**Prometheus & Grafana DevOps Lab - Assessment Questions**

**Instructions**

Answer the following questions based on your hands-on experience with the Prometheus and Grafana lab. Provide detailed explanations that demonstrate your understanding of the concepts. **Use of AI tools is strictly prohibited. Answers detected as AI-generated will receive a score of 0.**

---

**Section 1: Architecture and Setup Understanding**

**Question 1: Container Orchestration Analysis**

Examine the Docker Compose configuration used in this lab. Explain why the Node Exporter container requires mounting host directories (/proc, /sys, /) and what would happen if these mounts were removed or configured incorrectly. How does this design decision reflect the principle of containerized monitoring?

**Question 2: Network Security Implications**

The lab creates a custom Docker network named "prometheus-grafana". Analyze the security implications of this setup versus using the default Docker network. What potential vulnerabilities could arise from the current port exposure configuration (9090, 9100, 3000), and how would you modify the setup for a production environment?

**Question 3: Data Persistence Strategy**

Compare the volume mounting strategies used for Prometheus data versus Grafana data in the Docker Compose file. Explain why different approaches might be needed for different components and what would happen to your dashboards and historical metrics if you removed these volume configurations.

---

**Section 2: Metrics and Query Understanding**

**Question 4: PromQL Logic Breakdown**

The tutorial uses this query for calculating uptime:

node_time_seconds - node_boot_time_seconds

Explain step-by-step what each metric represents, why subtraction gives us uptime, and what potential issues could arise with this calculation method. Propose an alternative approach and justify when you might use it instead.

**Question 5: Memory Metrics Deep Dive**

The lab uses node_memory_MemTotal_bytes - node_memory_MemAvailable_bytes to calculate memory usage. Research and explain why this approach is preferred over using node_memory_MemFree_bytes. What's the fundamental difference between "free" and "available" memory in Linux systems, and how does this impact monitoring accuracy?

**Question 6: Filesystem Query Analysis**

Analyze this filesystem usage query:

```
1 - (node_filesystem_avail_bytes{mountpoint="/"} /
node_filesystem_size_bytes{mountpoint="/"})
```

Break down the mathematical logic, explain why the result needs to be subtracted from 1, and discuss what could go wrong if you monitoring multiple mount points with this approach. How would you modify this query to exclude temporary filesystems?

---

**Section 3: Visualization and Dashboard Design**

**Question 7: Visualization Type Justification**

The tutorial uses three different visualization types: Stat, Time Series, and Gauge. For each visualization created in the lab, justify why that specific type was chosen over alternatives. What criteria should guide visualization selection, and when might your choices be suboptimal?

**Question 8: Threshold Configuration Strategy**

Explain the reasoning behind the 80% threshold setting for disk usage in the gauge chart. Research industry standards and propose a more sophisticated alerting strategy that considers different types of systems (database servers, web servers, etc.). How would you implement multi-level thresholds that provide actionable insights?

**Question 9: Dashboard Variable Implementation**

The tutorial introduces dashboard variables using the $job variable. Explain how this variable system works internally in Grafana, what happens when you have multiple values for a variable, and design a scenario where poorly implemented variables could break your dashboard. How would you test variable robustness?

---

**Section 4: Production and Scalability Considerations**

**Question 10: Resource Planning and Scaling**

Based on your lab experience, calculate the approximate resource requirements (CPU, memory, storage) for monitoring 100 servers using this setup. Consider metric ingestion rates, retention periods, and dashboard query load. What bottlenecks would you expect to encounter first, and how would you address them?

**Question 11: High Availability Design**

The current lab setup is single-node. Design a high-availability architecture for this monitoring stack that can handle component failures. Explain your approach to data consistency, load balancing, and disaster recovery. What trade-offs would you make between complexity and reliability?

**Question 12: Security Hardening Analysis**

Identify at least five security vulnerabilities in the lab setup and propose specific remediation strategies. Consider authentication, authorization, network security, and data protection. How would you implement secrets management and secure communication between components?

---

**Section 5: Troubleshooting and Operations**

**Question 13: Debugging Methodology**

Describe a systematic approach to troubleshooting when Prometheus shows a target as "DOWN" in the targets page. Walk through the diagnostic steps you would take, including command-line tools, log analysis, and configuration verification. What are the most common causes and their solutions?

**Question 14: Performance Optimization**

After running the lab, analyze the query performance of your dashboards. Identify which queries might be expensive and explain why. Propose optimization strategies for both PromQL queries and Grafana dashboard design. How would you monitor the monitoring system itself?

**Question 15: Capacity Planning Scenario**

You notice that Prometheus is consuming increasing amounts of disk space over time. Analyze the factors that contribute to storage growth, calculate retention policies based on business requirements, and design a data lifecycle management strategy. How would you balance historical data availability with resource constraints?

---

**Submission Requirements**

1. **GitHub Repository**: Create a public repository containing all code, configurations, and documentation
2. **README.md**: Include setup instructions, architecture diagram, and answers to all questions
3. **Screenshots**: Provide clear screenshots of:
    - Docker containers running (docker compose ps)
    - Prometheus targets page showing "UP" status
    - Each Grafana dashboard panel created
    - Grafana data source configuration
    - Dashboard variable configuration
4. **Code Files**: Include all configuration files used (docker-compose.yml, prometheus.yml, etc.)
5. **Documentation**: Each answer should demonstrate practical understanding

**Grading Criteria**

- **Technical Accuracy** (40%): Correctness of technical explanations and understanding
- **Depth of Analysis** (30%): Demonstration of critical thinking and deeper understanding
- **Practical Application** (20%): Ability to apply concepts to real-world scenarios

- **Documentation Quality** (10%): Clarity, organization, and completeness of submission

**Note**: Answers that appear to be AI-generated will automatically receive a score of 0. Focus on demonstrating your personal understanding and insights gained from hands-on experience.