Advanced Linux Scripting & Automation — Edition

A practical guide with production-ready scripts, automation patterns, and examples for sysadmins and DevOps engineers.

1. Bash: Strict mode & best practices

Start scripts with strict mode to catch errors early and make scripts safer and more predictable.

```
#!/usr/bin/env bash
set -euo pipefail
IFS=$'\n\t'

# Description: Strict-mode template for robust bash scripts
main() {
  local logfile="/var/log/myscript.log"
  echo "$(date -u +%Y-%m-%dT%H:%M:%SZ) - Starting script" >> "$logfile"
  # ... script logic here ...
}

main "$@"
```

2. Modular functions, logging, and exit handling

Use functions, consistent logging helpers, and traps for clean shutdown and debugging.

```
log() { echo "$(date -u +%Y-%m-%dT%H:%M:%SZ) [${1:-INFO}] ${2:-}" >&2; }
error_exit() { log ERROR "$1"; exit "${2:-1}"; }

cleanup() {
  # perform cleanup actions
  log INFO "Cleaning up"
}

trap cleanup EXIT INT TERM

do_backup() {
  local src="$1" dst="$2"
  rsync -a --delete "$src" "$dst" || error_exit "rsync failed"
  log INFO "Backup finished: $src -> $dst"
}

main() {
  do_backup /var/www /backup/www
}

main "$@"
```

3. Idempotent server setup (users, packages, SSH hardening)

Example: idempotent script that can be run multiple times safely.

```
#!/usr/bin/env bash
set -euo pipefail
IFS=$'\n\t'

ensure_user() {
  local user="$1"
  if id -u "$user" >/dev/null 2>&1; then
    echo "User $user exists"
  else
    sudo adduser --disabled-password --gecos "" "$user"
    sudo usermod -aG sudo "$user"
    echo "Created user $user and added to sudoers"
  fi
}

ensure_package() {
  local pkg="$1"
  if dpkg -s "$pkg" >/dev/null 2>&1; then
    echo "Package $pkg is already installed"
  else
    sudo apt-get update -y
    sudo apt-get install -y "$pkg"
  fi
}

main() {
  ensure_user deploy
  ensure_package git
  ensure_package nginx
}

main "$@"
```

4. Robust backup script with retention & rotation

Uses rsync for efficient transfers and a simple rotation strategy.

```
#!/usr/bin/env bash
set -euo pipefail
IFS=$'\n\t'

SRC="/var/www/"
DST="/backups/www"
KEEP=7  # keep 7 daily backups

mkdir -p "$DST"
timestamp=$(date +%Y%m%dT%H%M%S)
tmpdir="$DST/tmp-$timestamp"
mkdir -p "$tmpdir"

# create new snapshot
rsync -a --delete --link-dest="$DST/latest" "$SRC" "$tmpdir/"
mv "$tmpdir" "$DST/$timestamp"
```

```
ln -snf "$DST/$timestamp" "$DST/latest"

# rotate old backups
cd "$DST"
ls -1dt */ | tail -n +$((KEEP+1)) | xargs -r rm -rf
```

5. systemd service & timer (run script via systemd)

Create a systemd unit and a timer to run a script daily.

```
# /etc/systemd/system/daily-backup.service
[Unit]
Description=Daily backup service
Wants=daily-backup.timer

[Service]
Type=oneshot
ExecStart=/usr/local/bin/daily-backup.sh
User=root
Group=root

# /etc/systemd/system/daily-backup.timer
[Unit]
Description=Run daily-backup daily

[Timer]
OnCalendar=daily
Persistent=true

[Install]
WantedBy=timers.target
```

Enable and start timer:

```
sudo systemctl daemon-reload
sudo systemctl enable --now daily-backup.timer
systemctl list-timers --all | grep daily-backup
```

6. Ansible: simple playbook to deploy an Nginx site

A minimal, idempotent playbook to install and configure Nginx with a site.

```
---
- name: Deploy static website
  hosts: webservers
  become: yes
  vars:
    site_root: /var/www/myapp
  tasks:
    - name: Ensure apt cache is up-to-date
      apt:
        update_cache: true

    - name: Install nginx
      apt:
        name: nginx
        state: present

    - name: Create site directory
      file:
        path: "{{ site_root }}"
        state: directory
        owner: www-data
        group: www-data
        mode: '0755'

    - name: Copy site files
      copy:
        src: ./site/
        dest: "{{ site_root }}/"
        owner: www-data
        group: www-data
        mode: '0644'

    - name: Deploy nginx site
      template:
        src: templates/myapp.conf.j2
        dest: /etc/nginx/sites-available/myapp
      notify: Reload nginx

  handlers:
    - name: Reload nginx
      service:
        name: nginx
        state: reloaded
```

7. Docker: build, tag, push automation script

Script to build an image, run tests, tag and push to a registry.

```
#!/usr/bin/env bash
set -euo pipefail
IMAGE_NAME="registry.example.com/myapp"
VERSION="$(git rev-parse --short HEAD)"

docker build -t "${IMAGE_NAME}:${VERSION}" .
docker run --rm "${IMAGE_NAME}:${VERSION}" /bin/sh -c "pytest -q || exit 1"
docker tag "${IMAGE_NAME}:${VERSION}" "${IMAGE_NAME}:latest"
docker push "${IMAGE_NAME}:${VERSION}"
docker push "${IMAGE_NAME}:latest"
```

Example Dockerfile (multi-stage):

```
# Dockerfile (multi-stage)
FROM python:3.11-slim as builder
WORKDIR /app
COPY pyproject.toml poetry.lock /app/
RUN pip install --upgrade pip && pip install poetry && poetry config virtualenvs.create false && poetry i

FROM python:3.11-slim
WORKDIR /app
COPY --from=builder /usr/local/lib/python3.11/site-packages /usr/local/lib/python3.11/site-packages
COPY . /app
CMD ["gunicorn", "myapp.wsgi:application", "--bind", "0.0.0.0:8000"]
```

8. CI/CD: GitHub Actions workflow (build, test, deploy)

```
name: CI

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.11'
      - name: Install deps
        run: |
          python -m pip install --upgrade pip
          pip install -r requirements.txt
      - name: Run tests
        run: pytest -q

  build-push:
    needs: build-test
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'
    steps:
      - uses: actions/checkout@v4
      - name: Build and push image
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: registry.example.com/myapp:latest
```

9. Logrotate config example and monitoring script

Logrotate configuration for application logs and a simple log-monitoring alert script.

```
# /etc/logrotate.d/myapp
/var/log/myapp/*.log {
  daily
  rotate 14
  compress
  delaycompress
  missingok
  notifempty
  create 0640 myapp myapp
  sharedscripts
  postrotate
    systemctl restart myapp.service >/dev/null 2>&1 || true
  endscript
}
#!/usr/bin/env bash
# monitor-errors.sh - send simple alert if error rate spikes (example)
LOG=/var/log/myapp/access.log
THRESHOLD=100 # errors in last hour

errors=$(grep "$(date --date='1 hour ago' +%d/%b/%Y:%H)" "$LOG" | grep -i error | wc -l || true)
if [ "$errors" -gt "$THRESHOLD" ]; then
  echo "High error rate: $errors in the last hour" | mail -s "Alert: MyApp errors" ops@example.com
fi
```

10. Networking automation examples

```
# Apply netplan config (idempotent)
sudo tee /etc/netplan/01-netcfg.yaml > /dev/null <<'EOF'
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: no
      addresses: [192.168.10.20/24]
      gateway4: 192.168.10.1
      nameservers:
        addresses: [8.8.8.8,8.8.4.4]
EOF
sudo netplan apply
# nmcli example - create connection
nmcli connection add type ethernet ifname eth0 con-name static-eth0 ipv4.addresses 192.168.10.20/24 ipv4.
nmcli connection up static-eth0
```

11. sed & awk: powerful text processing

```
# Extract 2nd column from CSV, ignoring header
awk -F, 'NR>1 {print $2}' data.csv

# Sum values in 3rd column
awk -F, '{sum += $3} END {print sum}' data.csv

# Replace tabs with commas
sed -i 's/\t/,/g' file.tsv
```

12. Python automation: Using subprocess, pathlib, and logging

```
#!/usr/bin/env python3
import subprocess, pathlib, logging, sys
logging.basicConfig(level=logging.INFO)

def run(cmd):
    logging.info('Running: %s', cmd)
    subprocess.run(cmd, shell=True, check=True)

def backup(src, dst):
    src = pathlib.Path(src)
    dst = pathlib.Path(dst)
    dst.mkdir(parents=True, exist_ok=True)
    run(f"rsync -a --delete {src}/ {dst}/")

if __name__ == '__main__':
    backup('/var/www', '/backups/www')
```

13. Security automation: fail2ban and unattended-upgrades

```
# Simple fail2ban jail local
[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 5
bantime = 3600
# unattended-upgrades config example (/etc/apt/apt.conf.d/50unattended-upgrades)
Unattended-Upgrade::Allowed-Origins {
  "\\"${distro_id}:${distro_codename}-security";
};
Unattended-Upgrade::Automatic-Reboot "true";
```

14. Database: mysqldump backup with compression

```
#!/usr/bin/env bash
set -euo pipefail
DB_USER="backup"
DB_PASS="secret"
DB_NAME="myappdb"
OUTDIR="/backups/db"
mkdir -p "$OUTDIR"
timestamp=$(date +%Y%m%dT%H%M%S)
mysqldump -u"$DB_USER" -p"$DB_PASS" "$DB_NAME" | gzip > "$OUTDIR/${DB_NAME}_$timestamp.sql.gz"
# rotate last 30 days
find "$OUTDIR" -type f -mtime +30 -delete
```

15. Zero-downtime deploy: atomic symlink swap pattern

```
#!/usr/bin/env bash
set -euo pipefail
RELEASES=/var/www/releases
CURRENT=/var/www/current
NEW_RELEASE="$RELEASES/$(date +%s)"

mkdir -p "$NEW_RELEASE"
# build into $NEW_RELEASE, e.g. extract tar or git archive
tar -xzf /tmp/myapp.tar.gz -C "$NEW_RELEASE"
# run migrations
cd "$NEW_RELEASE" && ./manage.py migrate --noinput
# update symlink atomically
ln -sfn "$NEW_RELEASE" "$CURRENT"
# restart service
systemctl restart myapp.service
```

Further reading & suggestions

This edition focused on practical, production-ready templates. You can adapt these scripts to your environment — remember to test in staging, add robust error handling, secrets management (use vaults/secret managers), and observability.