VINAY.T

LINUX SCENARIO BASED Q&A DAY 3

Q1. Random CPU Spikes, System Slows Down Intermittently

Scenario: The system becomes sluggish at random times. CPU usage spikes to 100% for a few minutes without a clear pattern.

🧠 Possible Root Causes:

- Scheduled cron jobs triggering resource-intensive tasks
- Background processes like file indexing (updatedb, mlocate) or antivirus scans

🛠 Solution:

- Review scheduled tasks:
  ```
  cat /etc/crontab
  ```

- Check system logs for recent activity:
  ```
  journalctl --since "10 min ago"
  ```

Tip:
Reschedule or disable non-critical cron jobs and background scans during peak hours to avoid performance degradation.

Q2. SSH Fails with "Permission denied (publickey)"

Scenario: Attempting to connect to a server via SSH using a key, but consistently receiving the error: Permission denied (publickey)

🧠 Possible Root Causes:

- Incorrect SSH key being used
- authorized_keys file missing or misconfigured
- Improper file or directory permissions on the server

🛠 Solution:

1. Verify the public key is present in the server's authorized_keys:
   ```
   cat ~/.ssh/authorized_keys
   ```

2. Set the correct permissions:
   ```
   chmod 700 ~/.ssh
   chmod 600 ~/.ssh/authorized_keys
   ```

✅ Tip:
Always check server-side logs for authentication issues:
```
journalctl -u ssh --since "10 min ago"# or check /var/log/auth.log (Debian/Ubuntu) or /var/log/secure (RHEL/CentOS)
```

Q3. SSH Server Active but Port 22 Not Reachable from Outside

Scenario: The SSH service is running (systemctl status sshd shows active), but remote users are unable to connect to the server via port 22.

🧠 Possible Root Cause:

- Port 22 is blocked by a firewall or cloud security group (e.g., AWS/GCP)

🛠️ Solution:

1. Confirm SSH is listening locally on port 22:

```
sudo ss -tuln | grep :22
```

2. Check firewall rules on the server:

```
sudo ufw status       # For UFW
sudo iptables -L      # For iptables
```

3. Allow port 22 and restart the SSH service if needed:

```
sudo ufw allow 22
sudo systemctl restart sshd
```

✅ Tip:
In cloud environments (AWS, GCP, Azure), verify that security groups or network firewall rules allow inbound traffic on port 22; OS-level changes alone won't be sufficient.

Q4. SSH Fails with "Connection Refused"

Scenario: When attempting to SSH into a server, the connection is immediately refused, no delay, no authentication attempt.

🧠 Possible Root Cause:

- The SSH daemon (sshd) is not running, or
- No service is listening on port 22

🛠️ Solution:

1. Start the SSH service:

```
sudo systemctl start sshd
```

2. Enable it to start on boot:

```
sudo systemctl enable sshd
```

3. Confirm a service is bound to port 22:

```
ss -tuln | grep :22
```

✅ Tip:
If ss -tuln shows nothing listening on port 22, SSH connections will always fail with "connection refused." Ensure SSHD is installed and correctly configured.

Q5. SSH Connection Takes ~30 Seconds Before Responding

Scenario: SSH login is unusually slow, it pauses for around 30 seconds before finally connecting.

🧠 Possible Root Cause:

- Delays caused by reverse DNS lookups
- GSSAPI authentication stalling the connection

🛠 Solution:

1. Edit the SSH daemon configuration:
   sudo vi /etc/ssh/sshd_config

   Add or modify the following lines:
   ```
   nginx
   UseDNS no
   GSSAPIAuthentication no
   ```

2. Restart the SSH service to apply changes:
   sudo systemctl restart sshd

✅ Tip:
Disabling UseDNS is a common optimization in enterprise environments to reduce SSH login delays, especially when DNS resolution is slow or misconfigured.

Q6. Can Ping Internal Hosts, But External Websites Fail

Scenario: You're able to ping internal servers, but attempts to reach external sites like google.com fail.

🧠 Likely Root Cause:

- Missing or incorrect DNS configuration, system can't resolve domain names.

🛠 Solution:

1. Check current DNS settings:
   cat /etc/resolv.conf

2. Add a valid DNS server (e.g., Google DNS):
   echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf

✅ Tip:
For a persistent fix, configure DNS through:

- /etc/systemd/resolved.conf
- /etc/netplan/ (Ubuntu)
- /etc/sysconfig/network-scripts/ (RHEL/CentOS)

Q7. Server Has No Internet Access Despite Correct IP and DNS

Scenario: IP address and DNS settings appear correct, but commands like ping 8.8.8.8 or curl to external sites fail.

🧠 Likely Root Cause:

- Missing or misconfigured default gateway, traffic has no route to the internet.

🛠 Solution:

1. Check the current routing table: ip route show

2. Add a default gateway (replace with your network's gateway IP):
   sudo ip route add default via 192.168.1.1

✅ Tip: Proper internet connectivity requires both valid DNS settings and a correctly defined default route, don't overlook either.

Q8. Cannot SSH as Root User Despite Correct Password

Scenario: Attempting to log in via SSH as root results in “access denied,” even though the password is correct. Other users can log in successfully.

🧠 Likely Root Cause:
Root SSH login is disabled by default on many Linux distributions for security reasons.

🛠️ Solution:

1. Edit the SSH daemon configuration:
   ```
   sudo vi /etc/ssh/sshd_config
   ```
   Find or add the line: `PermitRootLogin yes`

2. Restart the SSH service:
   ```
   sudo systemctl restart sshd
   ```

✅ Tip:
Enable root login only when absolutely necessary. It's best practice to use a regular user account with sudo for administrative access.

Q9. Port Open Locally But Not Reachable Externally

Scenario: An application (e.g., on port 3000) is accessible via localhost, but cannot be reached from another server.

🧠 Likely Root Cause:

- The application is bound to 127.0.0.1, making it accessible only from the local machine, not from external sources.

🛠️ Solution:

1. Check listening ports and IP bindings:
   ```
   sudo ss -tuln
   ```

2. Modify the application’s binding address:
   - From: 127.0.0.1:3000
   - To: 0.0.0.0:3000

3. Restart the application to apply changes.

✅ Tip:
In frameworks like Node.js, Flask, or Django, always bind to 0.0.0.0 if you want the app to be reachable from other machines.

Q10. Firewall Blocks Traffic After Reboot Even Though Ports Were Allowed

Scenario: Services were accessible before the reboot, but after restarting the server, previously allowed ports (e.g., 80, 443) are now blocked.

🧠 Likely Root Cause:
Firewall rules were not saved, so they didn't persist across reboots.

🛠️ Solution:
For UFW (Ubuntu):

```
sudo ufw enable
sudo ufw allow 80
```

UFW automatically persists rules once enabled.

For iptables-based systems:

1. Save current rules:
   ```
   sudo iptables-save > /etc/iptables/rules.v4
   ```
2. Ensure iptables-persistent is installed to load rules on boot.

✅ Tip:
To make rules persistent:

- Use iptables-persistent on Debian/Ubuntu
- Use the --permanent flag with firewalld on RHEL/CentOS
  ```
  sudo firewall-cmd --add-port=80/tcp --permanent
  sudo firewall-cmd --reload
  ```