

plugin system_commands

Dynatrace

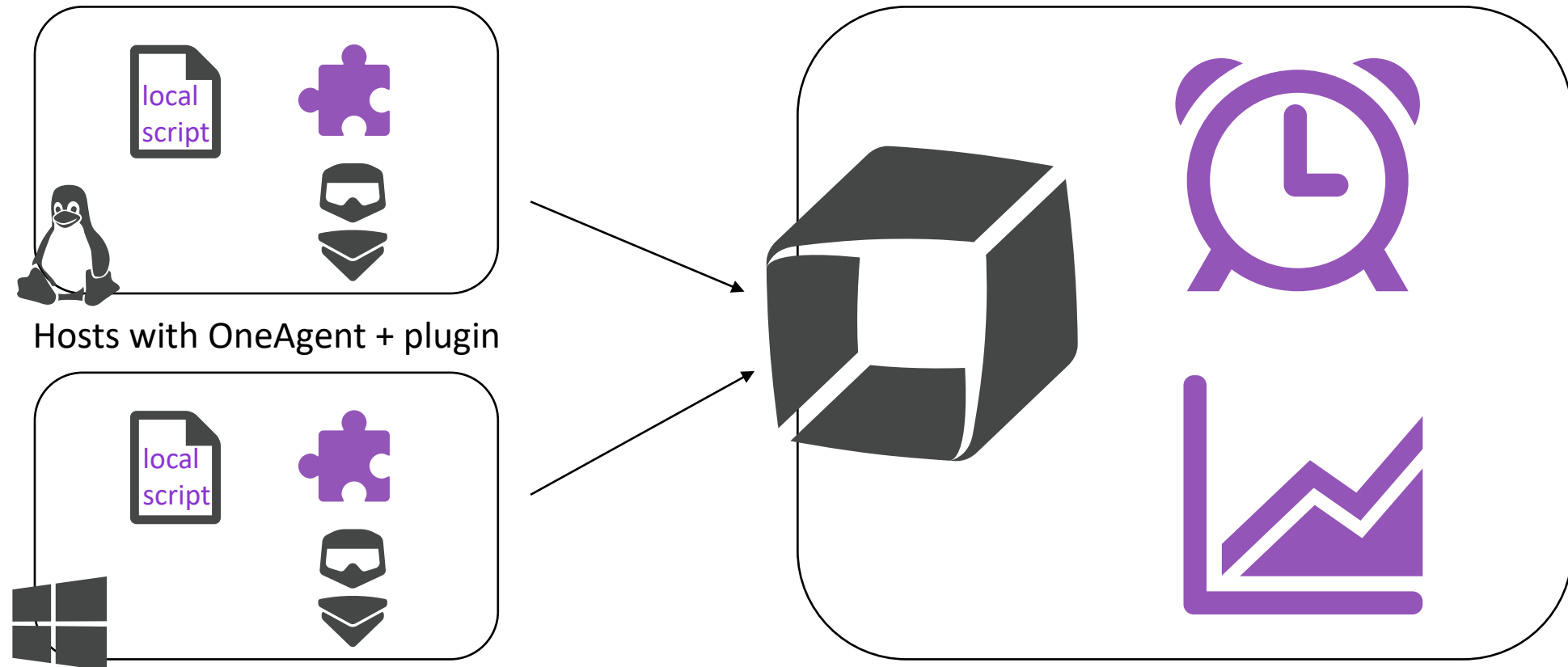


Use case



Generic extension – Plugin Command_System

- This plugin collects any metric and any status that are the result of local scripts.
- This OneAgent plugin works on the linux and the windows servers with a OneAgent.



installation



1) Upload OneAgent Extension on Dynatrace UI



custom.python.system_commands.zip

Settings > Monitoring > Monitored technologies

Monitored technologies

Look for container technology support? Find the [container monitoring](#) page here.

Dynatrace provides out-of-the-box monitoring of major technologies listed in the Supported technologies tab. Specify the technologies you want to monitor or set global monitoring settings.

To integrate any technology, detect its performance problems and get its metrics, add technology monitoring via custom extensions or Dynatrace API.

[Add technology monitoring](#)

Supported technologies

Custom extensions

Upload extensions

Directly


Upload a developed extension in ZIP format. JMX extension include JSON file only. Python extensions require Python and JSON files.

[Upload extension](#)

Via command line

To upload an extension via command line, use existing or generate new Dynatrace API token with **write configuration** access.

[Dynatrace API tokens](#)


Extension name	Technology	Extension type	Delete
 OneAgent System Commands Plugin (version 1.013)			





1) Upload OneAgent Extension on Dynatrace UI - Results

- The generic extension : OneAgent System Commands Plugin is uploaded.


 **OneAgent System Commands Plugin (version 1.013)**

Monitoring configuration

Metrics

Changelog

Monitoring configuration

 OneAgent System Commands Plugin monitoring will not start until you complete your credentials for global or host configuration.

Add global configuration

The global configuration is automatically applied to every OneAgent System Commands Plugin host in your environment. If necessary, you can override the global configuration at a single host level.

Json description of commands to execute

Enable debug logging

Click to activate debug logs

Save global configuration

Cancel

Monitoring configuration

Metrics

Display name ▼

OneAgent System Commands - Ok/Warning/Critical Metrics for alerting

OneAgent System Commands - Ok/Warning/Critical Metrics

OneAgent System Commands - Ok/Ko on message Metrics for alerting

OneAgent System Commands - Ok/Ko on message Metrics

OneAgent System Commands - Ok/Ko on exit status Metrics for alerting

OneAgent System Commands - Ok/Ko on exit status Metrics

OneAgent System Commands - Float Metrics

Do not apply global configuration. Leave empty



2) Copy OneAgent Extension on each Host



custom.python.system_commands.zip

Unzip and copy directory on :

- Linux

/opt/dynatrace/oneagent/plugin_deployment/custom.python.system_commands

```
plugin.json  
system_commands.py
```

+ service oneagent restart

- Windows

%PROGRAMFILES%\dynatrace\oneagent\plugin_deployment\custom.python.system_commands



plugin.json



system_commands.py



Restart-Service -Name "Dynatrace OneAgent"



3) Deploy your script on the host

Deploy your scripts on these specific directories

- On linux

```
/opt/dynatrace/oneagent/scripts/
```

For security reason, only the scripts on this specific directory are allowed to be run

- On windows

```
C:\\ProgramData\\dynatrace\\oneagent\\scripts
```

access to the scripts are
secured !



4) Configure the plugin on the host

Hosts Settings General

OneAgent System Commands Plugin (version 1.013) Custom extension Host On

Connected to one host:

Use host configuration

Activate OneAgent System Commands Plugin on this host

Json description of commands to execute

```
{
  "scripts": [
    {
      "metricname": "test script stime",
      "type": "status_ko_ok_on_message",
      "frequency": "1m",
      "timeout": "30",
      "shell": "",
      "command": "C:\\ProgramData\\dynatrace\\oneagent\\scripts\\scriptwindows.bat.lnk",
      "ok_pattern": "Taux_msg_M51;OK",
      "ok_message": "Taux_msg_M51 OK",
      "ko_pattern": "Taux_msg_M51;ATTENTION (*?)",
      "ko_message": "Critical error - trop de message en attente ${word1}"
    }
  ]
}
```

Enable debug logging
Click to activate debug logs

Cancel Save

- Example of script command:

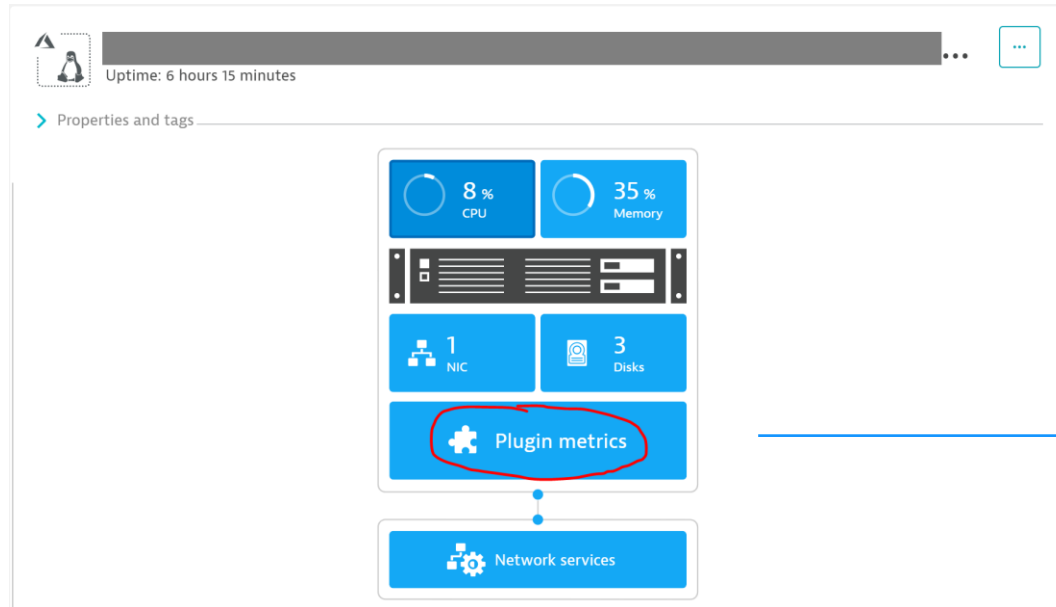
```
"command": "/opt/dynatrace/oneagent/scripts/demo_metric.ksh",
...
"command": "/opt/dynatrace/oneagent/scripts/test_threads.ksh 200",
...
"command": "/opt/dynatrace/oneagent/scripts/SystemctlServiceStatus.ksh httpd.service",
...
"command": "/opt/dynatrace/oneagent/scripts/CountFiles.ksh /opt/dynatrace/oneagent/scripts/testcountfiles",
...
"command": "C:\\ProgramData\\dynatrace\\oneagent\\scripts\\scriptwindows.bat.lnk",
...
```

Configure the extension on the local host to call your script

This configuration will be applied without any restart !!



5) Validation

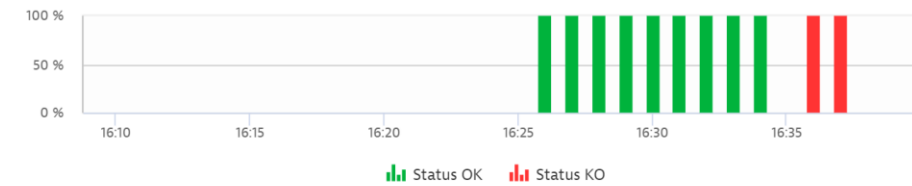


Ok/Ko Metrics on message

Further details

Ok/Ko Metrics on message

Current status: ■ KO



Status today, 16:09 - 16:39



Script



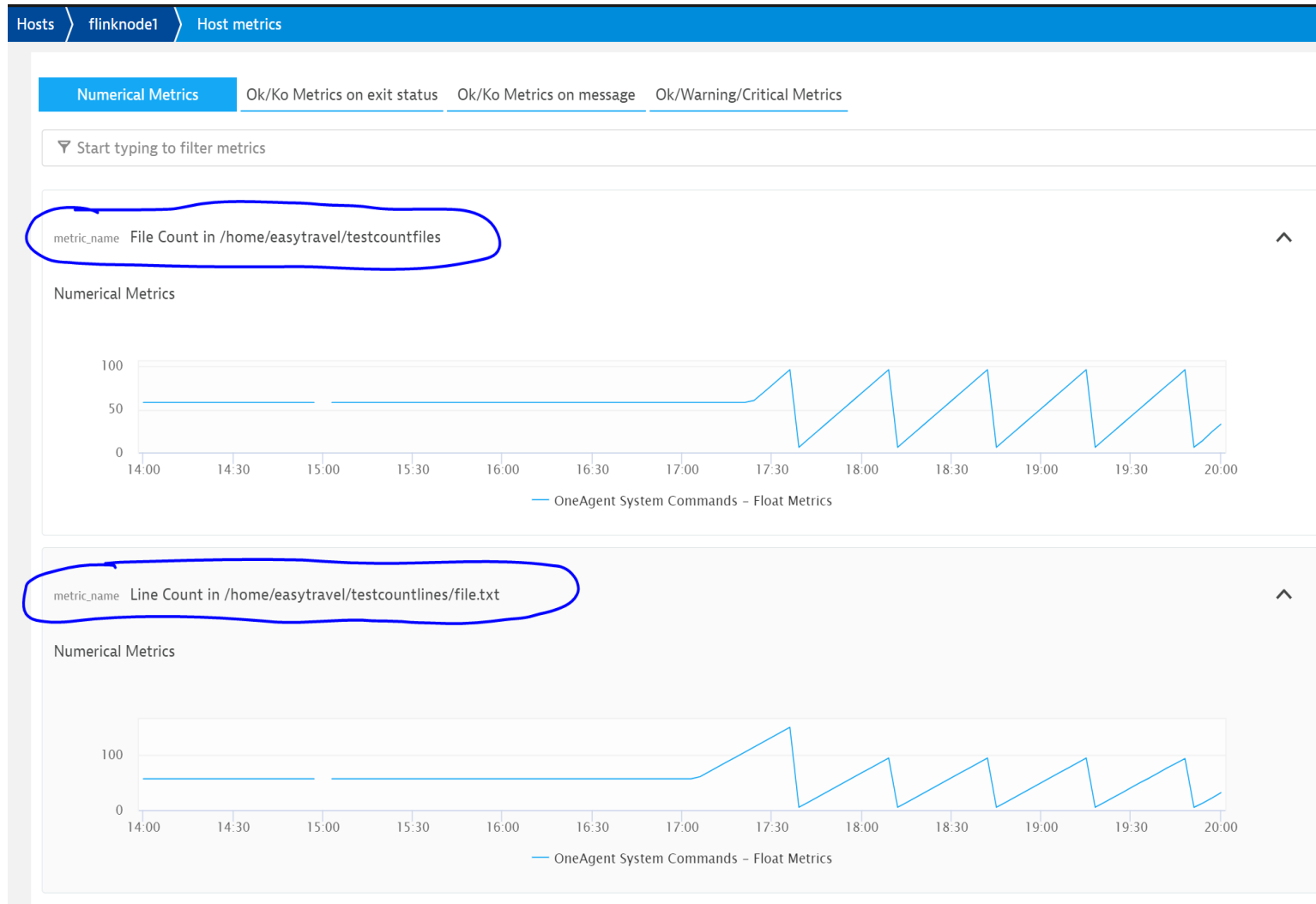
1) Type : metric (float)

- the script returns a metric :

```
{  
  "metricname" : "File Count in /home/easytravel/testcountfiles",  
  "type" : "float",  
  "frequency" : "1m",  
  "timeout" : "10",  
  "shell": "",  
  "command": "/opt/dynatrace/oneagent/scripts/CountFiles.ksh  
/opt/dynatrace/oneagent/scripts/testcountfiles"  
},
```



1) Type : metric (float)





2) Type : status_ko_ok_on_exit_status

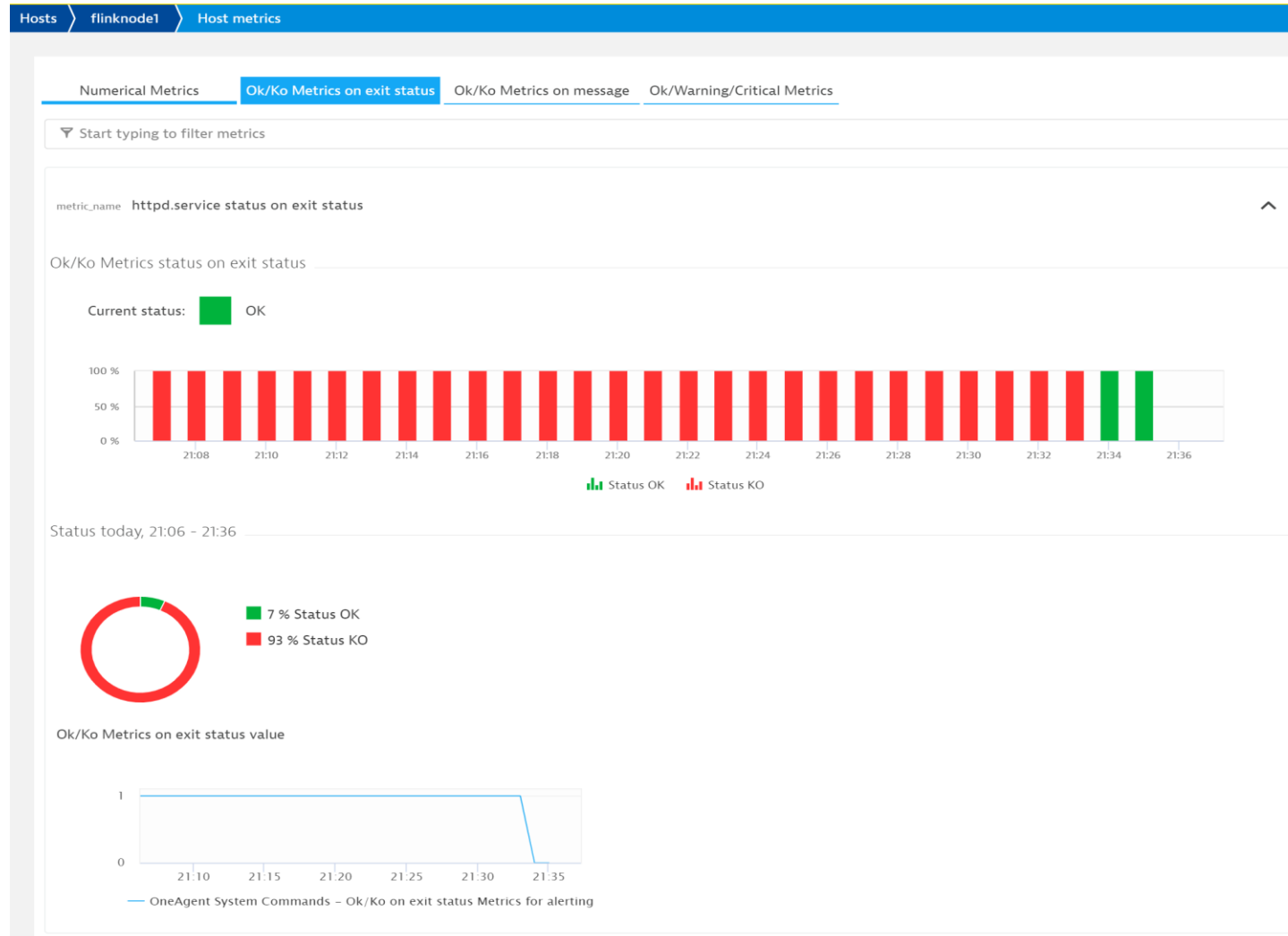
- the script returns a Ko or oK on exit status :

```
{  
  "metricname" : "httpd.service status on exit status",  
  "type" : "status_ko_ok_on_exit_status",  
  "frequency" : "5m",  
  "timeout" : "30",  
  "shell": "",  
  "command": "/opt/dynatrace/oneagent/scripts/SystemctlServiceStatus.ksh httpd.service"  
},
```

- If the exit code is 0, the command is OK. If the exit code is different from 0, this is considered as an KO execution.



2) Type : status_ko_ok_on_exit_status





3) Type : status_ko_ok_on_message

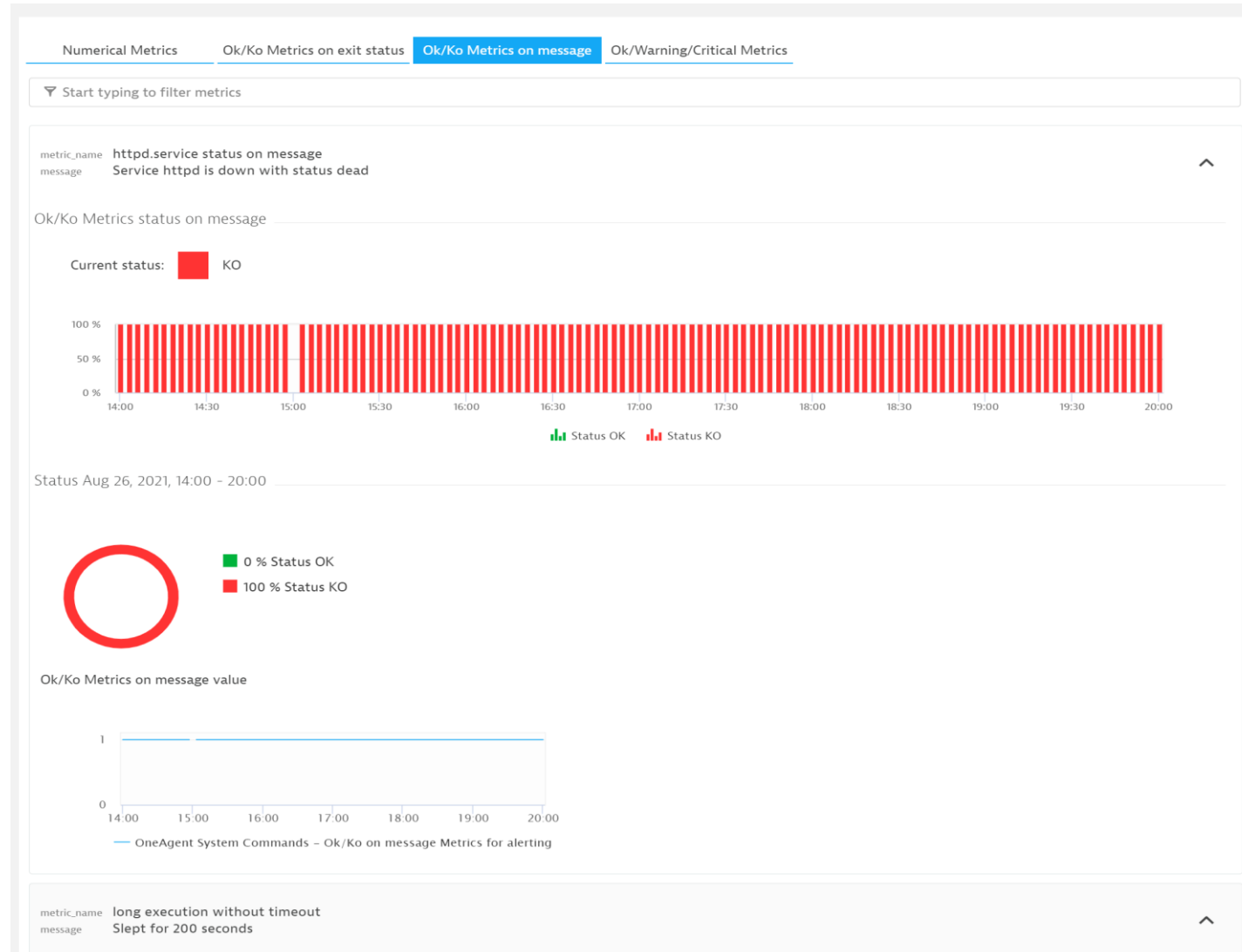
- the script returns a Ko or ok on message :

```
{  
  "metricname" : "httpd.service status on message",  
  "type" : "status_ko_ok_on_message",  
  "frequency" : "1m",  
  "timeout" : "30",  
  "shell": "",  
  "command": "/opt/dynatrace/oneagent/scripts/check_service_status.ksh httpd.service",  
  "ok_pattern" : "Active: (.*?) \\((.*)\\) since (.*?)",  
  "ok_message" : "Service httpd is ${word1} in status ${word2} since ${word3}",  
  "ko_pattern" : "Active: inactive \\((.*)\\)",  
  "ko_message" : "Service httpd is down with status ${word1}"  
},
```

- If the message matches with the **ok_pattern** => the plugin returns a status OK with the message ok_message
- If the message matches with **ko_pattern** => the plugin returns a status KO with the message ko_message.
- The regex is useful to variabilize the message.



3) Type : status_ko_ok_on_message





4) Type : status_ok_warning_critical

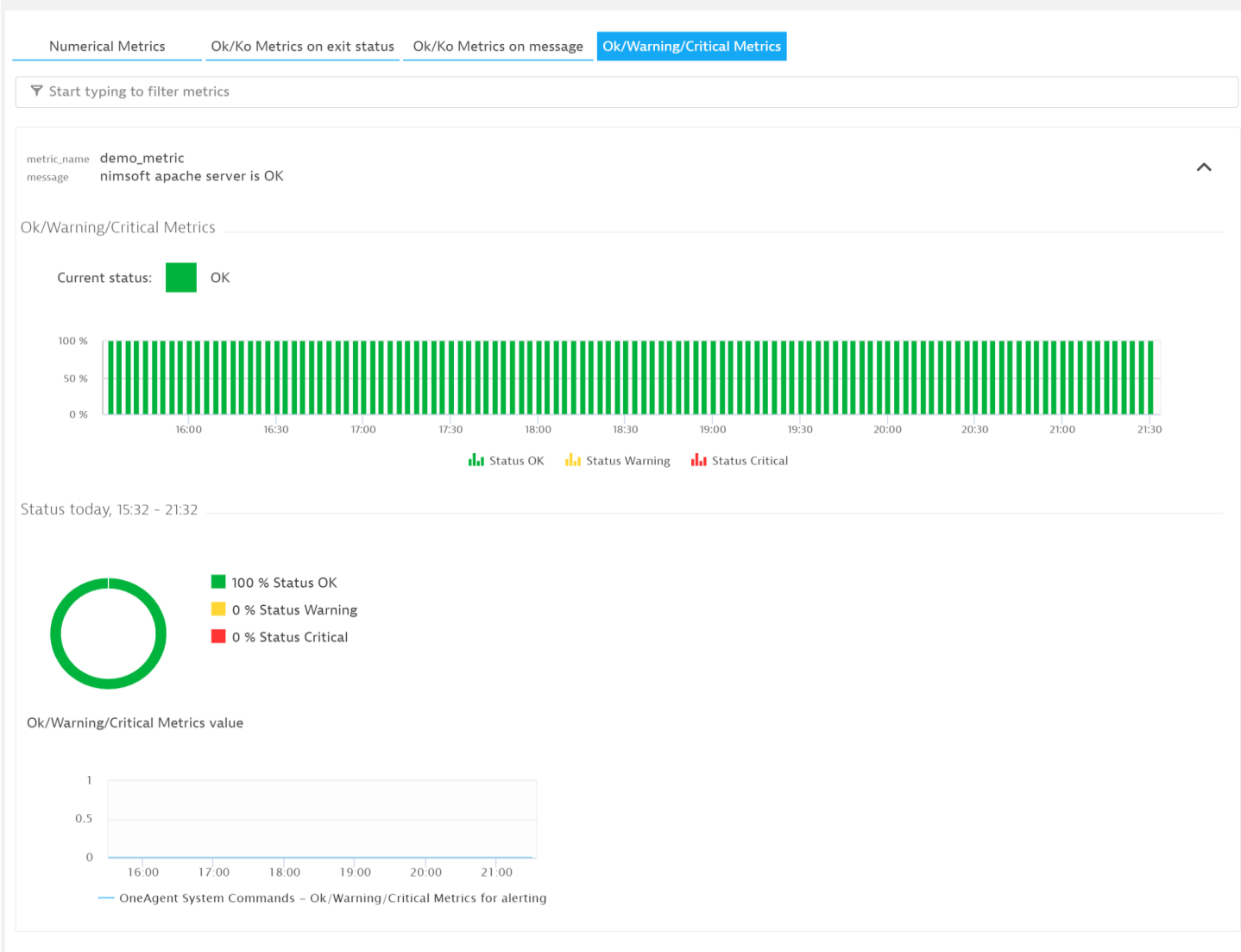
- the script returns a ok, warning or critical on message

```
{
  "metricname" : "demo_metric",
  "type" : "status_ok_warning_critical",
  "frequency" : "1m",
  "timeout" : "30",
  "shell" : "",
  "command" : "/opt/dynatrace/oneagent/scripts/demo_metric.ksh",
  "ok_pattern" : "Application: (.*?), Server: (.*?), Status: Ok",
  "ok_message" : "${word1} ${word2} server is OK",
  "warning_pattern" : "Application: (.*?), Server: (.*?), Status: Warning, ErrorCode: (.+)",
  "warning_message" : "${word1} ${word2} server is in Warning status. Error code is : ${word3}",
  "critical_pattern" : "Application: (.*?), Server: (.*?), Status: Critical, ErrorCode: (.+)",
  "critical_message" : "${word1} ${word2} server is in Critical status. Error code is : ${word3}"
},
```

- This type is similar as the ok_ko_on_message, with 3 levels of status.



4) Type : status_ok_warning_critical





5) Several scripts

- You can call several scripts with this plugin

```
{
  "scripts": [
    {
      "metricname": "File Count in /home/easytravel/testcountfiles",
      "frequency": "1m",
      "timeout": "10",
      "type": "float",
      "shell": "",
      "command": "/opt/dynatrace/oneagent/scripts/CountFiles.ksh /opt/dynatrace/oneagent/scripts/testcountfiles"
    },
    {
      "metricname": "Line Count in /home/easytravel/testcountlines/file.txt",
      "frequency": "1m",
      "timeout": "10",
      "type": "float",
      "shell": "",
      "command": "/opt/dynatrace/oneagent/scripts/CountLines.ksh /opt/dynatrace/oneagent/scripts/testcountlines/file.txt"
    },
    {
      "metricname": "httpd.service status on message",
      "type": "status_ko_ok_on_message",
      "frequency": "1m",
      "timeout": "30",
      "shell": "",
      "command": "/opt/dynatrace/oneagent/scripts/check_service_status.ksh httpd.service",
      "ok_pattern": "Active: (.*?) \\((.*?)\\) since (.*?)",
      "ok_message": "Service httpd is ${word1} in status ${word2} since ${word3}",
      "ko_pattern": "Active: inactive \\((.*?)\\)",
      "ko_message": "Service httpd is down with status ${word1}"
    },
    {
      "metricname": "httpd.service status on exit status",
      "type": "status_ko_ok_on_exit_status",
      "frequency": "5m",
      "timeout": "30",
      "shell": "",
      "command": "/opt/dynatrace/oneagent/scripts/SystemctlServiceStatus.ksh httpd.service"
    },
    {
      "metricname": "demo_metric",
      "type": "status_ok_warning_critical",
      "frequency": "1m",
      "timeout": "30",
      "shell": "",
      "command": "/opt/dynatrace/oneagent/scripts/demo_metric.ksh",
      "ok_pattern": "Application: (.*?), Server: (.*?), Status: Ok",
      "ok_message": "${word1} ${word2} server is OK",
      "warning_pattern": "Application: (.*?), Server: (.*?), Status: Warning, ErrorCode: (.)",
      "warning_message": "${word1} ${word2} server is in Warning status. Error code is : ${word3}",
      "critical_pattern": "Application: (.*?), Server: (.*?), Status: Critical, ErrorCode: (.)",
      "critical_message": "${word1} ${word2} server is in Critical status. Error code is : ${word3}"
    }
  ]
}
```

parameter



1) Frequency

- The frequency can be in minute :

```
"frequency" : "xm",
```

- The frequency can be in hours :

```
"frequency" : "xh",
```

```
"starttime" : "hh:mm",
```

- The frequency can be in days :

```
"frequency" : "d",
```

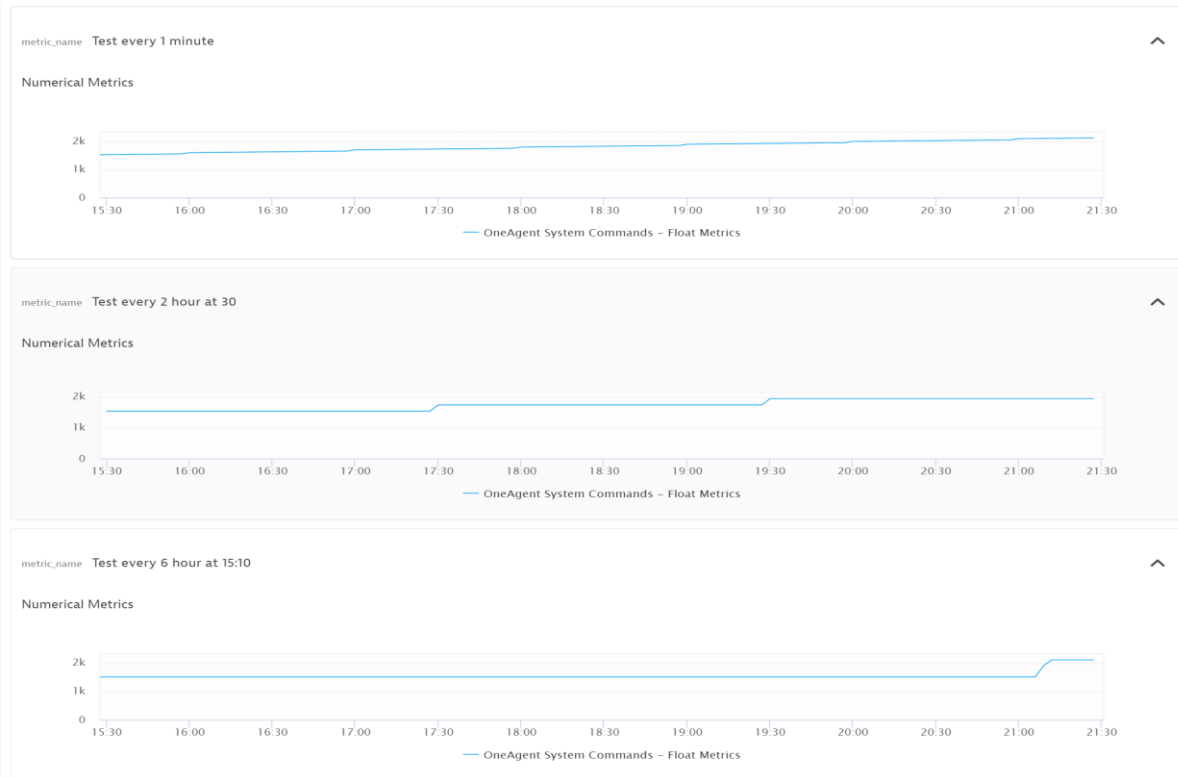
```
"starttime" : "hh:mm",
```

```
{  
  "metricname" : "Test every 6 hour at 15:10",  
  "frequency" : "6h",  
  "starttime" : "15:10",  
  "timeout" : "10",  
  "type" : "float",  
  "shell" : "",  
  "command" : "/opt/dynatrace/oneagent/scripts/TestSchedule.ksh"  
},
```



1) Frequency

- Different frequencies





2) timeout

- Timeout in seconde
- Mandatory parameter
- If a system command is to be executed every 5 minutes, the timeout must be less than 5 minutes

```
{  
  "metricname" : "long execution without timeout",  
  "type" : "status_ko_ok_on_message",  
  "frequency" : "5m",  
  "timeout" : "250",  
  "shell" : "",  
  "command" : "/opt/dynatrace/oneagent/scripts/test_threads.ksh 200",  
  "ok_pattern" : "slept for (.*) seconds",  
  "ok_message" : "Slept for ${word1} seconds",  
  "ko_pattern" : "No ko pattern",  
  "ko_message" : "No ko pattern"  
},
```




3) Shell & command

- **Shell** : This is the shell interpreter to be used to run the script. For instance : **powershell, python** etc..
- If the **command** is a linux sh, ksh ... , or a windows .bat, you don't need a shell interpreter :

```
{  
  "metricname" : "Test every 1 minute",  
  "frequency" : "1m",  
  "timeout" : "10",  
  "type" : "float",  
  "shell": "",  
  "command": "/opt/dynatrace/oneagent/scripts/TestSchedule.ksh"  
},
```

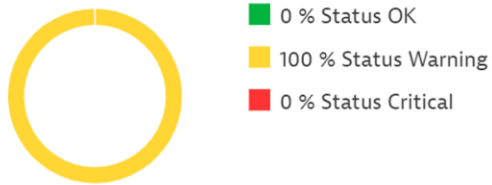
alert



Configure your alert

- For status type, there are specific metrics to configure your alert:
 - `oneagent_system_commands_ok_ko_metrics_on_message_alerting`
 - `oneagent_system_commands_ok_ko_metrics_on_exit_status_alerting`
 - `oneagent_system_commands_ok_warning_critical_metrics_alerting`

Status today, 21:04 - 21:34



Ok/Warning/Critical Metrics value



For status type, you need to use this metric for the alert

Configure your alert

Edit custom event for alerting

Define the condition and scope for detecting a metric anomaly. A custom event is raised if the selected metric exceeds the defined threshold.

Build

Code

Category

Technologies

Metric

OneAgent System Commands - Ok/Warning/Critical Metrics for alerting

Aggregation

Average

Dimensions

☐ metric_name

Split by dimension to filter

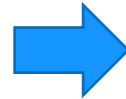
☐ message

Split by dimension to filter

Entities

Use rule-based filters to define the scope this event monitors.

Add rule-based filter



Monitoring strategy

The monitoring strategy defines what types of anomalies can be detected on a selected metric. While a static threshold is preferred for detecting breaches of hard set limits, auto-adaptive baselines are used to detect anomalies within metrics that show a regular change over time.

☒ Static threshold

☐ Auto-adaptive baseline

Static threshold settings

A static threshold monitoring strategy is preferred for alerting on hard limits within a given metric. An example is the violation of a critical memory limit. Use this section to adapt the suggested static threshold to your own needs. [Documentation](#)

Alert anomalies with a static threshold of Count (count)

Raise an alert if the metric is the threshold for minutes during any minute period.

If data is missing within the above observation period. We recommend to not alert on missing data for sparse timeseries as this leads to alert spam.

Alert preview

Basic metric query, continuously observing a total of 3 metric dimensions. The number of observed metric dimensions is not limited for basic metric queries.

1 Potential alert in the last 12 hours with these settings.

12 hours

1 day

7 days

Event description

Title

Critical Alerting on Ok/Warning/Critical Metrics

Severity

Availability

This is the message that is sent to every receiver of this event.

host : {dims:dt.entity.host.name}

metric_name : {dims:metric_name}

message : {dims:message}



Alert

Problems

Problem P-210858



flinknode1: Multiple infrastructure problems ⓘ This is a duplicate of [P-210857](#)

Problem P-210858 detected at 21:13 (open for 11 minutes).



Affected applications

–



Affected services

–



Affected infrastructure

1

1 impacted infrastructure component



flinknode1

Host

ok_pattern and ko_pattern not found for command /opt/dynatrace/oneagent/scripts/demo_metric.ksh
ok_pattern and ko_pattern not found for command /opt/dynatrace/oneagent/scripts/demo_metric.ksh : there is something wrong with the command

Critical Alerting on Ok/Warning/Critical Metrics

host : flinknode1
metric_name : demo_metric
message : nimsoft apache server is in Critical status. Error code is : 500

Comments



Add comment

No comments posted



dynatrace.com