

Audit and Assurance Program for NoSQL DBMS

Co-authored by

Kanwarpreet Singh Brar

Student ID #: 139410

Research Author

kbrar3@student.concordia.ab.ca

Hitesh Bhola

Student ID #: 143369

Research Author

hbhola@student.concordia.ab.ca

Dr. Sergey Butakov

Research Advisor

sergey.butakov@concordia.ab.ca

Project Report

Submitted to the Faculty of Graduate Studies,
Concordia University of Edmonton

in Partial Fulfillment of the
Requirements for the Final
Research Project for the Degree

**MASTER OF INFORMATION SYSTEMS SECURITY
MANAGEMENT**

Department of Information System Security and Assurance
Management

Concordia University of Edmonton

FACULTY OF GRADUATE STUDIES

Edmonton, Alberta

April 2021

Abstract

This research outlines a template of audit and assurance program for NoSQL DBMSs based on the COBIT framework. With the increase in the use of NoSQL technologies in enterprises, there are various technical and managerial risks associated with them. Some of the major technical risks include lack of data encryption, built-in user authentication and authorization, vulnerability to the injection attacks and technical concerns related to data redundancy features. Managerial risks include limited expertise available for this new technology, complex data migration to NoSQL solutions, issues related to potential data inconsistencies and limited tools for transaction audit. To handle such risks, it is important for an enterprise to implement rigorous controls to mitigate those risks effectively. And there must also be a mechanism to test the effectiveness of the implemented controls. A rigorous audit and assurance program, designed specifically for NoSQL DBMS, could serve as such a mechanism. As a part of this research, a sample audit and assurance template has been developed considering Elasticsearch as the primary database for testing purposes. This sample also includes the pictorial reference documents for the technical controls covered in the audit program.

Keywords - Database Audit, Information Assurance, Risk Management, COBIT and NoSQL.

Table of Contents

Introduction	1
NoSQL Technology	1
Architectural View of NoSQL DB	1
Key-Value Stores	1
Wide Column Store	1
Document Store Model	1
Graph Store Model	2
Review of existing NoSQL DBMS	2
MongoDB	2
Cassandra DB	2
CouchDB	2
Elasticsearch	2
Risk Analysis	3
Auditing	6
COBIT Cross Reference	6
Example of alignment of enterprise goals using COBIT-2019	9
Conclusion	10
Bibliography	10

List of Tables

Table 1 Comparison of Security Features in various NoSQL Databases.....	4
Table 2 Managerial Risks Related to NoSQL Technologies	4
Table 3 COBIT-2019 Enterprise Goals (ISACA, 2019)	6
Table 4 COBIT-2019 Mapping with Test Cases (ISACA, 2019)	9

List of Figures

Figure 1 Snippet of Audit and Assurance Spreadsheet Showing Use case UC3.2	9
Figure 2 Snippet of Audit and Assurance Spreadsheet Showing Use Case UC5.1 along with the Pictorial Reference Document.	10

Introduction

NoSQL approach is based on storing the data in an unstructured format instead of the row and column approach, followed by relational databases. The data is generally stored in the JSON document format. Unlike the conventional database management systems, NoSQL systems are distributed in nature. In the mid-90s, there had been a massive change in data generation and processing. With the increase in internet usage, a huge volume of data is getting generated, and the conventional database management system were not capable to cope as they were designed to operate on a single server. The only way to manage such a considerable amount of data was scale them vertically, i.e., upgrade the server memory, processing power, etc. (Couchbase.com, n.d.) This approach lacked scalability. In 1998, a distributed data management system was engineered by Carlo Strozzi named Not only SQL (NoSQL). Initially, it was developed in response to the requirement of processing the unstructured data. Since it is schema-less and distributed in nature, a group of many systems work together to share the load of user's requests and removes the limitation in handling the big data. Many IT organizations such as Facebook, Twitter, Google are using the NoSQL approach to manage the data. (Foote, A Brief History of Non-Relational Databases, 2018) (Hossain & Moniruzzaman, 2013) (Kumar, 2018)

As a result of the differences between the data handling behavior of NoSQL databases and relational databases, the risks associated with NoSQL technologies differ from those of the relational databases. Recent reports showed that data present in many Elasticsearch instances had been compromised. These risks could be segregated into two categories: managerial and technical risks, based on their impact domain in the enterprise. To control such risks, it is important for an organization to implement certain controls and mechanisms so that the enterprise could effectively guard their NoSQL database. Even after having such controls in the system, it is important to ensure the efficiency and controllability of them in handling all the major risks. To conduct such testing, there is a need for a rigorous audit and assurance program covering all the domains of risk. The Audit program is aligned with the COBIT framework in which the use cases are designed in such a way that they meet the governance and enterprise objectives.

NoSQL Technology

Architectural View of NoSQL DB

There are a variety of NoSQL database architecture models available for developers. All of them have their own unique uses and there are various risks related to these architectures:

Key-Value Stores

In this model, a unique key is assigned to every data object. The unique key is considered as an identifier. Most key-value databases do not require a query language to interact with data, but these databases have a simple command-line method such as “get, put, delete” to communicate with data. The most common example of NoSQL database following this principle is Voldemort. (Gupta, Singh, Kumar, & Tomar, 2018) (Foote, Understanding Key-Value Databases, 2020)

Wide Column Store

Wide column store is based on the principle of storing columns together as a group called “Column family” rather than storing as a group of rows. These databases generally offer high scalability and performance. Each column family contains a different number of rows, and each row again contains a different number of columns. (Gupta, Singh, Kumar, & Tomar, 2018) (IAN, 2016)

Document Store Model

In this database, the document is the most basic unit of data. The document is like a row in RDBMS. The document is an ordered set of key-value pairs. These documents are mostly in JSON and XML format. In the document model, data is stored in the form of tree-like structure and is the most organized storage model among all four. It is widely used in the areas where accuracy is priority as

it stores the exact data. The most popular Documented NoSQL database is MongoDB. (Gupta, Singh, Kumar, & Tomar, 2018) (MongoDB, Inc., 2021)

Graph Store Model

It uses symmetric queries with nodes, edges and other properties to store and represent the data in a graphical structure, and this query can be executed by traversing the edges. This structure is suitable for handling the complex data structure in which are numerous relationships that exist among datasets. Examples of Graph store models are OrientDB, Neo4j etc. (Gupta, Singh, Kumar, & Tomar, 2018) (Amazon Web Services, Inc., n.d.)

Review of existing NoSQL DBMS

Some of the most popular NoSQL DBMS include:

MongoDB

MongoDB is one of the dominant NoSQL databases around the globe. It is coded in C++ language and is based on NoSQL Approach which makes use of non-structured query language. It does not make any use of rows and columns, and it uses the JSON format of documents to store data inside its directories. In MongoDB, there are databases, collections and documents. Documents are stored in the collections, which are like the tables of a conventional database management system. There is no real schema enforced in it and there can be various fields in every document. There is no concept of a key but the combination of two fields is possible. MongoDB works with concepts of replication and sharding. (Abramov, Gudes, Gonen, Okman, & Gal-Oz, 2011) (Intellipaat, 2020)

Cassandra DB

Apache Cassandra is another popular open-source NoSQL system widely adopted by many organizations for a database management system due to its capability of handling a very large amount of data, developed by Facebook in 2008. Scalability, availability and continuous performance are the major pillars of its architecture. The Cassandra DB is a column-wide database system where values are stored in the columns. If a sudden increase in capacity is required, Cassandra is a very good choice. Its architecture is designed to provide high scalability and availability and reliability, as if any node fails the performance and availability will not be affected. It supports all kinds of structured and unstructured data types. (Instaclustr, 2020)

CouchDB

CouchDB is an open-source NoSQL database system developed by Apache Software Foundation. The data is stored in the form of a document which is in JSON format. The CouchDB queries use 'views' which functions based on JavaScript. The data structure which is implemented in CouchDB is known as a document, which is a kind of key-value pair. There is no fixed schema for data storage, but there is some internally defined schema as well, which is known to applications. CouchDB uses replication instead of sharding. Queries are implemented in the form of JavaScript views, which can be distributed over multiple nodes using the map to reduce functionality. (Intellipaat, 2020)

Elasticsearch

It is a distributed open-source search engine that is developed on the top of the Java search engine library called Apache Lucene. The Lucene strives to provide best-in-class performance as a local search engine thus providing developers with powerful tool to work with unstructured texts. (Amazon Web Services, Inc., 2021) Unfortunately, Lucene is not an application, it is just a code library (Zachary, Gormley, & Tong, 2015). So, to leverage its strength, Java comes into play and would require an understanding of how Lucene works but it seems to be difficult as this code library is very complicated. Elasticsearch, which is written in Java, uses the Lucene library for indexing and searching. It is not only used as a search engine, but also as a data storage and retrieval tool. Elasticsearch is gaining popularity day by day. As per Elastic, the company that developed Elasticsearch claimed that "3286 IT organizations are using Elasticsearch in their backbone stacks" including Shopify, Uber, Udemy, etc. (Apache Lucene 4.2.1) (Stackshare Elasticsearch, 2021)

Risk Analysis

The technical and managerial risks related to NoSQL have been reviewed for the following areas of concern:

- **Data at Rest:** A strong encryption mechanism for data residing in NoSQL databases should be offered by the database in case if access to the storage is compromised.
- **Authentication and Authorization:** A NoSQL database system should have a feature to authenticate and authorize user or application before providing access to the data.
- **Transaction Auditing:** Transaction auditing is an important feature that traces the actions taken on data in the database. The proper transaction auditing helps in analyzing the root cause of various transactional issues and ensures user accountability.
- **Injection Attacks:** A database or the application that has access to it should have tools to filter out injections that may be leading to data leaks, losses or unauthorised alternations.

In table 1, the above technical concerns have been considered and comparisons have been made among various existing NoSQL DBMS. Table 2 shows various generic risks associated with DBMS along with the risks that are specific to NoSQL technologies. The extended explanation of the specific risks is provided below:

- **Risk related to failure to provide redundancy instantaneously:** Elasticsearch provides redundancy features. The master node is sending a ping request after every specified amount of time, and every node responds to those requests. If the master node has not received a reply from any node, then that node will be removed from the cluster. Similarly, each node sends out a ping request to the master node to check whether it is still up or not. If they fail to get a response from the master node again, election starts and the master node would get selected, and the new master node will serve the requests and thus provide the redundancy. In the case of failure, if the system fails to shift the responsibilities of the master node to another node then the in-between transaction may be hampered, and it may result in the loss of valuable data.

Category / DBMS	Elasticsearch	Mongo DB	Cassandra DB	Couch DB
Data at Rest	Data at rest encryption is not provided but the platinum x-pack security version provided filesystem encryption.	Data at rest encryption is not provided in the main version but it is included in the MongoDB Atlas version.	Data at rest encryption is not provided in the normal version but it is offered by DataStax Enterprise releases of Cassandra.	Data at rest can only be encrypted with the help of third-party software.
Authentication	X-pack plug-in is required to enable user authentication	Authentication is only provided when the database runs in un-sharded mode.	By default, the authenticator setting of CassandraDB is AllowAllAuthenticator, e.g. no authentication.	By default, CouchDB runs in admin party mode and anyone can do anything.

Category / DBMS	Elasticsearch	Mongo DB	Cassandra DB	Couch DB
Authorization	X-pack plug-in is required to enable user authorization	By default, the authorization is disabled and it can be only achieved by a role-based approach which is very limited.	By default, the authorization setting of basic CassandraDB is AllowAllAuthorization which means there is no authorization	The authorization rules should configure security objects: admins and members. If there is no member in member elements it means all users can read/write and edit documents.
Transaction Auditing	Available but not enabled by default	Not Available, requires third party application to perform transaction auditing	Not Available, requires third party application to perform transaction auditing.	Available but not enabled by default.
Injection Attacks	CVE-2014-3120 vulnerability of unpatched Elasticsearch systems allows to execute the remote script to hijack the system. (Matt, 2014),	As per (Securing MongoDB from External Injection Attacks, 2019), MongoDB is vulnerable to JavaScript injection.	CQL (Cassandra Query Language) attack is possible in Cassandra DB.	As per (Apache Software Foundation CouchDB, 2010), CSFR attack is possible in CouchDB.

Table 1 Comparison of Security Features in various NoSQL Databases.

NoSQL DBMS Specific Risks	Generic DBMS Risks
Risk related to synchronization of nodes	Risk related to insufficient Monitoring
Risk related to failure of cluster	Risk related to improper functioning disaster recovery server
Risk related to master node election	Risk related to weak password and user lock-out policies
Risk related to failure of back-up nodes	Risk related to remote access of database
Risk related to synchronization of primary and replica shards	Risk related to access of database to third parties
Risk related to migration of data to other NoSQL System	Risk related to proper maintenance of logs
Risk related to poor vendor support	Risk related to incorrect privileges to users
Risk related to data consistency	High risk related to insider attacks
Risk related to improper configuration	Risk related to proper maintenance of logs
Risk related to lack of expertise	Risk related to failure of redundancy feature
Risk related to improper choice of storage model	Risk related to location of hardware

Table 2 Managerial Risks Related to NoSQL Technologies

- Risk Related to Lack of Expertise: Since Elasticsearch is a new technology, there are fewer trained resources who can configure and manage it properly. Most of the security issues related to Elasticsearch are due to improper configuration of the system because of lack of expertise. (Pretorius, 2013)
- Risk Related to Data Inconsistency: This problem arises in the case when data stored in one document is dependent on the data stored in another document and there would be no synchronization between them. This situation can be handled in a relational database system with the help of third normal form, whereas NoSQL lacks this concept as data is stored in documents instead of tables.
- Risk Related to Synchronization of Shards: As NoSQL DBMS have a concept of sharding in their architecture. If the system fails to synchronize the shards, it would result in inconsistent data. This is the most common issue in NoSQL technologies.
- Risk related to Assigning Incorrect Privileges to Users : If a user has incorrect privileges, there would be a risk of exploiting the system by an internal attacker.
- Risk Related to Data Back-up: If data is not being backed up properly then there would be a risk of improper functioning of the system when running on backup server in case of failure of production server.
- Risk Related to Insufficient Monitoring: Since Elasticsearch provides redundancy, in case of failure of master node, the responsibilities will be transferred to back-up node and sometimes due to insufficient monitoring, the cause of transferring to the back-up server remains unknown. This happens due to insufficient monitoring.
- Risk Related to Improper Functioning of Disaster Recovery Server: Since Elastic server has two server components, one is production(primary) and other one is secondary (back-up). Most of the time, the back-up server failed to provide services during failure of primary server due to lack of testing of secondary servers on regular intervals.
- Risk Related to Weak Password and User Lock-Out Policies: The weak password policies and user lock-out policies of database systems could lead to the exploitation of databases.
- Risk Related to Proper Maintenance of Logs: Most NoSQL systems have poor auditing subsystems. To log the transactions, the Elasticsearch system needs to be configured manually. Improper configuration may lead to the improper logging of transactions and logging systems will be insufficient to find the root cause of the incidents happening in the DBMS.
- Risk Related to Poor Vendor Support: As per the review the vendor support of Elasticsearch is not only expensive but the support system does not seem to be good also. In such case, poor vendor support could be a major risk for the organization using Elasticsearch. (Elasticsearch Reviews, n.d.)
- Risk Related to Access of Databases to Third Parties: Since NoSQL systems do not provide robust transaction logging systems, there would be lack of visibility of remote activities and would be a risk to organization.
- Risk Related to Migration of Data to Other Nosql Systems: Since Elasticsearch is a NoSQL system and there is no defined structure in which data is getting stored, it would be nearly impossible to do the migration of data to another NoSQL system without significant effort to parse and re-structure data in order to get them ready for migration. .
- Risk Related to Insider Attacks: NoSQL databases are prone to risk of insider attacks because NoSQL DBMS's lack adequate authentication. On the top of it, the technique used in NoSQL for authorization is also inefficient which can act as the weakest node and help to perform an attack. These databases also lack logging abilities which may help imposters to remain untracked.

All the risks described above should be properly addressed in the enterprise level deployments of NoSQL technologies. To achieve that, risk management should be properly established and

maintained for these new technologies. The next section of the paper uses a risk-based approach to build an Audit and Assurance program for NoSQL technology.

Auditing

An audit and assurance program has been designed in alignment to the COBIT IT governance and management framework. This program could be used to utilize NoSQL databases more securely and efficiently by achieving main security goals: confidentiality, integrity, and availability. The scope of this audit and assurance program is wide as it covers all the areas which can be potentially vulnerable and are at risk due to the impact of NoSQL technology. The initiation of this audit program, as aligned to the COBIT framework, sets at the point of evaluation. So, in the initial phase, all the risks associated with the NoSQL databases were required to be evaluated to conduct the gap analysis and design an effective approach. For NoSQL audit program, all the risks identified are mentioned in the former section of this document. These risks cover all the technical and managerial areas of the NoSQL technology. It focuses on both the NoSQL specific risks and the generic risks associated with the databases, irrespective of the technology being used.

Once all the risks are identified and gap analysis are completed, the next step is to follow a directive approach in which the effectiveness of the security measures and controls is assured. This program covers all the major domains such as identity and access management, network configuration and management, security incident response, logging and monitoring, and resource and data security for testing the security of these non-relational databases.

This audit and assurance program is unique as it has been practically performed on the Elasticsearch database servers deployed in the test cloud environment. The outcome of this audit program is a sample audit document which contains use cases of all the major domains, as mentioned above. For every use case, there is an objective – to give a brief overview of the requirement for the use case, controls – to demonstrate the specific security measures or controls required for them and the testing steps – to enlist all the steps required to perform the audit. Additionally, for the technical use cases specially belonging to the Elasticsearch technology, a testing pictorial reference document which contains the screenshots of testing steps has been prepared. These pictorial documents contain all the required information by an auditor to perform auditing for those test cases. This document not only covers the testing and auditing of the main non-relational database or servers but, it contains the steps to test for the security of the applications or services which depend on the NoSQL database for its use. Hence, all the testing steps are designed considering the vulnerabilities associated due to third-party usage. For the detailed information about the use cases, testing procedure and pictorial reference, kindly refer to the sample audit document for Elasticsearch.

COBIT Cross Reference

The audit and assurance program has been developed in alignment with the ISACA COBIT 2019 framework. The framework starts with the enterprise strategy and then finding the enterprise goals which supports that strategy. Following enterprise goals are identified with respect to the Audit and assurance program for NoSQL databases.

Reference	Balance Scorecard dimension (BSC)	Enterprise Goal
EG02	Financial	Managed Business Risk
EG03	Financial	Compliance with external law and regulation
EG06	Customer	Business service and continuity availability
EG10	Internal	Staff skills, Motivation and Productivity
EG11	Internal	Compliance with internal policies

Table 3 COBIT-2019 Enterprise Goals (ISACA, 2019)

The enterprise goal EGo2 - Managed Business Risks – should be considered because the goal of audit and assurance program is to manage the business risks and here the use cases are created to cover most of the business risk. Since the developed audit program includes the uses cases belonging to the data compliance and privacy it satisfies the EGo3 goal of the COBIT-2019 framework. The use case is defined to ensure the business service and continuity availability of information systems and hence satisfy the EGo6 of COBIT-2019 Framework.

Since there are few risks related to lack of skills, the use case is added to cover this risk and satisfies the EG10 enterprise goal. The use cases related to password policies are linked to the enterprise goal EG11.

After understanding the enterprise strategy and goals, the next step is to identify the risks, as per the risk profile of the COBIT framework. The risks which have been identified are aligned to the risk profile matrix provided in the COBIT-2019 framework and belong to the categories such as IT expertise, enterprise/IT architecture, IT operational infrastructure incidents, unauthorized actions, software adoption/usage problems, hardware failure, software failure, logical attacks, third-party/supplier incidents, non-compliance, and data information and management. (ISACA, 2019)

Before auditing the NoSQL databases, the following five areas have been identified to determine the scope of the audit and governance & management objective of COBIT-2019 framework have been linked to each of these areas. The governance and management objectives are grouped in five domains EDM (Evaluate Direct and Monitor), APO (Align, Plan, and Organize), DSS (Deliver, Service and Support) and MEA (Monitor, Evaluate and Assess).

1. **Identity and Access Management:** The identity and access management include the controls which need to be applied to manage the user access to the organizational data. In this the Use cases have been created to verify the controls related to user privileges, authentication, account lock-out policy has been covered.
2. **Network Configuration and Management:** Another important area that needs to consider while auditing the NoSQL database is Network Configuration and Management. This area helps in verifying whether the NoSQL database is operating under a secure network. The use cases which ensure network security with multiple defense vectors, activation of network, Environment segregation and implication of strong firewall rules.
3. **Security Incident Response:** The auditing of this area ensures that the organization has a rigid incident response plan in case of any incident and has proper detection systems installed to monitor the malicious activity. The use cases under this area cover the various aspects of incident response and verify the communication plan also in case of any emergency.
4. **Logging and Monitoring:** The use cases covered under this verify whether the logs are being generated for NoSQL transactions for each user or not. Apart from this, some uses have been developed to verify the logs generation related to operation of Primary and secondary nodes.
5. **Resource and Data Security:** This area includes the use cases which verify the system reliability and security. The uses cases related to server switch-over, data encryption, back-up policy, physical server and database directories security.

Enterprise Goals	Alignment Goals	Governance and Management Objectives (Primary)	Use-Cases
EGo2 (Managed Business Risk)	AGo2 (Managed I&T-Related Risk)	EDMo3 (Ensured Risk Optimization)	UC2.3, UC2.4, UC2.5, UC2.7, UC2.8, UC3.3, UC3.5
		APO12 (Managed Risk)	UC1.4, UC1.5, UC1.6, UC1.7, UC1.8, UC2.8, UC2.10
		DSSo5 (Managed Security Services)	UC1.9, UC1.10, UC2.8, UC2.10, UC2.11,

Enterprise Goals	Alignment Goals	Governance and Management Objectives (Primary)	Use-Cases
	AGo7 (Security of Information, Processing Infrastructure and applications, and privacy)		UC4.4, UC4.5, UC4.6, UC4.7, UC5.7, UC5.9
		EDMo3 (Ensured Risk Optimization)	UC2.3, UC2.4, UC2.5, UC2.7, UC2.9, UC2.10, UC3.3, UC3.5
		APO12 (Managed Risk)	UC1.4, UC1.5, UC1.6, UC1.7, UC1.8, UC2.9
		APO13 (Managed Security)	UC1.2, UC1.3, UC1.4, UC1.5, UC1.6, UC1.7, UC1.8, UC4.2, UC5.4, UC5.5, UC5.5, UC5.9
		BAI10 (Managed Configuration)	UC2.1, UC2.2, UC2.3, UC2.4, UC2.6, UC2.8, UC4.1, UC4.3, UC4.8, UC5.9
		DSSo4 (Managed Continuity)	UC3.2, UC4.4, UC5.6
		DSSo5 (Managed Security Services)	UC1.9, UC1.10, UC2.8, UC2.10, UC2.11, UC4.4, UC4.5, UC4.6, UC4.7, UC5.7, UC5.9
EGo3 (Compliance with External Laws and Regulations)	AGo1 (I&T Compliance and Support for Business compliance with external Laws and Regulations)	EDMo1 (Ensured Governance Framework Setting and Maintenance)	UC3.1, UC3.4
		MEAo3 (Managed Compliance with External Requirements)	UC3.1, UC3.4
	AG11 (I&T compliance with Internal Policies)	APO1 (Managed I&T Management Framework)	UC2.1
		MEAo2 (Managed System of Internal Control)	UC4.6
		MEAo4 (Managed Assurance)	UC4.4, UC4.5, UC4.6, UC4.7
EGo6 (Business Service Continuity and Availability)	AGo7 (Security of Information, Processing Infrastructure and applications, and privacy)	EDMo3 (Ensured Risk Optimization)	UC3.2, UC5.4, UC5.5
		APO12 (Managed Risk)	UC5.1, UC5.2, UC5.3
		APO13 (Managed Security)	UC5.4, UC5.5
		BAI10 (Managed Configuration)	UC5.1, UC5.2, UC5.3, UC5.4
		DSSo4 (Managed Continuity)	UC3.2, UC5.1, UC5.2, UC5.3, UC5.6
		DSSo5 (Managed Security Services)	UC5.8
EG10 (Staff skills, Motivation and Productivity)	AG12 (Competent and Motivated Staff with Mutual	APO7 (Managed Human Resources)	UC5.8
		BAIo8 (Managed Knowledge)	UC1.9, UC1.10

Enterprise Goals	Alignment Goals	Governance and Management Objectives (Primary)	Use-Cases
	Understanding of Technology and Business)		
EG11 (Compliance with Internal Policies)	AG11 (I&T compliance with Internal Policies)	APO1 (Managed I&T Management Framework)	UC2.1
		MEAO2 (Managed System of Internal Control)	UC1.1, UC1.2, UC1.3, UC1.11, UC3.1
		MEAO4 (Managed Assurance)	UC1.1

Table 4 COBIT-2019 Mapping with Test Cases (ISACA, 2019)

Example of alignment of enterprise goals using COBIT-2019

Example below looks at the enterprise goal EGO6-Business Service Continuity and Availability. To start with, the primary alignment goals which are mapped to the enterprise goals are required to be identified. As per table V, the alignment goal is AGO7 i.e., security of information, processing infrastructure and applications, and privacy. There are many controls corresponding to the AGO7 goal and one of them is DSSO4 (Manage Continuity). The use cases mapped to these controls that ensure the Managed Continuity are UC3.2, UC5.1, UC5.2, UC5.3, UC5.6.

After mapping of the auditing use cases to the COBIT goals and controls, testing using these cases are required. For this the audit and assurance program spreadsheet [<https://sites.google.com/student.concordia.ab.ca/cuegraduate-research-project/home>] should be referred as it contains testing procedures corresponding to each use case.

Referring to the audit and assurance program spreadsheet, first, the use case UC 3.2 which is under the contingency plan category should be conducted. This use case verifies the business continuity and disaster recovery plan and includes the verification of business continuity plan document, ensuring proper data backups of production environment and services are maintained in the alternative sites, interviewing key team members for business continuity plan awareness. Once this use case is conducted, the result with a comment should be submitted in the result column of an audit and assurance program spreadsheet by the auditor, as shown in the figure.

NoSQL DBMS Audit Program				
Ref. No.	s Sub-area	Control Objective	Control	Testing Step
UC3.2	Contingency	The enterprise must ensure to have an effective business continuity and disaster recovery plan to maintain seamless operations.	The business continuity and disaster recovery plan must be updated and practiced regularly as per the change in the operations. BCP drills must be conducted to ensure the proper functioning of hot, warm and cold sites. Organization must also prepare a document, consisting details and observations of all the BCP drills conducted, to serve the auditing purpose.	Business continuity plan is the responsible for seamless operations of the enterprise during an pandemic situation. Below steps are required to be adhered to check the effectiveness of the BCP. 1. Business continuity plan execution document is required to be reviewed so that the outcomes of the BCP drills conducted should be checked and compared to the baselines. 2. Ensure that proper data backups and services are maintained on the alternative sites to prevent the loss of data. 3. Interview key team members for the business continuity plans and its awareness. Active testing could be conducted by creating a scenario and then monitoring the performance.
				Pictorial Refer

Figure 1 Snippet of Audit and Assurance Spreadsheet Showing Use case UC3.2

Second, another use case UC5.1 should be conducted which verifies the health of the cluster by checking whether the Elasticsearch server has sufficient secondary nodes, and all are in healthy state. The testing steps for this use case are mentioned in the audit and assurance spreadsheet along with the pictorial reference which could be referred for screenshots of execution of each step, as shown in the figure.


NoSQL DBMS Audit Program					
Ref. No.	Process Sub-area	Control Objective	Control	Testing Step	Pictorial Reference
UC5.1		The enterprise ensures that there is clear understanding across the organization regarding strategy and plan of action to make system a robust and reliable system.	Verification of all the clusters health status as per the network architecture of the enterprise.	To verify the cluster health below mentioned steps are need to be followed: 1. Open the browser and execute the command - <IP_ADDRESS_OF_ELASTICSERVER:PORT_NUMBER>/_cluster/health?pretty 2. Check whether cluster has same number of nodes as mentioned in the enterprise network structure. 3. Check the status whether it is green or not.	

Figure 2 Snippet of Audit and Assurance Spreadsheet Showing Use Case UC5.1 along with the Pictorial Reference Document.

Similarly, the use case UC 5.2 and all the other use cases should be executed which is related to verification of data synchronization among the primary as well as secondary node. The testing steps and pictorial reference could be referred to execute the use case. Once all the use cases which are mapped to the specific alignment goal have been conducted and resulted as "PASS", the enterprise goal is considered as being properly supported by IT alignment goals with respect to NoSQL technology used.

Conclusion

Although NoSQL databases have rapidly become popular for data storage, there have been some risks related to authentication, data privacy, data redundancy and configuration issues. These risks could result into the compromise of organizational data or might impact the performance at operational level which might affect financial as well as reputational value of an organization. So, in this research, an audit and assurance program specifically for NoSQL database technologies has been designed which includes the verification of controls specifically for the NoSQL database management systems. The components of the audit program have been linked to the COBIT-2019 framework in such a way that the enterprise and alignment goals were supported by the IT goals. In addition to it, the template consisting of use cases, testing procedures and pictorial reference documents has been designed considering Elasticsearch as the primary database for testing. This could be used by the auditors to conduct the auditing process of NoSQL technologies.

Template for Audit and Assurance program for NoSQL is available at [https://sites.google.com/student.concordia.ab.ca/cuegraduate-research-project/home].

Bibliography

- Abramov, J., Gudes, E., Gonen, Y., Okman, L., & Gal-Oz, N. (2011). *Security Issues in NoSQL Databases*. Retrieved March 1, 2021, from IEEE.org: <https://ieeexplore.ieee.org/document/6120863>
- Amazon Web Services, Inc. (2021). *What Is Amazon Elasticsearch Service?* Retrieved January 20, 2021, from AWS: <https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/what-is-amazon-elasticsearch-service.html>
- Amazon Web Services, Inc. (n.d.). *What Is a Document Database?* Retrieved January 10, 2021, from AWS Amazon: <https://aws.amazon.com/nosql/document/>
- Apache Lucene 4.2.1. (n.d.). *Apache Lucene 4.2.1 Documentation*. Retrieved Jan 20, 2021, from Lucene: https://lucene.apache.org/core/4_2_1/
- Apache Software Foundation CouchDB. (2010, February 21). Retrieved January 29, 2021, from <https://docs.couchdb.org/en/2.3.1/cve/2010-2234.html>
- Couchbase.com. (n.d.). Retrieved January 16, 2021, from <https://www.couchbase.com/resources/why-nosql>. [Accessed: 16- Jun- 2020].
- Elasticsearch Reviews. (n.d.). Retrieved January 21, 2021, from TrustRadius: <https://www.trustradius.com/products/elasticsearch/reviews?qs=pros-and-cons>

- Foote, K. D. (2018, June 19). *A Brief History of Non-Relational Databases*. Retrieved January 14, 2021, from Dataversity: <https://www.dataversity.net/a-brief-history-of-non-relational-databases/>
- Foote, K. D. (2020, January 30). *Understanding Key-Value Databases*. Retrieved January 15, 2021, from Dataversity: <https://www.dataversity.net/understanding-key-value-databases/>
- Hossain, S. A., & Moniruzzaman, A. B. (2013). *NoSQL Database*. Retrieved January 26, 2021, from <https://arxiv.org/ftp/arxiv/papers/1307/1307.0191.pdf>
- IAN. (2016, June 23). *What is a Column Store Database?* Retrieved February 15, 2021, from Database.Guide: <https://database.guide/what-is-a-column-store-database/>
- Instaclustr. (2020). *What is Apache Cassandra?* Retrieved January 22, 2021, from Instaclustr: <https://www.instaclustr.com/apache-cassandra/#how-cassandra-works>
- Intellipaat. (2020). *What is MongoDB?* Retrieved January 20, 2021, from Intellipaat: <https://intellipaat.com/blog/what-is-mongodb/>
- ISACA. (2019). COBIT-2019. In *COBIT-2019 Design Guide*.
- Kumar, A. (2018). *An Introduction to NoSQL Databases*. Retrieved February 23, 2021, from <http://pages.di.unipi.it/turini/Basi%20di%20Dati/Slides/11.NoSQL-slides.pdf>
- Matt, B. (2014, May 18). Retrieved January 29, 2021, from <https://medium.com/@bromiley/exploiting-elasticsearch-c83825708ce1>
- MongoDB, Inc. (n.d.). *What is NoSQL?* Retrieved February 5, 2021, from mongoDB.: <https://www.mongodb.com/nosql-explained>
- Normalization step by step*. (n.d.). Retrieved January 27, 2021, from <https://www.w3computing.com/systemsanalysis/normalization-steps-example/>
- NoSQL Databases*. (n.d.). Retrieved February 15, 2021, from SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3172769
- Pretorius, D. J. (2013). NoSQL database considerations and implications for businesses. *Securing MongoDB from External Injection Attacks*. (2019, 11 October). Retrieved January 29, 2021, from <https://severalnines.com/database-blog/securing-mongodb-external-injection-attacks>
- Stackshare Elasticsearch*. (n.d.). Retrieved March 21, 2021, from <https://stackshare.io/elasticsearch>
- Zachary, Gormley, C., & Tong. (2015). *Elasticsearch The Definitive Guide*. Sebastopol, CA: O'Reilly. Retrieved from <https://storage.ey.md/Technology%20Related/PDFs%20and%20Books/Elasticsearch%20The%20Definitive%20Guide%20-%20Clinton%20Gormley%2C%20Zachary%20Tong.pdf>