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
```

DevOps — Ansible (25 Questions)

Q1: You run an Ansible playbook to provision EC2 instances, but some tasks hang indefinitely on SSH.

Answer:
This can happen if the new EC2 instances aren't ready for SSH yet or have restrictive SG rules. Add a `wait_for` task to ensure port 22 is open before gathering facts, and verify `ansible_user` matches the AMI's default.

Sample Points:

- Use `wait_for` before `gather_facts`.
- Ensure SG allows SSH from control node.

- Use correct remote user for AMI type.

Example Code:

```
- name: Wait for SSH
  wait_for:
    host: "{{ inventory_hostname }}"
    port: 22 delay: 5 timeout: 300
```

Q2: Your Ansible run modifies files every time even though the contents haven't changed.

Answer:

The task is not idempotent — for file edits, use `lineinfile`/ `blockinfile`with proper regex, and for templates, use template which only updates if checksums differ.

Sample Points:

- Idempotency avoids unnecessary changes.
- Use correct module for text changes.
- Compare generated vs. target file.

Example Code:

```
- name: Ensure line present
  lineinfile:
    path: /etc/sysctl.conf
    regexp: '^net.ipv4.ip_forward'
    line: 'net.ipv4.ip_forward=1'
```

Q3: A task needs to run only when a file exists on the target host.

Answer:

Use stat module to check and conditionally run the next task with `when`.

Sample Points:

- stat returns exists flag.
- Avoid failing on missing files.
- Clean conditional syntax.

Example Code:

```
 - stat: path=/etc/my.conf

  register: conf_file

- name: Do something
command: cat /etc/my.conf
 when: conf_file.stat.exists
```

Q4: Playbook fails on some hosts due to Python not being installed.

Answer:

Use raw module to install Python first, as Ansible modules need Python.

Sample Points:

- Cloud images (minimal) may lack Python.
- raw bypasses Python requirement.
- Install python3 early in play.

Example Code:

```
- name: Install Python
  raw: sudo apt-get update && sudo apt-get install -y python3
```

Q5: You want to encrypt sensitive variables in your repo.

Answer:

Use `ansible-vault encrypt` for the vars file, and store the vault password outside VCS.

Sample Points:

- Never commit plain secrets.
- Vault key in CI/CD via secret store.
- Can encrypt single vars or whole files.
  Example Code:

```
ansible-vault encrypt group_vars/prod/secrets.yml
```

Q6: You need to run a specific set of tasks only on RHEL-based hosts.

Answer:
Use `when:ansible_facts['os_family'] == 'RedHat'`.
Sample Points:

- Use gathered facts for OS-specific logic.
- Avoid hardcoding hostnames.
- Keep tasks portable with conditionals.
  Example Code:

```
when: ansible_facts['os_family'] == "RedHat"
```

Q7: Playbook must be rerun without re-executing heavy install tasks.

Answer:
Use `creates`param in `command` or register a state flag file.
Sample Points:

- Skips task if output already exists.
- Avoids redundant installs.
- Faster reruns.
  Example Code:

```
- name: Install app
  command: /opt/install.sh creates=/opt/app_installed.flag
```

Q8: You want to dynamically pull inventory from AWS EC2.

Answer:

Use the `aws_ec2` dynamic inventory plugin and configure AWS credentials in env/credentials file.

SamplePoints:

- No need for static host files.
- Tag filtering for host groups.
- Refresh inventory automatically.

Example Code:

```
plugin: aws_ec2
regions:
  - ap-south-1
keyed_groups:
  - key: tags.Name
```

Q9: Need to reuse task logic across multiple playbooks.

Answer:

Use roles to package tasks, vars, handlers, and templates together.

Sample Points:

- Roles promote reusability.
- Avoids duplicating tasks.
- Makes plays cleaner.

Example Code:

```
- hosts: web
```

```
  roles:
    -nginx_setup
```

Q10: Playbook must read a variable from another host in the same play.

Answer:

Use `hostvars` to access facts/vars from other hosts.

Sample Points:

- hostvars is a dictionary of all hosts.
- Requires both hosts in same play context.
- Useful for leader-worker configs.

Example Code:

```
db_host: "{{ hostvars['db1'].ansible_host }}"
```

Q11: Task needs to run only if a service is not already running.

Answer:

Check with `service_facts` and conditionally start.

Sample Points:

- Avoids restarting running services.
- service_facts gathers all services.
- Improves idempotency.

Example Code:

```
- service_facts:
- service:
    name: nginx
    state: started
  when: "'nginx' not in services or services['nginx'].state !=
'running'"
```

Q12: Playbook execution must stop if a critical task fails.

Answer:

Use `any_errors_fatal: true` at play level.

Sample Points:

- Ensures all hosts stop on failure.
- Useful for critical infra changes.
- Avoid partial config states.

Example Code:

```
- hosts: all
  any_errors_fatal: true
```

Q13: Need to set variables that are evaluated only at execution time.

Answer:

Use `set_fact` with Jinja templates.

Sample Points:

- set_fact is dynamic at runtime.
- Useful for computed values.
- Facts persist for rest of play.

Example Code:

```
- set_fact:
    backup_path: "/backups/{{ inventory_hostname }}/{{
ansible_date_time.date }}"
```

Q14: Playbook fails on missing variable in a template.

Answer:
Set `jinja2_native=True` and `default` filter in template to avoid undefined errors.
Sample Points:

- default avoids undefined var crash.
- Set safe defaults.
- Reduces fragile templates.

Example Code:

```
{{ some_var | default('N/A') }}
```

Q15: Need to run some tasks as another Linux user without switching SSH login.

Answer:
Use `become` with `become_user`.
Sample Points:

- become allows privilege escalation.
- Works for sudo or su target.
- Avoids separate SSH creds.

Example Code:

```
- name: Run as postgres
  command: psql -c "SELECT 1"
  become: true
  become_user: postgres
```

Q16: Deploying app needs different config files per environment.

Answer:
Use group_vars for environment-specific vars.

Sample Points:

- Separate vars by inventory group.
- Avoids conditionals inside playbooks.
- Cleaner separation of configs.
  Example Code:

```
group_vars/
prod.yml
dev.yml
```

Q17: Playbook should stop at a certain task for debugging.

Answer:
Use `meta: end_play` or `pause` for interactive debug.
Sample Points:

- end_play stops entire play.
- pause allows manual checks.
- Useful in staging/testing.
  Example Code:

```
- meta: end_play
```

Q18: Need to install packages on both Debian and RHEL hosts with one task.

Answer:
Use `package` module with variables mapping per OS.
Sample Points:

- package is generic across distros.
- Avoids duplicate tasks per OS.

- Use var mapping for names.

Example Code:

```
vars:
  pkg_name:
    RedHat: httpd
    Debian: apache2
- package:
    name: "{{ pkg_name[ansible_os_family] }}"
    state: present
```

Q19: A handler is not running even though task notifies it.

Answer:

Handlers run at the end of plays unless `meta: flush_handlers` is used.

Sample Points:

- flush_handlers triggers early.
- Handlers run once per play.
- Useful for service restarts mid-play.

Example Code:

```
- meta: flush_handlers
```

Q20: Need to securely fetch secrets from AWS SSM in playbook.

Answer:

Use `aws_ssm` lookup plugin with IAM role permissions.

Sample Points:

- Avoids hardcoded creds.
- IAM role least-privilege.

- Pull secrets at runtime.
  Example Code:

```
db_pass: "{{ lookup('aws_ssm', '/prod/db_pass', region='us-east-1')
}}"
```

Q21: Limit playbook run to a subset of tasks for quick testing.

Answer:
Use `--tags` and `--skip-tags` with well-tagged tasks.
Sample Points:

- Tags speed up testing.
- Skip irrelevant tasks.
- Tag logically by function.
  Example Code:

```
ansible-playbook site.yml --tags "nginx,deploy"
```

Q22: Playbook fails due to SSH host key verification prompt.

Answer:
Set `host_key_checking = False` in ansible.cfg for automation (with caution).
Sample Points:

- Disables fingerprint prompt.
- Security trade-off: trust host blindly.
- Prefer adding host key to known_hosts.
  Example Code:

```
[defaults]
host_key_checking = False
```

Q23: Need to run a task only on first host in a group.

Answer:

Use `when: inventory_hostname == groups['web'][0]`.

Sample Points:

- Controls execution scope.
- Avoids duplicate operations.
- Index 0 is first host.

Example Code:

```
when: inventory_hostname == groups['web'][0]
```

Q24: A variable value needs to be computed using output from a previous command.

Answer:

Register command output and use `set_fact`.

Sample Points:

- Register stores task output.
- Use stdout_lines for list processing.
- set_fact makes it available globally.

Example Code:

```
- command: hostname
  register: host_out
- set_fact:
    fqdn: "{{ host_out.stdout }}.example.com"
```

Q25: Need to ensure a directory exists with specific permissions.
Answer:

Use file module with `state: directory`.

Sample Points:

- Ensures idempotent creation.
- Sets ownership and mode.
- Works for nested paths.

Example Code:

```
- file:
    path: /opt/data
    state: directory
    owner: appuser
    mode: '0750'
```

DevOps — Git (25 Questions)

Q1: A feature branch shows hundreds of unrelated changes in `git diff` after merging main.

Answer:
This can happen if the branch was created from an outdated base or if a merge was done with incorrect ancestry. Run git fetch --all, then rebase onto the latest main to replay only your commits. For large drift, create a new branch from main and cherry-pick your changes.

Sample Points:

- Outdated base = merge noise.

- Rebase to clean commit history.
- Cherry-pick for precise isolation.

Example Code:

```
git checkout feature
git fetch origin
git rebase origin/main
```

Q2: You accidentally committed secrets to Git and pushed to remote.

Answer:
Use `git filter-repo` or `BFG Repo-Cleaner` to remove them from history, force-push to remote, and rotate credentials immediately.

Sample Points:

- History rewrite required.
- Rotate secrets even if removed.
- Inform collaborators to re-clone.

Example Code:

```
git filter-repo --path secret.txt --invert-paths
git push origin --force
```

Q3: Merge conflicts keep recurring in the same file across multiple sprints.

Answer:
This is usually due to parallel changes in high-churn files. Apply a consistent merge strategy (ours, theirs), or split config into smaller files. Encourage feature toggles to reduce long-lived branches.

Sample Points:

- High-churn files → constant conflicts.

- Modularize files to isolate changes.
- Short-lived branches minimize pain.

Example Code:

```
git merge -X theirs feature
```

Q4: `git pull` shows “unrelated histories” after repo migration.

Answer:
This happens when histories differ. Use `git pull --allow-unrelated-histories` during the first merge, then align branches.

Sample Points:

- Repo migration resets ancestry.
- Merge histories once, then clean.
- Use tags to mark migration point.

Example Code:

```
git pull origin main --allow-unrelated-histories
```

Q5: You need to squash all commits in a feature branch before merging.

Answer:
Use `git rebase -i` to squash commits, or `git merge --squash` when merging into target.

Sample Points:

- Squash for cleaner history.
- Rebase interactive for fine control.
- Squash merge preserves branch diff only.

Example Code:

```
git rebase -i HEAD~5
```

Q6: You cloned a large repo but only need a single folder.

Answer:

Use sparse checkout to pull only that folder.

Sample Points:

- Sparse checkout saves bandwidth.
- Useful for monorepos.
- Avoid full clone if unnecessary.

Example Code:

```
git sparse-checkout init --cone
git sparse-checkout set folder/path
```

Q7: A teammate force-pushed `main` and you have diverging histories.

Answer:

Fetch latest, back up your branch, then hard reset to remote main. Reapply your changes on top.

Sample Points:

- Back up before reset.
- Divergence due to force push.
- Use reflog if needed.

Example Code:

```
git fetch origin
git checkout main
git reset --hard origin/main
```

Q8: Need to sign all commits with GPG for compliance.

Answer:

Generate GPG key, add to Git config, and enforce signed commits in repo settings.

Sample Points:

- GPG key for identity assurance.
- Repo policy enforces signing.
- Protects against commit spoofing.

Example Code:

```
git config --global user.signingkey <KEY_ID>
git commit -S -m "Signed commit"
```

Q9: Large binary files cause repo size to explode.

Answer:

Use Git LFS for binaries, migrate existing ones with `git lfs migrate`.

Sample Points:

- LFS stores binaries outside Git history.
- Keeps repo lightweight.
- Requires LFS installed on clients.

Example Code:

```
git lfs install
git lfs track "*.zip"
```

Q10: Need to tag a release and push it to remote.

Answer:

Create annotated tags for release metadata. Push explicitly to remote.

Sample Points:

- Annotated tags store message/author.
- Push tags explicitly.
- Use semantic versioning.
  Example Code:

```
git tag -a v1.2.0 -m "Release 1.2.0"
git push origin v1.2.0
```

Q11: You need to revert a single commit in the middle of history without removing others.

Answer:
Use `git revert` to create a new commit undoing changes.
Sample Points:

- Revert is safe on shared branches.
- Doesn't rewrite history.
- Reapply later if needed.
  Example Code:

```
git revert <commit_hash>
```

Q12: CI pipeline triggers on every commit to any branch, but you only want it on main and dev.

Answer:
Restrict pipeline triggers in `.gitlab-ci.yml` or GitHub Actions workflows with branch filters.
Sample Points:

- Branch filters reduce wasted CI runs.
- Config in pipeline definition.

- Improves build efficiency.
  Example Code:

```
on:
  push:
    branches:
      - main
      - dev
```

Q13: Accidentally committed large log files you don’t want in repo at all.

Answer:
Remove from index, add to `.gitignore`, and commit removal. For complete removal, rewrite history.
Sample Points:

- Remove and ignore to prevent recurrence.
- History rewrite for permanent removal.
- Consider LFS for large assets.
  Example Code:

```
git rm --cached big.log
echo "big.log" >> .gitignore
```

Q14: Need to apply a fix from `main` to an older release branch.

Answer:
Use `git cherry-pick` for the specific commit.
Sample Points:

- Cherry-pick applies single commit.
- Resolve conflicts manually if needed.

- Preserve commit metadata.
  Example Code:

```
git checkout release-1.0
git cherry-pick <commit_hash>
```

Q15: Git merge shows binary files as conflicts with no merge tool option.

Answer:
Binary files can't be merged automatically. Decide which version to keep using `git checkout --ours` or `--theirs`.

Sample Points:

- Binary conflicts require manual choice.
- Ours/theirs for conflict resolution.
- Commit after resolving.
  Example Code:

```
git checkout --ours path/file.bin
git add path/file.bin
```

Q16: You need to apply the same commit to multiple repos.

Answer:
Create a patch with `git format-patch` and apply with `git am`.

Sample Points:

- Format-patch preserves metadata.
- Works across unrelated repos.
- Keep patch files in secure storage.
  Example Code:

```
git format-patch -1 <commit>
```

```
git am < patchfile
```

Q17: Developer needs read-only access to repo.

Answer:

Grant Read role in hosting platform or create a deploy key with read permissions only.

Sample Points:

- Use platform RBAC.
- Deploy keys for automation.
- Never give write if not required.

Example Code:

```
ssh-keygen -t rsa -b 4096 -C "read-only"
```

Q18: A submodule commit is missing in remote after cloning.

Answer:

Ensure you clone with `--recursive` or run `git submodule update --init --recursive`.

Sample Points:

- Submodules need explicit init.
- Recursive flag fetches nested submodules.
- Keep submodules synced.

Example Code:

```
git clone --recursive repo.git
```

Q19: Need to rename a branch both locally and remotely.

Answer:
Rename locally, push to remote, and delete old remote branch.

Sample Points:

- Push new branch, delete old one.
- Update tracking branch config.
- Inform collaborators.

Example Code:

```
git branch -m old new
git push origin new
git push origin --delete old
```

Q20: Git bisect required to find commit that introduced a bug.

Answer:
Mark known good and bad commits, test each step.

Sample Points:

- Bisect binary search speeds debugging.
- Requires reproducible test.
- Automate with git bisect run.

Example Code:

```
git bisect start
git bisect bad HEAD
git bisect good v1.1
```

Q21: CI build fails because `.gitmodules` points to wrong URL.

Answer:
Update `.gitmodules`, run `git submodule sync`, and commit changes.

Sample Points:

- Sync updates local config.
- Commit .gitmodules change.
- Ensure URL works for CI.

Example Code:

```
git submodule sync --recursive
```

Q22: Need to temporarily stash changes for a quick hotfix.

Answer:
Use `git stash` to save uncommitted work, apply hotfix, then `git stash pop`.
Sample Points:

- Stash to free working dir quickly.
- Pop restores + removes from stash.
- Multiple stashes possible.

Example Code:

```
git  stash  #
hotfix   work
git stash pop
```

Q23: Commit history is cluttered with merge commits from frequent syncs.

Answer:
Use `git pull --rebase` to replay commits on top of fetched branch.
Sample Points:

- Rebasing avoids merge commits.
- Cleaner history for review.

- Conflicts resolved as usual.
  Example Code:

```
git pull --rebase origin main
```

Q24: You need to change the author of the last commit.

Answer:
Use `gitcommit --amend --author` . Force-push if already pushed.
Sample Points:

- Amend for local fix.
- Force push to update remote.
- Inform collaborators.
  Example Code:

```
git commit --amend --author="New Name <email>"
```

Q25: Repo size is huge due to unused branches.

Answer:
Prune stale remote branches with `git fetch --prune`, delete local copies.
Sample Points:

- Prune removes tracking refs.
- Delete locals to free space.
- Review before deleting.
  Example Code:

```
git fetch --prune
git branch -d old-branch
```

DevOps — Linux (25 Questions)

Q1: A production server is unresponsive over SSH, but ICMP ping works.

Answer:
Check if SSH daemon is down or blocked by firewall. Use console access (cloud provider serial console) to log in and inspect systemctl status sshd. If sshd is running, check ss -tnlp | grep :22 and firewall rules (iptables -L or firewalld). Also inspect fail2ban for IP bans.

Sample Points:

- ICMP works → kernel/network up.
- SSH port/service may be blocked/stopped.
- Fail2ban or security group rules can block.

Example Code:

```
sudo systemctl status sshd
sudo ss -tnlp | grep :22
sudo iptables -L -n
```

Q2: Disk usage on `/var` hits 100% causing app crashes.

Answer:
Identify large files with du -sh /var/*|sort-h , check `/var/log` for oversized logs, rotate/compress logs via logrotate . Remove old journal logs (`journalctl --vacuum-time=7d`) .

Sample Points:

- Large logs are common culprit.
- Use du to locate offenders.

- Configure logrotate to prevent repeats.
  Example Code:

```
sudo du -sh /var/* | sort -h
sudo journalctl --vacuum-time=7d
```

Q3: CPU usage is maxed out by a runaway process; system is sluggish.

Answer:
Find offending process with top or `ps-eopid,ppid,cmd,%cpu --sort=-%cpu` . Renice process or kill it. Investigate logs to see why it misbehaved.
Sample Points:

- Identify heavy process.
- Adjust priority with `renice`.
- Kill only if safe.
  Example Code:

```
sudo renice 10 -p <PID>
```

Q4: Network latency spikes intermittently to an external service.

Answer:
Run mtr or `traceroute` to detect hop causing delay. Check `ethtool` for interface errors and `dmesg` for NIC resets.
Sample Points:

- MTR for real-time path analysis.
- Check NIC stats for errors.
- Could be external ISP hop.
  Example Code:

```
mtr --report google.com
```

```
sudo ethtool -S eth0
```

Q5: Cron job didn’t execute, but works manually.

Answer:
Check cron logs (`grep CRON /var/log/syslog`), ensure script path and environment variables are absolute. Cron uses minimal `PATH`, so commands must be full path.
Sample Points:

- Cron needs absolute paths.
- Minimal environment in cron.
- Check execution permissions.

Example Code:

```
* * * * * /usr/bin/python3 /opt/scripts/task.py
```

Q6: SSH brute-force attempts are filling logs.

Answer:
Enable fail2ban with sshd jail, set MaxAuthTries in sshd_config, and consider moving SSH to a non-standard port (security by obscurity + logging).
Sample Points:

- fail2ban blocks repeated offenders.
- Reduce max auth tries.
- Keep logs clean for real events.

Example Code:

```
sudo apt install fail2ban
sudo systemctl enable --now fail2ban
```

Q7: A service doesn’t start on boot but runs fine manually.

Answer:
Enable with `systemctl enable servicename`, ensure `[Install]` section in unit file has correct `WantedBy`. Check `journalctl -u servicename` for boot-time errors.
SamplePoints:

- Unit must have enable target.
- Boot-time deps can delay start.
- Journal logs reveal cause.
  Example Code:

```
sudo systemctl enable myservice
```

Q8: File permission changes revert after reboot.

Answer:
Likely due to immutable attribute (`lsattr`) or config management (e.g., Ansible, Puppet). Remove immutable bit (`chattr -i`) or update CM template.
Sample Points:

- Check immutable attribute.
- CM tools can reset perms.
- Edit source templates.
  Example Code:

```
lsattr /path/file
sudo chattr -i /path/file
```

Q9: Web app returns 502 after Nginx restart.

Answer:
Check if upstream app service is running (`systemctl status`), correct upstream socket/port in Nginx config, test with `curllocalhost:port`.
Sample Points:

- 502 = upstream unreachable.
- Confirm backend service status.
- Port/socket config must match.
  Example Code:

```
curl -v http://127.0.0.1:5000
```

Q10: SELinux blocks app from binding to port 8080.

Answer:
Use `semanageport-a-thttp_port_t-ptcp 8080` to allow, or set SELinux boolean if service-specific. Check audit logs for denials.

Sample Points:

- SELinux enforces port context.
- Add port mapping for new services.
- Audit logs guide rules.
  Example Code:

```
sudo semanage port -a -t http_port_t -p tcp 8080
```

Q11: Slow DNS resolution on server.

Answer:
Check /etc/resolv.conf order, ensure fastest DNS servers, and test with `dig +trace`. Disable reverse DNS lookups in SSH (UseDNS no).

Sample Points:

- Optimize resolver order.
- Test with dig to isolate slowness.

- Disable unnecessary reverse lookups.
  Example Code:

```
dig example.com
```

Q12: User accidentally deleted `/etc/passwd`.

Answer:
Restore from backup or copy from a similar system, adjust UIDs/GIDs. Boot into rescue mode if needed.
Sample Points:

- System won't authenticate without it.
- Restore quickly from backup.
- Verify UIDs match actual files.
  Example Code:

```
cp /mnt/backup/etc/passwd /etc/passwd
```

Q13: Root disk filling with `/var/lib/docker`.

Answer:
Prune unused images/volumes, move Docker data root via `/etc/docker/daemon.json` and symlink or mount to larger volume.
Sample Points:

- Docker cache can bloat quickly.
- Move to dedicated disk.
- Schedule prune jobs.
  Example Code:

```
sudo mkdir /mnt/docker
{"data-root": "/mnt/docker"}
```

Q14: Process hangs in `D` state (uninterruptible sleep).

Answer:
Caused by kernel waiting on I/O. Check disk health with `smartctl`, inspect dmesg for I/O errors. Usually hardware/storage problem.

Sample Points:

- Dstate = I/O wait.
- Investigate disk/NFS mounts.
- Might require hardware fix.

Example Code:

```
sudo smartctl -a /dev/sda
```

Q15: Swap usage grows constantly, even with free RAM.

Answer:
Check vm.swappiness value (`sysctl vm.swappiness`). Lower to 10-20 to prefer RAM. Investigate memory leaks with `smem`.

Sample Points:

- Adjust swappiness for usage pattern.
- Swap use with free RAM often config issue.
- Check per-process memory.

Example Code:

```
sudo sysctl -w vm.swappiness=10
```

Q16: High load average but low CPU usage.

Answer:
Likely I/O wait or many blocked processes. Check `iostat -xz`, look for `%util` and await.

Sample Points:

- Load includes I/O wait.
- Identify bottleneck device.
- Use iostat for detailed view.
  Example Code:

```
iostat -xz15
```

Q17: After adding new disk, it's not visible in `/dev`.

Answer:
Rescan SCSI bus (`echo"- - -" > /sys/class/scsi_host/hostX/scan`) or reboot. Partition with `fdisk` and mount.

Sample Points:

- Rescan before reboot.
- Partition and format.
- `Update /etc/fstab`.
  Example Code:

```
sudo fdisk /dev/sdb
```

Q18: System time drifts in VM.

Answer:
Enable NTP (`chronyd` or `systemd-timesyncd`), check hypervisor time sync.

Sample Points:

- NTP for accuracy.

- VM tools can sync time too.
- Avoid both sources conflicting.
  Example Code:

```
sudo timedatectl set-ntp true
```

Q19: User can’t write to shared directory despite group membership.

Answer:
Ensure directory has `g+w` and `setgid` bit so new files inherit group.

Sample Points:

- setgid preserves group ownership.
- Check umask settings.
- Group perms must be correct.
  Example Code:

```
chmod 2775 /shared
```

Q20: Network service fails to bind after reboot because interface name changed.

Answer:
Disable predictable interface names or update systemd service to use new name. Use networkctl to see current mapping.

Sample Points:

- Predictable names can change after NIC add/remove.
- Use persistent naming via udev rule.
- Update configs accordingly.
  Example Code:

```
sudo ln -s /dev/null /etc/udev/rules.d/80-net-setup-link.rules
```

Q21: Script fails in cron but works manually due to environment vars.

Answer:

Export needed vars in script or source profile file at start. Cron has minimal env.

Sample Points:

- Cron env minimal.
- Source ~/.profile if needed.
- Explicitly set PATH.

Example Code:

```
#!/bin/bash
. /home/user/.profile
```

Q22: `top` shows zombie processes.

Answer:

Zombies are dead processes not reaped by parent. Identify parent PID, restart it if safe.

Sample Points:

- Zombies don’t consume CPU.
- Restart parent to clear.
- Orphan adoption by init reaps.

Example Code:

```
ps -el | grepZ
```

Q23: Ulimit prevents app from opening more than 1024 files.

Answer:
Increase `nofile` in `/etc/security/limits.conf` and ensure PAM configs load it.
Sample Points:

- Raise limits in limits.conf.
- Check PAM session files.
- Apply at systemd service level.
  Example Code:

```
LimitNOFILE=65535
```

Q24: Package install fails due to broken apt/yum repo.

Answer:
Check repo URL in sources, refresh cache, disable failing repo temporarily.

Sample Points:

- Validate repo URL.
- Refresh metadata.
- Disable if not needed.
  Example Code:

```
sudo apt update
sudo yum --disablerepo=badrepo install pkg
```

Q25: Need to audit recent sudo commands run by a user.

Answer:
Check `/var/log/auth.log` or journal with `journalctl _COMM=sudo`.
Sample Points:

- Auth log records sudo usage.

- Use journalctl for query.
- Track time + command run.

Example Code:

```
sudo journalctl _COMM=sudo
```

DevOps — Terraform (25 Questions)

Q1: Terraform apply in prod fails with `Error locking state` because another apply is in progress, but that process crashed.

Answer:
If using an S3 backend with DynamoDB lock table, manually remove the stale lock item in DynamoDB **only afterconfirmingnootherappliesarerunning** . Then re-run. Consider -lock-timeout to wait for locks in future.

SamplePoints:

- Never delete lock unless sure it's stale.
- DynamoDB item removal unblocks state.
- Use -lock-timeout for busy teams.

Example Code:

```
aws dynamodb delete-item --table-name tf-lock --key '{"LockID": {"S": "prod/terraform.tfstate-md5"}}'
```

Q2: You must migrate manually created AWS resources into Terraform without downtime.

Answer:
Use terraform import to bring existing resources into state, then run terraform plan to ensure config matches reality. Avoid changing properties that would force replacement unless planned.

Sample Points:

- Import keeps IDs; no recreation.
- Align config with live settings before apply.
- Plan first to detect drifts.

Example Code:

```
terraform import aws_s3_bucket.mybucket my-bucket-name
```

Q3: Remote backend state on S3 shows drift after manual console changes.

Answer:
Run `terraform plan` to see differences. If drift is intended, update `.tf` files; if not, apply to revert. Use `terraform state rm` for resources no longer managed.

SamplePoints:

- Plan → decide to update config or infra.
- State rm for unmanaged resources.
- Avoid console edits in IaC environments.

Example Code:

```
terraform state rm aws_security_group.unmanaged
```

Q4: Terraform plan for a new VPC fails due to CIDR block overlap.

Answer:
Adjust CIDRs to non-overlapping ranges or use `cidrsubnet()` function for consistent

derivation. In multi-env setups, store base CIDR in a shared tfvars file.

Sample Points:

- Overlap blocks VPC creation.
- Derive subnets to avoid manual mistakes.
- Centralize network config.

Example Code:

```
cidrsubnet(var.vpc_cidr, 4, count.index)
```

Q5: Applying a module update forces recreation of critical RDS instance.

Answer:

Check if immutable properties (e.g., `allocated_storage` in some engines) changed. Use `lifecycle{prevent_destroy = true }` and perform updates with in-place changes only.

Sample Points:

- Prevent destroy for prod DBs.
- Review module changelog before apply.
- Change params in maintenance window.

Example Code:

```
lifecycle {
  prevent_destroy = true
}
```

Q6: Terraform workspace `prod` accidentally applied dev resources.

Answer:

Separate state files per environment instead of relying solely on workspaces, or use different backend key paths. Enforce var.environment as a required var.

Sample Points:

- Workspaces not isolation alone.
- Different backend keys safer.
- Always pass environment explicitly.

Example Code:

```
backend "s3" {
  key = "prod/terraform.tfstate"
}
```

Q7: `terraform destroy` in staging deleted a shared VPC used by prod.

Answer:
Add prevent_destroy lifecycle on shared resources, and separate shared infra into its own state. Enforce tagging and IAM policy to block deletes in shared projects.

Sample Points:

- Lifecycle protect critical shared assets.
- Separate state files for shared vs env-specific.
- IAM denies for delete API calls in shared.

Example Code:

```
lifecycle { prevent_destroy = true}
```

Q8: Applying infra in CI/CD fails due to missing provider creds.

Answer:
Inject creds via environment variables or use workload identity federation (OIDC) for ephemeral credentials in CI. Avoid hardcoding in tfvars.

Sample Points:

- CI pipeline must set creds securely.

- Prefer OIDC over long-lived keys.
- Use TF_VAR_ prefix for vars.

Example Code:

```
export AWS_ROLE_ARN=arn:aws:iam::123:role/tf-role
terraform apply
```

Q9: Terraform state file contains secrets.

Answer:
State stores all computed values. Use sensitive = true in variables, enable state encryption (S3 SSE or Vault backend), and limit state access via IAM.

Sample Points:

- Mark vars sensitive.
- Encrypt state at rest.
- Restrict read access.

Example Code:

```
variable "db_password"{
 type       = string
 sensitive = true
}
```

Q10: Resource creation fails due to provider version mismatch in team members' local setups.

Answer:
Pin provider versions in `required_providers` and commit `.terraform.lock.hcl` to repo.

Sample Points:

- Pin provider version for consistency.

- Lock file ensures deterministic builds.
- Upgrade via `terraforminit-upgrade`.
  Example Code:

```
terraform {
  required_providers {
    aws = { source = "hashicorp/aws", version = "~> 5.0"}
 }
}
```

Q11: Terraform plan is slow due to thousands of resources.

Answer:
Use -target for scoped applies when safe, split resources into multiple states/modules, and enable provider-side filtering.

Sample Points:

- Target only changed modules.
- Split state for scale.
- Provider filters speed up queries.
  Example Code:

```
terraform plan -target=module.network
```

Q12: Need to roll back to a previous infrastructure version after failed deployment.

Answer:
Check state history in backend (e.g., S3 versioning), download older state, and `terraform apply` with matching config.

Sample Points:

- Keep state versioning enabled.

- Rollback means re-applying older config.
- Document rollback procedure.
  Example Code:

```
aws s3 cp s3://bucket/prod.tfstate version-id=xyz ./terraform.tfstate
```

Q13: Developers must create S3 buckets only via Terraform, never manually.

Answer:
Use AWS Config or Cloud Custodian to detect manual resources, and enforce creation via CI pipelines with Terraform. Block manual create API via IAM condition on aws:ViaAWSService.

Sample Points:

- Guardrails + detection for manual drift.
- CI as only entrypoint for IaC changes.
- IAM denies outside Terraform role.
  Example Code:

```
"Condition":{"StringNotEquals":{"aws:CalledVia":"cloudformation.amazonaws.com"}}
```

Q14: Team often forgets to run `terraform fmt` before committing.

Answer:
Add pre-commit hook to run `terraform fmt -recursive` and fail commit on diff.

Sample Points:

- Pre-commit enforces formatting.
- CI can also check.
- Consistent style reduces churn.
  Example Code:

```
#!/bin/sh
terraform fmt -recursive -check || exit1
```

Q15: Need to generate unique names for resources across environments.

Answer:

Use `format()` with `var.environment` and random provider (`random_id`) to suffix names.

Sample Points:

- Combine env + random for uniqueness.
- Avoid collisions in global namespaces.
- Keep naming convention consistent.

Example Code:

```
resource "aws_s3_bucket" "b" {
  bucket = "${var.environment}-${random_id.suffix.hex}"
}
```

Q16: Provider credentials expire mid-apply for long deployments.

Answer:

Use short modules with smaller applies or refresh creds via token renewal during apply. For AWS, use session durations > apply time.

Sample Points:

- Split large applies.
- Extend token lifetime.
- Orchestrate apply steps.

Example Code:

```
aws sts assume-role --duration-seconds 3600
```

Q17: Need to avoid recreating immutable resource IDs when only tags change.

Answer:

Use `ignore_changes = [tags]` in lifecycle.

Sample Points:

- Ignore tag changes for immutable IDs.
- Helps for vendor-managed resources.
- Still apply tags manually if needed.

Example Code:

```
lifecycle {
  ignore_changes = [tags]
}
```

Q18: Must run a script after resource creation but before marking apply complete.

Answer:

Use `local-exec` provisioner with `when = create`. Keep idempotent.

Sample Points:

- Provisioners last resort.
- Idempotent scripts avoid drift.
- Prefer user_data/cloud-init when possible.

Example Code:

```
provisioner "local-exec" {
  when    = create
  command = "echo ${self.id} >> created.log"
}
```

Q19: Two resources depend on each other cyclically in config.

Answer:
Break the cycle by creating one with minimal config, outputting an ID, and updating the other in asecond apply. Or use depends_on to force order.

Sample Points:

- Break cycles into stages.
- Use depends_on explicitly.
- Avoid mutual references.

Example Code:

```
depends_on = [aws_security_group.sg]
```

Q20: Terraform in CI/CD must plan without exposing secrets in logs.

Answer:
Mark variables as sensitive and use `terraform plan -out=planfile` without showing values.

Sample Points:

- Sensitive vars mask in logs.
- Use planfile to hide secrets.
- Avoid -var with secrets inline.

Example Code:

```
terraform plan -out=planfile
```

Q21: Must ensure all resources are tagged with `Owner` and `Environment`.

Answer:

Use `default_tags` in provider config (AWS >=3.38), or a tagging module. Validate via `terraform validate` + custom rules.

Sample Points:

- Default tags enforce globally.
- Validation prevents missing tags.
- Use TF Cloud policy sets.

Example Code:

```
 provider "aws" {
  default_tags {
    tags = { Owner = var.owner, Environment = var.environment}
 }
}
```

Q22: Multi-cloud project requires different providers in same root module.

Answer:
Alias providers (`provider "aws" { alias = "east" }`) and pass into resources/modules.

Sample Points:

- Aliases for multi-provider configs.
- Pass providers explicitly to modules.
- Keep creds separate per alias.

Example Code:

```
provider "aws" { alias = "east" region = "us-east-1"}
```

Q23: Remote state backend migration from local to S3.

Answer:
Run `terraform init -migrate-state` after defining new backend block.

Sample Points:

- Migrate state without losing resources.

- Backend config in .tf file.
- Plan after migration to verify.
  Example Code:

```
terraform init -migrate-state
```

Q24: Must enforce no `terraform apply` in prod without PR approval.

Answer:
Use TF Cloud/Terragrunt with run tasks, or CI pipeline with approval stage before `apply` in prod workspace.
Sample Points:

- Manual gate before prod apply.
- PR review ensures compliance.
- Use TF Cloud policy-as-code.
  Example Code:

```
- stage: Approval
jobs:
    - manual: true
```

Q25: Need to refactor large root module into reusable modules without downtime.

Answer:
Refactor incrementally:

1. Extract a resource to new module.
2. Use terraform state mv to match new address.
3. Plan/apply with no changes. Repeat.
   Sample Points:

- State mv avoids recreation.
- One resource/module at a time.
- Validate after each step.

Example Code:

```
terraform state mv aws_s3_bucket.old module.new.aws_s3_bucket.old
```

DevOps — Jenkins (25 Questions)

Q1: A Jenkins pipeline intermittently fails during `docker build` with "no space left on device" on the agent node.

Answer:
Check workspace cleanup policies — old builds might be consuming disk. Enable `Workspace Cleanup` post-build, prune unused Docker images/volumes (`docker system prune`), and consider moving Docker storage to a larger disk. For multi-tenant agents, enforce quota per build.

Sample Points:

- Prune Docker artifacts periodically.
- Workspace cleanup plugin.
- Larger disk or separate volume for `/var/lib/docker`.

Example Code:

```
post {
    always {
        cleanWs()
        sh 'docker system prune -af'
    }
```

```
}
```

Q2: Jenkins shared library update isn’t reflecting in new pipeline runs.

Answer:
If library is cached, ensure @Library('my-lib@main') _ is used with version/tag. In Jenkins global config, set “Load implicitly” to false for manual control, and clear the SCM cache in `$JENKINS_HOME/caches`.

Sample Points:

- Explicit version tag avoids stale refs.
- Clear cache when updating.
- Avoid master default for stability.

Example Code:

```
@Library('my-lib@v1.2.3') _
```

Q3: Multi-branch pipeline triggers twice on a single commit.

Answer:
Likely duplicate webhooks or SCM polling enabled alongside webhooks. Remove extra webhook in repo settings, disable polling triggers in pipeline config.

Sample Points:

- Avoid mixed triggers (webhook + polling).
- Audit SCM webhook configuration.
- Use a single trigger source.

Example Code:

```
triggers { githubPush()}
```

Q4: Pipeline using `parallel` stages sometimes skips a branch without error.

Answer:

Ensure `failFast: false` if you want all branches to run independently. Also wrap each branch in `catchError` to prevent one failure stopping others.

SamplePoints:

- Use failFast intentionally.
- Catch errors in each parallel branch.
- Aggregate results after all run.

Example Code:

```
parallel(
  branch1: { catchError { sh 'run-tests.sh' } },
  branch2: { catchError { sh 'lint.sh' } },
  failFast: false
)
```

Q5: Credentials binding to environment variables leaks in console output.

Answer:

Use `withCredentials` block to mask secrets and avoid echoing variables. Enable “Mask Passwords” plugin, set `echo false`.

Sample Points:

- Never echo secrets.
- Use binding plugins to mask automatically.
- Store in Jenkins Credentials store only.

Example Code:

```
withCredentials([usernamePassword(credentialsId: 'dockerhub',
usernameVariable: 'USER', passwordVariable: 'PASS')]) {
    sh 'docker login -u $USER -p $PASS'
```

```
}
```

Q6: Distributed builds fail on a new agent with “Permission denied” when accessing workspace.

Answer:

Check that agent user has write permissions to Jenkins home and workspace dirs. Align file ownership between master and agents, and consider using chown post-workspace creation.

Sample Points:

- File ownership consistency.
- Use dedicated Jenkins user on agents.
- Avoid running as root unless required.

Example Code:

```
sudo chown -R jenkins:jenkins /var/lib/jenkins/workspace
```

Q7: You must dynamically provision agents for Kubernetes builds.

Answer:

Install Kubernetes plugin, define pod templates with container specs for build environments, use podTemplate in pipelines. Limit pod idle timeout for cost savings.

Sample Points:

- Pod templates per job type.
- Idle timeout to control costs.
- Avoid privileged containers unless needed.

Example Code:

```
podTemplate(label: 'maven-agent', containers: [
  containerTemplate(name: 'maven', image: 'maven:3.8', ttyEnabled:
true, command: 'cat')
```

```
]) {
    node('maven-agent') { sh 'mvn clean install'}
}
```

Q8: Pipeline artifacts aren’t available to downstream jobs.

Answer:
Use archiveArtifacts in upstream and copyArtifacts plugin in downstream, matching build numbers or tags. Ensure retention policy keeps artifacts until downstream completes.

Sample Points:

- Archive artifacts in upstream.
- Match build IDs in copy step.
- Adjust retention policies.

Example Code:

```
archiveArtifacts artifacts: '**/target/*.jar', fingerprint: true
```

Q9: Need to fail pipeline if code coverage drops below 80%.

Answer:
Parse coverage report (e.g., Jacoco XML) in a stage, compare value to threshold, fail build with error() if below.

Sample Points:

- Automate coverage checks.
- Fail early to enforce quality.
- Use coverage plugins or scripts.

Example Code:

```
def coverage = readFile('coverage.txt').trim().toInteger()
if (coverage < 80) error "Coverage below 80%: $coverage%"
```

Q10: Jenkins master CPU spikes due to concurrent builds.

Answer:
Move heavy jobs to agents, reduce master's executor count to 0, and use the master only for orchestration.

Sample Points:

- Master for orchestration only.
- Offload builds to agents.
- Reduce master executor count.

Example Code:

```
node('build-agent') { sh 'mvn clean install'}
```

Q11: Pipeline must rollback to previous app version on deployment failure.

Answer:
Use a post-deploy stage to check status and call rollback script or redeploy last stable artifact from `archiveArtifacts`.

Sample Points:

- Rollback script as part of pipeline.
- Keep last stable artifact.
- Automate detection of failure.

Example Code:

```
post {
    failure {
        sh './rollback.sh'
    }
}
```

Q12: Git shallow clone causes missing tags in Jenkins pipeline.

Answer:
Increase depth or disable shallow clone for builds requiring tags. Use `checkout` with `depth: 0` to fetch all history.
Sample Points:

- Shallow clone saves time but limits history.
- Fetch full history when tags are needed.
- Configure per-pipeline.
  Example Code:

```
checkout([$class: 'GitSCM', branches: [[name: '*/main']], extensions: [[$class: 'CloneOption', depth: 0]]])
```

Q13: Need to pass parameters from one pipeline to another triggered job.

Answer:
Use `build` step with `parameters` block. Ensure downstream is parameterized.
Sample Points:

- Upstream triggers with parameters.
- Downstream must accept parameters.
- Match parameter names exactly.
  Example Code:

```
build job: 'downstream', parameters: [string(name: 'VERSION', value: '1.0')]
```

Q14: Pipeline fails due to Maven repo corruption in agent cache.

Answer:
Clean local repo (`~/.m2/repository`) on failure, consider mounting a clean volume per build or using a cache restoration plugin.
Sample Points:

- Cache can corrupt over time.
- Periodic clean reduces flakiness.
- Use immutable cache artifacts.

Example Code:

```
sh 'rm -rf ~/.m2/repository && mvn clean install'
```

Q15: Secret text credential must be injected into a Docker build without leaking in image history.

Answer:
Use --build-arg with ARG in Dockerfile, then unset inside build step. Avoid embedding secrets in final layers.

Sample Points:

- Build args not persisted in layers.
- Avoid ADDing secret files.
- Mask secrets in Jenkins logs.

Example Code:

```
withCredentials([string(credentialsId: 'api-key', variable:
'API_KEY')]) {
    sh "docker build --build-arg API_KEY=$API_KEY ."
}
```

Q16: Long-running pipeline stage must auto-abort after 30 minutes.

Answer:
Wrap stage with `timeout` block. This prevents stuck builds from hogging agents.

Sample Points:

- Timeout for runaway stages.

- Auto-abort frees agents.
- Adjust per stage.

Example Code:

```
timeout(time: 30, unit: 'MINUTES') { sh './run-tests.sh'}
```

Q17: Blue Ocean view missing some pipeline stages.

Answer:
Ensure stages are defined with `stage('name')` and not inside raw script without stage. Blue Ocean only renders defined stages.

Sample Points:

- Define all logical steps as stages.
- Avoid script blocks swallowing stages.
- Use parallel stages for clarity.

Example Code:

```
stage('Build') { sh 'mvn clean package'}
```

Q18: Must conditionally skip deployment stage if commit message contains `[skip deploy]`.

Answer:
Read `env.GIT_COMMIT` message, skip stage with `when` directive.

Sample Points:

- Use regex in `when` for commit messages.
- Prevent accidental prod deploys.
- Lightweight safeguard.

Example Code:

```
stage('Deploy') {
  when { not { expression { sh(script: "git log -1 --pretty=%B | grep
-q '\\[skip deploy\\]'", returnStatus: true) != 0 } } }
  steps { sh './deploy.sh' }
}
```

Q19: Jenkins agent on Kubernetes fails due to missing Docker socket.

Answer:
Use DinD (Docker in Docker) container or mount host socket if security allows. Alternatively, build with Kaniko/Buildah in rootless mode.

Sample Points:

- Host socket mount vs DinD trade-offs.
- Kaniko avoids privileged mode.
- Security vs compatibility choice.

Example Code:

```
volumeMounts:
- mountPath: /var/run/docker.sock
  name: docker-sock
```

Q20: Pipeline must only run if specific files changed.

Answer:
Use `when { changeset pattern: 'src/**', comparator: 'ANT' }` to limit execution.

SamplePoints:

- Changeset condition reduces wasted builds.
- Pattern matching supports ANT/regex.
- Define in Jenkinsfile for visibility.

Example Code:

```
when { changeset pattern: "src/**", comparator: "ANT"}
```

Q21: Jenkins master restarted mid-build, and job didn’t resume.

Answer:
Ensure Pipeline is configured with `pipelinedurability hint: MAX_SURVIVABILITY` and Checkpoint plugin or external workspace to resume.

Sample Points:

- Durable pipelines survive restarts.
- Checkpoint allows resume points.
- Requires persistent workspace.

Example Code:

```
options { durabilityHint('MAX_SURVIVABILITY')}
```

Q22: Need to store pipeline logs in S3 for long-term audit.

Answer:
Use `S3Log Publisher` plugin or post-build script to upload logs to S3 with build metadata in the path.

SamplePoints:

- Long-term retention offloads from master.
- Metadata in S3 keys aids search.
- Secure bucket with IAM policy.

Example Code:

```
post {
    always {
        sh 'aws s3 cp ${BUILD_LOG}
s3://jenkins-logs/${JOB_NAME}/${BUILD_NUMBER}.log'
    }
```

```
}
```

Q23: Groovy sandbox prevents running certain methods in pipeline.

Answer:
Approve scripts in Script Approval page or run pipeline in “trusted” shared library. Limit who can bypass sandbox.
Sample Points:

- Sandbox protects from unsafe calls.
- Script Approval grants specific methods.
- Use shared libs for trusted code.
  Example Code:

```
@Library('trusted-lib') _
```

Q24: Pipeline must deploy to multiple environments sequentially with approval in between.

Answer:

Use `input` step between stages for manual approval.
Sample Points:

- Input pauses for human gate.
- Sequential deploys per environment.
- Timeout input to avoid hangs.
  Example Code:

```
stage('Deploy Staging') { steps { sh './deploy-staging.sh'}}
stage('Approval') { steps { input 'Deploy to Prod?'}}
stage('Deploy Prod') { steps { sh './deploy-prod.sh'}}
```

Q25: Jenkinsfile in monorepo must build only the affected service.

Answer:

Parse `git diff` to detect changed directories, set env var for service path, run targeted build.

Sample Points:

- Detect changed paths with git diff.
- Targeted build saves time/cost.
- Works in monorepo CI/CD.

Example Code:

```
def changed = sh(script: "git diff --name-only HEAD~1 | cut -d/ -f1|
sort -u", returnStdout: true).trim()
if (changed.contains('service-a')) { buildServiceA() }
```
