MYSQL JOB Monitoring

In this document we have monitoring the database job's and as well as cron job both we will monitor .

Author : Ajit Yadav (MSSQL , MYSQL , POSTGRESQL DBA)

📌Step 1 — Create One Monitoring Database

```
CREATE DATABASE job_monitor;

USE job_monitor;

CREATE TABLE job_status (
   id INT AUTO_INCREMENT PRIMARY KEY,
   server_name VARCHAR(50),
   job_name VARCHAR(100),
   job_type ENUM('CRON','EVENT'),
   start_time DATETIME,
   end_time DATETIME,
   status ENUM('RUNNING','SUCCESS','FAILED'),
   message VARCHAR(255)
);

CREATE USER 'monitor_agent'@'%' IDENTIFIED BY
'monitor@';

GRANT INSERT, UPDATE, SELECT
ON job_monitor.*
TO 'monitor_agent'@'%';

FLUSH PRIVILEGES;
```

Allow network access : /etc/mysql/mysql.conf.d/mysqld.cnf

```
bind-address = 0.0.0.0
```

📌Step 2 — Log from Cron Jobs (Shell Script)

```
#!/bin/bash

SERVER="DB01"
JOB="DB_BACKUP"
USER="monitor_agent"
PASS="monitor@"
DB="job_monitor"

START=$(date '+%F %T')

# Insert RUNNING
mysql -u"$USER" -p"$PASS" "$DB" <<EOF
INSERT INTO job_status
(server_name,job_name,job_type,start_time,status)
VALUES
('$SERVER','$JOB','CRON','$START','RUNNING');
EOF

# Run backup
/usr/local/bin/database_backup_job.sh

if [ $? -eq 0 ]; then
   STATUS="SUCCESS"
   MSG="Backup completed"
else
   STATUS="FAILED"
   MSG="Backup failed"
fi

END=$(date '+%F %T')
# Update status
mysql -u"$USER" -p"$PASS" "$DB" <<EOF
UPDATE job_status
SET end_time='$END',
   status='$STATUS',
   message='$MSG'
WHERE server_name='$SERVER'
AND job_name='$JOB'
AND start_time='$START';
EOF
```

On EACH Server: Modify Script

```
CENTRAL_DB="10.0.0.10"

USER="monitor_agent"
PASS="monitor@"
DB="job_monitor"

SERVER_NAME=$(hostname)

START=$(date '+%F %T')

# Log START
mysql -h "$CENTRAL_DB" -u"$USER" -p"$PASS"
"$DB" <<EOF
INSERT INTO job_status
(server_name,job_name,job_type,start_time,status)
VALUES
('$SERVER_NAME','DB_BACKUP','CRON','$START','R
UNNING');
EOF

END=$(date '+%F %T')

mysql -h "$CENTRAL_DB" -u"$USER" -p"$PASS"
"$DB" <<EOF
UPDATE job_status
SET end_time='$END',
   status='SUCCESS',
   message='Backup OK'
WHERE server_name='$SERVER_NAME'
AND job_name='DB_BACKUP'
AND start_time='$START';
EOF
```

📌Step 3 — Log from MySQL Events

```
DELIMITER $$

CREATE EVENT delete_old_data
ON SCHEDULE EVERY 1 DAY
STARTS '2026-01-20 01:00:00'
DO
BEGIN

DECLARE start_t DATETIME;
SET start_t = NOW();

INSERT INTO job_monitor.job_status
(server_name,job_name,job_type,start_time,status)
VALUES
('MYSQL01','DELETE_OLD','EVENT',start_t,'RUNNING');

-- Your real job
DELETE FROM logs WHERE created_at < NOW() - INTERVAL 90 DAY;

UPDATE job_monitor.job_status
SET end_time=NOW(),
   status='SUCCESS',
   message='Cleanup done'
WHERE job_name='DELETE_OLD'
AND start_time=start_t;

END$$

DELIMITER ;
```

📌Step 4 — View All Jobs (One Query)

```
SELECT
  server_name,
  job_name,
  job_type,
  start_time,
  end_time,
  status,
  message
FROM job_monitor.job_status
ORDER BY start_time DESC
LIMIT 50;
```

This command is use on create a db on slave server :

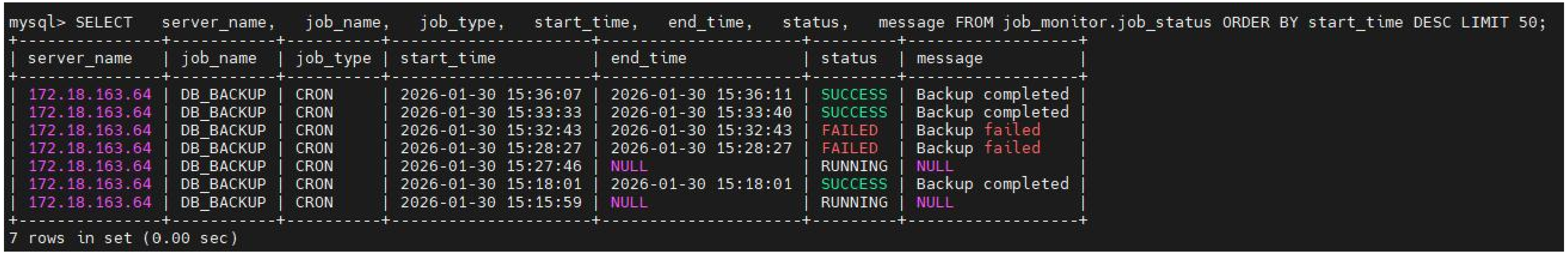
```
SET SESSION sql_log_bin = 0;

CREATE DATABASE job_monitor;

SET SESSION sql_log_bin = 1;
```

✓ No GTID generated
✓ Not written to binlog
✓ Not replicated
✓ Safe

⚠ Must be SUPER careful.

📌Step 5 — Detect Failed Jobs (Daily Check)

```
SELECT *
FROM job_monitor.job_status
WHERE status='FAILED'
AND start_time >= NOW() - INTERVAL 1 DAY;
```



```
mysql> SELECT   server_name,   job_name,   job_type,   start_time,   end_time,   status,   message FROM job_monitor.job_status ORDER BY start_time DESC LIMIT 50;
+---------------+-----------+----------+---------------------+---------------------+---------+------------------+
| server_name   | job_name  | job_type | start_time          | end_time            | status  | message          |
+---------------+-----------+----------+---------------------+---------------------+---------+------------------+
| 172.18.163.64 | DB_BACKUP | CRON     | 2026-01-30 15:36:07 | 2026-01-30 15:36:11 | SUCCESS | Backup completed |
| 172.18.163.64 | DB_BACKUP | CRON     | 2026-01-30 15:33:33 | 2026-01-30 15:33:40 | SUCCESS | Backup completed |
| 172.18.163.64 | DB_BACKUP | CRON     | 2026-01-30 15:32:43 | 2026-01-30 15:32:43 | FAILED  | Backup failed    |
| 172.18.163.64 | DB_BACKUP | CRON     | 2026-01-30 15:28:27 | 2026-01-30 15:28:27 | FAILED  | Backup failed    |
| 172.18.163.64 | DB_BACKUP | CRON     | 2026-01-30 15:27:46 | NULL                | RUNNING | NULL             |
| 172.18.163.64 | DB_BACKUP | CRON     | 2026-01-30 15:18:01 | 2026-01-30 15:18:01 | SUCCESS | Backup completed |
| 172.18.163.64 | DB_BACKUP | CRON     | 2026-01-30 15:15:59 | NULL                | RUNNING | NULL             |
+---------------+-----------+----------+---------------------+---------------------+---------+------------------+
7 rows in set (0.00 sec)
```