DevOps Shack

# ERRORS & TROUBLESHOOTING

With Step-by-step solutions

Prepared by:

Devops Shack

2025

www.devopsshack.com

[Click here for DevSecOps & Cloud DevOps Course]

# DevOps Shack

# Prometheus and Grafana Errors & Solutions

## Table of Contents

20. Error: "Grafana authentication via OAuth not working"

**3. Alerting & Notifications**

21. Error: "Duplicate metrics collector registration attempted" in Prometheus
22. Error: "Failed to send alert notification" in Grafana
23. Error: "No Alertmanager configured" in Prometheus
24. Error: "Dashboard fails to load variables" in Grafana
25. Error: "tsdb: compaction failed" in Prometheus
26. Error: "Cannot delete a Prometheus target" in Grafana
27. Error: "Datasource health check failed" in Grafana
28. Error: "Prometheus retention time is too short"
29. Error: "Grafana panels not updating automatically"
30. Error: "429 Too Many Requests" in Prometheus or Grafana

**4. Performance & Query Issues**

31. Error: "Prometheus query taking too long"
32. Error: "Prometheus web UI not loading"
33. Error: "Error fetching Prometheus metrics in Grafana"
34. Error: "Prometheus scrape loop stalled"
35. Error: "Grafana: Panel visualization not displaying data"
36. Error: "Prometheus: WAL corruption detected"
37. Error: "Error expanding environment variables in Grafana"
38. Error: "AlertManager alerts not firing in Prometheus"
39. Error: "Prometheus query range exceeded maximum points limit"
40. Error: "Grafana email alerts not working"

**5. Configuration & Setup Issues**

41. Error: "Prometheus retention settings ignored"
42. Error: "Grafana login loop issue"

43. Error: "Prometheus scrape interval not applied"
44. Error: "Grafana data source deleted automatically"
45. Error: "Prometheus scrape exceeds evaluation timeout"
46. Error: "Grafana panels not refreshing automatically"
47. Error: "Prometheus service fails to start with 'address already in use'"
48. Error: "Grafana provisioning not working"
49. Error: "Prometheus remote read/write errors"
50. Error: "Grafana dashboard import fails"

# Introduction

### Why Prometheus and Grafana Are Important

Prometheus and Grafana are among the most widely used open-source tools for monitoring and observability. **Prometheus** is a powerful time-series database that collects, stores, and queries metrics from various sources. **Grafana** complements it by providing a **visualization layer**, enabling users to create dashboards, alerts, and reports based on Prometheus data.

Together, these tools help DevOps teams, SREs, and system administrators **gain insights into application performance, detect issues, and optimize system health.** However, as powerful as they are, both tools can encounter various errors that may disrupt monitoring and alerting systems.

### Why It's Important to Solve Errors Quickly

Monitoring solutions are **mission-critical** for detecting system failures, performance degradation, or security threats in real time. If **Prometheus or Grafana stops working**, teams may face:

- **Delayed issue detection** leading to system downtime.
- **Loss of valuable monitoring data** that impacts decision-making.
- **Inaccurate alerts** causing false positives or missed incidents.

By **identifying and resolving errors quickly**, we can ensure system stability, reduce downtime, and maintain operational efficiency.

### Types of Errors That Commonly Occur

Errors in Prometheus and Grafana can arise due to multiple reasons, including **misconfigurations, resource limitations, query inefficiencies, or integration failures.** These issues generally fall into the following categories:

**1. Installation & Setup Issues**

- Prometheus or Grafana fails to start.
- Configuration files are missing or invalid.
- Incorrect permissions preventing execution.

**2. Data Collection & Scraping Errors**

- Prometheus cannot reach a target.
- Scrape jobs fail due to timeouts or incorrect configurations.
- Metrics not appearing in Grafana.

**3. Query & Performance Issues**

- PromQL queries taking too long or returning empty results.
- High memory usage in Prometheus leading to slow queries.
- Dashboards loading slowly due to inefficient queries.

**4. Alerting & Notification Failures**

- Alerts not triggering as expected.
- Grafana email or webhook notifications failing.
- AlertManager not receiving alerts from Prometheus.

**5. Authentication & Access Problems**

- Invalid username/password errors in Grafana.
- OAuth or LDAP authentication failing.
- Users unable to access specific dashboards or data sources.

**6. Storage & Retention Problems**

- Prometheus disk usage growing unexpectedly.
- WAL (Write-Ahead Log) corruption issues.
- Data retention policies not applying correctly.

**How to Solve Prometheus & Grafana Errors Effectively**

To **troubleshoot and fix errors efficiently**, follow these best practices:

**1. Check Logs for Clues**

Both Prometheus and Grafana provide logs that can **reveal the root cause of issues**.

- Check **Prometheus logs**:

journalctl -u prometheus --no-pager | tail -n 20

- Check **Grafana logs**:

journalctl -u grafana-server --no-pager | tail -n 20

**2. Verify Configuration Files**

Misconfigurations are a major cause of errors. Always check and validate configuration files before applying changes.

- Validate **Prometheus config**:

promtool check config /etc/prometheus/prometheus.yml

- Verify **Grafana datasources** in **Configuration → Data Sources**.

**3. Test Queries & Endpoints**

If a dashboard or alert is not working as expected, test individual components:

- Run Prometheus queries in **PromQL Expression Browser**.
- Test data sources in Grafana by clicking **Save & Test** under **Configuration → Data Sources**.
- Use **cURL** or **Postman** to test API and alert webhooks.

**4. Monitor System Resources**

If Prometheus or Grafana is **slow or crashing**, check:

- **CPU & Memory usage** using htop or top.
- **Disk space availability** using df -h.
- **Network connectivity** using ping or telnet.

**5. Restart Services & Apply Fixes**

Sometimes, simply restarting the services can resolve transient issues:

systemctl restart prometheus

systemctl restart grafana-server

**Key Things to Keep an Eye On**

To **prevent errors from happening**, regularly monitor:

✅ **Prometheus targets** – Ensure all scrape jobs are healthy.

✅ **Query performance** – Avoid overly complex queries that slow down dashboards.

✅ **System resources** – Make sure Prometheus has enough CPU, memory, and storage.

✅ **Alerting configurations** – Test alerts regularly to avoid false positives/negatives.

✅ **Authentication & security settings** – Prevent unauthorized access to Grafana dashboards.

✅ **Retention policies** – Ensure Prometheus is not storing unnecessary old data.

# ERRORS AND SOLUTIONS

**1. Error: "context deadline exceeded" in Prometheus**

**Cause:** Prometheus is unable to scrape metrics from a target due to timeout.

**Solution:**

1. Check the target endpoint manually using curl <target-url> to see if it responds.
2. Increase the scrape timeout in prometheus.yml:

```
scrape_configs:
 - job_name: 'my-job'
   scrape_interval: 30s
   scrape_timeout: 25s
```

3. Ensure the target service is running and accessible.
4. If using a firewall, allow incoming connections on the required port.
5. Restart Prometheus and check logs for further details.

**2. Error: "Error loading config file: yaml: unmarshal errors" in Prometheus**

**Cause:** There is a syntax error in prometheus.yml.

**Solution:**

1. Validate the YAML file using an online YAML validator or yamllint.
2. Ensure proper indentation and format.
3. Check logs (prometheus --config.file=prometheus.yml) for specific line numbers.
4. Fix any syntax errors and restart Prometheus.

**3. Error: "server returned HTTP status 500 Internal Server Error" in Grafana**

**Cause:** There is a misconfiguration in Grafana or its database.

**Solution:**

1. Check Grafana logs (/var/log/grafana/grafana.log).
2. If using SQLite, ensure the database file (grafana.db) has the right permissions.
3. If using MySQL/PostgreSQL, verify the database connection string in grafana.ini.
4. Restart Grafana:

```
systemctl restart grafana-server
```

5. If the issue persists, clear the Grafana cache:

```
rm -rf /var/lib/grafana/*
```

**4. Error: "Prometheus target down (instance not reachable)" in Grafana**

**Cause:** Grafana is unable to fetch data from Prometheus.

**Solution:**

1. Check Prometheus status using http://<prometheus-host>:9090/targets.
2. Verify Prometheus is running:

```
systemctl status prometheus
```

3. Check if the Prometheus URL in Grafana is correct (Data Sources → Prometheus → URL).
4. Restart Prometheus and Grafana if needed.

**5. Error: "Bad Gateway (502)" when accessing Grafana**

**Cause:** Grafana is behind a reverse proxy (e.g., Nginx) that is misconfigured.

**Solution:**

1. Check if Grafana is running using:

```
systemctl status grafana-server
```

2. Verify Nginx configuration (/etc/nginx/sites-available/grafana):

```
location /grafana/ {
```

```
    proxy_pass http://localhost:3000/;
}
```

3. Restart Nginx:

```
systemctl restart nginx
```

4. Ensure firewall rules allow traffic on port **3000**.

**6. Error: "Datasource is not working" in Grafana**

**Cause:** The Prometheus data source is not properly configured in Grafana.

**Solution:**

1. Go to **Grafana → Configuration → Data Sources**.
2. Verify that the Prometheus URL is correct (http://localhost:9090).
3. Click **Save & Test** to check the connection.
4. If failing, restart both Prometheus and Grafana.

**7. Error: "No data points found" in Grafana dashboards**

**Cause:** The Prometheus query is incorrect or there is no data for the selected time range.

**Solution:**

1. Check the Prometheus data source by running the query in Prometheus UI (http://localhost:9090).
2. Ensure the correct time range is selected in Grafana.
3. Check if Prometheus is scraping metrics (http://localhost:9090/targets).
4. Refresh the dashboard or restart Grafana if needed.

**8. Error: "PromQL error: parse error at char X"**

**Cause:** The PromQL query has a syntax issue.

**Solution:**

1. Verify the query syntax in the **Prometheus Query Editor** (http://localhost:9090).
2. Use correct operators (=~, !=, =, etc.).
3. Ensure the metric name is correct by listing all available metrics (http://localhost:9090/api/v1/label/__name__/values).

**9. Error: "Failed to start Prometheus: address already in use"**

**Cause:** Another process is already using the same port (**9090** by default).

**Solution:**

1. Check which process is using the port:

```
sudo netstat -tulnp | grep 9090
```

2. Kill the conflicting process:

```
sudo kill -9 <PID>
```

3. Change the Prometheus port in prometheus.yml if needed:

```
web.listen-address: ":9091"
```

4. Restart Prometheus:

```
systemctl restart prometheus
```

**10. Error: "Error opening storage: permission denied" in Prometheus**

**Cause:** Prometheus does not have write permissions for its storage directory.

**Solution:**

1. Check Prometheus user:

```
ps aux | grep prometheus
```

2. Change ownership of the storage directory:

```
sudo chown -R prometheus:prometheus /var/lib/prometheus
```

3. Restart Prometheus:

```
systemctl restart prometheus
```

Here are the next **10 common errors** in **Prometheus and Grafana** along with their step-by-step solutions:

### 11. Error: "tsdb: out of order sample" in Prometheus

**Cause:** Prometheus is receiving metric data with timestamps that are out of order.

**Solution:**

1. Check if the exporter or application sending metrics has time drift.
2. Synchronize system time using NTP:

```
sudo timedatectl set-ntp on
```

3. If using a custom exporter, ensure timestamps are increasing correctly.
4. Restart Prometheus:

```
systemctl restart prometheus
```

### 12. Error: "Error executing query: exceeded maximum resolution" in Grafana

**Cause:** The query is fetching too much data, exceeding the resolution limit.

**Solution:**

1. Reduce the query time range in Grafana.
2. Use **rate()** instead of **increase()** for better performance.
3. Optimize the query with the **step** parameter:

```
rate(http_requests_total[5m])
```

4. Increase the maximum resolution in Grafana's data source settings.

### 13. Error: "Connection refused" when Prometheus scrapes a target

**Cause:** The target is not reachable or the port is incorrect.

**Solution:**

1. Check if the target is running:

```
systemctl status <service>
```

2. Ensure Prometheus is configured with the correct target address in prometheus.yml.
3. Test the connection manually:

```
curl http://<target-host>:<port>/metrics
```

4. If using Docker, ensure the target container is reachable within the network.

**14. Error: "Error creating query" in Grafana**

**Cause:** The query syntax is incorrect.

**Solution:**

1. Verify the query in **Prometheus Query Editor** (http://localhost:9090).
2. Ensure metric names and labels are correct.
3. Use the **Explore** tab in Grafana to test different queries.
4. If using sum() or avg(), wrap the expression correctly:

```
sum(rate(http_requests_total[5m]))
```

**15. Error: "Error writing to WAL" in Prometheus**

**Cause:** Prometheus cannot write to its Write-Ahead Log (WAL) storage.

**Solution:**

1. Check disk space:

```
df -h
```

2. Change ownership of the data directory:

```
sudo chown -R prometheus:prometheus /var/lib/prometheus
```

3. Restart Prometheus:

```
systemctl restart prometheus
```

**16. Error: "Grafana dashboard not saving changes"**

**Cause:** Grafana does not have the correct permissions or the database is locked.

**Solution:**

1. Check Grafana logs for errors:

```
cat /var/log/grafana/grafana.log
```

2. Restart Grafana:

```
systemctl restart grafana-server
```

3. If using SQLite, remove any database locks:

```
rm /var/lib/grafana/grafana.db-shm
rm /var/lib/grafana/grafana.db-wal
```

**17. Error: "Prometheus remote write queue full"**

**Cause:** Prometheus is unable to send data to a remote storage due to overload.

**Solution:**

1. Increase queue_config in prometheus.yml:

```
remote_write:
 - url: "http://remote-storage-url"
   queue_config:
    max_samples_per_send: 10000
    capacity: 25000
```

2. Reduce the number of scraped metrics or increase hardware resources.
3. Restart Prometheus and monitor queue size.

**18. Error: "No rule evaluations were performed" in Prometheus**

**Cause:** Prometheus rules are misconfigured or failing to evaluate.

**Solution:**

1. Check syntax errors in rules.yml:

```
prometheus --config.file=prometheus.yml --log.level=debug
```

2. Ensure rules are correctly defined:

```
groups:
 - name: alert_rules
   rules:
    - alert: HighCPUUsage
      expr: cpu_usage > 80
      for: 5m
```

3. Restart Prometheus and check logs for errors.

**19. Error: "Error loading dashboard" in Grafana**

**Cause:** The JSON dashboard file has errors or is missing data.

**Solution:**

1. Check JSON syntax using an online JSON validator.
2. Verify the dashboard exists in Grafana's /var/lib/grafana/dashboards/ directory.
3. If importing a dashboard, ensure all dependencies (datasources, plugins) are installed.
4. Restart Grafana:

```
systemctl restart grafana-server
```

**20. Error: "OOMKilled" in Prometheus or Grafana (Out of Memory)**

**Cause:** The service is consuming too much memory, causing the OS to kill the process.

**Solution:**

1. Increase system memory or allocate more resources if using a container.
2. Reduce Prometheus retention period in prometheus.yml:

storage.tsdb.retention.time: "15d"

3. Enable Prometheus compression:

storage.tsdb.min-block-duration: "2h"

storage.tsdb.max-block-duration: "2h"

4. Optimize Grafana queries by reducing the number of data points fetched.

**21. Error: "Duplicate metrics collector registration attempted" in Prometheus**

**Cause:** The same metric is being registered multiple times in a custom exporter.

**Solution:**

1. Check the exporter code and ensure metrics are only registered once.
2. Restart the exporter and Prometheus.
3. If using a third-party exporter, update it to the latest version.

**22. Error: "Failed to send alert notification" in Grafana**

**Cause:** Incorrect alert notification configuration in Grafana.

**Solution:**

1. Check Grafana logs for detailed error messages:

tail -f /var/log/grafana/grafana.log

2. Verify the notification channel settings under **Alerting → Notification Channels**.
3. If using SMTP, ensure the correct **host, port, and authentication** settings in grafana.ini:

[smtp]

enabled = true

```
host = smtp.example.com:587
user = myemail@example.com
password = mypassword
```

4. Restart Grafana and test the alert again.

**23. Error: "No Alertmanager configured" in Prometheus**

**Cause:** Prometheus is not connected to an Alertmanager instance.

**Solution:**

1. Add Alertmanager configuration in prometheus.yml:

```
alerting:
 alertmanagers:
  - static_configs:
     - targets:
      - "localhost:9093"
```

2. Ensure Alertmanager is running:

```
systemctl status alertmanager
```

3. Restart Prometheus:

```
systemctl restart prometheus
```

**24. Error: "Dashboard fails to load variables" in Grafana**

**Cause:** The variables in the dashboard are not correctly defined.

**Solution:**

1. Go to **Dashboard → Settings → Variables** and check for any errors.
2. Test the variable query in **Explore** to ensure it returns values.
3. If using Prometheus, ensure the label exists:

```
label_values(node_cpu_seconds_total, instance)
```

4. Save and refresh the dashboard.

### 25. Error: "tsdb: compaction failed" in Prometheus

**Cause:** Prometheus cannot compact its TSDB storage due to corruption or lack of disk space.

**Solution:**

1. Check available disk space:

```
df -h
```

2. Run Prometheus in repair mode:

```
prometheus --storage.tsdb.path=/var/lib/prometheus --storage.tsdb.retention.time=30d --storage.tsdb.min-block-duration=2h --storage.tsdb.max-block-duration=2h --storage.tsdb.allow-overlapping-blocks=false --storage.tsdb.no-lockfile
```

3. If corruption persists, delete old data:

```
rm -rf /var/lib/prometheus/*
```

4. Restart Prometheus.

### 26. Error: "Cannot delete a Prometheus target" in Grafana

**Cause:** Grafana only reads from Prometheus and does not control its configuration.

**Solution:**

1. Remove the target from Prometheus prometheus.yml:

```
scrape_configs:
  - job_name: 'old_target'
    static_configs:
      - targets: ['localhost:9091']
```

2. Restart Prometheus:

```
systemctl restart prometheus
```

3. Refresh the Grafana dashboard.

**27. Error: "Datasource health check failed" in Grafana**

**Cause:** Grafana cannot reach Prometheus or the configured database.

**Solution:**

1. Go to **Configuration → Data Sources** and test the connection.
2. If using Prometheus, ensure it is running:

```
systemctl status prometheus
```

3. Check firewall rules to allow Grafana to connect to Prometheus on port **9090**.
4. Restart both Prometheus and Grafana.

**28. Error: "Prometheus retention time is too short"**

**Cause:** The configured data retention period is too low.

**Solution:**

1. Increase retention time in prometheus.yml:

```
storage.tsdb.retention.time: "30d"
```

2. Ensure there is enough disk space for extended retention.
3. Restart Prometheus.

**29. Error: "Grafana panels not updating automatically"**

**Cause:** The dashboard auto-refresh setting is disabled or the data source is slow.

**Solution:**

1. Enable auto-refresh in Grafana:
    - Go to **Dashboard → Settings → Time Range**
    - Set auto-refresh to **5s, 10s, 30s**, etc.

2. Check the Prometheus scrape interval:

```
scrape_interval: 15s
```

3. Restart Prometheus and refresh the dashboard.

**30. Error: "429 Too Many Requests" in Prometheus or Grafana**

**Cause:** The system is making too many API requests in a short period.

**Solution:**

1. Reduce the scrape interval in prometheus.yml:

```
- job_name: 'my-job'
  scrape_interval: 30s
```

2. If using an external API, ensure rate limits are not exceeded.
3. Optimize Grafana queries to fetch data efficiently.

**31. Error: "Prometheus query taking too long"**

**Cause:** The query is too complex, fetching too much data, or Prometheus is underpowered.

**Solution:**

1. Optimize the PromQL query:
    - Use rate() instead of increase().
    - Add filters to reduce the dataset:

```
rate(http_requests_total{job="my-service"}[5m])
```

2. Reduce the time range in Grafana (e.g., last **1h** instead of **7d**).
3. Increase Prometheus memory and CPU if using a container.
4. Restart Prometheus and monitor its performance.

**32. Error: "Prometheus web UI not loading"**

**Cause:** Prometheus is not running, or another process is using port **9090**.

**Solution:**

1. Check if Prometheus is running:

```
systemctl status prometheus
```

2. Ensure nothing else is using port 9090:

```
sudo netstat -tulnp | grep 9090
```

3. Restart Prometheus:

```
systemctl restart prometheus
```

4. Check logs for errors:

```
journalctl -u prometheus --no-pager | tail -n 20
```

**33. Error: "Error fetching Prometheus metrics in Grafana"**

**Cause:** Grafana cannot connect to Prometheus.

**Solution:**

1. Go to **Grafana → Configuration → Data Sources → Prometheus**.
2. Verify the Prometheus URL (e.g., http://localhost:9090).
3. Click **Save & Test** to check the connection.
4. Restart both Prometheus and Grafana.

**34. Error: "Prometheus scrape loop stalled"**

**Cause:** Prometheus is taking too long to scrape targets.

**Solution:**

1. Reduce the number of scraped targets in prometheus.yml.
2. Increase scrape_timeout:

```
scrape_configs:
 - job_name: 'my-app'
   scrape_interval: 30s
```

```
scrape_timeout: 25s
```

3. Monitor Prometheus CPU usage (htop or top).
4. Restart Prometheus and check logs for stalls.

**35. Error: "Grafana: Panel visualization not displaying data"**

**Cause:** The query is incorrect, or the selected time range is empty.

**Solution:**

1. Verify the query in **Explore** mode.
2. Check the **time range** in the top-right corner (e.g., change from "Last 7 days" to "Last 1 hour").
3. Make sure Prometheus is up and running.
4. If using a transformation, check if it's removing data unintentionally.

**36. Error: "Prometheus: WAL corruption detected"**

**Cause:** The Write-Ahead Log (WAL) is corrupted due to a crash or disk issue.

**Solution:**

1. Stop Prometheus:

```
systemctl stop prometheus
```

2. Move the WAL directory to a backup location:

```
mv /var/lib/prometheus/wal /var/lib/prometheus/wal-backup
```

3. Restart Prometheus:

```
systemctl start prometheus
```

4. If the issue persists, clear Prometheus data (last resort):

```
rm -rf /var/lib/prometheus/*
```

**37. Error: "Error expanding environment variables in Grafana"**

**Cause:** The Grafana configuration file does not support direct use of environment variables.

**Solution:**

1. Define variables before running Grafana:

```
export GF_SECURITY_ADMIN_PASSWORD="mypassword"
```

2. Start Grafana manually:

```
grafana-server --config /etc/grafana/grafana.ini
```

3. If using Docker, pass environment variables:

```
docker run -e GF_SECURITY_ADMIN_PASSWORD=mypassword grafana/grafana
```

**38. Error: "AlertManager alerts not firing in Prometheus"**

**Cause:** Alerts are not meeting the defined conditions or AlertManager is misconfigured.

**Solution:**

1. Check if Prometheus is evaluating alerts:

```
curl http://localhost:9090/api/v1/alerts
```

2. Verify alert rules in rules.yml:

```
groups:
 - name: alert_rules
   rules:
    - alert: HighMemoryUsage
      expr: node_memory_Active_bytes > 1000000000
      for: 5m
```

3. Restart Prometheus and AlertManager.

**39. Error: "Prometheus query range exceeded maximum points limit"**

**Cause:** The query is trying to return too many data points.

**Solution:**

1. Reduce the query time range in Grafana.
2. Use a higher step value in the PromQL query:

```
rate(http_requests_total[5m])
```

3. Increase the query limit in prometheus.yml (not recommended for large setups).

**40. Error: "Grafana email alerts not working"**

**Cause:** SMTP settings are incorrect or Grafana lacks email permissions.

**Solution:**

1. Verify SMTP settings in grafana.ini:

```
[smtp]
enabled = true
host = smtp.example.com:587
user = myemail@example.com
password = mypassword
```

2. Restart Grafana:

```
systemctl restart grafana-server
```

3. Test the email configuration:
    - Go to **Alerting → Notification Channels → Send Test Alert**.

**41. Error: "Prometheus retention settings ignored"**

**Cause:** Incorrect retention configuration in prometheus.yml.

**Solution:**

1. Ensure storage.tsdb.retention.time is set correctly:

```
storage.tsdb.retention.time: "30d"
```

2. Restart Prometheus to apply changes:

```
systemctl restart prometheus
```

3. Check if Prometheus is running with the correct retention settings:

```
prometheus --version
```

4. If using older versions, use --storage.tsdb.retention=30d instead.

**42. Error: "Grafana login loop issue"**

**Cause:** Incorrect cookie_samesite settings or authentication misconfiguration.

**Solution:**

1. Edit grafana.ini and update:

```
[security]
cookie_samesite = disabled
```

2. If using OAuth, ensure callback URLs are correctly set.
3. Restart Grafana:

```
systemctl restart grafana-server
```

4. Clear browser cookies and try logging in again.

**43. Error: "Prometheus scrape interval not applied"**

**Cause:** The scrape_interval is overridden elsewhere in the config.

**Solution:**

1. Check for conflicting settings in prometheus.yml:

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'my-job'
```

```
scrape_interval: 30s
```

2. Ensure there's no override in command-line flags.
3. Restart Prometheus and verify logs.

**44. Error: "Grafana data source deleted automatically"**

**Cause:** The Grafana configuration is set to restore defaults on restart.

**Solution:**

1. Check for provisioning settings under /etc/grafana/provisioning/datasources/.
2. Edit datasources.yml and disable deleteDatasources:

```
deleteDatasources: false
```

3. Restart Grafana:

```
systemctl restart grafana-server
```

**45. Error: "Prometheus scrape exceeds evaluation timeout"**

**Cause:** The scrape process takes too long, exceeding the timeout limit.

**Solution:**

1. Increase evaluation_interval in prometheus.yml:

```
global:
  evaluation_interval: 30s
```

2. Optimize slow queries or reduce the number of targets.
3. Restart Prometheus and monitor logs.

**46. Error: "Grafana panels not refreshing automatically"**

**Cause:** Auto-refresh settings are disabled or overridden.

**Solution:**

1. In Grafana, enable auto-refresh under **Dashboard Settings → Time Range → Auto Refresh**.
2. Ensure the panel queries use relative time ($__interval).
3. Restart Grafana and reload the dashboard.

**47. Error: "Prometheus service fails to start with 'address already in use'"**

**Cause:** Another process is using port **9090**.

**Solution:**

1. Find the process using the port:

```
sudo netstat -tulnp | grep 9090
```

2. Kill the process (if needed):

```
sudo kill -9 <PID>
```

3. Restart Prometheus:

```
systemctl restart prometheus
```

**48. Error: "Grafana provisioning not working"**

**Cause:** Incorrect provisioning settings in Grafana.

**Solution:**

1. Verify /etc/grafana/provisioning/dashboards/ contains valid JSON/YAML files.
2. Check the ownership of the files:

```
chown -R grafana:grafana /etc/grafana/provisioning/
```

3. Restart Grafana to apply changes:

```
systemctl restart grafana-server
```

**49. Error: "Prometheus remote read/write errors"**

**Cause:** Prometheus cannot communicate with remote storage.

**Solution:**

1. Check the remote storage URL in prometheus.yml:

```
remote_write:
  - url: "http://remote-storage-url"
```

2. Test connectivity manually:

```
curl -I http://remote-storage-url
```

3. Increase queue buffer size:

```
remote_write:
  - queue_config:
      max_samples_per_send: 10000
      capacity: 25000
```

4. Restart Prometheus.

**50. Error: "Grafana dashboard import fails"**

**Cause:** The JSON file is invalid or missing data.

**Solution:**

1. Validate the JSON file using an online JSON validator.
2. Ensure all required datasources exist before importing.
3. Try importing via API:

```
curl -X POST -H "Content-Type: application/json" -d @dashboard.json
http://localhost:3000/api/dashboards/db
```

4. Restart Grafana and reattempt the import.

## Conclusion

Prometheus and Grafana are essential tools for monitoring and observability, but like any system, they can encounter various errors that impact performance, alerting, and data visualization. Understanding common issues—ranging from installation failures and query inefficiencies to authentication problems and alerting failures—helps in maintaining a stable and efficient monitoring setup.

By following structured troubleshooting methods, such as checking logs, validating configurations, testing queries, and monitoring system resources, most issues can be resolved quickly. Regularly reviewing Prometheus scrape targets, optimizing queries, and ensuring sufficient system resources are key to preventing downtime and data loss.

Additionally, keeping Grafana dashboards well-configured and alerting mechanisms properly tested ensures that teams receive accurate and timely notifications about system health. Security settings, retention policies, and performance monitoring should also be reviewed periodically to avoid unexpected failures.

This guide provides step-by-step solutions to **50 common Prometheus and Grafana errors**, equipping teams with the knowledge to diagnose and fix issues efficiently. By implementing best practices and proactive monitoring, you can ensure a **reliable, scalable, and effective observability system**, minimizing disruptions and improving operational efficiency.