OpenAI

Bash scripting interview questions

give me list of important bash scripting on linux interview questions and answers which are most frequently asked in interviews

Absolutely, Sandy ✅ — here's a **complete and practical list of the most frequently asked Bash scripting questions and answers** for Linux / SRE / DevOps interviews.
These are grouped by **difficulty level (basic → advanced)** and focus on **real-world and scenario-based concepts** that hiring panels test for.

🟢 Basic Bash Scripting Interview Questions

1. What is a Bash script?

Answer:
A Bash script is a text file containing a series of commands that are executed sequentially by the Bash shell.
It allows automation of repetitive tasks like deployments, backups, system monitoring, etc.

Example:

```
bash

#!/bin/bash
echo "Hello, $USER"
```

---

2. How do you make a Bash script executable?

Answer:

```
bash

chmod +x script.sh
./script.sh
```

Or run directly:

```
bash

bash script.sh
```

---

3. What is `#!/bin/bash` (shebang)?

Answer:
It tells the system which interpreter to use to execute the script.

E.g., **`/bin/bash`** means the script should run in Bash.

---

4. How do you take user input in a script?

Answer:

```bash
read -p "Enter your name: " name
echo "Hello, $name"
```

---

5. How do you pass arguments to a script?

Answer:

```bash
#!/bin/bash
echo "First arg: $1"
echo "Second arg: $2"
echo "Total args: $#"
```

- **`$0`** → script name
- **`$#`** → number of arguments
- **`$@`** → all arguments as separate words

---

6. How do you check if a file exists?

Answer:

```bash
if [ -f /tmp/test.txt ]; then
  echo "File exists"
else
  echo "File does not exist"
fi
```

Common checks:

- **`-f`** → regular file
- **`-d`** → directory
- **`-e`** → file/directory exists

---

7. How to check if two strings are equal?

Answer:

```bash
if [ "$str1" = "$str2" ]; then
  echo "Equal"
else
  echo "Not equal"
fi
```

---

8. How to use loops in Bash?

For loop:

```bash
for i in 1 2 3 4 5; do
  echo "Number $i"
done
```

While loop:

```bash
i=1
while [ $i -le 5 ]; do
  echo "Count $i"
  ((i++))
done
```

---

9. What is the difference between `"` and `'` in Bash?

Answer:

- **Double quotes** (`" "`) allow variable and command substitution.
- **Single quotes** (`' '`) treat everything literally.

Example:

```bash
name="Sandy"
echo "Hi $name"   # → Hi Sandy
echo 'Hi $name'   # → Hi $name
```

---

10. What does `$?` mean?

Answer:

It stores the **exit status** of the last command.

- `0` → success
- Non-zero → failure

Example:

```bash
ls /tmp
echo $?
```

🟡 Intermediate Bash Scripting Questions

11. What is command substitution in Bash?

Answer:
Used to capture command output into a variable.

```bash
today=$(date)
echo "Today is $today"
```

or

```bash
today=`date`
```

---

12. What is the difference between `>` and `>>`?

Answer:

- `>` → overwrite file
- `>>` → append to file

Example:

```bash
echo "Hello" > file.txt
echo "World" >> file.txt
```

---

13. How to use `if-elif-else` in Bash?

```bash
if [ $num -gt 10 ]; then
  echo "Greater"
elif [ $num -eq 10 ]; then
  echo "Equal"
else
  echo "Smaller"
fi
```

---

14. How do you schedule a script using `cron`?

Answer:
Edit cron jobs using:

```bash
crontab -e
```

Example entry:

```bash
0 2 * * * /home/sandy/backup.sh
```

(Runs every day at 2 AM)

---

15. How to check for root user in a script?

```bash
if [ "$EUID" -ne 0 ]; then
  echo "Please run as root"
  exit 1
fi
```

---

16. How to perform arithmetic operations in Bash?

```bash
a=10
b=5
sum=$((a + b))
echo "Sum: $sum"
```

---

17. How to debug a Bash script?

Answer:
Run with debug mode:

```bash
bash -x script.sh
```

Or include inside script:

```bash
set -x   # start debugging
set +x   # stop debugging
```

18. What are functions in Bash?

Answer:
Reusable code blocks inside a script.

```
bash

greet() {
  echo "Hello, $1"
}

greet "Sandy"
```

---

19. How to handle errors gracefully in Bash?

Answer:
Use **`set -e`** to stop script on error.

```
bash

set -e
cp file1.txt /tmp/
echo "Copied successfully"
```

Or manually:

```
bash

if ! cp file1.txt /tmp/; then
  echo "Copy failed!"
  exit 1
fi
```

---

20. How to use case statements in Bash?

```
bash

read -p "Enter choice: " choice
case $choice in
  start) echo "Starting service";;
  stop) echo "Stopping service";;
  *) echo "Invalid choice";;
esac
```

🔴 Advanced / Real-World Bash Interview Questions

21. How do you process a file line by line in Bash?

```
bash

while IFS= read -r line; do
  echo "Line: $line"
```

```
done < /path/to/file.txt
```

---

22. How to check disk usage and alert if above threshold?

```
bash

#!/bin/bash
THRESHOLD=80
USAGE=$(df / | awk 'NR==2 {print $5}' | sed 's/%//')

if [ $USAGE -gt $THRESHOLD ]; then
  echo "Disk usage critical: $USAGE%"
fi
```

---

23. How do you find and delete files older than 30 days?

```
bash

find /var/log -type f -mtime +30 -exec rm -f {} \;
```

---

24. What's the difference between `&&` and `;` in Bash?

Answer:

- **`cmd1 && cmd2`** → Run **`cmd2`** **only if** **`cmd1`** succeeds.
- **`cmd1 ; cmd2`** → Run both regardless of success.

---

25. What is the use of `trap` command?

Answer:
Used to catch signals and execute cleanup code.

Example:

```
bash

trap "echo 'Script interrupted'; exit" SIGINT SIGTERM
```

---

26. What is the difference between `source` and `./`?

Answer:

- **`source script.sh`** → runs in the **current shell**.
- **`./script.sh`** → runs in a **subshell**.

---

27. How to export a variable for child processes?

```bash
export VAR="value"
```

---

28. How do you remove duplicate lines in a file using Bash?

```bash
sort file.txt | uniq > newfile.txt
```

---

29. How to find top 5 memory-consuming processes?

```bash
ps -eo pid,comm,%mem --sort=-%mem | head -n 6
```

---

30. Write a script to monitor a service and restart if down.

```bash
#!/bin/bash
SERVICE="nginx"
if ! systemctl is-active --quiet $SERVICE; then
  systemctl restart $SERVICE
  echo "$SERVICE restarted at $(date)" >> /var/log/service_monitor.log
fi
```

🧠 Pro Tips for Interviewers' Follow-ups

- They often ask you to **debug** or **optimize** a script on the spot.
- Be comfortable with **`awk`**, **`sed`**, **`grep`**, **`cut`**, **`tr`**, and **`xargs`**.
- Know how to use **loops + conditions + command substitution** together.
- Expect "What happens if…" questions (error handling, permissions, bad input, etc.)

Would you like me to create a **PDF or cheat sheet** version of this list (with syntax, output, and real-world examples for each)?
It's great to quickly revise before interviews.

more questions like last one

Excellent 👍 — you're thinking like a **real SRE/DevOps engineer**, Sandy.
Let's now go **deeper into scenario-based, practical Bash scripting questions** — like the "monitor a

service" one — which are the type you'll actually face in interviews for **SRE, DevOps, or Infrastructure roles**.

Below are **real-world Bash scripting interview questions (31–70)** with **answers and explanations**, focused on monitoring, automation, and troubleshooting.

🔥 Advanced & Scenario-Based Bash Scripting Questions

31. Write a script to check if a process is running, and start it if not.

```
bash

#!/bin/bash
PROCESS="sshd"
if ! pgrep -x "$PROCESS" >/dev/null; then
  echo "$PROCESS not running, starting it..."
  systemctl start $PROCESS
else
  echo "$PROCESS is running."
fi
```

✅ *Checks for a process by exact name and auto-restarts if down.*

---

32. Script to find the top 5 CPU-consuming processes.

```
bash

ps -eo pid,comm,%cpu --sort=-%cpu | head -n 6
```

✅ Often used in troubleshooting CPU spikes.

---

33. Write a script to get IP address of the system.

```
bash

hostname -I | awk '{print $1}'
```

or for specific interface:

```
bash

ip addr show eth0 | grep "inet " | awk '{print $2}' | cut -d/ -f1
```

---

34. Script to count number of logged-in users.

```
bash

who | wc -l
```

✅ Can be extended to alert if user count crosses a limit.

35. Write a script to monitor memory usage and alert if >80%.

```bash
#!/bin/bash
THRESHOLD=80
USED=$(free | awk '/Mem/{printf("%.0f"), $3/$2*100}')

if [ $USED -gt $THRESHOLD ]; then
  echo "High memory usage: $USED%"
fi
```

---

36. Write a script to ping a list of hosts and report their status.

```bash
#!/bin/bash
for host in $(cat hosts.txt); do
  if ping -c 1 $host &>/dev/null; then
    echo "$host is reachable"
  else
    echo "$host is down"
  fi
done
```

---

37. Script to take a backup of `/etc` daily with timestamp.

```bash
#!/bin/bash
BACKUP_DIR="/backup"
DATE=$(date +%F)
tar -czf $BACKUP_DIR/etc_backup_$DATE.tar.gz /etc
```

---

38. How to write a script to clean `/tmp` files older than 7 days?

```bash
find /tmp -type f -mtime +7 -exec rm -f {} \;
```

---

39. Script to count number of files and directories in current folder.

```bash
#!/bin/bash
echo "Files: $(find . -type f | wc -l)"
echo "Directories: $(find . -type d | wc -l)"
```

---

40. Script to check HTTP status code of a website.

```bash
#!/bin/bash
URL="https://google.com"
STATUS=$(curl -o /dev/null -s -w "%{http_code}" $URL)
echo "HTTP Status: $STATUS"
```

---

41. Script to read a file line by line and print with line number.

```bash
#!/bin/bash
count=1
while IFS= read -r line; do
  echo "$count: $line"
  ((count++))
done < input.txt
```

---

42. Write a script to display files larger than 100MB.

```bash
find /var -type f -size +100M -exec ls -lh {} \;
```

---

43. Script to extract unique IP addresses from a log file.

```bash
grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" access.log | sort -u
```

---

44. Script to monitor system load average.

```bash
#!/bin/bash
LOAD=$(uptime | awk -F'load average:' '{print $2}' | cut -d, -f1)
echo "Load average: $LOAD"
```

---

45. How to send an email alert from Bash when disk usage >90%.

```bash
#!/bin/bash
THRESHOLD=90
EMAIL="admin@example.com"

USAGE=$(df / | awk 'NR==2 {print $5}' | sed 's/%//')
if [ $USAGE -gt $THRESHOLD ]; then
  echo "Disk usage is ${USAGE}%" | mail -s "Disk Alert" $EMAIL
fi
```

✅ Requires `mail` command configured.

---

46. Script to archive logs older than 5 days.

```
bash

find /var/log -type f -mtime +5 -exec gzip {} \;
```

---

47. Script to rename all `.txt` files to `.bak`.

```
bash

for file in *.txt; do
  mv "$file" "${file%.txt}.bak"
done
```

---

48. Script to find the number of running Docker containers.

```
bash

docker ps -q | wc -l
```

---

49. Script to check the health of all Kubernetes pods.

```
bash

kubectl get pods --all-namespaces --field-selector=status.phase!=Running
```

(You can wrap this into a Bash script to alert when non-running pods are found.)

---

50. Write a script to monitor a log file in real time for errors.

```
bash

tail -Fn0 /var/log/syslog | while read line; do
  echo "$line" | grep "ERROR" && echo "Error found: $line"
done
```

---

51. Script to validate if a given port is open.

```
bash

#!/bin/bash
HOST=localhost
PORT=22

if nc -zv $HOST $PORT 2>/dev/null; then
  echo "Port $PORT is open"
```

```
else
  echo "Port $PORT is closed"
fi
```

52. Script to compress and remove old logs automatically.

```bash
#!/bin/bash
LOG_DIR="/var/log"
find $LOG_DIR -type f -name "*.log" -mtime +10 -exec gzip {} \; -exec rm {} \;
```

53. Script to monitor failed SSH login attempts.

```bash
grep "Failed password" /var/log/auth.log | awk '{print $11}' | sort | uniq -c | sort -nr
```

54. Script to calculate total disk usage per directory.

```bash
du -h --max-depth=1 / | sort -hr
```

55. Script to check uptime of multiple servers.

```bash
for host in $(cat servers.txt); do
  ssh $host uptime
done
```

56. Script to check SSL certificate expiry date.

```bash
#!/bin/bash
domain="google.com"
echo | openssl s_client -servername $domain -connect $domain:443 2>/dev/null \
| openssl x509 -noout -dates
```

57. Script to get the last login details of all users.

```bash
lastlog
```

58. Script to extract failed services from systemctl.

```
bash

systemctl list-units --type=service --state=failed
```

---

59. Script to list all users on the system.

```
bash

cut -d: -f1 /etc/passwd
```

---

60. Script to monitor high load and send an alert.

```
bash

#!/bin/bash
LOAD=$(awk '{print $1}' /proc/loadavg)
THRESHOLD=2.00

if (( $(echo "$LOAD > $THRESHOLD" | bc -l) )); then
  echo "High Load Detected: $LOAD"
fi
```

---

61. Script to count total number of open network connections.

```
bash

netstat -tun | tail -n +3 | wc -l
```

---

62. Script to extract specific column from CSV file.

```
bash

awk -F, '{print $2}' file.csv
```

---

63. Script to get list of users with sudo access.

```
bash

grep '^sudo:.*$' /etc/group
```

---

64. Script to print system summary (CPU, Mem, Disk).

```bash
echo "CPU: $(top -bn1 | grep 'Cpu(s)' | awk '{print $2 + $4}')%"
echo "Mem: $(free -m | awk '/Mem/{print $3"/"$2" MB"}')"
echo "Disk: $(df -h / | awk 'NR==2 {print $5}')"
```

---

65. Script to restart a service if it fails 3 times consecutively.

```bash
#!/bin/bash
SERVICE="nginx"
FAIL_COUNT=0

for i in {1..3}; do
  if ! systemctl is-active --quiet $SERVICE; then
    ((FAIL_COUNT++))
    sleep 10
  fi
done

if [ $FAIL_COUNT -eq 3 ]; then
  systemctl restart $SERVICE
  echo "$(date): Restarted $SERVICE" >> /tmp/restart.log
fi
```

---

66. Script to check which ports are listening on the server.

```bash
ss -tuln
```

---

67. Script to validate if a file is empty.

```bash
if [ ! -s file.txt ]; then
  echo "File is empty"
fi
```

---

68. Script to monitor number of failed pods in Kubernetes.

```bash
#!/bin/bash
FAILED=$(kubectl get pods --all-namespaces --field-selector=status.phase!=Running | wc -l)
if [ $FAILED -gt 0 ]; then
  echo "Warning: $FAILED pods not running"
fi
```

---

69. Script to generate a system health report.

```
bash

#!/bin/bash
echo "===== System Health Report ====="
date
echo "Uptime: $(uptime -p)"
echo "CPU Load: $(uptime | awk -F'load average:' '{print $2}')"
echo "Memory Usage:"
free -h
echo "Disk Usage:"
df -h
```

---

70. Script to find the top 10 largest files on the system.

```
bash

find / -type f -exec du -h {} + 2>/dev/null | sort -hr | head -n 10
```

Would you like me to continue with **30 more expert-level Bash questions (71–100)** — covering:

- File parsing (awk/sed/grep mix)
- Log monitoring automation
- Parallel processing
- Script optimization and debugging
- Integration with systemd, cron, and Jenkins?

I can also format it as a **Bash Interview Handbook (PDF)** with explanations and outputs if you want to revise quickly.

yes

Perfect, Sandy ✅ — let's dive into the **Expert-Level (71–100)** Bash scripting interview questions that senior DevOps / SRE roles often test.

These questions go beyond syntax — they're **scenario-driven**, focusing on **real-world automation, system troubleshooting, and efficiency**.

🚀 Expert-Level Bash Scripting Interview Questions (71–100)

---

71. Script to find the top 10 IPs making the most requests in an Apache log.

```
bash

#!/bin/bash
awk '{print $1}' /var/log/httpd/access.log | sort | uniq -c | sort -nr | head -10
```

✅ *Common in web troubleshooting to find abusive IPs.*

---

72. How to extract HTTP 500 errors from logs and count them.

```bash
grep " 500 " /var/log/httpd/access.log | wc -l
```

✅ *Used for analyzing web server health.*

---

73. Script to check if multiple services are active (multi-service check).

```bash
#!/bin/bash
SERVICES="nginx docker sshd"

for svc in $SERVICES; do
  if systemctl is-active --quiet $svc; then
    echo "$svc is running"
  else
    echo "$svc is NOT running"
  fi
done
```

---

74. Script to generate a CSV report of CPU, Memory, and Disk usage.

```bash
#!/bin/bash
echo "DATE,CPU(%),MEM(%),DISK(%)" > /tmp/system_report.csv
CPU=$(top -bn1 | awk '/Cpu/ {print $2 + $4}')
MEM=$(free | awk '/Mem/ {printf("%.2f"), $3/$2*100}')
DISK=$(df / | awk 'NR==2 {print $5}' | sed 's/%//')
echo "$(date +%F),$CPU,$MEM,$DISK" >> /tmp/system_report.csv
```

---

75. How to write a script that runs multiple commands in parallel.

```bash
#!/bin/bash
(cmd1 &)
(cmd2 &)
(cmd3 &)
wait
echo "All commands completed"
```

✅ *Used to speed up multi-host or multi-step operations.*

---

76. Script to monitor SSH connectivity to a remote server.

```bash
#!/bin/bash
HOST="server1"
if timeout 5 ssh -q $HOST exit; then
  echo "$HOST reachable"
else
  echo "$HOST unreachable"
fi
```

---

77. Script to automatically delete empty directories.

```bash
find /path/to/dir -type d -empty -delete
```

---

78. How to check and log the top 5 memory-consuming processes every 5 mins.

```bash
#!/bin/bash
while true; do
  date >> /var/log/mem_usage.log
  ps -eo pid,comm,%mem --sort=-%mem | head -n 6 >> /var/log/mem_usage.log
  echo "----" >> /var/log/mem_usage.log
  sleep 300
done
```

---

79. Script to check which users have passwordless sudo.

```bash
grep -r 'NOPASSWD' /etc/sudoers /etc/sudoers.d/
```

---

80. Script to rotate logs manually (without logrotate).

```bash
#!/bin/bash
LOGFILE="/var/log/myapp.log"
if [ -f $LOGFILE ]; then
  mv $LOGFILE ${LOGFILE}_$(date +%F)
  touch $LOGFILE
fi
```

---

81. Script to list failed systemd services and restart them automatically.

```bash
#!/bin/bash
FAILED=$(systemctl list-units --type=service --state=failed --no-pager | awk '{print $1}' | grep
.)
for svc in $FAILED; do
  echo "Restarting $svc..."
  systemctl restart $svc
done
```

---

82. Script to check total number of running containers across servers.

```bash
for host in $(cat servers.txt); do
  echo -n "$host: "
  ssh $host "docker ps -q | wc -l"
done
```

---

83. Script to get list of expired SSL certs (automated check).

```bash
#!/bin/bash
for domain in $(cat domains.txt); do
  EXPIRY=$(echo | openssl s_client -connect $domain:443 2>/dev/null \
           | openssl x509 -noout -enddate | cut -d= -f2)
  EXPIRE_DATE=$(date -d "$EXPIRY" +%s)
  TODAY=$(date +%s)
  DAYS_LEFT=$(( (EXPIRE_DATE - TODAY) / 86400 ))
  if [ $DAYS_LEFT -lt 10 ]; then
    echo "Certificate for $domain expires in $DAYS_LEFT days"
  fi
done
```

---

84. Script to identify zombie processes.

```bash
ps aux | awk '{if ($8=="Z") print $2, $11}'
```

---

85. Script to test database connection.

```bash
#!/bin/bash
if mysql -h dbhost -u user -p'password' -e "quit" &>/dev/null; then
  echo "Database reachable"
else
  echo "Database connection failed"
fi
```

---

86. Script to monitor system uptime and reboot count.

```bash
echo "Uptime: $(uptime -p)"
echo "Reboots: $(last reboot | wc -l)"
```

---

87. Script to detect files with permissions 777.

```bash
find / -type f -perm 0777 2>/dev/null
```

✅ *Security compliance question.*

---

88. Script to monitor number of established TCP connections.

```bash
netstat -ant | grep ESTABLISHED | wc -l
```

---

89. Script to kill a process consuming more than 80% CPU.

```bash
ps -eo pid,%cpu --sort=-%cpu | awk '$2>80 {print $1}' | xargs -r kill -9
```

---

90. Script to continuously monitor file changes in a directory.

```bash
inotifywait -m /var/log -e create,delete,modify
```

✅ *Requires* ***`inotify-tools`*** *package.*

---

91. Script to validate JSON file syntax.

```bash
jq empty file.json && echo "Valid JSON" || echo "Invalid JSON"
```

✅ *Interviewers love JSON parsing questions.*

---

92. Script to list all users who can SSH into the server.

```
bash

grep -E "AllowUsers|AllowGroups" /etc/ssh/sshd_config
```

---

93. Script to measure script execution time.

```
bash

#!/bin/bash
START=$(date +%s)
# your commands
sleep 3
END=$(date +%s)
echo "Execution time: $((END - START)) seconds"
```

---

94. Script to compare two files and print differences.

```
bash

diff file1.txt file2.txt
```

---

95. Script to fetch public IP of the system.

```
bash

curl -s ifconfig.me
```

---

96. Script to automatically push logs to S3.

```
bash

#!/bin/bash
aws s3 cp /var/log/myapp.log s3://mybucket/logs/$(date +%F)/
```

---

97. Script to extract the top 5 largest directories.

```
bash

du -h --max-depth=1 / | sort -hr | head -n 5
```

---

98. Script to detect login failures in last 24 hours.

```
bash

grep "$(date --date='1 day ago' '+%b %_d')" /var/log/auth.log | grep "Failed password"
```

99. Script to check Kubernetes node resource usage (via kubectl).

```
bash

kubectl top nodes | awk '{print $1, $2, $3}'
```

---

100. Script to send Slack alert from Bash.

```
bash

#!/bin/bash
WEBHOOK_URL="https://hooks.slack.com/services/XXXX/XXXX/XXXX"
MESSAGE="🚨 High CPU detected on $(hostname)"
curl -X POST -H 'Content-type: application/json' \
--data "{\"text\":\"$MESSAGE\"}" $WEBHOOK_URL
```

✅ *Used in SRE automation for real-time notifications.*

🧠 Bonus: Interviewer Follow-Up Tips

Interviewers might ask:

- How would you handle **error logging** for these scripts?
- How to **schedule** them with **`cron`** or **`systemd timers`**?
- How to **parameterize** scripts for multiple environments (prod/dev)?
- How to **combine** Bash with **Python / Ansible / Terraform** pipelines?

Would you like me to now:

1. 📘 Compile all **100 Bash scripting Q&As** (with categorized topics, syntax highlighting, and notes) into a **ready-to-revise PDF/cheat sheet**?
2. Or 🧩 add a **"Hands-on Challenge Section"** (10 real test problems with expected outputs) — similar to what big companies like **Apple, Amgen, or ServiceNow** ask in SRE rounds?

2nd

Awesome choice, Sandy ⚡ — the **Hands-On Bash Scripting Challenge Section** will train you for **real SRE / DevOps interviews** at companies like **Apple, ServiceNow, and Amgen**.

These are **practical challenges** where you're given a situation and asked to **write, debug, or explain a Bash script** live.
Each challenge includes:

- **Scenario / Objective**
- **Expected Approach**

- **Sample / Reference Solution**
- **Follow-up variations** interviewers often throw in.

🧩 Hands-On Bash Scripting Challenge Section (10 Practical Problems)

Challenge 1 — Log File Alerting

Scenario:
You have a continuously growing log file `/var/log/app.log`.
Write a script to monitor it in real-time and alert when it contains the keyword `ERROR`.

Expected Approach:

- Use `tail -F` to stream log lines.
- Use `grep` or pattern matching to filter errors.
- Print timestamp and line when error found.

Solution:

```bash
#!/bin/bash
tail -Fn0 /var/log/app.log | \
while read line; do
  echo "$line" | grep -q "ERROR" && \
  echo "$(date): ERROR found - $line"
done
```

Follow-up:
➡️ Add an email or Slack notification when error count exceeds 10 in 5 minutes.

Challenge 2 — Disk Usage Health Check

Scenario:
Write a script to check disk usage of all mounted partitions and alert if any exceeds 80%.

Expected Approach:

- Use `df -h`
- Loop through filesystems and check the usage percentage.

Solution:

```bash
#!/bin/bash
THRESHOLD=80
df -h | awk 'NR>1 {print $1,$5}' | while read fs usage; do
  usep=${usage%\%}
  if [ $usep -ge $THRESHOLD ]; then
```

```
    echo "Warning: $fs usage at ${usep}%"
  fi
done
```

Follow-up:
➡️ Modify to send alert emails to the admin team.
➡️ Add log rotation to track daily usage trends.

Challenge 3 — Application Restart Automation

Scenario:
You manage a service (`nginx`).
If it’s **not running**, restart it and log the event.

Solution:

```
bash

#!/bin/bash
SERVICE="nginx"
if ! systemctl is-active --quiet $SERVICE; then
  systemctl restart $SERVICE
  echo "$(date): $SERVICE restarted" >> /var/log/service_health.log
fi
```

Follow-up:
➡️ Add logic to count how many times it failed in the last hour.

Challenge 4 — Find Top Talkers (Network)

Scenario:
Identify top 5 IP addresses consuming the most bandwidth using `netstat`.

Solution:

```
bash

#!/bin/bash
netstat -ntu | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | sort -nr | head -5
```

Follow-up:
➡️ Extend this to monitor live connections using `ss` command.

Challenge 5 — Log Analysis with Date Filter

Scenario:
Extract all failed SSH logins from `/var/log/auth.log` in the last 24 hours.

Solution:

```bash
#!/bin/bash
grep "$(date --date='1 day ago' '+%b %_d')" /var/log/auth.log | grep "Failed password"
```

Follow-up:
➡️ Count per IP and block suspicious ones using `iptables`.

Challenge 6 — File Backup and Retention

Scenario:
Write a script that:

1. Creates a tar backup of `/etc/`
2. Stores it in `/backup/` with a timestamp
3. Deletes old backups older than 7 days

Solution:

```bash
#!/bin/bash
BACKUP_DIR="/backup"
DATE=$(date +%F)
tar -czf $BACKUP_DIR/etc_$DATE.tar.gz /etc
find $BACKUP_DIR -type f -mtime +7 -delete
```

Follow-up:
➡️ Add MD5 checksum verification post-backup.
➡️ Send completion status to monitoring system.

Challenge 7 — Dynamic Inventory for Ansible

Scenario:
Generate a list of reachable hosts from `hosts.txt` and output a JSON inventory.

Solution:

```bash
#!/bin/bash
echo '{ "all": { "hosts": ['
for host in $(cat hosts.txt); do
  ping -c1 -W1 $host &>/dev/null && echo "\"$host\","
done | sed '$ s/,$//'
echo '] } }'
```

Follow-up:
➡️ Integrate this with `ansible-playbook -i inventory.sh play.yml`

Challenge 8 — CPU and Memory Snapshot

Scenario:

Capture a snapshot of top 5 CPU and memory consuming processes and store it in `/var/log/perf.log`.

Solution:

```
bash

#!/bin/bash
{
  echo "===== $(date) ====="
  echo "Top 5 CPU processes:"
  ps -eo pid,comm,%cpu --sort=-%cpu | head -n 6
  echo "Top 5 MEM processes:"
  ps -eo pid,comm,%mem --sort=-%mem | head -n 6
  echo
} >> /var/log/perf.log
```

Follow-up:

➡️ Add loop to collect data every 10 mins.
➡️ Convert to CSV format for visualization.

Challenge 9 — Log Error Rate Monitoring

Scenario:

Your application logs contain “ERROR” and “INFO” entries.
Calculate error percentage in the last 1000 log lines.

Solution:

```
bash

#!/bin/bash
LOG="/var/log/app.log"
TOTAL=$(tail -n 1000 $LOG | wc -l)
ERRORS=$(tail -n 1000 $LOG | grep -c "ERROR")
RATE=$((ERRORS * 100 / TOTAL))
echo "Error rate: $RATE%"
```

Follow-up:

➡️ Send Slack alert if error rate > 20%.
➡️ Extend to multiple log files.

Challenge 10 — Self-Healing Cron Job

Scenario:

Create a cron-based script that:

- Checks a web application every 5 mins.
- If it returns non-200 HTTP code, restart the app container.

Solution:

```bash
#!/bin/bash
URL="http://localhost:8080"
STATUS=$(curl -s -o /dev/null -w "%{http_code}" $URL)
if [ "$STATUS" != "200" ]; then
  echo "$(date): App unhealthy, restarting container..." >> /var/log/webapp_health.log
  docker restart webapp
fi
```

Follow-up:
➡️ Add exponential backoff retries.
➡️ Record uptime metrics in Prometheus-friendly format.

⚙️ Bonus: How These Challenges Map to Interview Categories

Category	What Interviewers Evaluate	Example Challenges
Monitoring / Health Checks	Automation logic, parsing	1, 2, 3, 10
Log Processing	Text manipulation (**`awk`**, **`grep`**)	1, 5, 9
File / Backup Automation	Scheduling, safety	6
Networking	Troubleshooting	4, 10
Performance & System Insights	OS familiarity	8
DevOps Integration	Tooling (Ansible, Slack, Docker)	7, 10

Would you like me to:

- 🧾 **Add 10 more expert-level challenges** (e.g., parsing JSON APIs, building CLI tools, multi-threaded loops, or integrating with Prometheus / AWS)?
- Or 🎯 **compile all 100 Q&As + these 10 hands-on tasks into one "Bash Scripting Master Handbook" PDF** for pre-interview revision?

give me pdf format for this conversation

Your PDF is ready! You can download it using the link below:

📄 Download Bash_Scripting_Interview_Questions_HandsOn_Challenges.pdf