# Shell Scripting Assessment Answers – Product Support Team

This document contains model answers (scripts) for the Shell Scripting Assessment. Each answer is a standalone Bash script. Use responsibly in a safe test environment.

## Q1. Log Monitoring

Script (bash):

#!/usr/bin/env bash  
LOG="/var/log/app/app.log"  
ARCHIVE\_DIR="/var/log/app/archive"  
THRESHOLD=$((500 \* 1024 \* 1024)) # 500 MB  
  
mkdir -p "$ARCHIVE\_DIR"  
  
if [ -f "$LOG" ]; then  
 size=$(stat -c%s "$LOG")  
 if [ "$size" -gt "$THRESHOLD" ]; then  
 ts=$(date +%Y%m%d-%H%M%S)  
 mv "$LOG" "$ARCHIVE\_DIR/app.log.$ts"  
 : > "$LOG"  
 echo "Archived $LOG to $ARCHIVE\_DIR/app.log.$ts and created new empty log."  
 else  
 echo "Log size is under threshold."  
 fi  
else  
 echo "Log file not found: $LOG"  
fi

## Q2. Service Status Checker

Script (bash):

#!/usr/bin/env bash  
SERVICES=(nginx mongodb redis)  
LOGFILE="/tmp/service\_status.log"  
touch "$LOGFILE"  
  
all\_up=true  
for s in "${SERVICES[@]}"; do  
 if systemctl is-active --quiet "$s"; then  
 :  
 else  
 echo "$(date +'%F %T') - Service $s is DOWN" | tee -a "$LOGFILE"  
 echo "Service $s is down"  
 all\_up=false  
 fi  
done  
  
if $all\_up; then  
 echo "All services are healthy."  
fi

## Q3. File Integrity Check

Script (bash):

#!/usr/bin/env bash  
FILES=(/etc/passwd /etc/hosts /etc/fstab)  
  
for f in "${FILES[@]}"; do  
 if [ ! -e "$f" ] || [ ! -s "$f" ]; then  
 echo "Critical system file missing or corrupted: $f"  
 exit 1  
 fi  
done  
  
echo "All critical files exist and are non-empty."

## Q4. Disk Space Alert

Script (bash):

#!/usr/bin/env bash  
THRESHOLD=80  
LOGFILE="/tmp/disk\_alert.log"  
mounts=(/ /var)  
  
for m in "${mounts[@]}"; do  
 use=$(df -h "$m" | awk 'NR==2 {print $5}' | sed 's/%//')  
 if [ "$use" -ge "$THRESHOLD" ]; then  
 echo "$(date +'%F %T') - ALERT: $m usage ${use}%" | tee -a "$LOGFILE"  
 else  
 echo "$m: ${use}% (Sufficient disk space)"  
 fi  
done

## Q5. User Login Tracker

Script (bash):

#!/usr/bin/env bash  
read -p "Enter username: " username  
if who | awk '{print $1}' | grep -qw "$username"; then  
 who | awk -v u="$username" '$1==u {print "User logged in since", $3, $4}'  
else  
 echo "User is not currently logged in."  
fi

## Q6. Directory File Count

Script (bash):

#!/usr/bin/env bash  
base=/var/www  
if [ ! -d "$base" ]; then  
 echo "$base does not exist."  
 exit 1  
fi  
  
for d in "$base"/\*/; do  
 [ -d "$d" ] || continue  
 count=$(find "$d" -type f | wc -l)  
 echo "$(basename "$d"): $count files"  
done

## Q7. Backup Rotation

Script (bash):

#!/usr/bin/env bash  
BACKUP\_DIR=/opt/backups  
LOGFILE=/var/log/backup\_rotation.log  
find "$BACKUP\_DIR" -type f -mtime +7 -print0 | while IFS= read -r -d '' f; do  
 rm -f "$f" && echo "$(date +'%F %T') Deleted: $f" >> "$LOGFILE"  
done

## Q8. Application Log Search

Script (bash):

#!/usr/bin/env bash  
KEYFILE=./keywords.txt  
LOG=/var/log/app.log  
  
if [ ! -f "$KEYFILE" ]; then  
 echo "Keyword file $KEYFILE not found."  
 exit 1  
fi  
  
while IFS= read -r key; do  
 [ -z "$key" ] && continue  
 cnt=$(grep -i "$key" "$LOG" 2>/dev/null | wc -l)  
 echo "Keyword '$key' occurrences: $cnt"  
done < "$KEYFILE"

## Q9. CPU Usage Monitoring

Script (bash):

#!/usr/bin/env bash  
checks=6  
interval=10  
threshold=85  
  
for ((i=1;i<=checks;i++)); do  
 cpu\_idle=$(top -bn1 | grep "%Cpu(s)" | awk -F',' '{print $4}' | awk '{print $1}')  
 cpu\_usage=$(awk -v idle="$cpu\_idle" 'BEGIN{printf("%.0f", 100 - idle)}')  
 echo "Check $i: CPU usage ${cpu\_usage}%"  
 if [ "$cpu\_usage" -gt "$threshold" ]; then  
 echo "High CPU load detected!"  
 exit 0  
 fi  
 sleep $interval  
done  
echo "CPU usage within limits."

## Q10. Server Health Report

Script (bash):

#!/usr/bin/env bash  
LOG=/tmp/server\_health\_report.log  
count=6  
i=0  
while [ $i -lt $count ]; do  
 ts=$(date +'%F %T')  
 cpu=$(top -bn1 | grep "%Cpu(s)" | sed 's/.\*, \*\([0-9.]\*\)%\* id.\*/\1/' | awk '{print "idle="$1}')  
 mem=$(free -h | awk 'NR==2{print $3"/"$2}')  
 disk=$(df -h / | awk 'NR==2{print $5}')  
 echo "$ts | CPU: $cpu | MEM: $mem | DISK\_ROOT: $disk" >> "$LOG"  
 i=$((i+1))  
 sleep 3600  
done  
echo "Generated $count reports to $LOG"

## Q11. Database Connection Retry

Script (bash):

#!/usr/bin/env bash  
tries=5  
wait\_sec=10  
i=1  
while [ $i -le $tries ]; do  
 if mysql -h localhost -u root -p'yourpassword' -e 'SELECT 1' &>/dev/null; then  
 echo "DB connected on attempt $i"  
 exit 0  
 else  
 echo "Attempt $i failed, retrying in $wait\_sec seconds..."  
 i=$((i+1))  
 sleep $wait\_sec  
 fi  
done  
echo "Database connection failed after $tries attempts."  
exit 1

## Q12. Web Application Health Check

Script (bash):

#!/usr/bin/env bash  
url="https://app.example.com"  
max=3  
i=1  
while [ $i -le $max ]; do  
 if curl -s -o /dev/null -w "%{http\_code}" "$url" | grep -q "200"; then  
 echo "Site reachable on attempt $i"  
 exit 0  
 fi  
 echo "Attempt $i failed"  
 i=$((i+1))  
 sleep 2  
done  
# send mail (assumes mailx configured)  
echo "Subject: App down" | sendmail -v support@example.com  
echo "Alert sent to support team."

## Q13. Log File Cleaner

Script (bash):

#!/usr/bin/env bash  
SRC=/var/logs/app  
DEST=/var/logs/archive  
mkdir -p "$DEST"  
find "$SRC" -type f -mtime +3 -name "\*.log" -print0 | while IFS= read -r -d '' f; do  
 gzip -c "$f" > "$DEST/$(basename "$f").gz" && rm -f "$f" && echo "Archived $f"  
done

## Q14. Process Auto-Restarter

Script (bash):

#!/usr/bin/env bash  
proc\_name="payment-service"  
log=/var/log/payment-service-restarter.log  
  
while true; do  
 if pgrep -f "$proc\_name" >/dev/null; then  
 sleep 5  
 continue  
 else  
 echo "$(date +'%F %T') - $proc\_name not running, attempting restart" | tee -a "$log"  
 systemctl start $proc\_name || /usr/bin/$proc\_name &  
 sleep 5  
 fi  
done

## Q15. Configuration Drift Detector

Script (bash):

#!/usr/bin/env bash  
A=/etc/app/config.conf  
B=/opt/backup/config.conf  
  
if [ ! -f "$A" ] || [ ! -f "$B" ]; then  
 echo "One of the files is missing."  
 exit 1  
fi  
  
if ! diff -q "$A" "$B" >/dev/null; then  
 echo "Configuration mismatch detected."  
else  
 echo "Configuration is consistent."  
fi

## Q16. Environment Validator

Script (bash):

#!/usr/bin/env bash  
PASS=0; FAIL=0  
echo "Environment validation report:"  
ram\_kb=$(grep MemTotal /proc/meminfo | awk '{print $2}')  
ram\_mb=$((ram\_kb/1024))  
if [ "$ram\_mb" -ge 4096 ]; then echo "RAM >=4GB: PASS"; PASS=$((PASS+1)); else echo "RAM >=4GB: FAIL ($ram\_mb MB)"; FAIL=$((FAIL+1)); fi  
avail\_kb=$(df --output=avail /var | tail -1)  
avail\_mb=$((avail\_kb/1024))  
if [ "$avail\_mb" -ge 20000 ]; then echo "/var >=20GB: PASS"; PASS=$((PASS+1)); else echo "/var >=20GB: FAIL ($avail\_mb MB)"; FAIL=$((FAIL+1)); fi  
for b in docker curl git; do  
 if command -v $b >/dev/null 2>&1; then echo "$b: PASS"; PASS=$((PASS+1)); else echo "$b: FAIL"; FAIL=$((FAIL+1)); fi  
done  
echo "Summary: PASS=$PASS, FAIL=$FAIL"

## Q17. Auto-Archive Tool

Script (bash):

#!/usr/bin/env bash  
SRC=/opt/projects  
DEST=/opt/archives  
mkdir -p "$DEST"  
archived=0  
for d in "$SRC"/\*/; do  
 [ -d "$d" ] || continue  
 if [ $(find "$d" -maxdepth 0 -mtime +30 | wc -l) -gt 0 ]; then  
 tar -czf "$DEST/$(basename "$d")-$(date +%Y%m%d).tar.gz" -C "$SRC" "$(basename "$d")" && archived=$((archived+1)) && echo "Archived $d"  
 fi  
done  
echo "Archived projects: $archived"

## Q18. Log Analyzer with Loop

Script (bash):

#!/usr/bin/env bash  
DIR=/var/logs/app  
count=0  
for f in "$DIR"/\*.log; do  
 [ -f "$f" ] || continue  
 if grep -q "ERROR" "$f"; then  
 count=$((count+1))  
 fi  
done  
echo "Number of log files containing ERROR: $count"

## Q19. API Response Validator

Script (bash):

#!/usr/bin/env bash  
url="http://internal.api/health"  
fail=0  
for i in {1..5}; do  
 code=$(curl -s -o /dev/null -w "%{http\_code}" "$url")  
 if [ "$code" -ne 200 ]; then  
 fail=1  
 echo "Request $i failed with code $code"  
 fi  
done  
if [ $fail -eq 0 ]; then echo "API stable."; else echo "API instability detected."; fi

## Q20. Automated Cleanup

Script (bash):

#!/usr/bin/env bash  
DIR=/tmp  
LOG=/tmp/cleanup\_report.log  
find "$DIR" -type f -mtime +2 ! -name "\*.lock" -print0 | while IFS= read -r -d '' f; do  
 rm -f "$f" && echo "$(date +'%F %T') Deleted $f" >> "$LOG"  
done  
echo "Cleanup complete. See $LOG"