SAMPLE QUESTION Master Linux · 500+ real interview questions · Storage & Troubleshooting

df says 100%. du says 60%.

ASKED BY A TIER 4 SYSTEM ENGINEER

"It's 3 AM. Applications are failing with 'No space left on device'. `df -h` shows `/var` at 100%, but when you add up `du -sh /var` it only accounts for about 60%. Where is the rest of the space, and how do you get it back without rebooting the server?"

1. What the interviewer is really testing

Not whether you know the two commands. They want to see that you understand **how each tool measures space**, that you work in a calm order under pressure, and that you know the fix that **doesn't take production down further**. Candidates who answer "delete some logs" or "reboot" lose points here.

2. Why df and du disagree

`df` asks the **filesystem** how many blocks are used. `du` **walks the directory tree** and adds up the files it can see. Space that is used but not visible in the tree shows up as the gap.

Cause	What is happening	How you confirm it
Deleted but still open (most common in real life)	A log was deleted or rotated, but a process still holds the file open. The name is gone, the blocks are not freed until the process closes it.	`lsof +L1`
Mount shadowing	Files were written to `/var` before a separate partition was mounted on top. They still use space on the root disk, hidden under the mount.	bind-mount `/` and run `du`
Inodes exhausted	Not a gap, but the same error message. Millions of tiny files use all inodes while blocks remain.	`df -i`
Snapshots, sparse files, reserved blocks	LVM/btrfs snapshots, sparse files and the 5% root-reserved blocks on ext4 change what df reports.	`lvs`, `tune2fs -l`

3. How you actually work through it

Step 1: Confirm the problem, and rule out inodes.

```
$ df -hT /var
Filesystem             Type  Size  Used Avail Use% Mounted on
/dev/mapper/vg0-var    ext4   50G   50G     0 100% /var
$ df -i /var
Filesystem             Inodes  IUsed   IFree IUse% Mounted on
/dev/mapper/vg0-var    3276800 184211 3092589    6% /var   # inodes fine, so it's blocks
```

Step 2: See what `du` can account for. Use `-x` so it stays on this one filesystem.

```
$ du -xh --max-depth=1 /var | sort -rh | head
30G   /var
18G   /var/log
9.1G  /var/lib
...                  # about 30G visible, 50G used, so 20G is invisible
```

Step 3: Look for deleted files that are still open.

```
$ lsof +L1 | grep /var
COMMAND   PID   USER  FD  TYPE  SIZE/OFF       NLINK NAME
java     2814   app   4w  REG   20401094656  0     /var/log/app/app.log (deleted)
```

There it is: PID 2814 holds a **~19 GB log file** that someone (or a bad logrotate) already deleted. NLINK is 0, the name is gone, the space is not.

Step 4: Free the space without restarting the application. Truncate the file through the process's file descriptor.

```
$ : > /proc/2814/fd/4          # empties the deleted file; blocks are released immediately
$ df -h /var
/dev/mapper/vg0-var   50G   31G   17G  65% /var
```

Check the PID and FD number twice. Truncating the wrong descriptor can wipe a live database file.

Step 5: If lsof shows nothing, check for mount shadowing.

```
$ mkdir /mnt/rootview && mount --bind / /mnt/rootview
$ du -xh --max-depth=1 /mnt/rootview/var | sort -rh | head
$ umount /mnt/rootview
```

The bind mount shows what sits *underneath* the real mount. Move or delete the stray files from there.

4. Fix the cause, or it comes back

- **Logrotate:** make the service reopen its log after rotation (`postrotate systemctl reload app`, or a HUP signal). Use `copytruncate` only when the app cannot reload.
- **Monitoring:** alert at 80% on both `df -h` and `df -i`, so you find out at 2 PM, not 3 AM.
- **Design:** keep `/var/log` on its own volume so a runaway log cannot take down everything else.

5. The answer that gets you hired

> "`df` reads usage from the filesystem, `du` sums the files it can see, so a 40% gap usually means space is held by something with no visible name. First I rule out inodes with `df -i`. Then I run `lsof +L1` to find deleted-but-open files, which is the usual cause after a bad log rotation. If I find one, I truncate it through `/proc/PID/fd/N` so the application keeps running, then I fix logrotate so the service reopens its logs. If lsof is clean, I bind-mount the root filesystem to check for files hidden under a mount point."

6. Mistakes that fail the interview

- **"I'd rm the log files."** The file is already deleted. Deleting more will not free anything while the process holds it.
- **"I'd reboot."** It works, but you lose the evidence and cause downtime. Restart only the affected service if truncating isn't possible.
- **Running `du` without `-x`.** It crosses into other mounts and gives you misleading totals.
- **Skipping `df -i`.** "No space left" with free blocks is an inode problem.

7. Follow-ups to expect

- "The file is deleted, so how does `/proc` still let you reach it?" (The kernel keeps the inode alive while any open file descriptor references it.)
- "Why does ext4 show 0 available when Used is not 100%?" (Reserved blocks, 5% by default, kept for root.)
- "How would you stop this happening across 200 servers?" (Logrotate in config management, plus disk and inode alerts.)

Don't learn everything. Choose wisely. Crack it.