Linux Troubleshooting

1. Server is completely unresponsive (no SSH, no ping), but Serial Console / iLO / iDRAC works. Walk through your steps.

Answer

When a server stops responding to SSH and ping, but the out-of-band console works, the issue is **inside the OS**, not networking or hardware. Your goal is to diagnose without rebooting.

Step-by-Step Procedure

1. Login using Serial Console / iLO / iDRAC

This bypasses the network stack and gives you direct access.

2. Check if the system is overloaded

Run:

```
top
vmstat 1
uptime
```

Look for:

- Load average extremely high
- CPU 100% due to a stuck process
- Run queue stuck
- CPU soft lockup messages

3. Check if disk is 100% full

```
Full disk = SSH stops working.
df -h
mount | grep ro
```

```
If / is read-only → filesystem corruption.
```

4. Check network interface status

```
ip a
ip link
systemctl status network
journalctl -u network
```

Look for:

- Interface down
- Wrong IP
- DHCP failure
- NIC driver crash

Restart network service:

```
systemctl restart NetworkManager
```

5. Check firewall rules

Firewall may be blocking SSH and ICMP.

```
iptables -L -n
ufw status
```

```
Flush (temporary):
iptables -F
```

6. Check SSH service

```
systemctl status sshd
systemctl restart sshd
```

```
journalctl -u sshd
```

Fix common failures:

- Wrong permissions on `.ssh`
- SSHD config broken

7. Check kernel logs for panic or hardware errors

```
dmesg | tail -50
journalctl -xb
```

Look for:

- CPU soft lockup
- I/O errors
- Memory errors
- Kernel panic

8. If NIC driver is stuck, reload it

Example:

```
modprobe -r e1000e
modprobe e1000e
systemctl restart network
```

9. As last resort → controlled reboot

If:

- filesystem corrupted

- kernel panic
- stuck I/O

Then reboot via console, not AWS UI:

reboot

2. Disk is 100% full on a production server. First 5 commands and actions.

Answer

When disk is full, services crash, logs stop, SSH fails. Your job is to free space immediately without restarting.

1. Check filesystem usage

```
df -h
```

Identify:

- Which partition is full
- Usually `/` or `/var` or `/home`

2. Find the biggest directories

```
du -sh /* | sort -h

Then drill deeper:
du -sh /var/* | sort -h
```

Common culprits:

- `/var/log`

- `/var/lib/docker`
- `/tmp`

3. Find deleted files still held by processes

These do NOT show up in `du` or `df`.

lsof | grep deleted

If you see:

java 1234 10u /var/log/app.log (deleted)

Space will be released only after killing that process:

kill -9 1234

4. Clear journal logs

```
journalctl --vacuum-size=500M
```

5. Clear large rotated logs

```
rm -f /var/log/*.gz
rm -f /var/log/*-202*
> /var/log/messages
> /var/log/syslog
```

3. How do you find which process is holding a deleted file open (eating disk space)?

Answer

When a file is deleted but still opened by a running process, disk usage stays high.

Command:

lsof | grep deleted

Output example:

java 3245 txt REG /var/log/app.log (deleted)

Fix Steps

1. Identify process

Kill it:

2. Disk space instantly freed.
3. kill -9 3245

4. Server shows high load average (50+) but CPU and I/O are low. What is happening and how to prove?

Answer

High load average WITHOUT CPU or disk usage means:

Tasks stuck in D-state (uninterruptible sleep).

The load average counts:

- running processes
- **blocked I/O tasks (D-state)**

How to prove

```
ps -eo pid,state,cmd | grep "^.D"
```

You'll see tasks in `D` state.

Why it happens:

- NFS server down
- SAN / EBS latency
- Disk controller failure
- Kernel bug

Confirm with:

dmesg | grep -i "i/o"

5. A process is stuck in D state and killing does nothing. Explain & resolve.

Answer

A **D-state** task is waiting for kernel I/O and **cannot be killed**, even with `kill -9`.

Causes

- Disk I/O timeout
- NFS server unreachable
- Bad block on disk
- Controller failure

How to confirm

```
ps aux | grep D
dmesg | tail
```

If you see:

block device error on sda

Fix

If NFS hung:

```
umount -f /mount/path
```

- If disk error:
    - Stop services
    - Reboot into rescue
    - Run `fsck`

A reboot is often required because D-state processes are stuck inside kernel.

6. Application logs show "Cannot fork: Out of memory", but free -m shows 60% free RAM. Why?

✅ Answer

This happens when system hits **process limit**, not memory limit.

Two possible causes

1. Process limit reached (ulimit -u)

Check:

ulimit -u

2. PID exhaustion

Linux cannot create new processes if PIDs are exhausted.

Check:

cat /proc/sys/kernel/pid_max

and

ps -eLf | wc -l

Result

System cannot fork new processes → application fails even if memory is available.

7. “Too many open files” error — Immediate & permanent fix.

✅ Answer

Immediate fix

ulimit -n 65535

Permanent fix

Edit:

/etc/security/limits.conf

Add:

* soft nofile 65535
* hard nofile 65535

Apply at system-wide PAM level

/etc/pam.d/common-session

---

8. Server boots into Emergency Mode after reboot — how to recover without data loss?

✅ Answer

Emergency mode means the system encountered:

- corrupted filesystem
- broken `/etc/fstab`
- missing mount
- LVM failure

Step-by-step fix

1. Enter emergency shell

You'll get a root shell.

---

2. Check error logs

journalctl -xb

---

3. Fix incorrect /etc/fstab entry

Example issue:

UUID=xxxx /data ext4 defaults 0 2

But disk missing.

Edit:

nano /etc/fstab
comment incorrect entry

4. If filesystem error
fsck /dev/sda1

5. Reboot
reboot

9. `sudo: command not found` after package update — fix without reboot.

Answer

Causes:

- PATH broken
- sudo binary removed
- sudo package corrupted

Fix PATH first

export PATH=/usr/bin:/usr/sbin:/bin:/sbin

Reinstall sudo

For Ubuntu:

apt install --reinstall sudo

For RHEL/CentOS:

yum reinstall sudo

If sudo binary exists but missing links

```
ln -s /usr/bin/sudo /bin/sudo
```

10. Time on the server is off by 4 hours — how to fix and prevent it?

Answer

Time issues break logs, apps, and authentication.

Fix Time

1. Set correct timezone
timedatectl set-timezone Asia/Kolkata

2. Restart time sync service

Ubuntu:

systemctl restart systemd-timesyncd

RHEL:

systemctl restart chronyd

3. Force sync
timedatectl set-ntp true

Prevent Future Issues

- Ensure NTP servers reachable
- Disable manual time changes
- Ensure VM host clock is correct

Docker & Container Troubleshooting

11. Container exits immediately with exit code 137. What happened, and how do you confirm?
12. Cannot connect to Docker daemon" error top 3 causes and fixes.
13. Docker build fails with "no space left on device" but `df -h` shows space. Why?

14. Container is using 15GB memory but host shows only 2GB used by Docker. Explain.

11. Container exits immediately with exit code 137. What happened, and how do you confirm it?

✅ Answer

Exit code 137 = container was killed by the kernel (SIGKILL / signal 9).

This happens due to:

- **OOM (Out of Memory) kill**
- **Kernel killed it to protect the host**

How to confirm

Run:

dmesg | grep -i "kill"

Typical output:

Out of memory: Kill process 2134 (java) score 982

Container Logs

docker logs <container>

Fix

- Increase memory limit:
  `docker run -m 4g ...`
- Fix memory leaks
- Use swap (not recommended in production)
- Reduce JVM / Python memory options

12. “Cannot connect to Docker daemon” — top 3 causes and fixes

✅ Answer

1. Docker service not running

Check:

systemctl status docker
systemctl start docker

---

2. Permission issue on docker.sock

Error:

Got permission denied while trying to connect to socket...

Fix:

sudo usermod -aG docker $USER
newgrp docker

---

3. Wrong or missing Docker socket

Check:

ls -l /var/run/docker.sock

If missing:

systemctl restart docker

---

13. Docker build fails with “no space left on device” but `df -h` shows space. Why?

✅ Answer

This happens because:

Docker uses a separate storage backend:

- `/var/lib/docker/overlay2`
- `/var/lib/docker/aufs`
- `/var/lib/docker/containers`

Even if your root (`/`) partition has space, the **Docker storage layer** may be full.

Check actual Docker usage

docker system df
du -sh /var/lib/docker/*

Fix

docker system prune -af
docker builder prune --all

14. Container uses 15GB RAM but host shows only 2GB used by Docker. Explain.

✅ Answer

Docker memory usage ≠ host memory usage due to:

1. Page cache

Container memory counted inside container but not shown under docker processes.

2. Memory cgroups

Container can use:

- **RSS memory**
- **Page cache**
- **Kernel memory**
- **tmpfs memory**

Host only shows RSS, not full cgroup usage.

3. Shared memory

`/dev/shm` inside container can consume large memory.

How to confirm

cat /sys/fs/cgroup/memory/docker/<container-id>/memory.usage_in_bytes

This shows actual memory usage.

15. Pod stuck in CrashLoopBackOff — your debugging sequence

✅ Answer

1. Describe pod

kubectl describe pod <name>

Look for:

- OOMKilled
- Invalid args
- Failed liveness probe

---

2. Check current logs

kubectl logs <pod>

3. Check previous crashed container logs

kubectl logs <pod> -p

4. Check events

kubectl get events --sort-by=.metadata.creationTimestamp

Common Causes & Fixes

- Wrong ENV variables → fix ConfigMap/Secret
- App fails instantly → add startup timers
- Liveness probe too aggressive → increase timeout
- Out of memory → increase resources

16. Pod in ImagePullBackOff — fix in under 5 minutes

✅ Answer

1. Describe pod

kubectl describe pod <pod>

Look under *Events*:

- wrong image:tag
- permission denied

- pull rate limit
- DNS issue

Quick Fixes

A. Wrong image/tag

kubectl edit deployment <name>
update the image

B. Private repo — missing secret

Create secret:

```
kubectl create secret docker-registry regcred \
  --docker-username=... \
  --docker-password=...
```

Attach to deployment:

```
imagePullSecrets:
  - name: regcred
```

C. DNS issues

kubectl exec -it <pod> -- nslookup google.com

If fails → node DNS broken → restart kubelet.

17. Pod is Running but application is not accessible — step-by-step checks

✅ Answer

This is the **most common K8s debugging scenario**.

1. Check service

kubectl get svc

Correct port? Right service type?

2. Check service → pod connectivity

kubectl exec -it <pod> -- curl http://<pod-ip>:<port>

3. Check targetPort vs containerPort mismatch

Example:
Service exposes 80 but container listens on 8080.

4. Check network policies

kubectl get networkpolicy

They may be blocking traffic.

5. Check NodePort / LoadBalancer

Test from node:

curl <node-ip>:<nodePort>

6. Check Ingress

Validate:

- host rules
- path
- TLS

7. Check firewall / Security group (cloud)

AWS SG commonly blocks NodePort 30000–32767.

18. Node went NotReady after kubelet restart — top reasons and fixes

✅ Answer

1. Swap enabled

Kubelet refuses to start if swap=on.

Fix:

swapoff -a

2. Cgroup driver mismatch

Error:

failed to run Kubelet: misconfigured cgroup driver

Fix:
Ensure Docker/containerd uses same cgroup as kubelet:

- systemd
- cgroupfs

3. Container runtime down

systemctl status containerd
systemctl start containerd

4. Disk full

Check:

df -h

5. Kubelet certificate expired

Fix:

kubeadm cert renew all
systemctl restart kubelet

19. All pods evicted with “node affinity” or “taint” issues — how to resolve fast?

✅ Answer

Pod eviction happens when node **taints** or **affinity rules** prevent scheduling.

1. List taints

kubectl describe node <node>

Remove taint

kubectl taint nodes <node> key=value:NoSchedule-

2. Check pod affinity/anti-affinity rules

kubectl get deployment -o yaml

Fix:

- Remove strict affinity
- Use soft rules

3. Check disk pressure / memory pressure

kubectl describe node <node>

If `DiskPressure=True` → clear space in `/var/lib/kubelet`.

20. Kubernetes dashboard or metrics-server not working — common causes

Answer

Metrics-server common issues

1. Cannot reach kubelets

Error:

unable to fetch metrics from kubelet

Fix—add flags:

--kubelet-insecure-tls
--kubelet-preferred-address-types=InternalIP

2. Missing APIService registration

Check:

kubectl get apiservices | grep metrics

3. RBAC missing

Dashboard not visible → missing ClusterRoleBinding.

Fix:

```
kubectl create clusterrolebinding kubernetes-dashboard \
  --clusterrole=cluster-admin \
  --serviceaccount=kubernetes-dashboard:kubernetes-dashboard
```

4. Wrong Service type

If Dashboard is ClusterIP → not accessible externally.

Change to:

type: NodePort

or use a secure Ingress.

Here you go bro — **Part 3 (Questions 21–30)** with **proper questions + detailed, production-grade answers**.
Same clean, interview-ready format.

🟩 AWS TROUBLESHOOTING — PART 3 (21–30)

---

21. EC2 instance unreachable but status checks 2/2 passed — what exact steps do you follow?

✅ Answer

If **2/2 checks passed**, AWS confirms:

- Hypervisor OK
- Networking OK
- OS booted

This means **the issue is inside the OS or networking configuration**.

STEP 1 — Check Security Group

Ensure:

- Inbound: port 22 (SSH) open
- Source allowed (Your IP / VPN / 0.0.0.0/0 temporarily)

STEP 2 — Check NACL rules

NACLs are stateless → both inbound & outbound must allow:

- Port 22
- Ephemeral ports 1024-65535

STEP 3 — Check Route Table

The EC2 must have route:

- `0.0.0.0/0` → Internet Gateway (public subnet)
- OR route to NAT Gateway (private subnet)

STEP 4 — Check OS-level firewall

If you connected before but now can't → firewall (iptables/firewalld) probably changed.

Use **EC2 Serial Console**:

sudo iptables -L
sudo systemctl stop firewalld

STEP 5 — Check SSH demon

Via Serial Console:

systemctl status sshd
systemctl restart sshd
journalctl -u sshd

STEP 6 — Check interface config inside OS

ip a
cat /etc/netplan/*.yaml # Ubuntu
cat /etc/sysconfig/network-scripts/ifcfg-eth0 # RHEL

Fix wrong IP / subnet mask / gateway.

STEP 7 — Replace corrupted authorized_keys

Maybe edited or lost due to AMI update.

Conclusion

If 2/2 checks passed → **network path to Amazon is fine, OS has misconfiguration.**
Serial Console is your best friend.

---

22. EC2 failed system status check (0/2 or 1/2). How do you recover the instance?

✅ Answer

System status check failure = AWS infrastructure issue, not your OS.

Common reasons:

- Physical hardware failure
- Underlying host degraded
- Network card failure
- AWS hypervisor crash

How to recover

STEP 1 — Stop/Start (NOT reboot)

This moves instance to a new physical hardware.

Stop → Start (new host)

You keep:

- Same private IP
- Same EBS volume
- Same data

Public IP changes unless Elastic IP.

STEP 2 — Check underlying hardware events

CloudWatch → Event history

Look for:

instance-retirement
system-maintenance

STEP 3 — Create AMI for safety (optional)

If hardware recovery fails, create AMI and launch new instance.

STEP 4 — Open AWS Support Ticket

For persistent 0/2 failures.

---

23. You lost the .pem key of a running EC2. How do you regain SSH access?

✅ Answer

There is **NO way** to recover a lost PEM, but you can **add a new SSH key**.

Method (100% Working)

STEP 1 — Stop the instance

Important: **do NOT terminate**.

STEP 2 — Detach root EBS volume

From EC2 console:

- Detach `/dev/xvda`

STEP 3 — Attach the volume to another EC2

Attach as:

/dev/sdf

STEP 4 — Mount it

On helper EC2:

sudo mkdir /mnt/recover
sudo mount /dev/xvdf1 /mnt/recover

STEP 5 — Add new SSH key

Edit:

/mnt/recover/home/ec2-user/.ssh/authorized_keys

Paste your new SSH public key.

STEP 6 — Reattach volume to original instance

Attach back as:

/dev/xvda

STEP 7 — Start the instance and SSH using new key

This works for:

- Amazon Linux
- Ubuntu
- RHEL/CentOS

---

24. ALB showing 502/504 errors — systematic troubleshooting

✅ Answer

A 502 means:

Backend responded incorrectly or closed connection.

A 504 means:

Backend timeout.

STEP 1 — Check Target Group Health

403, 404, 500 = unhealthy
200 = healthy

STEP 2 — Verify health check path

ALB default:

/

But your app may serve:

/health
/api/status

Wrong path → 502.

STEP 3 — Check backend port

Most common mistake:

ALB → port 80
App → listening on 8080
→ Causes 502

STEP 4 — Check Security Groups

Backend SG must **allow inbound from ALB SG only**.

STEP 5 — Application level logs

Check:

- Nginx/Apache
- Node.js
- Gunicorn
- JVM crashes

STEP 6 — Timeout mismatch

If app timeout < ALB idle timeout → 504.

Fix:

- Change ALB timeout
- Increase app timeout

STEP 7 — Check DNS resolution

Microservice dependencies failing can cause 5xx.

Cause Summary

Error	Meaning
502	Bad response from backend
504	Backend did not respond in time

---

25. Target group showing “Unhealthy” — full checklist

✅ Answer

1. Target health check path correct?

/health
/ready
/status

---

2. Target health check port correct?

If service listens on 8080:

- Target group must also use 8080

---

3. Security Group allows inbound from ALB

Backend SG must allow:

source: ALB security group
port: health check port

---

4. Application returning 200 OK

Return codes allowed list:

200-299

If app returns:

- 301
- 401
- 404
  → unhealthy

---

5. Check that EC2 instance is listening

sudo netstat -tulnp | grep <port>

---

6. Routing / NACLs

Ensure:

- Outbound ephemeral ports open
- Routing to ALB reachable

---

7. App-level issues

Slow start? Add:

healthy_threshold: 5

interval: 10
timeout: 5

---

26. RDS error: “could not connect to server” — full debugging steps

✅ Answer

1. Check RDS instance status

- Available?
- Modifying?
- Storage full?

---

2. Check Security Groups

Inbound must allow:

- Port 3306 (MySQL)
- Port 5432 (Postgres)
- From application servers

---

3. Check Subnet Group

Ensure:

- RDS in correct private subnets
- Route to app servers exists

---

4. Check Network ACL

Must allow both inbound & outbound.

---

5. Validate DNS

nslookup your-db.xxxxx.rds.amazonaws.com

---

6. Test connectivity from EC2

telnet <rds-endpoint> 5432

If connection refused → SG issue
If no response → routing issue

---

7. Check Parameter Group

Port changes? SSL required?

---

8. RDS Logs

Check performance insights:

- Slow queries
- Connection pool exhausted

---

27. RDS CPU 100% — immediate actions + long-term solution

✅ Immediate Actions

1. Identify the heavy query

Use Performance Insights.

---

2. Kill culprit session

For MySQL:

```
SHOW PROCESSLIST;
KILL <id>;
```

For Postgres:

```
SELECT * FROM pg_stat_activity;
SELECT pg_terminate_backend(pid);
```

---

3. Scale instance (vertical scaling)

Increase:

- vCPU
- RAM

Takes < 2 minutes.

---

4. Enable auto-scaling (Aurora)

Long-Term Fixes

- Add indexes
- Fix slow queries

- Use caching (Redis/ElastiCache)
- Increase connection pool
- Add read replicas

---

28. S3 giving 503 SlowDown — cause & fix

✅ Cause

You are hitting **S3 request rate limits** for a single prefix.

Example:

bucket/folder/a/a/a/a...

Single prefix cannot handle massive parallel GET/PUT.

Fix

1. Randomize keys

Replace:

user123/profile.png

with:

12/user123/profile.png

2. Use UUID-based prefixing

S3 automatically load-balances better.

---

3. Enable S3 Transfer Acceleration

Reduces latency globally.

---

4. Add retries with exponential backoff in code

---

29. CloudFront 502/504 errors — differentiate origin vs edge issue

✅ Quick Rule

502 = Origin Problem

CloudFront connected to origin but got a bad response.

504 = Timeout (origin or network)

CloudFront waited too long.

How to differentiate

A. Check Origin Response

Access origin directly:

curl http://your-origin.com

If origin also returns 502 → issue is origin.

B. Check CloudFront console error details

Shows:

- Origin timeout
- Connection refused
- SSL handshake failed

C. Check CloudFront distributions

Misconfigurations:

- Wrong origin path
- Wrong protocol policy
- Bad cache behavior

D. Check WAF / Shield rules

WAF blocking → CloudFront returns 502.

---

30. Lambda timing out at 15 minutes — how to debug & fix

✅ Answer

AWS Lambda has **maximum allowed timeout = 15 minutes**.
If hitting exactly 15 minutes → your code is inefficient or waiting on slow dependency.

Debug Steps

1. View CloudWatch Logs

Look for:

- External API calls hanging
- DB queries hanging
- Infinite loops
- Large file processing

---

2. Test function locally using SAM

To reproduce slowness.

---

3. Add logging checkpoints

At start and end of each major block.

Fix Options

A. If code really needs > 15 min

Move workload to:

- ECS Fargate
- Lambda + SQS + Step Functions
- AWS Batch

B. Improve performance

- Use async calls

- Increase memory (Lambda CPU scales with memory)
- Use VPC endpoints for DB/S3 to reduce latency

C. Check cold starts

Enable Provisioned Concurrency.

Here you go bro — **Part 4 (Questions 31–40)** with **clear questions + detailed, production-level answers**.

🟦 AWS + DEVOPS TOOLS TROUBLESHOOTING — PART 4 (31–40)

---

31. EKS worker nodes not joining the cluster — top 5 reasons

✅ Answer

If worker nodes are “Ready = False” or never appear, it means the node **could not register with the API server**.

Top 5 Causes & Fixes

---

1. Wrong IAM Role attached to Node Group

Nodes need the following policies:

- `AmazonEKSWorkerNodePolicy`

- `AmazonEKS_CNI_Policy`
- `AmazonEC2ContainerRegistryReadOnly`

Without these, kubelet cannot authenticate.

---

2. API Server endpoint not reachable

From the worker node:

curl https://<cluster-endpoint>

If fails:

- Wrong route table
- No NAT Gateway
- Private cluster without correct access

---

3. Security Group blocking traffic

Nodes need:

- Outbound to port 443 (API server)
- Inbound from Control Plane SG

Control plane SG must be allowed in worker SG.

---

4. Bootstrap script failed

Check logs:

cat /var/log/cloud-init-output.log

cat /var/log/eks/bootstrap.log

Common mistake:

- Wrong cluster name
- Wrong CA file

---

5. CNI plugin failing

systemctl status kubelet

journalctl -u kubelet

If you see:

failed to start CNI bridge

→ Node won't join.

32. Sudden cost spike of ₹5 lakh in one day — fastest way to find the culprit

✅ Answer

STEP 1 — Go to Cost Explorer > Daily View

Switch to:

Group by → Service

This immediately shows the service causing the spike.

STEP 2 — Identify top causes

Usually:

- **EC2** – new large instances, AutoScaling gone wild
- **S3** – huge PUT/LIST charges
- **Data Transfer OUT** – CDN or public internet
- **NAT Gateway** – millions of requests
- **Athena / Glue** – expensive queries
- **EKS** – large node groups

STEP 3 — Check “Usage Type” breakdown

For example:

DataTransfer-Out-Bytes

$4000

→ Some service is sending massive data out of AWS.

STEP 4 — Check AWS CloudTrail

Look for:

- created EC2
- EMR

- NAT
- ALB
- S3 public access exploitation

STEP 5 — Use AWS “Cost Anomaly Detection”

Pinpoint which resource exploded.

STEP 6 — Immediate mitigation

- Disable AutoScaling
- Stop EC2/EMR/Redshift
- Disable public access for S3
- Add IAM SCPs to prevent unauthorized provisioning

---

33. VPC Flow Logs show massive traffic from unknown IP — how do you investigate?

Answer

If you see unexpected traffic, treat it as a potential **security incident**.

1. Identify the ENI (Elastic Network Interface)

VPC Flow Logs show:

eni-12345

Find the instance:

aws ec2 describe-network-interfaces --network-interface-id eni-12345

2. Identify which process is generating traffic

SSH into the instance:

sudo lsof -i

sudo netstat -tunlp

sudo ss -tup

Look for unknown:

- processes
- listening ports
- outbound connections

3. Check system for compromise

ps aux | grep crypto

crontab -l

journalctl -xe

Common signs:

- crypto miners
- backdoors
- malicious scripts

4. Check Security Groups / NACLs

Traffic may be allowed unintentionally.

5. Block IPs immediately

Via NACL:

deny all traffic from malicious IP

6. Snapshot instance for forensics

Before cleaning, create:

- EBS snapshot
- AMI

34. Route53 health check failing but endpoint is up most common misconfigurations

✅ Answer

Route53 health checks do NOT use your VPC networking — they come from AWS public IPs.

Common Misconfigurations

---

1. Health check port is wrong

Example:

- App runs on 8080
- Health check configured on 80

---

2. Health check path incorrect

If ALB expects `/health` but you configured `/`.

---

3. Server firewall blocking AWS Health Checker IPs

Your instance must allow inbound from:

route53-health-checkers

---

4. Health check type mismatch

You may have:

- HTTP check on HTTPS endpoint
- HTTPS check on invalid SSL certificate

---

5. TTL too high

DNS takes long to update.

---

6. Health check behind private IP

Route53 cannot reach private IPs.

Use:

- “VPC endpoint health check”
- Or use ALB health checks instead

---

35. IAM role suddenly giving “Access Denied” after working for months — how do you debug?

✅ Answer

1. Check if permissions were changed

Use:

IAM > Access Advisor

IAM > Policy Version History

---

2. Check for SCP (Service Control Policy) restrictions

SCPs can override IAM.

Organizations → Check:

Deny rules added?

---

3. Check if permission boundary added

Permission boundaries **restrict maximum allowed permissions**.

---

4. Check STS token expiry

Sometimes:

aws sts get-caller-identity

reveals session expired.

---

5. Check if IAM role trust relationship changed

Example:

"sts:AssumeRole" blocked

---

6. CloudTrail Investigation

Use:

eventName = "PutRolePolicy"

to see who changed the role.

---

36. Jenkins job stuck forever — how do you find the cause?

✅ Answer

1. Check Executor status

Jenkins → Manage → Nodes → Executors
If build waiting for executor:

- Increase executors
- Free dead agents

---

2. Check for hung shell commands

Open build logs.
If you see no output, run:

ps aux | grep <job_id>

strace -p <pid>

Hanging on:

- network
- file I/O
- SSH
- Docker

---

3. Check workspace locks

Sometimes Jenkins locks workspace:

lsof | grep workspace

Delete workspace:

rm -rf /var/lib/jenkins/workspace/<job>

---

4. Check dead Jenkins agent

Node offline → job stuck in queue.

---

37. Ansible playbook fails with “UNREACHABLE” on 5 out of 100 hosts.

✅ Answer

Possible Causes

---

1. SSH connectivity issues

Test:

ansible all -m ping

---

2. Wrong SSH keys / user

Try:

ssh -i <key> user@host

---

3. Python not installed on target

Ansible requires Python.

Install manually:

yum install python3 -y

---

4. Firewall blocking port 22

Check:

nmap -p22 <host>

---

5. DNS resolve issues

Try:

dig <hostname>

If failing, use IPs.

---

38. Terraform apply hangs at “Still creating…” for 40 minutes — how to debug & fix?

✅ Answer

1. Enable Debug Logs

export TF_LOG=DEBUG

export TF_LOG_PATH=tf.log

Look for:

- API throttling
- Resource dependency loops
- Timeouts

---

2. Identify the stuck resource

Terraform prints:

Still creating... [40m passed]

To see resource details:

terraform state show <resource>

---

3. Check AWS console

Common issues:

- EIP allocation stuck
- NAT gateway stuck

- RDS storage scaling
- Load balancer pending

---

4. Fix dependency loops

Example:

- SG referencing each other
- VPC depends on IGW which depends on VPC

Break loops using:

depends_on = []

---

5. If stuck due to AWS API

Apply again:

terraform apply -refresh-only

---

39. Terraform state locked by someone who left the company — safe way to release

✅ Answer

Terraform cloud or S3 backend stores a lock file.

Solution

terraform force-unlock <lock-id>

Get lock-id from:

Error: Error acquiring the state lock: LockID: "xxxxx"

Safety

Only run if:

- No one is applying
- No CI/CD pipeline running

---

40. GitLab runner stuck in “pending” state forever

✅ Answer

A pending runner means **no runner is assigned OR runner is offline**.

1. Runner is offline

Go to:

GitLab → CI/CD → Runners

Status = offline → restart runner:

gitlab-runner restart

---

2. Token mismatch

Runner registered with wrong token.

Re-register:

gitlab-runner register

---

3. Tags mismatch

Job requires:

tags: ["docker"]

Runner has:

tags: ["shell"]

→ Job never runs.

Fix by:

- Updating job tags
- OR runner tags

---

4. Runner concurrency = 0

Check file:

/etc/gitlab-runner/config.toml

Set:

concurrent = 3

---

5. GitLab shared runners disabled

Enable under:

Project → Settings → CI/CD → Runners