5GCity:  
 Monitoring System Description



Summary

[1. Scope 2](#_Toc32233173)

[2. System description 3](#_Toc32233174)

[3. Metrics 4](#_Toc32233175)

[4. Alerting 5](#_Toc32233176)

[5. Front End 6](#_Toc32233177)

[6. Web Gui 7](#_Toc32233178)

[6.1. Add nodes to be monitored 7](#_Toc32233179)

[6.2. Add a job 9](#_Toc32233180)

[6.3. Add a service 11](#_Toc32233181)

[7. API REST 13](#_Toc32233182)

# Scope

This document provides a quick view of monitoring system implemented within the 5GCity framework, along with the family metrics and the GUI interface and NorthBound APIs Rest.

For the installation and configuration issues see the README.md file in the 5GCity GitHub.

# System description

The monitoring system within the 5Gcity framework is based on:

* Prometheus - an open source, metrics-based monitoring system and alerting - (version 2.3.2);
* Grafana (version 5.2.4) - that supports querying Prometheus - is used for the graphic representation of the collected data from the instrumented nodes composing the monitored system;
* Alert Manager (version 0.19.0) - an open source that integrated with the functionality of alerting of Prometheus, permits to know the firing/resolved alerts.
* Node Exporter (version 0.16.0): exposes a wide variety of hardware and kernel related metrics in order to allow Prometheus to display and/or collect specific "system" metrics;
* Apache Exporter (tested on Apache 2.2 and Apache 2.4): exposes a wide variety of apache related metrics in order to allow Prometheus to display and/or collect specific "apache" metrics;
* FrontEnd: it’s a custom java application to manage the configuration of the instrumented objects; the Linux nodes can typically be grouped into services to combine the data related to metrics and/or related to applications.

The Front End application is based on (open) jdk1.8 and Wildfly14.1.0.Final.

# Metrics

In the third issued release (February 2020), the scraping is foreseen to the Linux nodes (Ubuntu or Alpine), through the metrics available to Node Exporter, imported by Prometheus and viewed through Grafana on ad hoc dashboard.

NOTE: Linux node exporter must be installed and running on each node that has to be monitored about “system” metrics.

Moreover, if Linux node\_exporter is installed with modality “collector” facility (FULL\_INSTALLATION), it can collect any “custom” metric: each “custom” metric has prefix collector\_ and it could have a customizable help.

For installation and launch see README.md.

Moreover the scraping is foreseen also to Apache nodes, through the metrics available to Apache Exporter, imported by Prometheus and viewed through Grafana on ad hoc dashboard.

NOTE: Apache exporter must be installed and running on each node that has to be monitored about apache metrics.

For installation and launch see README.md

The following metrics will be available for “system” monitoring of nodes

* up -> it is about the status of the node i.e. the reachability of the node\_exporter process;
* node\_cpu\_\* -> some metrics related to CPU usage;
* node\_memory\_\* -> some metrics related to RAM usage;
* node\_filesystem\_\* -> some metrics related to file system usage;
* node\_disk\_\* -> some metrics related to the I/O disk usage;
* node\_network\_\* -> some metrics to determine the transmission network quality.
* collector\_\* -> any custom metric

All “custom” metrics will be available with prefix “collector\_”

The following metrics will be available for “apache” monitoring of nodes:

* apache\_up -> it is about the status of the apache client i.e. the reachability of the apache exporter;
* apache\_accesses\_total\_\* -> metrics related to total number of apache accesses;
* apache\_sent\_kilobytes\_\* -> metrics related to total number of kb sent in apache requests;
* apache\_scoreboard\_\* -> metrics related to the scoreboard statuses
* apache\_workers\_\* -> metrics related to the apache workers statuses (busy,idle)
* apache\_cpuload\_\*-> metrics related to the apache cpu load

# Alerting

By FrontEnd’s rest API, it is possible to define some “Alerting Rules” based on which Prometheus creates some alerts that Alert Manager manages.

By Alert Manager, it is possible to Get All “firing” Alerts (pull mode) or it is possible receive all “firing” or “resolved” alerts notification to a pre-defined endpoint (push mode).

# Front End

On the monitoring system, by the FrontEnd, the monitored nodes/services have to be configured in order to get the measurements using the API Rest or Web GUI.

To do this, on the FrontEnd , you can configure:

* all the nodes (Nodes) to be  monitored;
* at least one job related to the configured nodes for the “system” metrics related to dashboard type=”NODE”
* at least one service (Service) grouping the configured jobs above.

Moreover for the “apache” metrics you can configure:

* at least one job related to one or more configured nodes for the “apache” metrics related to dashboard type=”APACHE”
* at least one service (Service) grouping the configured jobs above.

By default, the job “Monitoring” is already present and it is not removable.

Moreover a service “ServiceMonitoring” is already present: it is not updatable and it not not removable.

This service is associated to the job “Monitoring” with the node “Monitoring” in order to display on the dashboards the time series related to the “system” metrics described in chapter 3 for the 5G Monitoring System itself.

# Web Gui

About the Web GUI in the following a short manual is provided.

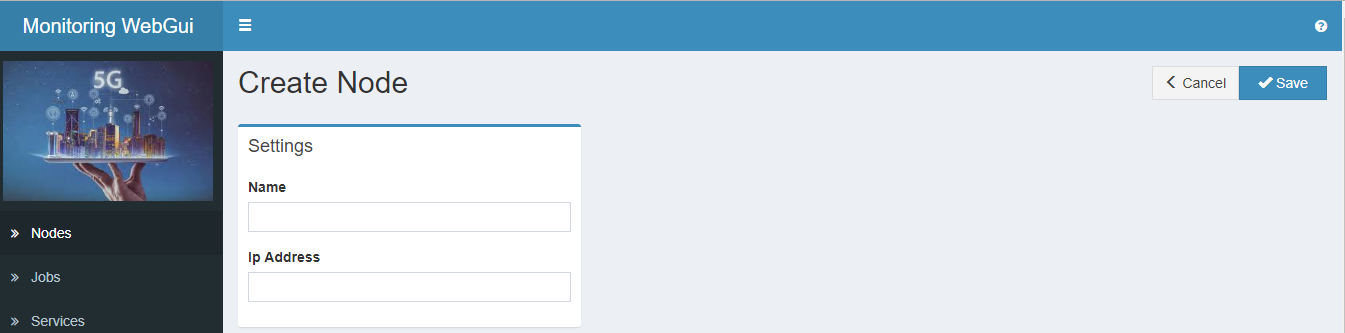
First connect by http to the ‘folder’ where the monitoring system is installed using port 8888 (default) or the one configured ad hoc if it is different from default; in the follows this port is defined <FEPort>

http://<IPFrontEnd>:<FEPort>/FrontEnd

## Add nodes to be monitored

To add nodes to be monitored by the monitoring system:

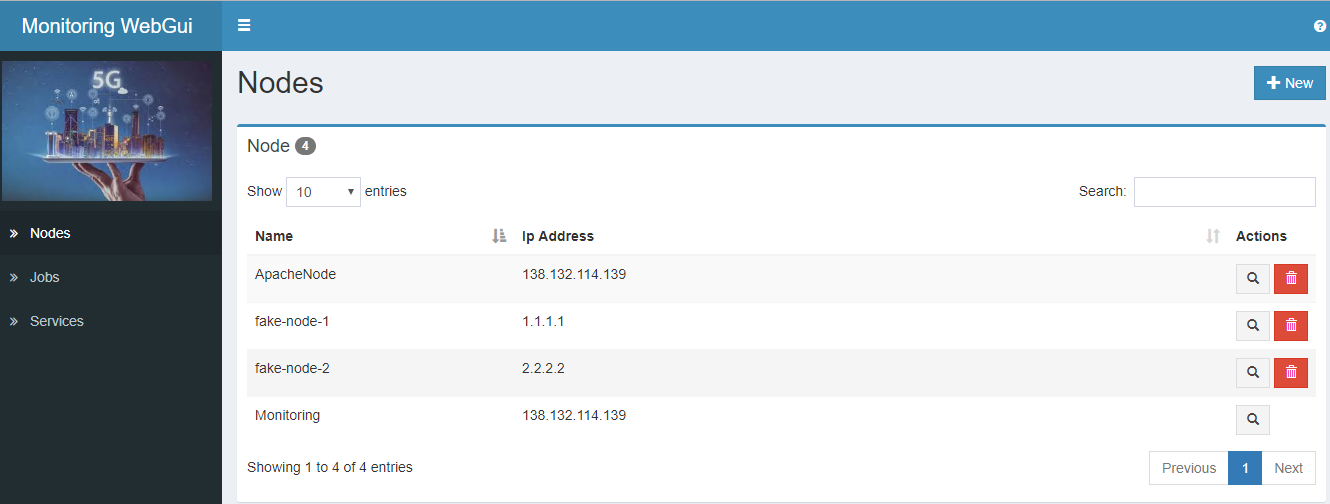
* Open Nodes; for every new node -> New
* Insert Name and address IP
* Save



Every new added node will be displayed in the node list, that is: Nodes.

For every node is possible to:

* Get a detailed visualization
* Remove the node itself



***Attention:*** the node “Monitoring” should be already present; it is not removable.

## Add a job

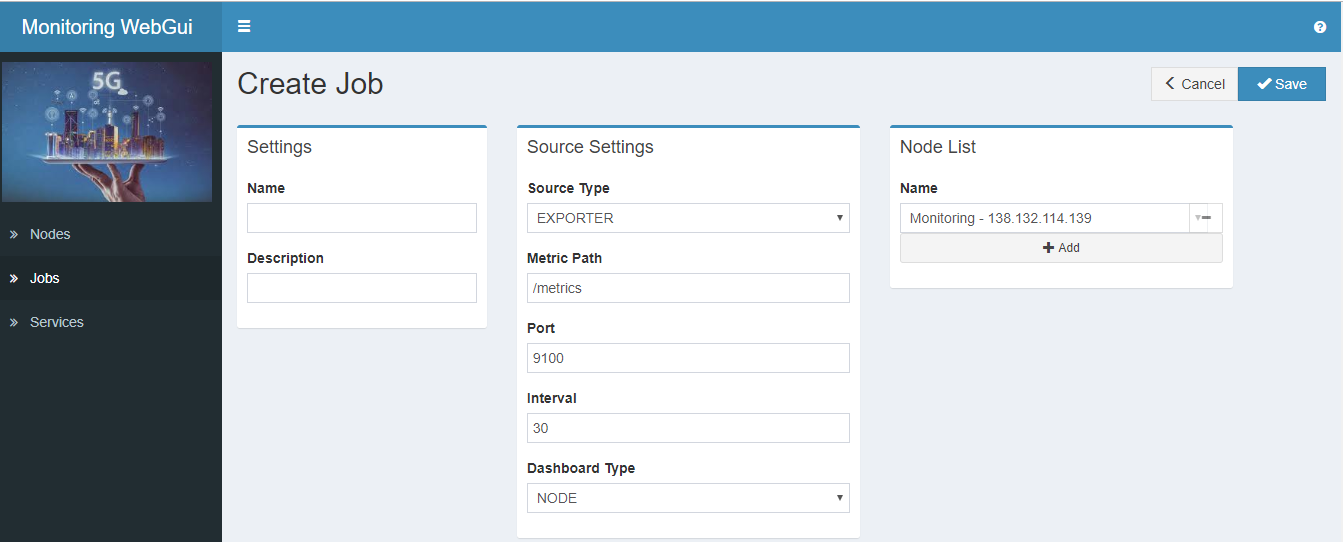
To add a job:

* Open Jobs ; for every new Job -> New
* Insert Name , Description (optional) in section Settings
* In Source Settings, you can choose of change the default values related to the “system” or “apache” metrics setting the correct value for Node exporter or Apache exporter. The pre-set defaults are for Node exporter.

***Note***: Source Type=GATEWAY or Source Type=CLIENT are for future use.

***Attention:*** for Apache metrics, Dashboard type must be set to APACHE and Port must be 9117.

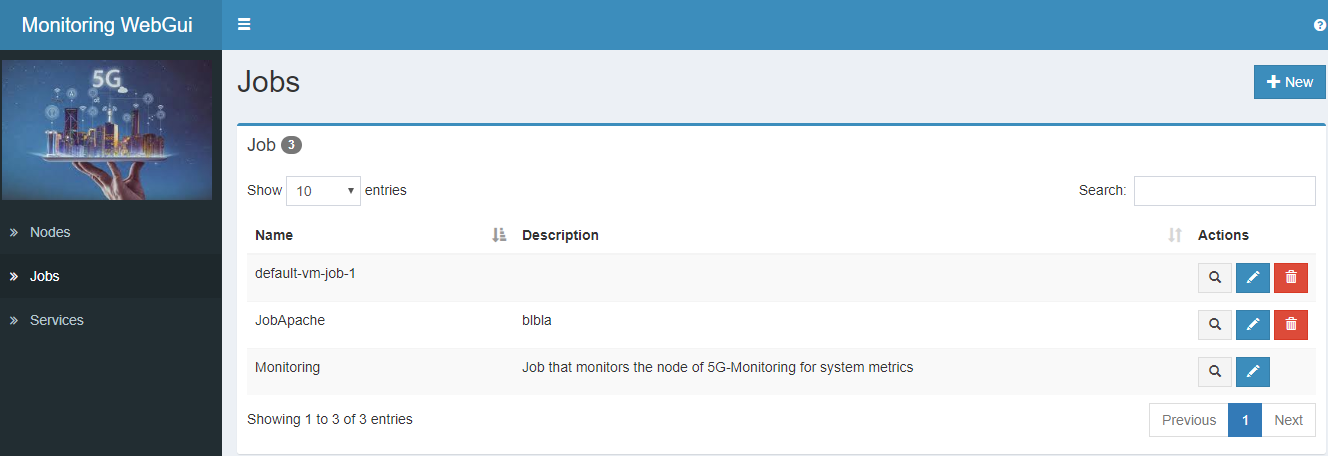
* Insert the nodes to be associated with the job (the first is already associated; use + or - signs to modify); these nodes will takes the Source Settings values of the job, so also the related metrics.
* Save



Every new added job will be displayed in the job list, that is: Jobs.

For every jobs is possible to:

* Get a detailed visualization
* Modify the job data
* Remove the job itself



***Attention:*** the job “Monitoring” should be already present; it is not modifiable and it is not removable.

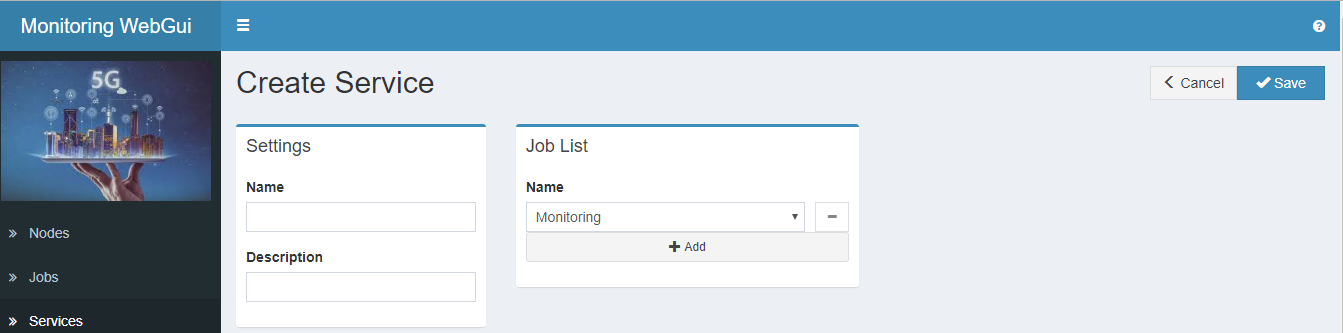
For a job the following data can be modified:

* Description
* Source Settings
* Nodes’ List

## Add a service

To add a service:

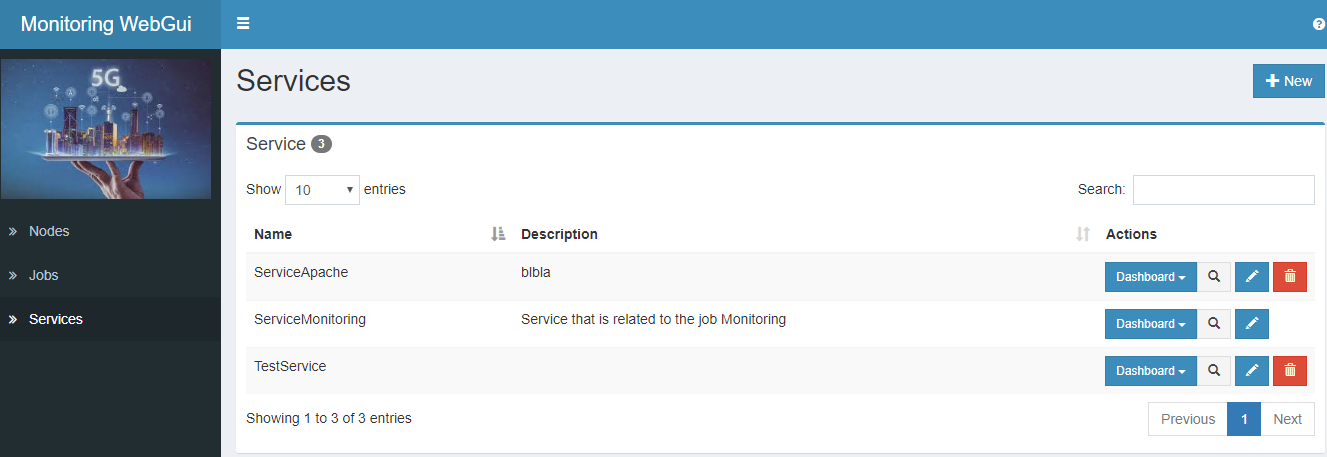
* Open the item Service ; for every new service -> New
* Insert Name , Description (optional)
* Insert the job to be associated with the service (the first is already associated; use + or - signs to modify)
* Save



Every new added service will be displayed in the service list, that is: Services.

For every service is possible to:

* Display the dashboard related to the summary’s time series for the “system” metrics for its jobs that are related to the dashboard type=”NODE” or it display the “apache” metrics for its jobs that are related to the dashboard type=”APACHE”
* Get a detailed visualization
* Modify the service data
* Remove the service itself



***Attention:*** the service “ServiceMonitoring” should be already present; it is not modifiable and it is not removable.

For a service the following data can be modified:

* Description
* Jobs’ List

# API REST

In addition to the GUI it is possible to insert / modify / remove the data of the monitoring FrontEnd objects (node, job, service) by the APIs rest.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Object** | **Url** | **Method** | **Description** | **Body** |
|  |  |  |  |  |
| Node | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/node | POST | Create | Yes |
| Node | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/node | GET | List all the objects | No |
| Node | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/node/<name> | GET | READ a single object | No |
| Node | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/node/<name> | DELETE | REMOVE a single object | No |
|  |  |  |  |  |
| Job | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/job | POST | Create | Yes |
| Job | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/job | GET | List all the objects | No |
| Job | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/job/<name> | GET | READ a single object | No |
| Job | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/job/<name> | DELETE | REMOVE a single object | No |
| Job | http://<IPFronEnd>:<FEPort>/FrontEnd/rest/fe/job | PUT | MODIFY  a single object | Yes |
|  |  |  |  |  |
| Service | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/service | POST | Create | Yes |
| Service | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/service | GET | List all the objects | No |
| Service | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/service/<name> | GET | READ a single object | No |
| Service | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/service/<name> | DELETE | REMOVE a single object | No |
| Service | http://<IPFronEnd>:<FEPort>/FrontEnd/rest/fe/service | PUT | MODIFY  a single object | Yes |

In the following some examples of Body are illustrated:

* **Create a Node**

{"name": "nodeTest", "ip": "10.10.10.10"}

* **Create/Modify a Job for metrics NODE**

{

"name":"job3",

"description":"test job system NODE metrics",

"jobSource":{

"sourceType":"EXPORTER","metricPath":"/metrics",

"port":9100, "interval":15, "dashboardType":"NODE"},

"nodes":[{"name":"nodeTest"}]

}

* **Create/Modify a Job for metrics Apache**

{

"name":"jobApache",

"description":"test job for Apache metrics",

"jobSource":{

"sourceType":"EXPORTER","metricPath":"/metrics",

"port":9117, "interval":15, "dashboardType":"APACHE"},

"nodes":[{"name":"nodeTest"}]

}

* **Create/Modify a Service with job for Apache exporter**

{"name": "serviceApache","jobs":[{"name": "jobApache" }]}

Moreover, only via rest API - by FrontEnd- it is possible to create and delete alert rules to permit Alert Manager to manage alerts.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Object** | **Url** | **Method** | **Description** | **Body** |
| Alert Rule | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/alertrule | POST | Create | Yes |
| Alert Rule | http://<IPFrontEnd>:<FEPort>/FrontEnd/rest/fe/alertrule/<name> | DELETE | REMOVE a single alert rule identified by <name> | No |

In the following some examples of Body for Alert Rules are illustrated:

* **Create a global Rule defining a CPU usage threshold**

|  |
| --- |
| {      "name": "HighCPULoad",      "expression": "100 - ((rate(node\_cpu\_seconds\_total{mode=\\\"idle\\\"}[1m])) \* 100) > 70",      "duration": "1m",      "severity": "critical",      "summary": "High CPU load (service {{ $\"labels\".service }} , node {{ $\"labels\".instance }})",      "description": "CPU load is > 70%\n  VALUE = {{ $\"value\" }}\n"  } |

* **Create a rule about a simple custom metric collector\_running\_docker\_containers**

|  |
| --- |
| {      "name": "testRule1",      "duration": "1m",      "severity": "major",      "expression": "collector\_running\_docker\_containers < 15",      "summary": "testRule1 on  (instance {{ $\"labels\".instance }})",      "description": "descr VALUE = {{ $\"value\" }} LABELS: {{ $\"labels\" }}"}  } |

* **Create a rule about a system metric applied to a specific service configured on Monitoring system**

|  |
| --- |
| {   "name": "testCPUSipP",      "duration": "1m",      "severity": "major",      "expression": "node\_cpu\_seconds\_total{service=\\\"serviceSipP\\\",mode=\\\"user\\\"} > 12000",      "summary": "testCPUSipP (instance {{ $\"labels\".instance }})",      "description": "descr VALUE = {{ $\"value\" }} LABELS: {{ $\"labels\" }}"  } |

All generated alerts are managed by Alert Manager: it is possible to Get Alerts List or to receive alerts notifications

**Get Alert List (pull mode)**

Dashboard (or other consumers) can view the list of open “firing” alerts through REST API:

[http://<5g-monitoring>:9093/api/v1/alerts](http://138.132.108.30:9093/api/v1/alerts)

Sample response:

{"status":"success","data":[{"labels":{"alertname":"HighMemoryUsage","instance":"Monitoring","job":"Monitoring","org":"SYSMONITORING","service":"ServiceMonitoring","severity":"critical","type":"NODE"},"annotations":{"description":"Memory Usage is over 5% : current value 20.388611878429263%","summary":"High Memory Usage (service ServiceMonitoring , node Monitoring)"},"startsAt":"2019-08-30T14:13:45.520693759Z","endsAt":"2019-08-30T14:25:30.530777099Z","generatorURL":"http://prometheus:9090/graph?g0.expr=%28%28node\_memory\_MemTotal\_bytes+-+node\_memory\_MemFree\_bytes%29+%2F+%28node\_memory\_MemTotal\_bytes%29%29+%2A+100+%3E+5\u0026g0.tab=1","status":{"state":"active","silencedBy":[],"inhibitedBy":[]},"receivers":["dashboard"],"fingerprint":"150f23f63fc31d03"},{"labels":{"alertname":"HighMemoryUsage","instance":"J\_Backup","job":"J\_Backup\_Node","org":"SYSMONITORING","service":"ServiceJenkins","severity":"critical","type":"NODE"},"annotations":{"description":"Memory Usage is over 5% : current value 97.26593574208314%","summary":"High Memory Usage (service ServiceJenkins , node J\_Backup)"},"startsAt":"2019-08-30T14:14:15.520693759Z","endsAt":"2019-08-30T14:25:30.530777099Z","generatorURL":"http://prometheus:9090/graph?g0.expr=%28%28node\_memory\_MemTotal\_bytes+-+node\_memory\_MemFree\_bytes%29+%2F+%28node\_memory\_MemTotal\_bytes%29%29+%2A+100+%3E+5\u0026g0.tab=1","status":{"state":"active","silencedBy":[],"inhibitedBy":[]},"receivers":["dashboard"],"fingerprint":"a09c9c7d89277a2e"},{"labels":{"alertname":"HighCpuLoad","cpu":"1","instance":"Monitoring","job":"Monitoring","mode":"idle","org":"SYSMONITORING","service":"ServiceMonitoring","severity":"critical","type":"NODE"},"annotations":{"description":"CPU load is \u003e 80%\n VALUE = 1.355555555653865\n LABELS: map[instance:Monitoring job:Monitoring mode:idle org:SYSMONITORING service:ServiceMonitoring type:NODE cpu:1]","summary":"High CPU load (service ServiceMonitoring , node Monitoring)"},"startsAt":"2019-08-30T14:20:15.520693759Z","endsAt":"2019-08-30T14:25:30.527470373Z","generatorURL":"http://prometheus:9090/graph?g0.expr=100+-+%28%28rate%28node\_cpu\_seconds\_total%7Bmode%3D%22idle%22%7D%5B1m%5D%29%29+%2A+100%29+%3E+1\u0026g0.tab=1","status":{"state":"active","silencedBy":[],"inhibitedBy":[]},"receivers":["dashboard"],"fingerprint":"fb3ace00522413f5"}]}

**Alerts notification (push mode)**

Dashboard (or other consumers) can be notified of incoming “firing” and “resolved” alerts.

To be notified they must expose an http endpoint (e.g.: <http://dashboard/notification-endpoint>) to receive POST notification from module Alert Manager of Monitoring System

**Example notification sent from Monitoring to Dashboard**: POST <http://dashboard/notification-endpoint>

{  
"receiver": "dashboard",  
"status": "firing",  
"alerts": [  
{  
"status": "firing",  
"labels": {  
"alertname": "InstanceDown",  
"instance": "Monitoring",  
"job": "Monitoring",  
"org": "SYSMONITORING",  
"service": "ServiceMonitoring",  
"severity": "critical",  
"type": "NODE"  
},  
"annotations": {  
"description": "Monitoring of job Monitoring has been down for more than 5 minutes.",  
"summary": "Instance Monitoring down"  
},  
"startsAt": "2019-08-28T14:44:00.520693759Z",  
"endsAt": "0001-01-01T00:00:00Z",  
"generatorURL": "<http://prometheus:9090/graph?g0.expr=up+%3D%3D+0\u0026g0.tab=1>"  
}  
],  
"groupLabels": {  
"alertname": "InstanceDown"  
},  
"commonLabels": {  
"alertname": "InstanceDown",  
"instance": "Monitoring",  
"job": "Monitoring",  
"org": "SYSMONITORING",  
"service": "ServiceMonitoring",  
"severity": "critical",  
"type": "NODE"  
},  
"commonAnnotations": {  
"description": "Monitoring of job Monitoring has been down for more than 5 minutes.",  
"summary": "Instance Monitoring down"  
},  
"externalURL": "[http://alertmanager:9093](http://alertmanager:9093/)",  
"version": "4",  
"groupKey": "{}:{alertname=\"InstanceDown\"}"  
}