**Website Health Check Logger**

#!/bin/bash

URL="https://github.com"

LOG\_DIR="$HOME/website\_health\_logs"

mkdir -p "$LOG\_DIR"

TODAY=$(date +"%Y-%m-%d")

LOG\_FILE="$LOG\_DIR/health\_log\_$TODAY.log"

response=$(curl -o /dev/null -s -A "Mozilla/5.0" -w "%{http\_code} %{time\_total}" "$URL")

status\_code=$(echo "$response" | awk '{ print $1 }')

response\_time=$(echo "$response" | awk '{ print $2 }')

timestamp=$(date +"%Y-%m-%d %H:%M:%S")

echo "$timestamp | Status: $status\_code | Response Time: ${response\_time}s" >> "$LOG\_FILE"

**🎯 Purpose:**

This script checks the **HTTP status code** and **response time** of a website (e.g., GitHub) and logs the result with a timestamp into a daily log file.

**🧩 Line-by-line Explanation:**

**✅ 1. Set the target URL**

URL="https://github.com"

→ The website whose health is being monitored.

**✅ 2. Define the log directory**

LOG\_DIR="$HOME/website\_health\_logs"

mkdir -p "$LOG\_DIR"

→ Ensures the log directory exists. $HOME means the current user’s home directory.  
→ Example: /home/ubuntu/website\_health\_logs

**✅ 3. Get today's date**

TODAY=$(date +"%Y-%m-%d")

LOG\_FILE="$LOG\_DIR/health\_log\_$TODAY.log"

→ This creates a unique log file for each day.  
Example: health\_log\_2025-04-03.log

**✅ 4. Send HTTP request**

response=$(curl -o /dev/null -s -A "Mozilla/5.0" -w "%{http\_code} %{time\_total}" "$URL")

* -o /dev/null: Discards actual webpage content.
* -s: Silent mode (no progress bar).
* -A: Sets a browser-style User-Agent (helps avoid blocks).
* -w: Formats curl's output to show only the **status code** and **response time**.

Example Output:

200 0.234

**✅ 5. Extract status code and response time**

status\_code=$(echo "$response" | awk '{ print $1 }')

response\_time=$(echo "$response" | awk '{ print $2 }')

* $1: Status code (200 = OK, 404 = Not found, etc.)
* $2: Time taken for the request (in seconds)

**✅ 6. Get current timestamp and log result**

timestamp=$(date +"%Y-%m-%d %H:%M:%S")

echo "$timestamp | Status: $status\_code | Response Time: ${response\_time}s" >> "$LOG\_FILE"

→ Appends the status and response time to the day's log file.

**📂 Sample Log Output**

2025-04-03 12:01:00 | Status: 200 | Response Time: 0.234s

2025-04-03 12:05:00 | Status: 200 | Response Time: 0.245s

🧠 **Use Case:** You can schedule this with cron to run every 5 minutes and monitor website health over time.

**📦Script 2: Archive Yesterday’s Log**

#!/bin/bash

LOG\_DIR="$HOME/website\_health\_logs"

YESTERDAY=$(date -d "yesterday" +"%Y-%m-%d")

YESTERDAY\_LOG="$LOG\_DIR/health\_log\_$YESTERDAY.log"

YESTERDAY\_ARCHIVE="$LOG\_DIR/health\_log\_$YESTERDAY.tar.gz"

ARCHIVE\_HISTORY="$LOG\_DIR/archive\_history.log"

if [ -f "$YESTERDAY\_LOG" ] && [ ! -f "$YESTERDAY\_ARCHIVE" ]; then

tar -czf "$YESTERDAY\_ARCHIVE" "$YESTERDAY\_LOG"

rm "$YESTERDAY\_LOG"

echo "$(date) | Archived $YESTERDAY\_LOG" >> "$ARCHIVE\_HISTORY"

else

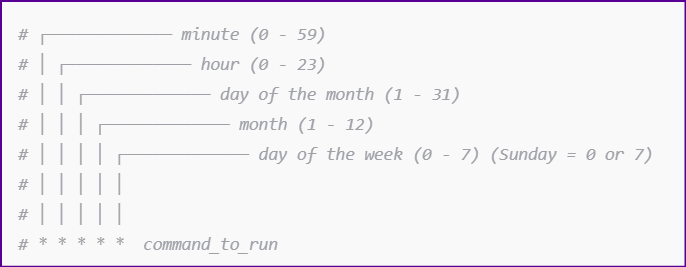
echo "$(date) | Nothing to archive or already archived" >> "$ARCHIVE\_HISTORY"

fi

**For Cron Job** :

**First check time zone**: timedatectl

**Switch to IST/Or ur region**: sudo timedatectl set-timezone Asia/Kolkata



**Cron Job Setup:**

**Crontab -e**

**Add :**

**If u want to run the script at 12:10 AM daily then : 10 0 \* \* \* Path/to/script**

**🎯 Purpose:**

This script:

1. Checks if **yesterday's log file** exists
2. Archives it into .tar.gz format (compressed)
3. Deletes the original .log file
4. Logs this activity in an archive\_history.log file

**🧩 Line-by-line Explanation:**

**✅ 1. Define the directory and date**

LOG\_DIR="$HOME/website\_health\_logs"

YESTERDAY=$(date -d "yesterday" +"%Y-%m-%d")

→ Computes yesterday’s date and uses it to locate the log file.

**✅ 2. Define filenames**

YESTERDAY\_LOG="$LOG\_DIR/health\_log\_$YESTERDAY.log"

YESTERDAY\_ARCHIVE="$LOG\_DIR/health\_log\_$YESTERDAY.tar.gz"

ARCHIVE\_HISTORY="$LOG\_DIR/archive\_history.log"

→ Determines:

* File to archive
* Archive output file
* History tracker file

**✅ 3. Check and archive**

if [ -f "$YESTERDAY\_LOG" ] && [ ! -f "$YESTERDAY\_ARCHIVE" ]; then

→ Only archive **if**:

* The .log file exists
* The .tar.gz version doesn’t already exist

**✅ 4. Archive, remove original, and log**

tar -czf "$YESTERDAY\_ARCHIVE" "$YESTERDAY\_LOG"

rm "$YESTERDAY\_LOG"

echo "$(date) | Archived $YESTERDAY\_LOG" >> "$ARCHIVE\_HISTORY"

→ Compresses the file, deletes the original, and logs the event.

**✅ 5. If no action needed**

else

echo "$(date) | Nothing to archive or already archived" >> "$ARCHIVE\_HISTORY"

→ Ensures you still have a record that the script was executed even if it skipped archiving.

**📂 Sample Archive History Log**

Thu Apr 3 00:01:01 UTC 2025 | Archived /home/ubuntu/website\_health\_logs/health\_log\_2025-04-02.log

**💡 Bonus: Automation Tip with Crontab**

You can automate both scripts like this:

# Every 5 minutes - run health check

\*/5 \* \* \* \* /home/ubuntu/scripts/health\_check.sh

# Every day at 12:01 AM - archive yesterday's log

1 0 \* \* \* /home/ubuntu/scripts/archive\_logs.sh

**✅ Summary**

| **Script** | **Purpose** |
| --- | --- |
| health\_check.sh | Logs website response time + status code into daily logs |
| archive\_logs.sh | Archives and removes yesterday's log, maintains a log history |