**Mysql InnoDB Cluster secure installation**

MySQL is the most popular Open Source Relational SQL database management system. MySQL is one of the best RDBMS being used for developing web-based software applications.

InnoDB is a general-purpose storage engine that balances high reliability and high performance. In MySQL 8.0, InnoDB is the default MySQL storage engine. Unless you have configured a different default storage engine, issuing a [CREATE TABLE](https://dev.mysql.com/doc/refman/8.0/en/create-table.html) statement without an ENGINE clause creates an InnoDB table.

An InnoDB Cluster consists of at least three MySQL Server instances, and it provides high-availability and scaling features. InnoDB Cluster uses the following MySQL technologies:

* [MySQL Shell](https://dev.mysql.com/doc/mysql-shell/8.0/en/), which is an advanced client and code editor for MySQL.
* MySQL Server, and [Group Replication](https://dev.mysql.com/doc/refman/8.0/en/group-replication.html), which enables a set of MySQL instances to provide high-availability. InnoDB Cluster provides an alternative, easy to use programmatic way to work with Group Replication.
* [MySQL Router](https://dev.mysql.com/doc/mysql-router/8.0/en/), a lightweight middleware that provides transparent routing between your application and InnoDB Cluster.

The following diagram shows an overview of how these technologies work together:



Being built on MySQL [Group Replication](https://dev.mysql.com/doc/refman/8.0/en/group-replication.html), provides features such as

* Automatic membership management,
* Fault tolerance,
* Automatic failover,
* And so on.

An InnoDB Cluster usually runs in a single-primary mode, with one primary instance (read-write) and multiple secondary instances (read-only).

Advanced users can also take advantage of a [multi-primary](https://dev.mysql.com/doc/refman/8.0/en/group-replication-multi-primary-mode.html) mode, where all instances are primaries. You can even change the topology of the cluster while InnoDB Cluster is online, to ensure the highest possible availability.

InnoDB Cluster supports [MySQL Clone](https://dev.mysql.com/doc/refman/8.0/en/clone-plugin.html), which enables you to provision instances simply. In the past, to provision a new instance before it joins a set of MySQL instances you would need to somehow manually transfer the transactions to the joining instance.

Using InnoDB Cluster, you can simply [add an instance](https://dev.mysql.com/doc/mysql-shell/8.0/en/add-instances-cluster.html) to the cluster and it is automatically provisioned.

**Key Advantages of InnoDB**

* Its DML operations follow the ACID model, with transactions featuring commit, rollback, and crash-recovery capabilities to protect user data. Row-level locking and Oracle-style consistent reads increase multi-user concurrency and performance.
* InnoDB tables arrange your data on disk to optimize queries based on primary keys. Each InnoDB table has a primary key index called the clustered index that organizes the data to minimize I/O for primary key lookups.
* To maintain data integrity, InnoDB supports FOREIGN KEY constraints. With foreign keys, inserts, updates, and deletes are checked to ensure they do not result in inconsistencies across related tables.

Installation configurations

Wee will install the following components on four nods

* Mysql innodb on three nods
* Mysqlsh on all four nods
* Mysql router on router node

==================Host resolving on three nodes===========================
**vi /etc/hosts**[mysql@prfinf1keycloack ~]$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

**192.168.56.71   host01
192.168.56.72 host02
192.168.56.73   host03**
===================Installing the MySQL on all three nodes=====================
             cd /stage/MySQL-Server

# dnf install \*.rpm

# grep /var/log/mysqld.log ‘password’

# systemctl start mysqld

#systemctl enable mysqld
========================================

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY MySQL8.0';
===============================================================
=============create root user that can access mysql from remote===================
**mysql -u root -p**
mysql>**CREATE USER 'root1'@'%' IDENTIFIED BY 'MySQL8.0';**
mysql>**GRANT ALL PRIVILEGES ON \*.\* TO 'root1'@'%' WITH GRANT OPTION;**
mysql>**FLUSH PRIVILEGES;**
mysql>**CREATE USER 'cbe'@'%' IDENTIFIED BY 'MySQL8.0';**
mysql>**GRANT ALL PRIVILEGES ON \*.\* TO 'cbe'@'%' WITH GRANT OPTION;**
mysql>**FLUSH PRIVILEGES;**
------------------------------------------------
**mysqlsh**

**\c root@192.168.56.71:3306**
----node1
**dba.configureInstance("root@192.168.56.71:3306", {mycnfPath: "/etc/my.cnf", clusterAdmin: "cladmin", clusterAdminPassword: "MySQL8.0"})**----node2 **dba.configureInstance("root@192.168.56.72:3306", {mycnfPath: "/etc/my.cnf", clusterAdmin: "cladmin", clusterAdminPassword: "MySQL8.0"})**----node3 **dba.configureInstance("root@192.168.56.73:3306", {mycnfPath: "/etc/my.cnf", clusterAdmin: "cladmin", clusterAdminPassword: "MySQL8.0"})**
----node1
**\c cladmin@192.168.56.71:3306
dba.checkInstanceConfiguration()**----node2 **\c cladmin@192.168.56.72:3306
dba.checkInstanceConfiguration()**----node3 **\c cladmin@192.168.56.73:3306
dba.checkInstanceConfiguration()

mysqlsh
cluster = dba.createCluster('CBEMYSQCLUSTER', {ipWhitelist: '192.168.56.71, 192.168.56.72, 192.168.56.73})
mysqlsh --uri cladmin@192.168.56.71:3306 --cluster
cluster.addInstance('cladmin@10.3.49.29:3306', {ipWhitelist: '192.168.56.71, 192.168.56.72, 192.168.56.73})
cluster.addInstance('cladmin@10.3.49.30:3306', {ipWhitelist: '192.168.56.71, 192.168.56.72, 192.168.56.73})
cluster.status()
exit;
mysqlsh --uri cladmin@10.3.49.28:3306 --cluster
cluster.status()**

**exit;**

**mysqlsh --uri cladmin@10.3.49.29:3306 --cluster
cluster.status()**

**exit;
mysqlsh --uri cladmin@10.3.49.30:3306 --cluster
cluster.status()**
======================================================