DevOps/SRE Log Parsing Practice Questions and Solutions

Question 1: POST Request Success Counter (Your Example)

Problem: Calculate the total number of successful POST requests in the log file events.log

Log Format: `[15/Sep/2023:13:25:34 -0400] "POST /upload/file HTTP/1.1" 201 48299`

Solution:

```python
def solution():
    import re
    import os

    log_file_path = "/var/logs/events.log"
    successful_post_count = 0

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                # Parse the log line using regex
                match = re.match(r'\[([^\]]+)\] "(\w+) ([^"]+)" (\d+) (\d+)', line.strip())
                if match:
                    method = match.group(2)
                    status_code = int(match.group(4))

                    # Check if it's a POST request with successful status code (200-299)
                    if method == "POST" and 200 <= status_code <= 299:
                        successful_post_count += 1

    except FileNotFoundError:
        return 0

    return successful_post_count

if __name__ == '__main__':
    print(solution())
```

Question 2: Error Rate Calculator

Problem: Calculate the error rate percentage for all HTTP requests in access.log over the last hour.

Log Format: `127.0.0.1 - - [01/Jan/2024:12:00:00 +0000] "GET /api/users HTTP/1.1" 404 512`

Task: Return the error rate as a float (4xx and 5xx status codes are errors)

Solution:

```
python
```

```
def solution():
    import re
    from datetime import datetime, timedelta

    log_file_path = "/var/logs/access.log"
    current_time = datetime.now()
    one_hour_ago = current_time - timedelta(hours=1)

    total_requests = 0
    error_requests = 0

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.match(r'(\S+) - - \[([^\]]+)\] "(\w+) ([^"]+)" (\d+) (\d+)', line.strip())
                if match:
                    timestamp_str = match.group(2)
                    status_code = int(match.group(5))

                    # Parse timestamp
                    log_time = datetime.strptime(timestamp_str, "%d/%b/%Y:%H:%M:%S %z")
                    log_time = log_time.replace(tzinfo=None)  # Remove timezone for comparison

                    if log_time >= one_hour_ago:
                        total_requests += 1
                        if status_code >= 400:
                            error_requests += 1

    except (FileNotFoundError, ValueError):
        return 0.0

    return (error_requests / total_requests * 100) if total_requests > 0 else 0.0

if __name__ == '__main__':
    print(solution())
```

Question 3: Top IP Addresses by Request Count

Problem: Find the top 5 IP addresses by request count from nginx.log

Log Format: `192.168.1.100 - - [01/Jan/2024:12:00:00 +0000] "GET /index.html HTTP/1.1" 200 1024`

Task: Return a list of tuples [(ip, count), ...] sorted by count descending

Solution:

```python
def solution():
    import re
    from collections import Counter

    log_file_path = "/var/logs/nginx.log"
    ip_counter = Counter()

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.match(r'(\d+\.\d+\.\d+\.\d+)', line.strip())
                if match:
                    ip_address = match.group(1)
                    ip_counter[ip_address] += 1

    except FileNotFoundError:
        return []

    return ip_counter.most_common(5)

if __name__ == '__main__':
    print(solution())
```

Question 4: Average Response Time Calculator

Problem: Calculate the average response time for successful requests (200-299) from application.log

Log Format: `2024-01-01 12:00:00 INFO [RequestHandler] GET /api/data - Status: 200, Response Time: 145ms`

Task: Return average response time in milliseconds as float

Solution:

```python
```

```
def solution():
    import re

    log_file_path = "/var/logs/application.log"
    response_times = []

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.search(r'Status: (\d+), Response Time: (\d+)ms', line)
                if match:
                    status_code = int(match.group(1))
                    response_time = int(match.group(2))

                    if 200 <= status_code <= 299:
                        response_times.append(response_time)

    except FileNotFoundError:
        return 0.0

    return sum(response_times) / len(response_times) if response_times else 0.0

if __name__ == '__main__':
    print(solution())
```

Question 5: Database Connection Pool Monitor

Problem: Count the number of database connection timeouts in db.log for the current day

Log Format: `2024-01-01 15:30:45 ERROR [ConnectionPool] Connection timeout after 30s - Pool: users_db`

Task: Return count of timeout errors as integer

Solution:

```
python
```

```
def solution():
    import re
    from datetime import datetime

    log_file_path = "/var/logs/db.log"
    current_date = datetime.now().strftime("%Y-%m-%d")
    timeout_count = 0

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                if current_date in line and "Connection timeout" in line and "ERROR" in line:
                    timeout_count += 1

    except FileNotFoundError:
        return 0

    return timeout_count

if __name__ == '__main__':
    print(solution())
```

Question 6: Memory Usage Alert Parser

Problem: Parse system.log and find the highest memory usage percentage recorded today

Log Format: `Jan 01 12:00:00 server01 kernel: Memory usage: 85.4% (7.2GB/8.0GB)`

Task: Return highest memory percentage as float

Solution:

```
python
```

```
def solution():
  import re
  from datetime import datetime

  log_file_path = "/var/logs/system.log"
  current_date = datetime.now().strftime("%b %d")
  max_memory_usage = 0.0

  try:
    with open(log_file_path, 'r') as file:
      for line in file:
        if current_date in line:
          match = re.search(r'Memory usage: (\d+\.?\d*)%', line)
          if match:
            memory_usage = float(match.group(1))
            max_memory_usage = max(max_memory_usage, memory_usage)

  except FileNotFoundError:
    return 0.0

  return max_memory_usage

if __name__ == '__main__':
  print(solution())
```

Question 7: Failed Login Attempts by User

Problem: Count failed SSH login attempts per user from auth.log in the last 24 hours

Log Format: `Jan 01 12:00:00 server sshd[1234]: Failed password for admin from 192.168.1.100 port 22 ssh2`

Task: Return dictionary {username: attempt_count}

Solution:

```
python
```

```
def solution():
    import re
    from datetime import datetime, timedelta
    from collections import defaultdict

    log_file_path = "/var/logs/auth.log"
    current_year = datetime.now().year
    yesterday = datetime.now() - timedelta(days=1)
    failed_attempts = defaultdict(int)

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.search(r'(\w{3} \d{1,2} \d{2}:\d{2}:\d{2}).*Failed password for (\w+)', line)
                if match:
                    timestamp_str = match.group(1)
                    username = match.group(2)

                    # Parse timestamp (assuming current year)
                    log_time = datetime.strptime(f"{current_year} {timestamp_str}", "%Y %b %d %H:%M:%S")

                    if log_time >= yesterday:
                        failed_attempts[username] += 1

    except (FileNotFoundError, ValueError):
        return {}

    return dict(failed_attempts)

if __name__ == '__main__':
    print(solution())
```

Question 8: API Endpoint Performance Analysis

Problem: Find the slowest API endpoint from api.log (response time > 1000ms)

Log Format: `2024-01-01T12:00:00Z GET /api/v1/users/search?q=test 200 1250ms`

Task: Return the endpoint with highest average response time as string

Solution:

```
python
```

```
def solution():
    import re
    from collections import defaultdict

    log_file_path = "/var/logs/api.log"
    endpoint_times = defaultdict(list)

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.search(r'(\w+) (/[^\s]+) (\d+) (\d+)ms', line)
                if match:
                    method = match.group(1)
                    endpoint = match.group(2)
                    response_time = int(match.group(4))

                    if response_time > 1000:
                        endpoint_key = f"{method} {endpoint}"
                        endpoint_times[endpoint_key].append(response_time)

    except FileNotFoundError:
        return ""

    if not endpoint_times:
        return ""

    # Calculate average response time for each endpoint
    avg_times = {}
    for endpoint, times in endpoint_times.items():
        avg_times[endpoint] = sum(times) / len(times)

    # Return endpoint with highest average
    return max(avg_times, key=avg_times.get) if avg_times else ""

if __name__ == '__main__':
    print(solution())
```

Question 9: Disk Space Alert Counter

Problem: Count critical disk space alerts (>90% usage) from monitoring.log today

Log Format: `2024-01-01 14:30:00 CRITICAL [DiskMonitor] /dev/sda1 usage: 95.2% (19.0GB/20.0GB)`

Task: Return count of critical alerts as integer

Solution:

python

```python
def solution():
    import re
    from datetime import datetime

    log_file_path = "/var/logs/monitoring.log"
    current_date = datetime.now().strftime("%Y-%m-%d")
    critical_count = 0

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                if current_date in line and "CRITICAL" in line and "DiskMonitor" in line:
                    match = re.search(r'usage: (\d+\.?\d*)%', line)
                    if match:
                        usage_percent = float(match.group(1))
                        if usage_percent > 90:
                            critical_count += 1

    except FileNotFoundError:
        return 0

    return critical_count

if __name__ == '__main__':
    print(solution())
```

Question 10: Container Restart Analysis

Problem: Find containers that restarted more than 3 times in docker.log today

Log Format: `2024-01-01T12:00:00.000Z container restart event container_id=abc123 container_name=web-service`

Task: Return list of container names

Solution:

python

```
def solution():
    import re
    from datetime import datetime
    from collections import Counter

    log_file_path = "/var/logs/docker.log"
    current_date = datetime.now().strftime("%Y-%m-%d")
    restart_counter = Counter()

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                if current_date in line and "container restart event" in line:
                    match = re.search(r'container_name=(\S+)', line)
                    if match:
                        container_name = match.group(1)
                        restart_counter[container_name] += 1

    except FileNotFoundError:
        return []

    return [name for name, count in restart_counter.items() if count > 3]

if __name__ == '__main__':
    print(solution())
```

Question 11: Load Balancer Health Check Failures

Problem: Count health check failures per backend server from lb.log in the last hour

Log Format: `2024-01-01 12:00:00 [HealthCheck] backend-server-01:8080 FAILED - Response timeout`

Task: Return dictionary {server: failure_count}

Solution:

```
python
```

```python
def solution():
    import re
    from datetime import datetime, timedelta
    from collections import defaultdict

    log_file_path = "/var/logs/lb.log"
    current_time = datetime.now()
    one_hour_ago = current_time - timedelta(hours=1)
    failures = defaultdict(int)

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.search(r'(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}).*\[HealthCheck\] (\S+) FAILED', line)
                if match:
                    timestamp_str = match.group(1)
                    server = match.group(2)

                    log_time = datetime.strptime(timestamp_str, "%Y-%m-%d %H:%M:%S")

                    if log_time >= one_hour_ago:
                        failures[server] += 1

    except (FileNotFoundError, ValueError):
        return {}

    return dict(failures)

if __name__ == '__main__':
    print(solution())
```

Question 12: Security Event Severity Counter

Problem: Count security events by severity level from security.log today

Log Format: `2024-01-01 12:00:00 SEVERITY:HIGH [SecurityAlert] Suspicious login attempt from IP 10.0.0.1`

Task: Return dictionary {severity: count}

Solution:

```
python
```

```python
def solution():
    import re
    from datetime import datetime
    from collections import defaultdict

    log_file_path = "/var/logs/security.log"
    current_date = datetime.now().strftime("%Y-%m-%d")
    severity_counter = defaultdict(int)

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                if current_date in line:
                    match = re.search(r'SEVERITY:(\w+)', line)
                    if match:
                        severity = match.group(1)
                        severity_counter[severity] += 1

    except FileNotFoundError:
        return {}

    return dict(severity_counter)

if __name__ == '__main__':
    print(solution())
```

Question 13: Cache Hit Rate Calculator

Problem: Calculate cache hit rate percentage from cache.log for the last 30 minutes

Log Format: `2024-01-01 12:00:00 [Cache] Key:user_123 Result:HIT Size:2KB TTL:300s` `2024-01-01 12:00:01 [Cache] Key:user_456 Result:MISS Size:0KB TTL:0s`

Task: Return hit rate as float percentage

Solution:

```
python
```

```
def solution():
    import re
    from datetime import datetime, timedelta

    log_file_path = "/var/logs/cache.log"
    current_time = datetime.now()
    thirty_minutes_ago = current_time - timedelta(minutes=30)

    total_requests = 0
    cache_hits = 0

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.search(r'(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}).*Result:(HIT|MISS)', line)
                if match:
                    timestamp_str = match.group(1)
                    result = match.group(2)

                    log_time = datetime.strptime(timestamp_str, "%Y-%m-%d %H:%M:%S")

                    if log_time >= thirty_minutes_ago:
                        total_requests += 1
                        if result == "HIT":
                            cache_hits += 1

    except (FileNotFoundError, ValueError):
        return 0.0

    return (cache_hits / total_requests * 100) if total_requests > 0 else 0.0

if __name__ == '__main__':
    print(solution())
```

Question 14: Microservice Communication Errors

Problem: Find services with the most inter-service communication errors from microservices.log

Log Format: `2024-01-01T12:00:00Z [service-a] ERROR calling service-b: Connection refused`

Task: Return top 3 services with most outbound errors as list of tuples [(service, error_count), ...]

Solution:

python

```python
def solution():
    import re
    from collections import Counter

    log_file_path = "/var/logs/microservices.log"
    error_counter = Counter()

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.search(r'\[(\w+-\w+)\] ERROR calling (\w+-\w+):', line)
                if match:
                    calling_service = match.group(1)
                    error_counter[calling_service] += 1

    except FileNotFoundError:
        return []

    return error_counter.most_common(3)

if __name__ == '__main__':
    print(solution())
```

Question 15: Deployment Success Rate

Problem: Calculate deployment success rate from deployment.log for the current month

Log Format: `2024-01-01 12:00:00 [Deploy] service=user-api version=v1.2.3 status=SUCCESS duration=120s` `2024-01-01 12:05:00 [Deploy] service=order-api version=v2.1.0 status=FAILED duration=45s`

Task: Return success rate as float percentage

Solution:

python

```
def solution():
    import re
    from datetime import datetime

    log_file_path = "/var/logs/deployment.log"
    current_month = datetime.now().strftime("%Y-%m")

    total_deployments = 0
    successful_deployments = 0

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                if current_month in line and "[Deploy]" in line:
                    match = re.search(r'status=(SUCCESS|FAILED)', line)
                    if match:
                        status = match.group(1)
                        total_deployments += 1
                        if status == "SUCCESS":
                            successful_deployments += 1

    except FileNotFoundError:
        return 0.0

    return (successful_deployments / total_deployments * 100) if total_deployments > 0 else 0.0

if __name__ == '__main__':
    print(solution())
```

Question 16: Network Latency Analysis

Problem: Find the 95th percentile network latency from network.log today

Log Format: `2024-01-01 12:00:00 [NetworkMonitor] ping target=8.8.8.8 latency=25.4ms status=OK`

Task: Return 95th percentile latency as float

Solution:

```
python
```

```
def solution():
    import re
    from datetime import datetime

    log_file_path = "/var/logs/network.log"
    current_date = datetime.now().strftime("%Y-%m-%d")
    latencies = []

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                if current_date in line:
                    match = re.search(r'latency=(\d+\.?\d*)ms', line)
                    if match:
                        latency = float(match.group(1))
                        latencies.append(latency)

    except FileNotFoundError:
        return 0.0

    if not latencies:
        return 0.0

    latencies.sort()
    index = int(0.95 * len(latencies))
    return latencies[min(index, len(latencies) - 1)]

if __name__ == '__main__':
    print(solution())
```

Question 17: Thread Pool Exhaustion Detector

Problem: Count thread pool exhaustion events from app.log in the last 6 hours

Log Format: `2024-01-01 12:00:00 ERROR [ThreadPool] Pool exhausted: 200/200 threads active, rejecting request`

Task: Return count of exhaustion events as integer

Solution:

```
python
```

```
def solution():
    import re
    from datetime import datetime, timedelta

    log_file_path = "/var/logs/app.log"
    current_time = datetime.now()
    six_hours_ago = current_time - timedelta(hours=6)
    exhaustion_count = 0

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.search(r'(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}).*Pool exhausted', line)
                if match:
                    timestamp_str = match.group(1)
                    log_time = datetime.strptime(timestamp_str, "%Y-%m-%d %H:%M:%S")

                    if log_time >= six_hours_ago:
                        exhaustion_count += 1

    except (FileNotFoundError, ValueError):
        return 0

    return exhaustion_count

if __name__ == '__main__':
    print(solution())
```

Question 18: SSL Certificate Expiry Warnings

Problem: Extract SSL certificates expiring within 30 days from ssl.log

Log Format: `2024-01-01 12:00:00 WARNING [SSLMonitor] Certificate for api.example.com expires in 15 days`

Task: Return list of domain names with expiring certificates

Solution:

```
python
```

```python
def solution():
    import re

    log_file_path = "/var/logs/ssl.log"
    expiring_domains = []

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.search(r'Certificate for (\S+) expires in (\d+) days', line)
                if match:
                    domain = match.group(1)
                    days_until_expiry = int(match.group(2))

                    if days_until_expiry <= 30:
                        expiring_domains.append(domain)

    except FileNotFoundError:
        return []

    return list(set(expiring_domains))  # Remove duplicates

if __name__ == '__main__':
    print(solution())
```

Question 19: Backup Job Status Monitor

Problem: Find failed backup jobs from backup.log in the last 24 hours

Log Format: `2024-01-01 02:00:00 [BackupJob] database=users_db status=FAILED error="Disk space insufficient"`

Task: Return list of failed database names

Solution:

```
python
```

```python
def solution():
    import re
    from datetime import datetime, timedelta

    log_file_path = "/var/logs/backup.log"
    current_time = datetime.now()
    twenty_four_hours_ago = current_time - timedelta(hours=24)
    failed_databases = []

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                match = re.search(r'(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}).*database=(\w+).*status=FAILED', line)
                if match:
                    timestamp_str = match.group(1)
                    database = match.group(2)

                    log_time = datetime.strptime(timestamp_str, "%Y-%m-%d %H:%M:%S")

                    if log_time >= twenty_four_hours_ago:
                        failed_databases.append(database)

    except (FileNotFoundError, ValueError):
        return []

    return list(set(failed_databases))  # Remove duplicates

if __name__ == '__main__':
    print(solution())
```

Question 20: Request Rate Spike Detector

Problem: Detect request rate spikes (>1000 requests/minute) from traffic.log

Log Format: `2024-01-01 12:00:00 [Traffic] requests_per_minute=1250 total_requests=75000`

Task: Return list of timestamps when spikes occurred

Solution:

```
python
```

```
def solution():
    import re
    from datetime import datetime

    log_file_path = "/var/logs/traffic.log"
    current_date = datetime.now().strftime("%Y-%m-%d")
    spike_timestamps = []

    try:
        with open(log_file_path, 'r') as file:
            for line in file:
                if current_date in line:
                    match = re.search(r'(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}).*requests_per_minute=(\d+)', line)
                    if match:
                        timestamp = match.group(1)
                        requests_per_minute = int(match.group(2))

                        if requests_per_minute > 1000:
                            spike_timestamps.append(timestamp)

    except FileNotFoundError:
        return []

    return spike_timestamps

if __name__ == '__main__':
    print(solution())
```

Bonus Questions

Question 21: GC Pause Analysis

Calculate average garbage collection pause time from jvm.log

Question 22: Circuit Breaker Events

Count circuit breaker trips from service.log

Question 23: Rate Limiting Violations

Find top IPs hitting rate limits from rate_limiter.log

Question 24: Kubernetes Pod Restarts

Analyze pod restart patterns from k8s.log

Question 25: Message Queue Processing Delays

Calculate message processing delay percentiles from queue.log

Each of these questions follows similar patterns and can be solved using regex parsing, datetime handling, and statistical analysis techniques commonly used in DevOps/SRE log analysis.