Day-9-Linux commands

To extract the **date and time** of all requests where `"Database connection lost"` appears in the logs, use the following `grep` command:

Command:

```
grep -oP '^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}' logfile.txt
```

Explanation:

1. **`grep -oP`** → Enables **Perl-compatible regex (`-P`)** and **extracts only matching parts (`-o`)**.
2. **Regex Breakdown:**
   - `^\d{4}-\d{2}-\d{2}` → Matches **date (`YYYY-MM-DD`)** at the **start of the line** (`^`).
   - `\s` → Matches a **space** between the date and time.
   - `\d{2}:\d{2}:\d{2}` → Matches **time (`HH:MM:SS`)**.

Filters only lines containing `"Database connection lost"`:

```
grep "Database connection lost" logfile.txt
```

3.
   - This ensures we **search only for relevant log entries**.

Final Command (Combining Both)

```
grep "Database connection lost" logfile.txt | grep -oP
'^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}'
```

This **first finds the relevant log entries**, then extracts only the **date and time**.

Example Log File (logfile.txt)

pgsql

```
2024-02-01 07:45:31 DEBUG [app-server-1] Request body: {"user_id":
1245}
2024-02-01 08:10:14 ERROR [app-server-1] Database connection lost
2024-02-01 08:30:22 ERROR [app-server-2] Database connection lost
```

Output:

yaml

```
2024-02-01 08:10:14
2024-02-01 08:30:22
```

This extracts only the timestamps of failed database connections!

To extract the **date and time** of all log lines containing `"WARN"`, you can use the following `grep` command:

Command:

```
grep "WARN" logfile.txt | grep -oP '^\d{4}-\d{2}-\d{2}
\d{2}:\d{2}:\d{2}'
```

Explanation:

1. **`grep "WARN" logfile.txt`**
   - Finds all log lines containing `"WARN"` (case-sensitive).
2. **`grep -oP '^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}'`**
   - `-o` → Extracts only the matching part (date and time).
   - `-P` → Enables **Perl-compatible regex**.
   - `^\d{4}-\d{2}-\d{2}` → Matches **date (`YYYY-MM-DD`)** at the **start of the line**.
   - `\s` → Matches the space between date and time.
   - `\d{2}:\d{2}:\d{2}` → Matches **time (`HH:MM:SS`)**.

Example Log File (`logfile.txt`):

pgsql

```
2024-02-01 07:45:31 DEBUG [app-server-1] Request received
2024-02-01 08:10:14 ERROR [app-server-1] Database connection lost
2024-02-01 08:30:22 WARN [app-server-2] High memory usage detected
2024-02-01 09:15:45 WARN [app-server-3] Disk space low
```

Output:

yaml

```
2024-02-01 08:30:22
2024-02-01 09:15:45
```

Alternative Approach (Single `grep` Command)

If your `grep` supports **Perl regex (`-P`)**, you can **combine both filters** into a single command:

```
grep -oP '^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}(?=.*WARN)' logfile.txt
```

- The **`(?=.*WARN)`** ensures that `"WARN"` is present in the log line **without capturing it**.
- This makes the command **faster and more efficient**.

This efficiently extracts timestamps of `"WARN"` log entries!

To extract logs where CPU usage is greater than 70% and Memory usage is greater than 80%, I use the following `awk` command:

Command:

```
awk -F'[:,%]+' '$2 ~ /CPU Usage/ && $3 > 70 && $5 ~ /Memory Usage/ && $6 > 80' logfile.txt
```

Explanation:

1. `-F '[:,%]+'` → Sets the field separator to `:` or `%` (splitting on both).
2. `$2 ~ /CPU Usage/` → Checks if the second field contains `"CPU Usage"`.
3. `$3 > 70` → Ensures CPU usage is greater than `70%`.
4. `$5 ~ /Memory Usage/` → Checks if the fifth field contains `"Memory Usage"`.
5. `$6 > 80` → Ensures Memory usage is greater than `80%`.

Example Log File (`logfile.txt`):

pgsql

```
2024-02-01 10:15:30 INFO CPU Usage: 72%, Memory Usage: 85%

2024-02-01 10:16:30 INFO CPU Usage: 68%, Memory Usage: 82%

2024-02-01 10:17:30 INFO CPU Usage: 75%, Memory Usage: 79%

2024-02-01 10:18:30 INFO CPU Usage: 80%, Memory Usage: 90%
```

Output:

pgsql

```
2024-02-01 10:15:30 INFO CPU Usage: 72%, Memory Usage: 85%

2024-02-01 10:18:30 INFO CPU Usage: 80%, Memory Usage: 90%
```

✅ This extracts only the lines where CPU usage > 70% and Memory usage > 80%.

Alternative Approach Using `grep` and `awk`

If `awk` seems complex, I can filter with `grep` first, then use `awk`:

```
grep "CPU Usage" logfile.txt | grep "Memory Usage" | awk -F'[:,%]+'
'$3 > 70 && $6 > 80'
```

This efficiently filters high CPU and Memory usage logs!

How I Connect Ollama in VS Code

To use **Ollama** in VS Code, I follow these steps:

1️⃣ Install Ollama on My System

First, I ensure **Ollama** is installed on my machine.

For Linux/macOS:

```
curl -fsSL https://ollama.com/install.sh | sh
```

For Windows:

- I download and install **Ollama** from https://ollama.com.

2️⃣ Verify Ollama Installation

After installation, I check if **Ollama is running** by executing:

```
ollama list
```

✅ If Ollama is installed, it should list available models.

3️⃣ Run Ollama Server

Ollama runs as a **local API server**, so I start it using:

```
ollama serve
```

✅ This ensures Ollama is ready to accept requests.

4️⃣ Install VS Code Extensions

To integrate **Ollama with VS Code**, I install:
1️⃣ **REST Client** extension (to make API calls).
2️⃣ **Python** extension (if using Python to interact with Ollama).

5️⃣ Test Ollama in VS Code Terminal

I open **VS Code Terminal** and run:

```
ollama run mistral
```

✅ This runs the **Mistral model** locally. I can replace `"mistral"` with any installed model.

6️⃣ Connect Ollama in a Python Script (Optional)

I create a new Python file (`ollama_test.py`) and add:

```
import requests

response = requests.post("http://localhost:11434/api/generate", json={
    "model": "mistral",
    "prompt": "Hello, Ollama!",
})
print(response.json()["response"])
```

✅ I run it using:

```
python ollama_test.py
```

7️⃣ Use Ollama with a VS Code Notebook

To use Ollama in a Jupyter Notebook inside VS Code:

I install **Jupyter**:
```
pip install jupyter
```

1.

I create a **new `.ipynb` file** and run:
```
!curl -X POST http://localhost:11434/api/generate -d '{"model": "mistral", "prompt": "Explain AI"}'
```

2.

Now, I Can Use Ollama in VS Code for AI Models!

By following these steps, I successfully integrate Ollama with VS Code and can run **local LLM models efficiently.**