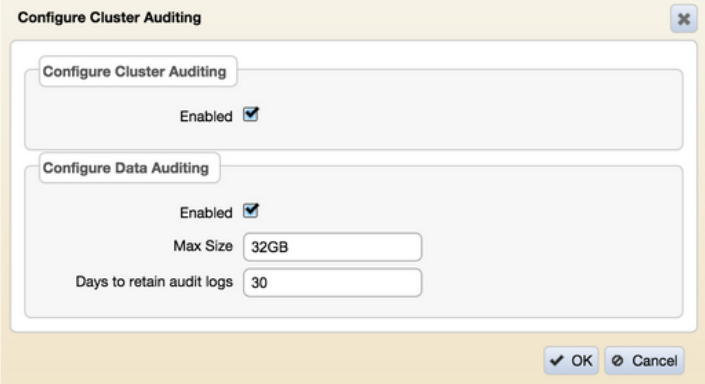
To review your cluster auditing settings, take a look at the 'cluster' object in the output of the following command:

```
maprcli audit info -json

output example:

{
    "timestamp":1437568253988,
    "timeofday":"2015-07-22 12:30:53.988 GMT+0000",
    "status":"OK",
    "total":1,
    "data":[
        {
            "data":{
                "enabled":"1",
                "maxSizeGB":"500",
                "retentionDays":"180"
            },
            "cluster":{
                "enabled":"1",
                "maxSizeGB":"NA",
                "retentionDays":"NA"
            }
        }
    ]
}
```

Besides using the MapR CLI, you can use the MCS to enable cluster management auditing. After signing on to the MCS, use the menu navigation on the left to browse to *'System Settings > Auditing'* and enable *'Configure Cluster Auditing'* in the screen as shown:



Enabling auditing of Directory, Files and Table operations

In contrast to auditing cluster-level operations, auditing of file system and table operations needs to be enabled at three separate levels. If auditing is not enabled at any one of these levels, operations on a filesystem object are not logged.

Data auditing should be enabled at three levels: the cluster level (through the maprcli audit data command), the volume level (through any of the three maprcli volume commands), and the level of the individual directory, file, or table. The following shows the hierarchy in enabling or disabling the auditing at any level of the cluster. Directory, file or table level auditing (item 3) will only be enabled if both volume level auditing (item 2) and cluster level auditing (item 1) are enabled:

```
1.  Cluster level auditing (enabled/disabled)

2.  - Volume level auditing (enabled/disabled)

3.  - Directory, file, table level auditing (enabled/disabled)
```

Data Auditing - Step 1: cluster level

Let's start with enabling data auditing at the cluster level (item 1). You can think of the data auditing parameter as the master switch for enabling or disabling audit logging at the data level for the complete cluster:

```
maprcli audit data -enabled true
-maxsize <gb>-retention <numberofdays>example:
maprcli audit data -enabled true -maxsize 500 -retention 180</numberofdays></gb>
```

By using either one or both of the parameters ('maxSize' and/or 'retention'), it is possible to maximize the audit log file size. When setting both parameters it depends which limit comes first (e.g., the size of the audit log in GB or the retention period in days) that forces the cleanup of the log automatically by the MapR cluster.

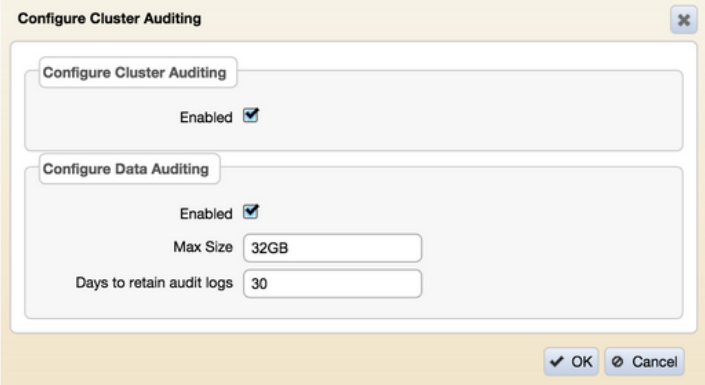
To review your data auditing settings, have a look at the 'data' object in the output of the following command:

```
maprcli audit info -json

output example:

{
    "timestamp":1437568253988,
    "timeofday":"2015-07-22 12:30:53.988 GMT+0000",
    "status":"OK",
    "total":1,
    "data":[
        {
            "data":{
                "enabled":"1",
                "maxSizeGB":"500",
                "retentionDays":"180"
            },
            "cluster":{
                "enabled":"1",
                "maxSizeGB":"NA",
                "retentionDays":"NA"
            }
        }
    ]
}
```

Besides using the MapR CLI, you can use the MCS to enable to data auditing at cluster level. After signing on to the MCS, use the menu navigation on the left to browse to *'System Settings > Auditing'* and enable *'Configure Data Auditing'* in the screen as shown:



Data Auditing - Step 2: volume level

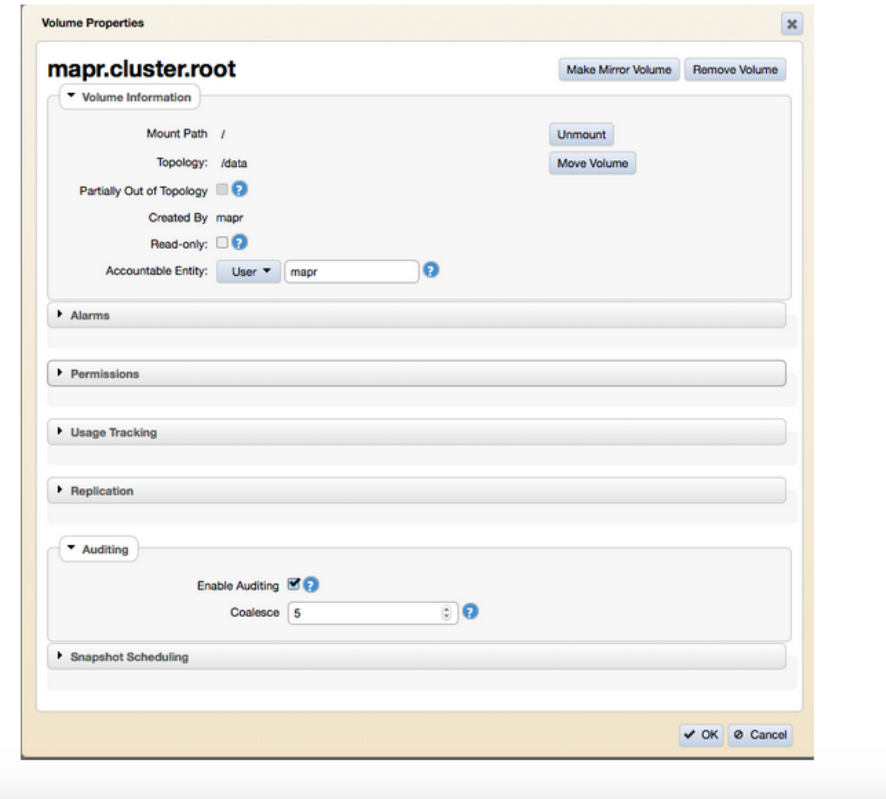
Now that we have the data auditing enabled at cluster level (item 1), let's take the next step by enabling auditing at volume level (item 2). There are three different ways to enable auditing at volume level, depending if the volume already exists or not. Besides the 'maprcli volume create' and 'mapcli volume modify' command, there is the 'maprcli volume audit' command, which I'll be using in this blog. All three maprcli commands have the same effect.

```
maprcli volume audit
[ -cluster cluster_name ]
-name volumeName
[ -enabled <true|false>]
[ -coalesce interval in mins ]

example for volume finance-project1:
maprcli volume audit -name finance-project1 -enabled true -coalesce 5</true|false>
```

By using the volume audit MapR CLI command, you can both enable and disable auditing at existing volumes. Additionally, you can specify the coalesce parameter which allows you to configure that identical activities taking place in the specified time interval will be stored as one record in the audit log. For example, when setting coalesce to 60 minutes (which is the default), all identical activities taking place within this timeframe will result in only a single record in the audit logs.

Enabling auditing at the volume level can also be done using the MCS. Go to 'MapR-FS > Volumes' and either edit an existing volume or create a new volume to set the auditing:



Data Auditing - Step 3: directory, file or table level

With data auditing enabled at both the cluster and volume level, it's time to enable auditing at specific directories, files or tables. This can only be done using the hadoop mfs setaudit command as shown below:

```
hadoop mfs -setaudit on|off <dir|file|table>

example to activate the auditing on a directory, file or table:
hadoop mfs -setaudit on /full/path/to/directory
hadoop mfs -setaudit on /full/path/to/file
hadoop mfs -setaudit on /full/path/to/table
```

After enabling auditing, please verify the audit bit has been set on your directory, file or table using:

```
hadoop mfs -ls /

drwxr-xr-x Z U U   < auditing disabled, last bit being 'U'
drwxr-xr-x Z U A   < auditing enabled, last bit being 'A'
```

With that, you're all set and every activity on your cluster is being audited and logged to the MapR file system at:

```
/var/mapr/local/<node_name>/audit/.</node_name>
```

If you have NFS setup and the MapR cluster mounted, you can find the logs at:

```
/mapr/<cluster_name>/var/mapr/local/<node_name>/audit/.</node_name></cluster_name>
```

When operations are performed on directories, files, or tables that are being audited, the full names for those objects, as well as the current volume and the name of the user performing the operation, are not immediately available to the auditing feature. What are immediately available are IDs for those objects and users. Converting IDs to names at run-time would be costly for performance. Therefore, audit logs contain file identifiers (FIDs) for directories, files, and tables; volume identifiers for volume; and user identifiers (UIDs) for users.

You can resolve identifiers into names by using the ExpandAuditLogs utility. This utility creates a copy of the log files for a specified volume, and in that copy are the names of the filesystem objects, users, and volumes that are in the audit log records. You can then query or process the copy.

Conclusion

With MapR auditing released in version 5.0, you have yet another powerful feature to manage your cluster and data needs. Besides the obvious cases of who is doing what from a cluster management and data usage point of view, MapR auditing also allows various other use cases when it comes to compliance, performance, and data protection policies.

In my next blog post, I'll show you how to setup Apache Drill to query through the audit logs being generated.

If you have any further questions, please ask them in the comments section below.

Want to learn more?

- MapR Auditing (http://doc.mapr.com/display/MapR/Auditing+of+Cluster+Administration+and+Operations+on+Directories%2C+Files%2C+and+Tables)
- MapR Security (http://doc.mapr.com/display/MapR/Security+Guide)
- MapR Volumes (http://doc.mapr.com/display/MapR/Managing+Data+with+Volumes)
- Apache Drill (https://drill.apache.org/)
- Download mapR (/products/hadoop-download)
- Top 10 reasons customers choose MapR (/top-ten-reasons)

This blog post was published October 13, 2015.