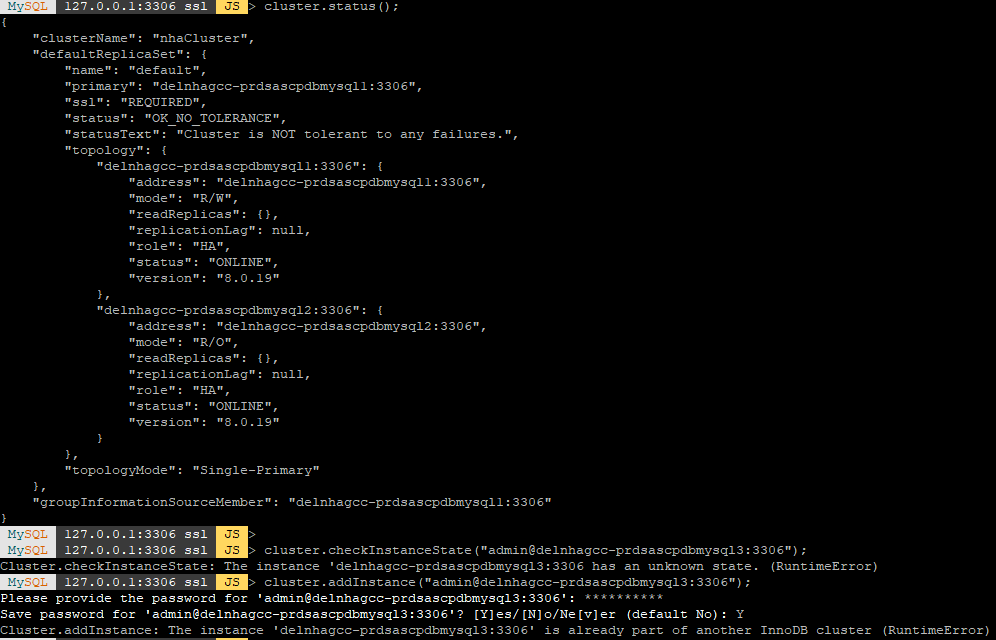
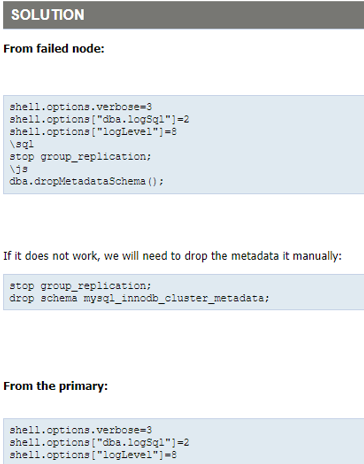
**How to adding Instances (slave node) after missed or unable to show node name in MySQL InnoDB Cluster**





### Rejoining an Instance to a Cluster: - cluster.rejoinInstance (Instance) …

### If an instance leaves the cluster, to rejoin it to the cluster at a later stage. To rejoin an instance to a cluster issue *Cluster*.rejoinInstance (*instance*).

If an instance is attempting to rejoin an instance fails, then a ***Cluster***.rescan() must be executed to add the instance to the metadata using its new [server\_uuid](https://dev.mysql.com/doc/refman/8.0/en/replication-options.html#sysvar_server_uuid).

cluster.removeInstance ("user\_name@host\_name\_or\_host\_ip:3306", {force: true})

cluster.rescan ();

We use cluster.removeInstance with force option because instance is unreachable from cluster, so we remove it from InnoDB Cluster metadata.

1. **Restoring a Cluster from Quorum Loss: - cluster.forceQuorumUsingPartitionOf (Instance)** …

### Connect to an instance which contains the cluster's metadata, then use the *Cluster*.forceQuorumUsingPartitionOf (*instance*) operation, which restores the cluster based on the metadata on *instance*, and then all the instances that are ONLINE from the point of view of the given instance definition are added to the restored cluster.

mysql-js> cluster.forceQuorumUsingPartitionOf("admin@nhadelgcc-india-psmysql-1:3306")

Restoring replicaset 'default' from loss of quorum, by using the partition composed of [icadmin@ic-1:3306]

Please provide the password for 'admin@ nhadelgcc-india-psmysql-1:3306': \*\*\*\*\*\*

Restoring the InnoDB cluster...

The InnoDB cluster was successfully restored using the partition from the instance 'admin@nhadelgcc-psmysql-1:3306'.

WARNING: To avoid a split-brain scenario, ensure that all other members of the replicaset

are removed or joined back to the group that was restored.

In the event that an instance is not automatically added to the cluster, for example if its settings were not persisted, use ***Cluster***.rejoinInstance () to manually add the instance back to the cluster.

### Rebooting a Cluster from a Major Outage: - dba.rebootClusterFromCompleteOutage ()

If cluster is complete outage, we can reconfigure it using dba.rebootClusterFromCompleteOutage (). This operation enables to connect to one of the cluster's MySQL instances and use its metadata to recover the cluster. A complete outage means that group replication has stopped on all member instances.

**Note:-**

Ensure all cluster members are started before running

dba.rebootClusterFromCompleteOutage(). The command will fail if any of the cluster members are unreachable. This check is ignored if the cluster is INVALIDATED and is a member of a ClusterSet.

### Connect to the most up-to-date instance and run the following command:

JS> var cluster = dba.rebootClusterFromCompleteOutage()

The dba.rebootClusterFromCompleteOutage () operation follows these steps to ensure the cluster is correctly reconfigured:

* Cluster metadata and the cluster topology is retrieved from the current instance.
* If a cluster member is in RECOVERING or ERROR, and all other members are OFFLINE or ERROR, dba.rebootClusterFromCompleteOutage () attempts to stop Group Replication on that member. If Group Replication fails to stop, the command stops and displays an error.
* The InnoDB Cluster metadata found on the instance which MySQL Shell is currently connected to is checked to see if it contains the GTID superset. If the currently connected instance does not contain the GTID superset, the operation aborts with that information.
* If the instance contains the GTID superset, the cluster is recovered based on the metadata stored in that instance.
* MySQL Shell checks which instances of the cluster are currently reachable and fails if any member is currently unreachable.
* Similarly, MySQL Shell detects instances which are currently not reachable. It is not possible to add or remove former members to the cluster as part of the dba.rebootClusterFromCompleteOutage () command, if they are currently unreachable.
* If enabled on the primary instance of the cluster, while in single-primary mode,

[super\_read\_only](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_super_read_only) is disabled.

#### **Options**

dba.rebootClusterFromCompleteOutage () has the following options:

* **force**: true | false (default): If true, the operation must be executed even if some members of the Cluster cannot be reached, or the primary instance selected has a diverging or lower GTID\_SET.
* **dryRun**: true | false (default): performs all validations and steps of the command, but no changes are made. A report is displayed when finished.
* **primary**: Instance definition representing the instance that must be selected as the primary.
* **switchCommunicationStack**: mysql | **xcom**: The Group Replication protocol stack to be used by the Cluster after the reboot.
* **ipAllowList**: The list of hosts allowed to connect to the instance for Group Replication traffic when using the XCOM protocol stack.
* **localAddress**: string value with the Group Replication local address to use instead of the automatically generated one when using the XCOM protocol stack.

#### **Force Option**

The force option enables you to ignore the availability of Cluster members or GTID-set divergence in the selected member and reboot the Cluster.

JS> var cluster = dba.rebootClusterFromCompleteOutage ("nhaCluster",{force: true})

The force option is not permitted in the following situations:

* If the Cluster belongs to a ClusterSet and is INVALIDATED or the primary Cluster is not in global status OK,
* The Cluster belongs to a ClusterSet, is the primary Cluster, and is INVALIDATED.

It is not possible to add or rejoin instances with rebootClusterFromCompleteOutage. If you used force to ignore unreachable members and reboot your Cluster, you must use ***cluster***.rejoinInstance () to add the unreachable members to the Cluster.

#### Selecting a Primary with rebootClusterFromCompleteOutage

You can define the Cluster primary in one of the following ways:

* Define the primary option in the dba.rebootClusterFromCompleteOutage () command.

For example, rebooting the Cluster nhaCluster and setting the member running on the local machine, on port 4001, as the primary:

var cluster = dba.rebootClusterFromCompleteOutage ("nhaCluster", {primary: "127.0.0.1:3306"})

* By using the primary option with the force option on a Cluster member with a lower GTID set than another member.

#### **Testing rebootClusterFromCompleteOutage:-**

You can test the changes by using the dryRun option. This option validates the command and its options and generates a log of results. An exception is thrown if there is a problem with the proposed changes.

The following example shows a dry run of rebooting the Cluster, nhaCluster, setting the primary to the local member running on port 4001, and the log message it returns:

JS > var cluster = dba.rebootClusterFromCompleteOutage ("nhaCluster", {primary: "127.0.0.1:3306", dryRun: true})

NOTE: dryRun option was specified. Validations will be executed, but no changes will be applied.

Cluster instances: '127.0.0.1:4000' (OFFLINE), '127.0.0.1:4001' (OFFLINE), '127.0.0.1:4002' (OFFLINE)

Switching over to instance '127.0.0.1:4001' to be used as seed.

dryRun finished.

#### 

#### **Considerations for ClusterSet and ReplicaSet:-**

rebootClusterFromCompleteOutage performs the following checks and generates a warning if the Cluster does not meet the requirements:

* Confirms the Replica Cluster was not forcibly removed from the ClusterSet.
* Confirms the ClusterSet's primary Cluster is reachable.
* Checks the Cluster for errant transactions which are not View Change Log Events (VCLE).
* Confirms the Cluster's executed transaction set ([GTID\_EXECUTED](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html#sysvar_gtid_executed)) is not empty.

The command automatically rejoins a Replica Cluster to the ClusterSet, ensuring the ClusterSet replication channel is configured for all Cluster members.

#### **Switching Communication Stack**

You can switch communication stack during a dba.rebootClusterFromCompleteOutage () operation.

For example:

js> dba.rebootClusterFromCompleteOutage("testCluster", {switchCommunicationStack: "mysql"})

Switching from the MYSQL protocol to XCOM requires an additional network address for the localAddress and may also require you to define ipAllowList values.

### Rescanning a Cluster: -

### The *Cluster*.rescan () operation can detect new active instances that are not registered in the metadata and add them, or obsolete instances (no longer active) still registered in the metadata, and remove them

The syntax of the command is ***Cluster***.rescan ([options]). The options dictionary supports the following:

* **interactive**: Boolean value used to disable or enable the wizards in the command execution.
* **addInstances**: list with the connection data of the new active instances to add to the metadata, or “auto” to automatically add missing instances to the metadata. The value “auto” is case-insensitive.
  + Instances specified in the list are added to the metadata, without prompting for confirmation
  + In interactive mode, you are prompted to confirm the addition of newly discovered instances that are not included in the addInstances option
  + In non-interactive mode, newly discovered instances that are not included in the addInstances option are reported in the output, but you are not prompted to add them
* **removeInstances**: list with the connection data of the obsolete instances to remove from the metadata, or “auto” to automatically remove obsolete instances from the metadata.
  + Instances specified in the list are removed from the metadata, without prompting for confirmation
  + In interactive mode, you are prompted to confirm the removal of obsolete instances that are not included in the removeInstances option
  + In non-interactive mode, obsolete instances that are not included in the removeInstances option are reported in the output but you are not prompted to remove them.
* **updateTopologyMode**: Boolean value used to indicate if the topology mode (single-primary or multi-primary) in the metadata should be updated (true) or not (false) to match the one being used by the cluster. By default, the metadata is not updated (false).
  + If the value is true then the InnoDB Cluster metadata is compared to the current mode being used by Group Replication, and the metadata is updated if necessary. Use this option to update the metadata after making changes to the topology mode of your cluster outside of AdminAPI.
  + If the value is false then InnoDB Cluster metadata about the cluster's topology mode is not updated even if it is different from the topology used by the cluster's Group Replication group
  + If the option is not specified and the topology mode in the metadata is different from the topology used by the cluster's Group Replication group, then:
    - In interactive mode, you are prompted to confirm the update of the topology mode in the metadata
    - In non-interactive mode, if there is a difference between the topology used by the cluster's Group Replication group and the InnoDB Cluster metadata, it is reported and no changes are made to the metadata
  + When the metadata topology mode is updated to match the Group Replication mode, the auto-increment settings on all instances are updated.
* **updateViewChangeUuid**: If you set updateViewChangeUuid to true, the rescan operation generates and sets a value for [group\_replication\_view\_change\_uuid](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_view_change_uuid) on all the member servers, following which you must reboot the cluster to implement the changes.

The ***Cluster***.rescan() command automatically generates and sets the system variable value in the same way as if true was set, with a cluster reboot required afterwards to implement the changes.

When you have rebooted the cluster, you can retry the InnoDB ClusterSet creation process.

* + **upgradeCommProtocol:** Boolean value used to indicate if the Group Replication communication protocol version should be upgraded (true).
  + If the value is false then the Group Replication communication protocol version is not upgraded.

### Fencing a Cluster: -

 Fence the cluster either from write traffic or all traffic. Even though you primarily use fencing on clusters belonging to a clusterset, it is also possible to fence standalone clusters from all traffic.

Three fencing operations are available:

* <Cluster>.fenceWrites (): Stops write traffic to a primary cluster of a ClusterSet.
* <Cluster>.unfenceWrites (): Resumes write traffic.
* <Cluster>.fenceAllTraffic (): Fences a cluster from all traffic.