🕵️ Linux Incident Investigation & Troubleshooting

📘 Scenario Overview

During the first major testing phase of **Project Phoenix**, the application experienced critical failures. With the senior DevOps engineer unavailable, the responsibility fell on me to investigate system logs, configuration files, and environment discrepancies using Linux command-line tools.

This lab simulates a **real-world incident response scenario**, requiring structured analysis, evidence collection, and configuration comparison to identify root causes and restore system stability.

🎯 Investigation Objectives

- Extract critical application errors from log files
- Identify possible system-level issues from kernel messages
- Review web server configuration for performance bottlenecks
- Compare staging vs. production configurations
- Detect missing files between server environments

🔎 Step 1: Analyze Application Logs for Errors

Task

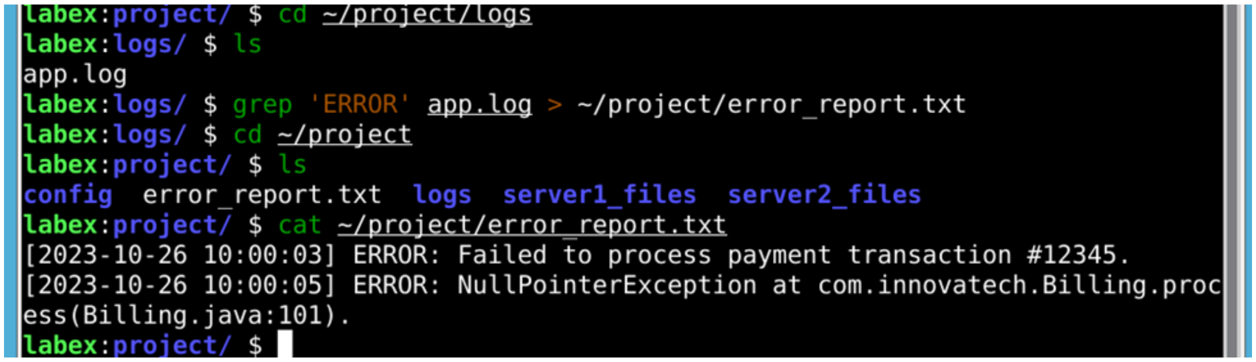
Filter the application log file to extract all error messages.

Command Used

```
grep "ERROR" ~/project/logs/app.log > ~/project/error_report.txt
```

Result

- Created `error_report.txt`
- File contains **only** log entries with the keyword `ERROR`
- Enables focused troubleshooting without noise



```
labex:project/ $ cd ~/project/logs
labex:logs/ $ ls
app.log
labex:logs/ $ grep 'ERROR' app.log > ~/project/error_report.txt
labex:logs/ $ cd ~/project
labex:project/ $ ls
config  error_report.txt  logs  server1_files  server2_files
labex:project/ $ cat ~/project/error_report.txt
[2023-10-26 10:00:03] ERROR: Failed to process payment transaction #12345.
[2023-10-26 10:00:05] ERROR: NullPointerException at com.innovatech.Billing.proc
ess(Billing.java:101).
labex:project/ $
```

🖥️ Step 2: Investigate Kernel Boot Messages

Purpose

Application errors can stem from underlying system or hardware issues.

Command Used

```
sudo dmesg | grep -iE "fail|error" > ~/project/boot_issues.txt
```

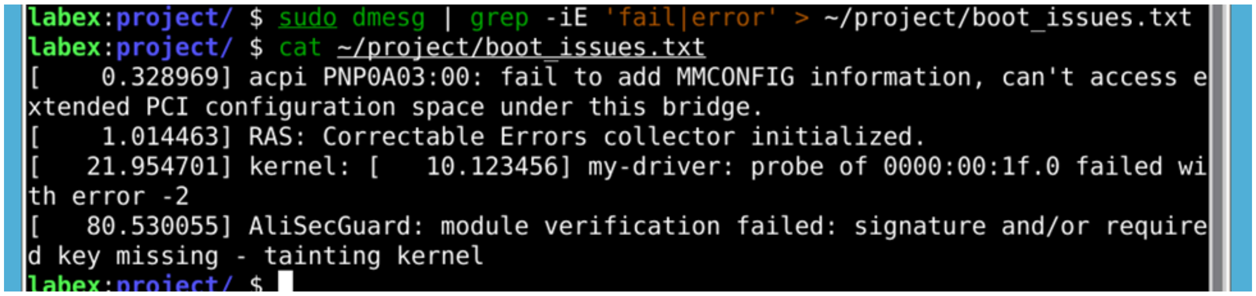
Key Techniques

- `dmesg` to read kernel ring buffer
- Case-insensitive search with `-i`
- Multiple pattern matching using `-E`

- Output redirection to preserve evidence

Result

- Created `boot_issues.txt`
- Captured kernel-level warnings and errors for review



```
labex:project/ $ sudo dmesg | grep -iE 'fail|error' > ~/project/boot_issues.txt
labex:project/ $ cat ~/project/boot_issues.txt
[    0.328969] acpi PNP0A03:00: fail to add MMCONFIG information, can't access e
xtended PCI configuration space under this bridge.
[    1.014463] RAS: Correctable Errors collector initialized.
[   21.954701] kernel: [   10.123456] my-driver: probe of 0000:00:1f.0 failed wi
th error -2
[   80.530055] AliSecGuard: module verification failed: signature and/or require
d key missing - tainting kernel
labex:project/ $
```

🌐 Step 3: Examine Web Server Configuration

Objective

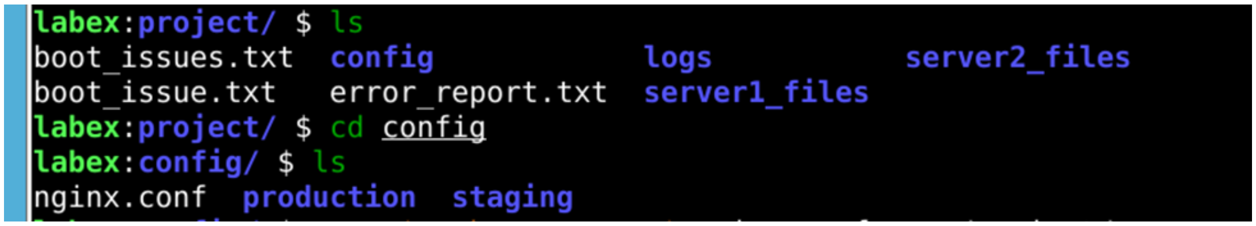
Identify possible misconfiguration in the Nginx setup that could impact performance.

Command Used

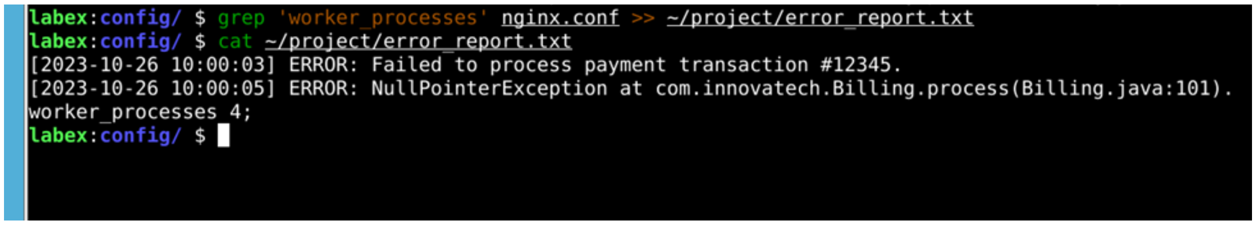
```
grep "worker_processes" ~/project/config/nginx.conf >>
~/project/error_report.txt
```

Outcome

- Appended (not overwritten) configuration data
- Centralized application errors and relevant configuration in one report
- Enabled correlation between errors and server settings



```
labex:project/ $ ls
boot_issues.txt  config            logs           server2_files
boot_issue.txt   error_report.txt  server1_files
labex:project/ $ cd config
labex:config/ $ ls
nginx.conf  production  staging
```

- 

```
labex:config/ $ grep 'worker_processes' nginx.conf >> ~/project/error_report.txt
labex:config/ $ cat ~/project/error_report.txt
[2023-10-26 10:00:03] ERROR: Failed to process payment transaction #12345.
[2023-10-26 10:00:05] ERROR: NullPointerException at com.innovatech.Billing.process(Billing.java:101).
worker_processes 4;
labex:config/ $
```

🧪 Step 4: Compare Staging vs Production Configuration Files

Purpose

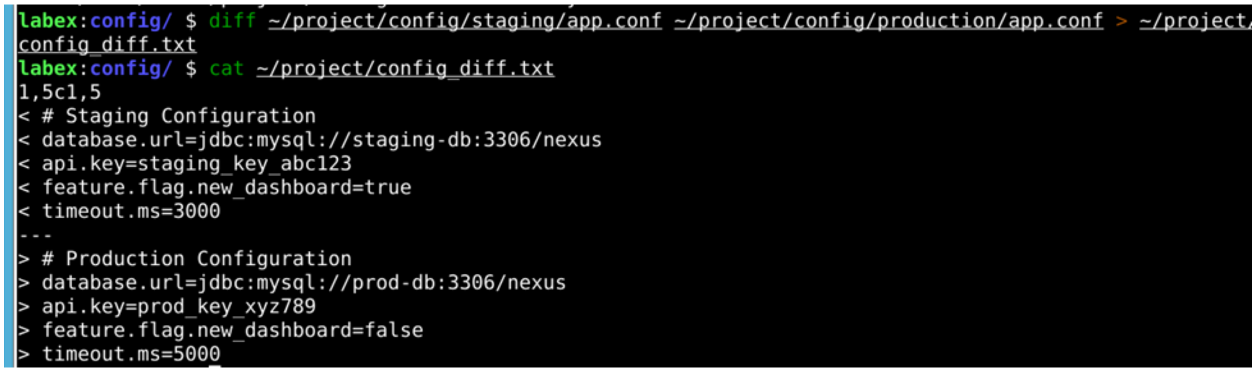
Configuration drift between environments is a common cause of production failures.

Command Used

```
diff ~/project/config/staging/app.conf
~/project/config/production/app.conf > ~/project/config_diff.txt
```

Result

- Created `config_diff.txt`
- Identified configuration mismatches impacting production behavior



```
labex:config/ $ diff ~/project/config/staging/app.conf ~/project/config/production/app.conf > ~/project/
config_diff.txt
labex:config/ $ cat ~/project/config_diff.txt
1,5c1,5
< # Staging Configuration
< database.url=jdbc:mysql://staging-db:3306/nexus
< api.key=staging_key_abc123
< feature.flag.new_dashboard=true
< timeout.ms=3000
---
> # Production Configuration
> database.url=jdbc:mysql://prod-db:3306/nexus
> api.key=prod_key_xyz789
> feature.flag.new_dashboard=false
> timeout.ms=5000
```

🗂️ Step 5: Verify Directory Consistency Between Servers

Scenario

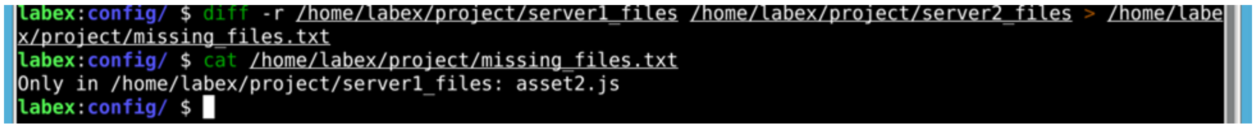
Production server may be missing files present in staging due to a failed deployment.

Command Used

```
diff /home/labex/project/server1_files
/home/labex/project/server2_files >
/home/labex/project/missing_files.txt
```

Outcome

- Generated `missing_files.txt`
- Revealed files present in staging but missing in production
- Confirmed incomplete deployment as a contributing factor



```
labex:config/ $ diff -r /home/labex/project/server1_files /home/labex/project/server2_files > /home/labe
x/project/missing_files.txt
labex:config/ $ cat /home/labex/project/missing_files.txt
Only in /home/labex/project/server1_files: asset2.js
labex:config/ $ 
```

🧠 Skills Demonstrated

- Incident response methodology
- Log analysis with `grep`
- Kernel diagnostics using `dmesg`
- Configuration auditing
- Environment comparison with `diff`
- Command pipelines and output redirection
- Evidence preservation and reporting