Backup Database

SSH Prepare

Update system packages (optional but recommended):

```
sudo apt update
```

Create Backup Directories for the Project

Create separate folders to store backups for the project:

```
sudo mkdir -p /var/backups/project-name/db
```

Give permission:

```
sudo chown -R youruser:youruser/var/backups/project-name
```

```
sudo chown -R root:root/var/backups/project-name
```

Backup Script for the project

Create a script file:

```
#if user is not root
nano /home/youruser/backup-projectName.sh


#if user is root
nano /root/backup-projectName.sh
```

Paste the following script (edit DB credentials and paths as needed)

```
#!/bin/bash

# Project name for clear folder structure
PROJECT_NAME="project-name"

# Database credentials (Laravel project specific)
DB_NAME="project_db_name"
DB_USER="your_db_user"
DB_PASS="your_db_password"

# Backup directories for DB and storage
BACKUP_DIR="/var/backups/$PROJECT_NAME/db"
# Date format for unique backup filenames
DATE=$(date +\%F_\%T)
# Backup filenames
DB_BACKUP_FILE="$BACKUP_DIR/${DB_NAME}_$DATE.sql.gz"

# Ensure backup directories exist
mkdir -p "$BACKUP_DIR"

# 1. Backup the MySQL database ONLY for this Laravel project
mysqldump -u "$DB_USER" -p"$DB_PASS" "$DB_NAME" | gzip > "$DB_BACKUP

# 3. Delete backups older than 7 days to save space
find "$BACKUP_DIR" -type f -name "*.gz" -mtime +7 -exec rm {} \;

# 4. Upload backups to Google Drive via rclone (if configured)
rclone sync "/var/backups/$PROJECT_NAME" gdrive:backups/$PROJECT_NAME
```

If you want to avoid backing up the database when **no changes have occurred**

```
#!/bin/bash

# Project name for clear folder structure
PROJECT_NAME="project-name"

# Database credentials (Laravel project specific)
DB_NAME="project_db_name"
DB_USER="your_db_user"
DB_PASS="your_db_password"

# Backup directories for DB and storage
BACKUP_DIR="/var/backups/$PROJECT_NAME/db"
# Current timestamp
TIMESTAMP=$(date +"%Y-%m-%d_%H:%M:%S")

# File paths
NEW_BACKUP="$BACKUP_DIR/${DB_NAME}_${TIMESTAMP}.sql.gz"
LAST_BACKUP=$(ls -t $BACKUP_DIR/*.sql.gz | head -n 1)

# Ensure backup directories exist
mkdir -p "$BACKUP_DIR"

# Dump and compress database
mysqldump -u "$DB_USER" -p"$DB_PASS" "$DB_NAME" | gzip > "$NEW_BACK

# === COMPARE with last backup ===
if [ -f "$LAST_BACKUP" ] && cmp -s <(zcat "$NEW_BACKUP") <(zcat "$LAST_B
    echo " No changes in DB since last backup. Removing new file: $NEW_BACKU
    rm "$NEW_BACKUP"
else
    echo " Database has changed. Keeping new backup: $NEW_BACKUP"

    # Optional: keep only last 7 backups locally
    find "$BACKUP_DIR" -type f -name "*.sql.gz" -mtime +7 -delete
```

```
    # Upload to Google Drive
    rclone copy "$NEW_BACKUP" gdrive:/backups/orolajide-db
fi
```

Make the Backup Script Executable

```
#if user is not root
chmod +x /home/youruser/backup-projectName.sh

#if user is root
chmod +x /root/backup-projectName.sh
```

Grant MYSQL `PROCESS` privilege (recommended)

Log in to MySQL as root:

```
GRANT PROCESS ON *.* TO 'your_db_user'@'localhost';
```

```
FLUSH PRIVILEGES;
```

Install and Configure `rclone` for Google Drive Upload

Install
`rclone`

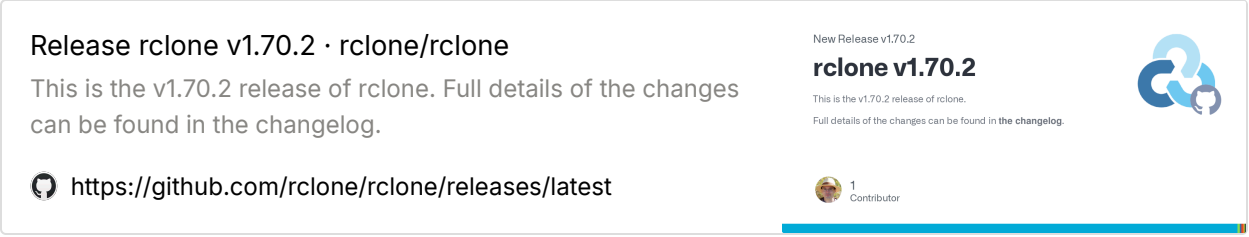
```
curl https://rclone.org/install.sh | sudo bash
```

Configure
`rclone` for Google Drive

```
rclone config
```

- No remotes found, make a new one? → n
- Enter the name of the new remote. name> → gdrive
- Type of storage to configure. Storage> → drive
- client_id> → **Enter** to skip
- client_secret> → **Enter** to skip
- Scope type: scope> → 1 (full access)
- service_account_file> → **Enter** to skip
- Edit advanced config? → n
- Use auto config: → n
- config_token>

→ Quick download of rclone for Windows (64-bit)



→ Find the file named like this and download

rclone-v1.70.2-windows-amd64.zip

→ Extract the ZIP anywhere you want, e.g. `C:\rclone\`

→ Inside the extracted folder, you'll find `rclone.exe`

→ Open **Command Prompt**

→ Change the directory to where you extracted rclone

```
cd C:\rclone
```

→ test

```
rclone.exe version
```

→ Run the exact command your VPS gave you

```
# Here this "eyJzY29wZSI6ImRyaXZlIn0" will be replaced by "YOUR_BASE64_TO

rclone.exe authorize "drive" "eyJzY29wZSI6ImRyaXZlIn0"
```

→ It will open a browser, let you authorize, and then print a successful message

→ Now go to the command prompt again, and you will see a long JSON token.

→ Copy that JSON token and paste it into your VPS prompt at `config_token>`

- Configure this as a Shared Drive (Team Drive). → n
- Keep this "gdrive" remote? → y

Test Google Drive Connection

```
rclone ls gdrive:
```

(Optional) Create the Google Drive folder for backups (only once)

```
rclone mkdir gdrive:backups/project-name
```

Automate Backup Script Using Cron

Edit your cron jobs:

```
crontab -e
```

Add this line to run the backup every day or every hour:

```
#if user is not root

0 2 * * * /home/youruser/backup-projectName.sh

#if user is root
#if want daily at 2 am backup

0 2 * * * /root/backup-projectName.sh


#if want for hourly backup
0 * * * * /root/backup-orolajide.sh
```

Auto-clean older backups

```
find /var/backups/project-name/db -type f -name "*.sql.gz" -mtime +7 -delete
```

Verify Cron Job Installation

```
crontab -l
```

Test the Backup Script Manually

Run it:

```
sudo /root/backup-orolajide.sh
```

You'll see either:

- The database `has changed. Keeping new backup: ...`
- Or `No changes in DB since last backup. Removing new file: ...`

Check your local backup folders:

```
ls -lh /var/backups/project-name/db
```