Linux Process Monitor - Usage Guide

🔍 Overview

The Linux Process Monitor script provides comprehensive monitoring of critical system processes with AppDynamics integration, using the exact same JSON configuration file as the Windows PowerShell version.

📋 Prerequisites

- **bash** (version 4.0 or later)
- **jq** (for JSON configuration parsing)
- Standard Linux utilities: `ps`, `pgrep`, `awk`, `grep`

Installing jq

```bash
# Ubuntu/Debian
sudo apt-get install jq

# CentOS/RHEL/Fedora
sudo yum install jq
# or
sudo dnf install jq

# Alpine Linux
apk add jq

# macOS
brew install jq
```

🚀 Quick Start

Make Script Executable

```bash
chmod +x process_monitor.sh
```

Basic Usage (AppDynamics Format)

```bash
# Run with default settings
./process_monitor.sh

# Output example:
# name=Custom Metrics|ProcessMon|splunkd,value=1
# name=Custom Metrics|ProcessMon|httpd,value=1
```

Using Configuration File (Same JSON as Windows!)

```bash
# Use the exact same JSON config file as PowerShell version
./process_monitor.sh -c processes.json
```

Different Output Formats

```bash
# JSON output for integration
./process_monitor.sh -f JSON

# CSV output for reporting
./process_monitor.sh -f CSV

# Console output for interactive use
./process_monitor.sh -f Console
```

📋 Parameters

Parameter	Short	Description
`--config FILE`	`-c`	Path to JSON configuration file
`--format FORMAT`	`-f`	Output format: AppDynamics, JSON, CSV, Console
`--log FILE`	`-l`	Path to log file for debugging
`--details`	`-d`	Include CPU and memory metrics
`--quiet`	`-q`	Suppress console output except metrics
`--help`	`-h`	Show help message
`--version`	`-v`	Show version information

💡 Usage Examples

1. AppDynamics Integration (Default)

```bash
# Standard AppDynamics metrics output
./process_monitor.sh

# With detailed metrics (CPU, Memory)
./process_monitor.sh -d

# With custom configuration
./process_monitor.sh -c production-processes.json
```

2. Logging and Debugging

```bash
# Enable detailed logging
./process_monitor.sh -l /var/log/process-monitor.log

# Quiet mode (only output metrics)
./process_monitor.sh -q -l /var/log/process-monitor.log
```

3. Reporting and Analysis

```bash
# Generate JSON report
./process_monitor.sh -f JSON -d > process-report.json

# Generate CSV for analysis
./process_monitor.sh -f CSV -d > process-report.csv

# Interactive console view
./process_monitor.sh -f Console -d
```

4. Scheduled Monitoring with Cron

```bash
```

```
# Add to crontab for monitoring every minute
# crontab -e
# */1 * * * * /path/to/process_monitor.sh -c /etc/appdynamics/processes.json -q -l /var/log/process-monitor-$(date +\%Y\
```

🔧 Configuration File

Uses the exact same JSON format as the Windows PowerShell version!

Create a `processes.json` file:

```json
{
  "ProcessNames": [
    "splunkd",
    "httpd",
    "nginx",
    "java",
    "mysqld",
    "postgres",
    "redis-server",
    "docker",
    "kubelet"
  ],
  "MetricPrefix": "Custom Metrics|ProcessMon",
  "TimeoutSeconds": 30
}
```

Environment-Specific Configurations

Development Environment (`dev-processes.json`):

```json
```

```
{
    "ProcessNames": [
        "java",
        "httpd",
        "nginx",
        "node",
        "python"
    ],
    "MetricPrefix": "Custom Metrics|ProcessMon|Dev"
}
```

Production Environment (`prod-processes.json`):

```json
{
    "ProcessNames": [
        "httpd",
        "nginx",
        "java",
        "mysqld",
        "postgres",
        "redis-server",
        "splunkd",
        "docker",
        "kubelet",
        "prometheus",
        "grafana-server"
    ],
    "MetricPrefix": "Custom Metrics|ProcessMon|Prod"
}
```

📊 Output Format Examples

AppDynamics Format

```
name=Custom Metrics|ProcessMon|splunkd,value=1
name=Custom Metrics|ProcessMon|httpd,value=1
name=Custom Metrics|ProcessMon|java,value=1
```

JSON Format

```json
{
  "Timestamp": "2025-01-15 10:30:00",
  "Processes": [
    {
      "Name": "splunkd",
      "Id": 1234,
      "CPU": 15.25,
      "WorkingSet": 512,
      "VirtualMemory": 1024,
      "Status": "Running"
    }
  ],
  "Summary": {
    "Total": 3,
    "MonitoredProcesses": 25
  }
}
```

Console Format

```
=== Process Monitor Results ===
Timestamp: 2025-01-15 10:30:00
Processes Found: 3 / 25 monitored

Name            PID    CPU%    Memory(MB)  Virtual(MB)  Status
----            ---    ----    ---------   ----------   ------
splunkd         1234   15.25   512         1024         Running
httpd           5678   8.50    256         768          Running
java            9012   25.75   1024        2048         Running
```

🔄 Integration with AppDynamics Machine Agent

1. Install Script in Machine Agent Directory

```bash
```

```
# Copy script to machine agent monitors directory
sudo cp process_monitor.sh /opt/appdynamics/machine-agent/monitors/ProcessMonitor/
sudo chmod +x /opt/appdynamics/machine-agent/monitors/ProcessMonitor/process_monitor.sh

# Copy configuration file
sudo cp processes.json /opt/appdynamics/machine-agent/monitors/ProcessMonitor/
```

2. Create monitor.xml

```xml
<monitor>
    <name>ProcessMonitor</name>
    <type>managed</type>
    <description>Custom Process Monitor for Linux</description>
    <monitor-configuration>
        <execution-style>periodic</execution-style>
        <execution-frequency-in-seconds>60</execution-frequency-in-seconds>
        <properties>
            <property name="command" value="./process_monitor.sh"/>
            <property name="command-arguments" value="-c processes.json -q"/>
        </properties>
    </monitor-configuration>
</monitor>
```

3. Restart Machine Agent

```bash
sudo systemctl restart appdynamics-machine-agent
# or
sudo service appdynamics-machine-agent restart
```

🔍 Troubleshooting

Common Issues

1. **Permission denied:**

   ```bash
   chmod +x process_monitor.sh
   ```

2. **jq not found:**

```bash
# Install jq using your package manager
sudo apt-get install jq  # Ubuntu/Debian
sudo yum install jq      # CentOS/RHEL
```

3. **Configuration file not found:**

```bash
# Verify file exists and is readable
ls -la processes.json
```

4. **No processes found:**

```bash
# Enable detailed logging and console output
./process_monitor.sh -l debug.log -f Console
```

5. **Process names not matching:**

```bash
# Check actual process names
ps aux | grep -i processname

# or use pgrep to test
pgrep -f "processname"
```

📝 Best Practices

1. **Test process name matching** before deploying to production
2. **Use absolute paths** in cron jobs and systemd services
3. **Rotate log files** to prevent disk space issues
4. **Monitor script performance** with large process lists
5. **Use consistent configuration** across Windows and Linux environments
6. **Set appropriate file permissions** for security

🔐 Security Considerations

```bash
```

```
# Set proper ownership and permissions
sudo chown appdynamics:appdynamics process_monitor.sh processes.json
sudo chmod 750 process_monitor.sh
sudo chmod 640 processes.json

# For log files
sudo mkdir -p /var/log/appdynamics
sudo chown appdynamics:appdynamics /var/log/appdynamics
```

Systemd Service Integration

Create a systemd service for regular monitoring:

```ini
# /etc/systemd/system/process-monitor.service
[Unit]
Description=AppDynamics Process Monitor
After=network.target

[Service]
Type=oneshot
User=appdynamics
ExecStart=/opt/appdynamics/process_monitor.sh -c /etc/appdynamics/processes.json -q -l /var/log/appdynamics/proc
```

```ini
# /etc/systemd/system/process-monitor.timer
[Unit]
Description=Run Process Monitor every minute
Requires=process-monitor.service

[Timer]
OnCalendar=*:*:00
Persistent=true

[Install]
WantedBy=timers.target
```

```bash
```

```
# Enable and start the timer
sudo systemctl enable process-monitor.timer
sudo systemctl start process-monitor.timer
```

🔄 Cross-Platform Consistency

The Linux script is designed to work with the **exact same JSON configuration files** as the Windows PowerShell version, ensuring:

- **Consistent process monitoring** across platforms
- **Shared configuration management**
- **Unified AppDynamics metrics**
- **Same output formats** and structure

This allows you to maintain one set of configuration files for both Windows and Linux environments!