DevOps — Shell Script (25 Questions)

Q1: Your bash deployment script sometimes fails silently when a command in the middle errors out.

Answer:
Use set -euo pipefail at the start to make the script exit on error, treat unset variables as errors, and fail pipelines if any command fails.
Sample Points:

- `-e` stops on first error.
- `-u` catches undefined vars.
- `-o pipefail` ensures pipeline errors propagate.

Example Code:

```
#!/bin/bash
set -euo pipefail
```

Q2: You need to trap cleanup commands if the script is interrupted (Ctrl+C).

Answer:

Use trap to catch `SIGINT` and `SIGTERM` and execute cleanup functions.
Sample Points:

- Prevents leftover temp files.
- Improves script resilience.
- Can handle multiple signals.

Example Code:

```
trap 'rm -f /tmp/mytmp; exit' INT TERM
```

Q3: Script must validate a JSON file’s syntax before processing it.

Answer:

Use `jq empty`or `python -m json.tool` for validation.

Sample Points:

- Prevents downstream parsing errors.
- jq gives clear error messages.
- Use in CI pre-check.

Example Code:

```
jq empty config.json
```

Q4: You want to ensure only one instance of the script runs at a time.

Answer:

Use a lock file with `flock` to prevent concurrent execution.

Sample Points:

- Avoids race conditions.
- flock auto-releases on exit.
- Works across processes.

Example Code:

```
exec 200>/var/lock/myscript.lock
flock -n 200 || { echo "Script already running"; exit 1;}
```

Q5: Need to safely handle file names with spaces in a loop.

Answer:

Use IFS and `read -r` with `find -print0 | xargs -0`.

Sample Points:

- Avoids word-splitting issues.
- Handles special chars in names.
- Prevents accidental file skipping.

Example Code:

```
find . -type f -print0 | while IFS= read -r -d '' file; do
  echo "$file"
done
```

Q6: Script must process a large log file efficiently without loading it fully into memory.

Answer:
Use `while read` loops or `awk` streaming.
Sample Points:

- Line-by-line avoids memory issues.
- Streaming is faster for large files.
- Use `grep` before processing to filter.

Example Code:

```
grep "ERROR" /var/log/app.log | while read -r line; do
  echo "$line"
done
```

Q7: A script should fail if a required environment variable is missing.

Answer:
Check with parameter expansion.

Sample Points:

- `:-` sets default; `:?` throws error.

- Ensures variables are set before use.
- Avoids runtime surprises.
 Example Code:

```
: "${DB_HOST:?Need to set DB_HOST}"
```

Q8: Need to create a temp file that's auto-deleted on script exit.

Answer:
Use `mktemp`and `trap`.
Sample Points:

- mktemp creates unique files.
- trap ensures cleanup.
- Avoids collision.
 Example Code:

```
tmpfile=$(mktemp)
trap "rm -f $tmpfile" EXIT
```

Q9: You want to check if another process is running before starting a new one.

Answer:
Use `pgrep`and conditional logic.
Sample Points:

- Avoids duplicate daemons.
- Can match exact process name.
- Use exit codes for flow.
 Example Code:

```
if pgrep -x "nginx" >/dev/null; then
  echo "nginx running"
fi
```

Q10: Need to measure execution time of a script section.

Answer:

Use `SECONDS`variable or `date +%s`.

Sample Points:

- Lightweight timing.
- Good for profiling scripts.
- Can log to monitoring system.

Example Code:

```
start=$SECONDS
# do work
echo "Elapsed: $((SECONDS - start))s"
```

Q11: Need to run commands in parallel to speed up processing.

Answer:

Use `xargs -P`or GNU parallel.

Sample Points:

- Improves performance for many items.
- Control parallelism with `-P`.
- Beware shared resource conflicts.

Example Code:

```
cat list.txt | xargs -n1 -P4 ./worker.sh
```

Q12: Script must verify network connectivity before proceeding.

Answer:

Use `nc` (netcat) or `curl` in a loop with retries.

Sample Points:

- Retry logic avoids transient fails.
- Check port availability.
- Timeout to avoid hanging.

Example Code:

```
for i in {1..5}; do
  nc -z db.example.com 5432 && break
  sleep 5
done
```

Q13: Need to handle different behavior depending on OS type.

Answer:

Check `uname` or `/etc/os-release`.

Sample Points:

- Portable OS detection.
- Switch-case for logic.
- Useful for cross-platform scripts.

Example Code:

```
os=$(uname)
case "$os" in
  Linux) echo "Linux detected";;
 Darwin) echo "macOS detected";;
esac
```

Q14: A command's stderr should be logged separately from stdout.

Answer:

Redirect with `2>` and `1>`.

Sample Points:

- Keeps logs organized.
- Useful in debugging pipelines.
- Combine if needed with `&>`.

Example Code:

```
cmd >out.log 2>err.log
```

Q15: Need to check exit code of the last command and act accordingly.

Answer:

Check `$?` immediately after the command.

Sample Points:

- Must check before running another command.
- Non-zero means failure.
- Use in conditionals.

Example Code:

```
if ! cp file1 file2; then
  echo "Copy failed"
fi
```

Q16: Script must prompt user for confirmation before destructive action.

Answer:

Use `read -p`and check response.

Sample Points:

- Protects against accidental deletes.
- Default to “no” on invalid input.
- Timeout for automation.

Example Code:

```
read -p "Delete all files? (y/N): " ans
[[ $ans == "y" ]] || exit 1
```

Q17: Need to extract specific column from CSV without a full parser.

Answer:
Use `cut -d, -fN` or `awk -F, '{print $N}'`.
Sample Points:

- Lightweight column extraction.
- Works for simple CSVs.
- Beware quoted fields with commas.

Example Code:

```
cut -d, -f2 data.csv
```

Q18: Script must ensure required binaries are installed before running.

Answer:
Check with `command -v`.
Sample Points:

- Avoids runtime missing command errors.
- Provide install hints.
- Exit gracefully if missing.

Example Code:

```
command -v jq >/dev/null || { echo "jq missing"; exit 1;}
```

Q19: Need to compress logs older than 7 days automatically.

Answer:

Use find with `-mtime` and `gzip`.

Sample Points:

- Automates log rotation.
- Reduces disk usage.
- Schedule via cron/systemd timer.

Example Code:

```
find /var/log -type f -mtime +7 -exec gzip {} \;
```

Q20: Script should run a background job and continue processing.

Answer:

Append & and optionally disown.

Sample Points:

- Avoids blocking script flow.
- Use logs to monitor background job.
- Track PID for control.

Example Code:

```
./long_task.sh &
```

Q21: Need to match only exact string in `grep` search.

Answer:

Use `grep -x` or `grep -w` for word match.

Sample Points:

- Avoids partial matches.
- Anchors pattern to line boundaries.
- Improves accuracy.

Example Code:

```
grep -x "ERROR" logfile
```

Q22: Script must create a tarball excluding certain files.

Answer:

Use `tar--exclude`.

Sample Points:

- Useful for packaging.
- Multiple `--exclude` allowed.
- Patterns support wildcards.

Example Code:

```
tar czf app.tar.gz --exclude='*.log' app/
```

Q23: Need to parse a command's output in a loop without losing spaces.

Answer:

Use `whileIFS= read -r line`.

Sample Points:

- Preserves whitespace.

- Works with pipelines.
- Avoids word splitting.

Example Code:

```
df -h | while IFS= read -r line; do
  echo "$line"
done
```

Q24: Script must generate a timestamp for filenames.

Answer:
Use `date` with a safe format.

Sample Points:

- Avoid spaces/colons in filenames.
- Include timezone if needed.
- Works in backups/logs.

Example Code:

```
ts=$(date +%Y%m%d_%H%M%S)
```

Q25: You want to debug each command before it executes in the script.

Answer:

Use `set -x`for execution tracing.

Sample Points:

- Prints each command to stderr.
- Useful for debugging complex flows.
- Turn off with `set +x`.

Example Code:

```
set -x
```