Bash Scripting Reference (DevOps – Industry Standard)

This document is a **quick-reference guide** while writing Bash scripts for DevOps. It focuses on **syntax, patterns, and best practices** used in real production systems and interviews.

---

1. Script Header (Mandatory)

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

Why: - `#!/bin/bash` → ensure bash (not sh) - `-e` → exit on error - `-u` → error on unset variables - `pipefail` → fail pipelines correctly

---

2. Variable Declaration

Global variables

```
SERVICE_NAME="nginx"
LOG_FILE="/var/log/app.log"
```

Local variables (inside functions)

```
local MESSAGE="$1"
```

Rules: - Always quote variables: `"$VAR"` - Use `local` inside functions

---

3. Command Substitution

```
NOW=$(date '+%Y-%m-%d %H:%M:%S')
```

✅Use `$( )` ❌Do not use backticks

---

4. Conditionals – WHEN TO USE WHAT (IMPORTANT)

4.1 Command check

```
if systemctl is-active --quiet nginx; then
  echo "running"
fi
```

4.2 String comparison

```
if [[ "$ENV" == "prod" ]]; then
  deploy_prod
fi
```

4.3 Numeric comparison

```
if (( COUNT > 5 )); then
  echo "threshold reached"
fi
```

4.4 POSIX shell only (avoid in bash)

```
if [ "$COUNT" -gt 5 ]; then
  echo "threshold reached"
fi
```

---

5. Loops

For loop (array / list)

```
for ITEM in "${ITEMS[@]}"; do
  echo "$ITEM"
done
```

While loop

```
while true; do
  sleep 5
done
```

---

6. Arrays (Very Important)

```
SERVICES=("nginx" "docker" "ssh")
```

Looping over array (SAFE)

```
for SVC in "${SERVICES[@]}"; do
  systemctl restart "$SVC"
done
```

Printing all values

```
echo "${SERVICES[*]}"
```

Rule: - Loop → `@` - Join → `*`

---

7. Functions

```
check_service() {
  local SERVICE="$1"
  systemctl is-active --quiet "$SERVICE"
}
```

---

8. Script Arguments

```
ENV="$1"
```

```
./deploy.sh prod
```

Special variables: - `$1` → first argument - `$@` → all arguments (safe) - `$#` → argument count

---

9. Exit Codes & Return

Function failure

```
return 1
```

Script failure

```
exit 1
```

Meaning: - `0` → success - non-zero → failure

---

10. Logging (Industry Standard)

```
log() {
  echo "$(date '+%F %T') : $1" >> "$LOG_FILE"
}
```

---

11. Cron Jobs

Edit cron

```
crontab -e
```

Every 5 minutes

```
*/5 * * * * /path/script.sh >> /var/log/script.log 2>&1
```

At reboot

```
@reboot /path/script.sh
```

---

12. Redirection (Must Know)

```
>> file.log      # append stdout
2>&1             # redirect stderr to stdout
```

---

13. Absolute Path Rule (CRON & PROD)

❌ `./script.sh`

✅ `/home/user/scripts/script.sh`

---

14. Golden DevOps Bash Rules

- Prefer `[[ ]]` and `(( ))`
- Avoid `[ ]` in bash scripts
- Always quote variables
- Always log actions
- Always return proper exit codes
- Never parse command output when exit code exists

---

15. Interview One-Liner

> "I choose condition syntax based on what I'm testing: commands directly, strings with `[[ ]]`, and numbers with `(( ))`. I follow strict mode, logging, and proper exit codes."

End of Phase-1 Bash Reference